

SANDIA REPORT

SAND2006-0074
Unlimited Release
Printed January 2006

Algorithm and Simulation Development in Support of Response Strategies for Contamination Events in Air and Water Systems

Editor: B. van Bloemen Waanders

Sandia Contributors: B. Bader, R. Bartlett, J. Berry, R. Bilisoly, P. Boggs, R. Carr, S. Collis, A. Cooper, A. Draganescu, G. Hammond, W. Hart, J. Hill, S. McKenna, P. Lin, K. Long, S. Margolis, C. Peyton, C. Phillips, M. Sala, A. Salinger, J. Shadid, C. Silva, R. Tuminaro, V. Tidwell, B. van Bloemen Waanders, J. Watson, L. Yarrington

External Contributors: V. Akcelik (CMU), L. Biegler (CMU), G. Biros (UP), S. Buchberger (UC), S. Bujanda (UNM), O. Ghattas (UT), H. Greenberg (UCD), P. Howard (WPI), R. Janke (EPA), G. Konjevod (ASU), C. Laird (CMU), E. Lauer (UNM), Z. Li (UC), H. Lin (UCB), T. Morrison (UCD), R. Murray (EPA), J. Uber (UC), K. Willcox (MIT)

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.doe.gov/bridge>

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: <http://www.ntis.gov/ordering.htm>



Algorithm and Simulation Development in Support of Response Strategies for Contamination Events in Air and Water Systems

Editor: B. van Bloemen Waanders

Sandia Contributors: B. Bader, R. Bartlett, J. Berry, R. Bilisoly, P. Boggs,
R. Carr, S. Collis, A. Cooper, A. Draganescu, G. Hammond, W. Hart,
J. Hill, S. McKenna, P. Lin, K. Long, S. Margolis, C. Peyton, C. Phillips,
M. Sala, A. Salinger, J. Shadid, C. Silva, R. Tuminaro, V. Tidwell,
B. van Bloemen Waanders, J. Watson, L. Yarrington

External Contributors: V. Akcelik (CMU), L. Biegler (CMU), G. Biros (UP),
S. Buchberger (UC), S. Bujanda (UNM), O. Ghattas (UT),
H. Greenberg (UC), P. Howard (WPI), R. Janke (EPA), G. Konjevod (ASU),
C. Laird (CMU), E. Lauer (UNM), Z. Li (UC), H. Lin (UCB), T. Morrison (UCD),
R. Murray (EPA), J. Uber (UC), K. Willcox (MIT)

Acknowledgment

This project was primarily funded by the LDRD office at Sandia National Laboratories. During the course of this three year project, the EPA, DHS, and SNL's CSRF funded other but related algorithmic extensions, which are included in this report. The EPA funding continues to support additional algorithmic development for the security of water distribution systems. DHS funded additional algorithmic research for security of internal facilities, both forward modeling and source inversion for transient simulations. CSRF funded research of the air security portion for the first year of this LDRD project.

We wish to thank Dan Quintana and members of the engineering department from the Tucson Water utility company for very useful meetings and technical information exchanges, in addition to access to datasets. Most of our algorithms were tested against these datasets which are representative of real and production quality networks.

Finally, we would like to acknowledge Ray Finley's enthusiastic support for our work.

Contents

Executive Summary	xxvi
1 Introduction	1
1.1 Background	1
1.2 Project Goals	3
1.2.1 Source Inversion	3
1.2.2 Sensor Placement	4
1.2.3 Forward Modeling	5
1.2.4 Uncertainties	6
1.2.5 Risk Assessment	6
1.2.6 Real Time Performance	7
1.3 Future Work	7
1.4 Outline of the Report	9
2 Contamination Source Determination for Water Networks	10
2.1 Background	10
2.1.1 Problem Formulation	11
2.1.2 Solution Techniques	13
2.2 Origin Tracking Algorithm	15

2.3	Discretized Nonlinear Program	18
2.4	Numerical Results	20
2.4.1	Example 1. Symmetric Grid Network.....	20
2.4.2	Example 2. Real Municipal Network Model	24
2.5	Conclusions and Future Work	28
3	Real-time, Large Scale Optimization of Water Network Systems using a Subdomain Approach	30
3.1	Background	30
3.1.1	Contamination Source Determination	31
3.2	Dynamic Optimization Formulation	32
3.2.1	Origin Tracking Algorithm	34
3.2.2	Network Subdomain Approach	38
3.3	Numerical Results	40
3.3.1	Results: Fixed Discretization, Variable Problem Size.....	41
3.3.2	Results: Fixed Problem Size, Variable Discretization.....	42
3.4	Conclusions and Future Work	44
4	A Mixed Integer Approach for Obtaining Unique Solutions in Source Inversion of Drinking Water Networks	47
4.1	Background	47
4.2	Mixed Integer Formulation	51
4.3	Results	54
4.4	Conclusions and Future Work	57
5	Sensor Placement in Municipal Water Networks with Dynamic Integer Programming Models	59

5.1	Background	59
5.1.1	Static Formulations	60
5.1.2	Dynamic Formulations	61
5.2	A Dynamic MIP Model	62
5.3	Empirical Results	64
5.3.1	Solution Methods	64
5.3.2	Methodology and Test Problems	65
5.3.3	Solution via Mixed-Integer Programming	66
5.3.4	Solution via the <i>RW</i> Heuristic	66
5.4	Conclusions	67
6	A Multiple-Objective Analysis of Sensor Placement Optimization in Water Networks	69
6.1	Background	69
6.2	Problem Description and Mixed-Integer Formulation	70
6.3	Experimental Results	73
6.3.1	The Test Networks	73
6.3.2	Node versus Edge Sensor Placements	73
6.3.3	Characteristics of Individual Performance Objectives	74
6.3.4	The Impact of Optimization on Complementary Objectives	75
6.3.5	Characterizing the Trade-offs Between Competing Objectives	77
6.4	Conclusions	78
7	Robust Optimization of Contaminant Sensor Placement for Community Water Systems	79
7.1	Background	79
7.2	Motivation for Robust MILP Models	80

7.3	Linearly Weighted Uncertainty	82
7.4	Unweighted Uncertainty	83
7.5	Bilinear Weighted Uncertainty	85
7.5.1	Complexity	86
7.5.2	Alternating Ascent	91
7.5.3	Linear Programming Relaxation	93
7.6	Preliminary Computational Study	93
7.6.1	Linearly Weighted Uncertainties	94
7.6.2	Bilinearly Weighted Uncertainty	95
7.7	Discussion	97
8	Examining the Effects of Variability in Short Time Scale Demands on Solute Transport	99
8.1	Background	99
8.2	PRP Generator	100
8.3	Analytical Expressions for Scaling Demand Variability	101
8.4	PRP Model Based Approach	102
8.5	Perturbation Theory Approach	102
8.6	Modeling Temporally Variable Demand	104
8.6.1	Single-Day	104
8.7	Multiple Day	105
8.8	Discussion	106
8.9	Conclusions	107
9	Modeling Solute Transport in Distribution Networks with Variable Demand and Time Step Sizes	110

9.1	Background	110
9.2	Water Demand Generator.....	111
9.3	Example Network	111
9.4	Reynolds Number Distribution	113
9.5	Solute Transport Timing: Tank Source	115
9.6	Solute Center of Mass: Node Sources	116
9.7	Solute Transport Timing: Node Sources	117
9.8	Solute Dispersion	119
9.9	Conclusions	119
10	Source Location Inversion and the Effect of Stochastically Varying Demand	120
10.1	Background	120
10.2	Simulation Approach	121
10.3	Demand Simulation	121
10.4	Non-Linear Optimization for Source Location	122
10.5	Example Problem.....	123
10.6	Results and Discussion	124
10.7	Conclusions and Future Work	127
11	Threat Assessment of Water Supply Systems Using Markov Latent Effects Modeling	128
11.1	Background	128
11.2	Methods	129
11.3	MLE Background	129
11.4	Decomposition Factors	130
11.5	Decision Element Structure	131
11.6	Data Aggregation	131

11.7 Model Implementation	132
11.8 Results	133
11.9 MLE Model	133
11.10 MLE Model Application	134
11.11 Discussion	137
11.12 Conclusions	137
12 Vulnerability Assessment and the Basis of Analysis	144
12.1 Background	144
12.2 Methods	145
12.3 Threat Assessment	145
12.4 Consequence Analysis	147
12.5 Results	147
12.6 Threat Assessment	148
12.7 Consequence Analysis	149
12.8 Risk Assessment	149
12.9 Conclusions	150
13 A Comparison of Navier Stokes and Network Models To Predict Chemical Transport In Municipal Water Distribution Systems	152
13.1 Background	152
13.2 Mathematical Formulation	153
13.3 Numerical Implementation	154
13.4 2D Numerical Results	156
13.5 Experimental Verification	158
13.6 Ongoing Research	159

13.6.1	3D Turbulence Simulation	159
13.6.2	First-Order Correction Strategies	160
13.7	Conclusions	161
14	Performance of Fully-Coupled Algebraic Multilevel Domain Decomposition Preconditioners for Incompressible Flow and Transport	162
14.1	Introduction	162
14.2	Governing Equations	163
14.3	Preconditioned Newton-Krylov Method	164
14.4	Multilevel Preconditioners	165
14.5	Results and Discussion	167
14.5.1	Comparison between the two-level Schwarz preconditioner with geometric coarse operator and algebraic coarse operator	167
14.5.2	Algorithmic scaling studies for three-level Schwarz preconditioner with algebraic coarse operators for the 3D thermal convection problem	169
14.5.3	Algorithmic scaling studies for three-level Schwarz preconditioner with algebraic coarse operators for Navier-Stokes flow in a 3D building geometry	172
14.5.4	Comparisons between one-level, two-level, and three-level preconditioners for fluid flow in a 3D building geometry	176
14.5.5	Algorithmic scaling studies for three-level Schwarz preconditioner with algebraic coarse operators for the solution of the convection-diffusion equation	178
14.6	Conclusions	179
15	Rapid Source-Inversion for Chemical/Biological Attacks, Part 1: The Steady-State Case	181
15.1	Background	181
15.2	Software Environment	183
15.3	Flow-Field Computation	184
15.3.1	The Turbulence Model	184

15.3.2	Implementation of the Flow-Field Model in Sundance	186
15.3.3	The Eikonal Equation and its Regularization	190
15.3.4	Pressure Stabilization	190
15.4	The Optimization Problem	191
15.4.1	The Toxin Transport Model	191
15.4.2	The Source-Inversion Optimization Problem	193
15.5	Numerical Results	196
15.5.1	Sample 2-Dimensional Problem Geometry and Flow Field	196
15.5.2	Source Inversion Tests	198
15.5.3	Coarse Meshes	199
15.6	Discussion and Conclusions	201
16	Offline/Online QP Strategies for Real Time Inversion	208
16.1	Background	208
16.1.1	Motivating Example Application: Source Inversion for Convection-Diffusion in an Airport Terminal	209
16.1.2	Statement of the Problem: General QP Formulation	211
16.2	Reduced QP problem	212
16.3	Decomposition of Reduced QP into Offline and Online Subproblems	212
16.3.1	Offline Subproblem	213
16.3.1.1	Computing Compressed Sensitivities using the Forward Approach	214
16.3.1.2	Computing Compressed Sensitivities using the Adjoint Approach	214
16.3.2	Online Subproblem	214
16.4	Linearly Constrained Transient Inversion Problems	216
16.4.1	Generalized Interpretation of \hat{D} , \hat{Q}_y and \hat{y}	219

16.4.2	Piecewise-Constant Temporal Discretization of the Source $u(t)$ and the Specialized Computation of \hat{D}	219
16.5	Source Inversion for Convection-Diffusion in an Airport Terminal : Discretization and Results	222
16.5.1	Discretization and General Numerical Approach	222
16.5.2	Transient Inversion Results	223
16.5.2.1	General Inversion Behavior and Quality	223
16.5.2.2	Impact of Scaling and Bounds on Inversion Quality	228
16.5.2.3	Online Computational Expense and Inexact QP Solves	229
16.5.2.4	Conditioning and Regularization	234
16.6	Summary and Conclusions	235
16.7	Recommendations and Future Work	236
17	Real-Time Identification of Airborne Contaminants	238
17.1	Background	238
17.2	Formulation and optimality conditions	241
17.3	Multigrid Preconditioner	244
17.4	Implementation and Numerical Results	246
17.4.1	Source inversion in the Greater Los Angeles Basin	247
17.4.2	Scalability of the Multigrid preconditioner	250
17.5	Conclusions	251
18	An Optimization Framework for Goal-Oriented, Model-Based Reduction of Large-Scale Systems	255
18.1	Background	255
18.2	Dynamical System Framework	257
18.2.1	Parametric input variations	257

18.2.2	Proper orthogonal decomposition	258
18.3	Optimized Reduced-Order Basis	259
18.3.1	Constrained optimization formulation for projection basis	259
18.3.2	Optimality conditions and the reduced gradient	260
18.3.3	Basis computation	262
18.4	Model Problem and Results	263
18.4.1	Model problem description	263
18.4.2	Optimized basis performance	265
18.4.3	Comparison with POD	265
18.5	Conclusions	268

Appendix

References	292
----------------------	-----

List of Figures

2.1	Link Boundary Designation. $I_i(t)$ indicates the inlet of the link, based on the current flow direction, while $O_i(t)$ indicates the outlet of the link. The index $k_i(t)$ always refers to the node connected at the inlet.	12
2.2	Origin Tracking Algorithm. This figure illustrates the flow of a single volume element through a pipe. Tracking these element allows us to determine relationships for the boundary concentrations at each point in time.	15
2.3	Grid Network Example A small symmetric grid network with sensors installed at every second node, indicated by the shading.	21
2.4	Grid Network Solution 1. Solution for an injection at node 13 using a 4 hour time horizon.	22
2.5	Grid Network Solution 2. Solution for an injection at node 13 with an 8 hour time horizon. Note that nodes 9 and 17 are now excluded as possible injection locations.	23
2.6	Municipal Water Network Injection Locations. Four simulated injection locations, A through D, are shown in the diagram of the network model.	25
2.7	Rank of Optimization Solution for Simulated Injection Node. This figure illustrates the effectiveness of the formulation in determining the correct injection node for simulated injections A through D. The rank of the injection node is shown with the shading to the right of each contour plot. A low ranking indicates that the formulation has been effective at identifying the injection location.	27
3.1	Link Boundary Designation. $I_i(t)$ indicates the inlet of the link, based on the current flow direction, while $O_i(t)$ indicates the outlet of the link. The index $k_i(t)$ always refers to the node connected at the inlet.	34
3.2	Origin Tracking Algorithm. This figure illustrates the flow of a single volume element through a pipe. Tracking these element allows us to determine relationships for the boundary concentrations at each point in time.	37

3.3	Municipal Water Network. This figure shows the entire water network. The circled area is shown in more detail in Figure 3.3.	41
3.4	Municipal Water Network. This figure shows increased detail near the injection location, where shaded nodes indicate sensor locations.	42
3.5	Solution with 900 Node Subdomain. This figure shows the solution of the inversion problem for a 5 minute time discretization and a subdomain size of 900 nodes. Although the solution is non-unique, showing 4 possible injection locations, the significant profiles, 2921 and 2879 are the actual injection location and its neighbor. The solution has also correctly identified the injection time.	43
3.6	Solution with 100 Node Subdomain. This figure shows the solution of the inversion problem for a 5 minute time discretization and a subdomain size of 100 nodes. The solution quality has improved over the larger 900 node subdomain.	43
3.7	Optimization Solutions for Fixed Problem Sizes. This figure shows the optimization solution for different subdomain sizes, where the size of the nonlinear problem was kept constant using the parameters specified in Table (3.2)	45
4.1	Small symmetric grid network with sensors installed at every second node, indicated by the shading.	49
4.2	Grid Network Example Solution. Solution of problem (4.1.1-4.1.4) on the grid network with an injection from node 13 at $t = 0.5$ hours.	50
4.3	Network model showing the four nodes of interest, A, B, C, and D	55
4.4	Recorded Solutions for Test 1. Results of Algorithm 1 for the contamination scenario with a single injection location.	56
4.5	Recorded Solutions for Test 2. Results of Algorithm 1 for the contamination scenario with two injection locations.	56
7.1	Example graph with induced constructed graph.	87
7.2	Partition of M upon selecting rows, columns, and elements.	88
7.3	An alternating ascent algorithm for the bilinear program (7.5.7), for a given value of x	92
8.1	Comparison of stagnant proportion results with analytical predictions made from Equation 3.	108

8.2	Comparison of perturbation theory results with base time scales of 1 second and 300 seconds to the simulation results.	109
9.1	Simplified algorithm for simulating PRP residential water demands. All random variables are generated independently	112
9.2	Water distribution network used in this study. The color scale show demands (nodes) and flows (pipes). The source and monitoring locations used in the simulation are labeled.	113
9.3	Distribution of Reynolds Numbers	114
9.4	Centers of consumed mass from the tank source for six different time steps. The centers of mass are connected in chronological order.	115
9.5	Solute breakthrough curves at three different monitoring nodes for the tank sources	116
9.6	Center of consumed mass results for injections at nodes J-4382 (left image) and J-4991 (right image).	117
9.7	Solute breakthrough curves at three monitoring nodes form injection at the nodes sources.	118
10.1	Example network used in simulations. The color scales show the base demand (gpm) at the nodes and the flow (gpm) in the pipes. The red circle shows the location of node 435, the contaminant source.	124
10.2	Example output showing amount of source mass assigned to different nodes	125
10.3	Summary of results showing the objective function value (left), the mass fraction assigned to the correct source node(middle), and the entropy (right) all as a function of time.	126
11.1	Basic MLE model structure.	141
11.2	Example elicitation guide for determining attribute values input to the MLE model.	142
11.3	Markov Latent Effects (MLE) model for evaluating the security of water utility assets against willful attack. Each open box represents a decision element and each shaded box an external input, while weighting values are the numbers outside the boxes. The connecting lines show the direction in which the data are aggregated. Output from the model is found in the Asset Security element at the top of the figure.	143

13.1	Schematic of pipe cross-type joint.	155
13.2	Velocity field within pipe cross-type joint at 10 seconds simulation time.	155
13.3	Turbulent viscosity within pipe cross-type joint at 10 seconds simulation time.	156
13.4	Normalized simulated tracer breakthrough at outlets.	157
13.5	Tracer concentration within pipe cross-type joint at 9, 10 and 11 seconds simulation time, respectively.	157
13.6	Schematic of experimental setup.	158
13.7	Normalized experimental tracer breakthrough at outlets.	159
13.8	Fully developed turbulence flow in a duct geometry.	160
14.1	(a) Constant x-component of velocity isosurfaces with streamlines and temperature contours on slice plane for thermal convection problem with $Ra=1000$; (b) Steady-state x-component of velocity on centerline cutting plane for model 3D building . . .	168
14.2	Comparison of iteration count as a function of problem size for the geometric and algebraic two-level preconditioners for the 3D thermal convection problem	170
14.3	Comparison of iteration count as a function of problem size for the three-level preconditioner with different size medium level for the 3D thermal convection problem	172
15.1	Calculated vector flow field for $Re = 10000$	197
15.2	Flow field with 30 irregularly spaced sensors	198
15.3	Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source locations are the points (10.0, 1.0) and (4.5, 10.0)	202
15.4	Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source locations are the points (1.0, 14.0) and (1.0, 4.0)	203
15.5	Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source location is the point (14.5, 0.5)	204

15.6	Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source locations are the points (3.0,4.0) and (10.0,8.0)	205
15.7	Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source locations are the points (6.0,12.0) and (16.0,6.0)	206
15.8	Table giving run times and the number of <i>O3D</i> iterations at various levels of mesh coarsening	207
16.1	Two-D cross-section of the 3D airport terminal model. Also shown is an example flow pattern and contaminate release simulation.	210
16.2	Source and state sensor profiles for a transient inversion problem for $P = 6$ source basis vectors with basis functions ϕ_j that span $M = 3$ sampling periods of the state and $\tilde{n}_u = 2\tilde{n}_s$ (i.e. twice as many source parameters that state observations in a time snapshot).	220
16.3	Example compressed direct sensitivity matrix \hat{D} for the example in Figure 16.2. . .	221
16.4	Example source release centered at time $t = 3$ shifted one time step to $t = 4$	221
16.5	Two impulse releases, one at spatial node 2 with magnitude 2.0 and another 16 seconds later at spatial node 6 with magnitude 0.5.	223
16.6	Progression of inversion results using 16 sensors with sensor snapshot readings every 8 seconds for the attach scenario shown in Figure 16.5.	225
16.7	Comparison of relative errors (16.5.39) of the inversion result for the attach scenario shown in Figure 16.5. These relative errors are shown for different numbers of random sensors 4, 8, 16, 32, 64 and for times after initial release of 8 through 64 seconds. The number of sensors increases from back to front and the successive inversions progress from left to right.	226
16.8	Comparison of the inversion results for the attach scenario shown in Figure 16.5 after $t = 64$ seconds for varying numbers of sensors.	227
16.9	Spike release of magnitude 2 at spatial node 4 after 8 seconds.	229
16.10	Comparison of relative errors (16.5.39) of the inversion result for using objective scaling and/or bounds for the source shown in Figure 16.9. These relative errors are shown for different numbers of random sensors 4, 8, 16, 32, 64 and for times after initial release of 8 through 64 seconds.	230

16.11	Comparison of the inversion result for 32 sensors at time $t = 40$ seconds after the initial release at $t = 0$ seconds for the source shown in Figure 16.9. Note the different scales of the z-axis in each case!	231
16.12	Relative error (16.5.39) and CPU time for online inversions for a 32×32 , $n_u = 1024$ source discretization of an impulse release at spatial node 8. The relative errors and CPU times are shown for different numbers of random sensors 4, 8, 16, 32, 64 and for times after initial release of 8 through 64 seconds. Note: The order of the number of sensors is reversed between the two charts to improve readability.	232
16.13	Relative error (16.5.39) and CPU time for online inversions for a 32×32 , $n_u = 1024$ source discretization of an impulse release at spatial node 8 with the QP solve limited to 2.0 seconds. Compare to Figure 16.12	234
17.1	Sensitivity of the inversion result to the sensor array density. The target initial concentration is shown in the upper-left corner, and inversion results using successively finer sensor arrays are shown in the subsequent images. As the number of sensors in each direction increases, the quality of the reconstruction of the initial concentration plume improves. $L^2(\Omega)$ norm relative errors are 0.79, 0.49, and 0.34 for the $6 \times 6 \times 6$, $11 \times 11 \times 11$, and $21 \times 21 \times 21$ sensor arrays, respectively. Inversion using the $21 \times 21 \times 21$ sensor array takes 2.5 hours on 64 processors of the Alphaserver EV68 system at the Pittsburgh Supercomputing Center. CG iterations are terminated when the norm of the residual of (17.2.8) is reduced by five orders of magnitude. Topographical elevation has been exaggerated for visualization purposes.	253
17.2	Illustration of the predictive capabilities of our inversion algorithm, using an $11 \times 11 \times 11$ sensor array. Forward transport of the actual initial concentration is compared with forward transport of the reconstructed initial concentration plume. The trajectories are close to each other, and the comparison improves with time.	254
18.1	Domain	264
18.2	The error (18.3.22) versus number of modes for the goal-oriented optimized basis, the POD basis, and balanced truncation.	265
18.3	Error in temperature prediction for each snapshot using POD and optimized basis. The optimized basis was selected so as to minimize the error over the region $0.5 < x < 1, 0.5 < y < 1$. Errors are shown for four of the nine points contained within this region.	267

18.4	Norm of the error in temperature prediction over the entire domain for each snapshot using POD and optimized basis. The optimized basis was selected so as to minimize the error over the region $0.5 < x < 1, 0.5 < y < 1$	267
18.5	Reduced-order model temperature prediction for snapshot number 41 using optimized (left) and POD (right) basis. The optimized basis was selected so as to minimize the error over the region $0.5 < x < 1, 0.5 < y < 1$	268

List of Tables

2.1	Notation for Continuous Formulation: Equations (2.1.1-2.1.8).....	29
3.1	Optimization Results with Fixed Time Discretization	44
3.2	Formulation Parameters for Different Subdomain Sizes	44
4.1	Enumerated Solutions to Grid Network Example.....	53
4.2	Detailed Results for Test 1	57
4.3	Detailed Results for Test 2	57
5.1	Computational results for MIP solution of each of the test networks.....	66
5.2	Computational results for the <i>RW</i> heuristic on each of our test networks.	67
6.1	Optimal values for various performance objectives under both node-only and edge-only sensor placements for a range of sensor budgets N_s on the SWU network.....	73
6.2	Optimal values for various performance objectives under edge-only placements for a range of sensor budgets N_s on the MET network.....	75
6.3	Percentage and absolute deviations from optimal for various objectives given a <i>pe</i> -optimal solution; results are for the SWU network, under node-only placements.	76
6.4	Percentage and absolute deviations from optimal for various objectives given a <i>wtd</i> -optimal solution; results are for the MET network, under edge-only placements.	77
7.1	Computational Results for Complete Sensor Placement Solutions [†]	95
7.2	Computational Results for Bilinear Model for a Fixed Sensor Placement [†]	96

8.1	Parameter values for residential water demand simulations	101
8.2	Reynolds number and stagnant proportion results of the single day simulation	105
8.3	Travel time and stagnant proportion results of the multi-day simulations	106
10.1	Input parameters for the PRP demand simulator, PRPsym	125
11.1	Description of each of the direct inputs to the MLE asset security model	138
11.2	External attribute values input to the MLE model for seven threat scenarios and two mitigated threat scenarios. Model results are given in Table 11.1	139
11.3	Aggregate decision element scores calculated for seven threat scenarios and two mitigated threat scenarios. Model inputs are given in Table 11.2	140
12.1	Threat scores, as calculated with the MLE model, for the 9 sub-sampled event scenarios.	151
12.2	Numerical consequence values calculated for the 9 sub-sampled event scenarios subject to the four different bases of analysis. Note that consequence values are assumed to be independent of the assailant.	151
12.3	Risk assessment scores and accompanying rankings for the 9 sub-sampled event scenarios subject to the four different bases of analysis. Note: Exposure refers to number or people	151
14.1	Comparison of geometric and algebraic two-level preconditioners for the 3D thermal convection problem; ASCI Red machine.	169
14.2	Scalability study of three-level preconditioner (GS1/ILU/SuperLU) for the steady 3D thermal convection problem (aggregation: METIS; ParMETIS); Cplant machine. 171	171
14.3	Scalability study of three-level preconditioner (GS1/ILU/SuperLU) for the transient 3D thermal convection problem. METIS (ParMETIS) aggregation used on the fine (middle) mesh with 100 nodes per aggregate; Cplant machine.	173
14.4	Scalability study of three-level preconditioner (GS1/ILU/SuperLU) for the steady 3D building problem with hexahedral mesh (aggregation: METIS, ParMETIS); Cplant machine.	175
14.5	Scalability study of two-level preconditioner (ILU/KLU) with geometric coarse mesh for the steady 2D thermal convection problem; ASCI Red machine.	175

14.6 Scalability study of 1-level (ILU) and three-level preconditioner (GS1/ILU/KLU) for the steady 3D building problem with tetrahedral mesh (aggregation: METIS, ParMETIS); Cplant machine.	176
14.7 Scalability study of three-level preconditioner (GS1/ILU/SuperLU) for the transient 3D building problem (LES-k) with hexahedral mesh. METIS (ParMETIS) aggregation used on the fine (middle) mesh with 100 nodes per aggregate; Cplant machine.	177
14.8 Comparison of different preconditioners for the 3D model building; steady-state; fine mesh with 10.3 million unknowns; 128 processors on Cplant machine.	177
14.9 Comparison of different preconditioners for 3D model building; transient LES; fine mesh with 13.1 million unknowns; 1000 processors on Cplant machine	178
14.10 Scalability study of three-level preconditioner (GS1/ILU/KLU and ILU/ILU/KLU) for solution of a convection-diffusion equation in the 3D building. METIS (ParMETIS) aggregation used on the fine (middle) mesh with 100 nodes per aggregate; Cplant machine.	179
16.1 Breakdown of CPU times for online inversion of a 32×32 , $n_u = 1024$ source discretization of an impulse release at spatial node 8 using 4 and 64 spatial sensors.	233
17.1 Sensitivity of the inversion quality to the sensor array density. The sensitivity of the inversion quality to the sensor density is reported in terms of the relative errors in both the L_2 and ∞ norms, the wall-clock time, and the number of CG iterations. As the number of sensors in each direction increases, the quality of the reconstruction of the initial concentration plume improves. The additional information resulting for a higher sensor density results in more CG iterations. In these simulations, the regularization parameter β was held constant at 0.01 and the diffusion coefficient was fixed at 0.05.	248
17.2 Sensitivity of the inversion quality to the choice of regularization parameter, β. The sensitivity of the inversion quality to the choice of regularization parameter is reported in terms of the relative errors in both the L_2 and ∞ norms, the wall-clock time, and the number of CG iterations. As the regularization parameter, β , is reduced, the quality of the inversion improves, when measured by the relative error. However, the problem becomes more ill-posed resulting in more CG iterations and a longer run-time. In these simulations, the an $11 \times 11 \times 11$ array of sensors was employed and the diffusion coefficient was fixed at 0.05.	249

- 17.3 **Sensitivity of the inversion quality to the amount of diffusion.** The sensitivity of the inversion quality to the amount of diffusion is reported in terms of the relative errors in both the L_2 and ∞ norms, the wall-clock time, and the number of CG iterations. As the transport problem becomes more diffusion dominant (i.e. the diffusion parameter increases), the quality of the inversion degrades, based on both the relative L_2 and infinity norms. In these simulations, the an $11 \times 11 \times 11$ array of sensors was employed and the regularization parameter was fixed at 0.01. 249
- 17.4 **Sensitivity of the inversion quality to noise in the sensor readings.** The sensitivity of the inversion quality to noise in the sensor measurements is reported in terms of the relative errors in both the L_2 and ∞ norms, the wall-clock time, and the number of CG iterations. Based on the relative errors, a small amount of random noise in the sensor readings does not significantly affect the inversion quality. In these simulations, the an $11 \times 11 \times 11$ array of sensors was employed, the diffusion coefficient was constant at 0.05, and the regularization parameter was fixed at 0.01. 250
- 17.5 **Fixed size scalability of unpreconditioned and multigrid preconditioned inversion.** Here the problem size is $257 \times 257 \times 257 \times 257$ for all cases. We use a three-level version of the multigrid preconditioner described in Section 17.3. The variables are distributed across the processors in space, whereas they are stored sequentially in time (as in a multicomponent PDE). Here *hours* is the wall-clock time, and η is the parallel efficiency inferred from the runtime. The unpreconditioned code scales extremely well since there is little overhead associated with its single-grid simulations. The multigrid preconditioner also scales reasonably well, but its performance deteriorates since the problem granularity at the coarser levels is significantly reduced. Nevertheless, wall-clock time is significantly reduced over the unpreconditioned case. 251
- 17.6 **Isogranular scalability of unpreconditioned and multigrid preconditioned inversion.** The spatial problem size per processor is fixed (stride of 8). Ideal speedup should result in doubling of wall-clock time. The multigrid preconditioner scales very well due to improving algorithmic efficiency (decreasing CG iterations) with increasing problem size. Unpreconditioned CG is not able to solve the largest problem in reasonable time. 252
- 18.1 Comparison of optimization results. The objective function given by (18.3.22) is evaluated for the optimized basis (\mathcal{G}_{opt}) and the POD basis (\mathcal{G}_{pod}). 266

Executive Summary

Chemical/Biological/Radiological (CBR) contamination events pose a considerable threat to our nation's infrastructure, especially in large internal facilities, external flows, and water distribution systems. Because physical security can only be enforced to a limited degree, deployment of early warning systems is being considered. However to achieve reliable and efficient functionality, several complex questions must be answered: 1) where should sensors be placed, 2) how can sparse sensor information be efficiently used to determine the location of the original intrusion, 3) what are the model and data uncertainties, 4) how should these uncertainties be handled, and 5) how can our algorithms and forward simulations be sufficiently improved to achieve real time performance? This report presents the results of a three year algorithmic and application development to support the identification, mitigation, and risk assessment of CBR contamination events. The main thrust of this investigation was to develop 1) computationally efficient algorithms for strategically placing sensors, 2) identification process of contamination events by using sparse observations, 3) characterization of uncertainty through developing accurate demands forecasts and through investigating uncertain simulation model parameters, 4) risk assessment capabilities, and 5) reduced order modeling methods. The development effort was focused on water distribution systems, large internal facilities, and outdoor areas.

Efficient inversion methods were developed in an attempt to make use of sparse sensor information to determine the location and character of the intrusions. The general idea is to reconstruct these initial conditions so that accurate forward predictions can then be issued. A least squares formulation constrained by convection-diffusion is solved using large scale optimization methods. In all three flow domains, a convection-diffusion system represents the transport of the contaminant. The velocity field is calculated in each flow domain by considering appropriate dynamics, such as hydraulics in the case of water networks and Navier Stokes equations in the case of internal and external flows. The challenging aspect of this least squares problem is primarily related to the computational expense of solving the large number of inversion and state variables, which exist at each discretization point of the flow domain. We make use of simultaneous analysis and design methods to solve this constrained problem as efficiently as possible. In an attempt to achieve real time performance, special solution methods were developed consisting of subdomain decompositions for water systems, off-line/on-line approaches for internal facilities, and multigrid preconditioners for external flows. Although many technical issues still need to be addressed, the implementation of our algorithms and

methods using prototype datasets demonstrate sufficiently scalable and efficient performance to suggest the potential of real time performance.

Assuming that only a few sensors can be installed, the question of where to appropriately place sensors must be addressed. Integer programming and combinatorial methods are used to perform optimal sensor placement. We focused our efforts on water distribution systems because these systems can easily be mapped to mathematical graphs so that current integer programming solution methods can be leveraged. Difficult issues however need to be addressed ranging from extensions to dynamical systems, identifying appropriate formulations, and compensating for uncertainties. A mixed-integer programming (MIP) formulation for sensor placement optimization is developed that precisely characterizes the temporal impact of contamination events. Our experiments demonstrate that exact and heuristic solvers can be effectively applied with reasonable computational effort to sensor placement problems for networks with ten thousand junctions, which is at least an order of magnitude larger than problems commonly used in the water distribution community literature. The general goal of providing maximum levels of protection can be formulated in a number of ways such as minimum detection time, maximum consumer protection, minimize installation and maintenance cost. We developed mixed-integer linear programming models over a range of design objectives. Using two real-world water systems, we show that optimal solutions with respect to one design objective are typically highly sub-optimal with respect to other design objectives and that robust algorithms must carefully and simultaneously consider multiple, disparate design objectives.

Both the inversion and sensor placement algorithms assume ideal conditions consisting of known boundary conditions, accurate historical records of consumption rates (for water networks), correct representation of the underlying physics and perfect sensor measurements. In order for our algorithms to eventually be considered for operational use, the management of model and data uncertainties need to be addressed. The complete quantification of model and data uncertainties of all three flow domains however was beyond the scope of this project, and we therefore limited our investigation of uncertainties to water distribution systems. Several uncertainty issues are investigated including the use of robust algorithms for sensor placement, inaccuracies with the chemical transport simulation, the variability of demands, and the refinement of the regularized source inversion solution. First, our algorithms make use of a network simulator that approximates the transport of chemical transport. We demonstrate through high fidelity simulation and laboratory validation that simulation models cannot assume perfect mixing in certain geometric configurations and even though these network simulators have been used successfully for general operational management, modifications will be necessary to use these tool for mitigating contamination events. Second, we use robust optimization algorithms to solve sensor placement problems which are formulated as mixed integer problems for which the objective coefficients are not known with certainty. The computational complexities of solving these problems are illustrated. Third, we investigate the use of demand prediction tools which are based on detailed field observations, to show that the time scale at which hydraulic demands are discretized can influence solute transport behavior in simple dead-end mains. Furthermore simulations of the flow and solute transport in a water distribution system show that flow across

the system as a whole is less sensitive to changes in the demand discretization than flow along dead-end mains. Fourth, we developed mixed integer quadratic program (MIQP) solutions to refine the multiple node solution that results from the solution of the regularized constrained least squares formulation. A regularization term is added to the least square formulation so that an undetermined problem can be solved uniquely. The solution however is a smoothed approximation which for water network results in the identification of multiple neighboring nodes as possible locations where the intrusion could have originated. The MIQP methodology can efficiently reduce the injection possibilities and more importantly can distinguish between single and multiple injections.

Development of efficient algorithms is critical to eventually support the decision making process at appropriate time scales associated with the particular contamination scenario. In water distribution systems, a subdomain method was developed to address large network datasets. By applying the source inversion algorithm to just the infected nodes we are able to successfully invert for the source of an intrusion, provided our subdomain contains enough sensors. Under these assumptions, we are able to invert in real time for any size dataset using minimal computational compute resources. For internal spaces, we developed steady state solutions and investigated the sensitivity of the level of detail of the discretization to the solution. Special quadratic programming solvers were applied. Also, for internal facilities, we developed an online-offline capability which also provides a real time source inversion capability using minimal compute resources. Although, the velocity field calculation could be considered an off-line calculation, an efficient implementation is still critical in providing support for mitigation procedures. We investigated algebraic multilevel domain decomposition preconditioners for solving linear systems associated with Newton-Krylov methods. We show excellent scalable results for 1024 processors. In the case of external flow, we solved the inversion problem with a null space method in which the primary solve involves a reduced Hessian. We developed a scalable geometric multigrid preconditioner that resulted in a reduction of linear solves with increased grid resolution and sufficiently reduced the overall solution time for real time decision making. However a significant number of processors are still required to achieve real time performance and in order for these methods to be considered for operational use, the computational requirements must be reduced ideally to a single processor. This operational requirement motivated the investigation of model reduction methods. We developed an approach for determining a projection basis that uses a goal-oriented, model-based optimization framework.

Recent amendments to the Safe Drinking Water Act emphasize efforts toward safeguarding our nation's water supplies against attack and contamination. Specifically, the Public Health Security and Bioterrorism Preparedness and Response Act of 2002 established requirements for each community water system serving more than 3300 people to conduct an assessment of the vulnerability of its system to a terrorist attack or other intentional acts. Integral to evaluating system vulnerability is the threat assessment, which is the process by which the credibility of a threat is quantified. Unfortunately, full probabilistic assessment is generally not feasible, as there is insufficient experience and/or data to quantify the associated probabilities. For this reason, an alternative approach was developed predicated on Markov Latent Effects (MLE) modeling, which

provides a framework for quantifying imprecise subjective metrics through possibilistic or fuzzy mathematics. We developed a MLE approach and demonstrated the utility of this method within the context of water supply system threat assessment.

Chapter 1

Introduction

Bart van Bloemen Waanders

1.1 Background

Chemical, Biological or Radiological (CBR) contamination events pose a considerable risk to national security through immediate and long term impact on human health, the pollution of the environment/facilities, complications associated with mitigation procedures and the long term impact on our local and national economy. Large internal facilities, water distribution systems, and external flow regions are especially vulnerable to intentional or accidental releases of harmful agents. Currently our nation's infrastructure is relatively unprotected and there are no real-time warning systems available other than general observations of human illnesses reported by health facilities during a contamination event. History has demonstrated the destructive nature of contaminations such as the outbreak of cryptosporidiosis in Milwaukee, the Union Carbide pesticide manufacturing plant explosion in Bhopal, and the sarin gas release in a Tokyo subway. In addition there are numerous accounts of the devastation caused by biological and chemical warfare. The outbreak of cryptosporidiosis in Milwaukee was one of the largest contamination event in the history of water distribution system in the US. An estimated 403,000 persons became ill, of whom 4,400 were hospitalized. The number of deaths were estimated at about 100. The cost associated with decontamination, equipment upgrades and legal suites were on the order of \$100M. In 1984, The Union Carbide pesticide manufacturing plant exploded in Bhopal, India, releasing a huge cloud of toxic gas that caused 8,000 deaths within a few days. An estimated half a million people suffered injuries. Over the last two decades, more than 20,000 additional people have died as a result of the original exposure, and an estimated 120,000 still suffer significant health impacts. In Tokyo, intentional releases of sarin gas in subway station caused 5 fatalities and hospitalization of 565 people. Although this was a relatively small scale attack, the ramifications were significant.

Several researchers have developed flow simulation tools to predict the transport of contaminations events. The primary motivation for providing timely predictions of the contamination spread is to identify the

safest evacuation routes, reroute vehicles, propose decontamination strategies, and implement flow control (for water distribution and internal flow systems). The National Atmospheric Release Advisory Center's (NARAC) simulation tools have provided support for several large scale contamination events, such as nuclear releases, battlefield situations, accidental toxic releases, and volcanic eruptions, by using particle-in-cell models [158]. In addition, high fidelity simulation capabilities have been developed to track contaminants in urban canyons [128]. Although very little technical detail is available, it appears that a sophisticated range of primarily forward modeling capabilities have been developed to address different types of contamination events at a range of spatial scales. NARAC's capabilities are focused on the prediction mode, although reconstruction capabilities are being explored [217]. The air quality group at Los Alamos National Laboratories has developed a multilevel transport capability to track contaminants in urban canyons [243, 165, 58, 124]. This capability provides a multilevel approach to the transport predictions, in which high fidelity CFD models are used to track contaminants around buildings and appropriate dynamics compensated with weather conditions are used certain vertical distances away from the urban canyons. Aliabadi et al. has also developed sophisticated CFD modeling capabilities to track contaminants in Urban canyons [20, 21]. Their capabilities are limited to incompressible Navier Stokes with LES-based turbulence models. Fast discretization methods have been developed to handle large datasets. Significant work has been conducted in the area of predicting pollutant transport in internal facilities [150, 199, 213, 98]. These methods are primarily based on nodal, multizone network models that assume simplified dynamics for performance issues. For the most part these models have been used in prediction mode assuming initial conditions and source terms. Reconstruction of initial conditions and source terms has been attempted through Bayesian framework techniques [216]. However, these methods do not scale to large systems and only small datasets have been used in numerical experiments with nodal models. Very little work had been done in the area of characterizing contamination events for water distribution systems. The primary goals of managing water distribution systems consist of maintaining water quality, designing expansions to growing demands, ensuring adequate pressure for fire fighting services, and providing uninterrupted service to consumers. Sophisticated simulation capabilities have been developed to address the general operations of water distribution[179]. Optimization methods have been applied to address water quality maintenance, design problems [184, 167, 53, 227].

All these forward prediction capabilities assume values for initial conditions, boundary conditions, and source terms. In the event of a contamination, the lack of exact location, magnitude, and characteristics of the contaminant introduces significant uncertainty in the forecasts. Without this important information, response strategies are forced to issue predictions with considerable uncertainty until the simulations can be validated by modifying the initial conditions and manually matching predictions with field observations. It is difficult to achieve real time performance with such an iterative procedure. In addition, most forward simulators were developed either to manage general facility operations (in the case of water distribution), or provide very efficient but approximate predictions (in the case of internal facilities). For security applications, the goal of the forward simulation is to predict the movement of contaminants at sufficient levels of fidelity so that maximum number of people and key components of the infrastructure can be protected (such as hospitals in water distribution networks and exits in internal facilities), in addition to effectively installing sensors, and providing accurate information to support decontamination procedures.

The implementation of early warning systems has been the subject of significant research and development in an attempt to improve protection measures. The general concept is to instrument internal facilities and distribution systems with sensors that can detect harmful agents in real time and sound an alarm. In the case of external contamination events, a variety of measurement methods are being considered, such as

deploying sensor systems in unmanned vehicles, weather balloons, satellite imagery. A number of difficult issues arise however as a result of considering the use of an early warning system, such as: 1) where should sensors be placed, 2) how can sparse sensor information be efficiently used to determine the location of the original intrusion, 3) what are the model and data uncertainties, 4) how should these uncertainties be handled, and 5) how can our algorithms and forward simulations be sufficiently improved to achieve real time performance? The goal of this project was to develop algorithms to address these issues. An important aspect of these developments is to be able to assess the value of early warning systems and supporting algorithms to the overall security of our infrastructure. To this end we have committed effort to the development of a risk assessment methodology. Fundamental algorithms based on probability and risk were developed and applied to a prototype water network. It should be noted that these tools can easily be applied to internal and external flow domains.

1.2 Project Goals

The goal of this project was to develop algorithms, methods, and numerical simulation capabilities in support of the identification and mitigation of contamination events. We focused on key components consisting of 1) source inversion and reconstruction of initial conditions for internal, external, and water distribution networks using sparse sensor information, 2) integer programming and combinatorial methods for sensor placement 3) development of online-offline, scalable preconditioners and reduced order methods in an attempt to achieve real time capabilities, 4) improving incompressible Navier Stokes fluid flow simulation for large internal facilities, 5) improvement of forward simulation to address model uncertainties, and 6) development of risk assessment tools. The following sections describe in further detail the goals for each of these topics. Although this project concentrated primarily on applications in support of contamination events, these methods and algorithms are applicable to many other application domains and problem types.

1.2.1 Source Inversion

Current numerical tools to support the characterization of contamination events are based on predicting the transport characteristics of a contaminant in certain flow domains. These forecasts however often exclude accurate information about the initial conditions and source terms. The location and character of the initial contamination event are typically not known and consequently the accuracy of the forward predictions is compromised. The determination of initial conditions and source terms is the goal of the inverse problem. Given some sparse concentration information from sensors, can we invert for the location and character of the contamination event? The goal is to use a least square formulation constrained by the transport dynamics to minimize the difference between observations and predictions. Because inversion parameters are imposed at every discretized point in the domain, the challenge and the goal is to develop algorithms and methods that can solve this problem robustly and efficiently. To address the large number of inversions in combination with the complex dynamics in the constraints, we appeal to intrusive optimization methods [233, 18]. The general strategy is to solve for state and design (inversion) parameters simultaneously and solve the inverse problem merely at the cost of a few forward solutions.

A range of implementation strategies could be considered and is somewhat dependent on the problem formulation in addition to the underlying dynamics. Several implementations and methods are demonstrated for different flow domains. For internal facilities, we concentrate on decoupling the calculation into an offline and online computations and making use of direct sensitivities. For the external flow inversion, we use adjoint based sensitivities which requires all online calculations. This in turn requires the consideration of special preconditioning techniques to achieve efficient performance. In the case of water distribution systems, we make use of a Newton based algorithm, even though the formulation requires a simpler quadratic program solution method. Our justification for pursuing a more general Newton solver is based on future extensions to handle a nonlinear problem, such as velocity inversion or the control of a contamination event. We translate chemical transport into a set of algebraic equations, making this calculation scalable to large datasets.

1.2.2 Sensor Placement

Assuming that early warning system eventually will be able to detect a range of contaminants and assuming that only small numbers of sensor can be installed, the critical question becomes: where should these sensors be installed. If a large number of sensors are available and logistically there are no installation limitations, the location of sensors are less critical to our ability to efficiently interpret the data. However the flow domains that are of interest are spatially distributed over a large geographic area, especially in the cases of water distribution and external flow areas. In the case of internal facilities, the installation is less problematic but still the instrumentation of the entire space is not practical. Therefore the goal is to optimally place sensors in an attempt to provide maximum levels of protection by using a limited number of sensors.

The optimal sensor placement problem is inherently a discrete problem and therefore be posed as a mixed integer program, which lends itself to graph based theory and combinatorial experimentation methods. For the sensor placement problem we have therefore focused our efforts on water distribution networks, since these datasets can be mapped conveniently to graphs. Although an obvious criteria to use in sensor placement is our ability to efficiently perform source inversion, the formulation for such a problem is not obvious. Instead we have focused on alternative, but related criteria such as minimum detection time and maximum population protection. We focus on three specific areas: 1) extending static MIP formulations to dynamics applications, 2) selection of different criteria for the sensor placement formulations, and 3) handling of modeling and data uncertainties in a robust optimization methodology. Significant amount of work has been done in this areas using static formulations. Unfortunately, static formulations fail to model the time-varying flow characteristic and only approximate temporal variations. The focus of this work is on dynamic methods so that the dynamics of chemical transport can be adequately represented. Our formulation is identical to the well-known p-median facility location problem [155]. MIP solver techniques for the p-median problem can therefore be leveraged. One of the fundamental issues associated with sensor placement is selecting the appropriate criteria for the minimization formulation. The goal is to evaluate a range of criteria, ranging from population expose to extent of the contamination. Our goal is evaluate different objectives and test different sensor budgets on real-world datasets. Model and data Uncertainties need to be addressed in our analysis. We leverage robust optimization methods which seek a solution that minimizes some measure of worst performance with respect to uncertainty in the data. We make use of mixed integer linear program models (MILP).

Both internal and external flow scenarios are simulated by three dimensional continuous domains and cannot be easily mapped to a particular flow graph. Sensor placement is therefore still an open question for these types of flow domains. These criteria and others that have been tested are all related to the underlying velocity fields, which ultimately dictate contaminants transport. The source inversion algorithm is constrained by the transport and therefore there is strong connection to the sensor placement criteria.

1.2.3 Forward Modeling

The underlying fluid flow dynamics can be described by the convection-diffusion (CD) equation:

$$\frac{\partial c(x,t)}{\partial t} - k\Delta c(x,t) + v(x,t) \cdot \nabla c(x,t) = 0 \quad (1.2.1)$$

where c represents concentration, k is the diffusion coefficient, and v is the velocity field. The calculation of the velocity field represents the main difference for the flow characteristics of the different transport mechanisms for internal, external and water distribution flows. For water distribution systems, simple hydraulic relationships between pressure, velocity and friction coefficients are used. For internal facilities, velocity field are calculated using incompressible Navier Stokes. Ideally weather models need to provide the velocity fields for external flow domains. For this project we used incompressible Navier Stokes as a first order approximation.

The goal of the forward modeling project was to develop efficient and accurate solutions of relatively complex fluid flow. We choose to focus our efforts on high fidelity, turbulent models for internal facilities, recognizing that these techniques could also be applied external flows and even for small geometric components in water distribution system. In addition to creating detailed models and representing the fluid flow as accurately as possible, it was equally important to investigate efficient solution techniques. A special multi-level preconditioning scheme was developed and testing on large datasets involving turbulent flow.

For water distribution systems, we concentrated on the prediction of demand information, which essentially are source and sinks for simulation models. In these network models a demand value is assigned to practically each node which represents consumption rates for that part of the dataset. This can represent a household, but in the case of most real datasets, it most likely represents a collection of taps within a neighborhoods. Naturally, these demands can vary considerably and the flow characteristics are highly dependent on the accuracy of this data. Although utility companies attempts to collect demand information, this is a considerable data management issues and as a result, most utilities only have course demand information covering limited amount of time periods. Our goal was to develop prediction tools for demand information in the absence of reliable demand measurements.

1.2.4 Uncertainties

Developing numerical algorithms to support an early warning system is a significant undertaking requiring the development of complete software infrastructure capable of continuously monitoring data streams, discriminating between false and positive readings, and performing in real time a range of functionality. In addition, this software needs to be sufficiently robust so that it is insensitive to model and data uncertainties. Although a comprehensive study of all possible uncertainties was beyond the scope of this project, the goal was to start investigating significant sources of uncertainties, initiate the development of robust algorithms to reduce these uncertainties and identify possible strategies for future work. This investigation was conducted in the context of water distribution systems consisting of 1) evaluating the forward chemical transport model in EPANET, 2) developing demand generation forecasting methods, 3) evaluation of source inversion with variable demand information, 4) refinement of the regularized solution using a MIQP method, 5) robust sensor placement strategies with data variations.

1.2.5 Risk Assessment

Haines and others [108] reviewed needs and opportunities to reduce the vulnerability of public water systems to willful attack. They developed a hierarchical holographic model [107] to better understand the complexity and interconnectedness that characterizes the security of water distribution systems. This model was also used to explore different approaches to hardening (as described in terms of security, robustness, resilience, and redundancy) water distribution systems against attack. Subsequent to this work, Ezell and others introduced an infrastructure risk analysis model [89] and applied it within the context of a municipal water distribution system [90]. The model provides an analytical methodology for quantifying risk that involves decomposition of utility operations along the dimensions of function, component, structure, state and vulnerability. Potential threats are identified through scenario modeling, while conditional and expected losses for each scenario are calculated via Asbeck and Haines [25] partitioned multi-objective risk method.

The events of September 11, 2001 had the effect of broadening and accelerating efforts to safeguard the nation's water utilities against terrorism and other threats. In particular, the Public Health Security and Bioterrorism Preparedness and Response Act of 2002 (PL 107-188) was enacted that added new drinking water security and safety requirements. The law required almost 8000 community water systems serving more than 3300 people to complete vulnerability assessments and prepare or update their emergency response plans. The required vulnerability assessments were intended to help water utilities evaluate the risk posed by potential threats and identify corrective actions that could reduce or mitigate the consequences of these adversarial actions [197]. Vulnerability assessments were to serve as a guide to the water utility by providing a prioritized plan for security upgrades, modification of operational procedures, and/or policy changes to mitigate risks. This was intended to be a dynamic process in which the utilities review their vulnerability assessments periodically to account for changing threats or changes to the water system.

In an attempt to quantify the impact of certain technologies on the protection of our infrastructure, some effort was devoted to the development of a risk assessment methodology. Our vision was to assess the overall risk factors associated with certain flow domains and demonstrate the potential reduction of risk by

installing early warning systems coupled to numerical algorithms. Assessing the risk of a contamination event is an important analysis to help determine the most vulnerable component of a system so that better protected can be applied, or possibly components can be redesigned. We investigate the use of Markov Latent Effect methods to calculate the cumulative possibility measure of attack likelihoods. The intent was to develop a framework to help assess the baseline risk and calculate the reduction in risk as a result of sensor installation and the use of numerical algorithms.

1.2.6 Real Time Performance

The need for real time performance is critical for the use of our algorithms in a practical and operational environment. All of our algorithms, such as source inversion and forward predictions must be capable of executing on relatively low budget computer resources so that this technology can be considered for general deployment to utilities and different response organizations. In addition, these tools must be capable of performing at a time scale equivalent to the decision making process. For internal facilities, this probably means order minutes, in the case of external flows and water distribution this can mean order tens of minutes or order hundreds of minutes depending on the size of the flow domain and flow conditions.

Our thrust has been to investigate a range of methods to exploit as many different solution strategies. Forward modeling and source inversion are the two primary targets for real time functionality, although manual sampling to help validate predictions can also be considered a real time functionality. The primary challenge for source inversion is the large number of state and inversion parameters that one needs to deal with. Typically, there are as many inversion parameters as there discretization points. In three dimensions the overall number of unknowns can easily reach well beyond order tens of millions of parameters (spatial-temporal-inversion parameters). To solve these types of systems efficiently, requires algorithms that scale with large numbers of processors but also are designed to fully leverage the internal linear algebra of the forward model. In this report we investigate several strategies: 1) calculating sensor dependent components during the online phase and pre-processing everything else in an “off-line” mode, 2) developing scalable multigrid and multilevel preconditioners to handle large linear solves, and 3) developing reduced order modeling strategies that can handle optimization.

1.3 Future Work

Delivering robust countermeasure tools will ultimately require coupling our algorithms to sensors and demonstrating real time performance in an automated environment. The functionality requirement of such a production system is considerable and, although we believe that all the fundamental technologies are available, a complete and robust implementation still requires effort in the areas of real time performance, uncertainty quantification, control, decontamination, and general software infrastructure development.

Our transport models are currently based on unstructured and structured finite element incompressible flow and transport capability. Our effort has focused mostly on the use of these codes to demonstrate the importance of accuracy through detailed discretized models and complete physics. We were able to leverage existing technology and quickly demonstrate that detailed discretizations and turbulence models

play a critical role in predicting the transport of contaminants. The resulting size of these problems required special solution technique and by implementing multi-level preconditioning schemes we demonstrated excellent speedup improvements. These performance observations are directly tied to all the underlying technologies of existing software and to accomplish the robust and real time functionality on small compute resources, we may have to investigate different technologies to manage these partial differential equations (i.e. finite volume with explicit time-stepping). In addition, even if we implement the most efficient algorithms to solve these systems, it is anticipated that the computer resources required to solve sufficient levels of detail are in excess of what may be available in the general community. To implement efficient and accurate flow and transport models along with source inversion capabilities for the response to, and possible control of, contaminant release events, the high fidelity continuum simulations described above (gas phase transport) must be reduced with a systematic methodology to representations that are computable in real time with reasonable computing resources. Mathematical reduced order modeling (ROM) techniques to provide these capabilities are potential technologies that can be leveraged. These methods differ from current operational modeling techniques, such as lower spatial resolution models (zonal models) or hierarchical physics based models (e.g. Gaussian plume dynamics), in that they are defined by abstract mathematical procedures from pre-computed off-line high fidelity simulation models. These techniques include for example proper orthogonal decomposition (POD) (also termed a Karhunen-Loeve expansion (KL)) and Centrodial Voronoi tessellation (or k-mean clustering) techniques.

Reliable and efficient numerical tools must be deployed to 1) identify the location of an attack, given sensor observations, 2) design nearly optimal sensor placement strategies, 3) control the HVAC systems in the event of an internal attack, and 4) support the clean up phase by providing simulation based decontamination designs. We have developed prototype capabilities that determine the location of a malicious attack event in internal facilities and external urban canyons. Our methods consist of special optimization methods applied to implicit convection-diffusion models that are explicitly coupled to Navier Stokes (NS) and hydraulic models for velocity field calculations. Considerable amount of prototyping work remains, including numerical experimentation to test quality of the solution and execution performance as a function of dataset size and problem complexity. Additionally, for the external source inversion problem appropriate flow physics for urban canyon simulations need to be accounted for.

Uncertainty quantification methods need to be investigated to take into account errors and variations associated with model parameters and measurements. One possible approach is to quantify uncertainties and perform optimization under uncertainty in attempt to provide a solution with some statistical characterization. Several formulations can be considered ranging from sampling to surrogate based methods. The selection of the appropriate method will depend on the target optimization goal and also on the type of inherent uncertainties. Another approach is to consider robust optimization where the goal is to minimize a cost function subject the maximum disturbances. This min-max problem can be reformulated as a regular minimization problem with inequalities, which in turn could be solved by an interior point method.

Optimization under uncertainty and robust optimization are growing areas of research and our strategy is to investigate several solution techniques. Several promising areas that appear to be suitable to our intrusive optimization methods and that may prove to be most efficient for large design spaces are multiperiod optimization, stochastic finite elements, and robust optimization. Our prototyping activities assume ideal sensor performance but clearly performance and characteristics of field measurements need to be incorporated into our computational models. Currently machine precision concentrations are reported to

our inversion algorithms whereas field sensors depend on a threshold value and are affected by measurement uncertainties. These operating conditions and uncertainties need to be identified and quantified so that they can be either characterized or accounted for in the final solution. Similar to sensor technologies, the threat needs to be accurately characterized in terms of flow properties, magnitudes, and uncertainties. Numerical prototyping has assumed ideal gas properties, but certain agents may require completely different representation in the physics models. CBR threats need to be characterized and accounted for in our operational models.

It should be noted that the development of these algorithms have far reaching benefits to other applications and problems. Large scale optimization problems arise in shape optimization, parameter estimation, complicated design issues, image processing, control of large plant facilities, environmental pollutant tracking, oil reservoir management, fire spread control, just to name a few. A significant number of these applications also require real time performance and have to deal with all the above mentioned issues. As a result of the interest in this area, several funding sources will provide the means to continue these activities. The reduced order modeling activities has been funded by our CSRF office. A decontamination proposal has been funded by the LDRD office. Uncertainty quantification for large design problems has funded some of our external collaborators (with some travel money to Sandia). A relatively large project is currently being funded by the EPA to transition some of these tools in an operational setting for water distribution systems.

1.4 Outline of the Report

This report is organized into two main algorithmic development parts for 1) water distribution systems and 2) internal/external flow domains. The first half of the report is devoted to algorithmic development for water distribution systems, starting with three chapters on source inversion. This is followed by a series of chapters on sensor placement problems using integer programming and combinatorial methods. The next three chapters discuss the development of demand data prediction and the effect of variable demands on source inversion. Risk assessment is the subject of the subsequent two chapter and the first half of the report is concluded with a discussion of the accuracy of chemical transport in the standard network simulator. The second half of the report is dedicated to internal and external flows, starting with a chapter on multilevel domain decomposition preconditioners for incompressible flow. The next two chapter discusses source inversion for the steady state and transient cases. The transient source inversion chapter discusses off-line and on-line methodologies in an attempt to achieve real time performance for large datasets. Real time reconstruction of initial condition for external flows is discussed in the next chapter and presents scalable results for a multi-grid preconditioning scheme. Finally, the report concludes with a chapter on reduced order modeling that is sensitive to a goal oriented performance objective.

Chapter 2

Contamination Source Determination for Water Networks

Carl D. Laird (Carnegie Mellon University), Lorenz T. Biegler (Carnegie Mellon University), Bart G. van Bloemen Waanders, Roscoe A. Bartlett

2.1 Background

The threat of accidental drinking water contamination is not new. More recently, however, concern over intentional contamination of municipal water networks has required us to consider novel protection measures. Drinking water networks are especially vulnerable to biological and chemical attack due to the large land area encompassed by the network and the number of access points. Any water outlet, such as a fire hydrant or even a household water faucet, can be an access point for backflow contamination into the network. As an alternative to physical security alone, sensors could be installed in the network to detect contaminant and initiate a means of protection from within the network itself. It is assumed that these sensors would be costly to purchase, install, and maintain, making it unreasonable to place sensors at every network node. Instead, it is desirable to consider as few sensors as possible. Should these sensors detect contaminant, it is important to provide an accurate measure of the source of the contamination. In this work, a nonlinear program is formulated to estimate the time and location of contamination sources, using time varying concentration data from an installed sensor grid.

Inverse problems like this one, are fundamentally different in nature from standard simulation problems. Traditional water quality simulations [147, 182, 181, 246, 210] assume injections of secondary species are known. They solve the network problem forward in time to find the propagation of these species through the network. This simulation of the output state of a model based on known inputs is referred to as the forward problem. By contrast, inverse techniques attempt to find the unknown inputs that give rise to a partially known output state. In the contamination source determination problem, injections are unknown. Instead, they are estimated using concentration measurements from the network. These inverse problems

are inherently ill-conditioned and pose unique difficulties not present in the forward problem [17, 235, 139, 177].

Nonlinear programming provides a framework for this inverse problem, which is formulated as a least squares minimization subject to the differential and algebraic constraints of the network water quality model. Unknown time dependent injection terms are introduced at every network node, and the solution of the nonlinear program provides an estimate of the time and location of contamination sources. The network water quality model contains partial differential pipe equations that are a function of both time and displacement. A naive discretization of this system in time and space produces a large scale, nonlinear math programming problem that is unreasonably large for current optimization tools. To overcome this difficulty, an origin tracking algorithm based on the Lagrangian technique of [147] is developed to reformulate the partial differential pipe expressions into a set of algebraic time delay constraints, removing the need to spatially discretize along the length of the pipes.

The effectiveness of this formulation is demonstrated on a small numerical test problem and a real municipal water network model.

2.1.1 Problem Formulation

Water distribution systems are often described (see, e.g., [180]) by a network of links and nodes, where links represent pipes, pumps, or valves, and nodes represent sources, tanks, or junctions. Usually, model size is reduced by collapsing regions of the network into single network nodes. Hydraulic calculations and water quality calculations are decoupled and, following dynamic hydraulic calculations, the resulting flow profiles are specified as known inputs to the water quality model. The water quality solution can then be determined by a variety of existing techniques [147, 182, 181, 246, 210]. Traditional water quality simulation methods can be classified as *Eulerian* or *Lagrangian* [181]. Eulerian methods discretize the network model in both time and space, tracking the concentration at fixed points or volumes within the pipe. Lagrangian methods discretize in time alone and track the concentrations of discrete volume elements as they move through the network. This work assumes that the flow profiles are known (or estimated) from flow measurements, hydraulic simulations, historical data, or some combination.

The water quality model is developed using \mathcal{P} , \mathcal{J} and \mathcal{S} to refer to the complete sets of all pipes, junctions, and storage tanks respectively, $\bar{c}_i(x,t), i \in \mathcal{P}$ to represent the concentration in the pipes, and $\hat{c}_k(t), k \in \mathcal{N}$ to represent the concentration at the nodes, where $\mathcal{N} = \mathcal{J} \cup \mathcal{S}$ is the complete set of all nodes, including junctions and storage tanks. Here, $t \in [0..t_f]$ is time, and $x \geq 0$ is the displacement along a pipe. The notation for designating pipe boundaries is based on flow direction, as shown in Figure 2.1, where $x=I_i(t)$ refers to the boundary where fluid is entering pipe i and $x=O_i(t)$ refers to the boundary where fluid is leaving pipe i . The index $k_i(t), i \in \mathcal{P}$ refers to the node connected at the inlet boundary (in the case of Figure 2.1, this is node A). Note that these designations are time dependent and change with the flow direction.

The following assumptions and simplifications are made. Pumps and valves are modeled as zero length pipes, reservoirs are modeled as junctions with known external sources. Plug flow is assumed for all pipes, and complete mixing occurs in all network nodes. Contamination is assumed to occur only at (modeled) network nodes, where an unknown, time dependent injection term is added to the model. All known sources are modeled as contaminant free. Contaminated sources will manifest in the new injection term



Figure 2.1. Link Boundary Design. $I_i(t)$ indicates the inlet of the link, based on the current flow direction, while $O_i(t)$ indicates the outlet of the link. The index $k_i(t)$ always refers to the node connected at the inlet.

added to these nodes. It is further assumed that flow rates of contaminations are negligible and do not affect existing network flow rates. The model is written with contaminant free initial conditions, although it could be formulated with initial concentrations as unknowns. The model is also written without decay reactions for the contaminant, although they can easily be included in the formulation. All but simple first order reactions will make the discretized constraint equations nonlinear. However, in this work, a nonlinear optimization package is used to solve the final problem and the inclusion of nonlinear reaction terms does not invalidate the approach.

Using the notation described in Table 2.1, the continuous problem is defined as,

$$\min_{m(t), \bar{c}(x,t), \hat{c}(t)} \Psi = \sum_{r \in \Theta_s} \sum_{k \in \mathcal{N}_k} \frac{1}{2} \int_0^{t_f} w_k(t) (\hat{c}_k(t) - \hat{c}_k^*(t))^2 \delta(t - t_r) dt \quad (2.1.1)$$

s.t.

$$\left. \begin{aligned} \frac{\partial \bar{c}_i(x,t)}{\partial t} + u_i(t) \frac{\partial \bar{c}_i(x,t)}{\partial x} &= 0, & (2.1.2) \\ \bar{c}_i(x=I_i(t), t) &= \hat{c}_{k_i(t)}(t), & (2.1.3) \\ \bar{c}_i(x, t=0) &= 0, & (2.1.4) \end{aligned} \right\} \forall i \in \mathcal{P},$$

$$\hat{c}_k(t) = \frac{\left(\sum_{i \in \Gamma_k(t)} Q_i(t) \bar{c}_i(x=O_i(t), t) \right) + m_k(t)}{\left(\sum_{i \in \Gamma_k(t)} Q_i(t) \right) + Q_k^{ext}(t) + Q_k^{inj}(t)}, \quad (2.1.5) \quad \forall k \in \mathcal{J},$$

$$\left. \begin{aligned} V_k(t) \frac{d\hat{c}_k(t)}{dt} &= \left(\sum_{i \in \Gamma_k(t)} Q_i(t) \bar{c}_i(x=O_i(t), t) \right) + m_k(t) - \left[\left(\sum_{i \in \Gamma_k(t)} Q_i(t) \right) + Q_k^{ext}(t) + Q_k^{inj}(t) \right] \hat{c}_k(t), & (2.1.6) \\ \hat{c}_k(t=0) &= 0, & (2.1.7) \end{aligned} \right\} \forall k \in \mathcal{S},$$

$$m_k(t) \geq 0,$$

$$(2.1.8) \quad \forall k \in \mathcal{N}.$$

Here the objective function (2.1.1) seeks to minimize the difference between measured sensor data and values calculated from the network model. Equations (2.1.2-2.1.4) model the network pipes, equation (2.1.5) models the junctions, equations (2.1.6) and (2.1.7) model the tanks, and (2.1.8) bounds the injection terms to be positive. Equation 2.1.6 is derived by first separating the overall mass accumulation term, $\frac{dM}{dt} = V_k(t) \frac{d\hat{c}_k(t)}{dt} + c_k(t) \frac{dV_k(t)}{dt}$.

Since the hydraulic information is a specified input, the only unknowns are the pipe and node concentrations, $\bar{c}_i(x,t)$, and $\hat{c}_k(t)$ respectively, and the mass injections, $m_k(t)$. The traditional forward problem is that of setting the mass injections, $m_k(t)$, and solving (using (2.1.2-2.1.7)) for the network concentrations. In the inverse problem (2.1.1-2.1.8), the optimization solution gives the complete time profiles for the observed concentrations. The values of the injection terms, $m_k(t)$, are the profiles of interest, where significantly positive values at a particular node indicate a potential contaminant source location. Problem (2.1.1-2.1.8) presents an infinite dimensional optimization problem subject to algebraic, ordinary differential, and partial differential constraints.

2.1.2 Solution Techniques

Solution approaches for dynamic optimization problems can be separated into two general classes, direct and indirect [39]. *Indirect* methods use a variational approach to write the first order optimality conditions as a boundary value problem. *Direct* methods, on the other hand, apply optimization tools directly to a discretized form of the differential model. Categories of direct methods differ in their treatment of model constraints and algorithms exist to solve the model constraints sequentially, simultaneously, or by some blend of the two.

Direct Sequential methods discretize the independent variables (control variables or inversion parameters) only. Given an initial guess for the profiles of these variables, standard solution techniques for the forward problem are used to evaluate the model at each iteration of the optimization and calculate values for the objective function. Derivative information is required with respect to the independent variables at each of the discretized points and can be calculated by various techniques, including sensitivity equations, adjoint equations, or finite differences. The optimization problem itself is in the space of the independent variables only and is small by comparison. However, calculation of derivative information can be computationally expensive.

Direct Simultaneous methods fully discretize all the unknown variables in the problem and solve the resulting system as a large scale optimization problem with algebraic constraints. Unlike the sequential technique, the forward problem is converged only once, along with the optimality conditions. Accurate analytical derivatives are often straightforward and efficient to calculate, and significant computational gains over the standard sequential approach are possible using this more intrusive technique. A review of direct and indirect techniques as applied to optimization of differentially constrained problems can be found in [68].

In previous work by [230], a direct sequential technique was used for problem (2.1.1-2.1.8), solving the small scale optimization problem with a standard successive quadratic programming tool. The model constraints representing the forward problem, (2.1.2-2.1.7), were solved at each iteration using the existing water network simulation package, EPANET [179] and, although the Lagrangian formulation used by EPANET provided efficient solution of the forward problem, it was not clear how to efficiently calculate derivatives so finite differences were used. The unreasonable computational cost of calculating finite differences across the model prevented complete discretization of the time dependent injection terms, $m_k(t)$. Instead, the time discretized profiles were reduced to single scalar parameters, allowing solutions for constant injection or initial condition contaminations only. Nevertheless, this approach demonstrated the potential for optimization techniques on this contamination source determination problem. In this current work, a direct simultaneous approach is used to overcome the difficulties encountered with the sequential method and solve the fully time dependent problem.

Straightforward application of the direct simultaneous method to problem (2.1.1-2.1.8) is not reasonable because of the size of the resulting nonlinear program. The pipe concentrations, as written, require a discretization in both time and space. (This is essentially the same as using an Eulerian technique to formulate the optimization constraints.) A simple discretization in both variables for all pipe concentrations will produce a nonlinear program that is too large for current optimization tools. The Lagrangian technique provides efficient simulation, but it is not obvious how to efficiently calculate derivative information, or even how to formulate this model in a simultaneous setting.

Fortunately, equations (2.1.5-2.1.7) only require pipe concentrations at the boundaries, and the objective (2.1.1) is dependent on node concentrations only. Spatial dependence is only introduced through the pipe expressions and a reformulation is sought that removes the need to discretize along the length of the pipes. In particular, a model reformulation should have the following properties:

- The water quality model is embedded within an optimization problem with additional constraints. As such, any reformulation should produce a straightforward mathematical representation.
- The resulting discretized model must be a reasonable size for current large scale nonlinear programming tools. This likely requires a reformulation of the pipe constraints that removes the need to discretize in space.
- Applications may require expressions for many nodes at many points in time. In particular, problem (2.1.1-2.1.8) requires expressions at all sensor nodes and all sample times. Any model preprocessing should be efficient for a large number of both source and output nodes.

The particle backtracking algorithm, proposed by [246], and extended by [210], characterizes the time delays associated with particular paths through the network. The algorithm tracks a particle, in reverse time, from an output node, back through the network, to source nodes. Calculating impact coefficients, the algorithm produces a set of algebraic equations describing the concentration of selected output nodes as a function of network sources and tank concentrations. Although this technique satisfies the first two criteria of our model reformulation, it does not scale well to a large number of output and source nodes. Instead an *origin tracking algorithm* is proposed that considers the time delays of each pipe individually and scales efficiently to large networks when considering all nodes in the network model.

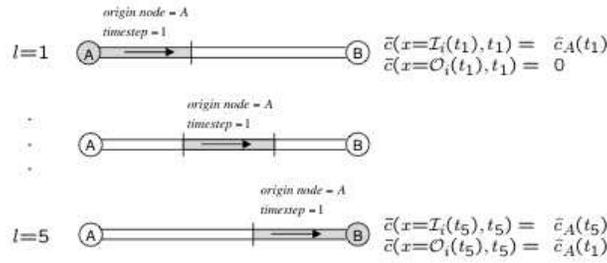


Figure 2.2. Origin Tracking Algorithm. This figure illustrates the flow of a single volume element through a pipe. Tracking these element allows us to determine relationships for the boundary concentrations at each point in time.

2.2 Origin Tracking Algorithm

The goal of the origin tracking algorithm is to reformulate the partial differential equations, (2.1.2-2.1.4), into a set of algebraic constraints that describe the time delays between pipe boundary concentrations and connected nodes. Based on the water quality method presented by [147], the origin tracking algorithm exploits the efficiency of the Lagrangian technique. While traditional Lagrangian methods track the actual concentration value of fluid elements as they move through the network, the origin tracking algorithm instead tracks the origin of each fluid element in the pipe. Any element of fluid in the pipe originated from one of three possible sources; either it entered from one of the two connected nodes, or it was present in the pipe initially. The algorithm tracks the origin of each fluid element as it enters the pipe. As flow conditions push the element past a pipe boundary, knowledge of the origin of the element is sufficient to write an expression for the concentration at that boundary.

To illustrate this idea, Figure 3.2 shows pipe i with flow conditions from left to right. Let Θ be the set of discretized time steps. For each timestep, $l \in \Theta$, expressions for the concentration at the pipe boundaries are needed. At timestep $l=1$, a fluid element is created at the left boundary and its originating node is recorded as “A” and its originating timestep as “1”. The concentration at this end of the pipe is equal to the immediate concentration from node A, that is $\bar{c}_i(x=I_i(t_1), t_1) = \hat{c}_A(t_1)$. The concentration at the right boundary of the pipe is equal to the initial loading in the pipe, $\bar{c}_i(x=O_i(t_1), t_1) = 0$. As time progresses, the element advances through the pipe. Assuming the element is pushed from the pipe at some timestep, say $l=5$, the pipe concentration at the left boundary is still equal to the current value from node A, $\bar{c}_i(x=I_i(t_5), t_5) = \hat{c}_A(t_5)$. The concentration at the right boundary, however, is now the same as the concentration from node A at timestep 1, $\bar{c}_i(x=O_i(t_5), t_5) = \hat{c}_A(t_1)$.

The linear relationships in Figure 3.2 describe the concentration of the pipe boundaries as a function of the concentrations of connected nodes. Choosing any appropriate time discretization, $l \in \Theta$, and letting τ_{il} be the originating timestep of the element bracketing the outlet of pipe i at timestep l , these expressions can be generalized as,

$$\bar{c}_i(x=I_i(t_l), t_l) = \bar{c}_i^I(t_l) = \hat{c}_{k_i(t_l)}(t_l) \quad \forall i \in \mathcal{P}, l \in \Theta, \quad (2.2.9)$$

$$\bar{c}_i(x=O_i(t_l), t_l) = \bar{c}_i^O(t_l) = \begin{cases} \hat{c}_{k_i(\tau_{il})}(\tau_{il}) & \text{if an element at outlet} \\ 0 & \text{if no element at outlet} \end{cases} \quad \forall i \in \mathcal{P}, l \in \Theta, \quad (2.2.10)$$

Equation (2.2.9) links the concentration at the pipe inlet boundary to the connected node and is uniquely determined by flow direction and network structure alone. Equation (2.2.10) describes the time delays for the pipe outlet boundary concentrations. This linear system, (2.2.9-2.2.10), then replaces equations (2.1.2-2.1.4) in the optimization problem, removing the spatial dependence.

The origin tracking algorithm formalizes the basic idea presented above, defining expressions for all network pipes under varying flow conditions. In developing this algorithm, the Lagrangian method of Liou and Kroon is simplified in two ways. First, it is assumed that the set of discrete points in time, Θ , is known and fixed. This allows us the flexibility to work with any discretization scheme selected for the differential tank equations. Second, the analysis is performed on a pipe by pipe basis, not on the network as a whole. The processing and memory requirements are then linear with the number of pipes and efficient for large networks. Although these simplifications introduce estimation errors in the time delays associated with paths through the network, they provide efficient scaling and favorable sparsity in the model.

A description of the origin tracking algorithm is shown below, where i is the current pipe, and l is the current timestep. For each volume element, the algorithm stores the originating node, the originating timestep, and the current position of each of the element boundaries within the pipe. At each iteration of the inner loop, the algorithm identifies the expressions (2.2.9) and (2.2.10) for pipe i and timestep l .

Algorithm 1. Origin Tracking Algorithm

Step 0. Initialize Overall Algorithm

- let $i = 0$, the first pipe in the network

Step 1. Initialize Pipe Iterations

- clear the list of tracked volume elements
- let $l = 0$
- write the expression: $\bar{c}_i(x=I_i(t_l), t=t_l) = \bar{c}_i(x=O_i(t_l), t=t_l) = 0$

Step 2. Advance Elements

- $\Delta x = u_i(t_l) \cdot (t_l - t_{l-1})$
- advance all currently tracked elements by Δx

Step 3. Add New Elements

- create a new element at the top or bottom of the list depending on flow direction
 - record the originating node as $k_i(t_l)$, the current “inlet” node
 - record the originating timestep as l

Step 4. Write Time Delay Expressions

- if stagnant flow,
 - write the expression: $\bar{c}(x=I_i(t_l), t=t_l) = (\text{expression from last timestep})$
 - write the expression: $\bar{c}(x=O_i(t_l), t=t_l) = (\text{expression from last timestep})$
- otherwise,
 - write the expression: $\bar{c}(x=I_i(t_l), t=t_l) = \hat{c}_{k_i(t_l)}(t_l)$
 - if there is no element bracketing a pipe boundary
 - write the expression: $\bar{c}(x=O_i(t_l), t=t_l) = 0$
 - otherwise,
 - read the data from the element bracketing the pipe boundary, store the originating node as “n” and the originating timestep as τ
 - write the expression: $\bar{c}(x=O_i(t_l), t=t_l) = \hat{c}_n(t_\tau)$

Step 5. Crop Elements

- remove any elements that have advanced outside the pipe boundary
- crop the length of any overhanging element

Step 6. Continue with the next timestep

- $l = l + 1$
- if $t_l \leq t_f$, goto Step 2.

Step 7. Continue with the next pipe

- $i = i + 1$
- if $i \in \mathcal{P}$, goto Step 1.

The origin tracking algorithm completely describes the linear system (2.2.9-2.2.10). The computational cost of the algorithm is linear in the number of pipes yet describes relationships for all pipe boundaries at all discretized points in time. Note that this algorithm requires flow data and network structure only and is performed with no prior knowledge of the source terms, $m_k(t)$. This resulting linear system provides a straightforward mathematical representation that characterizes the time delays and can easily be included in the discretized optimization problem, where junction mixing and storage tank dynamics are modeled.

2.3 Discretized Nonlinear Program

Using the reformulation of the pipe constraints from the origin tracking algorithm, problem (2.1.1-2.1.8) is discretized in time alone, producing a reasonably sized nonlinear program. Although the origin tracking algorithm allows the use of any discretization scheme, accuracy requirements on the pipe time delays tend to govern the stepsize and little benefit is gained from a higher order method. In this chapter, a simple backward Euler technique is used, however, more advanced techniques, like orthogonal collocation on finite elements [26], have been implemented with similar results.

Discretizing over $l \in \Theta$ with equally spaced intervals, $h=t_l - t_{l-1}$, problem (2.1.1-2.1.8) is written as,

$$\min_{m(t_r), \bar{c}^I(t_r), \bar{c}^O(t_r), \hat{c}(t_r)} \Psi_D = \sum_{r \in \Theta_s} \sum_{k \in \mathcal{N}_s^r} \frac{1}{2} w_k(t_r) (\hat{c}_k(t_r) - \hat{c}_k^*(t_r))^2 \quad (2.3.11)$$

s.t.

$$\bar{c}_i^I(t_l) = \hat{c}_{k_i(t_l)}(t_l) \quad \forall i \in \mathcal{P}, l \in \Theta, \quad (2.3.12)$$

$$\bar{c}_i^O(t_l) = \begin{cases} \hat{c}_{k_i(\tau_{il})}(\tau_{il}) & \text{if an element at outlet} \\ 0 & \text{if no element at outlet} \end{cases} \quad \forall i \in \mathcal{P}, l \in \Theta, \quad (2.3.13)$$

$$\hat{c}_k(t_l) - \frac{\left(\sum_{i \in \Gamma_k(t_l)} Q_i(t_l) \bar{c}_i(x=O_i, t_l) \right) + m_k(t_l)}{\left(\sum_{i \in \Gamma_k(t_l)} Q_i(t_l) \right) + Q_k^{ext}(t_l) + Q_k^{inj}(t_l)} = 0, \quad \forall k \in \mathcal{J}, l \in \Theta, l \neq 0, \quad (2.3.14)$$

$$\left[\frac{V(t_{l+1})}{h} + \left(\sum_{i \in \Gamma_k(t_{l+1})} Q_i(t_{l+1}) \right) Q_k^{ext}(t_{l+1}) + Q_k^{inj}(t_{l+1}) \right] \hat{c}_k(t_{l+1}) - \frac{V(t_{l+1})}{h} \hat{c}_k(t_l) - \left(\sum_{i \in \Gamma_k(t_{l+1})} Q_i(t_{l+1}) \bar{c}_i(x=O_i, t_{l+1}) \right) - m_k(t_{l+1}) = 0 \quad \forall k \in \mathcal{S}, l \in \Theta, l \neq 0, \quad (2.3.15)$$

$$\hat{c}_k(t=0) = 0, \quad \forall k \in \mathcal{S}, \quad (2.3.16)$$

$$m_k(t_l) \geq 0, \quad \forall k \in \mathcal{N}, l \in \Theta. \quad (2.3.17)$$

Difficulty arises when all flows to a junction are stagnant. In this case, equation (2.3.14) is replaced with $\hat{c}_k(t_l) = \hat{c}_k(t_{l-1})$ or $\hat{c}_k(t_0) = 0$, if required. With known flow information as inputs, problem (2.3.11-2.3.17) forms a convex quadratic programming problem. As such, any solution is a global minimum, but not

necessarily a unique minimizer.

To illustrate this, consider a simple two node network with flow from node A to node B and a sensor at node B only. Assume, given current flow rates, that the travel time between the nodes is one hour. If the sensor at node B registers concentration c^* at time t^* , it is clear that the contamination could have been injected at node B at time t^* . But it could also have been injected at node A at time t^* minus one hour. In fact, with the possibility of multiple injections allowed, any linear combination of these two injections summing to c^* is a possible solution, indicating infinitely many solutions to the optimization problem. In order to force a unique solution for this problem, a general regularization term is introduced in the objective.

Defining,

$$\bar{c}^I = \begin{bmatrix} \bar{c}_i(x=I_i(t_l), t_l) \\ \vdots \end{bmatrix}, \quad \bar{c}^O = \begin{bmatrix} \bar{c}_i(x=O_i(t_l), t_l) \\ \vdots \end{bmatrix}, \quad \begin{array}{l} \forall i \in \mathcal{P}, \\ \forall l \in \Theta. \end{array} \quad (2.3.18)$$

$$\hat{c} = \begin{bmatrix} \hat{c}_k(t_l) \\ \vdots \end{bmatrix}, \quad m = \begin{bmatrix} m_k(t_l) \\ \vdots \end{bmatrix}, \quad \begin{array}{l} \forall k \in \mathcal{N}, \\ \forall l \in \Theta. \end{array} \quad (2.3.19)$$

problem (2.3.11-2.3.17) is described as,

$$\min_{\bar{c}^I, \bar{c}^O, \hat{c}, m} \frac{1}{2} [\hat{c} - \hat{c}^*]^T W [\hat{c} - \hat{c}^*] + \rho \frac{1}{2} m^T R m \quad (2.3.20)$$

$$\text{s.t. } \bar{c}^I - P^I \hat{c} = 0, \quad (2.3.21)$$

$$\bar{c}^O - P^O \hat{c} = 0, \quad (2.3.22)$$

$$\bar{N} \bar{c} + \hat{N} \hat{c} + M m = 0, \quad (2.3.23)$$

$$m \geq 0, \quad (2.3.24)$$

where P^I and P^O will be matrices of zeros and ones with, at most, a single nonzero entry per row. The regularization term, $\frac{\rho}{2} \sum_{k \in \mathcal{N}} m^T R m$, is included in the objective to force a unique solution to the quadratic program, where R is a positive definite regularization matrix. The exact form of R will depend on the type of regularization used, and for numerical tests presented here, $R = I$. Any positive value for ρ is sufficient to guarantee a unique solution to the optimization problem, but this solution will approximate a linear combination of possible solutions. The junction and tank equations are grouped together and $[\bar{N} \hat{N} M]$ is the Jacobian of the discretized node equations, (2.3.14-2.3.16). W is the diagonal matrix of flow based weights for the concentration errors and will only have nonzero entries corresponding to sensor nodes at sample times. These weights are all positive, so, in general, W is a positive semidefinite matrix. In the unusual circumstance where there is a sensor at every node and data is sampled at every timestep, W will be nonsingular and positive definite.

This regularized formulation is a well-posed quadratic program that can be solved with existing optimization tools. In the next section the effectiveness of this formulation is demonstrated on some numerical examples.

2.4 Numerical Results

A software tool has been developed to formulate the discretized nonlinear program. First, a transient contamination scenario is simulated using EPANET [179]. After specifying the sensor configuration, time horizon, and integration time, the software tool reads the network structure, dynamic hydraulic information, and concentration results (as sensor measurements) from EPANET and formulates the discretized nonlinear program in AMPL [95]. AMPL is a modeling language for optimization that provides both first and second order derivative information using automatic differentiation. The nonlinear program is then solved with IPOPT, a nonlinear interior point optimization package [237]. Further description of this code can be obtained from www.coin-or.org and associated links. Interior point methods are appropriate for problems with many bound constraints (introduced by equation (2.3.17)). Although problem (2.3.11-2.3.17) forms a quadratic program, the general nonlinear optimization package, IPOPT, performs well and allows for future inclusion of nonlinear constraints (i.e. reaction terms). To verify the effectiveness of the formulation, the solution profile for the injection terms, m , is compared against the actual injection profiles used in the simulated contamination.

All examples have been formulated using a 5 minute integration timestep and a 5 minute sample interval. Since the sampling interval is equivalent to the integration timestep, the regularization matrix is set to $R = I$, approximating $\sum_{k \in \mathcal{N}} \int_0^{t_f} m_k(t)^2 dt$. For a sampling interval longer than the timestep, regularization of the form, $\sum_{k \in \mathcal{N}} \int_0^{t_f} \left(\frac{dm_k(t)}{dt} \right)^2 dt$, can be used to impose a smoothness condition on the injection profiles between sampling intervals. The regularization parameter, ρ , is set to $1 \cdot 10^{-4}$. Fortunately, the numerical results are reasonable over a wide range of values for ρ and no specific tuning is required to find solutions for the different injection locations studied.

2.4.1 Example 1. Symmetric Grid Network

A simple grid network is shown in Figure 4.1, where flows are in the directions indicated, consumption demands exist at the boundary nodes only, and the single known external water source is a reservoir at node 26. The time delays between the nodes range from approximately half an hour to five hours. The shaded nodes indicate installed sensors, and, with sensors at every second node, this example is symmetric about the diagonal from node 1 to node 25.

As a contamination scenario, a 30 minute long injection at time $t=30$ minutes is simulated from node 13. The optimization problem is formulated at hour 4, considering only the previous 4 hours of sensor and flow data. Figure 2.4 shows the solution profiles for all injection terms with significant values. Even though the regularized problem is guaranteed to have a single unique solution, this single solution indicates multiple possible injection scenarios. One possibility is the exact simulated contamination, a 30 minute injection from node 13 at $t=30$ minutes. However, the solution also indicates other possible injection locations, including nodes 8, 9, 12, and 17. A second possibility is a simultaneous contamination at nodes 8 and 12 at just over 2 hours. Although it is not available from the solution profiles in Figure 2.4, the network symmetry shows that this must be a simultaneous contamination at both nodes. Simultaneous injections may seem unlikely, but the formulation makes no restriction on the number of injections or the injection time. With the specified network structure, sensor configuration, and flow direction, it is impossible to

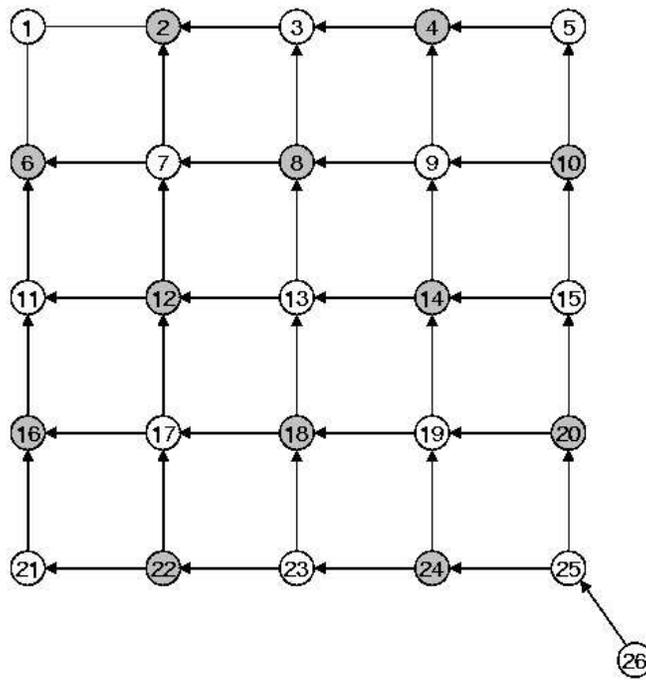


Figure 2.3. Grid Network Example A small symmetric grid network with sensors installed at every second node, indicated by the shading.

distinguish the actual injection node 13, from injections at nodes 8 and 12, and these two possibilities are expected. A third, more surprising, possibility is a simultaneous contamination at nodes 9 and 17 at approximately 5 minutes.

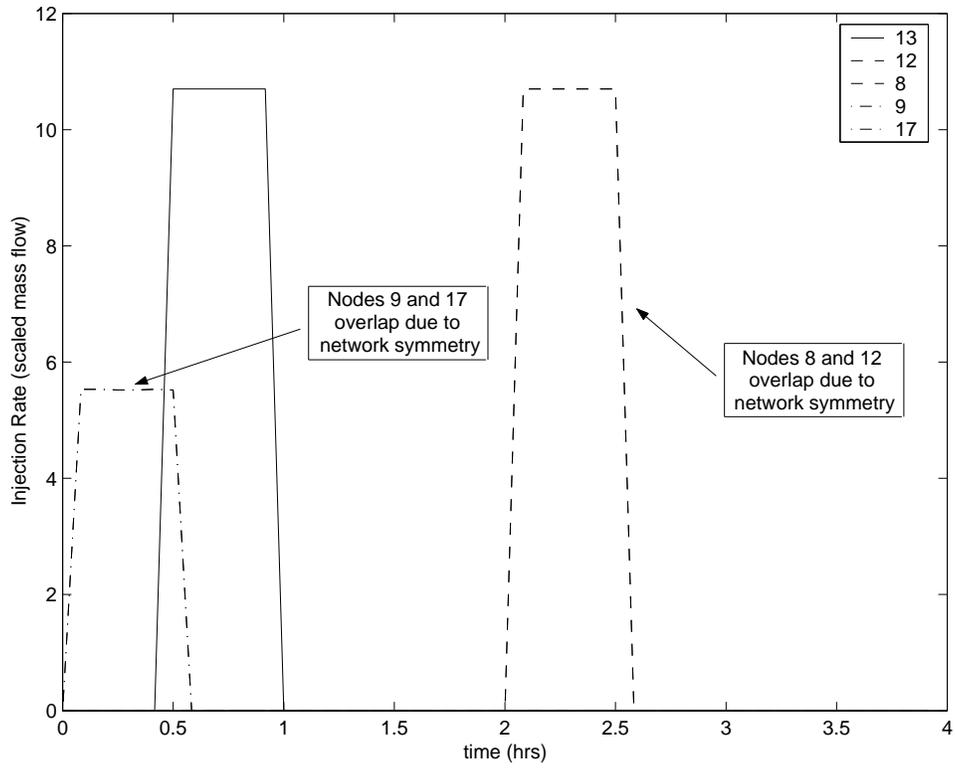


Figure 2.4. Grid Network Solution 1. Solution for an injection at node 13 using a 4 hour time horizon.

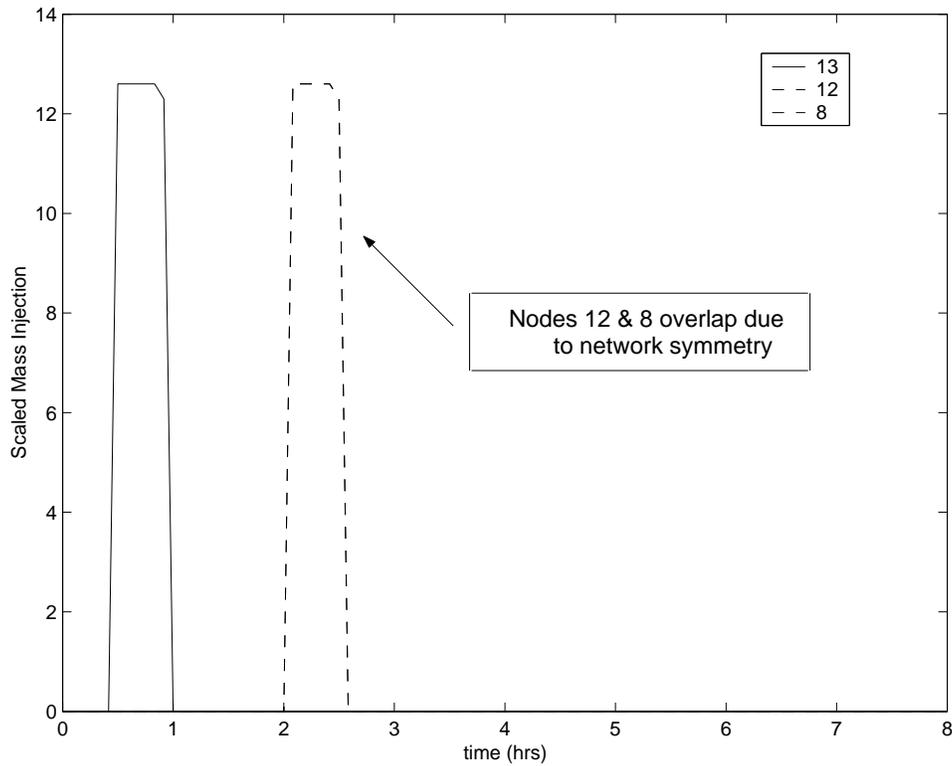


Figure 2.5. Grid Network Solution 2. Solution for an injection at node 13 with an 8 hour time horizon. Note that nodes 9 and 17 are now excluded as possible injection locations.

A simultaneous injection at nodes 9 and 17 could also produce the measured concentrations at sensor nodes 8 and 12, and it would then be expected that sensors at nodes 4 and 16 would also measure contaminant. On the other hand, zero measurements at nodes 4 and 16, over a sufficiently long period of time, would rule out injections at nodes 9 and 17; this requires an estimation time horizon long enough to confirm these effects. Here, the time delays associated with links 17 - 16 and 9 - 4 are just over 4 hours and, hence, an optimization problem formulated with only 4 hours of previous measurements has insufficient information to observe concentration measurements at nodes 4 and 16.

Using the same simulated injection, but running the optimization at hour 8, considering the previous 8 hours of simulation data, produces the solution shown in Figure 2.5. Here, fluid flow through nodes 9 and 17 are now observed by the sensors at nodes 4 and 16, and, since these sensors do not encounter any contaminant, nodes 9 and 17 have now been excluded as possible injection locations. This simple example shows the importance of null concentration measurements and sufficient time horizons.

2.4.2 Example 2. Real Municipal Network Model

The formulation has also been tested on a model of a real municipal water network with 469 nodes (including 4 storage tanks) and 635 links. Four injection locations are considered, shown in Figure 2.6. Location A is along a major feed line to the network, while locations B and C are interior to the network. Contaminant spreads readily through the network from these three locations. Location D, on the other hand, exhibits minimal spreading of contaminant through the network. For each of the four locations, a 16 hour time segment is simulated, using a 30 minute injection at hour 8. An attack or accidental injection would likely be significantly longer than 30 minutes, but this short injection time illustrates the algorithm performance on a difficult injection scenario.

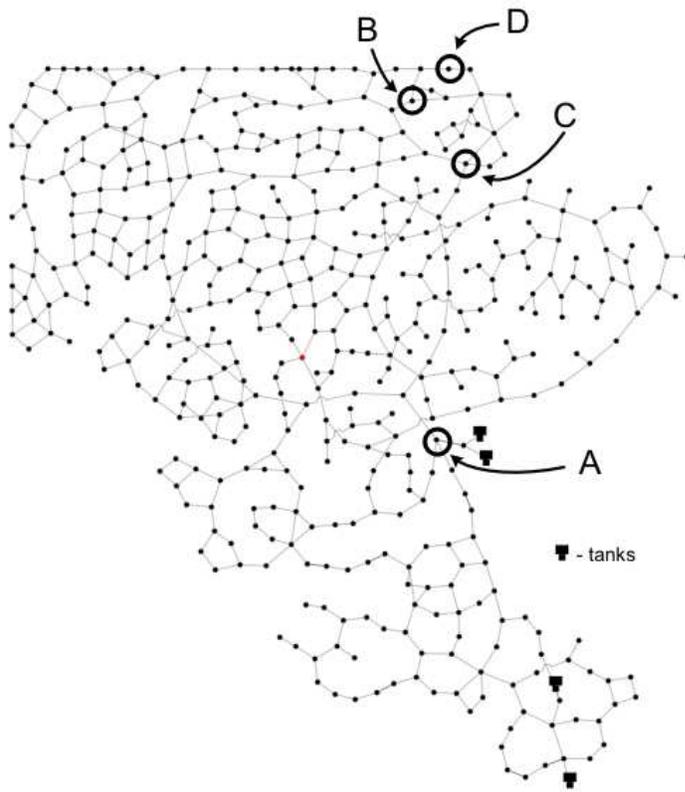


Figure 2.6. Municipal Water Network Injection Locations. Four simulated injection locations, A through D, are shown in the diagram of the network model.

Visually inspecting each of the profiles, like those shown in Figures 2.4 and 2.5 is impractical for a large number of test examples. Instead, a single scalar measure of the effectiveness is used. After running the optimization the solution profiles for all the injection terms are integrated, $\gamma_k = \int_{t=0}^{t_f} m_k(t) dt$, and the nodes are sorted in descending order of γ_k . This provides a ranking of each node in the network, defined by the injection profile from the optimization solution. It is desirable for the actual injection node to be ranked first (i.e. the optimization solution estimates the most prominent injection at the node where the actual injection occurred). Unfortunately, because of the sparse sensor grid and non-uniqueness, this may not be possible.

While measuring the effectiveness of the formulation, it is important to consider two key indicators.

1. **The Number of Installed Sensors:** A good algorithm should identify injection locations using as few sensors as possible. The placement of installed sensors will likely be critical in reducing this number.
2. **Identification Time:** The time required for the contaminant to reach installed sensors, plus the additional time required to accrue enough information for the optimization to be effective should be short. Note, this does not refer to the time for an optimization to execute, but rather constitutes the total elapsed time, following a contamination, before the system can make a reasonable estimate of the injection location.

Figure 2.7 shows the results of numerical tests for each of the four injection scenarios. The y-axis shows the number of randomly placed contaminant sensors. The same random seed was used for each sensor configuration (e.g. when the number of randomly placed sensors are increased from 70 to 80, only 10 new sensors are placed and the previous 70 sensors remain at their original location). All test runs were performed using an 8 hour time horizon. The x-axis shows the time, following the simulated injection, at which the optimization was formulated and run. For example, the optimization test result at hour 2 was formulated using a time horizon beginning 6 hours prior to the injection and finishing 2 hours following the injection. This gives us an example of the effectiveness of the algorithm in a real scenario, 2 hours following an injection. With the 5 minute integration stepsize and 8 hour time horizon, each optimization problem is a large scale nonlinear program with approximately 210,000 variables, 165,000 equality constraints, and 45,000 inequality constraints and solves in less than 2 minutes on a 2.2 GHz Pentium 4 machine. Figure 2.7 shows the results of these tests (over 1000), plotting the rank of the solution profile for the simulated injection node, shown by the shading scale to the right.

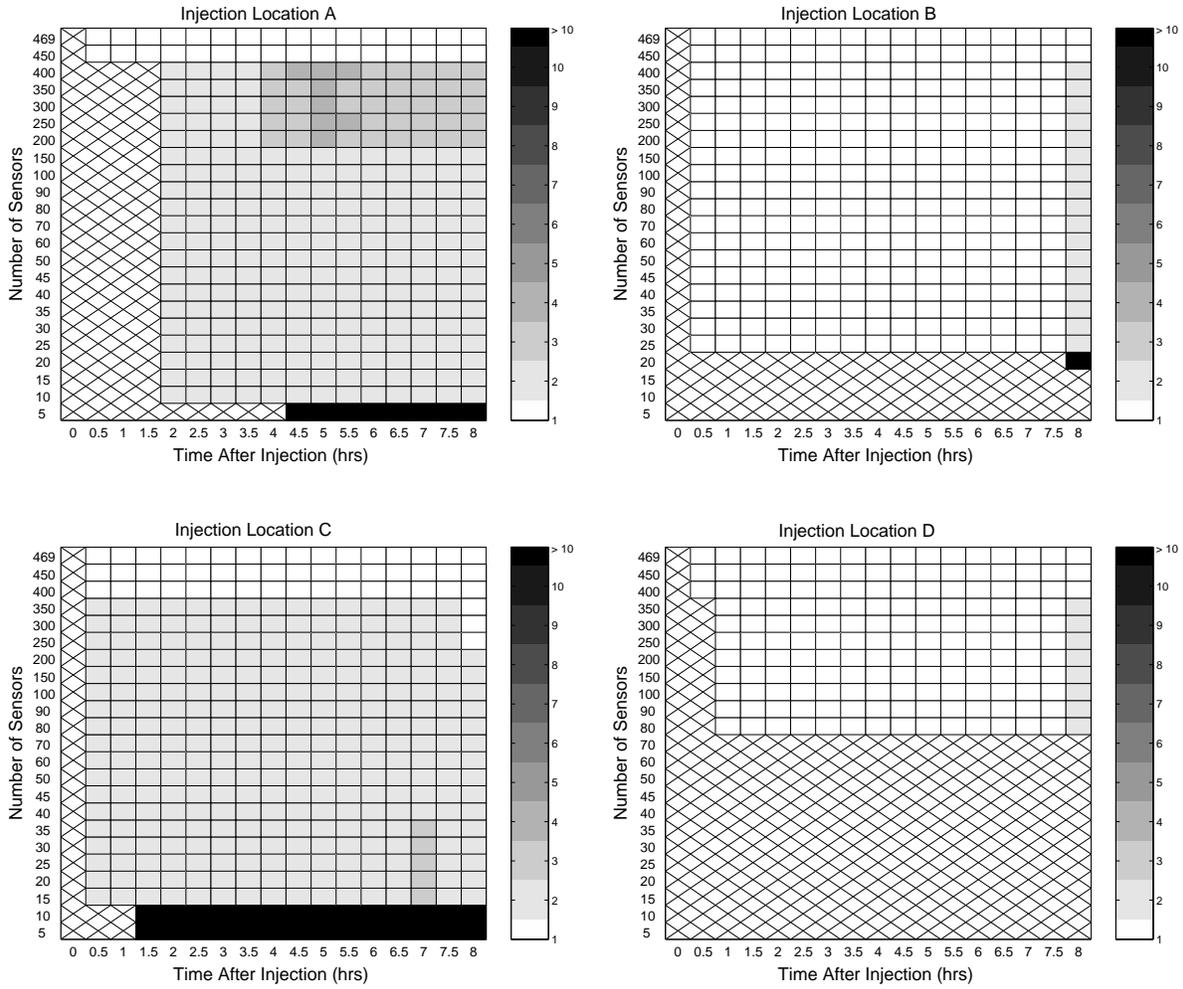


Figure 2.7. Rank of Optimization Solution for Simulated Injection Node. This figure illustrates the effectiveness of the formulation in determining the correct injection node for simulated injections A through D. The rank of the injection node is shown with the shading to the right of each contour plot. A low ranking indicates that the formulation has been effective at identifying the injection location.

The hashed regions indicate tests where none of the installed sensors had yet registered any contaminant and thus there is no information to run the optimization. At $t = 0$ hours, the time of the simulated injection, no sensors have measured contaminant, and no result is expected for this entire column. In the top row of each contour plot, sensors are installed at every node, and a ranking of 1 is observed for each of the simulated injection scenarios, as expected.

The formulation is very effective for locations A through C, where it determines the injection location with a short identification time and few sensors. The effectiveness of the sensor grid is influenced dramatically by the flow patterns at the injection location. The formulation is less effective for location D. The flow conditions from injection location D do not cause significant spreading of the contaminant through the network, and a higher requirement on the number of installed sensors is expected. Nevertheless, once there are enough sensors to detect the contamination, the formulation is extremely effective at determining the correct injection location. Optimal sensor placement should reduce the number of required sensors.

2.5 Conclusions and Future Work

In the event of an accidental or intentional contamination of a municipal drinking water network, it is important to identify the time and location of contamination sources. This chapter has presented an optimization formulation for this inverse problem, estimating time dependent contamination injections for every network node.

Some areas of future work include:

- The results for location D illustrate the importance of determining optimal sensor locations. It is also desirable to choose sensor locations that reduce the non-uniqueness in solutions. Current work in this area [31, 32, 163] is useful for detection systems, but the objective measures used may not be appropriate for the contamination source determination problem. It may be important to reduce the number of possible injection locations to make a guarantee of uniqueness for optimal sensor placement.
- No analysis has yet been done to test the reliability of this formulation in the face of sensor failure or noise in flow rates and sensor measurements. An actual implementation would likely need to include a robust estimation phase that could remove potential outliers in the measurement data and quantify the uncertainties associated with the data.
- Many contaminants will experience decay as they propagate through the network. Reaction terms can be included in pipe and tank constraints. Of course, this now requires that the sensors correctly identify the contamination species to include the correct rate expressions.
- It is necessary to examine the performance of this formulation for much larger community networks. Since the time delays associated with large water networks are often large, it is reasonable to assume that the contamination source determination problem could be formulated on a subset of the entire network

Table 2.1. Notation for Continuous Formulation: Equations (2.1.1-2.1.8)

Variable	Comments
$\mathcal{P}, \mathcal{J}, \mathcal{S}$	the complete sets of all pipes, junctions, and storage tanks
\mathcal{N}	the complete set of all nodes (i.e. $\mathcal{N} = \mathcal{J} \cup \mathcal{S}$)
$t \in [0, t_f]$	time
$x \geq 0$	displacement along a pipe
$\tilde{c}_i(x, t), i \in \mathcal{P}$	contaminant concentration in pipe i at displacement x and time t
$\hat{c}_k(t), k \in \mathcal{N}$	contaminant concentration of node k at time t
$m_k(t), k \in \mathcal{N}$	unknown contamination mass flow rate
$\mathcal{N}_s \subseteq \mathcal{N}, \Theta_s$	the set of nodes with installed sensors and the set of all sample times
$\hat{c}_k^*(t), k \in \mathcal{N}_s$	measured contaminant concentrations, these values will not be known continuously in time, but rather at discrete sampling points, $t_r, r \in \Theta_s$
$w_k(t), k \in \mathcal{N}_s$	time dependent weight for the concentration errors. A flow based weighting function is used, shifting the error measure from a concentration basis to a mass basis
$\Gamma_k(t), k \in \mathcal{N}$	the set of all pipes flowing into node k at time t
$I_i(t), O_i(t), i \in \mathcal{P}$	displacement along pipe i where fluid is entering and leaving the pipe respectively, these designations are time dependent and change with the flow direction
$k_i(t), i \in \mathcal{P}$	the index of the node connected at the inlet of pipe i , this designation is time dependent and changes with the flow direction
$u_i(t)$	known fluid velocity in pipe i
$Q_i(t), i \in \mathcal{P}$	the known volumetric flow rate in pipe i at time t
$Q_k^{ext}(t), k \in \mathcal{N}$	the volumetric flow rate for known external sources (e.g. reservoir flow)
$Q_k^{inj}(t), k \in \mathcal{N}$	the volumetric flow rate of the unknown contaminant mass injection, $m_k(t)$, in practice this value will not be known and they are set to a small quantity relative to other network flow rates
$V_k(t), k \in \mathcal{S}$	the volume in tank k at time t
$\delta(t)$	Dirac delta function

In conclusion, the origin tracking algorithm successfully reformulates the partial differential pipe expressions into a set of algebraic constraints with time delays, removing the need to discretize in space. The cost of this algorithm is linear in the number of pipes and the reformulation time is a small portion of the overall solution time. This reformulation allows the use of a direct simultaneous technique for discretizing the continuous problem. With the network model studied, solution times for the optimization were under two minutes on a 2.2 GHz machine, making it suitable for a real-time setting, a result that was not possible with the direct sequential technique. The effectiveness of the source determination formulation has been demonstrated on a real municipal network model and it produces a reasonable estimate of the injection location with a limited number of network sensors.

Chapter 3

Real-time, Large Scale Optimization of Water Network Systems using a Subdomain Approach

Carl D. Laird (Carnegie Mellon University), Lorenz T. Biegler (Carnegie Mellon University), Bart G. van Bloemen Waanders

3.1 Background

Certain classes of dynamic network problems can be modeled by a set of hyperbolic partial differential equations describing behavior along network edges and a set of differential and algebraic equations describing behavior at network nodes. In this chapter, we demonstrate real-time performance for optimization problems in drinking water networks. While optimization problems subject to partial differential, differential, and algebraic equations can be solved with a variety of techniques, efficient solutions are difficult for large network problems with many degrees of freedom and variable bounds. Sequential optimization strategies can be inefficient for this problem due to the high cost of computing derivatives with respect to many degrees of freedom [230]. Simultaneous techniques can be more efficient, but are difficult because of the need to solve a large nonlinear program; a program that may be too large for current solver. This study describes a dynamic optimization formulation for estimating contaminant sources in drinking water networks, given concentration measurements at various network nodes. We achieve real-time performance by combining an efficient large-scale nonlinear programming algorithm with two problem reduction techniques. D'Alembert's principle can be applied to the partial differential equations governing behavior along the network edges (distribution pipes). This allows us to approximate the time-delay relationships between network nodes, removing the need to discretize along the length of the pipes. The efficiency of this approach alone, however, is still dependent on the size of the network and does not scale indefinitely to larger network models. We further reduce the problem size with a subdomain approach and solve smaller inversion problems using a geographic window around the area of

contamination. We illustrate the effectiveness of this overall approach and these reduction techniques on an actual metropolitan water network model.

3.1.1 Contamination Source Determination

The vulnerability of municipal drinking water networks to intentional and accidental contaminations requires efficient systematic protection measures. Distribution networks cover a very large area, including private locations, making it impossible to use physical security alone to prevent drinking water contamination. If a contamination occurs, it is important to quickly identify the magnitude, time, and location of the contamination source to stop the spread of contaminant and devise a control strategy. Here, we describe an approach for identifying contamination sources in water networks using concentration measurements from a set of installed contaminant sensors. Assuming that the cost of these sensors is high, contaminant source must be effectively identified using few network sensors. It is also important to solve the inversion problem as quickly as possible. Since the inversion may be one component of an overall control strategy, it must be efficient and solve in real-time for large water distribution networks.

Using known network flowrates and concentration measurements, we formulate a least squares dynamic optimization problem subject to the constraints of the water quality model and unknown contaminant injection sources. Solving this dynamic optimization problem for the unknown, time dependent source terms identifies both the time and location of possible contaminant injections.

A number of different techniques can be used to solve the dynamic optimization problem, progressing from minimal interfacing with an existing simulator to a more intrusive fully simultaneous technique. Although the fully simultaneous discretization technique produces a very large nonlinear program, in many cases it can be significantly more efficient than sequential techniques [230, 229, 68]. Because of the large number of degrees of freedom and the high cost of calculating derivatives for the time discretized injection profiles, sequential techniques are not appropriate for the solution of the time dependent network source determination problem. A straightforward application of the simultaneous approach is not possible since a naive discretization of the network model produces a nonlinear program that is too large for general nonlinear programming tools. Instead, we recently developed a reformulation of the pipe constraints, allowing the use of a simultaneous technique for solving the time dependent problem [134]. In that study, efficient numerical results were presented for a network model of approximately 500 nodes, with solution times under two minutes wall clock time for all test scenarios. However, this approach performs the optimization over the space of the entire network and does not scale to significantly larger networks. The bottleneck in addressing very large systems is the need to find solutions for large linear systems at each iteration of the optimization algorithm. Since processing and memory requirements are not linear in the number of variables, increasing the problem size can cause a dramatic increase in solution times. Furthermore, with direct factorization techniques, memory limitations place restrictions on the overall size of the linear system, thus restricting the number of network nodes and the number of timesteps that can be considered.

To consider larger metropolitan water networks, we apply a network subdomain approach to reduce the problem size further. Since the time delays associated with paths through water networks are typically long, a rapid source inversion method should identify contamination sources before the contaminant

spreads significantly through the network. Therefore, the optimization problem can be formulated on a subset of the overall network. The required subdomain size is not dependent on the size of the entire network, so this approach can deal with large municipal drinking water networks very efficiently. Moreover, the location and size of the subdomain need not be fixed and tracking the spread of the contaminant and implementing a control strategy allows straightforward extensions of this approach to moving subdomains. While the primary purpose of the subdomain technique is to reduce computational effort, evidence suggests that this technique may also improve solution quality by allowing a finer discretization in time. To demonstrate this subdomain approach we consider the source inversion problem for a network model that describes the distribution system for a large city. We examine the quality of the inversion solutions under varying subdomain sizes. Our results indicate that, shortly following an injection, this subdomain approach is an effective problem reduction technique, and that the overall algorithm is an efficient method for contaminant source detection.

Section 3.2 presents the formulation of the contamination source determination problem and describes solution techniques for this problem. In particular, we consider an origin tracking algorithm that converts the PDE-constrained problem into a differential-algebraic optimization problem. This problem can then be discretized in time and solved with large-scale nonlinear programming methods. We then develop an alternative formulation that applies to any selected subdomain of the network. Section 3.3 applies the subdomain approach on a real municipal water network model. This approach considers hydraulics and sensor measurements from the entire network, selects a subdomain for the optimization, and solves the optimization problem in the space of these selected network nodes only. A case study illustrates the effectiveness of this approach and the effect of subdomain size on computational effort and solution quality. Section 3.4 then concludes the chapter and offers a number of directions for future work.

3.2 Dynamic Optimization Formulation

Before developing the optimization formulation for the subdomain approach, we first describe the formulation for the full network model as given in [134]. Water distribution systems can be represented as networks with nodes and edges. The nodes represent zero volume junctions, storage tanks, or reservoirs (network hydraulic sources). The network edges represent pipes, valves, or pumps. We assume that all time dependent pipe flows in the network can be determined in advance either from measurements or numerical simulation and these become defined inputs to the water quality model [147, 182, 181, 246, 210]. Generally, low flow residential areas are collapsed into single network nodes and only larger distribution pipes are modeled. Because the flow in these pipes have high Reynolds number, we assume plug flow behavior in all pipe segments. Also, contaminant decay through reactions within the pipe segments is assumed to be negligible, although the model could be formulated with first order decay kinetics without affecting the model properties.

The water quality model is developed using \mathcal{P} , \mathcal{J} and \mathcal{S} to refer to the index sets of all pipes, junctions, and storage tanks, respectively. In addition, $c_i(x, t), i \in \mathcal{P}$ represent the concentration in the pipes, and $\hat{c}_k(t), k \in \mathcal{N}$ represent the concentration at the nodes, where $\mathcal{N} = \mathcal{J} \cup \mathcal{S}$ is the index set of all nodes including junctions and storage tanks. We make no assumption about the time of the injection or the duration and introduce unknown, time dependent mass injection terms, $m_k(t)$, for each node, $k \in \mathcal{N}$.

Here, $t \in [0, t_f]$ defines the time horizon considered, and $x \geq 0$ is the displacement along a pipe. The notation for designating pipe boundaries is based on flow direction, as shown in Figure 3.1. Here $x=I_i(t)$ refers to the boundary where fluid is entering pipe i and $x=O_i(t)$ refers to the boundary where fluid is leaving pipe i . The index $k_i(t), i \in \mathcal{P}$ refers to the node connected at the inlet of pipe i at time t , again, determined by the flow direction. The source determination problem can therefore be stated as:

$$\min_{\substack{m_k(t), c_i(x,t), \hat{c}_k(t) \\ \forall i \in \mathcal{P}, k \in \mathcal{N}}} \Psi = \int_0^{t_f} \sum_{r \in \Theta^*} \sum_{k \in \mathcal{N}^*} w_k(t) (\hat{c}_k(t) - \hat{c}_k^*(t))^2 \delta(t - t_r) + \rho \sum_{k \in \mathcal{N}} m_k(t)^2 dt \quad (3.2.1)$$

s.t.

$$\left. \begin{aligned} \frac{\partial c_i(x,t)}{\partial t} + u_i(t) \frac{\partial c_i(x,t)}{\partial x} &= 0, \\ c_i(x=I_i(t), t) &= \hat{c}_{k_i(t)}(t), \quad c_i(x, 0) = 0 \end{aligned} \right\} \quad \forall i \in \mathcal{P}$$

$$\hat{c}_k(t) = \frac{\left(\sum_{i \in \Gamma_k(t)} Q_i(t) c_i(x=O_i(t), t) \right) + m_k(t)}{\left(\sum_{i \in \Gamma_k(t)} Q_i(t) \right) + Q_k^{ext}(t) + Q_k^{inj}(t)} \quad \forall k \in \mathcal{J} \quad (3.2.2)$$

$$V_k(t) \frac{d\hat{c}_k(t)}{dt} = \left(\sum_{i \in \Gamma_k(t)} Q_i(t) c_i(x=O_i(t), t) \right) + m_k(t) - \left[\left(\sum_{i \in \Gamma_k(t)} Q_i(t) \right) + Q_k^{ext}(t) + Q_k^{inj}(t) \right] \hat{c}_k(t), \quad \hat{c}_k(0) = 0 \quad \forall k \in \mathcal{S} \quad (3.2.3)$$

$$m_k(t) \geq 0 \quad \forall k \in \mathcal{N} \quad (3.2.4)$$

Letting Θ be the complete set of discretized points in time, $\Theta^* \subseteq \Theta$ is the index set of sample times for the measured concentration data and $\mathcal{N}^* \subseteq \mathcal{N}$ is the index set of nodes with installed sensors. The objective function (3.2.1) minimizes the difference between measured nodal concentration data $\hat{c}_k^*(t), k \in \mathcal{N}^*$ and corresponding values calculated from the network model, weighted by a flow based function, $w_k(t), k \in \mathcal{N}^*$. The measured concentrations are known at the selected sampling times, $t_r, r \in \Theta^*$ and $\delta(t)$ is the Dirac delta function. The final term in (3.2.1) regularizes the objective and forces a unique solution to the inversion problem, where ρ is the regularization parameter.

Equation (3.2.2) models the network pipes, equation (3.2.2) models the junctions, equation (3.2.3) models the tanks, and (3.2.4) bounds the injection terms to be nonnegative. Here $\Gamma_k(t), k \in \mathcal{N}$ is the set of all pipes flowing into node k at time t . This designation is time dependent and changes with the flow direction. In addition, $u_i(t)$ is the known fluid velocity in pipe i , $Q_i(t), i \in \mathcal{P}$ is the known volumetric flow rate in pipe i at time t , $Q_k^{ext}(t), k \in \mathcal{N}$ is the volumetric flow rate for known external sources (e.g. reservoir flow) and $Q_k^{inj}(t), k \in \mathcal{N}$ is the volumetric flow rate of the unknown contaminant mass injection, $m_k(t)$. In practice the unknown value of Q_k^{inj} is set to a negligible quantity relative to other network flow rates. Finally, $V_k(t), k \in \mathcal{S}$ is the volume in tank k at time t .

Since the network flows are specified inputs, the only unknowns are the pipe and node concentrations, $c_i(x,t)$, and $\hat{c}_k(t)$ respectively, and the mass injections, $m_k(t)$. The traditional forward problem is that of



Figure 3.1. Link Boundary Design. $I_i(t)$ indicates the inlet of the link, based on the current flow direction, while $O_i(t)$ indicates the outlet of the link. The index $k_i(t)$ always refers to the node connected at the inlet.

setting the mass injections, $m_k(t)$, and solving (using (3.2.2-3.2.3)) for the network concentrations. In the inverse problem (3.2.1-3.2.4), the goal is to solve for the values of the injection terms, $m_k(t)$. Significantly positive values for the solution at a particular node indicate a potential contaminant source location. The optimization solution also gives the complete time profiles for the network concentrations. Problem (3.2.1-3.2.4) presents an infinite dimensional optimization problem subject to algebraic, ordinary differential, and partial differential constraints.

Problems of this type can be solved with a *direct simultaneous* method, where we fully discretize all the state and control variables in the problem and solve the resulting system as a large scale optimization problem with algebraic constraints. The forward problem is converged only once, along with the optimality conditions. Accurate analytical derivatives are often straightforward and efficient to calculate, and significant computational gains over alternative approaches are possible using this more intrusive technique. A review of this and other methods as applied to optimization of differential-algebraic optimization problems can be found in [68]. However, straightforward application of the direct simultaneous method to problem (3.2.1-3.2.4) leads to a discretization in both time and space and a nonlinear program that is too large for current optimization tools. Real-time performance is a requirement for this application and many others, and while a Lagrangian viewpoint [147] of the pipe equations (3.2.2) provides a much more efficient simulation, a reformulation of the pipe equations is required within a simultaneous setting.

In the remainder of this section, we discuss two problem reduction techniques. The *origin tracking algorithm* [134] preprocesses the network flow patterns, derives a delay-differential system of equations and approximates the time delays. This reformulation removes the need to discretize in x . In addition, we briefly sketch a *network subdomain approach*. If the spread of contamination is slow, the relevant region is a small subdomain of the entire network and a much smaller nonlinear programming problem can be considered.

3.2.1 Origin Tracking Algorithm

The *origin tracking algorithm* approach has a physical interpretation for the water network inversion problem [134], but it is general in nature and applies to any problem where d'Alembert's principle can be applied to the partial differential equations governing behavior along network edges. Consider the

discretization of the node concentrations, $\hat{c}_k(t)$ and the injection terms $m_k(t)$ in time $t_l, l \in \Theta$. To reformulate the pipe equations, we use the known flow profiles to calculate the time-delay relationships between pipe boundaries for all pipes and each time discretization. Specifically, we need to develop expressions for $c_i(x = O_i(t_l), t_l)$ given $c_i(x = I_i(t_l), t_l)$ for all $i \in \mathcal{P}$ and all points $t_l, l \in \Theta$. Assuming that the form of $u_i(t)$ is known, the relationship for the inlet pipe boundary is straightforward to write from the pipe boundary conditions as:

$$c_i(x=I_i(t_l), t_l) = \hat{c}_{k_i(t_l)}(t_l) \quad (3.2.5)$$

for non-zero $u_i(t)$.

Writing the relationship for $c(x=O(t_l), t_l)$ requires further consideration of the pipe segment equations (3.2.2). Following elementary solutions of one-dimensional wave equations, we apply d'Alembert's principle and write:

$$\frac{dc_i}{dt} = \frac{\partial c_i(x, t)}{\partial t} + \left(\frac{dx}{dt} \right) \frac{\partial c_i(x, t)}{\partial x} \quad (3.2.6)$$

$$= \frac{\partial c_i(x, t)}{\partial t} + u_i(t) \frac{\partial c_i(x, t)}{\partial x} = 0 \quad (3.2.7)$$

for each pipe segment, $i \in \mathcal{P}$. Equating (3.2.6) and (3.2.7) yields:

$$\frac{dc_i}{dt} = 0, \quad \frac{dx}{dt} = u_i(t) \quad (3.2.8)$$

and from the boundary conditions, we have:

$$c_i(x, t) = c_i(I_i(\bar{t}), \bar{t}) = \hat{c}_{k_i(\bar{t})}(\bar{t}) \quad (3.2.9)$$

for values of x, t and \bar{t} that satisfy:

$$x(t) = I_i(\bar{t}) + \int_{\bar{t}}^t u_i(\tau) d\tau \quad (3.2.10)$$

Since the form of $u_i(t)$ is known, we can write this equation for the outlet boundary as:

$$O_i(t_l) = I_i(\bar{t}) + \int_{\bar{t}}^{t_l} u_i(\tau) d\tau, \quad (3.2.11)$$

where the only unknown is \bar{t} . Solving this equation for \bar{t} then allows us to write an expression for the outlet concentration

$$c_i(O_i(t_l), t_l) = c_i(I_i(\bar{t}), \bar{t}). \quad (3.2.12)$$

Of course, it is unlikely that \bar{t} will be in the set of discretized points $t_l, l \in \Theta$. Therefore, we accept an approximation to the time delay and exchange \bar{t} with some nearby discretized point in time, $t_{\bar{l}}$,

$$c_i(O_i(t_l), t_l) = c_i(I_i(t_{\bar{l}}), t_{\bar{l}}). \quad (3.2.13)$$

The method for selecting this nearby point is problem specific and for physical reasons, we select the nearest point smaller than \bar{t} (i.e. $t_{\bar{l}} \leq \bar{t} < t_{\bar{l}+1}$).

In the drinking water network source inversion problem, $u_i(t)$ represents the known pipe velocities and is assumed piecewise constant over each interval $t_l < t \leq t_{l+1}$. We write (3.2.11) as,

$$O_i(t_l) = I_i(\bar{t}) + \int_{\bar{t}}^{t_l} u_i(\tau) d\tau \quad (3.2.14)$$

$$= I(\bar{t}) + u_i(\bar{t}) \cdot (t_{\bar{l}+1} - \bar{t}) + \sum_{j=\bar{l}+1}^{l-1} u(t_j) \cdot (t_{j+1} - t_j) \quad (3.2.15)$$

In this expression we assume that the time discretization is sufficiently fine that neglecting the second term on the right hand side does not lead to large errors. Determining the time delay then reduces to the problem of finding the index \bar{l} such that we can satisfy the following inequalities:

$$\sum_{j=\bar{l}+1}^{l-1} u(t_j) \cdot (t_{j+1} - t_j) \leq O_i(t_l) - I_i(\bar{t}) \leq \sum_{j=\bar{l}}^{l-1} u(t_j) \cdot (t_{j+1} - t_j) \quad (3.2.16)$$

Here the quantity $O_i(t_l) - I_i(\bar{t})$ is the length of pipe segment i , or zero if the weighted sum of the flows is zero. In particular, if $\bar{l}=0$ is reached before the solution is bracketed, $O_i(t_l)$ would be set to the initial condition for the segment. To determine the time delay that satisfies (3.2.16) we apply an origin tracking algorithm that calculates the position of past boundary conditions as they progress along the length of the pipe segment. This approach is equivalent to tracking the origin of discretized packets of fluid as they enter and travel through the pipe, allowing us to write an expression for $c(x=O_i(t_l), t_l)$.

To illustrate this idea, Figure 3.2 shows an example for pipe i with flow conditions from left to right. At timestep $l_1 = 1$, a fluid element enters the left boundary of the pipe, and we can record its originating node as ‘‘A’’ and its originating timestep as $l_1 = 1$. The concentration at the left boundary of the pipe is equal to the immediate concentration from node A, that is $c_i(I_i(t_{l_1}), t_{l_1}) = \hat{c}_A(t_{l_1})$. As time progresses, the element advances through the pipe and at some future timestep, say $l_2 = 5$, the concentration at the right boundary, however, is now the same as the concentration from node A at timestep l_1 , $c_i(O_i(t_{l_2}), t_{l_2}) = \hat{c}_A(t_{l_1})$.

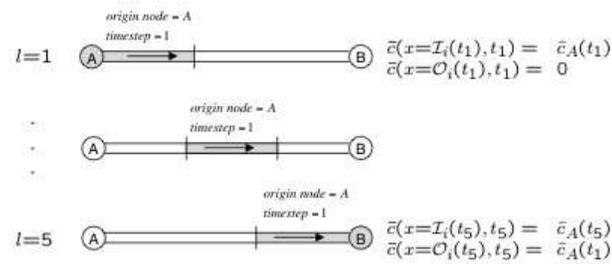


Figure 3.2. Origin Tracking Algorithm. This figure illustrates the flow of a single volume element through a pipe. Tracking these element allows us to determine relationships for the boundary concentrations at each point in time.

The linear time relationships in Figure 3.2 describe the concentration of the pipe boundaries as a function of the concentrations of connected nodes.

While this simple example illustrates the basic idea, the more detailed origin tracking algorithm developed in [134] formalizes the approach and handles situations of changing pipe velocity and flow reversals. This algorithm defines expressions for all network pipes under varying flow conditions. We denote these linear linking equations (3.2.13) by $p_l(c_i(O_i, t_l), \hat{c}_{k_l}(t_{\bar{l}})) = 0$ for all $l \in \Theta$. These linking equations replace equation (3.2.2) in the optimization problem, thus removing the spatial dependence. From [134], our Lagrangian-based algorithm has two advantages over previous work. First, it allows application of the simultaneous approach where the time discretization is fixed *a priori* by some scheme selected for the differential tank equations. Second, the analysis is performed on a pipe by pipe basis, not on the network as a whole. The processing and memory requirements are then linear with the number of pipes and efficient for large networks. While these simplifications introduce estimation errors in the time delays associated with paths through the network, they provide efficient scaling and sparsity.

3.2.2 Network Subdomain Approach

The above algorithm has successfully inverted for the magnitude, time and location of contamination sources on a relatively complex network model of approximately 500 nodes. Due to the limitations of direct linear solvers, it becomes significantly more expensive to solve the full network problem for increasingly larger networks. The challenge in this current work is to invert for contamination sources in much larger networks, where solving the optimization problem on the entire network is not possible. Instead, hydraulics and sensor measurements are considered from the entire network, a subdomain of the entire network is selected, and the optimization problem is formulated and solved for this subdomain only.

The subdomain is identified by choosing some subset of *selected* nodes and edges from the full network and is defined by the sets $\mathcal{SN} \subseteq \mathcal{N}$, $\mathcal{SJ} \subseteq \mathcal{J}$, $\mathcal{SP} \subseteq \mathcal{P}$ and $\mathcal{SS} \subseteq \mathcal{S}$. We then rewrite problem (3.2.1-3.2.4) for this subset of selected nodes. The node concentrations, $\hat{c}_k(t)$, injection terms, $m_k(t)$ and node expressions, (3.2.2-3.2.4), are only necessary for selected nodes. Given a set of selected nodes, it is straightforward to determine which pipes need to be considered in \mathcal{SP} . If pipe i is connected to a selected node at each end, the pipe itself is *selected* and must be modeled. On the other hand, if pipe i is not connected to any selected nodes, it is *deselected* and can be completely excluded from the model. Difficulty arises when pipe i is connected to a selected node at one end and a *deselected* node at the other. This pipe is still classified as *deselected*, but it represents a boundary to the subdomain. Since the flow conditions for the node at this boundary must be rectified, the flow into the subnetwork from this pipe is treated as a known external source and does not contribute any contaminant to the node.

The optimization problem in the space of the selected subdomain is given by,

$$\min_{\substack{m_k(t), c_i(x,t), \hat{c}_k(t) \\ \forall i \in \mathcal{SP}, k \in \mathcal{SN}}} \Psi = \int_0^{t_f} \sum_{r \in \Theta^*} \sum_{k \in \mathcal{SN}^*} w_k(t) (\hat{c}_k(t) - \hat{c}_k^*(t))^2 \delta(t - t_r) + \rho \sum_{k \in \mathcal{SN}} (m_k(t))^2 dt \quad (3.2.17)$$

s.t.

$$p_i(c_i(O_i, t), \hat{c}_{k_i(t)}(t)) = 0 \quad \forall l \in \Theta \quad \forall i \in \mathcal{SP}, \quad (3.2.18)$$

$$\hat{c}_k(t) = \frac{\left(\sum_{i \in \mathcal{SP} \cup \Gamma_k(t)} Q_i(t) c_i(x=O_i(t), t) \right) + m_k(t)}{\left(\sum_{i \in \Gamma_k(t)} Q_i(t) \right) + Q_k^{ext}(t) + Q_k^{inj}(t)} \quad \forall k \in \mathcal{S}\mathcal{J} \quad (3.2.19)$$

$$V_k(t) \frac{d\hat{c}_k(t)}{dt} = \left(\sum_{i \in \mathcal{SP} \cup \Gamma_k(t)} Q_i(t) c_i(x=O_i(t), t) \right) + m_k(t) - \left[\left(\sum_{i \in \Gamma_k(t)} Q_i(t) \right) + Q_k^{ext}(t) + Q_k^{inj}(t) \right] \hat{c}_k(t), \quad \hat{c}_k(0) = 0 \quad \forall k \in \mathcal{SS} \quad (3.2.20)$$

$$m_k(t) \geq 0 \quad \forall k \in \mathcal{S}\mathcal{N} \quad (3.2.21)$$

Note that the objective has been modified to sum the concentration errors for the selected sensor nodes in $\mathcal{S}\mathcal{N}^*$ and to regularize over the injection terms in $\mathcal{S}\mathcal{N}$. Similarly, the mass balances for the junctions and tanks have been modified include non-zero concentrations from selected pipes only. In the event of a contamination, a subdomain of the entire network can be selected and the solution techniques described in Section 3.2 can be used to solve the smaller, reduced network problem (3.2.17- 3.2.21). While the formulation presented is general enough for any set of selected network nodes, it remains to select a suitable subdomain. Any subdomain selection technique must satisfy the following goals:

1. The resulting nonlinear program must be small enough for current optimization techniques, therefore the selection technique must choose a reasonable number of nodes from the network.
2. This source inversion problem uses real time sensor measurements from the network and needs to be solvable in an online setting. Therefore, any preprocessing to select the network subdomain must be efficient for large networks.
3. The algorithm must select a subdomain that sufficiently represents the area of interest to the source inversion problem. This can be difficult since the actual source is, of course, not known.

In this study, a simple topological window is used to select the network subdomain. The geographical location of first sensor to detect contaminant is identified. Then, the n closest nodes to this location are selected for the subdomain, where the value of n is chosen large enough to represent the area of interest, but small enough to produce a solvable nonlinear program. Of course, there is no explicit guarantee that this subdomain will include the actual contamination locations, but if a contamination source is identified on the boundary of the subdomain, the window could be expanded or moved.

3.3 Numerical Results

Previous work has illustrated the effectiveness of formulation (3.2.1 - 3.2.4) and the origin tracking algorithm for identifying contamination sources for network models with approximately 500 nodes [134]. The primary purpose of the subdomain approach is to sufficiently reduce the size of the resulting nonlinear program, allowing solutions of large network inversion problems with optimization packages that use direct linear solvers. As the size of the subdomain is varied, we are interested in both the computational requirements (time and memory) and the quality of solutions. We first examine these effects by decreasing the size of the subdomain while keeping the time discretization fixed. The size of the nonlinear program decreases with the size of the subdomain, showing the efficiency gains that can be realized with the subdomain approach. Of course, the problem size can not be reduced indefinitely and we are more interested in determining the best subdomain size given a fixed budget of computational effort. A second set of results examines the effect of decreasing the subdomain size while keeping the size of the nonlinear program constant, by increasing the number of time discretizations. This result shows how the subdomain approach can be used to produce higher quality solutions by allowing a finer discretization for the same computational effort.

Results are demonstrated using a real municipal water network model, shown in Figure 4.3, with 3500 nodes with 350 randomly placed sensors¹. Figure 3.3 shows the circled portion of the network in more detail, where shaded nodes indicate nodes with installed sensors. The attack scenario is simulated using EPANET [179] where an hour long injection is simulated at hour 7 from node 2921.

We have developed a formulation tool that produces the nonlinear program for the subdomain problem. Specifying the optimization horizon, the subdomain size, and the time discretization, the software tool reads the simulated hydraulic information for the entire network and the concentration measurements for sensor nodes. The tool then forms the subdomain by identifying the first sensor to detect contaminant and selecting the nearest n nodes.

For a given time discretization, the formulation tool performs the origin tracking algorithm and determines the time delays for all the selected network pipes. It then writes the time discretized nonlinear program (3.2.17-3.2.21) using a simple Backward Euler technique for the tank equations(3.2.20). Although more elaborate discretization techniques could be used (like collocation on finite elements [26]), the stepsize tends to be governed by accuracy requirements on the time delays and a simple technique is sufficient for the tank equations. The discretized nonlinear program is written in AMPL [95] format. AMPL is an optimization modeling language that provides first and second order derivatives using automatic differentiation. Although the optimization problem (3.2.17-3.2.21) is a large scale quadratic program, the addition of general nonlinear reaction terms would destroy the linearity of the constraints. Axns well, equation (3.2.21), once discretized, produces a large number of variable bounds, and interior point algorithms provide an efficient alternative to active-set strategies for these problems. For these reasons, the nonlinear program is solved with IPOPT [237], a large scale, interior point, optimization tool (see <http://www.coin-or.org>). All optimization results were formulated with a 6 hour time horizon from the 5th to the 11th hour and were solved on a 1.8 GHz Pentium 4 machine.

¹This image has been cropped and does not show the full network model

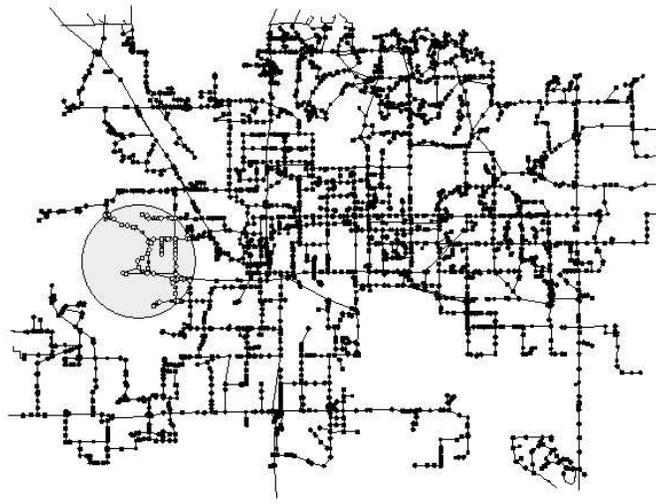


Figure 3.3. Municipal Water Network. This figure shows the entire water network. The circled area is shown in more detail in Figure 3.3.

3.3.1 Results: Fixed Discretization, Variable Problem Size

In this set of results, the size of the subdomain is varied, but the timestep is kept fixed at 5 minutes, giving 72 discretizations over the 6 hour optimization horizon. Figure 3.5 shows the inversion solution for a subdomain size of 900 nodes, plotting all injection profiles that, at some point, have significant values. Here, we see that the inversion solution is non-unique and has indicated four possible injection locations at nodes 2820, 2879, 2921, and 3612. Nevertheless, we can see two significant profiles for nodes 2921 (the actual injection location) and 2879 (a neighboring node) and the method has reasonably identified a small region containing the injection location (2921) and also correctly identified the injection time (7-8th hour) for this location.

The solution non-uniqueness is primarily due to the sensor locations and network topology [134]. Since the nonlinear program makes no assumption about the location, time, or number of injections, any linear combination of injections that produces the observed data is a valid solution, hence the need for the regularization term in the objective. Errors in the time delay approximation (the use of $t_{\bar{j}}$ instead of \bar{t}) can also contribute to poor solution quality. Since these errors propagate, they are more pronounced when optimizing over larger networks (or larger subdomains) and longer time horizons. We can see the evidence of this effect by examining the solution profile for a smaller subdomain. Smaller subdomains provide less information for the optimization problem, and it is expected that the solution quality would deteriorate as the subdomain size decreased. This is not necessarily the case. Figure 3.6 shows the inversion solution using a subdomain size of 100 nodes. Here, we can see that the two suspect locations (2820 and 3612) are no longer a part of the solution.

Table (3.1) summarizes the inversion results for subdomain sizes ranging from 900 to 50 nodes. The first

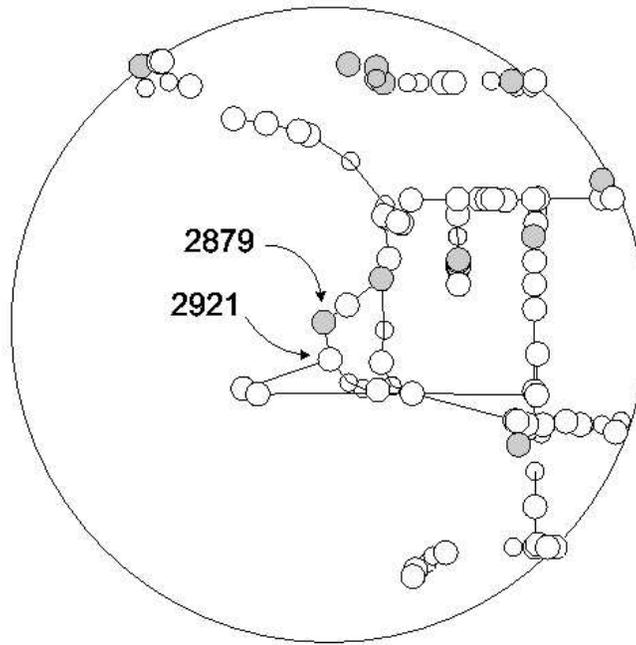


Figure 3.4. Municipal Water Network. This figure shows increased detail near the injection location, where shaded nodes indicate sensor locations.

column shows the number of nodes selected for the subdomain, the second column is the number of variables in the resulting nonlinear program, the third column is the approximate computational time required to perform the optimization, and the last column indicates the number of possible injection nodes identified in the solution. In all cases, the actual injection node was also identified as a possible injection location.

As expected, the computational time is dramatically reduced as the size of the subdomain is decreased. Interestingly, for this example, the solution quality does not deteriorate with smaller subdomains, but actually improves. This result is not general and using increasingly smaller subdomains may result in loss of necessary sensor information. On the other hand, our experience with smaller subdomains shows that they can remove network non-uniqueness and reduce the effect of errors in the time delay approximation.

3.3.2 Results: Fixed Problem Size, Variable Discretization

In this section, we examine the effect of changing the size of the subdomain while keeping the size of the nonlinear program constant. We are still interested in solutions using the 6 hour time horizon. Therefore, as the size of the subdomain is decreased, the number of time discretizations are increased proportionally, keeping the computational effort approximately constant. The quality of solutions is examined as the subdomain size decreases and the discretizations are refined.

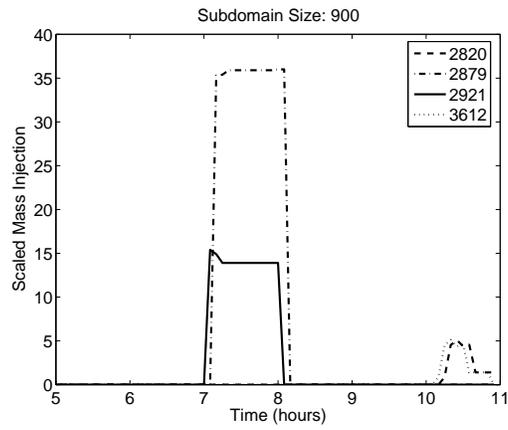


Figure 3.5. Solution with 900 Node Subdomain. This figure shows the solution of the inversion problem for a 5 minute time discretization and a subdomain size of 900 nodes. Although the solution is non-unique, showing 4 possible injection locations, the significant profiles, 2921 and 2879 are the actual injection location and its neighbor. The solution has also correctly identified the injection time.

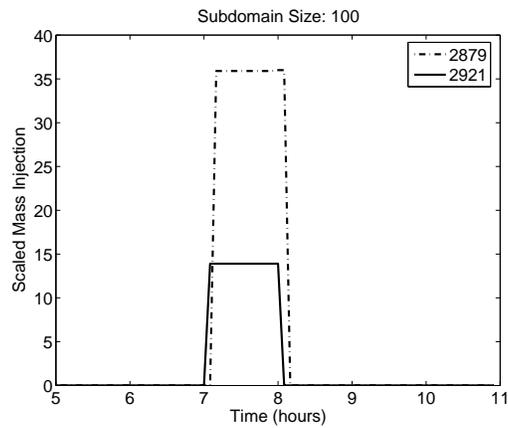


Figure 3.6. Solution with 100 Node Subdomain. This figure shows the solution of the inversion problem for a 5 minute time discretization and a subdomain size of 100 nodes. The solution quality has improved over the larger 900 node subdomain.

Table 3.1. Optimization Results with Fixed Time Discretization

Subdomain Size	Number of Variables	Execution Time (sec)	Identified Nodes
900	267698	70	4
800	238329	62	4
700	208656	56	4
600	178848	47	4
500	148176	38	4
400	117072	29	4
300	88128	18	2
200	57888	10	2
100	28800	5	2
50	14256	2	2

Table 3.2. Formulation Parameters for Different Subdomain Sizes

Subdomain Size	Number of Discretizations	Timestep Size (min)
300	24	15
200	36	10
100	72	5
40	180	2

Table 3.2 shows the parameters used to generate the inversion results. Here, we have kept the size of the nonlinear program approximately constant with $|\mathcal{N}||\Theta| = 7200$, where $|\cdot|$ denotes the cardinality of the set. For example, this corresponds to a subdomain of 300 nodes, and 24 time discretizations, requiring a 15 minute timestep to cover the complete 6 hour time horizon. Using this expression, we calculate the number of allowed time discretizations (and hence the length of each timestep) for each subdomain size. Therefore, finer discretizations are used at the expense of smaller subdomain sizes.

Using the formulation tool with the contamination scenario described previously, we generate the inversion solutions shown in Figure 3.7. The solution quality is very poor for large subdomains with the coarse discretizations. This can be attributed to the approximation errors resulting from the problem discretization and time delay calculation, which increase with the discretization size. Using a smaller subdomain reduces the modeling errors by decreasing the span of the network considered and by allowing finer discretization. Figure 3.7 shows how higher quality solutions can result as the subdomain size is decreased. Of course, if the subdomain size is decreased further, however, we may exclude nodes with important sensor information.

3.4 Conclusions and Future Work

The origin tracking algorithm successfully reduces the size of the optimization problem, allowing the use of simultaneous techniques on medium sized network models in a real time setting. Previous results have demonstrated the effectiveness of the dynamic optimization formulation on over 1500 test scenarios with

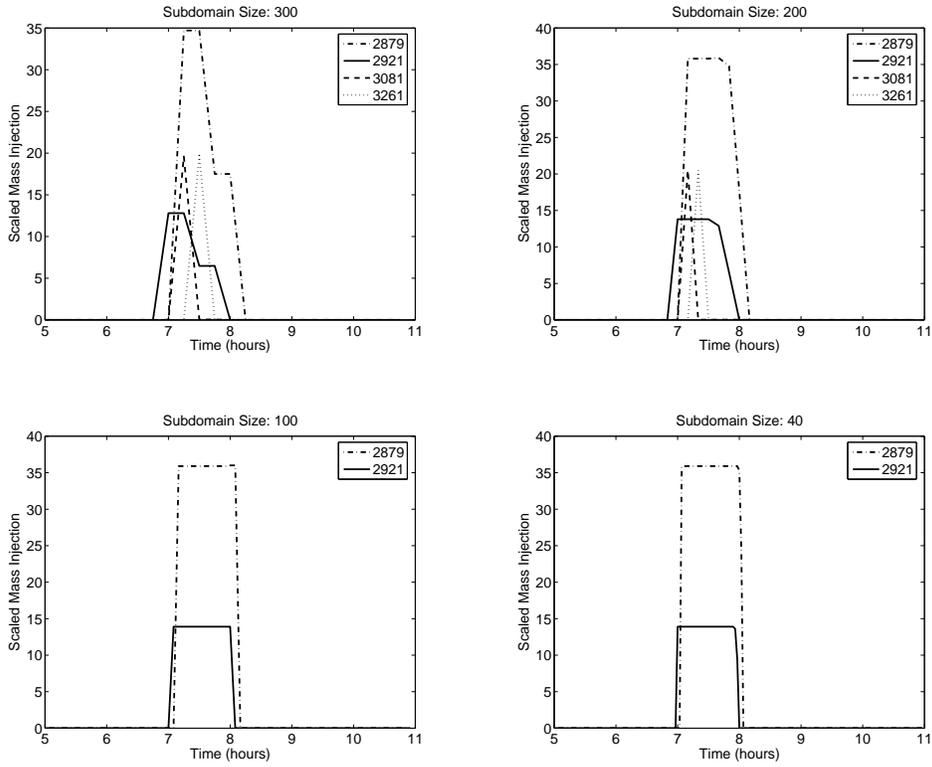


Figure 3.7. Optimization Solutions for Fixed Problem Sizes. This figure shows the optimization solution for different subdomain sizes, where the size of the nonlinear problem was kept constant using the parameters specified in Table (3.2)

real municipal network models. The use of the subdomain approach further reduces the problem size allowing real-time solutions for large network models.

The subdomain approach can also be an effective technique for improving the quality of the inversion solution. The cause for this is two-fold. While decreasing the subdomain size reduces the amount of information available to the optimization, it also allows a finer time discretization for the same computational effort. Furthermore, errors in the time delay approximations are less pronounced in a smaller network, helping to improve solution quality. Care must be taken, of course, to select a subdomain that does not exclude necessary information for the inversion problem.

In these examples, we found solutions with randomly placed sensors. It is important to identify optimal sensor locations to reduce solution non-uniqueness and decrease identification time. Also, statistical analysis is necessary to determine the effectiveness of the formulation when there are errors in flow and sensor data.

A rudimentary algorithm is presented for selecting the subdomain of interest based on geographical location. Future work includes the investigation of more advanced techniques for selecting this subdomain. If contamination sources are found on the boundary of the subdomain, this indicates that the subdomain may need to be expanded or moved. The subdomain can be expanded up to acceptable computational limits of the direct linear solver. Also, the window itself can easily be moved, solving the formulation again, until the identified contamination sources are interior to the subdomain. Moreover, multiple windows could also be used in a single optimization, each over a different period of time. This requires a multistage approach, where the first window is used for $t = [0, t_1]$, the second for $t = [t_1, t_2]$, etc. Multiple windows could also be solved in parallel. Using distributed parallel processing, each window can be solved independently, with an external algorithm to resolve the connected boundaries.

In conclusion, these preliminary results indicate the potential of simultaneous optimization techniques for efficient, effective identification of contamination sources in municipal drinking water networks. The subdomain approach allows consideration of large water networks, and reasonable solution times allow its use in a real time setting. Finally, although the simultaneous optimization formulation using the origin tracking algorithm has only been demonstrated on the contamination source inversion problem, its importance extends to other applications, including real time optimization for contaminant control of pipe networks.

Chapter 4

A Mixed Integer Approach for Obtaining Unique Solutions in Source Inversion of Drinking Water Networks

Carl D. Laird (Carnegie Mellon University), Lorenz T. Biegler (Carnegie Mellon University), Bart G. van Bloemen Waanders

4.1 Background

In previous work [136, 135, 231] the authors introduced a large scale nonlinear programming approach that used real-time concentration information from an installed sensor grid to accurately determine the time and location of the contamination event. This approach introduced unknown, time dependent injection terms at every node in the network and formulated a quadratic program to solve for the time profiles of the injections. The objective function was a least squares minimization of the errors between the calculated and measured node concentrations at the sensor nodes with a regularization term to force a unique solution. The constraints in the optimization problem were the partial differential equations of the water quality model for the network. In [136] this problem was then discretized with a fully simultaneous approach, using an *origin tracking algorithm* to characterize the pipe time delays and remove the need to discretize along the length of the pipes. The resulting large scale nonlinear program was solved using a nonlinear interior point code, IPOPT [236]. This approach was effective at identifying a family of possible injection scenarios.

The unregularized formulation of the source inversion problem can have many non-unique solutions. The regularized formulation, on the other hand, has a unique solution, but this solution is essentially linear combination of possible injection scenarios. With this approach alone, it is difficult to determine if the observed contamination was caused from a single injection location or multiple locations. In this work, we propose a problem reduction technique and formulate a mixed integer quadratic program (MIQP) to identify unique injection scenarios. This formulation includes constraints that further limit the solution

space and allows us to distinguish between single and multiple injection locations.

Section 4.1 gives a brief description of the formulation presented in [136], followed by a discussion of solution non-uniqueness and how this manifests in the regularized problem. In Section 4.2 we introduce the mixed integer formulation and show how the problem size can be reduced drastically using active-set information from the original continuous problem. We show the effectiveness of this approach on a real municipal water network in Section 4.3. Here we test both single and multiple location injection scenarios. Finally, we present some conclusions and directions for future work.

The purpose of the work in [136] was to present a formulation for solving the inverse problem of identifying the time and location of contaminant injections in real-time, using concentration information from a sparse sensor grid. This work assumed that contaminations occurred at network nodes and introduced unknown, time dependent contaminant injections at each node in the network. The optimization problem was then written as a weighted least squares minimization of the errors between the calculated and measured concentrations subject to the constraints of the water quality model. As with water quality simulation techniques [167, 227, 183], this approach assumed that the network flows are known inputs and modeled the water quality only. The solution to the least squares problem provides the complete time dependent profiles of the unknown mass injections at every node that give rise to the best weighted least squares match of the observed data.

In [136] an origin tracking algorithm was presented to characterize the pipe time delays and remove the need to discretize in space. Following this reduction, the remaining problem was then discretized in time alone, producing a nonlinear program of reasonable size. Using Θ to refer to the set of discretizations in time, and \mathcal{P} and \mathcal{N} to refer to the complete set of network pipes and nodes, respectively, the original continuous quadratic program (OCQP) for the inversion problem can be formulated as [136],

$$\min_{\bar{c}, c, m} f = \frac{1}{2} [c - c^*]^T W [c - c^*] + \frac{\rho}{2} m^T m \quad (4.1.1)$$

$$\text{s.t.} \quad \bar{c} - Pc = 0, \quad (4.1.2)$$

$$\bar{N}\bar{c} + Nc + Mm = 0, \quad (4.1.3)$$

$$m \geq 0, \quad (4.1.4)$$

where $\bar{c} = [\dots \bar{c}_{i,j}^I, \bar{c}_{i,j}^O \dots]$, $\forall i \in \mathcal{P}, j \in \Theta$ is a vector of pipe concentrations for the inlet (I) and outlet (O) of every pipe, discretized in time, $c = [\dots c_{i,j} \dots]$, $\forall i \in \mathcal{N}, j \in \Theta$ is the vector of calculated concentrations for every node at every time discretization, and $m = [\dots m_{i,j} \dots]$, $\forall i \in \mathcal{N}, j \in \Theta$ is the vector of unknown contaminant mass injections for every node at every time discretization. The matrix P is defined by the origin tracking algorithm, and the matrices \bar{N} , N , and M are the Jacobians of the discretized mass balance constraints (for junctions and storage tanks) with respect to the pipe concentrations, node concentrations, and unknown injections, respectively. The diagonal matrix W is a weighting matrix for the least squares errors, with nonzero elements only for sensor nodes at sample times. The integral of the absolute value of flow through the corresponding node is used for the nonzero elements. This provides a flow based weighting and shifts the least squares error from a concentration basis to a mass basis.

The second term in the objective is a regularization term that forces a unique solution to the problem. First, consider the unregularized case. With $\rho=0$ problem (4.1.1-4.1.4) is a convex (not necessarily strictly convex) quadratic program; therefore, the QP solution is a global minimum for the objective value, but the

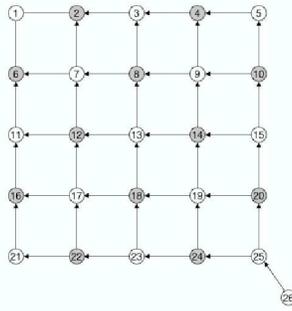


Figure 4.1. Small symmetric grid network with sensors installed at every second node, indicated by the shading.

minimizer it is not necessarily unique.

Consider the grid network shown in Figure 4.1 with sensors installed at every second node (indicated by the grey shading) and a single reservoir water source at node 26. Network flows are constant in the directions indicated by the arrows with demands at the boundary nodes only. These demands were selected to introduce symmetry about an axis through nodes 1 and 26 and time delays that range 0.5 to 5 hours. Selecting node 13 as the injection location, we introduce contaminant from time, $t = 0.5$ to $t = 1$ hours. With the flows used in this network, the contaminant will flow from node 13 to nodes 12 and 8 in about an hour and a half, where the sensors will then detect the contaminant. If we consider only the observed concentration measurements at nodes 8 and 12 (and zero concentration measurements from the other sensors), we see two possible injection scenarios capable of producing the observed concentrations. The contaminant could have been injected at node 13 (the actual injection) or at both nodes 8 and 12 simultaneously. As well, any linear combination of these two scenarios is also a possible solution to the unregularized problem.

There are an infinite number of solutions to the unregularized problem. With a small positive value for ρ , however, the QP obtains a unique *regularized* solution. [136] demonstrate the source inversion capability on the grid network example described above. Simulating the injection from node 13 with EPANET [179] and using the origin tracking algorithm to describe the pipe time delays, the authors formulate problem (4.1.1-4.1.4) with $\rho = 1 \cdot 10^{-4}$, 5 minute timesteps, and a time horizon of 4 hours (48 timesteps). The optimization problem was written in AMPL [96] format by a software tool that reads the EPANET input and output files. This AMPL problem was then solved using IPOPT [236] and the solution is shown in Figure 4.2. This figure shows the time profiles for the non-zero mass injections only with time in hours along the abscissa and the mass flowrate of the injections along the ordinate. As expected, we see that

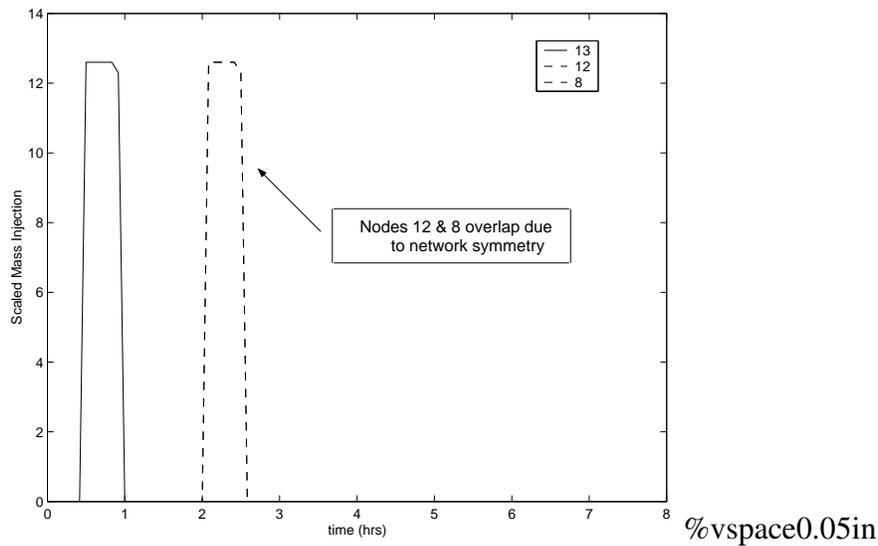


Figure 4.2. Grid Network Example Solution. Solution of problem (4.1.1-4.1.4) on the grid network with an injection from node 13 at $t = 0.5$ hours.

nodes 8, 12, and 13 all contribute injection profiles to the solution. The effect of the regularization term is apparent. The solution given is a linear combination of the two solutions discussed above. With the simplicity of the grid network topology and flow patterns, it is straightforward for us to examine Figure 4.2 and see the two solutions of interest, namely an injection at node 13 at $t = 0.5$ hours, or a simultaneous injection at nodes 8 and 12 at $t = 2$ hours. Unfortunately, for a more complex network, with time varying flow patterns, interpreting these solution profiles is non-trivial.

The existence of non-unique solutions is unavoidable, as they are a direct function of the incomplete sensor configuration. The regularized OCQP gives only one regularized solution. The unique scenarios that give rise to this linear combination need to be inferred. Knowing only that the contaminant could have been injection at node 8, 12, or 13 (or any combination) does not allow us to prioritize the locations or determine their individual likelihood. Given the knowledge that a single injection from node 13 could produce the observed concentrations, we would likely investigate this location first, as opposed to nodes 8 and 12, where simultaneous injections are required. Furthermore, while it is important to identify when a contamination event could have occurred from a single location, it is even more important to identify when this is not possible. We are particularly interested in immediately identifying when contamination events involve multiple injection locations.

4.2 Mixed Integer Formulation

In this section, we introduce an approach that not only indicates when multiple injections are likely, it also provides the time and location of each injection. We propose an algorithm that uses a mixed integer quadratic programming (MIQP) formulation to search the space of solutions provided by the OCQP, (4.1.1)-(4.1.4). The mixed integer formulation allows us to include discrete constraints on the problem, such as limiting the number of injection locations and provides more information than the OCQP solution alone. Consider the mixed integer reformulation of (4.1.1)-(4.1.4),

$$\min_{\bar{c}, c, m} f = \frac{1}{2} [c - c^*]^T W [c - c^*] \quad (4.2.5)$$

$$\text{s.t.} \quad \bar{c} - Pc = 0, \quad (4.2.6)$$

$$\bar{N}\bar{c} + Nc + Mm = 0, \quad (4.2.7)$$

$$Ly_i \leq m_{i,j} \leq Uy_i, \quad \forall i \in \mathcal{N}, j \in \Theta, \quad (4.2.8)$$

$$\sum_{i \in \mathcal{N}} y_i = n. \quad (4.2.9)$$

Here, $y_i \in \{0, 1\}$ is a binary variable, n is a parameter representing the number of injection locations. The parameters L and U correspond to the nonzero detection limit and an upper bound on the injection variables, respectively. Problem (4.2.5)-(4.2.9) has the same number of continuous variables as the OCQP and the number of binary variables equals the number of nodes in the network. However, since our goal is to search the space of solutions arising from the OCQP, we do not need to formulate the MIQP with the full space of variables.

Consider the grid network problem presented in Section 4.1. With 26 nodes and 48 timesteps, this problem has 1248 continuous mass injection variables (one for each node at each timestep). If we look at the solution of the OCQP given in Figure 4.2, we immediately notice that most of the bound constraints (4.1.4) are active, meaning that the majority of the mass injection variables are at their lower bound of zero. In this example, there are only 18 nonzero mass injection variables (6 timesteps for each of the 3 nodes). Furthermore, there are only 3 nodes represented in the solution, 8, 12, and 13. To search within the solution space of the OCQP, we only need to consider nonzero mass injection variables and only the reduced set of node locations.

Therefore, if we first solve the OCQP and identify the injection variables that are zero (or below some detection limit), we can remove these variables and formulate the mixed integer problem in the space of the nonzero variables only. Let m_r refer to the set of mass injection variables that were inactive¹ at the solution of the OCQP. The set \mathcal{N}_r contains all the nodes represented by this selection of mass injection variables. The set $(\Theta_r)_i$ contains the timesteps of the selected injection variables for node i . With this notation, $m_r = [\dots (m_r)_{i,j} \dots]$, $\forall i \in \mathcal{N}_r, j \in (\Theta_r)_i$. Defining M_r as the columns of M in the space of m_r , we let $A_r = -(\bar{N} \cdot P + N)^{-1} M_r$, we further reduce the problem by removing the pipe and node concentrations. Defining m_r^* as the solution of the original problem in the space of the selected variables only, we introduce a new variable, $\Delta m_r = m_r - m_r^*$. Since the value of the regularization parameter ρ is small, we can neglect

¹With an interior point method, the active variables will not be pushed completely to their bounds and we must select the active set using a small tolerance.

the regularization term and, using the reductions described above, rewrite the objective function (4.1.1) as,

$$\begin{aligned} & \frac{1}{2} [A_r(m_r^* + \Delta m_r) - c^*]^T W [A_r(m_r^* + \Delta m_r) - c^*] \\ &= \frac{1}{2} [A_r m_r^* - c^*]^T W [A_r m_r^* - c^*] \\ & \quad + [A_r m_r^* - c^*]^T W [\Delta m_r] + \frac{1}{2} [A_r \Delta m_r]^T W [A_r \Delta m_r]. \end{aligned} \quad (4.2.10)$$

The term $\frac{1}{2} [A_r m_r^* - c^*]^T W [A_r m_r^* - c^*]$ is constant and, from the optimality conditions for the original problem, we know that $[A_r m_r^* - c^*]^T W = 0$ (the multipliers for the inactive bound constraints are all zero). Therefore, both of these terms can be removed from the objective and we write the reduced form of problem (4.2.5)-(4.2.9) as,

$$\min_{m_r} \hat{f} = \frac{1}{2} [\Delta m_r]^T Q [\Delta m_r] \quad (4.2.11)$$

$$\text{s.t.} \quad Ly_i \leq (m_r^* + \Delta m_r)_{i,j} \leq Uy_i, \quad \forall i \in \mathcal{N}_c, j \in (\Theta_r)_i, \quad (4.2.12)$$

$$\sum_{i \in \mathcal{N}_c} y_i = n, \quad (4.2.13)$$

where, $Q = A_r^T W A_r$. The matrix A_r can be calculated by perturbing each element of m_r and simulating to find the response on c . Formulation (4.2.11-4.2.13) is a Mixed Integer Quadratic Program (MIQP) and forms the base problem for our refinement of solutions.

In addition to constraint (4.2.13) which limits the space of solutions to those with n injection locations, we can also add integer cuts [41] eliminating unlikely scenarios or previously visited solutions. To cut a particular combination of injections from the solution space, define \mathcal{Y}_1 to be the set of y_i variables that are one and \mathcal{Y}_0 to be the set of binary variables that are zero. We can then add the cut,

$$\sum_{i \in \mathcal{Y}_1} y_i - \sum_{i \in \mathcal{Y}_0} y_i = |\mathcal{Y}_1| - 1, \quad (4.2.14)$$

to remove that particular combination from the set of possible solutions. Here, $|\mathcal{Y}_1|$ refers to the cardinality of \mathcal{Y}_1 (the number of elements in the set).

To show the effectiveness of this MIQP formulation, we again consider the grid network example from Section 4.1 and formulate problem (4.2.11-4.2.13) using the solution of the OCQP, shown in Figure 4.2. Since the number of injection locations in the solution is only 3, we can completely enumerate all the possible combinations for the binary variables.

Table 4.1 shows all 8 enumerated solutions, sorted first by objective value and then by $\sum_{i \in \mathcal{N}_c} y_i$. Notice that the top entry in the table, Solution 1, tells us immediately that an injection at node 13 alone is able to reproduce the observed data precisely. Considering the possibility of two injection locations, the second solution with injections at node 8 and 12 together are also able to reproduce the data precisely. These numbers confirm what we already know from the network topology and flow conditions. Solution 3 involves all three nodes and is equivalent to the solution from the OCQP. Solution 4, appears to have a reasonable objective value with injections at both nodes 12 and 13. In this case, the mass injection profile (not shown) for node 12 is at its lower bound, L for all timesteps considered. The solution matches the data

Table 4.1. Enumerated Solutions to Grid Network Example.

Solution No.	Objective Value	n	y_8	y_{12}	y_{13}
1	0.000	1	0	0	1
2	0.000	2	1	1	0
3	0.000	3	1	1	1
4	1.439	2	0	1	1
5	1.439	2	1	0	1
6	169.9	1	1	0	0
7	169.9	1	0	1	0
8	339.7	0	0	0	0

well with the injection at node 13, and then adds as little as possible from node 12. Increasing the value of L will force an increase in the objective value for this case. The same is true for Solution 5 with node 8 and 13. Nevertheless, we need not be concerned since better objective matches have been seen with node 13 alone. Solutions 6 and 7 shows that a single injection at node 8 or node 12 is not able to match the observed data.

For larger problems, we do not enumerate all the possible combinations. Instead, we set a particular value for the number of injection locations, n , and evaluate the MIQP repeatedly, searching for good solutions. Starting with $n=1$ we find all solutions that provide a reasonable match, excluding each previous solution as we proceed. We then increase n and continue. Our recommended approach for solving the source inversion problem and identifying the significance of solutions is as follows.

Algorithm 1

1. Solve the OCQP:
Given sensor and flow data, formulate and solve the OCQP, as described in [135?]. Identify the set of selected nodes and timesteps for \mathcal{N}_r and $(\Theta_r)_i, \forall i \in \mathcal{N}_r$.
2. Calculate Q:
Each column in A_r is the gradient of the vector c with respect to a particular entry in m_r . Set all the entries of m_r to zero except for one, and solve the forward problem. The values of c from the simulation form a column of A_r . Performing the perturbation for each $i \in \mathcal{N}_r, j \in (\Theta_r)_i$ gives us the entire matrix A_r . Calculate $Q=A_r^T W A_r$.
3. Formulate the base MIQP:
Select reasonable upper and lower bounds U , and L , for the injection variables and formulate the mixed integer quadratic program (4.2.11-4.2.13). This problem forms the base MIQP.
4. Find reasonable upper bound on the objective, \hat{f} :
Force all injections to zero by setting $n=0$. Solve the base MIQP to find \hat{f} for the case where

injection variables are all zero. Since this represents a reasonable upper bound on the objective, store the objective value as \hat{f}_0 .

5. for each $n=1..|\mathcal{N}_c|$:
 - (a) Initialize the problem for the current value of n :
Set $\hat{f}=0$ and formulate the base MIQP for n injection locations.
 - (b) while $(\hat{f}/\hat{f}_0 < \epsilon)$:²
 - i. Solve the MIQP:
Solve problem (4.2.11)-(4.2.13) with any additional integer cuts and record the values of the binary variables and the objective. Store the value of the objective in \hat{f} .
 - ii. Exclude previous solution:
Add the integer cut (4.2.14) for the values of the binary variables from the last solution.
Return to Step 5.2 to continue finding new solutions for the current value of n .

Upon completion of this algorithm all the possible solutions with objective values less than ten percent of \hat{f}_0 will be recorded. One can then examine these solutions to determine which injections correspond to likely scenarios.

4.3 Results

In this section, we demonstrate the effectiveness of Algorithm 1 and the MIQP formulation using a real municipal water network model. We simulate two injection scenarios, one with a single injection location and one with two injection locations. We then perform Algorithm 1 on both these scenarios, demonstrating that this approach is capable of distinguishing between the single injection scenario and the multiple injection scenario.

We test the approach on a real municipal water network with approximately 500 nodes and 600 links, shown in Figure 4.3. We select 50 sensor nodes based on a weighted random selection. In this selection, we first assign a weight to each node, using the total volume of water that has flowed through each node over a simulated 16 hour period. To ensure we do not skew the selection drastically, any node with a weight that is less than five percent of the maximum weight is set to this lower bound. We then randomly sample this distribution 50 times to select the sensor nodes.

In Test 1, we simulate an hour long contamination from node A at $t=1$ hour. In Test 2, we keep this contamination event and add another hour long injection from node D, also at time $t=1$ hour. Performing Algorithm 1, we first formulate and solve the OCQP. The solution of OCQP for both of these tests is not shown since they contain such a large number of injection nodes. For Test 1, the solution of the OCQP indicates 10 nodes and 73 injection variables to consider (3 nodes with 13 timesteps, 6 nodes with 5 timesteps, and one node with 4 timesteps). In Test 2, the scenario with two injection locations, there are 13

²The parameter ϵ represents the limit on \hat{f}/\hat{f}_0 and is set to 0.1 in this study.

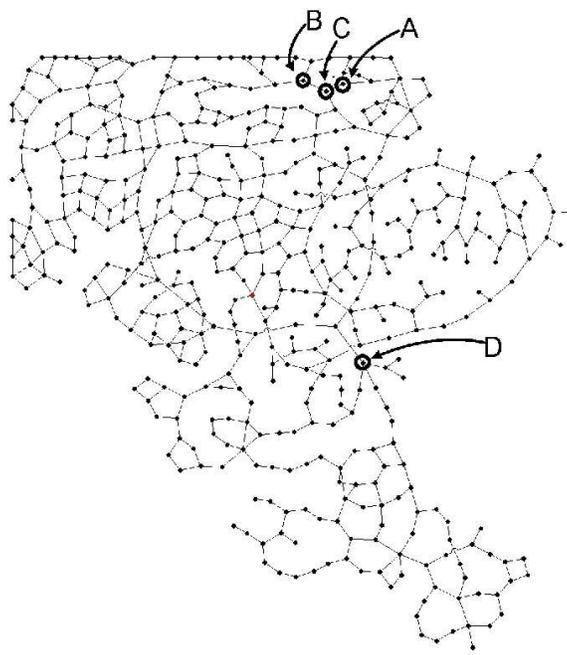


Figure 4.3. Network model showing the four nodes of interest, A, B, C, and D

nodes and 79 injection variables to consider (one node with 13 timesteps, 3 nodes with 12 timesteps, 2 nodes with 5 timesteps, 4 nodes with 4 timesteps, and 3 nodes with 2 timesteps).

Using the solutions from the OCQP we formulate the base MIQP in AMPL [96] format. We then continue the algorithm, using the NEOS Server for Optimization (www-neos.mcs.anl.gov) where each MIQP was solved with the mixed integer, nonlinearly constrained algorithm, MINLP branch and bound solver by Fletcher and Leyffer [144].

Figures 4.4 and 4.5 show all the recorded solutions from Algorithm 1. The horizontal axis shows n , the number of allowed injection locations, and the vertical axis shows the fractional objective value for each solution on a log scale. Many different solutions may have the same objective values and points may overlap in these figures. With the approach outlined in Algorithm 1, we have a column of solutions for each value of n . For Test 1 there are single injection solutions that provide a reasonable match. For Test 2, the two injection contamination scenario, there are no single injection solutions that match the observed data. This immediately tells us that we have a situation with multiple injections and demonstrates the ability of this approach to differentiate single location contaminations from multi-location contaminations.

While Figures 4.4 and 4.5 indicate the number of likely injection locations, they do not show us the nodes involved in the injections. This information was recorded in the algorithm, however, and Tables 4.2 and 4.3 summarize the solutions near the lower left corner of Figures 4.4 and 4.5. The results in Table 4.2 show three single injection solutions that match the observed data (the fractional objective values are all under one percent). These three are neighboring nodes in the network and, with no sensor between them, are indistinguishable with the observed data. These results indicate that a single injection location is possible at any one of these three locations. Even more encouraging are the results for Test 2, shown in Table 4.3. In

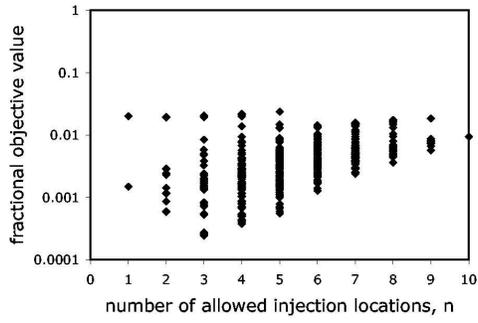


Figure 4.4. Recorded Solutions for Test 1. Results of Algorithm 1 for the contamination scenario with a single injection location.

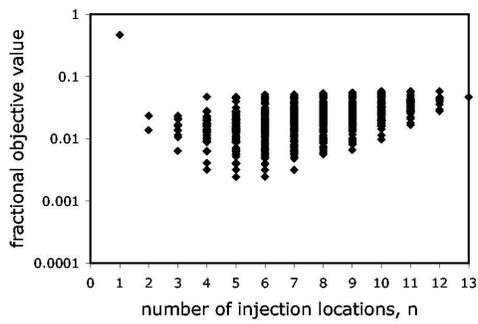


Figure 4.5. Recorded Solutions for Test 2. Results of Algorithm 1 for the contamination scenario with two injection locations.

Table 4.2. Detailed Results for Test 1

n	Frac. Obj.	Comment
1	0.001	$y_A=1$, others 0
1	0.001	$y_C=1$, others 0
1	0.020	$y_B=1$, others 0
2	0.001	$y_A=1^a$
\vdots	\vdots	\vdots

^aOne of the other nodes, not shown in Figure 4.3 has a value of 1, all others 0.

Table 4.3. Detailed Results for Test 2

n	Frac. Obj.	Comment
1	0.473	$y_D=1$, others 0
2	0.014	$y_C=y_D=1$, others 0
2	0.014	$y_A=y_D=1$, others 0
2	0.024	$y_B=y_D=1$, others 0
2	0.465	$y_D=1^a$

^aOne of the other nodes, not shown in Figure 4.3 has a value of 1, all others 0.

this test, we simulated two injections, one from Node A and one from node D. The results show that there are no single injection scenarios that could have produced the observed data (the best fractional objective for a single injection solution is almost fifty percent). On the contrary, there are three solutions with two injection locations that match the data. Table 4.3 shows that all three solutions with two injection locations include node D. We also know from Test 1 that nodes A, B, and C are indistinguishable with the selected sensor layout. It is not surprising then, that the three solutions that match the data are node A with D, node B with D, and node C with D. Considering only the results from the solution of the OCQP, we could not determine if the contamination was the result of a single injection or multiple injections. With the mixed integer analysis on Test 2, however, we can immediately see that there are multiple injection sites to consider, and, with surety, we know which sites we should immediately investigate.

4.4 Conclusions and Future Work

These results demonstrate that the MIQP formulation and the algorithm presented are extremely effective at identifying unique solutions from the family of solutions provided by the original problem (OCQP). The approach is able to identify unique solutions and find good solutions with a minimum number of injection locations. The greatest strength of this approach, however, is its ability to distinguish multiple injections. The results clearly show, for the scenarios considered, that the two injection scenario was distinguishable from single injection scenario and the injection locations were identified as accurately as possible with the sparse sensor layout.

Furthermore, the general MIQP takes only minutes to formulate (dominated by the calculation of Q) with a computational complexity that is linear in the number of selected injection variables. Tables 4.2 and 4.3 only show a portion of the number of problems that were solved. For completeness, we enumerated the entire set of possible binary variables and each MIQP solved in under a second. These favorable computation results make this approach reasonable for a real-time setting.

While these results are very encouraging, both in their effectiveness and their computational effort, they only represent two injection scenarios. It remains for us to analyze the behavior of the MIQP approach and Algorithm 1 on additional injection scenarios. It will also be interesting to see how well the approach can differentiate injections from more than two locations.

In all tests, the sensor data was simulated and the demands were characterized perfectly. In a real life situation, the sensor data will be noisy and the demands will only be loosely characterized. We need to develop a framework to perform the source inversion considering the uncertainty associated with these parameters. For this we propose to use a multi-scenario framework, where a single optimization problem is formulated with a large number of scenarios, each being a statistical sample from the expected demand distributions. The injection variables will be common across each of these scenarios and a solution of this problem will yield an expected value for the injection variables under the uncertainty of the demands. This approach has been previously described for the design of chemical plants under uncertain conditions in [178]. While the optimization problem grows linearly with the number of scenarios considered, the structure of the linear system solved at each iteration of the optimization can be exploited to provide efficient solutions for a very large number of scenarios. Furthermore, great performance gains can be made by solving this system in parallel. We believe that this is the next critical step in solving the problem of effective source inversion in drinking water networks.

Chapter 5

Sensor Placement in Municipal Water Networks with Dynamic Integer Programming Models

Jonathan Berry, William E. Hart, Cynthia A. Phillips, James Uber (University of Cincinnati), Jean-Paul Watson

5.1 Background

In this chapter, we describe a mixed-integer programming (MIP) formulation for sensor placement optimization in water distribution networks that precisely characterizes the impact of a contamination event. This MIP is formulated using data from detailed water quality simulations, which is used to compute contaminant concentration time-series for each junction in a network. These time-series are then used to exactly determine the impact of any contamination event, including the impact of detection at different junctions in the network. This MIP can implicitly capture the impact of temporal dynamics of contaminant flow to determine whether a downstream population is affected before the contaminant is detected. Moreover, the fidelity of the water quality simulations and corresponding contaminant impact calculations is independent from the MIP model itself. Thus our MIP model can be used with a very diverse set of sensor placement objectives without changing the underlying sensor placement solution strategy.

The next section reviews the previous literature on combinatorial sensor placement formulations for water distribution networks, which highlights the fact that few sensor placement formulations have been developed that adequately address temporal dynamics in contaminant flows. Section 5.2 describes our MIP formulation, as well as a revision of this formulation that facilitates its application on large-scale problems. Section 5.3 describes the application of exact and heuristic solvers to solve large sensor placement problems. Finally, we discuss the significance of these results in Section 5.4.

Sensor placement problems can be naturally formulated as optimization problems. Although our focus is on detecting contamination events within an EWS, methodologies for placing water quality monitoring stations are directly related to sensor placement problems for EWS design. Consequently, we include them in our comparison of modeling approaches, and for simplicity of presentation we refer to the placement of water quality monitoring stations as a sensor placement problem.

For EWS design, the general goal of sensor placement optimization is to place a limited number of sensors in a water distribution network such that the impact to public health of an accidental or intentional injection of contaminant is minimized. However, there is no specific formulation of the problem that is widely accepted by the water resources management community. A major contributing factor is the wide range of design objectives that are important when considering sensor placements, e.g., minimizing the cost of sensor installation and maintenance, the response time to a contamination event, and the extent of contamination – which impacts recovery costs. Additionally, it is difficult to precisely quantify the health impact of a contamination event; human water usage is often poorly characterized, in terms of both water consumption patterns and how water consumption impacts population health. Consequently, surrogate measures like the total volume of contaminated water consumed have been used to model health impacts; this measure assumes that human water consumption is proportional to demand.

One common feature of sensor placement formulations is the simplifying assumption that sensors can accurately measure water quality and/or the presence of contaminants. Although this may be reasonable for water quality measurements, it remains unclear how well this assumption will apply to EWS design activities. New sensor technologies are needed to detect contaminant threats, but the robustness and accuracy of these contaminant-specific sensors remains unclear.

Sensor placement formulations can also be categorized by the manner in which contaminant events are modeled. In a *dynamic* sensor placement formulation, the impact of a contamination event at a network junction is determined exactly, using a detailed water quality simulation to compute contaminant concentration time-series for each junction in the network. In a *static* sensor placement formulation, the impact of a contamination event is estimated by analyzing some combination of (1) flow directions and velocities obtained via hydraulic simulation, (2) pipe lengths, and (3) junction demands. This categorization reflects the fact that dynamic formulations capture temporal dynamics, while static formulations assume one or more patterns of water flow separately. Although water network models typically describe water flow with a set of flow patterns, static sensor placement formulations do not model transitions between these flow patterns.

5.1.1 Static Formulations

Almost all of the research on sensor placement optimization has considered static formulations. These formulations are distinguished by (1) the design and/or performance objective considered and (2) how network flows are modeled. A formulation related to set covering problem was developed by Lee et al. [142, 141] and subsequently refined by several researchers (see [164] for a review). Sensor placement objectives are motivated by the observation that water quality measurements at a sensor reflect the quality of water at nearby points within the network. Specifically, network flow information is used to compute a matrix that determines what fraction of water flow passes through each junction, and this information is

used to find sensors that maximize the coverage of water flow.

[125] and [162] introduced a static formulation in which the objective is to ensure that a pre-specified maximum volume of contaminated water consumed prior to detection can be guaranteed. This formulation is based on a set covering problem, where sensors cover junctions for which detection can be guaranteed within the pre-specified “level of service.” This is a static sensor placement formulation, so the calculation of contaminant consumption is an estimated quantity. This calculation is performed using an auxiliary network, which has directional edges that are determined via analysis of hydraulic simulation outputs. A directed edge is added in this network from junction v_i to v_j if there is flow from v_i to v_j at any point in the simulation. These directed edges are weighted by the *average* velocity from v_i to v_j over the course of simulation, which allows the use of this auxiliary graph (along with network pipe lengths) to estimate the shortest travel time between all pairs of vertices in the original water network.

Finally, Berry et al. [31, 32] and [239] introduce a variety of static formulations that are expressed as MIPs. In Berry et al. [31, 32], the objective is to minimize the expected fraction of the population exposed to a contamination event. Hydraulic simulation results are used to compute a fixed flow orientation for each pipe in the network for each of p distinct non-overlapping time intervals, referred to as patterns. A population consuming water at a junction v_j is considered exposed to contaminant injected at a junction v_i if and only if there exists a flow path from v_i to v_j along which there is no sensor. [239] generalize this formulation to consider a range of optimization objectives, some of which account for travel times by considering contaminant propagation rates within each flow pattern separately, as opposed to aggregating these into a single auxiliary network as is done by [125].

5.1.2 Dynamic Formulations

Static formulations fail to model time-varying flow characteristics like contaminant dilution, concentration level, and transport interactions. Instead, these models simply track the presence or absence of contaminant at various points in a network. The rate of contaminant flow can be captured, but these calculations are only approximate as they do not account for temporal variations such as the effect of shifts between flow patterns.

In contrast, dynamic sensor placement formulations can precisely characterize the impact of a contamination event on a network. Dynamic formulations use detailed water quality simulations to compute contaminant concentration time-series for each junction in a network. These time-series can then be used to exactly determine the impact of any contamination event, e.g., the amount of contaminant consumed at every network junction.

[164] propose a dynamic sensor placement formulation whose objective is to ensure that the expected impact of a contamination event is within a pre-specified “level of service”, defined as the maximum volume of contaminated water consumed prior to detection in which the contamination exceeds a pre-specified threshold. Mirroring the earlier approach of Kessler et al., Ostfeld and Salmons introduce a formulation based on set covering, in which sensors are allowed to cover only those junctions for which detection can be guaranteed within the pre-specified level of service. The calculation of contaminant consumption is computed using PiplineNet [27].

The sensor placement model described in this article was first presented by [34]. This model was recently refined by [170], who consider an alternate formulation of the underlying combinatorial structure of the sensor placement problem. They argue that the structure of this formulation can be exploited to limit the number of water quality simulations that are needed for sensor placement.

5.2 A Dynamic MIP Model

We now introduce a dynamic formulation of the sensor placement optimization problem. We assume a fixed budget of p of sensors, each of which can be placed at any junction in a distribution network; installation of sensors on pipes is disallowed because we rely on water quality simulations that cannot provide this information. We assumed that sensors are capable of detecting contaminants at any concentration level, and we assume that a general alarm is raised when contaminant is first detected by a sensor, such that all further consumption is prevented.

We model a water distribution network as a graph $G = (V, E)$, where vertices in V represent junctions, tanks, or other sources, and edges in E represent pipes, pumps, and valves. In higher-granularity (i.e., skeletonized) network models, each vertex may represent an entire neighborhood or other geographic region. We assume that demands follow a small set of patterns, e.g., one pattern per hour throughout the day. Each pattern represents the demand during a particular time interval on a “typical” day. Because each pattern holds steady for one or more hours, we assume the gross flow characteristics induced by these demands holds steady during the time period associated with that pattern.

Let \mathcal{A} denote the set of contamination scenarios against which a sensor configuration consisting of p sensors is intended to protect. A contamination scenario consists of individual contamination events, each of which can be characterized by quadruples of the form (v_x, t_s, t_f, X) , where $v_x \in V$ is the origin of the contamination event, t_s and t_f are the contamination event start and stop times, and X is the contamination event profile, e.g., arsenic injected at a particular concentration at a given rate. The quadruples can easily be extended to account for multiple coordinated contamination events. Let t_s^a and t_f^a respectively denote the start and stop times of the contamination event for scenario a . The impact of a given contamination scenario can be evaluated using water quality analysis software (e.g., EPANET [179]) to compute the contaminant concentration at each junction in the network from time t_s^a to an arbitrary point $t_h \geq t_s^a$ in the future. The results of such an analysis are expressed in terms of concentration time-series τ_j for each $v_j \in V$, with samples at regular (arbitrarily small) intervals within $[t_s^a, t_h]$.¹

It is straightforward to compute the total impact, $d_a(t)$, the total network-wide impact of a contamination scenario a at any given point in time $t \geq t_s^a$. We defer precise specification of an “impact” to Section 17.4; a key characteristic of our dynamic formulation is that it captures a wide range of possible definitions. Let γ_{aj} denote the earliest time t at which a hypothetical sensor at junction v_j can detect contaminant due to a contamination scenario a . If no contaminant ever reaches v_j , then $\gamma_{aj} = t^*$, where t^* denotes the stop time imposed on the water quality simulations; otherwise, γ_{aj} can be easily computed from τ_j . We next define

¹Our discussion throughout the chapter assumes that when contamination scenarios consist of multiple events, these events involve identical contaminant types. It should be clear from our definition of contamination events that this is not necessarily true, and thus the approach described here naturally generalizes.

$d_{aj} = d_a(\gamma_{aj})$, i.e., the total impact of a contamination scenario a if the contaminant is first detected by a sensor at v_j . Finally, let q denote a “dummy” location that corresponds to failed detection of contamination scenario a , such that d_{aq} is defined as the total impact of contamination scenario a if it is not detected before t^* .

Our formulation models the placement of p sensors on a set $L \subseteq V$ vertices, with the objective of minimizing the expected impact of a set \mathcal{A} of contamination scenarios. A likelihood α_a is assigned to each contamination scenario $a \in A$, such that $\sum_{a \in A} \alpha_a = 1$. Let \mathcal{L}_a be the subset of vertices in $L \cup \{q\}$ that could possibly be contaminated by scenario a . The design objective is then expressed as:

$$\sum_{a \in \mathcal{A}} \alpha_a \sum_{i \in \mathcal{L}_a} d_{ai} x_{ai},$$

where x_{ai} is an indicator variable with value equal to 1 if location i raised the alarm (i.e., first detected contaminant) for contamination scenario a and 0 otherwise.

Our complete dynamic model formulation – which we denote by DSP – is easily expressed as the following MIP:

$$\begin{aligned} \text{(DSP) minimize} \quad & \sum_{a \in \mathcal{A}} \alpha_a \sum_{i \in \mathcal{L}_a} d_{ai} x_{ai} \\ \text{where} \quad & \begin{cases} \sum_{i \in \mathcal{L}_a} x_{ai} = 1 & \forall a \in \mathcal{A} \\ x_{ai} \leq s_i & \forall a \in \mathcal{A}, i \in \mathcal{L}_a \\ \sum_{i \in L} s_i \leq p \\ s_i \in \{0, 1\} & \forall i \in L \\ 0 \leq x_{ai} \leq 1 & \forall a \in \mathcal{A}, i \in \mathcal{L}_a \end{cases} \end{aligned}$$

The binary decision variable s_i for each potential sensor location $i \in L$ equals 1 if a sensor is placed at location i and 0 otherwise. The first set of constraints assures that exactly one sensor raises the alarm for each contamination scenario. The second set requires that a location cannot raise an alarm unless there is an installed sensor. The last constraint enforces the limit on the total number of sensors.

A special case of DSP was first described by [34]. Remarkably, the DSP is identical to the well-known p -median facility location problem [155]. In the p -median problem, p facilities (e.g., central warehouses) are to be located on m potential sites such that the sum of distances d_{aj} between each of n customers (e.g., retail outlets) a and the nearest facility j is minimized. In contrasting the DSP and p -median problems, we observe equivalence between (1) sensors and facilities, (2) contamination scenarios and customers, and (3) contamination impacts and distances. While the DSP allows placement of *at most* p sensors, p -median formulations generally enforce placement of all p facilities; in practice, the distinction is irrelevant unless p approaches the number of possible locations.

Finally, we note that a slightly revised formulation of DSP is actually used in our computational experiments. We have observed that for any given contamination scenario a , there are often many total impacts d_{aj} that have the same value. If the contaminant reaches two junctions at approximately the same time, then these two junctions could witness the contamination event with the same impact values. For example, this occurs frequently when a coarse reporting time-step is used with the water quality simulation. This observation has led us to consider the following reformulation of DSP:

$$\begin{aligned}
(\text{cDSP}) \quad & \text{minimize} \quad \sum_{a \in \mathcal{A}} \alpha_a \sum_{i \in \hat{\mathcal{L}}_a} d_{ai} x_{ai} \\
& \text{where} \quad \begin{cases} \sum_{i \in \hat{\mathcal{L}}_a} x_{ai} = 1 & \forall a \in \mathcal{A} \\ x_{ai} \leq s_i + \sum_{j \in \mathcal{L}_a \setminus \hat{\mathcal{L}}_a: d_{aj} = d_{ai}} s_j & \forall a \in \mathcal{A}, i \in \hat{\mathcal{L}}_a \\ \sum_{i \in \mathcal{L}} s_i \leq p & \\ s_i \in \{0, 1\} & \forall i \in \mathcal{L} \\ 0 \leq x_{ai} \leq 1 & \forall a \in \mathcal{A}, i \in \hat{\mathcal{L}}_a \end{cases}
\end{aligned}$$

where $\hat{\mathcal{L}}_a \subseteq \mathcal{L}_a$ such that $d_{ai} \neq d_{aj}$ for all $i, j \in \hat{\mathcal{L}}_a$. This revised formulation treats sensor placement locations as equivalent if their corresponding contamination impacts are the same for a given contamination event. In doing so, the fundamental structure of this formulation changes only slightly, but this IP may require significantly less memory (by eliminating duplicate d_{ai} values). However, it is important to note that cDSP and DSP have the same set of feasible solutions, so they can be used to find the same optimal sensor placements. In preliminary experiments, cDSP was often ten times smaller than DSP , and corresponding reductions in optimization runtime have been observed.

For simplicity of presentation, our subsequent discussion will refer to DSP when describing these two formulations. However, the cDSP is the actual MIP model used in our experiments.

5.3 Empirical Results

We now describe the application of MIP and heuristic solvers for DSP on a number of large-scale, real-world water distribution networks. We describe a heuristic search strategy in Section 5.3.1; our methodology and test networks are detailed in Section 5.3.2; results for MIP and heuristic approaches are described in Section 5.3.3 and Section 5.3.4, respectively.

5.3.1 Solution Methods

Equivalence with the p -median problem has an immediate bearing on our approach to solving DSP, because we can directly leverage the extensive literature on algorithms for solving the p -median problem. The p -median problem can in principle be solved with a MIP solver. Further, optimal integer solutions frequently result by relaxing the integral constraints and solving the corresponding pure linear program (LP) [175]. However, heuristics are often used in practice when dealing with large problem instances due to the rapid growth in the number of constraints and variables as problem size increases.

The current state-of-the-art heuristic for the p -median problem is a hybrid approach recently introduced by Resende and Werneck, which we denote RW . The core mechanism underlying RW is a Greedy Randomized Adaptive Search Procedure (GRASP), which is used to generate a set of high-quality solutions using biased greedy construction techniques. Steepest-descent hill-climbing is then used to move from each of the resulting solutions to a local optimum. Finally, path relinking is used to further explore the set of solutions lying at the intersection of the resulting local optima. For a complete description of RW , we refer the reader to [174].

5.3.2 Methodology and Test Problems

Each contamination scenario consists of a single EPANET mass injection event of strength 100 mg/L and duration 24 hours. The duration of the entire simulation is 96 hours. EPANET [179] is used to perform water quality simulations for each contamination scenario, and the resulting concentration time-series τ_j are used to compute the impact factors d_{aj} for each combination of $a \in \mathcal{A}$ and $v_j \in V$. Simulations begin at time $t_s^a = 0$ and the 96 hour duration covers multiple iterations of a daily demand cycle.

Although the objective function of DSP allows for meaningful contamination probabilities α_a , for simplicity our experiments address only the case in which $\alpha_a = 1/|\mathcal{A}|$. The impact values d_{ai} are obtained from water quality simulations performed by the EPANET toolkit, which has been instrumented for the objective of contaminant mass consumed. Other objectives, such as population exposed, number of failed detections, etc., are addressed by generating different impact numbers, not by changing the MIP.

We consider three real-world test networks, which we denote SNL-1, SNL-2, and SNL-3. These networks respectively contain roughly 400, 3000, and 12000 junctions, and 450, 4000, and 14000 pipes. Due to the magnitude of SNL-3, we consider only contamination scenarios at $t_s = 0$. The actual identities, exact dimensions, and pump/valve/tank/reservoir/well counts of these networks are withheld for security purposes. We observe that these models are *not* all-pipes models; the complexity is strictly due to size of the region served by the particular utilities from which the models were obtained. The numbers of nonzero demand junctions for these three networks are 105, 1621 and 9705. Networks SNL-1 and SNL-2 were solved with four contamination event start times $t_s = 0, 6, 12, \text{ and } 18$ (units are in hours), while network SNL-3 was solved with one start time at $t_s = 0$. Furthermore, the water quality reporting step for networks SNL-1 and SNL-2 was 5 minutes, but to address scalability issues, the water quality reporting step for network SNL-3 was 60 minutes.

SNL-3 is an order of magnitude larger than any previously considered in the sensor placement optimization literature, and SNL-1 is an order of magnitude larger than that typically investigated. The largest network considered in most analyses (e.g., see [125] and [164]) is the “Anytown U.S.A.” network [238], which consists of 34 pipes, 16 junctions, two tanks, one pump, and one well. [34] solve DSP with a MIP solver for on a network containing roughly 450 junctions and 600 pipes. [239] solve static MIP models with MIP solvers, using both the smaller 450 junction network in addition to a larger network with roughly 3500 junctions.

All experiments were conducted on a dual-processor 64-bit 2.2GHz AMD Opteron workstation with 20 GB of RAM and 60GB of total (RAM plus swap) memory. The execution of the water quality simulations to compute impact values required non-trivial amounts of computation. For SNL-1, SNL-2, and SNL-3, the respective mean times required to perform a water quality analysis for a single contamination scenario are approximately 0.75, 1.25, and 4 seconds using EPANET on our workstation. The run-times required to obtain the full suite of water quality simulations range from under an hour for SNL-1 to over 2 days for SNL-3. However, it is noteworthy that this computation could be easily parallelized across a set of standard workstations.

Test Instance	#Attacks	p	Linear Program Statistics			Performance Statistics	
			#Rows	#Columns	#Non-Zeros	Memory	Run-Time
SNL-1	420	20	10364	10354	9523	13Mb	0.88s.
SNL-2	6484	20	181K	184K	4.1M	534Mb	107.56s.
SNL-3	9705	20	218K	219K	1.4M	239Mb	912.62s.

Table 5.1. Computational results for MIP solution of each of the test networks.

5.3.3 Solution via Mixed-Integer Programming

We solve DSP problems with ILOG’s² AMPL/CPLEX 9.0 MIP solver, which is a state-of-the-art MIP solver. We compute optimal solutions to cDSP for each of our test networks for a range of sensor budgets, which were selected to be realistic examples of what might be used in practice. The computational results for these experiments are shown in Table 5.1. All MIPs for SNL-1 and SNL-2 solved without branching.

The MIP results shown in Table 5.1 do demonstrate a specific problem size for which cDSP cannot be practically solved with a standard workstation. However, in other experiments we have utilized our high performance 64-bit machine. For example, the solution of cDSP for SNL-3 with 4 contamination times per junction required roughly 2 hours of running time and 9 gigabytes of RAM. Exploring another scalability dimension, allowing 96 different contamination times per junction on a network with roughly 3500 junctions required roughly 4 days of CPU time and more than 20 gigabytes of RAM.

Another scalability issue concerns the fidelity of the data used for cDSP. In order to make the water quality simulations more generally useful, our data collection process involves an intermediate step in which the concentrations at each junction at each reporting step are saved. These files become prohibitively large when small reporting steps are used in conjunction with large models. In particular, in our experiments with SNL-3, our choice of a 60 minute reporting step reduced the space requirements for concentration data, but also increased the number of indistinguishable “first hits” at each reporting step.

5.3.4 Solution via the *RW* Heuristic

Next, we consider the performance of the *RW* heuristic on each of our test networks; the results are reported in Table 5.2. On both SNL-1 and SNL-2, *RW* executes in negligible run-times and requires at most modest amounts of memory. Further, the solutions generated by the heuristic are provably *optimal*; the impact is equivalent to that yielded by the exact MIP solvers, as obtained during the course of the experiments described in Section 5.3.3. Although not reported, we observe identical behavior on a range of sensor budgets. Relative to the MIP solver, results can be an order of magnitude or more faster, and can require less than 1/10th of the total memory. However, it is important to note that the heuristic cannot in isolation *prove* the optimality of its result.

On SNL-3, the *RW* heuristic generates a final solution in roughly 5 minutes, while requiring 2 GB of RAM.

²www.ilog.com.

		Performance Statistics	
Test Instance	p	Memory	Run-Time
SNL-1	20	8MB	< 1 s.
SNL-2	20	500 MB	31 s.
SNL-3	20	2 GB	300 s.

Table 5.2. Computational results for the *RW* heuristic on each of our test networks.

In contrast, the MIP approach to solving the same test network consumed half a GB of total memory in 10 minutes. The heuristic’s advantages are more pronounced on experiments with larger numbers of attacks. Furthermore, *RW* has always generated provably optimal solutions for our experiments on real networks.

5.4 Conclusions

The DSP/cDSP MIP formulation represents a natural evolution of combinatorial modeling for contaminant sensor placement. DSP is a particularly interesting MIP formulation because the value of solutions to this MIP are exactly the same as if the solution was evaluated independently. Consequently, with DSP we can disassociate issues related to combinatorial modeling and water quality analysis. For example, current modeling limitations with DSP are now principally due to the fidelity of the water quality simulations or invalid assumptions relating to the attack scenario, sensor behavior, or emergency response protocols.

Our experiments demonstrate that exact and heuristic solvers can be effectively applied to DSP instances that are at least an order of magnitude larger than problems commonly used in the water distribution community literature. Most research on sensor placement methods has focused on small-scale water distribution networks with at most 200 junctions and pipes. We have demonstrated that exact and heuristic solvers can compute optimal solutions to cDSP for networks with ten thousand junctions, with reasonable computational effort.

Our successful application of cDSP and p -median heuristics to DSP takes advantage of the mathematical structure in this problem to (a) significantly reduce the cost of evaluating sensor placement ensembles, (b) tailor the heuristic to the particular structure of this application, and (c) rigorously assess whether the value of the final solution is close to value of an optimal solution. This heuristic is qualitatively different from general-purpose simulation-based optimization methods that have been previously considered for sensor placement (e.g. [164]) by not treating the search strategy as an outer loop around the water quality analysis. Instead, we precompute the impact of contamination scenarios on the entire network, and then re-use these values to assess the quality of any ensemble of sensors. Additionally, our heuristic optimizer is specifically tailored to the p -median structure of DSP. Thus we can reasonably expect that it will outperform general-purpose heuristics, particularly because the p -median problem is well-studied. Finally, we have leveraged the fact that the LP-relaxation of the DSP MIP formulation can provide performance bounds for the final solution provided by the heuristic. Although many authors have claimed that their sensor placement heuristics are capable of locating optimal solutions, we can quantify how close to optimality our heuristic has achieved in a rigorous fashion, either by comparing with an optimal solution provided by a

MIP solver, or by comparing with an LP-bound. Taken together, these observations have enabled the effective application of the p -median heuristic to large-scale water networks that are 500 times larger than the largest water network considered by [164], who consider the application of a genetic algorithm to a similar dynamic sensor placement formulation.

Scalability challenges remain a critical research focus, even for these methods. To adequately account for important temporal effects, we may need to (1) use a large number of attack scenarios and (2) use a small water quality reporting step. These factors dramatically increase the size and difficulty of DSP. The number of attack scenarios determines the number of water quality simulations used to define a DSP instance. Furthermore, the water quality reporting step impacts the number of variables and constraints in the DSP formulation. Addressing these scalability issues will require the development of techniques to perform parallel simulations and to solve large instances of DSP. For example, we expect that methodology used to formulate cDSP could be generalized to formulate reduced-fidelity MIP models that can be solved much more efficiently.

Our experience with the RW algorithm suggests that this heuristic method is not as sensitive to these scalability challenges as the MIP solvers for DSP. Specifically, RW appears less sensitive to the number of attack scenarios used in a DSP formulation. However, we expect RW to be able to solve most large-scale problems using high-performance 64-bit workstations. Even in cases where the memory requirements are large, this methodology can trade-off runtime for maximum memory utilization. As detailed in [174], the large memory requirements are due to pre-computations that yield significant run-time improvements. Consequently, it is therefore possible to take the complementary approach and sacrifice run-time for reduced memory requirements.

Finally, we note that related research on more fundamental modeling issues is also needed to make the application of DSP effective in practice. For example, methods to address solution robustness [66], worst-case optimization objectives, and multiple-objective analysis [239] are needed. Practical assessment of sensor placement configurations will require the analysis of trade-offs between sensor placement objectives, as well as assessments of solution robustness to data variabilities.

Chapter 6

A Multiple-Objective Analysis of Sensor Placement Optimization in Water Networks

Jean-Paul Watson, Harvey J. Greenberg (University of Colorado at Denver), William E. Hart

6.1 Background

There is growing interest in the use of contaminant sensors to provide continual monitoring of water quality. Effective deployment requires a sensor placement that ensures an adequate coverage of the network's flow for detection and remediation of contaminants. Ideally, an optimal sensor placement is available, such that the deployment cost is minimized and the level of protection afforded is maximized.

There are numerous measures with which the efficacy of sensor placements can be quantified. For example, researchers have developed algorithms for optimizing sensor placements with respect to the following objectives: population exposed [31], volume of contaminated water consumed [125], and time to detection [131]. Although there has been some debate as to the "best" objective for sensor placement, we argue that in practice multiple objectives must be simultaneously considered when designing sensor grids.

In our analysis, we consider the following performance objectives, each quantifying some aspect of the damage incurred by the intentional release of a contaminate:

Population Exposed. The number of people exposed to the contaminant before detection by a sensor.

Time to Detection. The time between the attack and detection of the contaminate by a sensor.

Volume Consumed. The amount of contaminated water consumed by the population before detection by a sensor.

Number of Failed Detections. The proportion of attacks that are undetected by all sensors.

Extent of Contamination. The length of pipe contaminated by an attack.

For each objective, we formulate the associated optimization problem as a mixed-integer linear program and examine optimal placements on two real-world water distribution networks. We consider trade-offs between the various objectives, focusing on two fundamental questions: (1) What is the relationship between optimal solutions generated for different objectives? (2) How do these relationships impact the design of optimization algorithms for the sensor placement problem?

The rest of this paper is organized as follows. We describe our modeling assumptions in §6.2 and develop mixed-integer linear programming formulations for each of the five objectives. We then use these formulations to obtain optimal solutions for each individual objective and analyze their relationships in §6.3. Finally, the implications of our analysis are summarized in §6.4.

6.2 Problem Description and Mixed-Integer Formulation

We now describe the sensor placement optimization problem in detail and introduce mixed-integer linear programming (MILP) models of the problem with respect to each of the objectives described in §6.1. Our MILP models are based on the time-independent flow model introduced by Berry et al. [31]. We have selected this model despite known deficiencies, such as its inability to account for transitions between qualitatively different flow patterns. However, the simplicity of the time-independent model is the enabling factor underlying our analysis; optimal placement algorithms for more realistic models are currently not tractable for moderate-to-large distribution networks.

A water distribution network is modeled as a undirected graph $G = (V, E)$. The nodes (V) correspond to junctions, tanks, and reservoirs. The edges (E) correspond to pipes that connect a pair of distinct nodes. We assume network dynamics can be represented as a set of steady-state flow patterns, P , which we obtain from EPANET [179]. We let $|P| = 4$, such that flow patterns represent contiguous 6-hours periods during a day. The direction of flow across an edge is pattern-dependent; we denote the start (upstream) and finish (downstream) nodes of an edge during pattern p by $V^s(e, p)$ and $V^f(e, p)$, respectively. A *path* from v_i to v_j during p is a sequence of edges, e_{j_1}, \dots, e_{j_k} , such that: $V^s(e_{j_1}, p) = v_i$, $V^f(e_{j_k}, p) = v_j$, and $V^f(e_{j_l}, p) = V^s(e_{j_{l+1}}, p)$ for $l = 1, \dots, k-1$. If such a path exists, we say v_j is *reachable* from v_i during p , and let $\mathcal{R}^V(i, p) = \{j : v_j \text{ is reachable from } v_i \text{ during } p\}$. The set of reachable edges is defined as $\mathcal{R}^E(i, p) = \{e \in E : V^s(e, p) \in \mathcal{R}^V(i, p)\}$. With some abuse of notation, we let $\mathcal{R}^V = \{(i, j, p) : j \in \mathcal{R}^V(i, p)\}$ and $\mathcal{R}^E = \{(i, e, p) : e \in \mathcal{R}^E(i, p)\}$.

We assume a single attack occurs during exactly one flow pattern. We denote α_{ip} as the *attack weight* for $v_i \in V$, $p \in P$. Clearly, $\alpha_{ip} \geq 0$ and $\sum_{i,p} \alpha_{ip} = 1$.

The primary decision variables define where sensors are placed. Let $s^E(e) = 1$ if a sensor is placed at edge e , and $s^E(e) = 0$ otherwise. Similarly, let $s^V(v) = 1$ if a sensor is placed at node v ; otherwise, $s^V(v) = 0$. The following constraints then define the core of the MILP feasible region, independent of objective:

$$s^E \in \{0, 1\}^{|E|}, s^V \in \{0, 1\}^{|V|}, \sum_{e \in E} s^E(e) + \sum_{v \in V} s^V(v) \leq N_s \quad (6.2.1)$$

where N_s is the limit on the number of available sensors.

To formulate the *pe* and *ec* objectives, we introduce auxiliary decision variables that describe how contamination is propagated through the network. For $(i, j, p) \in \mathcal{R}^V$, let $x(i, j, p) = 1$ if node v_j is contaminated due to an attack at node v_i during pattern p ; otherwise, $x(i, j, p) = 0$. The constraints on $x(i, j, p)$ are given as follows:

$$\begin{aligned} x &\in \{0, 1\}^{|\mathcal{R}^V|}, x(i, i, p) = 1 \\ x(i, j, p) &\geq x(i, V^s(e, p), p) - s^V(V^s(e, p)) - s^E(e) \\ \forall (i, j, p, e) : &(i, V^s(e, p), p) \in \mathcal{R}^V \text{ and } V^f(e, p) = j \end{aligned} \quad (6.2.2)$$

The constraint $x(i, i, p) = 1$ represents the fact that node v_i is contaminated by an attack at node i during p . The third constraint requires node v_j to be contaminated if (1) it is a descendant under p of an attacked node v_i and (2) there is no sensor along some path from v_i to v_j . To see this, note that such a constraint forces $x(i, j, p) = 1$ if and only if the following conditions are met: (1) there is a predecessor node ($V^s(e, p)$) that is contaminated by the attack ($x(i, V^s(e, p), p) = 1$) and that has no sensor ($s^V(V^s(e, p)) = 0$); and (2) the connecting edge ($e = (v_k, v_j)$) has no sensor ($s^E(e) = 0$). If either of these conditions does not hold, $x(i, j, p)$ is not constrained to be 1, and the minimization will choose $x(i, j, p) = 0$ if all such constraints allow it. Finally, we note that an edge $e \in E$ is contaminated by an attack at node v_i during pattern p if $x(i, V^s(e, p), p) = 1$. Due to the structure of constraints (6.2.2), the contamination variables are implicitly binary.

Next, we formulate the *wtd*, *wvc*, and *nfd* objectives. An attack at v_i during p can be detected by any sensor on a downstream edge $e \in \mathcal{R}^E(i, p)$ or node $\mathcal{R}^V(i, p)$. However, given worst-case placements of sensors along an edge (i.e., near $V^f(e, p)$), we assume detection is effectively restricted to nodes. Let $DP(i, j, p) = 1$ if the detection point is at node v_j . If there are no sensors downstream of v_i , then the attack goes undetected, in which case we define $DP(i, 0, p) = 1$. If the attack is detected by a downstream sensor, we define $DT(i, p)$ as the interval required. For undetected attacks, we introduce a penalty term τ ; in our experiments, we let $\tau=1,440$, the number of minutes in a day. Finally, we let $E^f(v, p) = \{e : V^f(e, p) = v\}$, i.e., the set of edges whose finishnode is v during pattern p . The constraints on the detection point (DP) and time-to-detection (DT) variables are then given by:

$$DP \in \{0, 1\}^{|\mathcal{R}^V| + |V \times P|} \text{ and } \sum_{j \in \mathcal{R}^V(i, p)} DP(i, j, p) + DP(i, 0, p) = 1 \quad (6.2.3)$$

$$DT(i, p) = \sum_{j \in \mathcal{R}^V(i, p)} DP(i, j, p) t_{ijp} + DP(i, 0, p) \tau \quad (6.2.4)$$

$$DP(i, j, p) \leq s^V(v_j) + \sum_{k \in E^f(j, p)} s^E(k) \quad (6.2.5)$$

$$DP(i, 0, p) \leq 1 - \left(\sum_{j \in \mathcal{R}^V(i, p)} s^V(v_j) + \sum_{e \in \mathcal{R}^E(i, p)} s^E(e) \right) / N_s \quad (6.2.6)$$

where t_{ijp} is the minimum flow time between v_i and v_j during p .

Constraint (6.2.3) requires exactly one assignment for each attack, either to a detection point or to the '0' node in the case of a failed detection. Constraint (6.2.4) defines DT as either the travel time to a detection point or τ depending on the assignment. Constraint (6.2.5) forces $D(i, j, p) = 0$ when there is no downstream sensor, and constraint (6.2.6) disallows the assignment to the '0' node when a potential detection point exists. Following this, we denote the duration before detection over which a node v_j consumes contaminates injected at v_i during p by $CT(i, j, p)$, subject to the following constraints:

$$CT(i, j, p) \geq 0, CT(i, j, p) \geq DT(i, p) - t_{ijp}, \text{ and } CT(i, j, p) \geq \tau DP(i, 0, p) \quad (6.2.7)$$

Given the contamination (x), detection time (DT), and consumption time (CT) variables, the objectives introduced in §6.1 are easily expressed as follows:

$$\begin{aligned} \text{Population exposed (pe):} & \quad F_1 = \sum_{(i,j,p) \in \mathcal{R}^V} \alpha_{ip} \delta_{jp} x(i, j, p) \\ \text{Weighted time to detection (wtd):} & \quad F_2 = \sum_{(i \in V, p \in P)} \alpha_{ip} DT(i, p) \\ \text{Failed detections (nfd):} & \quad F_3 = \sum_{(i \in V, p \in P)} \alpha_{ip} D(i, 0, p) \\ \text{Weighted volume consumed (wvc):} & \quad F_4 = \sum_{(i \in V, p \in P)} \alpha_{ip} (\sum_{(i,j,p) \in \mathcal{R}^V} d_{jp} CT(i, j, p)) \\ \text{Extent of contamination (ec):} & \quad F_5 = \sum_{(i,e,p) \in \mathcal{R}^E} \alpha_{ip} \lambda_e x(i, V^s(e, p), p), \end{aligned}$$

where δ_{jp} denotes the number of people at v_j during p , d_{jp} denotes the demand (in gallons per minute) at node v_j during p , and λ_e denotes the length of the pipe (in feet) for edge e . Finally, the MILP corresponding to a particular objective is given simply as $\min F_i$ subject to the relevant subset of constraints (6.2.1)–(6.2.7).

While space prevents a comprehensive discussion of our formulations, we do briefly consider two points of note. The first involves the parameter τ , used in the computation of both the wtd and wvc objectives. Without τ , both quantities equal zero if we simply do not place any sensors. We could have rectified the situation by forcing all N_s sensors to be placed. However, this simply masks a fundamental modeling issue. Conceptually, τ represents the time required to detection without sensors, i.e., when exposure symptoms are observed in the population. Mathematically, wtd and wvc are really weighted sums of two more fundamental objectives: (1) the number of failed detections and (2) the time-to-detection and volume-consumed conditioned on the subset of attacks that were detected. Second, when given a choice, optimization will always prefer to place a sensor on $V^s(e, p)$ than on e itself, because e cannot be attacked directly and by moving the sensor upstream, any population at $V^s(e, p)$ is additionally protected.

N_s	<i>pe</i>		<i>wtd</i>		<i>wvc</i>		<i>nfd</i>		<i>ec</i>	
	Node	Edge	Node	Edge	Node	Edge	Node	Edge	Node	Edge
0	281.1	281.1	1440	1440	69013.5	69013.5	1.0	1.0	21244.4	21244.4
10	84.1	98.0	868.4	961.0	1095	3804.4	0.08	0.16	6412.2	7664.0
25	46.0	60.9	658.8	793.4	60.3	2049.7	0.001	0.07	3274.8	4677.3
50	24.6	39.0	480.0	695.8	12.3	1894.4	0.0	0.07	1561.2	2938.4
100	10.9	24.3	235.0	607.1	2.99*	1878.9	0.0	0.07	545.3	1803.9
200	3.81	15.3	44.2	544.7	0.26*	1877.1	0.0	0.07	117.1	1217.6
300	1.43	11.4	4.74	534.7	0.02	1876.6	0.0	0.07	15.3	1031.0
400	0.01	9.82	0.0	534.2	0.0	1876.6	0.0	0.07	0	962.1

Table 6.1. Optimal values for various performance objectives under both node-only and edge-only sensor placements for a range of sensor budgets N_s on the SWU network.

6.3 Experimental Results

We now use the MILP models introduced in §6.2 to analyze the trade-offs between various performance objectives when optimizing sensor placements for two real-world water distribution networks. We codify the MILPs using the AMPL modeling language [97], which are then solved using ILOG’s AMPL/CPLEX 9.0 commercial package.

6.3.1 The Test Networks

We report computational results for models of two real water distribution networks. The first network model was provided by a small local water utility, and contains roughly 450 junctions and 600 pipes; we denote this network by SWU. The second network model, denoted MET, was obtained from a utility responsible for a medium-sized US metropolis, and contains roughly 3,500 junctions and 3,800 pipes. The α_{ip} and δ_{ip} for the SWU model were estimated by personnel familiar with the network. The size of the city corresponding to the MET network prevents a similar analysis, and we were unable to obtain recent census track data. Consequently, the δ_{ip} for the MET model are consistent with the aggregate population statistics, while the α_{ip} are randomized.

6.3.2 Node versus Edge Sensor Placements

Recall from §6.2 that although our MILP models allow sensors to be placed at both nodes and edges, placements are generally restricted to vertices in optimal configurations. In real applications, any preference of node versus edge placements is necessarily influenced by external factors such as physical accessibility and sensor operation regimes. However, in the absence of such detailed information, we can only assess the relative impact of a given preference on overall system performance.

For the SWU network, we computed optimal values for each combination of performance objective and the sensor budgets $N_s = \{0, 10, 25, 50, 100, 200, 300, 400\}$, subject to both node-only and edge-only

placements; the latter are respectively achieved by adding the constraints $\sum_{v \in V} s^V(v) = 0$ and $\sum_{e \in E} s^E(e) = 0$ to the MILP models. The results, reported in Table 6.3.2, clearly indicate that for any combination of performance objective and N_s , optimal edge-only sensor placements are always worse than edge-only placements, and often by a significant margin. Further, edge-only placements asymptotically approach a non-zero lower bound in solution quality as N_s approaches $|V|$. In contrast, node-only placements yield zero-impact solutions once $N_s \approx 400$.

Placement of sensors at all nodes clearly yields a completely protected system, which is consistent with the observed asymptotic approach of each performance objective toward 0 as $N_s \rightarrow |V|$. In contrast, edge-only placements can never guarantee complete system protection. For the *nfd* objective, the non-zero lower bounds are due to an inability to protect “leaf” nodes, i.e., nodes whose out-flow is always 0. The *pe* objective is additionally hampered by the inability to protect populations at the attack nodes. Similar effects result in non-zero upper bounds for the remaining objectives, e.g., *wtd* is always > 0 in an edge-only placement due to the fact that contaminate must flow from the attack node through at least some portion of an adjoining pipe.

Although node-only placements are clearly preferred, we note that the computational cost associated with solving the corresponding MILP models is significantly higher than in the edge-only case. Our computing environment consisted of a 3.0 GHz Pentium IV workstation with 4 GB of RAM running the Linux 2.4 operating system. Under edge-only placements, solutions for the SWU network were obtained in no more than 24 minutes for any combination of performance objective and N_s , with a median solution time of under 2 minutes. In contrast, solution times under node-only placements frequently exceeded 30 minutes for some performance objectives. Further, there are some values of N_s for which the optimal value of *wvc* could not be determined by AMPL/CPLEX before memory was exhausted. The observed differences are even more pronounced on the MET network, e.g., we failed to compute optimal solutions under node-only placements for numerous combinations of performance objectives and N_s .

6.3.3 Characteristics of Individual Performance Objectives

Before analyzing the trade-offs between different performance objectives, we consider in more detail the qualitative nature of each objective in isolation, on both the SWU and MET networks. The analysis is warranted, as we have both introduced several novel performance objectives and are considering extant objectives in the context of significantly larger networks than previously reported.

We first consider the node-only results for the SWU network, as reported in Table 6.3.2. Independent of objective, a budget of only $N_s = 10$ is sufficient to yield *at least* a 40% reduction relative to the $N_s = 0$ scenario (observed for the *wtd* objective), although the difference is typically much larger. The *nfd* and *wvc* objectives require fewer than 100 sensors to achieve near-perfect protection, while roughly 200 sensors are required for the remaining objectives. In all cases, excellent protection can be achieved with a sensor budget N_s significantly less than than of the maximal value $|V|$.

In contrast to the node-only results, it is impossible – independent of N_s – to achieve perfect protection of the SWU network under edge-only sensor placements. For example, at least 2,000 gallons of contaminate are consumed and a minimum of $\approx 2/3$ of a mile of pipe is exposed. However, near-*maximal* protection, i.e., with respect to the asymptotic upper bound, can still be achieved with a modest sensor budget; in the

N_s	pe	wtd	wvc	nfd	ec
0	9248.7	1440	2627310	1.0	42577
25	3893.3	924.7	176051*	0.5126	16598.9
50	2988.6	772.2	94459*	0.3869	12273.9
100	2163.7	627.4	54921*	0.2767	8692.7
250	1305.5	447.7	30388*	0.1691	4905.4
500	865.2	347.7	22784*	0.1450	3097.8
750	679.5	311.1	20785*	0.1450	2371.6
1000	573.9	295.6	20050*	0.1450	1948.8
3803	275.4	276.6	19387	0.1450	1163.7

Table 6.2. Optimal values for various performance objectives under edge-only placements for a range of sensor budgets N_s on the MET network.

case of the nfd and wvc objectives, maximal or near-maximal protection is achieved with only 25 sensors. Thus, mirroring the node-only results, high levels of protection can be achieved with a sensor budget N_s significantly less than that of the maximal value $|E|$.

Due to computational limitations, we only consider edge-only placements for the MET network. The results are shown in Table 6.2. We were unable to compute optimal values for the wvc objective over the full range of N_s before memory was exhausted; entries followed by an '*' represent an upper bound on the optimal value. Although the asymptotic bounds indicate a surprisingly large impact in the best case, e.g., ≈ 570 persons exposed, the results are consistent with differences in magnitudes of the SWU and MET networks. Mirroring the edge-only results for the SWU network, the data clearly indicate that a proportionally small number of sensors can provide reasonable levels of protection. Similarly, a modest number of sensors can yield maximal or near-maximal levels of protection.

6.3.4 The Impact of Optimization on Complementary Objectives

Several researchers have argued for the use of specific performance objectives to develop sensor placements in water distribution networks, e.g., see [125] and [131]. Populations would undoubtedly prefer minimization of pe as the primary objective. In contrast, economic factors often influence decision-makers, leading to an argument for minimization, or at least consideration, of ec . In any case, no single view is likely to prevail, and planners will realistically have to understand the trade-offs between the various objectives. Ideally, the objectives of interest are highly correlated, such that optimal solutions with respect to one objective yield near-optimal solutions with respect to the others. Unfortunately, we do not observe this behavior on the networks we analyze, raising the strong possibility that similar behavior may be observed on a broader range of real water distribution networks.

We begin with an example, by investigating how minimization of the number of failed detections in the SWU network impacts solution quality with respect to the other objectives; we consider node-only placements. At $N_s = 25$, $nfd \approx 0.0$, such that nearly all attacks are eventually detected. However, the solution that yields an optimal value of nfd at $N_s = 25$ also results in $pe = 260.637$, which represents a 567% deviation from the optimal value of pe given $N_s = 25$. Even larger deviations are observed for ec and

N_s	<i>nfd</i>		<i>wtd</i>		<i>wvc</i>		<i>ec</i>	
	%	Absolute	%	Absolute	%	Absolute	%	Absolute
10	850.4	0.625	140.7	353.0	2805.7	1206.5	118.8	1206.5
25	51,300	0.512	159.5	391.4	32556.7	19577.6	126.5	868.2
50	N/A	0.342	177.9	373.9	77280.0	9535.6	182.7	1291.3
100	N/A	0.111	232.1	310.3	N/A	3348.9*	252.4	831.3
200	N/A	0.058	558.4	201.4	N/A	1891.1*	390.5	340.2
300	N/A	0.028	2272.6	102.9	N/A	639.2	1329.8	215.5
400	N/A	0.002	N/A	0.1	N/A	0.0	N/A	1.1

Table 6.3. Percentage and absolute deviations from optimal for various objectives given a *pe*-optimal solution; results are for the SWU network, under node-only placements.

wvc (1,347% and 9,452%, respectively), while the value of *wtd* is only 122% greater than optimal. Although such large deviations were unexpected, their underlying cause was determined via straightforward analysis. For example, minimization of *nfd* subject to a limited sensor budget tends to yield sensor placements near the sinks or leaves of the distribution network. By doing so, many upstream nodes are exposed, resulting in a large overall population impact.

Characterizing and analyzing the interactions between all of the objectives introduced in §6.1 is beyond the scope of this paper. Rather, we examine two illustrative cases: how optimization of *pe* and *wtd* respectively impact solution quality with respect to the complementary objectives. We first consider optimization of *pe* over a range of N_s on the SWU network, subject to node-only sensor placements. The results, shown in Table 6.3, are expressed in terms of both percentage and absolute deviations from the optimal objective values. Entries labeled 'N/A' indicate the percentage was either undefined or extremely large, due to an optimal objective value of 0 or ≈ 0 , respectively. As in Table 6.3.2, an '*' indicates the optimal value was not computable, such that the entry represents an lower bound on the actual deviation. The data support two general conclusions. First, *pe* is not significantly correlated with any of the other objectives; with the exceptions noted below, percentage and absolute deviations from the optimal values of the complementary objectives are unexpectedly and uniformly large. Second, and most surprisingly, small-to-moderate increases in N_s do *not* generally improve the correlation. For *wtd*, *wvc*, and *ec*, deviations remain significant until $N_s \approx 400$, i.e., until the network is saturated with sensors. The sole exception occurs for *nfd*, where near-optimal values are achieved at $N_s = 300$.

Next, we consider optimization of *wtd* over a range of N_s on the MET network, subject to edge-only sensor placements; the results are shown in Table 6.4. As with optimization of *pe* on the SWU network, optimization of *wtd* typically yields sub-optimal values with respect to the complementary objectives. The most significant deviations occur for the *pe* and *ec* objectives. The *wvc* deviations quickly diminish as $N_s \geq 100$, while *nfd* is consistently near-optimal across the range of N_s ; the latter represents the strongest correlation observed for any two objectives for either test network.

Overall, our results support three general conclusions. First, there are significant risks associated with optimization of sensor placements with respect to any particular objective. The results shown in Tables 6.3 and 6.4 clearly demonstrate that optimal solutions with respect to any given performance objective can be far from optimal with respect to a range of complementary objectives. Second, and counter-intuitively, the

N_s	<i>nfd</i>		<i>pe</i>		<i>wvc</i>		<i>ec</i>	
	%	Absolute	%	Absolute	%	Absolute	%	Absolute
25	4.04	0.02	123.9	4824.04	149.33*	262900	146.1	24259.2
50	7.67	0.03	172.38	5151.79	375.02*	30527*	213.0	26141.6
100	15.05	0.04	250.82	5425.31	47.28*	25968*	309.7	26925.1
250	26.65	0.05	298.23	3893.51	23.03*	9079*	382.8	18777.2
500	12.38	0.02	259.28	2243.26	14.62*	3332*	354.3	10974.2
750	6.63	0.01	150.72	1024.12	6.73*	1398*	231.4	5488.2
1000	4.46	0.01	98.55	565.57	3.56*	714*	149.8	2973.9

Table 6.4. Percentage and absolute deviations from optimal for various objectives given a *wtd*-optimal solution; results are for the MET network, under edge-only placements.

lack of significant correlation between objectives may not improve with small-to-moderate increases in N_s . In other words, a large sensor budget does not necessarily mitigate the risk associated with optimization with respect to a single performance objective. Third, the nature of the correlation between various objectives is highly problem-dependent, suggesting that a comprehensive analysis is required on a per-application basis.

6.3.5 Characterizing the Trade-offs Between Competing Objectives

A potential source of risk associated with interpreting the results presented in Section 6.3.4 is the sensitivity to small deviations from the optimal values of the primary performance objectives. Specifically, it may be possible to achieve significant improvements in the complementary objective values by allowing a relatively small deviation from optimality in the primary performance objective. To test this hypothesis, we consider the relationship between the *pe* and *wtd* objectives on the SWU network, subject to edge-only placements with $N_s = 10$. The minimal value of *pe* is 98, which yields $wtd = 1251.64$; the latter represents a 140.7% deviation from the optimal value of 961. However, if we minimize *wtd* subject to the constraint $pe \leq 115$ – a 17% deviation from optimal – we obtain $wtd = 976.27$, which is less than 2% from optimal. Similar trade-offs are observed in all other pairs of performance objectives.

Due to space limitations, we omit a broader analysis of the trade-offs between the various objectives. Rather, we simply observe that by sacrificing solution quality with respect to a primary performance objective, it is generally possible to obtain significant improvements with respect to secondary objectives. In other words, optimal solutions with respect to individual performance measures are typically brittle, indicating that both planners and developers should shift their focus toward multiple-objective analysis and design of algorithms for the sensor placement optimization problem.

6.4 Conclusions

To date, all research on sensor placement optimization in water distribution networks presupposes a given, fixed design objective. Several disparate objectives have been proposed, and there are associated arguments — both implicit and explicit — for why one particular objective should be preferred over another. Yet, preference for any fixed objective is potentially risky, given the current lack of understanding as to the relationships among the proposed objectives. We have characterized the inter-dependencies among a range of optimization objectives on two real-world water distribution networks. The majority of these objectives are uncorrelated, in that optimal solutions with respect to any one objective are often highly sub-optimal with respect to complementary objectives. Further, increasing the number of sensors frequently fails to improve the correlation. However, the risks can be mitigated by considering solutions that are sub-optimal with respect to all performance objectives, which in turn requires a thorough understanding of how different objectives are related. Overall, the implications of our results for both researchers and planners is clear: robust algorithms for the sensor placement problem must carefully and simultaneously consider multiple design objectives.

Chapter 7

Robust Optimization of Contaminant Sensor Placement for Community Water Systems

Robert D. Carr, Harvey J. Greenberg (University of Colorado), William E. Hart, Goran Konjevod (Arizona State University), Erik Lauer (University of New Mexico), Henry Lin (University of California at Berkeley), Tod Morrison (University of Colorado), Cynthia A. Phillips

7.1 Background

Combinatorial optimization techniques need to address modeling and data uncertainties in many applications. As early as the 1950s, Dantzig [82] introduced stochastic programming to deal with *aleatory uncertainty*, which describes the inherent variation associated with the system being modeled [114, 161, 185].

More recently, researchers have developed robust optimization methods [129] to deal with *epistemic uncertainty*, which describes our lack of knowledge about information in our model [114, 161, 185]. For example, a common assumption is that the coefficients in the objective function of the problem are uncertain in the sense that they each can assume any value within a finite interval. Interval data occurs often in practice [92, 129], and when quantitative parameters have a subjective nature, interval values can be used interactively to provide more intuition about the model.

Robust optimization methods generally seek a solution that minimizes some measure of worst performance with respect to the uncertainty in the data. Commonly studied criteria for robust optimization are *absolute robustness* (or minimax), and *robust deviation* (or minimax regret). Yaman, Karason and Pinar [245] survey recent research for these methods and observe that many absolute robust formulations of problems with interval uncertainties can be solved with little more difficulty than the deterministic case. Bertsimas

and Sim [36, 37] adopt the interval model of uncertainty and consider a restricted version of the absolute robustness criterion. This model of robustness limits the conservativeness of the robust solution by arguing that it is quite unlikely that all data elements will assume their worst possible values simultaneously; both the absolute robustness and robust deviation criteria may find solutions that have this property. Furthermore, solving a 0-1 mixed-integer linear program (MILP) under this model is no more difficult than solving the original problem.

In this chapter we consider a version of the absolute robustness criterion that is naturally restricted by properties of the uncertain data. Specifically, we consider the case where the uncertain coefficients sum to a constant value. This restricted absolute robustness criterion is motivated by our recent work with MILP formulations for sensor placement in water distribution networks [31, 34, 240]. These MILPs rely on information like attack probabilities and water consumption statistics that are difficult to assess in detail, but for which we have good aggregate estimates. For example, water utilities have little information about the water consumption within a given household at a given hour, but they have accurate information about total water consumption within the entire water distribution network.

We analyze three cases of this restricted absolute robustness criterion that are motivated by this water security application:

- **unweighted uncertainty:** the objective has the form $\sum_{ij} \alpha_i x_{ij}$, for uncertain coefficients α_i ;
- **linearly weighted uncertainty:** the objective has the form $\sum_{ij} \alpha_i p_j x_{ij}$, for uncertain coefficients α_i (p_j known with certainty); and,
- **bilinearly weighted uncertainty:** the objective has the form $\sum_{ij} \alpha_i \delta_j x_{ij}$, for uncertain coefficients α_i and δ_j .

We argue that the constant-sum constraints on the uncertain parameters make the solution to these problems more realistic than formulations with a simple absolute robustness criterion. Furthermore, these problem formulations do not require a user-defined parameter to restrict the data uncertainties, so the robustness trade-off in this problem is more intrinsic than the trade-offs considered in the restricted absolute robustness models developed by Bertsimas and Sim [36, 37].

The rest of this chapter is organized as follows. Section 7.2 describes and motivates the three robust MILP formulations. The subsequent three sections analyze these models and present some preliminary computational experience using them. We conclude with a discussion of avenues for further research. The mathematical programming terms are generally defined here as needed, but one can consult the *Mathematical Programming Glossary* [101].

7.2 Motivation for Robust MILP Models

Recent terrorist attacks have heightened concerns about whether community water systems are sufficiently well protected to ensure a safe and reliable supply of drinking water in the United States and around the world. Consequently, there is growing interest in the use of contaminant sensors to provide ongoing

monitoring of water quality. A good sensor placement maximizes the information available for contamination control and remediation across a wide range of possible contamination scenarios, so that the fewest users consume contaminated water. A variety of MILP formulations have been developed to identify good sensor placement configurations [31, 34, 141, 240]. Berry et al. [31, 34] have recently solved moderately large MILP models of sensor-placement problems.

We model an attack as the release of a large volume of harmful contaminant at a single junction in the network. For any particular attack, we assume that all points downstream of the release point can be contaminated. Let \mathcal{R} be the set of pairs of junctions (i, j) such that junction j is downstream of junction i .¹ The primary decision variables for optimization are where to place each of a given number of sensors. Secondary binary decision variables are x_{ij} , for $(i, j) \in \mathcal{R}$, where $x_{ij} = 1$ if a contaminant injected at junction i can reach junction j without passing a sensor. (As will be evident, these secondary variables are completely determined by the sensor placements, but they appear as decision variables from a computational view.) Let X denote the set of feasible 0-1 x -values.

Consider the following parameter vectors over the nodes: (1) α_i is the probability of an attack at junction i , and (2) δ_i is the number of people who consume water at junction i . Note that α is not a probability in the classical sense and is sometimes called an *attack weight*. It is estimated from expert judgement about the vulnerabilities in the network. We estimate δ from census data. If the node i represents a contracted sub-network, then δ_i is the sum of the estimated population numbers for all nodes in the sub-network. We assume that all people at a contaminated node are potentially exposed.

The following two problems illustrate the data uncertainties that arise in these applications:

1. Minimize the expected extent of network contamination, as defined by the number of pipe junctions that become contaminated

$$\text{NC} : \min_{x \in X} \sum_{(i,j) \in \mathcal{R}} \alpha_i x_{ij},$$

2. Minimize the expected population exposed

$$\text{PE} : \min_{x \in X} \sum_{(i,j) \in \mathcal{R}} \alpha_i \delta_j x_{ij}.$$

In practice, we do not know the values of α and δ . Although water utilities can accurately estimate the total population served by their water distribution network, most utilities do not currently maintain detailed statistics about the fraction of the population that is consuming water at each junction. Similarly, risk assessment methodologies provide a coarse assessment of attack weights.

Let $\hat{\alpha}$ and $\hat{\delta}$ denote specified values of α and δ , respectively. These are sometimes called the *central values* (viz., most likely values, or best estimates). The central MILP is thus: $\min_{x \in X} \sum_{(i,j) \in \mathcal{R}} \hat{\alpha}_i \hat{\delta}_j x_{ij}$. As we let α and δ deviate from their central values, their possible values are limited by *constant-sum constraints*, $\sum_i \alpha_i = \sum_i \hat{\alpha}_i = 1$ and $\sum_j \delta_j = \sum_j \hat{\delta}_j$, respectively.

¹To simplify our presentation, we assume a stable flow pattern for water in this paper. In particular, the models that we describe do not explicitly account for temporal effects. Berry et al. [31, 34] describe more detailed IP models.

In the next two sections, we consider robust formulations for an absolute robustness criterion that is restricted in this sense. For objectives like NC, we show that the solution to a specific class of restricted absolute robustness problems is exactly the solution to the central MILP. More complex objectives, like PE, contain terms with products of uncertain parameters $(\alpha_i \delta_j)$. The PE model is less complex if population values are presumed known, which is what we consider in Section 7.3.

We use the following notation in the next sections to define the domain of the robust optimization problems:

$$\mathcal{B}(\hat{c}, L, U) = \{c : L \leq c \leq U, \sum_k c_k = \sum_k \hat{c}_k\},$$

where we suppose $L \leq \hat{c} \leq U$. The set $\mathcal{B}(\hat{c}, L, U)$ defines a multidimensional interval of uncertainty about a central value, \hat{c} , where c can be α or δ .

7.3 Linearly Weighted Uncertainty

Consider the PE problem with known population values at the nodes. The following robust optimization formulation applies our restricted absolute robustness criteria:

$$\min_{x \in X} \max_{\alpha \in \mathcal{B}(\underline{\alpha}, \underline{\alpha}, \bar{\alpha})} \sum_{(i,j) \in \mathcal{R}} \alpha_i \delta_j x_{ij}, \quad (7.3.1)$$

where δ is presumed known. Since each uncertain term is weighted by a constant value, we say that this absolute robustness criteria uses *linearly weighted uncertainty*.

We can apply duality to reformulate problem (7.3.1) as follows:

$$\begin{aligned} \min \pi + \mu \bar{\alpha} - \lambda \underline{\alpha} \quad & : \quad x \in X \\ & \lambda, \mu \geq 0 \\ \pi + \mu_i - \lambda_i = & \sum_{j \in J(i)} \delta_j x_{ij} \text{ for all } i, \end{aligned}$$

where $J(i) = \{j : (i, j) \in \mathcal{R}\}$. The dual variable π is associated with the primal constant-sum constraint, and λ, μ are associated with the lower and upper bounds, respectively.

Thus, we can cast a linearly weighted robust optimization as a single MILP, having replaced the max with min. In particular, this is an augmented MILP formulation, which simply includes an extended objective and some additional side-constraints on dual variables from the maximization subproblem.

Alternatively, instead of casting problem (7.3.1) as one MILP, we can decompose it and solve the inner maximization problem to obtain $\alpha^*(x)$ for each x in the outer minimization. The inner maximization problem can be solved simply by sorting the coefficients, which requires no more than $O(|\mathcal{R}| \ln |\mathcal{R}|)$ time. This may be computationally more efficient than the integrated formulation.

7.4 Unweighted Uncertainty

Consider the NC problem with interval uncertainties on the attack probabilities. The following robust optimization formulation applies our restricted absolute robustness criteria:

$$\min_{x \in X} \max_{\alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \bar{\alpha})} \sum_{(i,j) \in \mathcal{R}} \alpha_i x_{ij}. \quad (7.4.2)$$

We can solve this problem using the methods described in Section 7.3 (letting $\delta_j = 1$ for all j), but in this section we consider the restricted case where we have intervals of uncertainty that are proportional to the central value vector. Let $\mathcal{P}(\hat{c}, \varepsilon) = \mathcal{B}(\hat{c}, (1 - \varepsilon)\hat{c}, (1 + \varepsilon)\hat{c})$ for $\varepsilon \in [0, 1)$. Given this restricted notion of interval uncertainty, we prove that the sensor placement decision for the central attack weight values $\hat{\alpha}$ remains optimal for any allowed variation.

Let $Q(\varepsilon)$ denote a generalized robust optimization problem:

$$\min_{x \in X} \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx,$$

where X is any subset of binary vectors. The following theorem demonstrates that the solution to the central MILP (where \hat{c} is the coefficient of x) is the solution to a robust formulation that allows percentage deviations within a constant proportion of its central value. Consequently, no additional computational effort is needed to generate a robust solution for these problems.

Theorem 7.4.1. *Let $\varepsilon \in [0, 1)$. Then, x^* is an optimal solution to $Q(0)$ if, and only if, x^* is an optimal solution to $Q(\varepsilon)$.*

Proof. We begin with some notation and general observations. Let $S = \sum_j \hat{c}_j$. Let $\sigma(x) = \{j : x_j \neq 0\}$ (called the ‘‘support set’’ of x). Also, let $\mathbf{1}$ denote the vector of all ones: $(1, 1, \dots, 1)^\top$. The following identities follow from the definitions of S and σ : $\hat{c}x = \sum_{j \in \sigma(x)} \hat{c}_j = S - \sum_{j \notin \sigma(x)} \hat{c}_j$, and $\hat{c}(\mathbf{1} - x) = S - \hat{c}x = \sum_{j \notin \sigma(x)} \hat{c}_j$. Let $L = (1 - \varepsilon)$ and $U = (1 + \varepsilon)$.

The dual of $\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx$ is

$$\begin{aligned} \min \pi S + U\mu\hat{c} - L\lambda\hat{c} : \lambda, \mu \geq 0, \\ \pi + \mu_j - \lambda_j = x_j \text{ for all } j = 1, \dots, n. \end{aligned}$$

The dual variable π is associated with the constant-sum constraint, and λ, μ are associated with the lower and upper bound constraints on c , respectively.

Let x^0 be an optimal solution to $Q(0)$ and let x^ε be an optimal solution to $Q(\varepsilon)$. Our proof divides into two cases, depending on whether $\hat{c}x^0$ is greater or less than $\frac{1}{2}S$.

Case 1. $\hat{c}x^0 \geq \frac{1}{2}S$.

Consider the dual solution $\pi = 1$, $\mu = 0$, and $\lambda^\top = \mathbf{1} - x^0$. This is dual-feasible, where $\lambda \geq 0$ because $x^0 \leq \mathbf{1}$. The dual objective value is

$$\pi S + U\mu\hat{c} - L\lambda\hat{c} = S - L\hat{c}(\mathbf{1} - x^0) = S - L(S - \hat{c}x^0) = \varepsilon S + L\hat{c}x^0.$$

Therefore, we have

$$\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^0 \leq \varepsilon S + L\hat{c}x^0. \quad (7.4.3)$$

Now we define $c_j^\varepsilon = L\hat{c}_j$ for $j \notin \sigma(x^\varepsilon)$. Since we assume that $\hat{c}x^0 \geq \frac{1}{2}S$, it follows that $\hat{c}x^\varepsilon \geq \frac{1}{2}S$, which implies that $\hat{c}(1 - x^\varepsilon) \leq \frac{1}{2}S$. Consequently, we have

$$\begin{aligned} c^\varepsilon x^\varepsilon &= S - \sum_{j \notin \sigma(x^\varepsilon)} c_j^\varepsilon \\ &= S - L \sum_{j \notin \sigma(x^\varepsilon)} \hat{c}_j \\ &= S - L(S - \hat{c}x^\varepsilon) = \varepsilon S + L\hat{c}x^\varepsilon, \end{aligned}$$

which gives us the bound:

$$\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon \geq \varepsilon S + L\hat{c}x^\varepsilon. \quad (7.4.4)$$

Using (7.4.3) and (7.4.4), we then obtain the following chain of inequalities:

$$\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon \geq \varepsilon S + L\hat{c}x^\varepsilon \geq \varepsilon S + L\hat{c}x^0 \geq \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^0 \geq \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon.$$

Thus, equality must hold throughout. This establishes the following two results:

$$\begin{aligned} \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^0 &= \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon \quad (\text{first} = \text{last expression}) \\ \hat{c}x^0 &= \hat{c}x^\varepsilon \quad (\text{second} = \text{third expression and } L > 0), \end{aligned}$$

which completes this case.

Case 2. $\hat{c}x^0 \leq \frac{1}{2}S$.

The dual objective value of any dual-feasible solution is an upper bound on the primal value, cx^0 . Choose $\pi = 0$, $\mu^\top = x^0$, and $\lambda = 0$. This is clearly dual-feasible, and its dual objective value is $U\hat{c}x^0$. Therefore,

$$\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^0 \leq U\hat{c}x^0. \quad (7.4.5)$$

Now consider the value of $\hat{c}x^\varepsilon$. Suppose $\hat{c}x^\varepsilon \leq \frac{1}{2}S$. Then define $c_j^\varepsilon = U\hat{c}_j$ for $j \in \sigma(x^\varepsilon)$, and note that $c^\varepsilon \in \mathcal{P}(\hat{c}, \varepsilon)$. This is feasible (i.e., $c^\varepsilon \in \mathcal{P}(\hat{c}, \varepsilon)$) because $c^\varepsilon x^\varepsilon \leq \frac{1}{2}S$. It follows that $c^\varepsilon x^\varepsilon = U\hat{c}x^\varepsilon$, so we have $\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon \geq U\hat{c}x^\varepsilon$. On the other hand, suppose if $\hat{c}x^\varepsilon > \frac{1}{2}S$. Then, define $c_j^\varepsilon = L\hat{c}_j$ for $j \notin \sigma(x^\varepsilon)$, and note that $c^\varepsilon \in \mathcal{P}(\hat{c}, \varepsilon)$. It follows from our analysis in Case 1 that $\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon \geq \varepsilon S + L\hat{c}x^\varepsilon$. Taken together, this gives us the bound:

$$\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon \geq \min \{U\hat{c}x^\varepsilon, \varepsilon S + L\hat{c}x^\varepsilon\}. \quad (7.4.6)$$

Using Equations (7.4.5) and (7.4.6), we then obtain the following chain of inequalities:

$$\begin{aligned} \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon &\geq \min \{U\hat{c}x^\varepsilon, \varepsilon S + L\hat{c}x^\varepsilon\} \geq \min \{U\hat{c}x^0, \varepsilon S + L\hat{c}x^0\} \\ &= U\hat{c}x^0 \geq \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^0 \geq \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon. \end{aligned}$$

The equality in this chain follows from our assumption that $\hat{c}x^0 \leq \frac{1}{2}S$. We conclude that equality must hold throughout, and $\max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^0 = \max_{c \in \mathcal{P}(\hat{c}, \varepsilon)} cx^\varepsilon$. Furthermore, this shows that $U\hat{c}x^0 = \min\{U\hat{c}x^\varepsilon, \varepsilon S + L\hat{c}x^\varepsilon\}$ (fourth expression = second), so either $U\hat{c}x^0 = U\hat{c}x^\varepsilon$ or $U\hat{c}x^0 = \varepsilon S + L\hat{c}x^\varepsilon$. In the former case, we have immediately that $\hat{c}x^0 = \hat{c}x^\varepsilon$. In the latter case, we have the following chain of inequalities:

$$U\hat{c}x^0 \leq \varepsilon S + L\hat{c}x^0 \leq \varepsilon S + L\hat{c}x^\varepsilon = U\hat{c}x^\varepsilon,$$

from which it follows that $\hat{c}x^0 = \hat{c}x^\varepsilon$. Consequently, we conclude $\hat{c}x^0 = \hat{c}x^\varepsilon$. \square

In terms of the sensor placement problem, this result implies that we can solve problem (7.4.2) by solving the central MILP,

$$\min_{x \in X} \sum_{(i,j) \in \mathcal{R}} \hat{\alpha}_i x_{ij},$$

because every optimal solution remains optimal for any variation allowance on the attack weights that are bounded by a common proportion of the central value, $\hat{\alpha}$. This is because the α -maximization increases the objective by a *proportion* of S , independently of x . This is what is revealed in the proof and highlights the simplicity of the robust model.

While we have let attack weights (α) be the uncertain parameters, we could let it be population (δ) if we assume a uniform distribution on attack location. This is the case when there is no risk analysis, and one fixes $\alpha_i = \frac{1}{n}$ for all $i = 1, \dots, n$, where n is the number of nodes. In that case we also have the unweighted model, but the meaning of the objective changes to the max-expected population contamination.

Following Yaman et al. [244], this result may be called a *permanent solution*. They found a spanning tree that remains optimal within interval data; we have a sensor placement that remains optimal under fixed-proportionate interval data and a constant-sum constraint. In our case, the fixed proportion is necessary — Theorem 7.4.1 is not true if we consider the more general set of uncertainties defined by $\mathcal{B}(\hat{c}, \underline{c}, \bar{c})$.

7.5 Bilinear Weighted Uncertainty

Consider the PE problem with interval uncertainties on both the attack weights and population. Although the total population remains fixed, in practice we may not have complete knowledge of the population's geographic distribution. Consequently, there may be uncertainties in the values of the δ_i . The following robust optimization formulation applies our restricted absolute robustness criteria considering uncertainties in both α and δ :

$$\min_{x \in X} \max_{\substack{\alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \bar{\alpha}) \\ \delta \in \mathcal{B}(\hat{\delta}, \underline{\delta}, \bar{\delta})}} \sum_{(i,j) \in \mathcal{R}} \alpha_i \delta_j x_{ij}. \quad (7.5.7)$$

We say that this absolute robustness criteria uses *bilinearly weighted uncertainty* because we have a bilinear maximization problem for the inner maximization.

This is a special case of the bilinear fractional program considered by Malivert [151]. The general problem is NP-hard, but this inner bilinear program has several simplifications. The main simplification is that the

polyhedron separates for the two sets of variables, and each polyhedron (the ball) is much simpler than the general case — just one equation with bounds on the variables.

In Section 7.5.1, we show that the inner bilinear optimization problem remains NP-hard with this special structure and even with further special structure related to sensor placement in water networks. Section 7.5.2 gives a straightforward algorithm that reaches a solution that need not be a global maximum. Section 7.5.3 gives a constant-approximation algorithm, whose error is proportional to the square of the radius of the ball (ϵ).

7.5.1 Complexity

In this section we prove that the inner bilinear optimization problem is NP-hard. We consider the restricted version of the problem:

$$\begin{aligned} \max_{\substack{\alpha \in \mathcal{P}(\hat{\alpha}, \epsilon) \\ \delta \in \mathcal{P}(\hat{\delta}, \epsilon)}} \sum_{(i,j) \in \mathcal{R}} \alpha_i \delta_j x_{ij}, \end{aligned} \quad (7.5.8)$$

where both intervals have the same ϵ , and x satisfies the following two properties of our system:

1. The water networks we consider are directed acyclic graphs (dags) — we cannot have $(i, j) \in \mathcal{R}$ and $(j, i) \in \mathcal{R}$.
2. x satisfies transitive closure — if there is a path from i to j with no sensors ($x_{ij} = 1$) and there is a path from j to k with no sensors ($x_{jk} = 1$), then there is a path from i to k with no sensors ($x_{ik} = 1$).

We further note our special structure:

3. The domain is separable, $\mathcal{P}(\hat{\alpha}, \epsilon) \times \mathcal{P}(\hat{\delta}, \epsilon)$.
4. Each domain is defined by simple bounds and one constant-sum constraint.

We prove NP-hardness by reduction from the clique problem. In particular, given a graph $G = (V, E)$ with $|V| = n$ and n even, we prove that one can determine whether this graph has an $n/2$ clique by solving a bilinear program with our special structure on a transitively closed dag.

Given G , we construct a bipartite dag $G' = (A \cup P, E')$ as follows. For each vertex $v_i \in V$, create two sets of nodes and define the centers $\hat{\alpha}$ and $\hat{\delta}$:

Attack nodes. $A = \bigcup_{i=1}^n A_i$, where $A_i = \{a_{ij} : 1 \leq j \leq n\}$ for $1 \leq i \leq n$, with $\hat{\alpha}_{a_{ij}} = 1$ and $\hat{\delta}_{a_{ij}} = 0$ for all i, j .

Population nodes. $P = \bigcup_{i=1}^n P_i$, where $P_i = \{p_{ij} : 1 \leq j \leq n\}$ for $1 \leq i \leq n$, with $\hat{\alpha}_{p_{ij}} = 0$ and $\hat{\delta}_{p_{ij}} = 1$ for all i, j .

(Defining the centers in this way requires the modification of the constant-sum constraint on α to n^2 , rather than 1. For the sake of keeping the notation simple, we invoke a simple scaling argument to allow this.) To build the arc set of G' , for each $v_i \in V$ we add arcs forming a complete directed bipartite subgraph between the associated attack and population nodes: put $(a_{ij}, p_{ik}) \in E'$ for $j = 1, \dots, n$ and $k = 1, \dots, n$. These are *structural edges* that relate vertices associated with the same node in G . Further, for each edge $(v_i, v_j) \in E$, we add arcs (a_{ij}, p_{ji}) and $(a_{ji}, p_{ij}) \in E'$. These are *graph edges* that reflect the structure of the given graph G (in which we are searching for an $n/2$ clique). There is at most one graph edge adjacent to any node in G' .

Thus, G' is a bipartite dag. Further, G' is transitively closed because all arcs go from A to P . The x of our bilinear problem is the $n^2 \times n^2$ adjacency matrix of an $n \times n$ bipartite graph. Let $M = [m_{ap}]$ be defined by $m_{ap} = 1$ if, and only if, $(a, p) \in E'$ (and $m_{ap} = 0$ otherwise) for all $a \in A$, $p \in P$. Finally, we set $\varepsilon = 1 - 1/n^3$ to complete the definition of the bilinear problem.

Figure 7.1 shows a 6-node graph G and the constructed graph G' with 72 nodes (6 attack and 6 population per v_i for $i = 1, \dots, 6$).

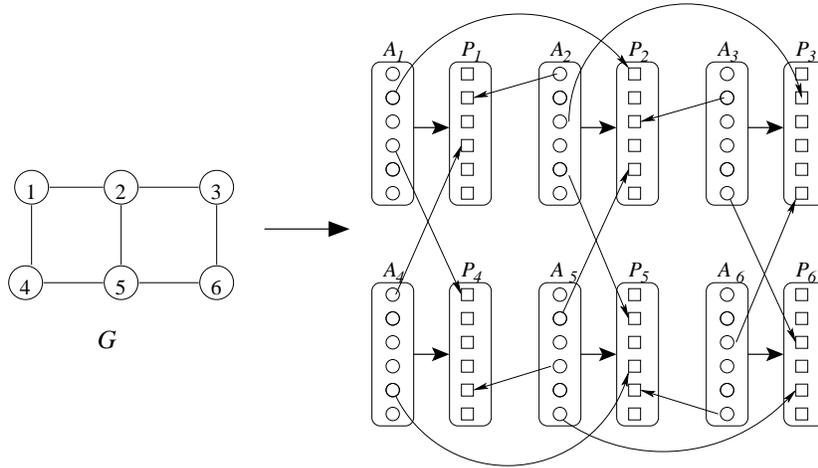


Figure 7.1. Example graph with induced constructed graph.

Summarizing the construction, we have two sets of n^2 nodes each, and two sets of arcs, all directed from $a \in A$ to $p \in P$. One set of arcs (structural edges) have the form (a_{ij}, p_{ik}) , so $x_{a_{ij}p_{ik}} = 1$ for all $v_i \in V$, $j, k = 1, \dots, n$. (In figure 7.1 they appear as one thick arc from A_i to P_i .) The other set of arcs (graph edges) have the form (a_{ij}, p_{ji}) and (a_{ji}, p_{ij}) for $(v_i, v_j) \in E$, so $x_{a_{ij}p_{ji}} = x_{a_{ji}p_{ij}} = 1$ for all $(v_i, v_j) \in E$. For all other $f, g \in A \cup P$, $x_{fg} = 0$. The bilinear program has the following objective value:

$$\begin{aligned}
 \sum_{(f,g) \in \mathcal{R}} \alpha_f \delta_g x_{fg} &= \sum_{(a,p) \in E'} \alpha_a \delta_p x_{ap} \\
 &= \sum_{(a,p) \in E'} \alpha_a \delta_p \\
 &= \sum_{v_i \in V} \sum_{j=1}^n \sum_{k=1}^n \alpha_{a_{ij}} \delta_{p_{ik}} \\
 &\quad + \sum_{(v_i, v_j) \in E} (\alpha_{a_{ij}} \delta_{p_{ji}} + \alpha_{a_{ji}} \delta_{p_{ij}})
 \end{aligned}$$

The first equality follows from the fact that $E' = \mathcal{R} \subset A \times P$, and the second equality follows from the fact that $x_{ap} = 1$ for $(a, p) \in E'$. The α and δ values missing from the expression above (viz., α_p for $p \in P$ and δ_a for $a \in A$) are required to be zero anyway, because we defined $\hat{\alpha}_p = 0$ and $\hat{\delta}_A = 0$. We prove that an optimal solution to our bilinear program over G' with $\varepsilon = 1 - 1/n^3$ answers the question of whether G has an $n/2$ clique.

As defined above, let M be the bipartite incidence matrix for G' , so its rows correspond to A and its columns to P . Because only attack nodes can have nonzero α and only population nodes can have nonzero δ , a feasible solution to the bilinear problem can redistribute α values only among the attack nodes (rows of M), and the δ values only among the population nodes (columns of M). In other words, α is constrained by $\mathcal{P}(\hat{\alpha}, \varepsilon)$ to satisfy $1 - \varepsilon \leq \alpha_{a_{ij}} \leq 1 + \varepsilon$ for each row a_{ij} of M . Similarly, δ is constrained by $\mathcal{P}(\hat{\delta}, \varepsilon)$ to satisfy $1 - \varepsilon \leq \delta_{p_{kl}} \leq 1 + \varepsilon$ for each column p_{kl} of M .

Given a feasible solution (α, δ) to the bilinear program, we say that a row or column of the matrix M is *selected* if respectively the α or δ value is $1 + \varepsilon$. We say that an entry of the matrix is selected if both its row and column are selected. We consider only those feasible solutions that set each variable to one of its bound values, so the number of rows (columns) selected is always $\frac{n^2}{2}$ to satisfy the constant-sum constraints. This is illustrated in figure 7.2.

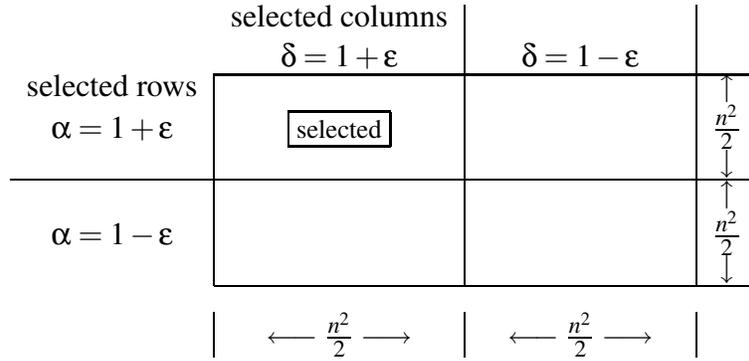


Figure 7.2. Partition of M upon selecting rows, columns, and elements.

Further, we say that a vertex $v_i \in V(G)$ is selected if all rows and all columns associated with it (all $a_{ij} \in A_i$ and $p_{ik} \in P_i$) are selected. Finally, v_i is *partially-selected* if at least one row or column associated with it is selected and at least one such row or column is not selected.

In what follows, let $K = \frac{1}{2}n^3 + \frac{n}{2}(\frac{n}{2} - 1)$.

Lemma 7.5.1. *If there is an $n/2$ clique in G , the optimal value of the bilinear program is at least $K(1 + \varepsilon)^2$.*

Proof. Assume there is an $n/2$ clique in G . Select all the vertices in the clique, so that $\alpha_{a_{ij}} = \delta_{p_{ij}} = 1 + \varepsilon$ for all v_i in the clique and all $j = 1, \dots, n$, and $\alpha_{a_{kj}} = \delta_{p_{kj}} = 1 - \varepsilon$ for all v_k not in the clique and all

$j = 1, \dots, n$. Now count the ones in M among the selected elements. Each of the $n/2$ selected vertices contributes n^2 structural edges to G' , so there are $\frac{1}{2}n^3$ ones in M from the structural edges. The clique has $\frac{1}{2}\frac{n}{2}(\frac{n}{2} - 1)$ edges, and each edge corresponds to two arcs (graph edges), so there are another $\frac{n}{2}(\frac{n}{2} - 1)$ ones among the selected elements of M . Altogether, this gives K ones in the selected portion of M , so the value of the objective function must be at least $K(1 + \varepsilon)^2$. (The other terms in the objective, corresponding to ones in the other three portions of M , have value $(1 - \varepsilon)^2$ or $(1 - \varepsilon)(1 + \varepsilon)$, contributing a nonnegative amount.) \square

Lemma 7.5.2. *There is an extreme point optimal solution to our bilinear problem in which every α and δ is $1 + \varepsilon$ or $1 - \varepsilon$, and both α s and δ s split evenly between the two extremes.*

Proof. It is already known [151] that the bilinear program has a solution at an extreme point of its domain. In our case, each extreme point is the Cartesian product of an extreme point of $\mathcal{P}(\hat{\alpha}, \varepsilon)$ and one of $\mathcal{P}(\hat{\delta}, \varepsilon)$. An extreme point of $\mathcal{P}(\hat{\alpha}, \varepsilon)$ has every variable, except at most one, at a bound value. Because n is even, the number of nodes in G' with nonzero bounds in $\mathcal{P}(\hat{\alpha}, \varepsilon)$ is even; the same applies to $\mathcal{P}(\hat{\delta}, \varepsilon)$. The constant sum constraint then requires an even split between variables at $1 + \varepsilon$ and those at $1 - \varepsilon$. \square

Lemma 7.5.3. *If an extreme point optimal solution attains a value of at least $K(1 + \varepsilon)^2$, at least K matrix elements with value 1 (edges of G') have been selected.*

Proof. Assume an extreme point attains a value of at least $K(1 + \varepsilon)^2$. The entire matrix has at most $n^3 + n(n - 1)$ ones. Recall $K = \frac{1}{2}n^3 + \frac{n}{2}(\frac{n}{2} - 1)$. So, selecting $K - 1$ ones leaves at most $\frac{1}{2}n^3 + \frac{n}{2}(\frac{3n}{2} - 1) + 1$ unselected ones. For any $n \geq 2$, $\frac{1}{2}n^3 + \frac{n}{2}(\frac{3n}{2} - 1) + 1 < n^3$. Therefore, fewer than n^3 ones would be unselected, and each contributes a value of at most $(1 + \varepsilon)(1 - \varepsilon)$ to the objective. But, $n^3(1 - \varepsilon) = 1$ by definition of ε , so the total value of the unselected ones is at most $(1 + \varepsilon) < (1 + \varepsilon)^2$. Finally, the bilinear value is at most

$$(K - 1)(1 + \varepsilon)^2 + (1 + \varepsilon) < (K - 1)(1 + \varepsilon)^2 + (1 + \varepsilon)^2 = K(1 + \varepsilon)^2$$

if there are fewer than K selected ones in the extreme point. \square

Theorem 7.5.4. *The bilinear problem has a maximum value of at least $K(1 + \varepsilon)^2$ if, and only if, there is an $n/2$ clique in G .*

Proof. Lemma 7.5.1 establishes the “if” direction. Assume that G contains an $n/2$ clique. To prove the “only if” direction, we prove that if there is an optimal extreme point with K selected ones, there is an optimal extreme point with K selected ones and $n/2$ selected vertices. Then, the claim of the theorem follows.

Indeed, if there are K selected ones and $n/2$ selected vertices, there must be a clique: the $n/2$ selected vertices give us $n^3/2$ selected ones. That leaves no other selected rows or columns, and $\frac{n}{2}(\frac{n}{2} - 1)$ selected ones from the edges, which is all the possible arcs, hence a clique.

To prove that if there is an optimal extreme point with K selected ones, there is an optimal extreme point with K selected ones and $n/2$ selected vertices, we use induction on the size of the counterexample. In

other words, we consider a minimal counterexample and then show that it can be made smaller still, thus proving that a counterexample cannot exist.

We define the minimal counterexample as one with the fewest partially-selected vertices. If there are multiple such counterexamples, we take one that has a partially-selected vertex with the smallest number of selected rows plus selected columns. (Every counterexample must have at least one partially selected vertex.)

Let v_i be a partially-selected vertex with the fewest selected rows and columns.

Case 1. Suppose vertex v_i has no rows selected. Then we can find a partially-selected vertex v_j with at least one selected row and fewer than n selected columns. To see that such a v_j must exist, consider two sets: V_1 , containing all v with all columns selected, and V_2 , containing all v with at least one selected row. Since in an extreme point solution there are exactly $n^2/2$ selected rows and $n^2/2$ selected columns, we have $|V_2| \geq n/2$. Furthermore, v_i has at least one column selected so $|V_1| < n/2$. Therefore $V_2 \cap (V \setminus V_1) \neq \emptyset$ and we choose a v_j from this set.

We now unselect a column in vertex v_i and select a column in vertex v_j . This unselects at most one 1 (corresponding to the edge for the column originally selected), and selects at least one 1 (in the submatrix for vertex v_j), so we still have at least K selected ones. In a single row or column swap, all the newly-selected ones now contribute the greatest value of $(1 + \varepsilon)^2$ to the objective (previously they contributed the middle value of $(1 - \varepsilon^2)$). All unselected ones previously contributed the greatest value and now contribute the middle value. Therefore, as long as the number of selected ones after the swap does not decrease, the objective function also does not decrease. Thus, we obtain a smaller counterexample.

Case 2. Symmetrically, if the minimal vertex v_i has no selected columns, swap a row associated with a partially-selected vertex with at least one column selected and room to select another row. Therefore, the minimal vertex has at least one row and one column selected.

Case 3. Suppose the minimal vertex v_i has $r_i \geq 1$ rows selected and $c_i \geq 1$ columns selected. Suppose there is a partially-selected vertex v_j with $r_j > r_i$ selected rows. If v_j has fewer than n selected columns, we can swap a column from v_i to v_j and get a smaller counterexample. If v_j has n selected columns, swapping a row from v_i to v_j again gives a smaller counterexample provided $c_i < n$. If $c_i = n$, then r_i is the smallest among all partially-selected vertices (that is, all other partially-selected vertices v_j have $r_j \geq r_i$). There must be at least $n/2 + 1$ partially-selected or selected vertices. These cannot all have n selected columns. Therefore, there exists a vertex v_j with $c_j < n$. We must also have $r_j > r_i$ because if $r_j = r_i$ and $c_j < n$, then v_j would be the minimal vertex (recall $c_i = n$). A similar argument holds if there is a v_j with more than c_j columns selected.

Therefore, in any counterexample all partially-selected vertices have the same number of rows and columns selected as our minimal vertex v_i , say $n > k \geq 1$ rows and $n > \ell \geq 1$ columns. We know $k < n$ because otherwise (if $k = n$), there are exactly $n/2$ vertices with all rows selected and exactly $n/2$ vertices with no row selected. Since no partially-selected vertex can have zero rows selected (from Case 1), we can select only those columns corresponding to the $n/2$ vertices that have a row selected. Because we must select $n^2/2$ columns, we must select all columns for all these vertices and in fact no vertices are partially selected. A similar argument shows that $\ell < n$.

Pick one partially-selected vertex v_j , unselect a row and a column of v_i , and select at the same time a row and a column of v_j . Consider the submatrices M_{v_i} and M_{v_j} of M , defined by the rows and columns of v_i and, respectively, v_j . By unselecting a row of v_i , we unselect ℓ ones in M_{v_i} , and unselect at most one edge. Then, unselecting the column unselects at most $k - 1$ ones in M_{v_i} , and unselects at most one edge. When we select a row of v_j , we select ℓ ones in M_{v_j} . When we select a column of v_j , we select $k + 1$ ones (including the 1 for the new row and column) in M_{v_j} . So, we have unselected at most $k + \ell + 1$ ones, and selected at least $k + \ell + 1$ ones, yielding a smaller counterexample.

We have shown that in all cases, it is possible to reduce our counterexample to a smaller one with at least as many ones. This implies that there is no counter-example, and thus proves the theorem. \square

We have thus established that our inner bilinear optimization problem is NP-hard despite the simple domain. Our proof used ϵ arbitrarily close to 1. In practice, we generally have $\epsilon \leq \kappa$ for some $\kappa < 1$, such as $\kappa = \frac{1}{4}$. In such a case our proof does not apply, and the NP-hardness remains an open question. Further, this is a theoretical result, and it remains to examine some real applications to determine how difficult it is to solve this problem in practice (see §7.6).

7.5.2 Alternating Ascent

Although we have shown that the inner bilinear program (7.5.7) is NP-hard, we may still need to solve this in a practical manner. We consider a heuristic search strategy for solving the inner bilinear program for a given set of x values, which could be used as an inner loop for a general-purpose search strategy (e.g., meta-heuristic methods).

Figure 7.3 describes a simple local search strategy for solving the bilinear program for given values of x . Given x , one way to seek a solution is to alternate between α and δ , solving an LP in each iteration. Each maximization is a linear program, the same as the inner maximization of the linearly weighted case. Since each instance has different weights, we may need to sort each time. After a finite number of iterations, we terminate with extreme point optimal solutions to each polytope.

Alternating Ascent Algorithm

0. Initialize. Choose $\alpha^0 \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \bar{\alpha})$, and set $k = 0$.

1. Solve for δ . Given α^k , compute

$$\delta^k \in \operatorname{argmax} \left\{ \sum_{(i,j) \in \mathcal{R}} \alpha_i^k \delta_j x_{ij} : \delta \in \mathcal{B}(\hat{\delta}, \underline{\delta}, \bar{\delta}) \right\}$$

2. Solve for α . Given δ^k , compute

$$\alpha^{k+1} \in \operatorname{argmax} \left\{ \sum_{(i,j) \in \mathcal{R}} \alpha_i \delta_j^k x_{ij} : \alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \bar{\alpha}) \right\}$$

3. Increment $k \leftarrow k + 1$ and repeat steps 1–3 until $\delta^k = \delta^{k-1}$ ($k > 0$, step 1) or $\alpha^{k+1} = \alpha^k$ (step 2).

Figure 7.3. An alternating ascent algorithm for the bilinear program (7.5.7), for a given value of x .

The bilinear program has an optimal solution among the extreme points of $\mathcal{B}(\hat{\alpha}, \underline{\alpha}, \bar{\alpha})$ and $\mathcal{B}(\hat{\delta}, \underline{\delta}, \bar{\delta})$. This follows from elementary theory of LP and noting a necessary condition for (α^*, δ^*) to be optimal [151]:

$$\alpha^* \in \operatorname{argmax} \left\{ \sum_{(i,j)} \alpha_i \delta_j^* x_{ij} : \alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \bar{\alpha}) \right\}$$

$$\delta^* \in \operatorname{argmax} \left\{ \sum_{(i,j)} \alpha_i^* \delta_j x_{ij} : \delta \in \mathcal{B}(\hat{\delta}, \underline{\delta}, \bar{\delta}) \right\}.$$

Unfortunately, this is not sufficient. If we terminate the algorithm with (α^*, δ^*) , all we can say is that this satisfies the above necessary condition. We cannot rule out the possibility that a simultaneous change in α and δ would increase the objective value. We call a point that satisfies these necessary conditions an *alternating ascent solution*.

An alternating ascent solution is a KKT point (i.e., satisfies the first-order, Karush-Kuhn-Tucker conditions for optimality). Further, if the algorithm goes one iteration, the alternating ascent solution cannot be a minimum. It can, however, be a saddle point.

We have found examples where the alternating ascent solution is not a global maximum. Although the objective can not be improved by changing α or δ , keeping the other fixed, we were able to increase the objective value with a simultaneous change in α and δ (just to neighboring extreme points of their respective polyhedra). We ran some experiments to see how often this occurs, and the particular runs indicate that alternating ascent gets trapped at a non-global-maximum point from relatively few starting points. In all cases, an we found an improvement by combining neighbors of α in $\mathcal{B}(\hat{\alpha}, \underline{\alpha}, \bar{\alpha})$ with neighbors of δ in $\mathcal{B}(\hat{\delta}, \underline{\delta}, \bar{\delta})$. While inconclusive, this indicates that it may be possible to identify conditions under which such a search procedure would reach the global maximum. The extreme points of the bilinear

program are easy to characterize, so it may be possible to exploit this information in some cases to guide the local search. For example, it might be possible to jump to a non-adjacent extreme point to avoid terminating at a solution that is not a global maximum.

7.5.3 Linear Programming Relaxation

In this section we describe a linear relaxation of the bilinear case, using McCormick's bounds, commonly used by the α BB method [23]. McCormick's bounds have been used to approximate a bilinear form, $u^\top v$, on a rectangle $[\underline{u}, \bar{u}] \times [\underline{v}, \bar{v}]$ with linear functions using the simple inequalities [23, 187]:

$$\begin{aligned} (u - \underline{u})^\top (v - \underline{v}) &\geq 0, & (u - \bar{u})^\top (v - \bar{v}) &\geq 0 \\ (u - \underline{u})^\top (v - \bar{v}) &\leq 0, & (u - \bar{u})^\top (v - \underline{v}) &\leq 0. \end{aligned}$$

These yield linear (affine) functions that bound $u^\top v$ from below and above. In our case, since we are maximizing, we use only the upper bounds in the following LP relaxation which we call the *McCormick Approximation*:

$$\begin{aligned} \max z : \quad & \alpha \in \mathcal{B}(\hat{\alpha}, \underline{\alpha}, \bar{\alpha}), \quad \delta \in \mathcal{B}(\hat{\delta}, \underline{\delta}, \bar{\delta}), \\ z - \sum_{(i,j) \in \mathcal{R}} (\alpha_i \underline{\delta}_j + \bar{\alpha}_i \delta_j) x_{ij} &\leq - \sum_{(i,j) \in \mathcal{R}} \bar{\alpha}_i \underline{\delta}_j x_{ij} \\ z - \sum_{(i,j) \in \mathcal{R}} (\alpha_i \bar{\delta}_j + \underline{\alpha}_i \delta_j) x_{ij} &\leq - \sum_{(i,j) \in \mathcal{R}} \underline{\alpha}_i \bar{\delta}_j x_{ij} \end{aligned}$$

More generally, the McCormick Approximation has a guaranteed error of $\frac{1}{4}(\bar{\alpha} - \underline{\alpha})^\top (\bar{\delta} - \underline{\delta})$ times the optimal, established by Androulakis, et al. [23] for just the rectangle, $[\underline{\alpha}, \bar{\alpha}] \times [\underline{\delta}, \bar{\delta}]$. It is easy to extend this to account for only the contaminated nodes, giving an approximation guarantee of $\frac{1}{4}(\bar{\alpha} - \underline{\alpha})^\top x (\bar{\delta} - \underline{\delta})$. For example, if the uncertainties are proportional to epsilon ($\bar{\alpha} = (1 + \varepsilon)\hat{\alpha}$, $\underline{\alpha} = (1 - \varepsilon)\hat{\alpha}$, $\bar{\delta} = (1 + \varepsilon)\hat{\delta}$ and $\underline{\delta} = (1 - \varepsilon)\hat{\delta}$) then it follows that we have a $1 + \varepsilon^2$ approximation.

7.6 Preliminary Computational Study

In this section we report preliminary computational results for sensor placement problems on two water networks. We explore the computational difficulty of solving the full sensor placement problem, which we can solve for the versions that involve MILPs. We also explore the difficulty of computing optimal and approximate solutions to the bilinear inner optimization problem for the case with uncertainty in both attack weights and population distribution.

We performed computations on two real water networks with 97 and 470 nodes respectively. Water networks for large cities have tens of thousands of nodes, so these test cases are still about 2 orders of

magnitude smaller than many important sensor placement problems. However, these experiments serve to illustrate performance difference even for problems of this modest scale.

When we described the sensor placement problems earlier, we assumed only one flow pattern. Most cities, however, have multiple normal demand patterns cycling throughout each day. For example, desert cities may recharge tanks at night and drain them during the day. Our water networks have four daily demand patterns, which are used in our experiments. This makes the computations more realistic and more difficult. Sensor placements must protect against attacks that can have fundamentally different effects based upon when they occur. The only changes required to the models is to add an extra dimension to the input data. Attack weights and population are now associated with each of the four patterns, and the expectation in the objective is now taken over these patterns as well.

7.6.1 Linearly Weighted Uncertainties

Our first set of computational results considers the linearly weighted robust formulation, which can be modeled as a MILP. Our “base case” is the *central value model*, where we assume attack weights and population values are exactly equal to their central values (no robustness modeling). Our experiments illustrate how much additional computational effort is required to obtain a robust solution. We proved that the incremental effort is zero for the unweighted robust model, and we include computational results for two linearly weighted cases:

1. attack weights are uncertain and populations are fixed at their central values.
2. attack weights are fixed at their (non-uniform) central values and populations are uncertain.

In all experiments, we consider uncertainties that are proportional to the central value.

Table 7.1 shows the total computation time for the central value model (base case) and for robustly solving the sensor placement problem for our two example networks. The time reported in the table is the CPU seconds required to solve each problem running on a dual 3.06 GHz Intel[®]Xeon[™] processor, Linux 2.6.10 system with 2 GB RAM. The number of nodes, reported just after the time, is the size of the search tree computed by CPLEX[®].

Table 7.1. Computational Results for Complete Sensor Placement Solutions[†]

Network				Central Value		Linearly Weighted							
nodes	arcs	N_s	ϵ	time	nodes	Population time	nodes	Attack weights time	nodes				
97	234	5	0.1	21.9	16	188.8	24	135.9	20				
			0.2							50.0	93	25.2	54
			0.3										
		10	0.1	2.9	0	15.0	0	17.7	11				
			0.2							27.7	14	20.1	3
			0.3										
		20	0.1	11.9	4	28.5	0	27.1	0				
			0.2							197.7	9	38.7	0
			0.3										
470	1198	5	0.1	165.5	0	145.1	0	504.2	0				
			0.2							463.5	0	546.7	0
			0.3										
		10	0.1	69.9	2	678.7	2	549.9	2				
			0.2							790.3	9	617.8	2
			0.3										
		20	0.1	69.8	21	75.4	13	1472.9	63				
			0.2							1335.6	27	1441.1	44
			0.3										

[†]These were solved with AMPL[®] and CPLEX.

These results suggest that computational difficulty will generally increase with the number of sensors, N_s , and the robustness tolerance, ϵ . The robust formulation is almost always more difficult to solve than the central value model. Furthermore, the results with the larger network show that this robust formulation requires over an order of magnitude more time to solve in many cases.

7.6.2 Bilinearly Weighted Uncertainty

Although we have shown that just the bilinear subproblem, itself, is NP-hard, this result provides limited information about the practical computational difficulty of this formulation. Unfortunately, no currently available system can solve the bilinearly weighted formulation with global optimality confirmed. Recall that this model has the form:

$$\min_{x \in X} \max_{\alpha, \delta} \alpha^T x \delta$$

The minimand is a maximum, and no system can solve this at present. Although methods like BARON [190] can solve the inner bilinear program with confirmed optimality, we must combine that with an outer search strategy on x . Designing and implementing an outer search coupled with BARON is

Table 7.2. Computational Results for Bilinear Model for a Fixed Sensor Placement[†]

Network	N_s	Central value	ϵ	Alternating Ascent	McCormick Approx	Global Value	Time	
1	5	4100	0.1	4949	4949	4949	49.91	
			0.2	5878	5878	5878	65.43	
			0.3	6887	6887	6887	68.51	
	10	3955	0.1	4777	4777	4777	37.87	
				0.2	5677	5677	5677	37.92
				0.3	6654	6654	6654	38.35
	20	3907	0.1	4721	4721	4721	27.94	
				0.2	5613	5613	5613	24.76
				0.3	6582	6581	6582	22.49
2	5	134.54	0.1	158.62	158.61	158.62	2848.73	
			0.2	184.70	184.67	184.70	2953.75	
			0.3	212.81	212.70	212.81	2765.20	
	10	98.12	0.1	115.70	115.70	115.70	2289.35	
				0.2	134.75	134.72	134.75	2285.12
				0.3	155.30	155.19	155.30	2275.44
	20	70.13	0.1	82.55	82.54	82.55	1908.85	
				0.2	95.98	95.97	95.98	1901.59
				0.3	110.46	110.40	110.46	1910.16

[†] Alternating Ascent and McCormick Approximation were computed within MATLAB[®] (m files available upon request). McCormick Approximation used Mosek [22] to solve the LP. Global maxima were computed by BARON [190, 218]. The time reported is the CPU seconds for BARON to compute the global maximum; the approximations took no more than a few seconds each.

beyond the scope of this paper. The full sensor placement model could be written as a semi-infinite program by replacing the maximand with z and adding the infinite number of constraints:

$$z \geq \alpha^T x \delta \text{ for all } \alpha, \delta.$$

However, no currently available system can solve this model either.

Consequently, our computational experiments have focused on the time required to solve the bilinear subproblem for a fixed sensor placement. Table 7.2 compares the value of the central value model with the values of solutions for the bilinear subproblem. This subproblem is solved with the alternating ascent heuristic, with the exact BARON solver, and it is approximated with the McCormick approximation method. In these experiments, we set the contamination array (x) for the bilinear experiments using the values from the central value MILP model.

In all of these experiments, the BARON solver is one or more orders of magnitude slower than the other

methods, which should be expected because it is confirming global optimality. Remarkably, the alternating ascent heuristic generates solutions with the same value in every case. This contrast provides strong evidence that it is worth exploring the application of these heuristics for assessing the robustness of solutions with bilinearly weighted uncertainties. The approximation method was less effective than the alternating ascent heuristic overall, though the value of the solution that it generates was often quite similar. Finally, it is noteworthy that the runtime for BARON to solve the bilinear subproblem was often at least as long as the cost of solving the linearly weighted models. This suggests that a solver that can exactly solve the entire bilinearly weighted model will be significantly more expensive to solve than the MILP models for the linearly weighted uncertainties.

7.7 Discussion

There are many possible objectives for sensor placement that reflect various costs and risks of an attack on a network [240]. Previous work has considered problem formulations that minimize the volume of water consumed before detection [125], minimize the time to detection [130], and minimize the population exposed to contaminants before detection [31, 141]. This paper presents a foundation upon which these objectives, taken separately or multiply, can be considered in a manner that addresses data uncertainties.

Robust optimization addresses a need to hedge against uncertainty, and these uncertainties are a fundamental property of sensor placement problems. Data like attack weights and population distribution are based on expert judgement and incomplete source data. Furthermore, these data are expected to vary during the years after sensors are deployed in an early warning system. Although a number of criteria for robust optimization have been studied, the interval data model, with a constant-sum constraint, fits these sensor placement problems well.

The simplest case that we have considered is the unweighted uncertainty model, which represents two cases: (1) only the attack weights are uncertain, and the objective is the expected number of nodes that are contaminated without detection; and, (2) the attack weights are uniform, and the objective is the expected population that become contaminated without detection. In this case, if the interval is restricted to a fixed proportion of the central vector, we have shown this problem has a permanent solution. This is counter-intuitive, as data uncertainties *should* affect our decision. However, we have proven that robust solutions can be obtained from just the central value (which may be the most likely realization). Thus, obtaining a robust solution does not make the computation more difficult.

The next level of difficulty is the linearly weighted case: one set of parameters is fixed at their central values, while the other is uncertain. If we let the population be uncertain, the attack weights are presumed non-uniform; otherwise, the model reduces to the unweighted case. In our preliminary experiments, the linearly weighted case added a modest amount of computational effort to solve the overall problem.

Although the linearly weighted case can address robustness issues, the bilinear model is the most general by allowing uncertainty around non-uniform central values of both attack weights and population. We proved that the maximization subproblem is NP-hard, even with the simplifications of being in a ball with fixed-proportionate bounds and one constant-sum constraint. However, it is unclear whether this maximization subproblem remains NP-hard if we require that $\epsilon \leq \kappa$ for some $\kappa < 1$. Our NP-hardness

proof requires ε to be arbitrarily close to 1. Further, our computational experiments suggest that we can obtain solutions for the bilinear subproblem, or at least provide bound information in solving the master problem. However, we expect that it will be very difficult to compute exact solutions for bilinearly weighted formulations.

This paper has focused on modeling robustness and analyzing robust formulations. Our computational experiments are preliminary, and thus we cannot make strong predictions with them. A more comprehensive computational study of these techniques is clearly an important avenue of future research, including the development of computational techniques for exactly solving the bilinearly weighted formulation.

Chapter 8

Examining the Effects of Variability in Short Time Scale Demands on Solute Transport

Sean A. McKenna, Steven G. Buchberger (University of Cincinnati), Vince Tidwell

8.1 Background

The main focus of water supply system operation in the United States has traditionally been quantity. As concern over the health effects of disinfectant byproducts as well as the effects of inadvertent and malicious contamination events has grown, the operation of water supply systems has begun to place more attention on the fate and transport of dissolved phase solutes within these systems. One area where solute transport may prove to be of special concern is the dead-end mains within a water supply system.

Dead-end mains are pipes that have only one connection to the looped portion of the water distribution system. Under normal operating conditions, water moves through a dead-end stem in one direction only, from the entrance to the point of withdrawal, at rates dictated by local downstream use. Dead-end mains are surprisingly common. A survey of 14 randomly selected service zones in Cincinnati's water distribution system revealed an average of 54 dead-ends per zone accounting for about 23 percent of the total pipe length in the district. The average length of a dead-end stem in the Cincinnati survey was about 800 feet (Buchberger et al 2003). Dead-end mains are most prevalent in the peripheral suburban regions of the service zone; hence, they reach a disproportionately high percentage of the residential consumer base. Owing to chronic water quality problems with taste, odor, color and turbidity, many utilities periodically purge dead-end zones by flushing out fire hydrants [24]).

We focus on the effect that short-term variability in the demand has on solute transport through a dead-end main. This effect is examined through the use of a recently developed Poisson rectangular pulse (PRP)

simulator. Three performance measures that have a direct effect on solute transport are considered: 1) The variation in Reynolds number within the dead-end main; 2) the proportion of time for which stagnant conditions exist within the pipe; and 3) the travel time necessary to move a slug of water through a fixed distance of pipe. A single day simulation is run as well as an ensemble of 100 single day simulations. The three performance measures are calculated at each of 11 different time step sizes ranging from 1 second to 1 day. Results of the simulations are compared to previously developed analytical expressions (Buchberger et al., 1998; Buchberger and Lee, 1999) and a new approach to predicting the reduction in the variability of the performance measures is developed and compared to the simulated results.

This work addresses the issues of what level of time discretization is necessary for accurately simulating the flow of water and the transport of dissolved phase constituents in water supply network simulations and how the flow regime within the network changes under different levels of time discretization.

8.2 PRP Generator

Residential water demands at single family homes are assumed to behave as a nonstationary Poisson Rectangular Pulse (PRP) process [203]. Under the PRP hypothesis, the frequency of residential water use follows a Poisson arrival process with a time dependent rate parameter. When a water use occurs, it is represented as a single rectangular pulse of random duration and random but steady intensity. When a home draws a pulse of water from the supply network, it is considered to be "busy"; otherwise, the home is considered "idle".

Buchberger et al. [202] found that over 80 percent of indoor residential water demands occur as single pulses and that complex demand patterns are easily converted to an equivalent single pulse. By virtue of the Poisson assumption, it is unlikely that more than one pulse will start at the same instant. Owing to the finite duration of each water pulse, however, it is possible that two or more pulses with different starting times will overlap for a limited period. When this occurs, the total water use at the residence is the sum of the joint intensities from the coincident pulses.

In service zones fed by branching pipes (i.e., dead-end stems and other water mains where the flow direction is one-way only), the PRP premise leads to explicit expressions for the theoretical moments of busy fixtures, busy homes, flow rates, travel times and disinfectant concentrations [63] [61], [200]). Owing to its simplicity and versatility, the PRP approach offers a promising new way to model the temporal and spatial variability of the flow regime and the water quality in dead-end zones of the distribution system.

A computer program was developed to simulate instantaneous PRP residential water demands at nodes in a municipal distribution system. The simulation process involves five steps outlined below. Demand nodes may be configured to represent from 1 to over 1,000 homes. Each node requires a PRP pulse template to specify statistical properties of indoor and outdoor water demands and a multiplier pattern to define the hourly variation in arrivals for indoor and outdoor water use. Some typical parameter values for residential water demand simulations are listed in Table 8.1¹. Indoor and outdoor water demands are generated separately and then combined to give total demand. In the simulations discussed in this chapter, only

¹Values are taken from Buchberger et al. [202] [201].

Demand Characteristic	Indoor Use	Outdoor Use
Intensity : Mean	2.25 gpm	4.00 gpm
Intensity: Variance	1.55 gpm^2	1.00 gpm^2
Intensity: Coef of var	0.55	0.25
Intensity: Distribution	log-normal	log-normal
Duration: mean	45 sec	15 min
Duration: Variance	8100 sec^2	2700 sec^2
Duration: coef of var	2.00	3.00
Duration: distribution	log-normal	log-normal

Table 8.1. Parameter values for residential water demand simulations

indoor demands are considered. Two types of output are produced for each node: (i) water demand time series and (ii) water demand simulation statistics.

1. Compute time to next water demand (T) from an exponential distribution
2. For each water demand, generate a pulse duration (D) from a log-normal distribution
3. For each water demand, generate a pulse intensity (Q) from a log-normal distribution
4. For each water demand, generate a pulse volume (V) as product of duration and intensity, $V=QD$
5. Check for end of simulation. If no, then return to Step 1.

8.3 Analytical Expressions for Scaling Demand Variability

Two different approaches are examined for predicting the first and second moments of the different performance measures. The first is the approach developed by Buchberger et al. [61] [200]. The second approach is developed herein and is based on perturbation theory [?] [9]. The performance measures examined herein are the Reynolds number, Re, the travel time and the proportion of stagnant flow regime times to total simulation time. For flow through a pipe, the Reynolds number is:

$$Re = \frac{4Q}{\pi Dv} \quad (8.3.1)$$

(1) Where Q is the volumetric flow rate (ft^3/s), D is the pipe diameter (ft) and v is the kinematic viscosity - taken as $1.41E-05 ft^2/s$ for water at 50 deg F.

The mean travel time, T, through the pipe is calculated as the volume of the pipe upstream of the demand node divided by the volumetric flow rate: $T = V/Q$.

8.4 PRP Model Based Approach

Work by Buchberger et al [63] has used the PRP model of demands to develop expressions for the mean and variability of travel time within a pipe as well as for the proportion of the time during which there is no flow in the pipe. The mean and the variance of travel time, $E[T]$ and $Var[T]$ respectively, are given as:

$$\begin{aligned} E[T] &= \frac{V}{N\lambda\phi}, \\ Var[T] &= \left[\frac{V}{N^2\lambda^2\phi}\right][1 + \theta^2[p]] \end{aligned} \quad (8.4.2)$$

(2) where V is the volume of pipe to be drained within the travel time, N is the number of homes served by the single node, λ is the average arrival rate, ϕ is the mean volume of a demand pulse and θ is the coefficient of variation of the volume of the demand pulses. It is noted that the expressions for the mean and variance of the travel time are independent of the size of the time step over which they are calculated.

The proportion of the time for which there is no flow in the pipe, $P[0]$ or the stagnant flow regime, is given by:

$$P[0] = \exp[-N\lambda(\tau + \delta t)] \quad (8.4.3)$$

(3) where τ is the mean demand pulse duration and δt is the size of the time step.

8.5 Perturbation Theory Approach

Perturbation theory is used to develop expressions that estimate the changes in the first two moments of the performance measure distributions. Presently, this development is done by focusing solely on the head fluctuations in the water main. The resulting expressions are valid for any variable that is a linear function of head, such as Re and the same approach should prove to be applicable to calculating scaling expressions for solute dispersion.

The governing equation for head fluctuations in a pipe network is:

$$\frac{dh}{dt} = \frac{Q_s}{A} - \frac{Q_d}{A} \quad (8.5.4)$$

(4) Where $Q_s(L^3/T)$ is the volumetric supply, $Q_d(L^3/T)$ is the volumetric demand, $h(L)$ is the head, A is the cross-sectional area of the pipe (L^2) and $t(T)$ is time. To make the solution more tractable, head loss will be ignored for the current analysis. Equation 8.5.4 might represent the head fluctuation at a single node or represent the spatially averaged response for a network of pipes and nodes.

To maintain consistency with the calculations in this chapter we assume a single pipe of constant diameter supplying water to a single node with multiple users. Also, Q_s is treated as constant (i.e., considered a constant head reservoir); however, this could easily be changed to temporally varying. The mean nodal demand is modeled by the PRP function with the form

$$Q_d = \lambda \tau \alpha \quad (8.5.5)$$

(5) where $(1/T)$ is the arrival rate, (T) is the pulse duration, and $(L3/T)$ the pulse intensity of a busy server. Recall that these three variables are independent and each is randomly distributed in time. Substituting 8.5.5 into 8.5.4 yields

$$A \frac{dh}{dt} = Q_s - \lambda \tau \alpha \quad (8.5.6)$$

(6) where A and Q_s are considered constant while $\lambda \tau \alpha$ are random variables, thus making h a random variable as well. We treat 8.5.6 as an ordinary stochastic differential equation and seek to solve this equation using the small perturbation approach where each random variable is comprised of its expected value (denoted by the overbar) plus a zero-mean perturbation (designated by $\hat{\cdot}$),

$$h = \bar{h} + \hat{h}, \quad \lambda = \bar{\lambda} + \hat{\lambda}, \quad \tau = \bar{\tau} + \hat{\tau}, \quad \alpha = \bar{\alpha} + \hat{\alpha} \quad (8.5.7)$$

(7) Substituting 8.5.7 into 8.5.6 and taking the expected value yields an expression for the mean head

$$A \frac{d\bar{h}}{dt} = Q_s - \bar{\lambda} \bar{\tau} \bar{\alpha} \quad (8.5.8)$$

(8) Applying the initial condition $h(t=0) = H_0$ and solving Equation 8.5.8 yields

$$\bar{h} = \left[\frac{Q_s - \bar{\lambda} \bar{\tau} \bar{\alpha}}{A} \right] t + H_0 \quad (8.5.9)$$

(9)

Recalling that supply must match demand, Equation 8.5.9 reduces to . From this simple relation it is apparent that mean head is constant over time, and thus lacks a time constant. However, the head fluctuation still exhibits random variability with time. To explore this we construct the perturbation equation by expanding Equation 8.5.6 using the relations in 8.5.7 and then subtracting Equation 8.5.8. Because each random variable is independent, all perturbation terms of second order and higher are zero. This yields

$$A \frac{dh}{dt} = \lambda \bar{\tau} \bar{\alpha} + \bar{\lambda} \tau \bar{\alpha} + \bar{\lambda} \bar{\tau} \alpha \quad (8.5.10)$$

(10)

Using the spectral representation theorem [6] we have

$$S_h h(\omega) = \left[\frac{\bar{\tau} \bar{\alpha}}{A \omega} \right]^2 S_{\lambda \lambda}(\omega) + \left[\frac{\bar{\lambda} \bar{\alpha}}{A \omega} \right]^2 S_{\tau \tau}(\omega) + \left[\frac{\bar{\tau} \bar{\lambda}}{A \omega} \right]^2 S_{\alpha \alpha}(\omega) \quad (8.5.11)$$

(11)

where are the spectra of each random variable. Noting that the spectrum of each random variable is characterized by white-noise, the head variance σ , is given by:

$$\sigma_h^2 = \left[\frac{\bar{\tau}\bar{\alpha}}{A}\right]^2 \varepsilon_\lambda \sigma_\lambda^2 + \left[\frac{\bar{\lambda}\bar{\alpha}}{A}\right]^2 \varepsilon_\tau \sigma_\tau^2 + \left[\frac{\bar{\tau}\bar{\lambda}}{A}\right]^2 \varepsilon_\alpha \sigma_\alpha^2 \quad (8.5.12)$$

(12)

where σ_i^2 is the variance of each of the three random variables and $\varepsilon(T)$ is the corresponding correlation time scale. Finally, the head variance in time will decrease according to the simple volume-variance relation [?].

$$\sigma_h^2\left(\frac{t_0}{t} = \bar{C}_h(t_0, t_0) - \bar{C}_h(t, t) \quad (8.5.13)$$

(13)

that is the variance will decrease as the difference of the mean head covariance, \bar{C}_h , as calculated for time steps of different sizes, t_0 and t .

8.6 Modeling Temporally Variable Demand

The dead-end simulations done here are for a single node containing 20 homes on the end of a 2000-foot long, 6-inch diameter pipe. A single day simulation is done to examine the temporal distribution of the flow regime in the pipe. Simulations for multiple days are performed to determine the variability in the proportion of the time for which there is no flow, the stagnant flow regime, and the variability in the time for a volume of water to travel 1000 feet along the pipe, the travel time.

Statistics for the different flow measures are calculated using 11 different discretizations of the time step from one second to one day in length. For these simulations, only indoor use is considered and the parameters controlling the arrival rate, the intensity and the duration are set to be constant in time. The utilization factor in the Poisson arrival rate distribution is 0.048/min, and the intensity and duration of the demands are both log-normally distributed with mean and variances of 2.50 gpm and 1.56 gpm² and 1.0 minutes and 4.0 minutes², respectively.

8.6.1 Single-Day

Simulation As a first step in understanding the effects of short time scale variability in demand on the solute transport of a dissolved contaminant, a single day is simulated using the PRP generator. This single day is simulated using 11 different levels of time discretization, time steps, and the results from the different time steps are compared.

Time Step (sec)	Minimum Re	Maximum Re	Mean Re	Std Dev Re	Stagnant Proportion
1	212.9	9138.6	1562.9	1049.1	0.38
30	13.9	7698.2	1261.4	1015.2	0.23
60	10.0	7057.7	1128.1	959.9	0.14
300	18.7	4619.5	977.9	726.8	0.00
600	36.3	3554.7	974.5	620.8	0.00
1800	324.0	1832.0	974.5	383.5	0.00
3600	469.8	1685.8	974.5	284.1	0.00
10800	773.8	1402.7	974.5	187.5	0.00
21600	904.8	1088.2	974.5	79.1	0.00
43200	952.5	996.5	974.5	31.1	0.00
86400	974.5	974.5	974.5	0.0	0.00

Table 8.2. Reynolds number and stagnant proportion results of the single day simulation

The performance measures examined in the single day simulation are the temporal distribution of Reynolds number summarized by the min, max, mean and standard deviation, and the proportion of time for which there is no flow in the system. Results for the different performance measures for a total simulation time of one day are given in Table 8.2.

The mean value of Re indicates that flow in the dead-end main is laminar for all time step sizes. However, the maximum Re values for time steps less than or equal to 10 minutes are turbulent. These turbulent periods are due to multiple servers at the node being active at a given time, but these periods make up a relatively small proportion of the total time period.

The stagnant periods are not included in the calculation of the Re statistics. This approach leads to an initial decrease in the minimum Re as the time step increases and as fewer of the time periods are stagnant. This decrease is due to the smoothing nature of the increased time steps. For example, with a 1 second time step, flow either occurs in the pipe with a Re of at least 212.9 or it does not occur at all. As the time step increases, this same amount of flow is distributed over a larger time step and the Re decreases as do the proportion of stagnant time periods. As the time step increases to the point where there are no stagnant periods, the minimum Re begins to increase until it equals the maximum Re at a time step of 86400 seconds.

The results in Table 8.2 show that for one-second time steps, the flow regime in the dead-end main is stagnant 38 percent of the time. For time steps of 300 seconds or greater, there are no stagnant periods.

8.7 Multiple Day

Simulation The single day simulation provides detailed results on the full distribution of Re as a function of the time step, yet it only provides a single value of the proportion of stagnant times and travel time for each time step. To begin to understand the variability in these quantities, the PRP simulator is used to generate 100 different days of demand. The PRP parameters used for the single day simulation are held constant for

Time Step	Travel Time Max	Mean	Min	Std Dev	Stagnation Prop Max	Mean	Min	Std Dev
1	45727.8	37362.3	27623.2	3542.5	0.446	0.385	0.334	0.023
30	45727.1	37362.8	27622.4	3543.2	0.390	0.290	0.242	0.017
60	45727.1	37362.3	27622.3	3542.9	0.291	0.205	0.150	0.014
300	45687.2	37360.2	27617.1	3543.5	0.188	0.117	0.003	0.004
600	45735.0	37367.9	27617.6	3546.4	0.017	0.000	0.000	0.001
1800	45756.0	37363.4	27528.6	3553.9	0.007	0.000	0.000	0.000
3600	45610.8	37348.3	27810.9	3500.7	0.000	0.000	0.000	0.000
10800	46056.1	37284.3	27982.4	3437.6	0.000	0.000	0.000	0.000
21600	45867.2	37350.8	27512.0	3351.9	0.000	0.000	0.000	0.000
43200	46076.2	37426.9	29146.7	3178.1	0.000	0.000	0.000	0.000
86400	43318.5	37186.2	32161.7	2350.4	0.000	0.000	0.000	0.000

Table 8.3. Travel time and stagnant proportion results of the multi-day simulations

each simulation, but the total demand from one day to the next is not constrained to be a constant. The proportion of stagnant time and the travel time is calculated for each day of demand and the results are given in Table 8.3.

Results of the multi-day simulations show that the parameters defining the travel time distributions are nearly identical from one time step to the next until the time step size reaches 3600 seconds, 1 hour. At this time step size and beyond, the minimum and maximum times get closer to one another and the standard deviation decreases. The mean travel time is essentially constant across all time steps.

The variability in the proportion of stagnant time periods in the dead-end main from one day to the next also decreases as the time step increases. The standard deviation of this proportion as calculated across all 100 simulations drops nearly two orders of magnitude from a time step size of 1 second to a time step size of 600 seconds. After this time step size, there is no more variation in the proportion of stagnant times. The largest level of variability in the proportion of stagnant times occurs at the one-second time step where the proportion of stagnant periods can vary over ten percent from one day to the next.

8.8 Discussion

The results of the multi-day simulations show that the variability in travel time remains relatively constant as the size of the time step increases until the mean travel time approaches the size of the time step. This result indicates that it may be possible to accurately model travel times using time steps that are much larger than the time scale of the variability in demand. This result is due to the travel time being a function of the integrated amount of demand in the system from one end of the pipe to the other. If the time steps being considered are significantly smaller than the expected travel time, then the variation in demand seen at these time steps is smoothed out. As the time step increases to be near the travel time, the variability in the travel times will increase.

It is not yet clear if these results imply that it may be possible to accurately model solute transport in water

supply systems with relatively large time steps. Certainly, the changes in the flow regime within the dead-end pipe considered here suggest that using large time steps will not allow for modeling of periods of stagnation within the pipe. The single day simulations show that for the 20 home case considered here, simulation time steps greater than 5 or 10 minutes will not reproduce any of the stagnation periods seen at the one second time scale. Accurate modeling of the stagnation periods will be especially important for any solute that reacts with materials on the pipe walls.

The simulation results shown above can be compared to the predictions made with the analytical expressions for the first two moments of the different performance measures. Equation 2 is used to predict the mean and standard deviation of the travel times at: 36,750 and 3801 seconds respectively. These predictions are not a function of the time step size. Comparison of these predictions with the results in Table 8.3 shows that the analytical expressions over predict the standard deviation for the shorter time steps by approximately 7 percent. The mean travel time in the numerical simulations, roughly 37350 seconds across all time step sizes, is also over predicted by equation 2 by approximately 1.5 percent.

The proportion of stagnant flow regime periods is calculated for all time steps using Equation 3. These calculations are compared to the results from the single day simulation and the multi-day simulations in Figure 8.1. This figure shows that the analytical results tend to under predict the mean values of the multi-day simulations, but that they are very close to the values calculated in the single-day simulation. These results indicate that the single-day simulation was one that predicted fewer stagnant periods than the majority of the multi-day simulations and that, in general, the analytical expression slightly under predicts the simulated proportions of stagnant flow, although the analytical predictions are within the range of the simulated results.

At this point, there are no analytical expressions formulated explicitly for predicting the temporal moments of the Re distribution. However, the perturbation theory approach developed above can be applied to predict the decrease in the variability of Re as the time step size increases (Equation 13). That is the Reynolds number should fluctuate linearly with the head. This prediction requires an estimate of the head covariance and associated time scale. Assuming an exponential covariance function with time scale of 300 second, the variance reduction was calculated and is shown in Figure 8.2. This time scale is reasonable as it is roughly the mean plus two standard deviations of the demand pulse duration. These results show that the perturbation approach fits the numerically simulated results reasonably well.

8.9 Conclusions

This work has demonstrated the application of a PRP simulator for simulating variable demands in a water supply network across a range of time scales. Results of these simulations show that travel time variation does not decrease with increasing time step size until the time step size approaches that of the mean travel time. Contrary to the travel time results, the proportion of time with a stagnant flow regime time does decrease rapidly with increasing time step size. Previously developed analytical expressions for mean and variance of travel time and decrease in proportion of stagnant times provide reasonably accurate prediction of the simulation results with the exception of not being able to predict the decrease in travel time variability as the time step size approaches that of the mean travel time. A new perturbation approach to predict the decrease in head variance, is applied to Re results and gives accurate predictions of the variance

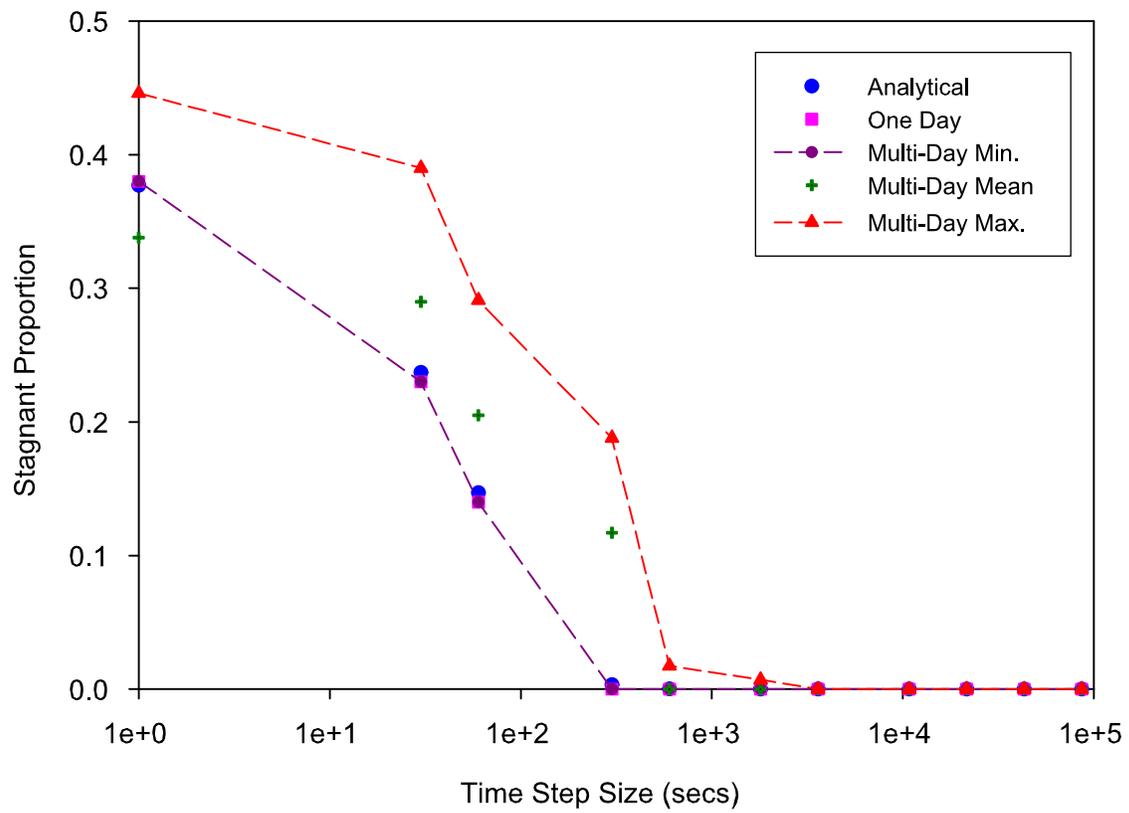


Figure 8.1. Comparison of stagnant proportion results with analytical predictions made from Equation 3.

reduction. Application of this technique requires definition of the temporal correlation of the demand.

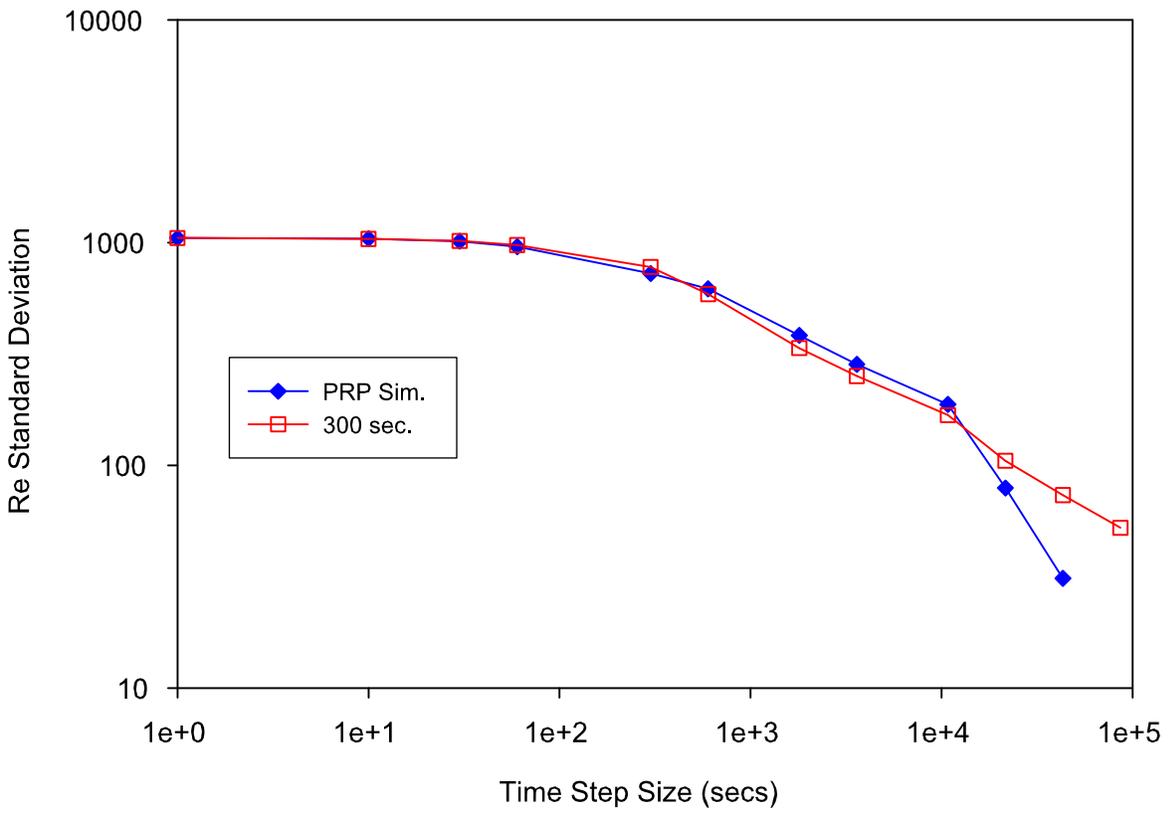


Figure 8.2. Comparison of perturbation theory results with base time scales of 1 second and 300 seconds to the simulation results.

Chapter 9

Modeling Solute Transport in Distribution Networks with Variable Demand and Time Step Sizes

Sean A. McKenna, Chad Peyton, Lane Yarrington, Steven G. Buchberger (University of Cincinnati), Roger Bilisoly

9.1 Background

Security of water supply systems within the U.S. depends on being able to understand the fate and transport behavior of any contaminant that might be inadvertently, or malevolently, introduced into a water supply distribution network. The transport behavior of a solute in a distribution network is closely tied to the hydraulic behavior of the network [225]. The previous chapter and recent work (e.g., [62], [153]) have shown that the time scale at which hydraulic demands are discretized can influence solute transport behavior in simple dead-end mains. This previous work has focused on the different flow regimes: stagnant, laminar or turbulent, present in a dead-end main as a function of the time scale of the observations and/or simulations of the hydraulic demands. The current work extends this research from the dead-end main to the case of a more complex water distribution network.

Here we focus on solute transport simulation using EPANET in a water distribution system of moderate size and complexity. The hydraulic demands on the system are generated using a Poisson Rectangular Pulse (PRP) demand generator with parameters based on observed residential demands. The hydraulic time step size is varied from one minute to two hours. Solute is introduced at both a tank in the system and at three different nodes selected to be representative of three different local flow systems that occur in distribution networks. The timing and magnitude of the two different contaminant source functions, tank or node, are defined to represent possible malevolent attacks on a water distribution system, although the solute transport results have general applicability for water quality problems as well. Results of this work

show the hydraulic time step discretization necessary for consistent modeling of solute transport within EPANET and show the relative differences in contaminant transport for tank versus node sources across a range of hydraulic time step sizes.

9.2 Water Demand Generator

Residential water demands at single family homes are assumed to behave as a nonstationary Poisson Rectangular Pulse (PRP) process [65]. Under the PRP hypothesis, the frequency of residential water use follows a Poisson arrival process with a time dependent rate parameter. This process produces an exponential distribution of arrival times. When a water use occurs, it is represented as a single rectangular pulse of random duration and random but steady intensity. When a home draws a pulse of water from the supply network, it is considered to be "busy"; otherwise, the home is considered "idle".

Buchberger et. al [64] found that over 80 percent of indoor residential water demands occur as single pulses and that complex demand patterns are easily converted to an equivalent single pulse. By virtue of the Poisson assumption, it is unlikely that more than one pulse will start at the same instant. Owing to the finite duration of each water pulse, however, it is possible that two or more pulses with different starting times will overlap for a limited period. When this occurs, the total water use at the residence is the sum of the joint intensities from the coincident pulses. Owing to its simplicity and versatility, the PRP approach offers a promising new way to model the temporal and spatial variability of the flow regime and the water quality in dead-end zones of the distribution system.

A computer program PRPsym was developed to simulate instantaneous PRP residential water demands at nodes in a municipal distribution system. The simulation process involves five steps outlined in 9.1 Figure 9.1. Demand nodes may be configured to represent from 1 to over 1,000 homes. Each node requires a PRP pulse template to specify statistical properties of indoor and outdoor water demands and a multiplier pattern to define the hourly variation in arrivals for indoor and outdoor water use. Parameter values for residential water demand simulations done here are listed in Table 1 of the previous chapter. Indoor and outdoor water demands are generated separately and then combined to give total demand. In the simulations discussed in this chapter, only indoor demands are considered. Two types of output are produced for each node: (i) water demand time series and (ii) water demand simulation statistics. The PRP demand generator is used to create demands at 60, 300, 600, 1800, 3600 and 7200-second time steps.

9.3 Example Network

The example network used for this project contains 470 nodes and 621 pipes. The network covers an area that is roughly 6x8 miles. The calculations done herein focus on the northern portion of the network, shown in 9.2 Figure 9.2, which covers an area that is roughly 3 x 2.5 miles. The color scales in Figure 9.2 are log₁₀ based and show both the demands at the nodes and the flow through the pipes in units of gpm. The contaminant source locations are labeled in red and the monitoring nodes are labeled in black (Figure 9.2). The source tank T-7812 is approximately 1 mile further east of the edge of Figure 9.2.

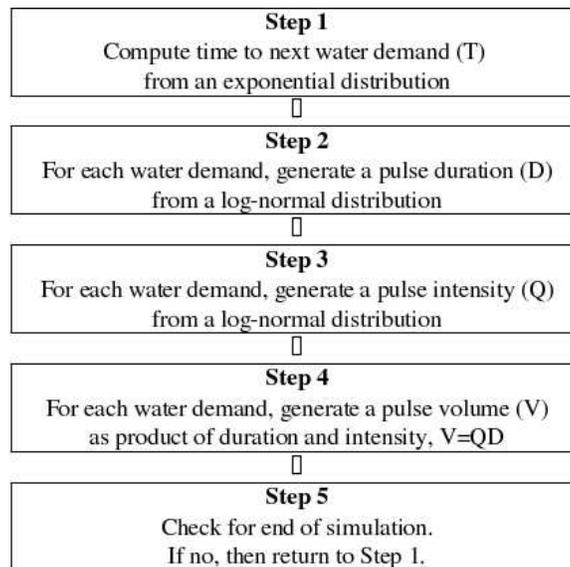


Figure 9.1. Simplified algorithm for simulating PRP residential water demands. All random variables are generated independently

Pressures and flows within this network have been calibrated with multiple pressure release events. The base demands at each active node have been recorded over a period of several months. For this work, base demand at each node is assumed to be solely due to residential demand and it is assumed that per capita water consumption is 160 gpd and an average of four people reside in each home. These assumptions allow for the calculation of the number of homes at each node, which is a necessary input to the PRP demand simulator. Figure 9.2. Water distribution network used in this study. The color scales show demands (nodes) and flows (pipes). The source and monitoring locations used in the simulations are labeled.

A large number of the nodes in this system have average base demands of zero gpm. These nodes are generally fire hydrants. Other zero demand nodes are those in the northeast section of the network where residential neighborhoods are being demolished. Flow and Transport Simulations The hydraulic response of the system to the aggregation of demands at different time steps is summarized by examining the variation in Reynolds number across all pipes in the network. Two separate sets of solute transport simulations were conducted. The first set introduces 40 kg of contaminant that is perfectly mixed into the water in tank T-7812 on the extreme eastern edge of the distribution system. The second set of simulations introduces 0.45 kg of contaminants through injection at three different nodes within the distribution system (Figure 9.2). For all simulations the six different levels of demand discretization are used. The size of the hydraulic time step is varied to match that of the demand discretization, but the water quality time step is fixed at 10 seconds for all simulations. All simulations are run for 86,400 seconds (24 hours).

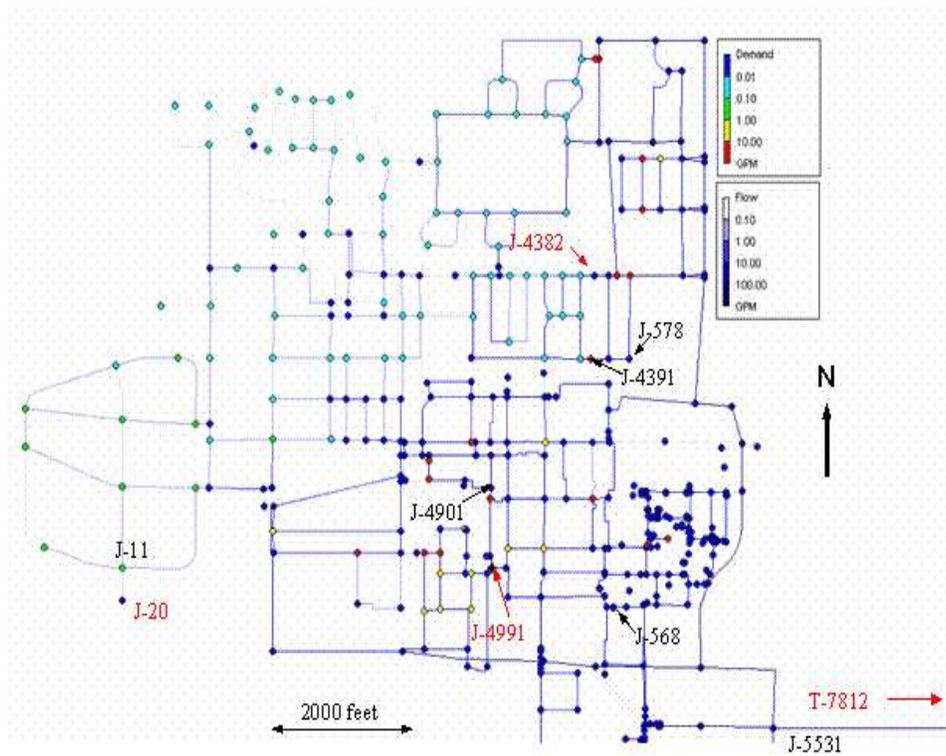


Figure 9.2. Water distribution network used in this study. The color scale show demands (nodes) and flows (pipes). The source and monitoring locations used in the simulation are labeled.

9.4 Reynolds Number Distribution

For flow through a pipe, the Reynolds number, Re , is given by: $(4Q)/(\nu d)$, where Q is the volumetric flow rate (ft³/s), d is the pipe diameter (ft) and ν is the kinematic viscosity $\hat{\nu}$ taken here as $1.41E-05$ ft²/s for water at 50° F. The values of Re are calculated for each of the 621 pipes in the network. A value of Re is computed for each time step in each pipe for the first hour of the total 24 hour simulation. The number of points in the cdf varies from 37,260 (60, 60 second time steps over 621 pipes) to 621 (1, 3600 or 7200 second time step over 621 pipes)

The cumulative distribution of Re values across all pipes as calculated under different hydraulic demand time step sizes is shown in Figure 9.3. A significant fraction of the pipes in the system, approximately 38 percent, have Re values of zero. These pipes are mainly those connecting nodes with zero demands, such as fire hydrants and locations in the northeast portion of the network where homes are being demolished. Results in Figure 9.3 show some variability in the distribution of Re between time step sizes, but there is no consistent change in the shape of the distribution across the different time step sizes. These results indicate that, for a one-hour sample period when viewed across all pipes, the strong dependence of the flow regime

on the size of the hydraulic time step seen in dead-end mains is not significant. Solute Center of Mass: Tank Source In order to define the location of the solute plume within the distribution network with a single parameter, the moment of inertia of consumed mass was calculated. For each time step the amount of mass removed from the system at the i th node is the product of the average demand, D , throughout that time step, the average solute concentration, C , over that time step and the length of the time step, Δt : $D_i \cdot C_i \cdot \Delta t$. Using mass removed from each node as a weight and knowing the coordinates of each node, it is possible to solve for the location of the moment of inertia, or centroid, of the mass removed from the distribution system during that time step.

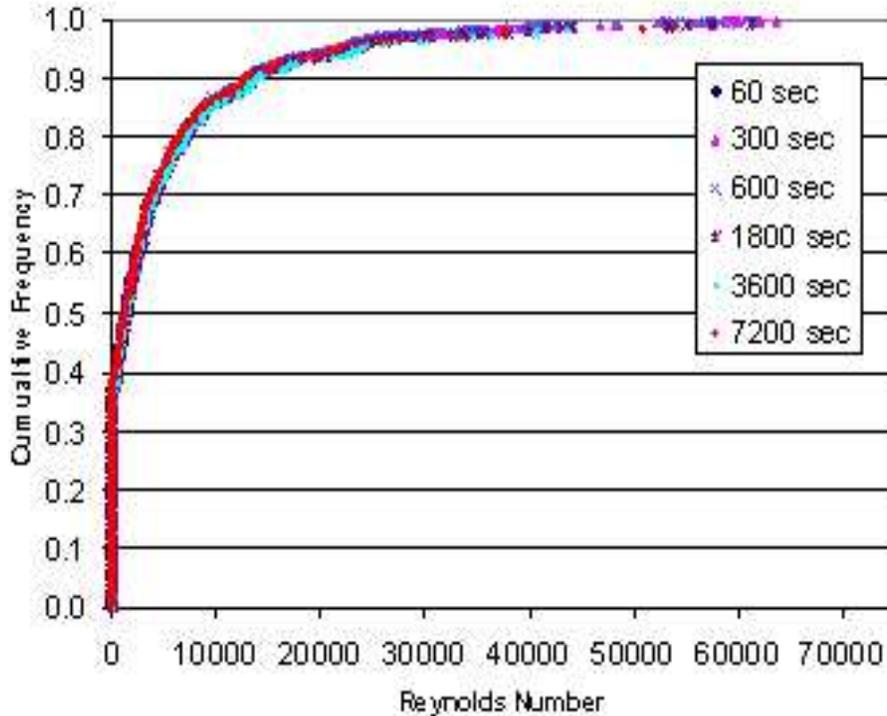


Figure 9.3. Distribution of Reynolds Numbers

The center of mass removed aggregated at 1-hour intervals is shown for simulations with different size time steps in Figure 9.4. The solute source for these simulations is tank T-7812 at the extreme right edge of the network. These results show that the center of solute mass within the system is not significantly affected by the hydraulic time step size. There are some deviations to the NE for the 3600 and 7200 sec time steps relative to the other time steps, but these only occur in one section of the results. These results also show that the center of mass of the plume moves a considerable distance, over 3000 feet from South to North during the one day simulation time. This range of movement indicates a large amount of contaminant spreading throughout the network.

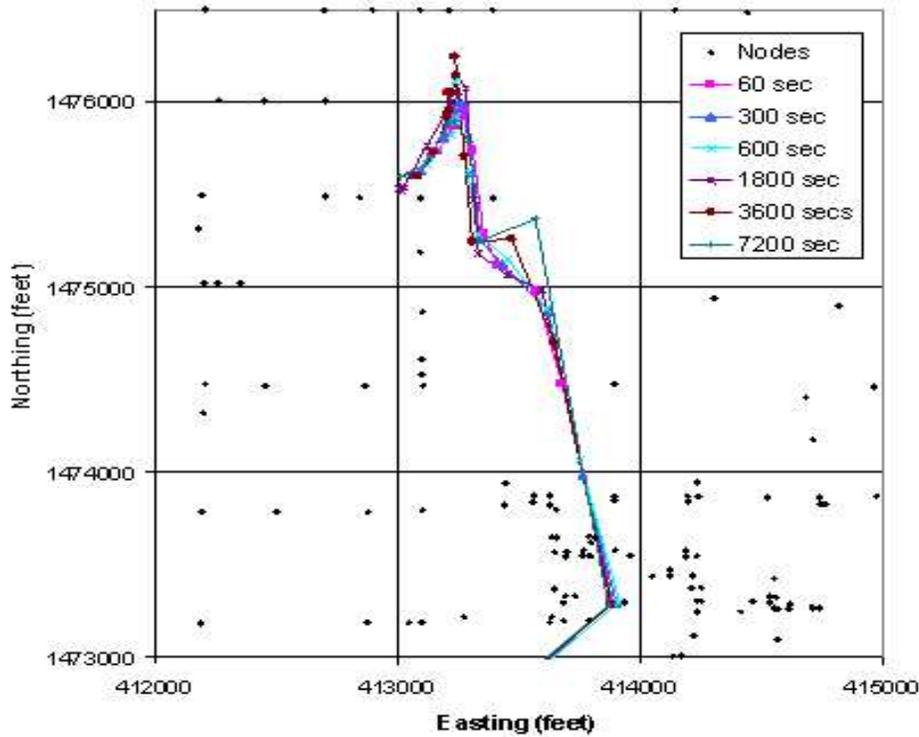


Figure 9.4. Centers of consumed mass from the tank source for six different time steps. The centers of mass are connected in chronological order.

9.5 Solute Transport Timing: Tank Source

The timing of solute transport from the tank source through the system is checked by comparing the breakthrough curves of the solute pulse at several locations in the system. The three locations chosen are nodes: J-568, J-578 and J-5531 (see Figure 9.2). The resulting solute breakthrough curve for each time step size is shown at each of these nodes in Figure 9.5. The results in Figure 9.5 show that as the time step is decreased to 1800 sec (30 minutes) the breakthrough curves converge to a consistent result. The simulations with the 3600 and 7200-second time steps do not provide enough resolution to accurately capture the timing of the solute transport. The BTCs show a sharp rise to a peak concentration equal to that of the injection concentration in the tank, 12.3 mg/l, followed by a long decay back to zero concentration. The sharp rise in concentration and the lack of variation in this rise for different time step sizes below 3600 seconds shows that the variable demands at different time scales cause minimal mixing between the contaminated and clean water.

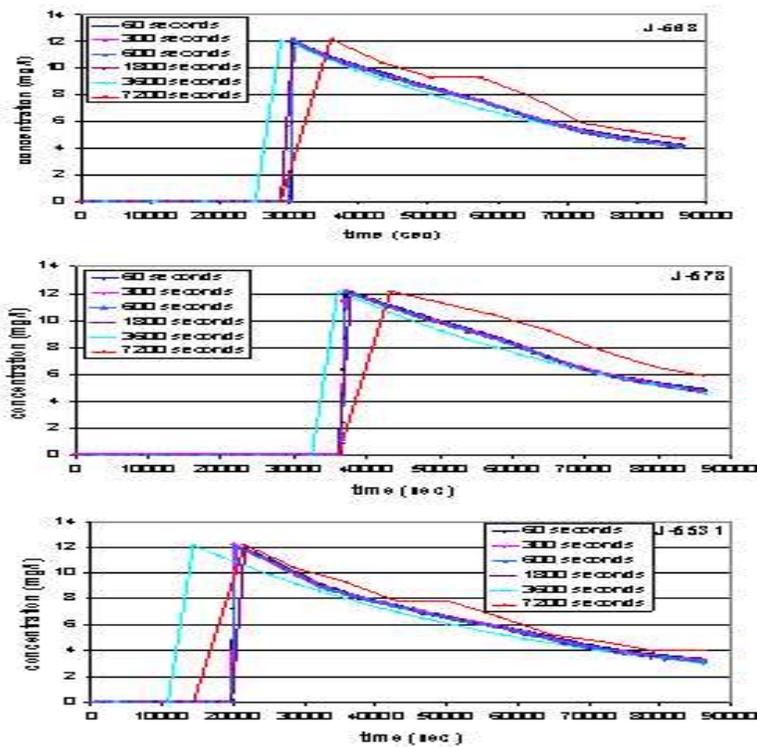


Figure 9.5. Solute breakthrough curves at three different monitoring nodes for the tank sources

9.6 Solute Center of Mass: Node Sources

Three different nodes in the system with zero demand were selected as injection points. These three nodes are: J-20, J-4991 and J-4382 (Figure 9.2). These nodes were selected to provide a broad range of local hydraulic demand conditions under which a contaminant could be injected. These range from a very low demand portion of the network (J-20) to an area with high demand but flow in principally one direction (J-4991) to an area with relatively high demand in multiple directions (J-4382). Each source is a square-wave pulse injection at a rate of 10 gpm and a concentration of 100 mg/l for 2 hours. The total amount of contaminant mass injected is 454 grams (0.45kg). Clean water was also injected for another 3 hours, also at 10 gpm, beyond the end of the contaminant injection to flush the contaminant from the dead-end node.

Results show that for the injection into J-20, the contaminant stays close to the injection point and after 24 hours, only 1.4 percent of the injected mass has been removed from the system. If the demands remained constant from day to day, it would take approximately 74 days for all of the mass to leave the system. For all time step sizes, all mass leaves the system at a single node, J-11, and therefore the center of consumed mass is at the J-11 node.

The center of consumed mass results for the injections at J-4382 and J-4991 are shown in Figure 9.6. For the source at J-4382, roughly 85 to 90 percent of the mass is removed from the system within one day and for the source at J-4991, all of the mass is removed from the system by the end of one day of simulation time. The center of consumed mass for the injection at J-4382, in an area of diverging flow directions, changes over time and covers an area approximately 1500x1500 feet (left image, Figure 9.6). Results show that there is variation in the center of mass across the different time step sizes. The center of consumed mass for the injection at J-4991 occurs at only two nodes that are located roughly 1000 and 1500 feet from the injection point (right image, Figure 9.6) and changing the time step size does not introduce additional variation.

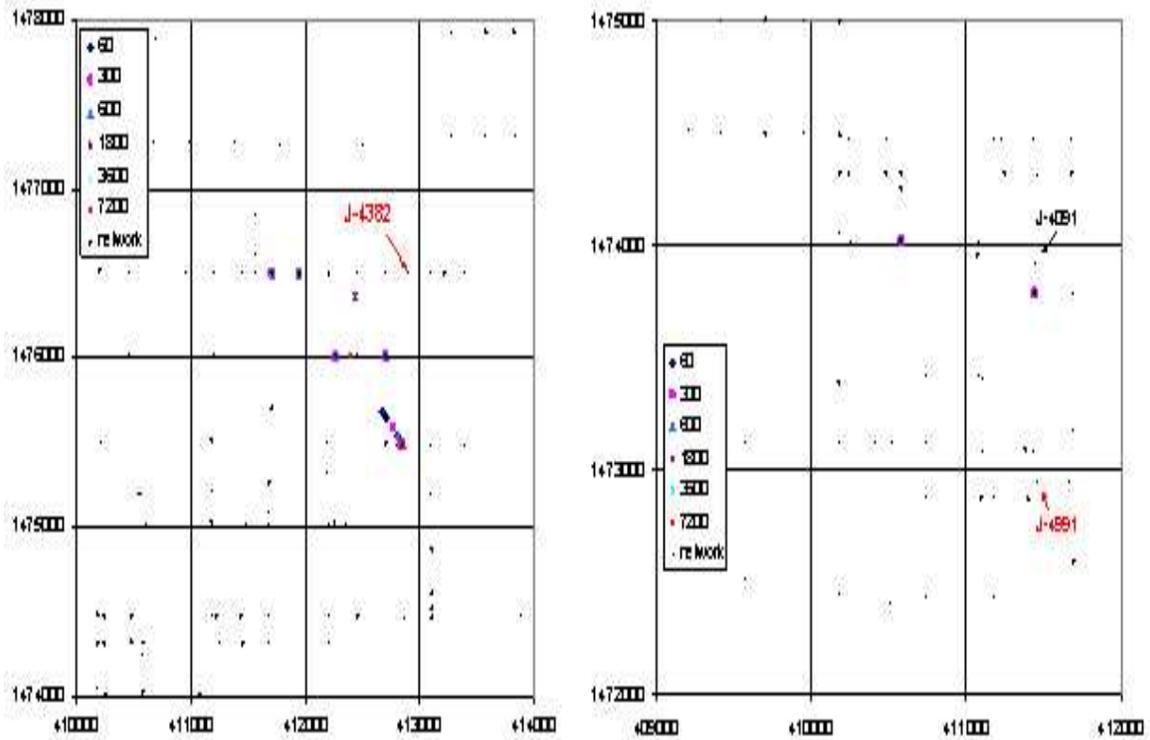


Figure 9.6. Center of consumed mass results for injections at nodes J-4382 (left image) and J-4991 (right image).

9.7 Solute Transport Timing: Node Sources

For each of the three node sources, a different, nearby node in the system is selected at which to monitor the concentration of the contaminant as it passes through that node. Graphs of the concentration as a function of time for each of these observation nodes are shown in Figure 9.7. The breakthrough curves are

shown for all six different hydraulic time step sizes in each graph. The source nodes and the corresponding node at which concentration is recorded are: J-20 (J-11); J-4382 (J-4391); and J-4991 (J-4901).

The breakthrough curves in Figure 9.7 show a number of interesting results. First of all, the curves for time step sizes of 3600 and 7200 seconds are not consistent with the curves calculated at the smaller time steps. Even within the smaller time steps, there is more variation from one curve to the next than was seen in the results for a contaminant source in the tank.

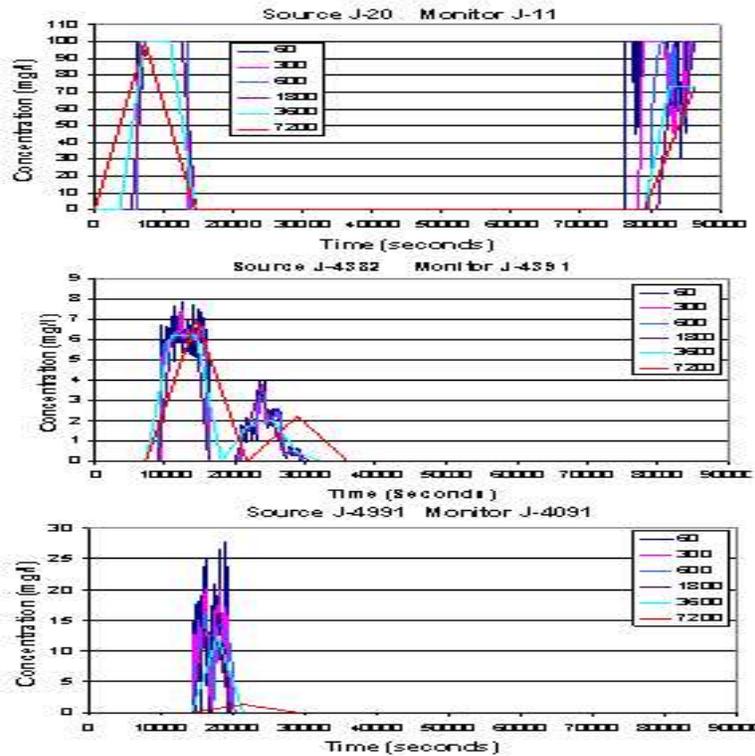


Figure 9.7. Solute breakthrough curves at three monitoring nodes form injection at the nodes sources.

The injection at J-20 creates a breakthrough curve with a double peak when monitored at J-11. This is due to the injection pushing the contaminant past J-11 and then the small amount of demand at J-11 (base demand = 0.23 gpm) pulling the contaminant back after the injection has ceased. In this area of the network, demands are low enough such that the period of contaminant injection dominates the local hydraulics. The double peak BTC at J-4391 from the injection at J-4382 is caused by looping within the pipe network. There are two flow paths between the source location and the monitoring location and transport along both paths causes the two distinct peaks. The transport between J-4991 and J-4091 is along one flow path and the BTCs show a single peak with considerable high frequency variation at the 60-second time step.

9.8 Solute Dispersion

Work in dead-end mains with multiple nodes shows that a pulse of solute can be smoothed out due to variable demands removing water and solute from the main at locations both in front of and behind the current location of the solute pulse. One approach to model this smoothing of the pulse is to apply a dispersion-based model. Dispersion is essentially the observed macroscopic behavior of many small-scale diffusive and advective variations acting on the solute pulse in either laminar or turbulent flow conditions (e.g., [220], [219], [7], [5]). These small-scale random perturbations generally result in the macroscale behavior of the pulse being described by a Gaussian distribution. While a Gaussian-based dispersion model may be capable of describing the smoothing of a solute pulse in a dead-end main, it does not appear to be capable of doing so in a pipe network with variable flow directions and mixing of water due to looping. In these cases, a more sophisticated description of the flow system must be incorporated into the solute transport equations such as has been introduced by Tzatchkov et al. [225].

9.9 Conclusions

Simulations of the flow and solute transport in a water distribution system show that flow across the system as a whole is less sensitive to changes in the demand discretization than is flow along dead-end mains. For both sets of simulations, tank source and node sources, discretization of the demand into 3600 (1 hour) and 7200 (2 hour) time steps is too coarse to obtain accurate transport results. The demand must be simulated at the 1800-second or smaller time scale. It is noted that the time step changes in these simulations are applied only to the demand discretization and the hydraulic time step; a water quality simulation time step of 10 seconds was used in all simulations. Variations in time steps at and below the 1800 second limit caused only minor differences in solute transport timing. These smaller time steps also modeled the sharp front of the breakthrough curves from the tank source with only minimal mixing of the solute with clean water. These results indicate that high-frequency variations in flow do not cause mixing; however, these results may be strongly influenced by the Lagrangian transport scheme in EPANET.

Results obtained for simulations with node sources show more variation than those calculated with the tank source. The timing, transport distance, distortion of the contaminant source function and the amount of contaminant mass removed from the system in one day are all strongly influenced by the local flow system at different points in the network. The variability in the node source breakthrough curves increases with decreasing demand time step scale. Multiple peaks in the breakthrough curve can be caused by changes in the hydraulics due to starting and stopping of injection (J-20) or multiple paths between a source location and a monitoring location (J-4382). The center of consumed mass locations show that contaminants introduced into a tank will spread further and faster than those injected into a node; however, those results are dependent on the system and the details of the source injection. From a safety perspective, these results indicate the emphasis on security should be placed on tanks rather than nodes in the system. However, the complete mixing of the contaminant in a tank results in longer times for all of the mass to be consumed relative to some of the node injections.

Chapter 10

Source Location Inversion and the Effect of Stochastically Varying Demand

Sean A. McKenna, Bart van Bloemen Waanders, Carl D. Laird (Carnegie Mellon University), Steven G. Buchberger (University of Cincinnati), Zhiwei Li (University of Cincinnati), Rob Janke (EPA)

10.1 Background

The potential of accidental contamination of water distribution systems is not new. Additionally, any water outlet, such as a hydrant or even a household water faucet, can be an access point for backflow contamination into the network. Physical security can only provide a limited amount of protection. As an alternative to physical security alone, sensors can be installed in the network to detect contamination and initiate a containment and restoration strategy.

Given the necessity of applying a limited number of sensors within a distribution system to detect a contamination event, the optimal location of sensors within networks has been an area of active research. Much of this research effort has been focused on development of algorithms for both the optimal placement of water quality and, eventually, contaminant specific, sensors within water distribution systems as well as inversion algorithms for using information from sensor networks to identify the source location of the contaminants. The sensor placement algorithms include both heuristic sensor location optimization schemes (e.g., Ostfeld and Salomons [164]; Uber et al. [226]) and exact solution techniques including integer programming approaches e.g. ([33]). Relatively less attention has been paid to the source location inversion problem, but in recent years advances in the application of gradient-based optimization to solve this problem have been realized [137] [232].

An aspect of water distribution systems that has not yet been considered as part of the source location inversion process is that demands within the system are unknown and typically of a much shorter time scale than the length of the hydraulic time step used in distribution system models. Over the past decade,

considerable effort has gone into characterizing small-scale demands within networks. Buchberger et al [60] monitored demands in a small neighborhood at a one-second time interval for a period of nine months. This extensive data set was used to corroborate the Poisson Rectangular Pulse (PRP) model of residential water use, first proposed by Buchberger and Wu [65]. This model has been used to examine the effect of fine scale demand patterns on the transport of tracers within distribution systems [153]. Errors in the prediction of tracer concentrations can occur when the true fine time scale demands are averaged over larger and larger time steps.

Here we examine the ability of a contaminant source location inversion approach to correctly identify the source location using a network model with larger hydraulic time steps than those used in the true contaminant transport. This work combines recently developed source location inversion algorithms for water distribution networks with a new PRP-based demand simulator. The goal of this work is to identify how well the inversion algorithm can identify the true source location as the variable demands are aggregated over larger and larger time steps.

10.2 Simulation Approach

This chapter brings together two distinct lines of research: characterization of instantaneous water demands and non-linear inversion schemes for source location identification. Stochastic simulation is used to generate water demands at different scales of temporal discretization and dynamic optimization techniques are used to solve the source location problem in real time. A brief background on each of these simulation approaches is given below along with references where more detail can be obtained.

10.3 Demand Simulation

Residential water demands at single family homes are assumed to behave as a nonstationary PRP process [65]. Under the PRP hypothesis, the frequency of residential water use follows a Poisson arrival process with a time dependent rate parameter. This process produces an exponential distribution of arrival times. When a water use occurs, it is represented as a single rectangular pulse of random duration and random but steady intensity, or consumption rate. When a home draws a pulse of water from the supply network, it is considered to be "busy"; otherwise, the home is considered "idle".

Buchberger and Wells [64] found that over 80 percent of indoor residential water demands occur as single pulses and that complex demand patterns are easily converted to an equivalent single pulse. By virtue of the Poisson assumption, it is unlikely that more than one pulse will start at the same instant. Owing to the finite duration of each water pulse, however, it is possible that two or more pulses with different starting times will overlap for a limited period. When this occurs, the total water use at the residence is the sum of the joint intensities from the coincident pulses. Owing to its simplicity and versatility, the PRP approach offers an effective new way to model the temporal and spatial variability of residential water demands across a municipal distribution system.

A computer program PRPsym was developed to simulate residential water demands at various time averaging intervals for each node in a municipal distribution system. The simulation process involves three steps:

Step 1: Generate instantaneous PRP water demands at a 1-second resolution. a) Draw the time, T , to the next demand from the exponential distribution of times between demand, $\phi_T(t)$, consistent with the specified Poisson arrival process. b) For each new demand pulse draw the duration, $D[T]$, and the intensity, $Q [\frac{L^3}{T}]$, from their respective log-normal distributions, $\phi_D(d)$ and $\phi_Q(q)$. c) Check for end of simulation time, if not then return to step 1a.

Step 2: Integrate the instantaneous demands. Some water demand pulses at a node will overlap. Add the coincident pulses to get the total instantaneous demand at a node.

Step 3: Convert the total instantaneous water demands to an equivalent series of time-averaged intensities. Divide the total volume of water use during a specified time step by the duration of the time step to get the time-averaged water demand.

Scaling of the Poisson arrival parameter allows for demand nodes to be configured to represent from 1 to over 1,000 homes. Each node requires a PRP pulse template to specify statistical properties of indoor and outdoor water demands and a multiplier pattern to define the hourly variation in arrivals for indoor and outdoor water use that occur throughout the day.

10.4 Non-Linear Optimization for Source Location

Details on the application of non-linear optimization approaches to the problem of identifying the location of a contaminant source within a water distribution network can be found in Laird et al [137]. The approach is outlined briefly below.

With known flow rates and velocities as inputs, the water quality model for the network, using P , J and S to refer to the complete sets of all pipes, junctions, and storage tanks, respectively, is developed. The concentration in the pipes is denoted as $c_i^p(x;t)$; $i \in P$ and $c_k^n(t)$; $k \in N$ represents the concentration at the nodes, where $N = J \cup S$ is the complete set of all nodes, including junctions and storage tanks. Here, $t \in [0 :: t_f]$ is time, and $x \geq 0$ is the displacement of flow along a pipe. In developing the model, we refer to connections and concentrations at pipe boundaries. Note that these designations are time dependent and change with the flow direction. Pumps and valves are modeled as zero length pipes, and reservoirs are modeled as junctions with known external sources. We assume there is no decay reaction for the contaminant, although first order decay can easily be included in the formulation. Consistent with the transport formulation within EPANET, plug flow and perfectly mixed concentrations, no dispersion, are assumed.

The goal of the optimization algorithm is to minimize an objective function, Ψ , that is the weighted sum of squared errors between predicted and observed concentrations at the nodes:

$$\min_{m(t), c^p(x,t), c^n(t)} \Psi = \sum_{r \in \Theta_s} \sum_{f \in N_s} \frac{1}{2} \int_0^{t_f} w_k(t) [c_k^n(t) - c_k^{n*}(t)]^2 \delta(t - t_r) dt \quad (10.4.1)$$

Where $m(t)$ is the unknown contaminant mass flow rate, $w_k(t)$ is a time dependent flow based weighting scheme for each node, this weighting shifts the errors from being between measured and predicted concentration to mass, $c_k^{n*}(t)$ is the measured concentration at node k , $\delta(t)$ is the Dirac delta function, t_f is the final simulation time and r indicates the time step at which concentrations at sensors are evaluated. This function is minimized subject to physical constraints on transport of the contaminant within the pipes:

$$\frac{\partial c_i^p(x,t)}{\partial t} + u_i(t) \frac{\partial c_i^p(x,t)}{\partial x} = 0 \quad \forall i \in \mathcal{P} \quad (10.4.2)$$

$$c_i^p[x = I_i(t), t] = c_{k_i(t)} \quad \forall i \in \mathcal{P} \quad (10.4.3)$$

$$c_i^p(x,t) = 0 \quad \forall i \in \mathcal{P} \quad (10.4.4)$$

As well as conservation of mass within the tanks and nodes. Additionally, a regularization term is added to the overall objective function to force a unique solution to the inverse solution. The discrete forms of the objective function and the physical constraints on mass transport as well as the equations for conservation of mass are solved using a direct simultaneous method (see Laird et al. [137]).

10.5 Example Problem

The problem of inverse source location using different hydraulic time step length under conditions of variable demand is illustrated with a set of calculations on an example distribution network. The distribution network is shown in Figure 1 and has 1 tank, 394 nodes and 534 pipes. The base demands are shown in Figure 1. Base demand at any one node ranges from zero to over 150 gpm. The base demand at each node has a periodic trend throughout the day. The PRPsym code is used to generate demand multipliers that are then applied to these base demand values.

The concentration is injected at node # 435 (Figure 10.1) as a square wave at hour 6 with a duration of 1 hour. EPANET is used to solve the hydraulic and transport equations. A 5 minute water quality time step is used in all simulations and the hydraulic time step is varied as discussed below. A total of 100 sensors are distributed randomly throughout nodes in the system with the probability of selecting any given node as a sensor location being approximately proportional to the amount of flow through that node. The sensors are assumed to provide error and noise free readings of the actual concentrations.

The parameter values for residential water demand simulations done herein are listed in Table 1. Two types of output are produced by the PRPsym software for each node: (i) water demand time series and (ii) water demand simulation statistics. The PRPsym code is used to generate demands for 0.5, 1, 2, 4, 6, 12 and 24 hour time step lengths. The base case simulation used as the ground truth in this work simulates

instantaneous demands using the parameters in Table 1 and then aggregates these demands over 30 minute time steps. These aggregated demands are used in EPANET for simulation of the transport from node 435. The concentrations observed at the nodes with sensors are then recorded as the observed concentrations.

The same demand generation parameters, Table 1, are then used to generate demands aggregated over 1, 2, 4, 6, 12 and 24 hour time steps. The base demand at each node is the same for all simulations and the total demand across the entire network is preserved across all aggregation levels. The random number seed used to generate the stochastic demands is also kept the same across all time step sizes.

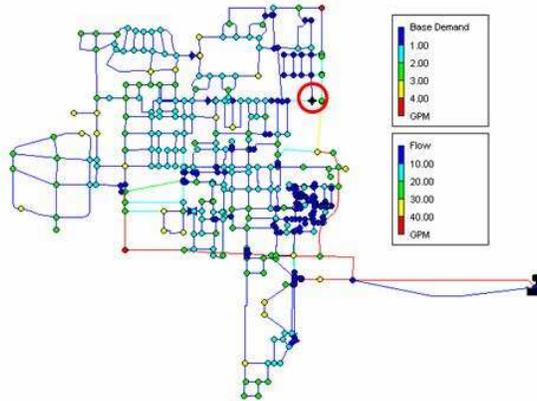


Figure 10.1. Example network used in simulations. The color scales show the base demand (gpm) at the nodes and the flow (gpm) in the pipes. The red circle shows the location of node 435, the contaminant source.

(*) Values are derived from Buchberger and Wells [64] and Buchberger et al [60].

The inverse source location approach minimizes Equation 1 by matching the data observed in the base case simulation with 30 minute time steps with each of the models having larger time steps. The inverse approach optimizes the source location to achieve this minimization. Only a single EPANET simulation is necessary to provide the hydraulic information for each time step size. For each time step length, the ability of the inverse approach to identify the correct source node is quantified as discussed below.

10.6 Results and Discussion

For each inverse estimation, the fraction of the total contaminant mass in the source term is partitioned to different nodes within the network. An exact solution would assign a fraction of 1.00 to the true source node. The inverse approach with regularization results in a very small fraction of mass being assigned at all nodes in the network; however, here we only identify the nodes in the solution that are assigned a mass fraction of at least one percent of the maximum mass fraction assigned to any node. For example, Figure

Demand Characteristic	Indoor Use
Intensity: mean	2.00 gpm
variance	1.56 gpm^2
coef of var	0.62
distribution	log-normal
Duration: mean	1.00 min
variance	4.00 min^2
coef of var	2.00
distribution	log-normal

Table 10.1. Input parameters for the PRP demand simulator, PRPsym

10.2 shows the scaled mass fraction for an inverse solution with 2 hour time steps. Four nodes are assigned a significant fraction of the source mass, nodes 435, 2301, 431 and 2401, with the majority of the mass assigned to the true source node (435).

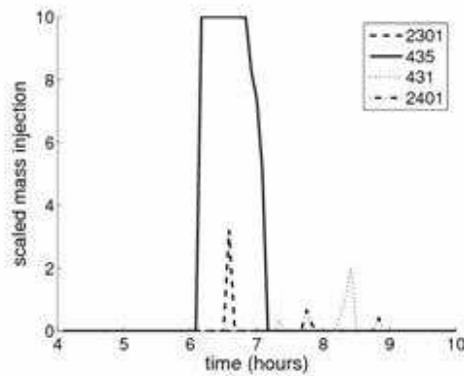


Figure 10.2. Example output showing amount of source mass assigned to different nodes

The results of the inverse source location calculations are summarized using three different quantities. The final value of the objective function (Equation 1) indicates the ability of the inverse model to match the concentrations observed at the sensor locations through adjusting the location of the source. These final objective function values are shown in the left image of Figure 10.3 and show that the sum of the squared errors between the observed and modeled concentrations increases linearly with increasing time step size up to time steps of six hour length.

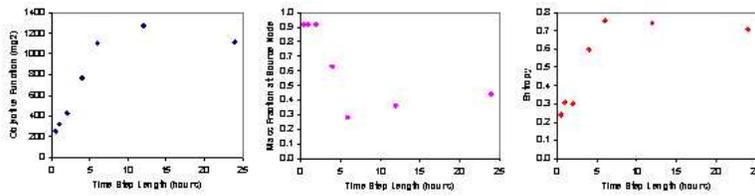


Figure 10.3. Summary of results showing the objective function value (left), the mass fraction assigned to the correct source node (middle), and the entropy (right) all as a function of time.

The ability of the inverse approach to find the correct source node is quantified by the fraction of the total mass that is assigned to the true source node (middle image, Figure 3). For time step lengths of 2 hours or less, over 90 percent of the total mass is assigned to the correct source node. For longer time steps, the mass fraction correctly assigned to the source node decreases to less than 50 percent for all time step lengths greater than 6 hours. The assigned mass fraction results in the middle image of Figure 3 do not directly address the number of nodes to which a significant amount of mass is assigned. For example, in cases where the mass fraction correctly assigned to the source node is less than 50 percent it is not clear whether the majority of the mass was incorrectly assigned to a different node, or if the largest proportion of the mass was still assigned to the correct node, but the remaining amount of mass was spread over a greater number of incorrect nodes. All results were checked and in each case, the true source node was the node assigned the largest amount of mass. In general, the larger the time step length, the larger the number of nodes that were assigned a significant fraction of the mass.

Ideally, the amount of mass assigned to the correct node and the number of nodes over which the mass is spread could be summarized in a single term. Toward this goal, a measure of entropy is introduced as a way to quantify the uncertainty in assignment of a single node as the source node. The entropy is calculated as [7]:

$$H = \frac{-\sum_{k=1}^g \ln(m_k)}{\ln g} \quad (10.6.5)$$

where m_k is the mass fraction assigned to the k th node out of a total of g nodes assigned a significant fraction of the mass. As $m_k \leq 1.0$ for any k , $H \geq 0$. Conversely, H reaches a maximum value of 1.0 when all m_k are equal and the determination of the most likely source node is highly uncertain. The entropy of mass fraction provides a convenient single measure of uncertainty and is adopted from similar approaches with classification probabilities; however, it is noted that the mass fraction is not directly equal to the probability of the node being the true source location. This change in entropy definition will be examined further in future work. Here, g is set to equal the total number of nodes with a mass fraction that is at least one percent of the maximum mass fraction assigned to any node. Therefore, g varies in these calculations from 3 to 14. The calculated entropy value increases to nearly 0.80 with increasing time step size up to a time step size of 6 hours after which it decreases slightly (Figure 3, right image). An alternative calculation for entropy would be to set g to the total number of nodes in the distribution system for all calculations. This

approach will be also be explored in future work.

Figure 3. Summary of results showing the objective function value (left), the mass fraction assigned to the correct source node (middle) and the entropy (right) all as a function of time step size.

10.7 Conclusions and Future Work

The results above show that the inversion approach is capable of identifying the correct source node over all time step sizes investigated. These results indicate that an actual contamination event that is responding to the very fine-scale demands within a distribution system could be identified using a simulation model with much larger time steps. The ability of the inverse source location identification approach to find the true source node becomes less certain for larger time steps, see the measures plotted in Figure 3, but for all of the cases examined here, the identification was accurate – the approach assigned the majority of the source mass to the correct source node across all time step sizes.

The results show that, in general, the quality of the source identification using the inverse approach degrades with increasing aggregation of fine-scale demands up to a time step size of approximately 6 hours. The quality of the source identification, whether measured in terms of the objective function, the fraction of the total mass assigned to the true source or the entropy, is roughly constant for the time step sizes beyond 6 hours (Figure 3). The reasons for this asymptotic behavior in the results will be examined in future work. The results in Figure 3 suggest that the solutions become less unique with larger time steps. One way to examine the uniqueness of these results with respect to the aggregated demands will be to generate multiple stochastic demand patterns for each time step size and characterize the range of objective function, mass fraction and entropy values across all simulated demands. Other simulations that can be done to assess the sensitivity of the results presented here include varying the parameters used in PRPsym (Table 1) to generate the demands and to vary the number of sensors within the network. Rapid assessment of these additional simulations is feasible due to the minimal computational burden that can be achieved by the inversion approach used herein.

Chapter 11

Threat Assessment of Water Supply Systems Using Markov Latent Effects Modeling

Vincent C. Tidwell, J. Arlin Cooper, Consuelo J. Silva

11.1 Background

To assist the water utility industry in conducting vulnerability assessments, EPA in partnership with the American Water Works Association Research Foundation and Sandia National Laboratories developed the Risk Assessment Methodology for Water Utilities (RAM- WTM) [94]. This performance-based, consequence-driven, risk management tool aids in identifying system vulnerabilities, earmarking critical facilities and assets, and determining the level of protection to which the security system should be designed. Central to the methodology is the evaluation of risk,

$$R = P(1 - P)C \quad (11.1.1)$$

where R is risk, P_a is the probability of attack, P_e is the probability of system effectiveness, and C is the consequence. Operationally, risk is determined through a coordinated consequence analysis (to define C) and threat assessment (for quantifying the term $P_a(1-P_e)$). Risk assessment is performed primarily through a process of expert elicitation in which values for each term in the risk equation are quantified according to a structured and defined scale of high, medium or low. In this way, RAM-WTM is designed to facilitate comparative analyses relying on relative rankings determined from the risk equation.

Long before the events of September 11, 2001, water utilities were performing risk-based analyses to quantify vulnerabilities to natural events such as earthquakes, floods and the like. A variety of

methodologies are available to the analyst for performing these risk assessments (e.g., [10]; [11]; [2]). Recently, the American Water Works Association Research Foundation conducted a comprehensive review of the experience base with water utility disasters and offers guidance for risk management and analysis [103]. The project showed that available methods for risk analysis are in limited use by utilities because of a lack of data on threats and vulnerabilities and a lack of training and priority within utilities. Similar experiences have been encountered by water utilities performing vulnerability assessments, in which relevant threat data has been difficult to obtain. In turn, the utilities are faced with decisions concerning risk reduction programs, potentially costing millions of dollars, driven by this highly ambiguous data [81].

These results underscore the difficulty in conducting the threat assessment phase of any risk-based analysis. Although the governing relation for threat assessment is quite simple ($P_a(1 - P_e)$ term in Equation 1), defining the associated terms is difficult. The problem is insufficient experience and data for quantifying the associated probabilities; particularly, in the case of willful attacks. For this reason we propose a possibilistic approach to threat assessment as an alternative to the traditional probabilistic methodologies. Specifically, we propose Markov Latent Effects (MLE) modeling as a convenient threat assessment framework for analyzing subjective metrics within a quantitative, repeatable, and defensible process.

The purpose of this chapter is to introduce MLE modeling as an approach for water supply system threat assessment. We begin by describing the fundamental underpinnings of MLE modeling and then describe how it might be applied within the context of a water distribution system. Application of this approach is then demonstrated. Specifically, a generic MLE model is developed and applied to evaluate the security of select assets for a large municipal water utility.

11.2 Methods

The possibilistic framework adopted for water supply system threat assessment is Markov Latent Effects modeling [76]. The MLE approach provides a construct for quantifying imprecise subjective metrics through possibilistic or fuzzy mathematics. Below, the theoretical background underpinning MLE modeling is provided. Then, attention is turned to describing how MLE modeling can be applied within the context of a water distribution system threat assessment.

11.3 MLE Background

The Markov Latent Effects approach had its genesis in the mid-1990s, following a series of accident and security-breach investigations. These investigations revealed that there were strong cultural (e.g., lack of personal responsibility, poor communication, failure to address problems) and environmental (poor working conditions, time constraints, pressure to "look good") factors contributing to the events. As these factors were generally initiated at a time well before the occurrence of the event, they were termed "latent effects" [12]. Although the accident and security-breach investigations identified the need for mathematical treatment of these latent effects, their subjective nature defied conventional analysis. To address this need, Cooper [76] combined the concept of latent effects with a chained subjective analysis

methodology to formulate the $\text{\$Markov Latent Effects\T}$ modeling framework, named after A. A. Markov, who explored (in the late 19th century) the formal mathematical role of a chain of occurrences in determining subsequent events [3].

In order to give a systemic structure to the approach, a top-down mathematical decomposition strategy is employed. This enables determination of the most appropriate items to be measured, and for aggregation of the measurements as imprecise subjective numbers through possibilistic or fuzzy mathematics. The resulting metrics provide an excellent reference point for assessment and for aiding management decisions. Having a mathematical model to run test cases facilitates predictive exercises and helps convey the top-down system perspective, while mathematically portraying the results.

11.4 Decomposition Factors

The obvious focus in any operation is on the results of the operation itself. However, lying behind these operations are modes in which the operation was set up and implemented, the management of the operation, and the environment. A $\text{\$top-down\T}$ approach inverts the focus to make sure that potentially important hidden factors are identified and not neglected. This is best accomplished according to the following steps:

1. recognize environmental constraints and threats, 2. determine the management philosophy that is established within that environment, 3. evaluate the working conditions, and 4. assess the risks inherent in actual operation. This approach helps better identify risks, but more importantly, helps determine why problems might arise (a forward-looking approach) and helps identify what can be done to improve overall integrity (similar to an in-depth root cause and correction analysis, but also accommodating hypothetical events).

The MLE model has a performance-focused approach built around the timing of latent effects. These latent effects are identified through decomposition of the complex threat system into more manageable subsystems or decision elements that trace a particular threat from its inception to the point of consequence (Figure 1). In this way, the decomposition helps visualize the network of events that must coincide for a threat to achieve its intended consequence. For example, consider a contamination event targeting a treated water storage tank that might require the following set of events: utility budget cuts limit security upgrades; failure of utility to protect sensitive system information; poor maintenance practices by utility; assailant acquires critical system information; assailant training and mission preparation; defeat of the limited physical security protecting the tank; and, undetected breach of the tank hatch due to inoperable alarm. In this way, decomposition provides a basis for identifying credible threats and for visualizing all the latent effects that contribute to the success of a threat event.

A latent effect is an occurrence, condition, or behavior that does not necessarily cause an immediate problem, but can combine later with other occurrences, conditions, or behaviors.

Latent effects are represented sequentially because of the chained nature of the decomposition. As in the example above, a breach of the water supply tank was made possible by a sequence of latent effects; specifically, failure of the utility to make security upgrades, protect sensitive information and maintain the intrusion alarm system.

11.5 Decision Element Structure

Each decision element identified in the decomposition process represents a single factor that influences the likelihood that a threat will yield its intended consequence. Each decision element produces an output subject to a set of inputs (Figure 1). These inputs include external effects and/or latent effects. External effects are inputs unique to that decision element. Each external effect is subjectively assigned an attribute value. Attribute values are qualitative choices, mapped onto a scale of 0 (very weak) to 1 (very strong) that reflect the strength of the relationship between the decision element and the external effect. Latent effects represent the influence that one element imposes on another. Their attribute values are simply represented by the output value of the latent decision element.

Along with an attribute value each input is assigned a weighting factor. These weighting factors reflect the ability of an input, whether an external or latent effect, to influence the decision element relative to all other attributes contributing to that element. As such the aggregated sum of the weighting factors contributing to an individual decision element must equal one.

11.6 Data Aggregation

Data aggregation allows observed metrics to be combined to derive various sorts of information, such as combined ratings of subsystems or the entire system, and trends information. The strategy extends naturally to decision analysis, where decision aids must be developed for assessing system integrity, determining the need for operational restrictions, and selecting among alternative approaches or forensic hypotheses. Since measurements can lead to assessments when compared to norms or acceptance margins, subsequent analytical methodology and information presentation can contribute to a structured approach for defensible decision-making.

According to the discussion above, care must be given to the selection of the prescribed data aggregation scheme; that is, the manner by which the observed metrics (i.e., attribute values) are combined to yield quantitative information toward the decision of interest, in our case, information defining the credibility of a particular threat. For each decision element, input values are aggregated to provide an assessment score, and this process continues for each decision element until an overall assessment score is derived. There are a variety of weighted sums that can be employed, each having a different influence on the outcome. The choice of the weighed sum depends on one's confidence in the data gathering process and the nature of the decision process. One form that has been used is termed soft aggregation. An example of a soft aggregation weighted sum is:

$$WS_k = \frac{1}{1 + e^{-5.5(\sum_{i=1}^n w_i x_i + \sum_{j=1}^m v_j y_j - 0.5)}} \quad (11.6.2)$$

where WS_k is the weighted sum for the k th decision element, x_i are the individual attribute values input to each decision element, y_j are the latent effect inputs from prior elements, w_i and v_j are the corresponding weights, and n and m are the number of attribute values and latent effects contributing to the decision element, respectively. An alternative to soft aggregation is the simple linear weighted sum in equation ??:

$$WS_j = \sum_{i=1}^n w_i x_i + \sum_{j=1}^m v_j v_j \quad (11.6.3)$$

All aggregation schemes must have appropriate mathematical properties; specifically, the aggregate value is the same as each input value if all inputs are equal, regardless of weights, and the aggregate is no greater than the maximum input and no less than the minimum input, regardless of weights. Also, the results obtained from individual decision elements are simply an aggregate measure of the combined influence of all of the contributing variables.

11.7 Model Implementation

With the resulting MLE model, assessments are performed for specific threats by simply inputting attribute values specific to that threat. As attribute values are assigned, primarily through expert elicitation, care must be taken to reduce subjective variation in the attribute value to a minimum. That is, the process must be repeatable; it must be assured that similarly qualified people seeing the same situation would record similar, but not necessarily exactly the same, results. Repeatability is pursued by specifically defining the numerical representation of the qualitative information during the elicitation process. This is accomplished through the use of elicitation guides (e.g., Figure 2) during the elicitation process.

Issues of uncertainty can easily be addressed within the MLE modeling framework. Uncertainty inherent to the Markov inputs arises from two sources. One is that a person entering an input may be unsure of the precise value. Another is that a collection of people collaborating on the input may not agree on exactly the same value regardless of the amount of guidance provided. To capture this uncertainty, an attribute value can be represented by a range of numbers rather than a single deterministic value. This suggests that the resulting output will be in the form of a range of numbers rather than a single value.

Other important features of MLE modeling are the Importance and Sensitivity metrics. The Importance metric allows one to identify those features that most significantly contribute to the success of a threat. The Importance metric is simply calculated by deriving the difference between the output value with the input as entered and the output value when one of the input values has been set to zero. By repeating this process the input with the greatest influence on the output can be found. Similarly, the Sensitivity is calculated as the difference between the output value with the input as entered and the output value when one of the input values has been set to one. The Sensitivity metric measures the potential for improvement in that input to result in a measurable improvement in the total result.

To summarize, an MLE threat assessment model consists of a network of decision elements and inputs as defined through the decomposition process. The model is populated with a set of fixed weighting factors and a prescribed data aggregation scheme. Evaluation of a particular threat is accomplished by assigning the corresponding attribute values (with help from the elicitation guides, e.g., Figure 2) and aggregating over all decision elements to provide the assessment score. The resulting assessment score provides a measure of the credibility of that threat. By consistently following the same analysis scheme, assessment scores calculated for different threats can be compared and ranked to identify the most credible threat.

11.8 Results

We now explore MLE modeling in more concrete terms, drawing specific application to a threat assessment of water distribution systems. We begin our application by developing an MLE model for assessing the security of water distribution system assets to willful attack. We then use this model to perform a preliminary asset security assessment for a large municipal water utility.

11.9 MLE Model

An MLE model for assessing water utility asset security was developed for demonstration purposes. The model is formulated within a fully generic context; that is, it has direct application to most any water utility. The model is generalized to a particular utility by the attribute values supplied in the analysis. The MLE model structure is presented in Figure 3 while a description of each external input variable is given in Table 11.1.

The model is structured around three basic decision elements, Detection, Delay, and Response (Figure 3). These same three elements are the basis of many critical asset assessments (e.g., [13]) and also underpin the RAM-WTM process. Each of these elements are latent effects of the output decision element, Asset Security. However, Detection is not a direct latent effect of Asset Security, but rather feeds decision elements contributing to the Response decision element. This choice of structure is deliberate in that detection has no value unless there is an associated response. Alternatively, both the Delay and Response elements are direct latent effects of Asset Security. This structure reflects the causality that security depends on whether a response will occur and that the response occurs before the attack reaches its full effect (e.g., a high score for Asset Security represents the case where a mitigating response is likely to occur and will occur well before the attack takes effect).

The Security Environment decision element is a latent effect of elements contributing to the Detection, Delay and Response elements. This decision element is a measure of the concern and awareness that utility employees have toward system security, which influences the likelihood and speed with which personnel will detect and respond to abnormal operations. The Security Environment element is comprised of two decision elements, Employees and Management, each carrying a weighting of 0.4. Each of these elements have external inputs (see Table 11.1 for description) representing the security Culture in which they operate and security Training they receive. In addition, the Management element has an additional input of Security Administration. Intelligence, weighted with a value of 0.2, is also a direct external input to the Security Environment decision element.

The Detection decision element relates to the different methods of detecting breaches/contamination of a water distribution system. There are four primary decision elements that form latent effects to the Detection element. The first is Remote Monitoring including Alarms and Video surveillance. Each of these elements are subject to inputs that measure confidence in the detection system, these include the Surveillance of the detector output, and the Reliability of the detector. The Reliability decision element is further broken into Quality, Maintenance, and Sensitivity. The second decision element that is a latent effect of Detection is System Monitoring, which addresses the ability of inline sensors (e.g., pressure, flow,

chlorine) to detect changes resulting from an attack. Inputs to this element are similar to that discussed above. The third decision element is Physical Monitoring that considers the presence of Onsite Personnel and Onsite Security stationed at an asset. Finally, Public Surveillance considers the likelihood of a member of the general public noticing something unusual and reporting it to the utility. Remote Monitoring, System Monitoring, Physical Monitoring and Public Surveillance are each weighted according to how likely utility personnel are to respond to the alarm.

The Delay element quantifies the balance between the amount of time for an attack to reach its full effect and the time before a response occurs. Three decision elements are latent effects to the Delay element. Time to Respond accounts for the time required to respond to an attack. External inputs to this element include Detection Time, Response Time, and Decision Time. The other two latent decision elements to Delay are Critical Attack Time and Physical Security, both of which are measures of the time it takes to accomplish an attack. The Physical Security element considers any delay measures that an assailant would have to breach in order to carry out a successful attack. These include the overall physical security of the site like Barriers, Accessibility, and Distance. The Critical Attack Time measures the time it takes an attacker to carry out an attack on a water utility. Note that all weighting values and attribute values are scaled so as to represent equivalent times. Thus, a high Delay value represents the case where a response is likely to occur well before the assailants could accomplish their attack, while an intermediate value represents the case where the response and attack time are nearly equal.

The Response element measures both the type of response and the likelihood of a response. The Type of Response (i.e., strength) has inputs of Number of Respondents, Degree of Training, and Respondent Capability. Likelihood of Response is a function of Response Procedures, Dedicated Response Personnel, and Detection. Likelihood of Response is weighted more heavily, 0.7, than the Type of Response, 0.3 given that the type of response does not matter if no response is made.

The MLE model presented above has been implemented within the commercial spreadsheet package, Excel. Interactive interfaces have also been developed using Visual Basic. The interfaces provide the user with a set of menu driven instructions that act as a guide through the analysis process. Specifically, the user queries each of the elicitation guides and makes an input selection. These values are uploaded into the model and the accompanying calculations performed. Results are then organized and ranked for inspection by the user.

11.10 MLE Model Application

Application was performed to demonstrate the capabilities of the MLE model. Specifically, the model was exercised to show how it could be used in the threat assessment process, to demonstrate the type of results that are obtained and how they can be interpreted in the context of a threat assessment, and how the MLE model can be used to identify mitigation strategies. The MLE model demonstration was performed for a real municipal water distribution system that supports a community with over 500,000 residents distributed over a 300-square mile area. The source of data for our analysis is the vulnerability assessment performed for the community approximately 2 years ago.

The first step in the analysis process involved defining a suite of threat scenarios for testing. These

scenarios largely followed from those utilized in the vulnerability assessment. Seven different threat scenarios were selected for testing. These scenarios cover a range of utility assets (well field, treatment plant, clearwater reservoir, and back-flow injection at a residential connection) and consider two different modes of attack (bomb and injection of a toxic contaminant).

To model a particular threat scenario, each of the 36 attribute values were quantified and input in the model. This was accomplished with the aid of elicitation guides (e.g., Figure 2). Data specific to an attribute value were taken from the vulnerability assessment documentation and compared to the criteria given in the elicitation guide to determine an appropriate score for that attribute value. In this way, each attribute value was assessed according to a unique set of criteria as spelled-out in its elicitation guide.

Attribute values used in the assessment of each of the threat scenarios are given in Table 11.2. Attribute values supporting the Security Environment and Response decision elements are constant across all scenarios as these components deal with characteristics associated with the utility as a whole (see Table 11.1). In general scores are low for these attribute values as the utility has a relatively poor respect for security related issues and is poorly prepared to respond to a threat event. Scores for attribute values contributing to the Detection element vary according to the monitoring systems employed at a particular asset. Values tend to be higher at the treatment plant where there are multiple forms of monitoring (e.g., on site security and personnel, video/alarm systems, in-line sensors) and much lower for residential monitoring (in-line sensors only). Attribute values contributing to the Delay element are based in part on the types of barriers and degree of access available to the asset. The number and sophistication of the barriers an assailant must defeat adds time to the attack and thus results in a higher score for the Physical Security decision element (latent effect of Delay). Likewise the more time required to implement an attack and for the attack to have impact, the larger the score for Critical Attack Time (latent effect of Delay). Because a bomb has immediate impact while a contaminant must be dispersed to utility customers, the Critical Attack Time is higher for contaminant scenarios relative to those involving a bomb. Alternatively, the longer the time to detect, decide and respond, the smaller the score for Time to Respond (latent effect of Delay).

Scores for each of the threat scenarios were calculated by inputting each of the attribute values. The attribute values were then linearly aggregated (Equation 3) subject to the applied weighting functions for each decision element. Aggregation began with the decision elements with no latent effects and subsequently progressed up the decision structure until the final decision element was reached, Asset Security. Aggregate scores for each of the decision elements are given in Table 11.3 along with the model output given by the Asset Security element.

Scores calculated for the seven different scenarios range from 0.26 to 0.39. To help interpret these results, consider that a score of zero implies that the attack will always succeed given the current set of security measures; alternatively, a value of one suggests that the attack will never succeed. Within this context, these scores suggest that these system assets are accompanied by a low to moderate level of security. That is, a bomb or contaminant attack waged against these assets would be successful in most cases. Additionally, these scores provide a ranking of assets most susceptible to attack. In this example, the clearwater reservoir is at greatest risk followed by the well field, residential connections and finally the treatment plant. Utility assets are also more susceptible to a bomb attack than to the injection of a toxic contaminant.

To demonstrate how the MLE model handles uncertainty, input values to one of the threat scenarios were treated as uncertain. Specifically, the contaminant attack on the clearwater reservoir was modeled with 5 of

its attribute values defined over a range rather than as a single number. In each of the 5 cases the attribute values were allowed to vary by 0.2 (Table 11.2). In turn, each of the decision elements that are direct or latent effect of these uncertain attribute values scored aggregate values over a range (Table 11.3). The overall effect is seen in the output score, Asset Security, that ranges from 0.36 to 0.39. From this example it is apparent that the range of uncertainty in the input is considerably different than that of the aggregate output score.

Analysis of the decision structure can help understand why a particular threat scored the way it did. From the model (Figure 3) it is apparent that the Security Environment and Response decision elements play a key role in asset security; however, the utility scored poorly in these two areas. For this reason, all asset scores fell in the low to moderate range (Table 11.3). Differences in the scores among the four utility assets can be tracked to the Detection and Physical Security decision elements. Specifically, the treatment plant always scored best relative to other assets (for the same attack) because of increased detection capabilities, primarily in the form of on-site security and personnel. The well field and clearwater reservoir scored better than attacks on a residence as these assets are protected by fences and buildings (i.e., better Physical Security) and are monitored by intrusion alarms on gates and doors. Further inspection of the model indicates why assets were more susceptible to a bomb attack relative to a contaminant attack. First, there is no in-line monitoring that can detect a bomb attack prior to detonation (in-line sensors could detect a contaminant). Second, the time to stage the attack and for the attack to have impact is much shorter for a bomb than contaminant. While the bomb has impact at the time of detonation, the contaminant must disperse in the system, thus providing increased time to respond before the event achieves its full impact.

Sensitivity analyses were performed for each threat scenario and attribute value. The two attributes having the greatest effect on Asset Security, for each threat scenario, are designated in Table 11.2. Although not shown, change in any single attribute value had relatively small effect on the outcome. Even in the case of the most sensitive parameters, Asset Security only changed by 0.01-0.02 when increasing the attribute value to 1 (while leaving all other attribute values unchanged). This suggests that no single attribute dominates asset security; rather, one must think in terms of systems of attributes when assessing system security.

Results of the threat assessment taken with the decision structure can help utility managers formulate constructive mitigation strategies. First, the ranked threat scores can help managers identify assets that need particular attention. Second, subsequent runs of the MLE model can be performed to determine how specific system upgrades will improve asset security. For example, consider improving the inline water quality monitoring system by installing sensors that are highly accurate, analyte specific, and that are optimally distributed throughout the water supply network. Such an improvement increases the Asset Security score from 0.38 to 0.39 in the case of a contaminant attack on the clearwater reservoir (Table 11.3). Alternatively, if the new sensor system is accompanied with an improved sense of commitment to security by managers and employees alike and improved response procedures/capabilities the score almost doubles (0.38 to 0.66) over the base line case (Table 11.3). Again, this result shows the importance of thinking in terms of systems of security rather than focusing on single attributes.

11.11 Discussion

In the section above, a generic MLE model was developed and applied to evaluate asset security for a municipal water utility. Purposes of this effort were purely demonstrational. As such, it should be noted that the asset security model developed here does not constitute a full threat assessment; rather, other elements, potentially of similar or more complexity, are required. Specifically, the system effectiveness term in Equation 1 will require additional modeling of the assailant's capabilities. This might involve consideration of such factors as the level of motivation, experience/training, and support network. The sophistication and access to the weapon to be used in the attack will also contribute to the system security term.

Threat assessment also requires a quantification of the probability (or possibility in our case) of attack (see Equation 1). Our current thinking has three broad decision elements contributing to this term. That is, the decision to attack a particular asset is based on a balance between the perceived consequence, the ease with which the attack can be made, and the proximity of the assailant to the utility. The perceived consequence deals with such factors as health effects, degree of disruption, economic loss, and attention gained. The ease of attack would deal with the degree of planning, cost of time and money, and likelihood of being caught. The assailant's proximity addresses the idea that it is unlikely for a terrorist to attack a relatively obscure utility, while the chances of attack by an unhappy employee or vandal may be much higher. However, these are only preliminary ideas and are subject to considerable change as our understanding matures.

Consider again the results in Table 11.3 that address only asset security. Although these results help identify which assets are most susceptible to attack, they don't fully address which assets are most likely to be attacked. To go this next step requires integration of the additional decision elements as described above.

As this integrated threat assessment model is developed, several checks of the model will be performed. First, an exhaustive set of comparisons drawn with utility vulnerability assessments will be completed to evaluate the reasonableness of the model's results. Such efforts will follow closely the procedure demonstrated here. Once a reasonable level of confidence is developed in the model, a peer-review by subject experts will be performed. Requests will be made of a broad cross-section of experts with training in utility operations, security, assailant profiling, weapon systems, health effects, and economics. Finally, comparisons between modeling results and data available on willful attacks on water systems will be drawn. Specifically, data concerning vandalism, and acts of retaliation by employees will be targeted. To the extent available, comparisons will also be drawn with the limited data on terrorist activities against water systems. Comparisons will attempt to link the actual frequency of attacks against utility assets with the ranked threat assessment scores determined from the MLE model.

11.12 Conclusions

Water utilities are faced with the daunting task of assessing the vulnerability of their utilities to disruption by natural and malevolent acts. Integral to such efforts is threat assessment, which is complicated by the lack of data and experience to conduct a full probabilistic assessment. Markov Latent Effects modeling is

Response Type	Number of Respondents	Number of people dispatched to investigate an alarm
Response Type	Degree of Training	Type and duration of training for respondents
Response Type	Respondent Capability	Measures how well the respondents are equipped
Likelihood of Response	Response Procedures	Existence, training and adherence to rules on alarm response
Likelihood of Response	Dedicated Response Personnel	Number of personnel with priority of responding to alarms
Physical Security	Asset Barriers	Level of asset protection by fences and other types of barriers
Physical Security	Distance to Barrier	Distance between exterior point of access and the physical asset
Physical Security	Asset Accessibility	Is equipment required to access asset (e.g. buried or raised?)
Time to Respond	Response Time	Time for respondent to arrive at site of alarm
Time to Respond	Detection Time	Time to detect breach or attack
Time to Respond	Decision Time	Time required to decide to respond to alarm
Critical Attack Time	Time to Implement Attack	Time necessary to prepare for attack once asset is accessed
Critical Attack Time	Time for Attack to Impact	Time for attack to exceed 50% of its potential total impact
Alarms/Video/Sensors	Surveillance	Percentage of time that the alarm is monitored
Alarms/Video/Sensors Reliability	Quality	Frequency of false positives
Alarms/Video/Sensors Reliability	Maintenance	Frequency of maintenance performed on the security system
Alarms/Video/Sensors Reliability	Sensitivity	Ability of sensor to accurately distinguish abnormal event
System monitoring	Sensor Location	Sufficiency and optimality of distribution of sensors
Physical Monitoring	On-Site Security	Number of man-hours site monitored by security officers
Physical Monitoring	On-Site Personnel	Number of utility personnel and time they spend at the site
Public Surveillance	Likelihood of Reporting	Historical frequency of public reporting abnormal events
Public Surveillance	Visibility	Percent of unobstructed and lighted view of facility
Public Surveillance	Location	Number of people passing by site on daily basis
Employee/Management	Culture	Measure of general concern for safety issues
Employee/Management	Training	Type and frequency of training on security matters
Management	Security Administration	Management structure responsible for security oversight
Security Environment	Intelligence	Reliability of prior information of likelihood of an attack

Table 11.1. Description of each of the direct inputs to the MLE asset security model

proposed that avoids these pitfalls by providing a possibilistic framework to threat assessment. Here, MLE modeling is introduced and demonstrated within the context of threat assessment for water distribution systems.

There are a number of advantages to MLE modeling. MLE modeling provides a framework for utilizing both qualitative and quantitative data to score threat scenarios. Further, these calculations are performed within a spreadsheet environment, supported by an interactive, menu-driven user interface, which makes the model operationally manageable for utility staff and management. MLE modeling also provides a consistent, quantitative basis for evaluating threat scenarios. This allows the head-to-head comparison of results and ultimate ranking of threats. Finally, and most importantly, MLE modeling provides a structured framework for integrating information from a wide range of disparate sources. This is important because security cannot be assessed from a handful of disjoint metrics; rather, security must be assessed from a systems perspective. This was demonstrated in the case of the asset security example (Figure 3). Specifically, asset security did not depend directly on detection capability, but required a response to the detected event. But even a response is meaningless unless it occurs prior to the attack reaching its full impact. In addition, the strength of detection and response are strongly influenced by utility personnel attitudes and concern for security. In this way, MLE modeling captures and structures these disparate information sources and aggregates the data to produce a truly integrated threat score.

Scenario:	A	B	C	D	E	F	G	H	I
External Inputs	Input Values								
Alarm Surveillance	0.90	0.90	0.90	0.90	0.90	0.90	0.00	0.90	0.90
/hline Alarm Quality	0.20	0.20	0.20	0.20	0.20	0.20	0.00	0.20	0.90
Alarm Maintenance	0.50	0.40-0.60	0.50	0.50	0.50	0.50	0.00	0.50	0.90
Alarm Sensitivity	0.90	0.90	0.90	0.90	0.90	0.90	0.00	0.90	0.90
Video Surveillance	0.00	0.00	0.90	0.90	0.00	0.00	0.00	0.00	0.00
Video Quality	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00
Video Maintenance	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00
Video Sensitivity	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00
Sensor Location	0.00	0.20-0.40	0.00	0.80	0.00	0.30	0.40	0.90	0.90
Sensor Quality	0.00	0.50	0.00	0.50	0.00	0.50	0.50	0.90	0.90
Sensor Maintenance	0.00	0.50	0.00	0.50	0.00	0.50	0.50	0.90	0.90
Sensor Sensitivity	0.00	0.50	0.00	0.50	0.00	0.50	0.50	0.90	0.90
On Site Security	0.30	0.20-0.40	0.80	0.80	0.30	0.30	0.00	0.30**	0.50
On Site Personnel	0.30	0.30	0.80	0.80	0.30	0.30	0.00	0.30	0.30
Likelihood of Reporting	0.30	0.30	0.30	0.30	0.30	0.30	0.00	0.30	0.30
Visibility	0.30	0.30	0.50	0.50	0.30	0.30	0.00	0.30	0.30
Location	0.30	0.30	0.50	0.50	0.30	0.30	0.00	0.30	0.30
Asset Barriers	0.50	0.40-0.60	0.70	0.70	0.50	0.50	0.00	0.50	0.50
Distance to Asset	0.70	0.70	0.20	0.20	0.50	0.50	0.00	0.70	0.70
Asset Accessibility	0.30	0.30**	0.30	0.30	0.50	0.50	0.00	0.30*	0.30**
Response Time	0.70	0.70	0.70	0.70	0.70	0.70	0.70	0.70	1.00
Detection Time	0.00	0.50-0.70	0.00	0.80	0.00	0.60	0.50	0.60	1.00
Decision Time	0.30	0.30	0.30	0.80	0.30	0.30	0.40	0.30	0.80
Time to Implement Attack	0.20*	0.20	0.20*	0.20	0.20*	0.20	0.20**	0.20	0.20
Time for Attack to Impact	0.00	0.70	0.00	0.70	0.00	0.70	0.50	0.70	0.70
Number of Respondents	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.40
Degree of Training	0.30	0.30	0.30	0.30**	0.30	0.30**	0.30	0.30	0.80
Respondents Capability	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.70
Response Procedures	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.90
Dedicated Response Personnel	0.10**	0.10*	0.10**	0.10*	0.10**	0.10*	0.10*	0.10	0.70*
Employee Culture	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.70
Employee Training	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.90
Management Culture	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.70
Management Training	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.90
Security Administration	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.90
Intelligence	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30

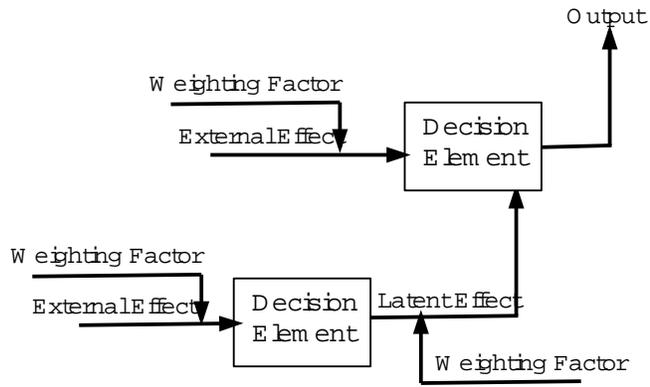
Table 11.2. External attribute values input to the MLE model for seven threat scenarios and two mitigated threat scenarios. Model results are given in Table 11.1

Scenario:	A	B	C	D	E	F	G	H	I
Decision Elements	Score	Score	Score	Score	Score	Score	Score	Score	Score
Detect:	0.28	0.29-0.35	0.57	0.63	0.28	0.32	0.12	0.37	0.50
Remote Monitoring	0.26	0.26-0.27	0.66	0.66	0.26	0.26	0.00	0.26	0.36
Alarms	0.66	0.64-0.68	0.66	0.66	0.66	0.66	0.00	0.66	0.90
Alarm Reliability	0.50	0.47-0.53	0.50	0.50	0.50	0.50	0.00	0.50	0.90
Video	0.00	0.00-0.00	0.66	0.66	0.00	0.00	0.00	0.00	0.00
Video Reliability	0.00	0.00-0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.00
System Monitoring	0.00	0.38-0.46	0.00	0.62	0.00	0.42	0.46	0.90	0.90
Sensor Reliability	0.00	0.50-0.50	0.00	0.50	0.00	0.50	0.50	0.90	0.90
Physical Monitoring	0.30	0.24-0.36	0.80	0.80	0.30	0.30	0.00	0.30	0.42
Public Surveillance	0.30	0.30-0.30	0.42	0.42	0.30	0.30	0.00	0.30	0.30
Delay:	0.24	0.45-0.50	0.25	0.59	0.26	0.50	0.34	0.48	0.68
Physical Security	0.40	0.37-0.43	0.44	0.44	0.48	0.48	0.01	0.41	0.42
Time to Respond	0.27	0.42-0.48	0.27	0.66	0.27	0.45	0.45	0.45	0.85
Critical Attack Time	0.04	0.60-0.60	0.04	0.60	0.04	0.60	0.44	0.60	0.60
Response:	0.27	0.27-0.29	0.37	0.39	0.27	0.28	0.21	0.30	0.64
Response Type	0.28	0.28-0.28	0.28	0.28	0.28	0.28	0.28	0.28	0.68
Likelihood of Response	0.26	0.27-0.30	0.41	0.44	0.26	0.28	0.18	0.31	0.62
Security Environment:	0.36	0.36-0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.70
Employees	0.40	0.40-0.40	0.40	0.40	0.40	0.40	0.40	0.40	0.80
Management	0.36	0.36-0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.80
Asset Security:	0.26	0.36-0.39	0.31	0.49	0.27	0.39	0.28	0.39	0.66

Table 11.3. Aggregate decision element scores calculated for seven threat scenarios and two mitigated threat scenarios. Model inputs are given in Table 11.2

Scenarios: A) Reservoir-Bomb B) Reservoir-Contamination C) Treatment Plant-Bomb D) Treatment Plant-Contamination E) Well Field-Bomb F) Well Field Contamination G) Home Contamination H) Reservoir-Contamination Sensors Improved I) Reservoir-Contamination All Improved

Scenarios: A) Reservoir-Bomb B) Reservoir-Contamination C) Treatment Plant-Bomb D) Treatment Plant-Contamination E) Well Field-Bomb F) Well Field Contamination G) Home Contamination H) Reservoir-Contamination Sensors Improved I) Reservoir-Contamination All Improved



External Input Variable: On Site Security

Elicitation Question: What is the level of physical security monitoring at a site?

Enter any number or range of numbers between 0 and 1 to indicate a qualitative judgment (or range of possible judgments) of the strength of the relationship. 0 represents extremely weak, and 1 represents extremely strong

Attribute Value

0 to 0.3	If no direct monitoring of the site. Site is remote and is not visited by security personnel.
0.3 to 0.5	If limited monitoring occurs for the site. Security personnel visit site at random intervals.
0.5 to 0.7	If there is limited direct monitoring of site. Security checks site regularly during off-hours.
0.7 to 0.9	If security personnel on site 24 hours a day but is only stationed at the entrance of facility, not at site.
0.9 to 1.0	If direct monitoring of the site occurs around the clock by security personnel. Security personnel are on site 24-hours and are constantly patrolling facility.

Figure 11.2. Example elicitation guide for determining attribute values input to the MLE model.

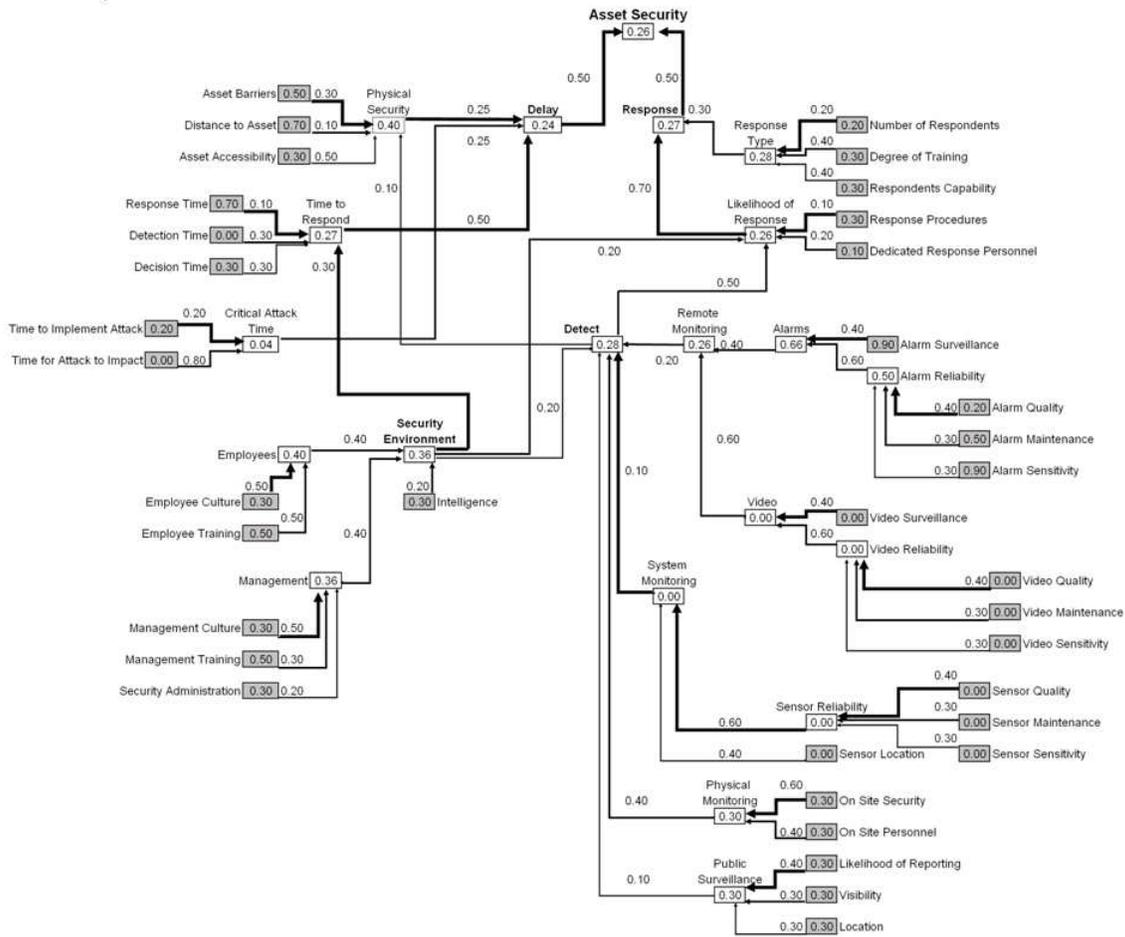


Figure 11.3. Markov Latent Effects (MLE) model for evaluating the security of water utility assets against willful attack. Each open box represents a decision element and each shaded box an external input, while weighting values are the numbers outside the boxes. The connecting lines show the direction in which the data are aggregated. Output from the model is found in the Asset Security element at the top of the figure.

Chapter 12

Vulnerability Assessment and the Basis of Analysis

Vincent C. Tidwell, Consuelo J. Silva, Sariah R. Bujanda

12.1 Background

In efforts to safeguard the nation's water utilities against terrorism and other malevolent threats, the Public Health Security and Bioterrorism Preparedness and Response Act of 2002 (PL 107-188) was enacted to address drinking water security and safety concerns. The law requires almost 8000 community water systems serving more than 3300 people to complete vulnerability assessments and prepare or update their emergency response plans. The required vulnerability assessments are intended to help water utilities evaluate the risk posed by potential threats and identify corrective actions that can reduce or mitigate the consequences of adversarial actions ([197]).

Concepts and methodologies toward the performance of vulnerability or risk assessment are well established (e.g., [4]; [8]; [10], [2]). Furthermore, growing concern over water distribution system security has prompted the development of several vulnerability assessment methodologies tailored directly to the needs of utility managers ([107], [108]; [89] [90]). While development, validation and application of such methodologies are critical, other aspects of the vulnerability process have received little to no attention. Although subtle, these aspects still have the potential for impact. Of concern here are the metrics used as the basis of vulnerability assessment. Specifically, security priorities will differ significantly depending on the vulnerability metrics used (i.e., system damage, number of customers impacted, customer illnesses/deaths) and the extent to which the threat of attack is factored into the analysis.

To explore this dependence a vulnerability assessment is performed for a small water utility. A broad suite of event scenarios is considered involving different combinations of utility assets, assailants, and weapons. The vulnerability for each event is calculated as the product of the threat and the consequence, where four

different consequences are considered. Results are then presented in terms of the relative ranking of the unique event scenarios, subject to the differing bases of analysis (i.e., three different consequences).

12.2 Methods

Vulnerability or risk assessments are generally predicated on the simple relation

$$R = P_a(1 - P_e)C \quad (12.2.1)$$

where R is risk, P_a is the probability of attack, P_e is the probability of system effectiveness, and C is the consequence (e.g., [93]). Operationally, risk assessment is organized around a two component processes, the threat assessment (to calculate the $P_a(1 - P_e)$ term in Equation 12.2.1 and consequence analysis. Below we describe the approach taken to calculating each of these terms.

12.3 Threat Assessment

The purpose of the threat assessment is to define the credibility of a particular threat, which requires quantifying P_a and P_e from Equation 1. Specifically, P_a is a measure of the likelihood of the occurrence of an event, while P_e is a measure of how effective the system is at protecting itself, either through application of security measures or by the physical/operational characteristics of the system (e.g., dilution, disinfection). In most instances insufficient experience and data are available for quantifying the associated probabilities; particularly, in the case of willful attack. For this reason we adopt a possibilistic approach to threat assessment as an alternative to the traditional probabilistic methodologies. Markov Latent Effects (MLE) modeling provides a construct for quantifying imprecise subjective metrics through possibilistic or fuzzy mathematics ([222]).

The MLE model has a performance-focused approach built around the timing of latent effects. A latent effect is an occurrence, condition, or behavior that does not necessarily cause an immediate problem, but can combine later with other occurrences, conditions, or behaviors. These latent effects are identified through decomposition of the complex threat system into more manageable subsystems or decision elements that trace a particular threat from its inception to the point of consequence (Figure 1). In this way, the decomposition helps visualize the network of events that must coincide for a threat to achieve its intended consequence.

Each decision element identified in the decomposition process represents a single factor that influences the likelihood that a threat will yield its intended consequence. Each decision element produces an output subject to a set of inputs. These inputs include external effects and/or latent effects. External effects are inputs unique to that decision element. Each external effect is subjectively assigned an attribute value. Attribute values are qualitative choices, mapped onto a scale of 0 (very weak) to 1 (very strong) that reflect the strength of the relationship between the decision element and the external effect. Latent effects

represent the influence that one element imposes on another. Their attribute values are simply represented by the output value of the latent decision element.

Along with an attribute value each input is assigned a weighting factor. These weighting factors reflect the ability of an input, whether an external or latent effect, to influence the decision element relative to all other attributes contributing to that element. As such the aggregated sum of the weighting factors contributing to an individual decision element must equal one.

Integral to the model is the prescribed data aggregation scheme; that is, the manner by which the observed metrics (i.e., inputs) are combined to yield quantitative information toward the decision of interest. In our case, information defining the credibility of a particular threat. For each decision element, input values are aggregated by a modified weighted sum to provide an assessment score, and this process continues for each decision element until an overall assessment score is derived. There are a variety of weighted sums that can be employed, each having a different influence on the outcome. The choice of the weighed sum depends on one's confidence in the data gathering process and the nature of the decision process.

Figure 1 shows a generic MLE model for conducting threat assessments on municipal water distribution systems. Shown here is only the basic decision structure. Input variables, weighting factors, and subsidiary decision elements have been excluded from this figure to facilitate presentation and discussion. In fact there are over 15 secondary decision elements and more than 75 input variables not shown in this figure. A fuller accounting of some of these variables can be found in Tidwell et al. [222].

The model is organized according to two distinct branches consistent with Equation 1, one pertaining to the possibility of attack and the other to the system effectiveness. The system effectiveness branch is formulated as a balance between asset security and the assailant threat. System effectiveness depends on the three principal elements of response, delay and detection ([13]). Assailant threat is in turn a balance between assailant capability (represented by motivation, qualifications, funding, preparation and access) and the difficulty of attack [91].

The possibility of attack branch is formulated according to a balance between the difficulty of attack and the assailant's perceived consequence. The difficulty of attack is comprised of elements quantifying the complexity of the attack, likelihood of being caught, accessibility of target to the assailant and weapon complexity ([50]). Assailant's perceived consequence measures the value of the target in the eyes of the assailant through such metrics as disruption, health impact, economic impact and level of attention ([91]).

With the resulting MLE model, assessments are performed for specific threats by simply inputting attribute values specific to that threat. Attribute values are assigned primarily through expert elicitation employing elicitation guides that standardize the scoring. The attribute values are then aggregated over all decision elements to provide the threat assessment score, yielding a measure of the credibility of the threat. By consistently following the same analysis scheme, assessment scores calculated for different threats can be compared and ranked to identify the most credible threat.

12.4 Consequence Analysis

An attack on a water system could lead to various consequences, examples include death or illness, lost service time, repair/disinfection costs, and/or loss of confidence in the utility. To evaluate such consequences one must consider the hydraulic and chemical transport processes that govern the transmission of water from the point of the attack to the tap where the consequence is felt. To model these transmission processes in the context of a water distribution system, we use EPANET.

EPANET is a public domain software package that was developed by the U.S. Environmental Protection Agency's National Risk Management Research Laboratory and has become the standard for water utility managers across the country. EPANET performs simulations of hydraulic and water-quality behavior within pressurized pipe networks. EPANET tracks the flow of water in each pipe, pressure at each node, height of water in each tank, and concentration of a chemical species through the network during a simulation period of multiple time steps. Additional detail on EPANET can be found in [1].

For purposes of this chapter, focus is placed on intentional contamination of a water distribution network. Consequences are scored according to four different metrics, total exposed population, the population exposed above the LD50 (i.e., dose at which 50% exposed population would die) at 24 and 240 hours, and the length of pipe contaminated. Simulations were performed for 240 hours following the contaminant injection using a hydraulic time step of 30 minutes and contaminant transport time step of 1 minute. Contaminants introduced into the system through fresh water storage tanks followed a slug injection, while contaminants introduced at other assets (e.g., buildings, transmission pipes, well fields) occurred as a step injection over 3-10 hour period of time (depending on the mass injected). In the examples presented the injected contaminant mass is assumed to be 106 times the LD50.

Simulations were performed using calibrated pipe networks provided by the utility along with the average daily demands at each node in the system. The population at each node was estimated from the nodal demands (assuming a constant per capita consumption across the utility). Additionally, each receptor was assumed to ingest two liters of water a day, which was evenly distributed across the hours of 6 am to 11 pm.

12.5 Results

Using the risk assessment framework described above we perform a vulnerability assessment for a small water utility. The selected utility is supplied by a well field that feeds two freshwater storage tanks. In turn, the storage tanks feed a water distribution system that services a number of commercial and light industrial buildings. Approximately 10,000 people work in the buildings supplied by the water system. The water distribution network utilized in the EPANET modeling includes 330 nodes representing the various tanks, pipe junctions, building/fire hydrant connections, and water mains comprising the system.

To date, a relatively complete vulnerability assessment has been performed for the water utility. Consequence analyses have been performed for most of the 330 nodes assuming a wide range of attack modes (e.g., various contaminants, service disruption, vandalism). For each of the different asset/attack couples, threat scores have been calculated with the aid of the MLE model for differing assailants (i.e.,

terrorist, insider, vandal). When combined this yields thousands of different event scenarios, which for all practical purposes precludes us from presenting all the data here.

Below, we present results for a limited number of event scenarios. The presented event scenarios are selected so as to provide a representative sub-sample of the thousands of event scenarios considered in the vulnerability assessment. Specifically, nine different event scenarios are presented. These event scenarios involve all possible combinations between three utility assets (a treated water storage tank, a junction between two high capacity water lines, and a commercial building connection) and three assailants (a terrorist, a disgruntled utility employee or insider, and a vandal). Additionally, four different bases of analyses (i.e., consequences) are used to rank the different event scenarios.

12.6 Threat Assessment

To improve confidence in the MLE model, validation exercises were performed. Comparison were drawn between MLE model output and historical data tabulating actual attacks and planned attacks against municipal water utilities in the United States. This data set was compiled by the American Water Works Association Research Foundation (2003). From this data set of several hundred actual/planned attacks, 20 unique event scenarios were identified as defined by the assailant, weapon and asset. A relative ranking of event scenarios was then calculated based on the number of actual/planned attacks associated with a given event scenario relative to all scenarios. These relative percentages were then compared to the threat score calculated with the MLE model. Due to lack of detailed information, poor security measures were assumed for purposes of the MLE modeling. Results of the comparison suggest a good fit between historical trends and threat scores produced with the MLE. Unfortunately, the proprietary nature of the historical data precludes a fuller disclosure of model comparison results.

Once confidence was gained in the MLE model a full threat analysis for the water utility was performed. Modeling of each event scenario required scoring the 75 attribute values forming the input to the MLE model. Scoring was accomplished with the aid of the elicitation guides. Data specific to each attribute value were taken from existing vulnerability assessment documentation and compared to the criteria given in the elicitation guide to determine an appropriate score for that attribute value. In this way, each attribute value was assessed according to a unique set of criteria as defined in its elicitation guide.

Threat scores for the sub-sample of nine representative event scenarios are given in Table 1. Scores range from 0.05 to 0.77. To help interpret these results, consider that a score of zero implies that there is no chance of the event occurring given the current state of the system; alternatively, a value of one suggests that the attack will occur without question. Within this context, terrorist lead events scored low, while insider led events tend to score moderate to high.

The threat scenario with the highest score (i.e., highest threat) involved an intentional contamination attack on the fresh water storage tank by an insider. This threat scored highest because it has a high perceived consequence (i.e., services the entire network) while the asset is relatively accessible and poorly secured. The lowest threat score was associated with an attack by a terrorist on the commercial building connection. The building connection always scored lowest among the three assets because of the relative difficulty of staging such an attack and the low consequence (when compared to the tank and pipe junction). The

insider scored consistently higher than the terrorist because of the ease of utility access afforded the insider and the low attention level of the utility (there is little reason for this utility to attract the attention of a terrorist). The vandal generally scored low because of low capability and motivation relative to accomplishing a contamination attack.

12.7 Consequence Analysis

EPANET simulations were performed for most of the 330 network nodes. Simulations involved injection of a contaminant at a node while tracking the number of people exposed, those exposed above the LD50, and the length of contaminated pipe over time (240 hours). As an example of our modeling results, Figure 2 presents the number of people exposed above the LD50 threshold for two system tanks and three commercial building connections. The cumulative exposures for all five assets show similar trends; that is, each begins with a short time delay before any effects are felt, followed by a rapid rise in impact that asymptotically approaches some equilibrium point. Other system assets show similar behavior as does the trends in the length of contaminated pipe over time. Although all follow the same basic trends the time delays, equilibrium exposure levels and time to equilibrium all differ. Variations in the time delay reflect differences in the distance between the injection node and distance to the first receptor. Differences in the equilibrium exposure and time to equilibrium reflect the complex interplay between the hydrodynamics in the pipe network, proximity of receptors to the point of injection, and dilution. From Figure 2 it is apparent that considerable differences in exposure and time to exposure are realized among system assets.

Results for the sub-sampled event scenarios are given in Table 2. Specifically, results are recorded for three network assets (a fresh water tank, pipe junction, and a commercial building connection) with each tabulated according to the four different metrics. Review of this data clearly indicates that the relative severity of the consequence depends strongly on the basis of analysis. Where total exposure at 240 hours is considered the attack on the fresh water tank has the most severe consequence (6335 people) while attacks on the building connection and junction have a little less than half the tank's impact (2848 and 2232 people, respectively). For exposure above the LD50, the tank again carries the most significant consequences effecting 6335 people, however the pipe junction exposes 1335 people and the building connection only 299 people. If the utility had a very good early detection system, perhaps only 24 hours would be needed to detect and mitigate any contaminant intrusion. In such case no exposures would be felt from the fresh water tank while attacks on the pipe junction and building connection would impact 108 and 278 people, respectively. Finally, if contaminated pipe was the chosen basis of analysis then the fresh water tank would dominate the consequences with 139,614 feet of pipe contaminated compared to 34,164 feet for the pipe junction and 241 feet for the building connection.

12.8 Risk Assessment

Risk assessment scores are determined by combining the threat scores and calculated consequences according to Equation 1 (Table 3). Specifically, the threat scores tabulated for each of the 9 event scenarios (Table 1) are multiplied by the corresponding consequence (Table 2). For example, the consequence value

calculated for an attack on the fresh water tank is multiplied by each of the threat scores associated with the terrorist attack on the tank, insider attack on the tank, and vandal attack on the tank. Here we assume that the same consequence will be realized regardless of the assailant. Because four different bases of analysis are employed, risk scores are determined for each of the four different consequences calculated. Finally, the risk scores are ranked for each of the four bases of analysis.

Review of the data indicates that both the risk and relative rankings of the 9 event scenarios depend on the basis of analysis. That is, the risk score and rankings differ depending on whether total exposure, exposure above LD50 at 24 or 240 hour, or length of contaminated pipe are used as the basis of analysis. These differences can be considerable in that the highest ranked risk for one adopted consequence can be the lowest ranked risk for another consequence. This disparity largely reflects the differences in the calculated consequences according to the different bases of analysis. The threat scores are also seen to have a significant influence on the overall risk. This is apparent in the terrorist attacks, which rank low in response to the low threat scores. In contrast insider perpetrated attacks scored high in agreement with the higher threat scores. Finally, attacks on building connections tended to score poorly relative to other assets.

From this limited analysis the fresh water tank is clearly the most important asset in the system to protect as it generally ranked number one and two. The fact that these attacks are most likely to be perpetrated by an insider or vandal provides insight into the types of security measures that might be taken. Simple physical security improvements along with improved administrative controls would be preferred over asset hardening.

12.9 Conclusions

Integral to the vulnerability assessment process is the basis of analysis. That is, resulting security priorities differ significantly depending on the vulnerability metrics used (i.e., system damage, number of customers impacted, customer illnesses/deaths) and the extent to which the threat of attack is factored into the analysis.

To explore this dependence a vulnerability assessment was performed for a small water utility. A sub-set of 9 event scenarios were considered in the analysis involving all possible combinations between three utility assets (fresh water tank, pipe junction, and building node), and three assailants (terrorist, insider, and vandal). The vulnerability for each event was calculated as the product of the threat assessment score and the consequence. Threat scores were calculated using Markov Latent Effects modeling, while attack consequences were determined using EPANET. Four different consequences were considered, total number of customers exposed, number of customers exposed above the contaminant's LD50 at 24 and 240 hours, and the length of pipe contaminated.

Results were presented in terms of the relative ranking of the 9 different event scenarios, which in turn represent security priorities for the water utility. Depending on the basis of analysis (i.e., four calculated consequences) used and whether the threat score was factored into the analysis, significantly different rankings were realized. These results clearly articulate the need for utility managers to carefully consider and broadly vet decisions concerning the basis metrics used in vulnerability assessments.

	Terrorist	Insider	Vandal
Tank	0.05	0.61	0.38
Building	0.04	0.58	0.12
Junction	0.05	0.77	0.15

Table 12.1. Threat scores, as calculated with the MLE model, for the 9 sub-sampled event scenarios.

	Total Exposure at 240 hrs (People)	Exposure above LD50 at 240 hrs (People)	Exposure above LD50 at 24 hrs (People)	Total Pipe Contaminated (Feet)
Tank	6335	6335	0	139614
Building	2848	299	278	241
Junction	2232	1335	108	34164

Table 12.2. Numerical consequence values calculated for the 9 sub-sampled event scenarios subject to the four different bases of analysis. Note that consequence values are assumed to be independent of the assailant.

Assailant	Event Scenarios	Total Exposure at 240 hrs		Exposure above LD50 at 240 hrs		Exposure above LD50 at 24 hrs		Total Pipe Contaminated (Feet)	
		Risk	Rank	Risk	Rank	Risk	Rank	Risk	Rank
Terrorist	Asset	317	7	317	4	0	7	6981	4
	Tank	142	8	15	9	14	5	12	9
	Building	89	9	53	7	4	6	1367	6
Insider	Junction	3864	1	3864	1	0	7	85165	1
	Tank	1652	4	174	6	161	1	140	7
	Building	1719	3	1028	3	83	2	26306	3
Vandal	Junction	2407	2	2407	2	0	7	53053	2
	Tank	342	5	36	8	33	3	29	8
	Building	335	6	200	5	16	4	5125	5

Table 12.3. Risk assessment scores and accompanying rankings for the 9 sub-sampled event scenarios subject to the four different bases of analysis. Note: Exposure refers to number or people

Chapter 13

A Comparison of Navier Stokes and Network Models To Predict Chemical Transport In Municipal Water Distribution Systems

B.G. van Bloemen Waanders, G.E. Hammond, J.N. Shadid, S.S.Collis, R. E. Murray (EPA)

13.1 Background

In this chapter, we investigate the accuracy of chemical transport within water distribution system network models by applying high-fidelity computations to model individual network components. Network models have been used for years to simulate chemical transport within water distribution systems in order to better manage chlorine distribution and water quality. Tools such as EPANET have been developed to efficiently predict hydraulic flow and chemical transport behavior within a system. Due to the extreme size and detail of real-world datasets, simplifying assumptions must be applied to fluid flow and chemical transport in order to reduce computational demands. However, these simplifications often reduce the accuracy of a network model since the impact of small-scale phenomenon is neglected. We believe that the resolution of such small-scale phenomenon is necessary in order to more accurately characterize and remediate a water distribution system during a contamination event.

This work is motivated by the need to utilize novel mathematical algorithms within numerical simulation to support water distribution system management during a contamination event. Our goal is to improve the mitigation performance by leveraging high-fidelity modeling to more accurately characterize the transport of contaminants during an event. In response to this need, Laird et al [136] have developed inversion algorithms to reconstruct a contamination event assuming that measurements of contaminant concentrations are available at sparsely located sensors, and Berry et al. [35] have developed combinatorial

methods to provide optimal sensor placement strategies. Although these algorithms have been numerically verified, their utility is limited by the accuracy of the numerical models used to predict or characterize chemical transport.

High-fidelity computational fluid dynamics (CFD) tools have advanced significantly, enabling the characterization of complex phenomena ranging from chemical reactions in chemical vapor depositions [196] to highly turbulent flows using Direct Numerical Simulation (DNS) and Large Eddy Simulation (LES) [156, 143, 166]. Combined with large computational resources, these CFD tools can more accurately characterize fluid flow and chemical mixing in complex geometries. To date, very little work has been performed to characterize the complex, fine-scale chemical transport behavior within the turbulent flows of a water distribution system. For practical purposes, modeling such detail is unnecessary, and perhaps more importantly, too expensive for entire networks. However, the state-of-the-art network model is only as good as the accuracy of its underlying assumptions for the smallest, fundamental physics within these networks. By carefully evaluating the fluid flow behavior at this fundamental level, we hope to extract certain corrections to help improve the accuracy of chemical transport in network models. During the initial phase of this investigation, we investigate the cross-joint intersection of four pipes (two inflowing, two outflowing). The standard network simulator assumes an even distribution of chemical within the joint based on an instantaneous mixing assumption. Although intuitively we can predict less than perfect mixing under prescribed inlet conditions, it is difficult to quantify the behavior with sufficient accuracy to develop and apply first-order corrections within a network model. We attempt to characterize this mixing behavior assuming no chemical reaction in the system, although chemical reactions can be handled in our current formulation and may be included in the future.

The remainder of the chapter presents the mathematical formulation for the high fidelity computational fluid dynamics. Numerical and experimental results are presented.

13.2 Mathematical Formulation

The final goal of this investigation is to develop a first-order correction for mixing phenomena in network models by more accurately characterizing the chemical transport behavior within important pipe geometries through the use of 3D turbulent Navier-Stokes simulation. To accomplish this goal, we first test smaller components (i.e. pipe joints) in 2D under the assumption of laminar flow or using a Reynolds Averaged Navier-Stokes (RANS) model without resolving the small-scale, unsteady fluctuations of a 3D fully turbulent flow. Eventually, we will build up to a high-fidelity characterization consisting of fully-developed 3D turbulence. The underlying physics for these studies is based on the Navier-Stokes equations and convection-diffusion equations. Here, we present equations incorporating conservation of mass and momentum. The conservation of momentum can be formulated as follows:

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) - \nabla T - \rho \mathbf{g} = 0 \quad (13.2.1)$$

where ρ is the density, \mathbf{u} is the velocity vector, t is time, $T = -PI + \mu$ is the stress tensor for a Newtonian fluid, P is the hydrodynamic pressure, μ is the viscous stress tensor, and \mathbf{g} is the gravitational force.

Conservation of mass is as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (13.2.2)$$

Chemical transport is formulated as a separate convection-diffusion equation:

$$\frac{\partial C}{\partial t} + \nabla \cdot (C\mathbf{u}) - \nabla \cdot (D_{\text{eff}}\nabla C) = 0. \quad (13.2.3)$$

where C represents concentration, D_{eff} is the effective diffusion coefficient, and \mathbf{u} is the velocity vector.

13.3 Numerical Implementation

The incompressible Navier Stokes equations are solved using MPSalsa [205, 204]. Our implementation of incompressible Navier Stokes is designed to handle large datasets on massively parallel computers in addition to an efficient implementation for complex dynamics in three dimensions. The code uses a Petrov Galerkin finite element formulation for unstructured meshes and is capable of solving steady state and transient problems using a fully implicit time integration. The nonlinear systems are solved using inexact Newton methods which in turn depend on Krylov based linear solvers. This implementation (MPSalsa) is capable of resolving turbulence using a variety of Reynolds averaged Navier Stokes (RANS) and Large Eddy simulation (LES) based methods and can handle chemical reactions [120].

Using DNS or LES to more accurately evaluate mixing phenomenon within a water distribution network is computationally demanding, even when modeling is focused on small components of the network. To resolve turbulent flow, a high-resolution 3D mesh is necessary to capture the small-scale effects of turbulence, which results in extensive computational processing and memory use. To alleviate these computational demands and establish insight during the initial stages of this investigation, we limited flow and transport to 2D recognizing

that turbulence would not be captured properly. For the purposes of the 2D investigation, we employed an 1152 element mesh on 8 to 16 computational processors to simulate the mixing phenomenon within a 2 inch cross joint (see Figure 13.1).

The first step in modeling cross-joint mixing was to develop 2D flow inside a hypothetical pipe joint within which chemical tracer could be added to assess the amount of mixing at the joint. This flow field was developed by incorporating oscillatory (sinusoidal) boundary conditions at the inlets shown in Figure 13.1, thereby emulating turbulent-like flow. Inlet bulk velocities were prescribed at 0.78 meters per second, resulting in an average pipe Reynolds number of 44,000 based on the characteristics of water at 25° C (i.e. viscosity of 8.9×10^{-4} kg/m-s, density of 997.0 kg/m³) and a 2 inch (.0508 meter) pipe diameter. Figure 13.2 illustrates the velocity field 10 seconds into the 20 second simulation, at which point 32.5 pipe volumes of water have passed through the joint.

Although the Reynolds number is well in excess of 2000 suggesting turbulent flow, the 2D model is not capable of fully resolving turbulence, as shown in Figure 13.2. MPSalsa accounts for subgrid scale turbulent kinetic energy enabling it to compute a turbulent viscosity term μ_t (see Figure 13.3), and therefore is able to predict the general fluid flow characteristics for this geometry. By definition, this turbulent viscosity represents the turbulent transfer of momentum by eddies. In the case of chemical transport, the term can also be thought of as a coefficient of enhanced diffusion due to mechanical mixing. MPSalsa

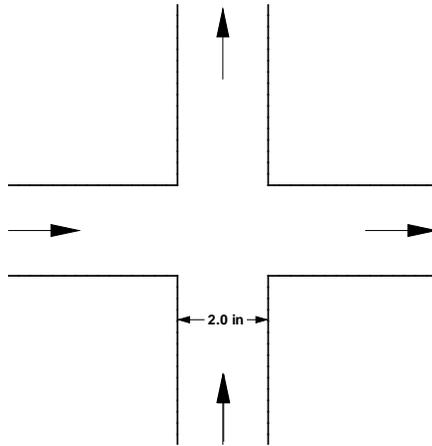


Figure 13.1. Schematic of pipe cross-type joint.

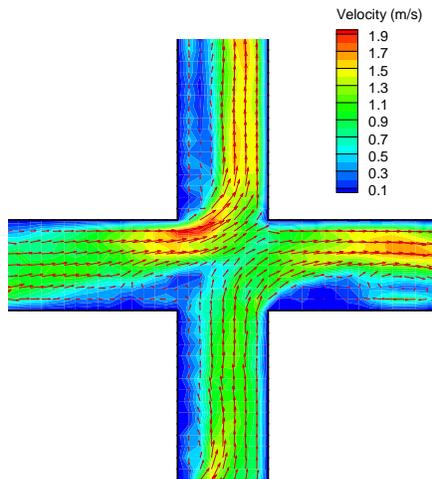


Figure 13.2. Velocity field within pipe cross-type joint at 10 seconds simulation time.

utilizes the turbulent viscosity to calculate an effective diffusivity D_{eff}

$$D_{\text{eff}} = D_{\text{molecular}} + \frac{\mu_t}{Sc}, \quad (13.3.4)$$

where $D_{\text{molecular}}$ is the coefficient of molecular diffusion and Sc represents the dimensionless Schmidt number. Therefore, in regions of high turbulent viscosity, one can expect mechanical mixing to dominate the diffusion process.

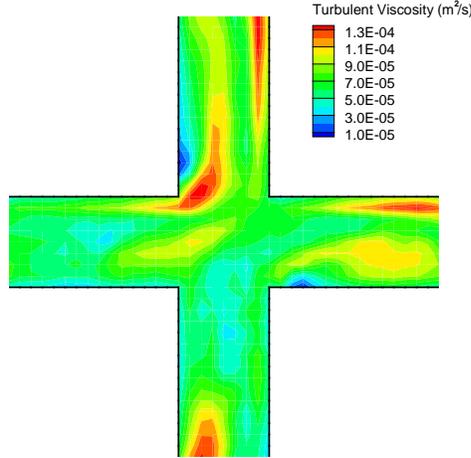


Figure 13.3. Turbulent viscosity within pipe cross-type joint at 10 seconds simulation time.

Coupling the velocity field (\mathbf{u}) and effective diffusivity above, MPSalsa uses the convection-diffusion equation (13.2.3) to simulate chemical transport.

13.4 2D Numerical Results

For this work, cross-joint mixing was simulated by introducing a tracer at the bottom inlet boundary condition while pristine water entered the model from the left inlet (see Figure 13.1). The degree to which mixing occurred at the cross joint was measured by comparing the outlet flux of tracer at the top and right exit boundaries. Although a direct comparison of the two outlet fluxes is unrealistic (i.e. discrepancies in local pipe velocities most likely result in asynchronous breakthrough), a general trend is observed. Figure 13.4 illustrates the normalized outlet flux concentrations, averaged across the width of the pipe, over the 20 second simulation. It is evident that most of the tracer, entering the joint from the bottom inlet, exits to the right following the flow velocity vectors shown in Figure 13.2. Mixing, defined here as the change in concentration relative to the inlet tracer concentration

$$\frac{|(C_{\text{in,left}} - C_{\text{out,top}})|}{C_{\text{in,bottom}}} \quad \text{and} \quad \frac{|(C_{\text{in,bottom}} - C_{\text{out,right}})|}{C_{\text{in,bottom}}}, \quad (13.4.5)$$

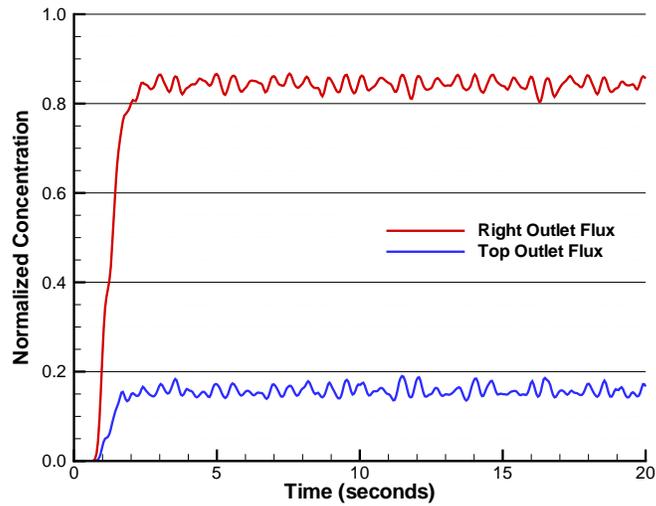


Figure 13.4. Normalized simulated tracer breakthrough at outlets.

is approximately 13-18% and is clearly not complete as would be the case if both outlet fluxes plotted at 50%. Figure 13.5 illustrates snapshots of tracer concentration at 9, 10, and 11 seconds simulation time.

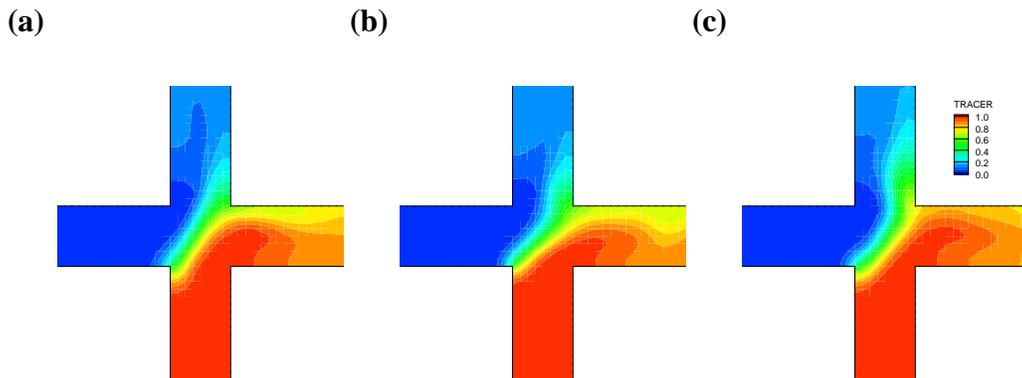


Figure 13.5. Tracer concentration within pipe cross-type joint at 9, 10 and 11 seconds simulation time, respectively.

Although flow exceeds a Reynolds number of 2000, the bulk flow clearly interacts in a more laminar manner within the cross joint.

13.5 Experimental Verification

In order to verify the results generated by MPSalsa, a physical model of the pipe cross joint above was assembled and experiments were run at approximately the same Reynolds number (i.e. 44,111). This physical model consisted of four 50+ gallon tanks interconnected by 2 inch PVC pipe intersecting at a PVC cross joint. Two Teel centrifugal pumps of identical size were placed up-gradient from the cross joint as illustrated in Figure 13.6. Two tanks, one containing tap water and another containing a NaCl tracer, served

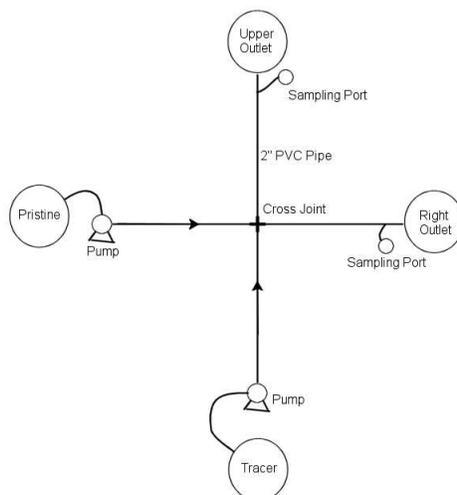


Figure 13.6. Schematic of experimental setup.

a sources for the two pumps which subsequently pumped the water through the joint and into the two outlet tanks. We assumed that the tracer was fully mixed within the bottom inlet tank and turbulence within the up-gradient pipes was fully developed before water passed through the joint. We also assumed that tracer concentration was uniform across the entire pipe cross-section by the time it reached the sampling ports 6 feet down-gradient from the cross joint, which based on mass balance calculations appears to be valid.

The experiment ran for 100 seconds, while effluent water was sampled at 10, 25, 40, 55 and 85 seconds. The conductivity of effluent water extracted from the sample ports was compared to that of the tap water (background) and the NaCl mixture (tracer) in the two inlet tanks, and normalized concentrations were calculated. Figure 13.7 shows the normalized tracer breakthrough at both outlets over time. Based on Equation 13.4.5, 12-14% mixing was observed at both outlets. Here, we see that the numerical simulation is close to approximating the experimental results, and since simulated flow within the pipe is more laminar than turbulent, it suggests that turbulence within the numerical model will not increase the mixing in a cross-section significantly.

To develop a first order approximation, numerical simulation can provide an appropriate range of predictions to characterize the behavior of chemicals in certain pipe geometries. It is critical therefore to

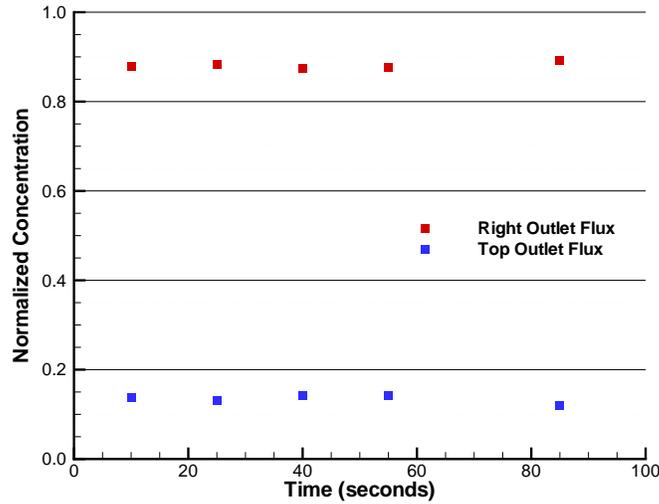


Figure 13.7. Normalized experimental tracer breakthrough at outlets.

continue to pursue the three dimensional model and try to establish fully resolved turbulence.

13.6 Ongoing Research

13.6.1 3D Turbulence Simulation

In a closely coordinated but parallel effort, another fluid flow code was utilized to investigate turbulence in 3D. The reasons for selecting another code to investigate turbulence was to 1) efficiently leverage human resources (code familiarity and experience), 2) provide for a verification mechanism (other than limited experimentation), and 3) make use of other capabilities, such as discontinuous Galerkin (DG) methods with the associated advantages of high-order accuracy, local hp -refinement, and multiscale modeling. This code, SAGE, uses a new approach to large eddy simulation (LES) that is based on a local variational multiscale (ℓ VMS) method [72, 73, 75] which is an extension of the VMS method of Hughes [116, 71] that is extended to support local multiscale modeling on an element-by-element basis using DG. SAGE has been validated on a variety of laminar [74] flows and the ℓ VMS approach within SAGE has been carefully studied for planar turbulent channel flows [75, 173]. In these studies, ℓ VMS is demonstrated not only to be more accurate than the dynamic model for LES but also simpler to implement, especially on unstructured meshes for complex geometries.

The current work extends our use of ℓ VMS to the more complicated cross-section geometry. To date, we

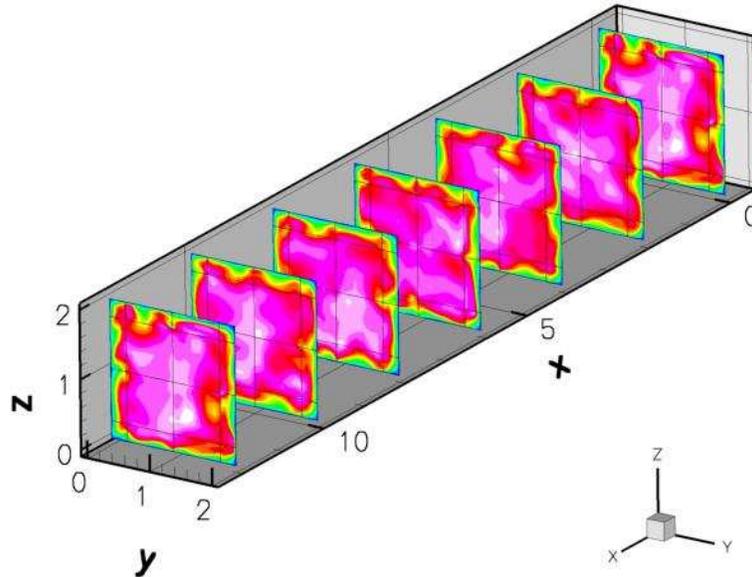


Figure 13.8. Fully developed turbulence flow in a duct geometry.

have established fully turbulent flow in an inlet section of square duct. Figure 13.8 shows contours of streamwise momentum on several slices through the computational domain. The results of this single-duct simulation will be used to specify fully-turbulent inflow conditions for the cross-section model.

13.6.2 First-Order Correction Strategies

Based on the 2D numerical results and laboratory verifications, a first order correction in network models needs to be developed to address chemical transport as a function of geometry, fluid flow dynamics and chemical characterization. Specifically, some obvious parameters can be identified that most likely will provide the foundation for the development of the first order correction scheme, such as 1) Reynolds number, 2) retention time as function of pipe geometry, 3) chemical concentration. To develop a correction strategy, significant numerical studies need to be conducted to quantify the amount of mixing for a range of geometries.

13.7 Conclusions

High fidelity numerical and laboratory results indicate that chemical transport behavior is not accurately simulated in the state of the art network models. These network models assume instantaneous mixing and although these network simulators have been successfully used for general operational management, it is not clear that chemical transport can be accurately predicted for contamination events. Contrary to the network model predictions of 50% in each outlet, our numerical models and laboratory results show 12-14%. Even though the 2D implementation is not capable of fully resolving turbulence, the numerical model still provides a reasonable match to the laboratory tests, which were conducted in the turbulent regime. This suggests that the small volume at the intersection is not sufficient to allow for complete mixing and that fully developed turbulence in our numerical model may not provide much additional mixing. A three dimensional model has been implemented and we show fully developed turbulence in a simple duct geometry.

Chapter 14

Performance of Fully-Coupled Algebraic Multilevel Domain Decomposition Preconditioners for Incompressible Flow and Transport

Paul T. Lin, Marzio Sala, John N. Shadid, Ray S. Tuminaro

14.1 Introduction

Computational fluid dynamic simulations often require the solution of strongly-coupled interacting physics on high resolution unstructured meshes. The discretization and linearization of the equations produce large linear systems of equations, for which robust and efficient parallel iterative solution methods are necessary. Preconditioned Krylov iterative methods are among the most robust and fastest iterative solvers over a wide variety of CFD applications [104]. The key factor influencing the robustness and efficiency of these solution methods is the choice of preconditioners. Multilevel Schwarz domain decomposition based preconditioners [215] are considered in this study. The additive Schwarz approach partitions the original domain into overlapping subdomains and approximately solves the discrete problem corresponding to the individual subdomains in parallel. A known disadvantage is that this method (referred to as a one-level Schwarz preconditioner) does not scale as the number of subdomains increases. That is, the number of iterations rises as the number of subdomains increases. To remedy this situation, multigrid ideas can be employed. The general multigrid philosophy is to use a sequence of meshes to damp errors at different frequencies thereby accelerating the convergence to the solution on the finest mesh. In a typical multigrid method, many coarse meshes are used in conjunction with fairly lightweight smoothers. The coarsening rate to define the next mesh is normally fairly modest (e.g. the next coarser mesh contains one tenth as many grid points as the previous finer mesh). In a domain decomposition setting, only one coarse mesh is usually employed and this mesh is significantly coarser than the finest mesh. The one-level Schwarz

preconditioner can be viewed as a heavyweight somewhat expensive smoother from a multigrid perspective. This expensive smoother is natural when the coarse grid is much coarser than the fine mesh.

Multigrid methods come in two varieties. Geometric multigrid methods use grid coordinate type information (and perhaps finite element information) to define grid transfers. While these methods can work well, they typically require users to define a sequence of meshes along with operators to transfer solutions between them. The other variety of multigrid fall into the category of algebraic methods. In this case, the coarse meshes as well as the grid transfers are defined using only information from the fine grid linear system. While the generation of these operators is reasonably understood for Poisson-like operators [223], optimal grid transfers for highly convective flows and chemical reactions is still not well-understood in the context of algebraic methods.

In this chapter we present a scalable multilevel preconditioner based on aggregation [55, 234]. This work is an extension of previous work [209] where a two-level preconditioner with geometric coarse operator was compared with the one-level counterpart. Here, we extend this analysis by comparing several multilevel domain decomposition preconditioners for accelerating the convergence of a parallel Newton-Krylov solution method. Aggregation has been used to define two-level domain decomposition preconditioners by several authors [119, 140, 193]. One important aspect is to evaluate whether a two-level preconditioner with an algebraic coarse operator could perform (in terms of iterations and parallel scalability) as well as a two-level preconditioner with a geometric coarse operator. In [191], some results are reported that compare two-level preconditioners with geometric coarse grids with smoothed and nonsmoothed aggregation techniques for a Laplacian problem, showing that both methods share comparable behavior for symmetric coercive problems. Here, we numerically compare the two approaches for nonsymmetric problems, namely the incompressible Navier-Stokes equations with heat transfer and convection-diffusion equation with a given velocity field. We show that, even for problems defined on simple geometries, the algebraic preconditioner is not worse than the geometric one. To the best of our knowledge, these results are the first presented in the literature that compare two-level preconditioners using algebraic coarse operators with those using geometric coarse operators on massively parallel computers. We also present a scalable three-level algebraic method, and we show that we can drastically reduce the CPU-time to solve fluid flow problems by using three-level aggregation-based preconditioners.

14.2 Governing Equations

The governing PDEs describing fluid flow, thermal energy transfer, and mass transfer for variable density low-speed flow are as follows.

Total Mass Conservation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

Momentum Transport:

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho (\mathbf{u} \cdot \nabla \mathbf{u}) - \nabla \cdot \mathbf{T} - \rho \mathbf{g} = 0$$

Energy Transport:

$$\rho C_p \left[\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T \right] - \nabla \cdot (\lambda \nabla T) = 0$$

Species Transport:

$$\rho \left[\frac{\partial Y_k}{\partial t} + \mathbf{u} \cdot \nabla Y_k \right] - \nabla \cdot (\rho D_k \nabla Y_k) = 0 \quad k = 1, 2, \dots, N_s - 1$$

Constitutive equation for Newtonian stress tensor:

$$\mathbf{T} = -P\mathbf{I} - \frac{2}{3}\mu(\nabla \cdot \mathbf{u})\mathbf{I} + \mu[\nabla \mathbf{u} + \nabla \mathbf{u}^T]$$

In these equations the unknowns are the velocity vector \mathbf{u} , the temperature T , the hydrodynamic pressure P , and the N_s species mass fractions Y_k . The transport properties, $\rho, \mu, C_p, \lambda, D_k$, are respectively, the density, dynamic viscosity, specific heat capacity, heat conductivity, and diffusion coefficient. These equations are approximated by a stabilized finite element formulation [117, 221, 207] that allows for equal order interpolation of pressure and velocity (without spurious pressure solutions) and for stabilization of highly convected flows.

The computer code used in this study is MPSalsa [207], a parallel finite element method on unstructured meshes that solves incompressible reacting flow problems.

14.3 Preconditioned Newton-Krylov Method

The discretized equations are solved using a Newton-Krylov method [59, 206], which is an implementation of Newton's method in which a Krylov accelerator is used to solve the linear systems that are generated at each step of Newton's method. A Newton-Krylov method is usually implemented as an inexact Newton method [88], meaning that the linear systems are solved only approximately.

For the considered class of problems, convergence cannot be achieved without preconditioning [189]. One widely used preconditioner, well-suited for parallel computations, is the one-level Schwarz preconditioner [215, 172]. The procedure is as follows: first, we decompose the computational domain Ω into M overlapping subdomains Ω_i , and we assign each subdomain to a different processor. Then, the preconditioner is applied by solving, usually with a direct method, a Dirichlet problem on each subdomain Ω_i (with homogeneous boundary conditions on $\partial\Omega_i$). Using a minimal overlap, the resulting preconditioner can be seen as a block Jacobi preconditioner, where each block contains all the nodes assigned to a given subdomain. Better performance can be obtained by increasing the overlap between the subdomains.

Because of the large cost of direct factorization techniques, an approximate factorization technique is employed for the solution of the local Dirichlet problems. For this work, ILUT [188] is used, typically with a drop tolerance of zero and keeping the number of nonzeros in the ILUT factors approximately equal to the number of nonzeros in the original discretization matrix. The Aztec library [118] is used to implement

the Krylov accelerator and the one-level Schwarz preconditioner. We note that the one-level preconditioner is completely black-box, since Aztec automatically constructs the overlapping submatrices. However, a drawback of this preconditioner is that as the number of subdomains increases, the convergence rate deteriorates due to the lack of global coupling in the preconditioner [215]. To remedy this situation, coarse operators can be added to approximate the global coupling in the linear system [224, 209], as presented in the following section.

14.4 Multilevel Preconditioners

Our implementation of the presented multilevel preconditioners is based on the ML multilevel preconditioning package [224, 194]. For the two-level Schwarz preconditioner with a geometric coarse operator, ML expects the user to supply both the fine and coarse meshes as well as finite element basis functions on the coarse mesh. Using this information, ML constructs the interpolation or prolongation operator that corresponds to the coarse mesh basis functions. ML handles all of the bookkeeping (such as determining the coarse mesh element that contains each fine grid point) and uses callback functions to the application providing a data-structure neutral interface. The restriction operator is calculated as the transpose of the projection operator. The coarse matrix \mathbf{A}_c is defined by the Galerkin projection $\mathbf{A}_c = \mathbf{R}\mathbf{A}_f\mathbf{P}$, where $\mathbf{R} = \mathbf{P}^T$ and \mathbf{A}_f is the fine mesh matrix.

The use of a coarse mesh to accelerate the convergence of a one-level Schwarz preconditioner is similar to multigrid methods that use a sequence of coarser meshes to accelerate the convergence of the solution on a fine mesh. Typically, more than two levels are employed in a multigrid approach. One of the disadvantages of a geometric multigrid approach is the need to generate a sequence of geometrical grids. Alternatively, one can use algebraic multilevel preconditioners, which do not require a sequence of coarser grids. First, a graph is built from the linear system. This graph contains an edge between two vertices i and j , if $A_{i,j} \neq 0$. For systems of PDEs, it is often natural to define this graph in a block fashion. That is, one vertex is associated with each block of unknowns (e.g. velocities and pressure at a particular grid point) and an edge between two vertices i and j is added if there are any nonzeros in the block matrix defined by the i^{th} block row and j^{th} block column. The basic characteristic of all algebraic multigrid methods is to coarsen the graph representation of the matrix. There are many ways to define coarse meshes and grid transfers. Perhaps the most well-known is the classical AMG approach of Ruge-Stüben [186] where a subset of fine mesh vertices are used to define the next coarser mesh. In this chapter, an alternative scheme is employed where fine mesh vertices are grouped in aggregates. Each aggregate effectively represents a coarse grid vertex. Once the coarse mesh is determined, a grid transfer must be defined. The simplest possible grid transfer is to use piecewise constant interpolation. In this case, the grid transfer, P , contains only zeros and ones. In the scalar PDE case, $P(i, j)$ is equal to one only if the i^{th} fine grid point has been assigned to the j^{th} aggregate. Within a PDE system, the grid transfer is a block system where a small identity matrix is inserted within the $(i, j)^{th}$ block if the i^{th} fine grid point has been assigned to the j^{th} aggregate.

There are many ways to define aggregates. In a standard algebraic multigrid method the most common way to create an aggregate is via some kind of greedy graph algorithm where an initial node is chosen along with all of its nearest neighbors. The net affect of this type of procedure is to produce aggregates which are “sphere-like” with an approximate diameter of three nodes. Unfortunately, this type of procedure leads to

many aggregates and must be repeated several times in order to obtain a coarse matrix that is small enough to be efficiently solved by a direct solver. Since our goal is to mirror a domain decomposition method where only a couple of coarse meshes are used, we consider a more aggressive coarsening scheme that produces larger aggregates and is better suited for parallel computations. Specifically, the graph partitioning codes METIS and ParMETIS [123] are used to define the aggregates. The algorithms within these packages partition the fine matrix so that each partition has no more nodes than a user supplied parameter. All the nodes in each partition are then combined to form a single aggregate. METIS is a serial code and so can only be used to partition the graph corresponding to the local matrix entries within a processor. This implies that no aggregate can span more than one processor. ParMETIS is a parallel code and so it can be applied to the entire distributed global graph to produce aggregates that span more than one processor.

The procedure we have outlined is sometimes referred to as *nonsmoothed aggregation*. All the results presented in this chapter will refer to nonsmoothed aggregation. In fact, even if for elliptic problems the so-called *smoothed aggregation* performs significantly better than nonsmoothed [234, 54], the correct smoothing procedure for nonsymmetric operators is still an open problem. Often, nonsmoothed aggregation avoids stability problems that arise when smoothed aggregation is used. Further analysis on this subject is reported in [195].

For all preconditioners presented in this chapter, the ratio between the size of the matrices for two consecutive levels is rather high, e.g. of order 500 for two-level methods and of order 100 for three-level methods. Because of this, often more effective (and heavyweight) smoothers are required, as lightweight smoothers, typically used in multigrid applications, may be ineffective.

The aggregation scheme and construction of restriction and prolongation operators is implemented within the ML library [194]. Support for the coarse matrix direct solvers KLU [192] and distributed version of SuperLU [83] are provided through the Amesos interface [192]. ML, Amesos, and Aztec are available through the Trilinos framework [112].

The single level j^{th} overlap Schwarz preconditioner is given by

$$\mathbf{M}_{1level}^{-1} = \sum_{k=1}^P \mathbf{I}_k^i (\tilde{\mathbf{A}}_k^i)^{-1} (\mathbf{I}_k^i)^T,$$

where P is the number of subdomains, \mathbf{I}_k^i interpolates from the k^{th} subdomain to the entire domain, and $\tilde{\mathbf{A}}_k^i$ refers to the approximate solver on the k^{th} subdomain.

The general form of the two-level method (with finest grid corresponding to level j) is given by

$$M_{2level,j}^{-1} = \hat{A}_j^{-1} + P_j \hat{A}_{j-1}^{-1} P_j^T (I - A_j \hat{A}_j^{-1})$$

where A_j is the discrete operator on the j^{th} level, \hat{A}_j^{-1} is the approximate solver on the j^{th} level, P_j is the interpolation operator from the $(j-1)^{th}$ level to the j^{th} level. The standard two-level scheme is given by taking $j = 1$, $\hat{A}_1^{-1} = \mathbf{M}_{1level}^{-1}$, and \hat{A}_0^{-1} to be the approximate solve on the coarse mesh.

The three-level method can be obtained by taking $j = 2$, $\hat{A}_2^{-1} = \mathbf{M}_{1level}^{-1}$, and replacing \hat{A}_1^{-1} in the above expression by an additional two-level method $M_{2level,1}^{-1}$. More general recursive expressions for multigrid algorithms with an arbitrary number of levels can be found in [52].

14.5 Results and Discussion

Algorithmic scaling studies comparing the one-level additive Schwarz preconditioner and two-level Schwarz preconditioner with geometric coarse operator were presented in a previous work [209]. These previous results demonstrated that the two-level Schwarz preconditioner with geometric coarse operator provided a one to two order of magnitude reduction in solution time in two dimensions and a factor of 5–8 reduction in solution time in three dimensions for a standard CFD benchmark problem of thermal convection in a square or cube geometry. The present work considers algorithmic scaling studies for the two- and three-level Schwarz preconditioner using an algebraic coarse grid applied to the following problems:

- 3D thermal convection problem
- Navier-Stokes flow in a 3D building geometry
- convection-diffusion equation.

On a subset of these problems we also compare these algebraic multilevel preconditioners with the one-level DD and the two-level geometric preconditioner.

For all the studies, two different example problems will be used. For the first example problem, previously used in [209], a thermal convection (or buoyancy-driven) flow in a differentially heated 1x1x1 cube in the presence of gravity is modeled. The momentum transport, energy transport and total mass conservation equations are solved. No-slip boundary conditions are applied on all walls. The temperature on the heated wall and other parameters are chosen so that the Rayleigh number is 1000. Figure 14.1a shows isosurfaces for x-component of velocity for a typical steady-state solution. This simple geometry facilitates the construction of regular-shaped subdomains as well as the coarse grid for the two-level preconditioner with geometric coarse operator. The second example involves the calculation of fluid flow, without thermal effects, in a simple prototype model of a building. This test case has been selected because the geometry and boundary conditions are representative of actual large-scale indoor structures. The dimensions of the 3D structure are 105 m x 24 m x 11 m. Inlets are located on the 24m sidewalls of both floors, the 105m length faces, the ceiling of the lower level, and the floor of the upper level. Outlets are located on the ceiling of the upper floor, with the main outlet collectors located above the atria between the two floors. Figure 14.1b shows a typical laminar steady-state solution for each problem. The centerline cutting plane shows the x-component of velocity. Unless otherwise stated, both test cases employ hexahedral meshes with bilinear finite elements.

14.5.1 Comparison between the two-level Schwarz preconditioner with geometric coarse operator and algebraic coarse operator

Table 14.1 presents the results for a scalability study that compares the two-level geometric coarse operator preconditioner with a two-level algebraic coarse operator preconditioner with nonsmoothed aggregation for the 3D thermal convection problem. These results were obtained on the ASCI Red computer at Sandia

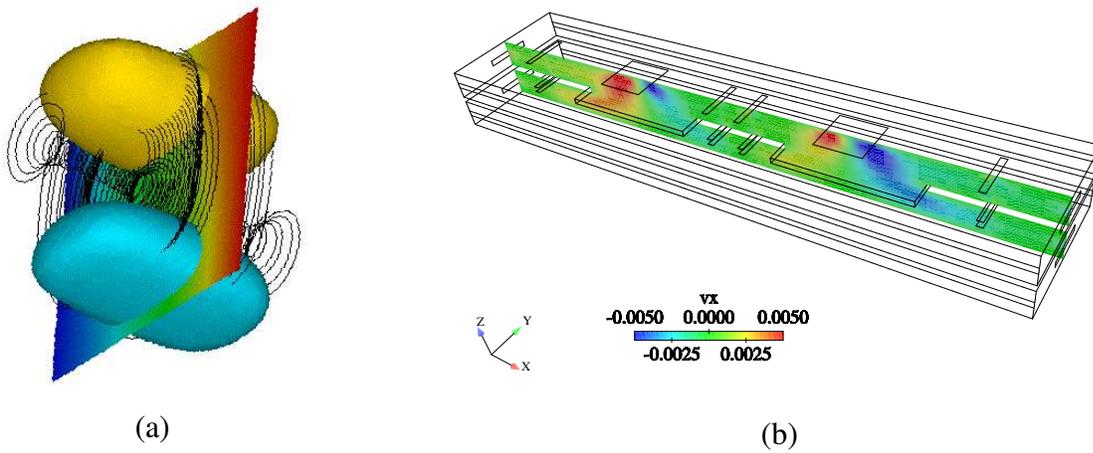


Figure 14.1. (a) Constant x-component of velocity isosurfaces with streamlines and temperature contours on slice plane for thermal convection problem with $Ra=1000$; (b) Steady-state x-component of velocity on centerline cutting plane for model 3D building

National Laboratories, each node containing 256 MB RAM and 333 MHz Pentium II Xeon processors. Figure 14.2 plots these results for the iteration count. The same fine mesh was used for both cases. The number of fine mesh nodes per aggregate was chosen in an attempt to produce coarse matrices of comparable sizes. The mesh for each successive row in the table is produced by a uniform refinement of the mesh in the previous row by cutting each cube into eight cubes. This roughly increases the number of unknowns by a factor of eight. The number of processors is also increased by a factor of eight so that the number of unknowns per processor stays roughly constant. For an algorithm with perfect scalability, both the average number of iterations per Newton step and the CPU time should stay roughly constant. For most of the subsequent scaling studies, the sequence of meshes is produced in this fashion. “GS2/SuperLU” in columns 5-8 indicates that the fine mesh smoother employs two sweeps of local Gauss-Seidel and the coarse mesh smoother uses the distributed SuperLU package [146]. Local Gauss-Seidel is a domain decomposition hybrid where each processor performs standard point Gauss-Seidel on its subdomain (as opposed to applying Gauss-Seidel to the entire global domain). Distributed SuperLU is a parallel direct solver package. “GS2/GMRES-ILU” in columns 9-12 denotes that the fine mesh smoother is two sweeps of local Gauss-Seidel and the coarse mesh smoother employs GMRES with an ILU preconditioner. The coarse grid GMRES is iterated to convergence upon each invocation of the preconditioner. This variant is needed to tackle the relatively large coarse mesh in the 2048-processor case. This coarse mesh is essentially the smallest size that is possible with our geometric coarse mesh software. Unfortunately, the parallel machine does not have sufficient memory for the direct solver to function. The column “avg iter” denotes the average number of iterations per Newton step; the second number in brackets following the average number of iterations is the number of Newton steps. The “time” column reports the CPU time for the steady-state solve. From these results, the two-level preconditioner with an algebraic coarse operator seems competitive with the two-level preconditioner with a geometric coarse operator. As a point of comparison, the one-level solver using a DD ILU preconditioner took 650 iterations per Newton step and

proc	fine grid unks	coarse unknowns		2-level: GS2/SuperLU				2-level: GS2/GMRES-ILU			
				geometric		algebraic		geometric		algebraic	
		geom	alge	avg iter	time (s)	avg iter	time (s)	avg iter	time (s)	avg iter	time (s)
4	24.6K	135	120	33 [5]	78	30 [4]	71	33 [5]	94	30 [4]	69
32	180K	625	480	44 [4]	94	50 [4]	109	44 [4]	114	51 [4]	147
256	1.37M	3645	2560	47 [5]	219	58 [4]	152	47 [5]	238	58 [4]	196
2048	10.7M	24.6K	20.5K	ran out of memory				47 [4]	598	59 [4]	681

Table 14.1. Comparison of geometric and algebraic two-level preconditioners for the 3D thermal convection problem; ASCI Red machine.

2915 seconds for the 10.7 million unknown 2048-processor case [209]. From Table 14.1 and Figure 14.2 it is clear that the preconditioner with an algebraic coarse operator also maintains the optimal iteration count per Newton step, i.e. iteration count asymptotically levels out as the problem is scaled up while maintaining fixed problem size per processor, although the average iterations per Newton step tended to be slightly higher than with the geometric coarse operator. A possible explanation is that while the fine mesh has the same number of unknowns for the both cases, the two-level preconditioner with algebraic coarse operator has a coarse level with fewer unknowns than the preconditioner with geometric coarse operator. The number of iterations is the same when using either SuperLU or GMRES-ILU as the coarse iterative solve is iterated to convergence. However, the run time is noticeably longer using the coarse iterative solver. This is partially due to the direct solver's reuse of the matrix factorization (that need only be computed once for each Newton step). It may be possible to improve the reuse capability within the iterative coarse solver (e.g. save and reuse Krylov vectors from the previous coarse solve) to make the times closer to that of the direct solver.

14.5.2 Algorithmic scaling studies for three-level Schwarz preconditioner with algebraic coarse operators for the 3D thermal convection problem

Scalability studies were performed for the 3D thermal convection problem using the three-level preconditioner for both the laminar steady-state and transient cases. Unless otherwise stated, for the three-level preconditioner, the fine and medium mesh smoothers are one sweep of Gauss-Seidel and ILU respectively, and the coarse solver is the distributed version of SuperLU. These calculations, as well as the remaining calculations in this chapter (with the exception of the calculations in Table 14.5), were performed on the Sandia Cplant machine which is composed of 1024 nodes, each with 500 MHz Dec Alpha processor and 1 GB of RAM, connected together by Myrinet. The performance of each processor is very roughly comparable to a 1-GHz Intel Pentium III processor.

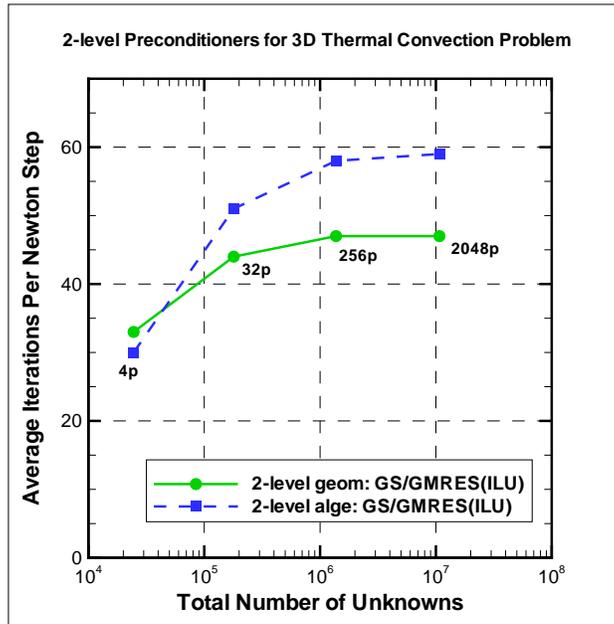


Figure 14.2. Comparison of iteration count as a function of problem size for the geometric and algebraic two-level preconditioners for the 3D thermal convection problem

nodes per aggregate	proc	unknowns			avg its/ Newt step	time (sec)
		fine	medium	coarse		
50,200	2	180K	3590	15	49 [4]	488
	16	1.37M	27440	135	93 [4]	835
	128	10.7M	214K	1070	111 [4]	1191
	1024	84.9M	1.69M	8470	109 [4]	2050
100,100	2	180K	1790	15	52 [5]	615
	16	1.37M	13680	135	93 [4]	812
	128	10.7M	107K	1065	104 [4]	1101
	1024	84.9M	845K	8445	109 [4]	2117
200,50	2	180K	890	15	64 [4]	574
	16	1.37M	6800	135	95 [4]	844
	128	10.7M	53120	1060	111 [4]	1149
	1024	84.9M	420K	8395	119 [4]	2428

Table 14.2. Scalability study of three-level preconditioner (GS1/ILU/SuperLU) for the steady 3D thermal convection problem (aggregation: METIS; ParMETIS); Cplant machine.

The first group of scalability studies concern the steady-state solution for the thermal convection problem with Rayleigh number of 1000. Table 14.2 shows an algorithmic scaling study for different numbers of nodes per aggregate. In the first column, the first and second number are the nodes per aggregate when constructing the medium level and coarse level respectively. “Avg its/Newt step” denotes the average number of iterations per Newton step, and the number of Newton steps is denoted by the number in brackets. The final column is the time for the steady-state solve. Within each group with the same number of nodes per aggregate, the smallest calculation is with a starting mesh that is run on two processors. As in the previous scaling study, the mesh is then uniformly refined three times, each level of uniform refinement increasing the number of hexahedra by a factor of eight, which increases the number of unknowns by a factor of roughly eight. The number of processors is also increased by a factor of eight so that the number of unknowns per processor stays roughly constant. After each level of uniform refinement, the fine mesh is load-balanced using Recursive Coordinate Bisection (RCB) through the Zoltan data management services for parallel applications package [84]. Aggregates are generated by resorting to local graph partitioning algorithms for the fine level (METIS) and a global graph partitioning algorithm (ParMETIS). Figure 14.3 plots the results from Table 14.2 for the number of iterations as a function of problem size. From the figure, the number of iterations per Newton step is asymptoting to a fixed value, especially for the “50,200” nodes per aggregate case. Although one more level of uniform refinement (which would be run on 8192 processors) would probably be necessary to fully demonstrate this, the trend is clear.

The next group of algorithmic scaling studies concern transient flow for the 3D thermal convection problem. Table 14.3 shows the algorithmic scaling study. In this table, “avg its/Newt step” denotes the

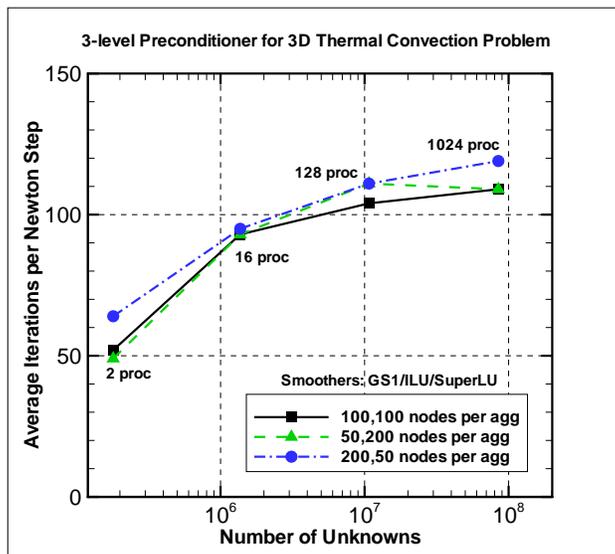


Figure 14.3. Comparison of iteration count as a function of problem size for the three-level preconditioner with different size medium level for the 3D thermal convection problem

average number of iterations per Newton step and “time” denotes the average time per time step with both quantities calculated by averaging the first ten time steps. 100 nodes per aggregate with METIS and ParMETIS schemes for the first and second levels of aggregation respectively was used. Note that for each time step size, the average iterations per Newton step grows slowly. These iterations appear to be slowly asymptoting to a fixed value but a further level of uniform refinement would probably be needed to clearly demonstrate this.

14.5.3 Algorithmic scaling studies for three-level Schwarz preconditioner with algebraic coarse operators for Navier-Stokes flow in a 3D building geometry

As can be seen from the previous results, when the problem is scaled in size and proper number of processors to roughly maintain the same number of unknowns per processor, the three-level preconditioner obtains the optimal convergence property for a simple cube geometry. The next group of studies will investigate whether this optimal iteration convergence property can be maintained for a more realistic computational domain. Scalability studies were performed for the model 3D building for the three-level preconditioner for both steady-state and transient cases. Many of the same scalability studies performed with the 3D thermal convection problem are repeated for the 3D building, including the effect of varying the number of nodes per aggregate. As before, unless otherwise stated, the smoothers and solvers chosen for the three-level preconditioner are one sweep of Gauss-Seidel on the fine level, ILU on the medium

time step	approx max CFL	proc	unknowns			avg its/ Newt step	time (sec)
			fine	medium	coarse		
0.01	1	2	180K	1790	15	30	223
		16	1.37M	13680	135	48	330
		128	10.7M	107K	1065	59	527
0.1	10	2	180K	1790	15	44	296
		16	1.37M	13680	135	76	490
		128	10.7M	107K	1065	87	730
1.0	100	2	180K	1790	15	51	324
		16	1.37M	13680	135	90	568
		128	10.7M	107K	1065	103	852

Table 14.3. Scalability study of three-level preconditioner (GS1/ILU/SuperLU) for the transient 3D thermal convection problem. METIS (ParMETIS) aggregation used on the fine (middle) mesh with 100 nodes per aggregate; Cplant machine.

level, and the distributed version of SuperLU for the coarse level.

For the steady-state calculations, the Reynolds number is restricted to two orders of magnitude lower than normal operating conditions based on an inlet duct reference length and velocity to provide a laminar flow. The low Reynolds number allows quick direct-to-steady-state solution and simplifies the presentation of the results. The transient turbulent large eddy simulation (LES) calculations are performed at the standard operating conditions of the ventilation systems. The 3D hexahedral meshes and tetrahedral meshes were generated by the Sandia Cubit mesh generation software [87].

Table 14.4 shows an algorithmic scaling study for different numbers of nodes per aggregate for the steady-state calculations on hexahedral meshes. The finer meshes are generated by uniform refinement of previous meshes, with the number of processors being increased to maintain a roughly constant number of unknowns per processor. In contrast to the 3D thermal convection problem, after each level of uniform refinement of the building geometry, the fine mesh is load-balanced using the ParMETIS graph partitioner through Zoltan. Note that in contrast to the thermal convection problem, the optimal iteration count is not obtained for the 3D building problem. This is most likely due to the fact that the combination of the aggregation scheme and the smoother is not handling hexahedral elements with medium or high aspect ratios properly. This is a well-known problem with multigrid methods where the performance of the smoother deteriorates in the “long” direction of elements with high aspect ratios. Remedies have been extensively studied in the context of structured meshes. These are primarily based on block relaxation techniques to improve smoothing in the long direction or semi-coarsening to coarsen less in the direction where smoothing is poor. This subject has been studied (though to a lesser extent) for algebraic methods and we are exploring possible enhancements to both our aggregation and smoothing methods to address

this. For this particular problem, the hexahedral elements vary in aspect ratio with the worst aspect ratio elements having a longest dimension that is a factor of 5.3 larger than the shortest dimension. The hexahedral meshes used are actually structured meshes. Previous experience has shown that when the elements are close to squares and cubes for two and three dimensions respectively, then the optimal convergence property is obtained. Reference [69] notes the effect of the aspect ratio of the elements on algebraic methods and suggests a promising alternative approach.

Further insight into the problem can be gained by considering a 2D thermal convection problem with differentially heated walls (2D version of Figure 14.1a) in a rectangular domain that is sixteen times longer than it is high. The two-level preconditioner with geometric coarse mesh is used, with ILU smoother on the fine mesh and KLU [192] solver on the coarse mesh. The top half of Table 14.5 considers the discretization of the domain with square-shaped finite elements, so there are 16 times as many elements in the lengthwise direction. For each successive row, the mesh is refined by splitting a square element into four squares. Note that the optimal convergence property is obtained as the problem is scaled from 4 to 64 processors. The bottom half of the table considers the discretization of the domain with rectangular-shaped finite elements that are four times longer than they are high. The number of unknowns per processor is roughly the same as the corresponding row in the top half of the table. Note that the optimal iteration count is not obtained. This shows that even the preconditioner with geometric coarse mesh (where aggregation is not used) does not properly handle the medium aspect ratio finite elements. This is most likely an indication that the smoother does not sufficiently smooth error in the stretched direction. It also illustrates the difficulties in applying black-box solvers to a wide range of problems.

To further demonstrate that the issue of failure to obtain the optimal convergence property is with the aspect ratio of the finite elements and not the geometry, a scalability study was performed on a tetrahedral mesh of the 3D building where the tetrahedra are of relatively low aspect ratio. This tetrahedral mesh was generated from a triangular surface mesh by using Cubit's volume mesh generator rather than taking the previous hexahedral mesh and cutting a hexahedron into six tetrahedra. In contrast to the hexahedral mesh case, the finer tetrahedral meshes were not created from uniform refinement of coarser tetrahedral meshes. They were created by initially reducing the interval size between nodes on the surface triangulation by half, with further reduction of interval size in order to give an increase in the number of nodes by a factor of 7–8. Table 14.6 shows this algorithmic scaling study. It also shows a comparison between the one-level ILU preconditioner and the three-level preconditioner. Note that the three-level preconditioner provided a factor of four reduction in time over the one-level preconditioner for the 128-processor case. The number of iterations per Newton step for the one-level preconditioner scales as roughly the number of unknowns to the power of $\frac{1}{3}$ while for the three-level preconditioner seems to be asymptoting to a constant value. This demonstrates that the optimal convergence property is obtained when good aspect ratio finite elements are used. We are currently working on a fix to this problem so that the optimal convergence property will be obtained for both medium and high aspect ratio finite elements.

The next group of algorithmic scaling studies concern transient flow in the 3D building. The large eddy simulation (LES) model used was the LES-k model [208]. Table 14.7 shows the algorithmic scaling study. Two different time steps were used: $\Delta t = 0.1$ seconds and $\Delta t = 1.0$ seconds. In this table, “avg its/Newt step” denotes the average number of iterations per Newton step and “time” denotes the average time per time step with both quantities calculated by averaging the first ten time steps. 100 nodes per aggregate with METIS and ParMETIS schemes for the first and second levels of aggregation respectively was used. The scaling in Table 14.7 is much better than that exhibited in the steady-state study even though similar quality

nodes per aggregate	proc	unknowns			avg its/ Newt step	time (sec)
		fine	medium	coarse		
50,200	2	227K	4544	20	38 [5]	458
	16	1.70M	33988	168	52 [5]	638
	128	13.1M	263K	1312	67 [5]	1142
	1024	103M	2.06M	10316	121 [5]	2656
100,100	2	227K	2272	20	40 [5]	466
	16	1.70M	16976	168	53 [5]	629
	128	13.1M	131K	1308	76 [5]	1160
	1024	103M	1.03M	10304	136 [5]	2924
200,50	2	227K	1136	20	43 [5]	482
	16	1.70M	8480	168	60 [5]	689
	128	13.1M	65460	1308	89 [5]	1344
	1024	103M	514K	10284	160 [5]	3479

Table 14.4. Scalability study of three-level preconditioner (GS1/ILU/SuperLU) for the steady 3D building problem with hexahedral mesh (aggregation: METIS, ParMETIS); Cplant machine.

element aspect ratio	proc	fine mesh	coarse mesh	avg its/ Newt step	time (sec)
1:1	4	257 × 17	33 × 3	29	15
	16	513 × 33	65 × 5	31	22
	64	1025 × 65	129 × 9	32	48
4:1	4	129 × 33	17 × 5	55	23
	16	257 × 65	33 × 9	135	65
	64	513 × 129	65 × 17	convergence failed	

Table 14.5. Scalability study of two-level preconditioner (ILU/KLU) with geometric coarse mesh for the steady 2D thermal convection problem; ASCI Red machine.

proc	fine	medium	coarse	1-level		3-level	
				avg its/ Newt step	time (sec)	avg its/ Newt step	time (sec)
2	227K	2272	20	83 [5]	910	35 [5]	454
16	1.68M	16832	168	148 [6]	1571	56 [5]	605
128	13.1M	131K	1308	302 [6]	4099	51 [6]	1000

Table 14.6. Scalability study of 1-level (ILU) and three-level preconditioner (GS1/ILU/KLU) for the steady 3D building problem with tetrahedral mesh (aggregation: METIS, ParMETIS); Cplant machine.

hexahedral meshes were used. Though further analysis is needed, it appears that the lumped mass matrix term may offset the suboptimal behavior observed for medium aspect ratio hexahedral mesh steady-state problems.

14.5.4 Comparisons between one-level, two-level, and three-level preconditioners for fluid flow in a 3D building geometry

Table 14.8 compares various one-level, two-level, and three-level preconditioners for steady laminar flow in the 3D model building with Reynolds number two orders of magnitude lower than actual conditions. This geometry has 2.6 million grid points (10.3 million unknowns) and 2.5 million hexahedral elements. These cases were run on 128 nodes of the Sandia Cplant machine. For this test case, the two-level preconditioner with geometric coarse operator performed marginally better than the one-level preconditioner. This is mainly due to the substantial cost of the direct solve on the coarse mesh. The better CPU time of the algebraic two-level preconditioner compared with the geometric preconditioner is likely due to the smaller coarse matrix. The algebraic three-level preconditioner performed well due to the fact that an inexpensive smoother such as Gauss-Seidel could be used on the fine mesh.

Table 14.9 presents a comparison between various one-level, two-level, and three-level preconditioners for a transient LES simulation in the 3D model building with realistic Reynolds number based on inlet dimensions and inlet velocity. This geometry has 3.3 million nodes (13.1 million unknowns) and 3.2 million hexahedral elements. The size of the time step is $\Delta t = 0.1$ seconds and the approximate maximum CFL on the fine grid is 0.01. These cases were run on 1000 nodes of the Sandia Cplant machine. Because of limited access to 1000 nodes of the machine, only the first four Newton steps of the first time step were run for comparison. Note that the two-level preconditioner with geometric coarse operator did worse than the one-level preconditioner. This is likely due to the expensive coarse grid solve and the relatively good convergence obtained by the one-level preconditioner when a well-conditioned transient system is solved. The algebraic three-level preconditioner did substantially better than the one-level or two-level

time step	approx max CFL	proc	unknowns			avg its/ Newt step	time (sec)
			fine	medium	coarse		
0.1	0.03	2	224K	2230	20	71	530
		16	1.67M	16670	165	72	501
		128	12.9M	129K	1285	75	619
1.0	0.3	2	224K	2230	20	46	322
		16	1.67M	16670	165	52	371
		128	12.9M	129K	1285	58	574

Table 14.7. Scalability study of three-level preconditioner (GS1/ILU/SuperLU) for the transient 3D building problem (LES-k) with hexahedral mesh. METIS (ParMETIS) aggregation used on the fine (middle) mesh with 100 nodes per aggregate; Cplant machine.

method	smoothers/solver	nodes per aggregate	unknowns		avg its/ Newt step	time (sec)
			medium	coarse		
1-level DD	ILU				188 [5]	3074
2-L geom	ILU/SuperLU			25520	38 [5]	2694
2-L geom	GS2/SuperLU			25520	did not converge	
2-L alge	ILU/SuperLU	512		19948	37 [5]	2104
2-L alge	GS2/SuperLU	512		19948	237 [5]	5918
3-L alge	GS2/ILU/SuperLU	64	201K	1256	85 [5]	1419
3-L alge	GS2/GS2/SuperLU	64	201K	1256	103 [5]	1604

Table 14.8. Comparison of different preconditioners for the 3D model building; steady-state; fine mesh with 10.3 million unknowns; 128 processors on Cplant machine.

method	smoothers/ solver	nodes per aggregate	unknowns		avg its/ Newt step	time (sec)
			medium	coarse		
1-level DD	ILU				113	150
2-L geom	ILU/GMRES-ILU			32336	24	255
3-L alge	GS/ILU/SuperLU	100	129K	1292	31	38
3-L alge	GS/ILU/SuperLU	512	20376	44	56	53

Table 14.9. Comparison of different preconditioners for 3D model building; transient LES; fine mesh with 13.1 million unknowns; 1000 processors on Cplant machine

preconditioners because the use of three levels allowed an inexpensive smoother such as Gauss-Seidel to be used on the large fine mesh. This becomes a significant advantage as the size of the fine mesh increases, especially when a two-level preconditioner will not converge with an inexpensive smoother on the fine mesh and an expensive smoother such as ILU is required. This was very apparent when a steady-state laminar calculation (with Reynolds number two orders of magnitude too low) was performed for the 3D model building with 25.8 million nodes (103 million unknowns) on 1000 nodes of the Cplant machine. Both the one-level and two-level methods require an expensive fine mesh smoother (ILU), and after 3 hours, the solver was still computing the first ILU factorization. The algebraic three-level preconditioner with Gauss-Seidel, ILU, and SuperLU on the fine, medium, and coarse mesh respectively reached steady-state after 75 minutes. Note that in the three-level preconditioner scaling studies, with a choice of 50 and 200 nodes per aggregate for first and second levels of aggregation, a steady-state solution could be obtained in 45 minutes.

14.5.5 Algorithmic scaling studies for three-level Schwarz preconditioner with algebraic coarse operators for the solution of the convection-diffusion equation

Scalability studies were performed for a single convection-diffusion equation, with the three-level Schwarz preconditioner. For the three-level preconditioner, the fine and medium mesh smoothers are one sweep of Gauss-Seidel and ILU respectively, and the coarse solver is the KLU direct solver. METIS and ParMETIS were used for aggregation for the fine and middle mesh respectively. A flow field that was obtained from a prior steady-state laminar Navier-Stokes calculation in the model 3D building provides the velocity field for the transient convection-diffusion equation calculation. Table 14.10 shows a comparison between the one-level Schwarz preconditioner and the three-level Schwarz preconditioner for different sized time steps. The transient calculation was run for ten time steps. The reported “average iterations per Newton step” and “time” is the averaged iterations per Newton step and time over the ten time steps. Note that the one-level preconditioner required fewer iterations than the 3-level preconditioner (when Gauss-Seidel is the

time step	proc	unknowns			1-level		3-level			
		fine	medium	coarse	avg its/ Newt step	time (sec)	GS/ILU/KLU		ILU/ILU/KLU	
							avg its/ Newt step	time (sec)	avg its/ Newt step	time (sec)
0.1	2	45K	446	4	2	9	3	9	—	—
	16	334K	3334	33	2	15	3	15	2	15
	128	2.58M	26K	257	2	21	4	20	—	—
1.0	2	45K	446	4	2	9	4	10	—	—
	16	334K	3334	33	2	15	5	15	2	15
	128	2.58M	26K	257	3	21	6	21	—	—

Table 14.10. Scalability study of three-level preconditioner (GS1/ILU/KLU and ILU/ILU/KLU) for solution of a convection-diffusion equation in the 3D building. METIS (ParMETIS) aggregation used on the fine (middle) mesh with 100 nodes per aggregate; Cplant machine.

smoother on the fine mesh). This is due to the fact that this problem is very well-conditioned and therefore the one-level method performs extremely well. As presented in the table for the 16-processor case, the use of ILU rather than one sweep of Gauss-Seidel as the fine mesh smoother for the three-level preconditioner would yield the same iteration count as for the one-level preconditioner. This demonstrates that for simple, well-conditioned problems, the one-level ILU preconditioner performs well and there is no advantage to using the multilevel preconditioner.

14.6 Conclusions

This study compares, for low to medium Reynolds number Navier-Stokes flows, several preconditioners based on domain decomposition and multilevel schemes, including one-level additive Schwarz, a two-level Schwarz with geometric coarse operator, and two- and three-level Schwarz based on aggregation. Except for trivial problems, as expected, the one-level preconditioner is not scalable. The two-level preconditioners, one based on a geometric coarse operator and the other based on an algebraic coarse operator, are scalable for medium-size problems, and have comparable performance. For large-size problems the CPU-time required to solve the coarse level problem becomes predominant, and the preconditioner fails to be scalable. By resorting to three-level preconditioners based on aggregation, the optimal iteration count for a 3D benchmark thermal convection problem hexahedral meshes was obtained as well as for the 3D model building problem with tetrahedral meshes. Results demonstrate that often the use of more than two levels for the algebraic preconditioner resulted in faster CPU times. Work is currently in progress to allow the optimal convergence property to be obtained for medium and high aspect ratio

finite elements. Further study is needed to compare how the preconditioners perform for highly convective flows and for reactive flows.

Chapter 15

Rapid Source-Inversion for Chemical/Biological Attacks, Part 1: The Steady-State Case

Paul T. Boggs, Kevin R. Long, Stephen B. Margolis Patricia A. Howard (Worcester Polytechnic Institute)

15.1 Background

The key in responding to a chemical or biological attack in a building, e.g., an airport, is speed. In an attack using an airborne toxin, emergency officials need to know as soon as possible what the toxin is and where its source is located. This will allow them to take appropriate action to minimize the impact of the attack, including moving people to safety, confining or venting the release, and possibly even determining who is responsible.

Two things are needed to determine the type and source of an attack: First, one needs sensors that can reliably determine the nature of the toxin and the local concentration; and second, one needs to have a mathematical model of the toxin dispersion in the air flow in the building that can be inverted to provide the location of the source. Here we assume that appropriate sensors have been placed in the building. Such sensors are being actively developed in several laboratories, including Sandia National Laboratories, (see, e.g., [77]); we comment on some related work designed to improve these sensors in section 15.6. The location of the sensors within the building is clearly an important issue in being able to reconstruct the source field. We do not consider the optimal location of the sensors in this chapter, but, again, we discuss the issues involved in section 15.6. In this chapter we concentrate on the source inversion problem that simultaneously predicts both the number of sources in the attack and the location of each, given the concentration data from the sensors and the air flow field.

Due to the dramatic increase in computational power afforded by massively parallel computers, the

substantial gains in our optimization technology, and the major improvements in our understanding and ability to precondition the linear systems that arise in these problems, PDE-constrained optimization has emerged as important research area. (See [228] for an introduction and the papers in [40] for more extensive coverage.) Source inversion has been considered by [18] and excellent work in preconditioning is given in [43].

The source inversion problem that we have briefly described above has important features that have motivated our work. In particular, speed of solution is much more important than accuracy. Indeed, if we can correctly predict the number of sources and their locations to within a few meters, emergency personnel can easily find the release devices. As stated above, we need to know the flow field in the building to determine the locations quickly. The air flow in a modern building is determined by the heating, ventilating, and air-conditioning (HVAC) system. It is our understanding that the flows in such a building do not change much over the course of a day. Thus we can assume that a small number of flow fields can be computed in advance and the appropriate one can be selected depending on the time of the attack. We can also assume that there will be moderate computing capability in the building. Indeed, given the expense of security in general, the purchase of a small cluster of processors seems quite reasonable.

Since one of our main motivations is speed, we also investigate some strategies for decreasing the time for the calculations. In particular, we will require a relatively fine mesh to solve for the flow field within a given building. The question arises, however, of whether the estimation of the source, or sources, can be done on a coarser mesh. We show that, in fact, a much coarser mesh will suffice and that using such a coarse mesh reduces the computing time by factors of 40–100. This rather surprising result indicates that a practical system using these strategies may be possible. Thus, novel contributions in this chapter include the model that allows the simultaneous determination of both the number and the locations of the source(s) and the fact that very coarse meshes can be used to accelerate the inversion algorithms significantly.

Finally, the problem is clearly time-dependent, but we think that much can be learned initially from considering the simpler steady-state case. We demonstrate that consideration of a two-dimensional, steady-state model can provide much insight that will guide our efforts in the general 3-D, time-dependent case. We do not claim that we have answered all of the questions for this case, but we can do the additional development and experiments in the context of the more general case that will be considered in part 2 of this work. In addition, the steady-state case is of interest in its own right; one can use the techniques developed here to find, for example, a persistent leak in a complex facility.

In conducting such a project, it was crucial to have a powerful software environment in which to develop and test ideas rapidly. This work would have been far more difficult without the tools that have been developed at Sandia National Laboratories by many people. Most important for this work is Sundance, a code that takes a symbolic description of PDEs and then efficiently creates finite element (mass) matrices and associated right-hand side vectors. The details of how we formulate the flow field problems and the optimization problems are dictated by our use of Sundance. An optimization code, Split and O3D, has been built to work directly with Sundance operators.

The chapter is organized as follows. Given the importance of the software environment, and its impact on some of our analyses and decisions, we describe it first in section 15.2. We next describe (in section 15.3) the flow problem that we formulate and solve. In creating the flow field, we tried to specify realistic models that take into account the fact that the air flow in a building such as an airport is subject to many

perturbations, including the opening and closing of doors and the movement of people, luggage, and equipment. We then provide (in section 15.4) the appropriate source dispersion model and the resulting optimization problems that we consider to locate the sources. In section 15.5, we give a simple two-dimensional example of a building and the resulting flow field. We then give the results of our numerical tests based on the optimization approaches in section 15.4. We also develop our coarse mesh approximations and show the results using these. In section 15.6 we provide some discussion about the work that needs to be done to extend this approach to the time-dependent case and to the question of optimal sensor location.

15.2 Software Environment

In this section we briefly describe the software environment that was used for all of the computations in this chapter. We emphasize that the development of and experimentation with the models and approaches described here were greatly aided by this powerful set of software tools. We also point out that most of the tools described here are publically available.

The most important tool for our work with PDE-constrained optimization models is Sundance (see [149]). Sundance is a system for specifying, building, and applying finite element approximations to general PDEs. Sundance consists of user-callable components written in C++ that allow the user to specify the PDE and associated boundary conditions in weak form using operator overloading on a family of symbolic objects. The Sundance symbolic objects and operators can be used to assemble virtually any PDE. Each test or unknown function in a Sundance problem is constructed with a specifier of its finite-element basis, and any integral can be given a specifier for the type and order of quadrature rule to be used. Stabilization terms can be added at the symbolic level; typically, these involve the mesh size h , so a special symbolic object `CellDiameterExpr` has been created which when evaluated, refers to the mesh to obtain a numerical value of h on each element. The ability to specify basis, quadrature, and optional stabilization terms gives the Sundance user fine control over the discretization process.

Since it is easy to change the equation and/or the BC, it is easy to experiment with different models by making a small number of changes to a code that uses Sundance. This ability to modify models and solution procedures at a high level of abstraction was key in rapidly creating the flow field models to be described in section 15.3. In fact, the code for the Reynolds-averaged Navier-Stokes equations were created and solved in fewer than 200 lines of Sundance code. The symbolic problem setup capability of Sundance is useful not only for rapid development of forward simulators such as our flow model, but even more importantly, it makes possible the concurrent specification of gradient and/or adjoint equations, greatly facilitating the application of gradient-based optimization methods.

Sundance does the work of assembling matrices and vectors from a problem specification; computations on those mathematical objects are then done using the Trilinos family of solver components (see [112]). Trilinos includes a high-performance, low-level matrix/vector library (EPetra), incomplete factorization preconditioners (Ifpack), algebraic multilevel solvers and preconditioners (ML), Krylov solvers (belos), and nonlinear solvers (nox). Trilinos also provides a set of abstract interfaces allowing interoperability with other solver libraries.

Finally, we use a particular optimization method that was also built using Trilinos components and is therefore directly compatible with Sundance. This method, called *O3D* (see [46]), is an interior-point quadratic programming solver. It has been used as the basis for a sequential quadratic programming (SQP) algorithm that has been successfully applied to a number of problems. (See [48] and [47].) *O3D* requires the solution of linear systems that may conveniently be expressed mathematically as block matrices. Trilinos allows such linear operators to be created and manipulated easily. It also allowed us to experiment with different formulations and solution strategies quickly.

15.3 Flow-Field Computation

As noted above, there are only a few flow fields that need to be calculated to solve the source inversion problem. Since all of these can be computed in advance, there is no need to be overly concerned with efficiency at this stage. We first describe and justify the turbulent flow model that we have adopted. We next show the manipulations necessary to create the appropriate weak form required by Sundance. This leads naturally to the means of handling the boundary conditions. We then discuss the use of the eikonal equation to calculate a distance variable that appears in our turbulence model. We conclude this section with a discussion of a pressure-stabilization technique that was useful in formulating a stable and well posed computational algorithm.

15.3.1 The Turbulence Model

The time-dependent continuity and momentum (Navier-Stokes) equations for an incompressible fluid are given, in the absence of body forces, by

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{15.3.1}$$

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j}, \tag{15.3.2}$$

where repeated unhatted indices denote summation. Here, the dependent variables u_i and p are, respectively, the i th component of the fluid velocity and pressure, the independent variables x (with components x_i in a domain Ω) and t are the spatial and temporal coordinates, and ρ and ν are the density and kinematic viscosity, both of which are assumed constant.

For air flow in large rooms, typical values of the dimensionless Reynolds number $Re = UL/\nu$, where U is a characteristic flow velocity ($\sim 1\text{ m/s}$) and L is a characteristic length scale ($\sim 10\text{ m}$), are quite large. In particular, the kinematic viscosity of air at standard temperature and pressure is $\nu \sim 10^{-6}\text{ m}^2/\text{s}$, which gives a value for the Reynolds number of $Re \sim 10^7$. In this regime, the flow is inherently unstable and thus certain to exhibit turbulent behavior in the vicinity of walls and other obstacles. Hence, (15.3.1) and (15.3.2), short of direct numerical simulation, must be supplemented by an appropriate turbulence model.

Our approach to the problem follows the classical methodology of first decomposing the dependent variables into steady and fluctuating quantities, where the latter are not necessarily small. In particular, we define time-averaged (mean) and fluctuating quantities as $u_i = U_i + u_i'$ and $p = P + p'$ where $U_i = \bar{u}_i \equiv \lim_{\tau \rightarrow \infty} \int_{t_0}^{t_0 + \tau} u_i dt / \tau$ and similarly for P . Substituting the definitions for u_i and p into (15.3.1) and (15.3.2) and performing the above time-averaging operation yields the well known Reynolds-averaged Navier-Stokes (RANS) equations given by

$$\frac{\partial U_i}{\partial x_i} = 0, \quad (15.3.3)$$

$$U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho} \frac{\partial P}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_j \partial x_j} + \frac{1}{\rho} \frac{\partial \tau_{ij}}{\partial x_j}. \quad (15.3.4)$$

Here, the components τ_{ij} of the Reynolds stress tensor, which arise from the nonlinear terms in (15.3.2), are given by $\tau_{ij} = -\rho \overline{u_i' u_j'}$ (cf. [198]).

The closure of (15.3.3) – (15.3.4) for the mean quantities U_i and P require that additional relations be specified for the quadratic averages τ_{ij} . Although equations can be developed for these quantities, such equations generally introduce higher-order correlations (e.g., triple averages), thus giving rise to the well known closure difficulty associated with turbulent flow modeling (cf. [198], [67], [241]). Alternatively, physically-motivated assumptions regarding the relationship of τ_{ij} to the mean-flow variables can be introduced, and it is a variant of this approach that we use here. In particular, algebraic relationships (so-called zero-equation turbulence models) are introduced for the dominant components of the turbulent stress tensor based on the classical eddy-viscosity/mixing-length concept for turbulent shear- and boundary-layer flows.

For an essentially 2-D parallel mean flow [e.g., $U_2 \sim 0$, $U_1 \sim U_1(x_2)$], the standard analogy with laminar flow, originally postulated by Boussinesq [51], is to assume that $\tau_{12} = \tau_{21}$ can be expressed as

$$\tau_{12} = \tau_{21} = \rho \nu_t \frac{\partial U_1}{\partial x_2}, \quad (15.3.5)$$

where ν_t is the kinematic eddy viscosity. In contrast to the kinematic laminar viscosity ν , however, it is generally an unacceptably poor approximation to simply regard ν_t as a constant parameter (cf. [168]). For example, it is clear by definition that τ_{12} should approach zero at a no-slip and/or no-penetration boundary, but this is only necessarily true if ν_t vanishes there. Hence, it is customary to appeal to a mixing-length argument originally put forth by Prandtl [169] and to further express ν_t in terms of a mixing length ℓ as

$$\nu_t = \ell^2 \left| \frac{\partial U_1}{\partial x_2} \right|, \quad (15.3.6)$$

where ℓ is also a function of position that must approach zero near a wall.

For boundary-layer flows, an analysis of the turbulent boundary layer indicates that its structure consists of an inner viscous sublayer in which laminar viscous effects are important and the Reynolds stress is negligible an outer defect layer in which the effects of ν are small but those due to ν_t are significant, and an intermediate overlap, or matching, region in which both kinematic viscosities ν_t and ν are non-negligible. It is in the latter regime where the mean-flow velocity varies logarithmically with distance x_2 from the wall, giving rise to the well known “law of the wall”. One modern-day uniformly valid expression for ℓ is given by

$$\frac{\ell}{\delta} = \lambda \left[1 - \exp\left(-\frac{x_2}{A}\right) \right] \tanh\left(\frac{\kappa}{\lambda} \cdot \frac{x_2}{\delta}\right), \quad (15.3.7)$$

[154], where δ is the local turbulent boundary-layer thickness, A is a damping constant and κ and λ are fitted constants which, for smooth walls and no mass transfer, are found to have the approximate values $\kappa = 0.41$, $\lambda = 0.085$ and $A = 26\nu/u_\tau$, where $u_\tau = (\tau_w/\rho)^{1/2}$ is the friction velocity expressed in terms of a specified wall stress τ_w . It is observed that ℓ/δ approaches zero in either a linear or quadratic fashion (depending on whether or not the damping factor is present) as $x_2 \rightarrow 0$, and approaches a constant value as x_2 exceeds the boundary-layer thickness.

In the present work, we propose an extension of (15.3.7), applicable to multidimensional wall-bounded flows, as follows. However, we first remark that since the boundaries in the present problem are not necessarily smooth (due to carpeting, acoustic tiles, structural protrusions, and various objects such as chairs, desks, counters, people, etc.), neither the laminar viscosity ν nor the above quoted turbulence parameters are regarded as given. Rather, they should ideally be fitted to actual data. Secondly, while the functional form given by (15.3.7) is retained to describe the turbulent boundary layers, an appropriate generalization must be given to account for multiple boundaries (e.g., both a floor and a ceiling) and the fact that the local mean flow is not necessarily parallel.

In generalizing the classical mixing-length model to the present 2-D problem, we assume that the dominant effects of turbulence are felt in the effective turbulent boundary layers in the vicinity of walls, floors and ceilings. Consequently, it is assumed that the shear components of the Reynolds stress tensor, τ_{ij} , $i \neq j$, dominate the normal components τ_{ii} . Hence, we define the shear components of the stress tensor $\tau_{ij}^s = \tau_{ij}(1 - \delta_{ij})$, where δ_{ij} is the Kronecker delta and hatted indices imply no summation, and approximate τ_{ij} with τ_{ij}^s in Eq. (15.3.4). Restricting further consideration to two dimensions, we then generalize Eqs.(15.3.5) and (15.3.6) by representing the only shear component $\tau_{12} = \tau_{21} = -\overline{\rho u_1' u_2'}$ as

$$\frac{1}{\rho} \tau_{12} = \nu_{12}' \frac{\partial U_1}{\partial x_2} + \nu_{21}' \frac{\partial U_2}{\partial x_1} = \nu_{ij}' \frac{\partial U_i}{\partial x_j} (1 - \delta_{ij}), \quad (15.3.8)$$

where

$$\nu_{ij}' = \ell_j^2 \left| \frac{\partial U_i}{\partial x_j} \right|, \quad i \neq j, \quad (15.3.9)$$

with the mixing length $\ell_j = \ell_j(x_j)$ defined in a manner consistent with (15.3.7). In particular, we adopt the functional form of (15.3.7) for each ℓ_j , replacing the coordinate x_2 with a distance variable ℓ^* that represents the distance from the nearest *wall* boundary. The latter is calculated by first solving the eikonal equation, as described in section 15.3.3. For simplicity, we take the turbulence parameters δ , λ , κ and A as constants independent of \hat{j} , so that in fact

$$\ell_1 = \ell_2 \equiv \ell = \delta\lambda [1 - \exp(-\ell^*/A)] \tanh [(\kappa/\delta\lambda)\ell^*]. \quad (15.3.10)$$

We note that other multidimensional generalizations are possible (cf. [99], [241]); however, the present flow-decomposition approach is appealing in that it facilitates direct use of generally accepted functional forms for parallel flows, such as (15.3.7).

15.3.2 Implementation of the Flow-Field Model in Sundance

The use of the symbolic PDE package Sundance requires a weak form of the problem, which is then solved iteratively from an appropriately defined linearized form to determine the solution of the nonlinear

flow field. While Sundance can generate such a scheme internally, we choose instead to specify this aspect of the solution procedure explicitly. We thus first derive the exact weak form of the nonlinear problem, and then prescribe an efficient functional iteration scheme for solving, via Sundance, a convergent sequence of appropriately linearized approximations.

The weak formulation of the flow-field problem is derived as follows. We first rewrite (15.3.3) and (15.3.4) in vector form as

$$\overline{\nabla} \cdot \overline{U} = 0, \quad (15.3.11)$$

$$(\overline{U} \cdot \overline{\nabla}) \overline{U} + \frac{1}{\rho} \overline{\nabla} P - \nu \nabla^2 \overline{U} - \frac{1}{\rho} (\underline{\underline{\tau}} \cdot \overline{\nabla}) = 0, \quad (15.3.12)$$

where a double underlined quantity denotes a tensor (e.g., the Reynolds stress tensor $\underline{\underline{\tau}}$) and the arrow over the gradient operator denotes the location (left or right) of the object on which the corresponding operation is to be applied. Introducing test functions q and \overline{v} , we multiply (15.3.11) by q , (15.3.12) by \overline{v} , and integrate over the space Ω to obtain the weak forms

$$\int_{\Omega} q \overline{\nabla} \cdot \overline{U} \, d\Omega = 0, \quad (15.3.13)$$

$$\int_{\Omega} \overline{v} \cdot \left[(\overline{U} \cdot \overline{\nabla}) \overline{U} + \rho^{-1} \overline{\nabla} P - \nu \nabla^2 \overline{U} - \rho^{-1} (\underline{\underline{\tau}} \cdot \overline{\nabla}) \right] \, d\Omega = 0. \quad (15.3.14)$$

Equation (15.3.13) is satisfactory in its present state, but in order to eliminate second derivatives (Sundance requires at most first derivatives of the variables and test functions in the integrands) and to otherwise simplify (using boundary conditions) the weak form (15.3.14), we manipulate the various terms in (15.3.14) as follows. First, though not essential, we write

$$\begin{aligned} \int_{\Omega} (\overline{v} \cdot \overline{\nabla}) P \, d\Omega &= \int_{\Omega} \left[\overline{\nabla} \cdot (\overline{v} P) - P (\overline{\nabla} \cdot \overline{v}) \right] \, d\Omega \\ &= \int_{\partial\Omega} P (\overline{v} \cdot \overline{n}) \, d(\partial\Omega) - \int_{\Omega} P (\overline{\nabla} \cdot \overline{v}) \, d\Omega, \end{aligned} \quad (15.3.15)$$

where \overline{n} is the unit normal and the last term introduces a symmetry with respect to (15.3.11). Second, we remove explicit second derivatives according to

$$\begin{aligned} \int_{\Omega} \overline{v} \cdot (\nabla^2 \overline{U}) \, d\Omega &= \int_{\Omega} v_i \partial_j \partial_j U_i \, d\Omega \\ &= \int_{\Omega} [\partial_j (v_i \partial_j U_i) - (\partial_j v_i) (\partial_j U_i)] \, d\Omega \\ &= \int_{\partial\Omega} v_i (\overline{\nabla} U_i) \cdot \overline{n} \, d(\partial\Omega) - \int_{\Omega} (\overline{\nabla} v_i) \cdot (\overline{\nabla} U_i) \, d\Omega \\ &= \int_{\partial\Omega} \overline{v} \cdot (\overline{U} \overline{\nabla}) \cdot \overline{n} \, d(\partial\Omega) - \int_{\Omega} (\overline{v} \overline{\nabla}) : (\overline{U} \overline{\nabla}) \, d\Omega, \end{aligned} \quad (15.3.16)$$

where repeated indices imply summation, $\partial_j \equiv \partial/\partial x_j$ and the tensor contraction operator is defined by $\underline{\underline{a}} : \underline{\underline{b}} = a_{ij} b_{ij}$. Finally, since the Reynolds stress $\underline{\underline{\tau}}$ involves first derivatives of the flow variables, which

would thus lead to the appearance of second derivatives in Eq. (15.3.14), we rewrite the last term in (15.3.14) as

$$\begin{aligned}
\int_{\Omega} \vec{v} \cdot (\underline{\underline{\tau}} \cdot \overleftarrow{\nabla}) d\Omega &= \int_{\Omega} v_i \partial_j \tau_{ij} d\Omega \\
&= \int_{\Omega} [\partial_j (v_i \tau_{ij}) - \tau_{ij} (\partial_j v_i)] d\Omega \\
&= \int_{\partial\Omega} \vec{v} \cdot \underline{\underline{\tau}} \cdot \vec{n} d(\partial\Omega) - \int_{\Omega} \underline{\underline{\tau}} : (\vec{v} \overleftarrow{\nabla}) d\Omega.
\end{aligned} \tag{15.3.17}$$

Substituting (15.3.15) – (15.3.17) into (15.3.14), the weak form of momentum conservation thus becomes

$$\begin{aligned}
\int_{\Omega} \left[\vec{v} \cdot (\vec{U} \cdot \overleftarrow{\nabla}) \vec{U} - \rho^{-1} P (\overleftarrow{\nabla} \cdot \vec{v}) + \nu (\vec{v} \overleftarrow{\nabla}) : (\vec{U} \overleftarrow{\nabla}) + \rho^{-1} \underline{\underline{\tau}} : (\vec{v} \overleftarrow{\nabla}) \right] d\Omega \\
+ \int_{\partial\Omega} \left[\rho^{-1} P (\vec{v} \cdot \vec{n}) - \nu \vec{v} \cdot (\vec{U} \overleftarrow{\nabla}) \cdot \vec{n} - \rho^{-1} \vec{v} \cdot \underline{\underline{\tau}} \cdot \vec{n} \right] d(\partial\Omega) = 0.
\end{aligned} \tag{15.3.18}$$

The boundary integral in (15.3.18) can be simplified further by considering the boundary conditions [at inlet(s), outlet(s) and walls (including floor and ceilings)] for the problem of interest. At outlets, denoted by $\partial\Omega_o$, we assume no forcing, which is expressed as

$$\left[\rho^{-1} P \vec{n} - (\vec{U} \overleftarrow{\nabla}) \cdot \vec{n} - \rho^{-1} \underline{\underline{\tau}} \cdot \vec{n} \right]_{\partial\Omega_o} = 0. \tag{15.3.19}$$

Consequently, the condition (15.3.19) implies that the boundary integral in (15.3.18) vanishes on an outlet boundary $\partial\Omega_o$. At an inlet, denoted by $\partial\Omega_I$, or at walls, denoted by $\partial\Omega_w$, the boundary conditions for the present problem are given by $\vec{U} = \vec{U}_I$ and $\vec{U} = \vec{U}_w = 0$, respectively, where the latter corresponds to a no-slip, no-penetration condition at $\partial\Omega_w$. (It is also true, by definition, that the Reynolds stress tensor $\underline{\underline{\tau}} = 0$ on $\partial\Omega_w$.) However, since (15.3.19) involves only gradients of \vec{U} , we approximate these conditions as

$$-(\vec{U} \overleftarrow{\nabla}) \cdot \vec{n} \Big|_{\partial\Omega_{I,w}} = \varepsilon^{-1} (\vec{U} - \vec{U}_{I,w})_{\partial\Omega_{I,w}}, \quad \varepsilon \ll 1, \tag{15.3.20}$$

which implies $\|\vec{U} - \vec{U}_{I,w}\| \rightarrow 0$ on $\partial\Omega_{I,w}$ as $\varepsilon \rightarrow 0$. Substituting (15.3.19) and (15.3.20) into (15.3.18) and taking the limit $\varepsilon \rightarrow 0$ thus allows us, after invoking the assumption that the discrete basis functions used to represent the variables are nodal [148], to replace the boundary integral in the latter by a simple penalty term according to

$$\begin{aligned}
\int_{\partial\Omega} \left[\rho^{-1} P (\vec{v} \cdot \vec{n}) - \nu \vec{v} \cdot (\vec{U} \overleftarrow{\nabla}) \cdot \vec{n} - \rho^{-1} \vec{v} \cdot \underline{\underline{\tau}} \cdot \vec{n} \right] d(\partial\Omega) \\
\sim \int_{\partial\Omega} \vec{v} \cdot (\vec{U} - \vec{U}_{I,w}) d(\partial\Omega).
\end{aligned} \tag{15.3.21}$$

Hence, the final weak form of momentum conservation to be used in the present study is given by

$$\begin{aligned}
\int_{\Omega} \left[\vec{v} \cdot (\vec{U} \cdot \overleftarrow{\nabla}) \vec{U} - \rho^{-1} P (\overleftarrow{\nabla} \cdot \vec{v}) + \nu (\vec{v} \overleftarrow{\nabla}) : (\vec{U} \overleftarrow{\nabla}) + \rho^{-1} \underline{\underline{\tau}} : (\vec{v} \overleftarrow{\nabla}) \right] d\Omega \\
+ \int_{\partial\Omega_{I,w}} \vec{v} \cdot (\vec{U} - \vec{U}_{I,w}) d(\partial\Omega) = 0,
\end{aligned} \tag{15.3.22}$$

where the (shear) components of $\underline{\underline{\tau}} \approx \underline{\underline{\tau}}^s$ are given by (15.3.8) – (15.3.10).

We observe that (15.3.22) is nonlinear with respect to \vec{U} through the appearance of the factors $(\vec{U} \cdot \vec{\nabla}) \vec{U}$ and the Reynolds stress tensor $\underline{\underline{\tau}}$, where, from (15.3.8) and (15.3.9), the latter consists of the shear components $\tau_{12} = \tau_{21} = \rho \ell_1^2 |\partial \bar{U}_2 / \partial x_1| (\partial U_2 / \partial x_1) + \rho \ell_2^2 |\partial U_1 / \partial x_2| (\partial U_1 / \partial x_2)$. Consequently, since we ultimately require a weak form that is linear with respect to the dependent variables (as well as the test functions), we solve the coupled system (15.3.13) and (15.3.22) by an iterative approach. In particular, the nonlinear terms are formally linearized and functional iteration on the resultant linear problem is employed to obtain the solution to the original nonlinear problem. Thus, if the solution to the k th iterate \vec{U}_k is known, the next iterate \vec{U}_{k+1} is determined by solving a linear problem obtained from the original nonlinear equations by linearizing nonlinear terms about \vec{U}_k . In particular, making the *a priori* assumption that $|\vec{U}_{k+1} - \vec{U}_k|$ is sufficiently small, we have

$$\begin{aligned} (\vec{U}_{k+1} \cdot \vec{\nabla}) \vec{U}_{k+1} &= \left\{ [\vec{U}_k + (\vec{U}_{k+1} - \vec{U}_k)] \cdot \vec{\nabla} \right\} [\vec{U}_k + (\vec{U}_{k+1} - \vec{U}_k)] \\ &\approx (\vec{U}_k \cdot \vec{\nabla}) \vec{U}_k + [(\vec{U}_{k+1} - \vec{U}_k) \cdot \vec{\nabla}] \vec{U}_k + (\vec{U}_k \cdot \vec{\nabla}) (\vec{U}_{k+1} - \vec{U}_k) \\ &= (\vec{U}_{k+1} \cdot \vec{\nabla}) \vec{U}_k + (\vec{U}_k \cdot \vec{\nabla}) \vec{U}_{k+1} - (\vec{U}_k \cdot \vec{\nabla}) \vec{U}_k \end{aligned} \quad (15.3.23)$$

and, denoting $\partial U_i / \partial x_j$ by $U_{i,j}$,

$$\begin{aligned} |U_{i,j}^{(k+1)}| U_{i,j}^{(k+1)} &= [U_{i,j}^{(k+1)} U_{i,j}^{(k+1)}]^{1/2} U_{i,j}^{(k+1)} \\ &= \left\{ [U_{i,j}^{(k)} + (U_{i,j}^{(k+1)} - U_{i,j}^{(k)})] [U_{i,j}^{(k)} + (U_{i,j}^{(k+1)} - U_{i,j}^{(k)})] \right\}^{1/2} [U_{i,j}^{(k)} + (U_{i,j}^{(k+1)} - U_{i,j}^{(k)})] \\ &= \left\{ U_{i,j}^{(k)} U_{i,j}^{(k)} + 2(U_{i,j}^{(k+1)} - U_{i,j}^{(k)}) U_{i,j}^{(k)} + [U_{i,j}^{(k+1)} - U_{i,j}^{(k)}]^2 \right\}^{1/2} [U_{i,j}^{(k)} + (U_{i,j}^{(k+1)} - U_{i,j}^{(k)})] \\ &\approx [U_{i,j}^{(k)} U_{i,j}^{(k)}]^{1/2} \left\{ 1 + [U_{i,j}^{(k)}]^{-1} [U_{i,j}^{(k+1)} - U_{i,j}^{(k)}] \right\} \left\{ U_{i,j}^{(k)} + [U_{i,j}^{(k+1)} - U_{i,j}^{(k)}] \right\} \\ &\approx [U_{i,j}^{(k)} U_{i,j}^{(k)}]^{1/2} [2U_{i,j}^{(k+1)} - U_{i,j}^{(k)}]. \end{aligned} \quad (15.3.24)$$

Using these linearized representations in (15.3.22), we thus arrive at a functional iteration scheme that computes the successive approximations \vec{U}_{k+1} and P_{k+1} in terms of initial guesses \vec{U}_k and P_k according to

$$\int_{\Omega} q \vec{\nabla} \cdot \vec{U}_{k+1} d\Omega = 0, \quad (15.3.25)$$

$$\begin{aligned} \int_{\Omega} \left\{ \vec{v} \cdot [(\vec{U}_{k+1} \cdot \vec{\nabla}) \vec{U}_k + (\vec{U}_k \cdot \vec{\nabla}) \vec{U}_{k+1} - (\vec{U}_k \cdot \vec{\nabla}) \vec{U}_k] \right. \\ \left. - \rho^{-1} P_{k+1} (\vec{\nabla} \cdot \vec{v}) + \mathbf{v} (\vec{v} \cdot \vec{\nabla}) : (\vec{U}_{k+1} \cdot \vec{\nabla}) + \rho^{-1} \underline{\underline{\tau}}^l : (\vec{v} \cdot \vec{\nabla}) \right\} d\Omega \\ + \int_{\partial\Omega_{l,w}} \vec{v} \cdot (\vec{U} - \vec{U}_{l,w}) d(\partial\Omega) = 0, \end{aligned} \quad (15.3.26)$$

where $\underline{\underline{\tau}}^l$, whose components are given by $\tau_{ij}^l = \rho [U_{i,j}^{(k)} U_{i,j}^{(k)}]^{1/2} [2U_{i,j}^{(k+1)} - U_{i,j}^{(k)}] (1 - \delta_{ij})$, is the linearized form of $\underline{\underline{\tau}}$ based on (15.3.24). Equation (15.3.26) is linear with respect to the unknown function \vec{U}_{k+1} and can be handled directly by Sundance.

The above sequence of approximations is expected to be convergent provided the k th iterate is a sufficiently good approximation to the actual solution of the nonlinear problem. In practice, this generally requires that the iteration scheme defined by (15.3.25) and (15.3.26) be embedded in an outer iterative loop with respect to increasing values of a suitably defined Reynolds number $U^* L^* / \nu$, where L^* and U^* are appropriately defined length and time scales. Indeed, for a sufficiently small initial value of the Reynolds number, the linear terms dominate and convergence is therefore more likely to be achieved in that case. The resulting converged solution for such a given Reynolds number then provides a good starting guess for the next sequence of iterations at a modestly larger value of that parameter, and so forth. It is also found, consistent with the need to resolve boundary layers and other finer aspects of the flow field, that the ability to achieve convergence at a given Reynolds number depends on the discretization (mesh). In particular, larger Reynolds numbers are generally found to require finer discretizations, especially in the vicinity of boundaries, independent of the goodness of the previous iterate with respect to the true solution.

15.3.3 The Eikonal Equation and its Regularization

The turbulence model described in section 15.3.1 requires knowing the wall distance ℓ^* introduced in the expression (15.3.10) for the mixing length ℓ . The wall distance function can, in principle, be computed exactly given knowledge of the geometry of the walls; however, this computation is tedious and must be repeated for each new geometry considered. To simplify and automate the computation of ℓ^* , we use the eikonal equation of geometrical optics (e.g., [49]) which, with appropriate boundary conditions, has the wall distance ℓ^* as a solution. We can thus bypass tedious computational geometry and obtain the wall function as the solution of a PDE.

15.3.4 Pressure Stabilization

It is well known that representing velocity and pressure with basis functions of the same order tends to be unstable (cf. [102]). Consequently, various stabilization schemes have been proposed to allow these variables to be represented with the same set of basis functions. One popular approach is to augment the continuity equation with a (small) term proportional to the Laplacian of pressure according to

$$\vec{\nabla} \cdot \vec{U} = \beta h^2 \nabla^2 P, \quad (15.3.27)$$

where h is the linear dimension (diameter) of the finite element discretization and β is an optimally chosen small parameter (cf. [102]). Aside from allowing equal-order interpolants, the added term has the positive effect of removing the indefiniteness of the discretized linear system and smoothing out numerical oscillations in the pressure variable.

Multiplication of (15.3.27) by the test function q followed by an integration over the domain Ω then leads

to the weak form

$$\int_{\Omega} \left[q \vec{\nabla} \cdot \vec{U} + \beta h^2 (\vec{\nabla} q) \cdot (\vec{\nabla} P) \right] d\Omega - \int_{\partial\Omega} \beta h^2 (q \vec{\nabla} P) \cdot \vec{n} d(\partial\Omega) = 0, \quad (15.3.28)$$

where we have used the identity $q \nabla^2 P = \vec{\nabla} \cdot (q \vec{\nabla} P) - (\vec{\nabla} q) \cdot (\vec{\nabla} P)$ and applied the divergence theorem to obtain the form (15.3.28). Dropping the boundary term, which implies that we are actually only regularizing the weak form (15.3.28) rather than (15.3.27), thus leads to

$$\int_{\Omega} \left[q \vec{\nabla} \cdot \vec{U} + \beta h^2 (\vec{\nabla} q) \cdot (\vec{\nabla} P) \right] d\Omega = 0 \quad (15.3.29)$$

and

$$\int_{\Omega} \left[q \vec{\nabla} \cdot \vec{U}_{k+1} + \beta h^2 (\vec{\nabla} q) \cdot (\vec{\nabla} P_{k+1}) \right] d\Omega = 0 \quad (15.3.30)$$

in place of (15.3.13) and (15.3.25), respectively.

15.4 The Optimization Problem

In this section we consider the steady-state transport of the toxin in the flow field developed in the preceding section. We thus make the implicit assumption that the toxin concentration is insufficient to affect the flow field itself, but that the transport influences of the flow clearly play a key role in determining the distribution of the toxin. After first formulating the toxin-transport model, the optimization problem is then constructed so as to recover the source location(s) from specific concentration data, where the latter consist of readings obtained from sensors placed strategically throughout the building. Prior to that phase of the calculation, we consider several possible optimization formulations before settling on the one used in our numerical experiments.

15.4.1 The Toxin Transport Model

Given the time-averaged flow field \hat{u} calculated in section 15.3, we can write the corresponding time-averaged continuity equation describing the transport of the toxin as

$$k \nabla^2 c - (\hat{u} \cdot \vec{\nabla}) c + \vec{\nabla} \cdot \vec{\mathcal{J}} + s = 0, \quad (15.4.1)$$

where $c(\vec{x})$ and $s(\vec{x})$ are, respectively, the (time-averaged) concentration and source fields at a point $\vec{x} \in \Omega$, k is the binary mass diffusivity (assumed constant) of the toxin with respect to the mixture, and $\vec{\mathcal{J}}$ is the turbulent mass-flux vector whose components are defined as $\mathcal{J}_i = -\overline{u_i' c'}$. The derivation of an expression for \mathcal{J} in terms of time-averaged quantities follows in an analogous fashion to the derivation of the expression for the Reynolds stress tensor in section 15.3.1. In particular, by analogy with Fick's law of diffusion, it is logical to write, at least for nearly parallel flows $\vec{U} \sim (U_1(x_2), 0)$,

$$\vec{\mathcal{J}} \sim (0, \mathcal{J}_2), \quad \mathcal{J}_2 = k^t \frac{\partial c}{\partial x_2}, \quad k^t = \ell^2 \left| \frac{\partial U_1}{\partial x_2} \right|, \quad (15.4.2)$$

where k^i is the turbulent, or eddy, diffusivity and ℓ is the same mixing length that was defined by (15.3.6) and (15.3.7) (cf. [42]). This expression may be extended to multidimensional flows using somewhat the same reasoning as that which led to (15.3.8) and (15.3.9). Hence, we define

$$J_j = k_j^i \frac{\partial c}{\partial x_j}, \quad k_j^i = \ell_j^2 \left| \frac{\partial U_i}{\partial x_j} \right|, \quad i \neq j, \quad (15.4.3)$$

where the ℓ_j are the same as those introduced in (15.3.9) and, for our present purposes, given by the single expression for $\ell_j \equiv \ell$ in (15.3.10).

As in the flow-field problem, a corresponding reduction of the weak form of the toxin continuity equation (15.4.1) is obtained by first multiplying that equation by a scalar test function r and integrating. This gives

$$\int_{\Omega} r \left[(\vec{U} \cdot \vec{\nabla})c - k \nabla^2 c - \vec{\nabla} \cdot \vec{J} - s \right] d\Omega = 0, \quad (15.4.4)$$

where the components J_j of \vec{J} are given in terms of $\partial c / \partial x_j$ according to (15.4.3). As in the case of the Navier-Stokes equations, we manipulate the individual terms appearing in (15.4.4) so as to eliminate second derivatives. In particular, by analogy with (15.3.16), we have

$$\begin{aligned} \int_{\Omega} r \nabla^2 c d\Omega &= \int_{\Omega} r \partial_j \partial_j c d\Omega \\ &= \int_{\Omega} [\partial_j (r \partial_j c) - (\partial_j r) (\partial_j c)] d\Omega \\ &= \int_{\partial\Omega} r (\vec{\nabla} c) \cdot \vec{n} d(\partial\Omega) - \int_{\Omega} (\vec{\nabla} r) \cdot (\vec{\nabla} c) d\Omega, \end{aligned} \quad (15.4.5)$$

and, since \vec{J}^i depends on derivatives of c , we also write, analogous to (15.3.17),

$$\begin{aligned} \int_{\Omega} r \vec{\nabla} \cdot \vec{J} d\Omega &= \int_{\Omega} [\vec{\nabla} \cdot (r \vec{J}) - \vec{J} \cdot (\vec{\nabla} r)] d\Omega \\ &= \int_{\partial\Omega} r \vec{J} \cdot \vec{n} d(\partial\Omega) - \int_{\Omega} \vec{J} \cdot (\vec{\nabla} r) d\Omega. \end{aligned} \quad (15.4.6)$$

Finally, since $\vec{\nabla} \cdot \vec{U} = 0$,

$$\begin{aligned} \int_{\Omega} r (\vec{U} \cdot \vec{\nabla})c d\Omega &= \int_{\Omega} r \vec{\nabla} \cdot (c \vec{U}) d\Omega \\ &= \int_{\Omega} [\vec{\nabla} \cdot (rc \vec{U}) - c \vec{U} \cdot (\vec{\nabla} r)] d\Omega \\ &= \int_{\partial\Omega} (rc \vec{U}) \cdot \vec{n} d(\partial\Omega) - \int_{\Omega} c \vec{U} \cdot (\vec{\nabla} r) d\Omega. \end{aligned} \quad (15.4.7)$$

Thus, in place of (15.4.4), the weak form of toxin mass conservation may be written as

$$\begin{aligned} \int_{\Omega} \left\{ (\vec{\nabla} r) \cdot [k \vec{\nabla} c - c \vec{U} + \vec{J}] - rs \right\} d\Omega \\ - \int_{\partial\Omega} r (k \vec{\nabla} c - c \vec{U} + \vec{J}) \cdot \vec{n} d(\partial\Omega) = 0. \end{aligned} \quad (15.4.8)$$

As before, the boundary integral in (15.4.8) can be simplified by applying the boundary conditions. In particular, the assumption of no mass transport across the walls implies that the integrand in the boundary integral vanishes on $\partial\Omega_w$ since the mixing lengths ℓ_j , and hence \vec{j} , also vanish there. At the inlet and outlet boundaries $\partial\Omega_I$ and $\partial\Omega_o$, we neglect the contribution of \vec{j} because the mixing length ℓ is likely to be small along most of such boundaries. In addition, at the inlet, we specify the incoming flux fraction (or concentration) of toxin α , which implies the set of conditions

$$\left[(c\vec{U}_I - k\vec{\nabla}c) \cdot \vec{n} \right]_{\partial\Omega_I} = -\alpha\vec{U}_I \cdot \vec{n}, \quad (15.4.9)$$

where \vec{n} always denotes the outward-facing normal at the boundary. At the outflow boundary, we represent the boundary condition in the general form

$$\left[(\vec{\nabla}c) \cdot \vec{n} + \beta c \right]_{\partial\Omega_o} = 0, \quad (15.4.10)$$

where $\beta > 0$ is a loss coefficient; the choice $\beta = 0$ would correspond to a finite-boundary approximation if the actual outlet were at infinity. Applying these conditions to (15.4.8) then yields the final weak form of species mass conservation as

$$\begin{aligned} \int_{\Omega} \left\{ (\vec{\nabla}r) \cdot [k\vec{\nabla}c - c\vec{U} + \vec{j}] - rs \right\} d\Omega - \int_{\partial\Omega_I} r\alpha_i \vec{U}_I \cdot \vec{n} d(\partial\Omega) \\ + \int_{\partial\Omega_o} rc(\vec{U} \cdot \vec{n} + k\beta) d(\partial\Omega) = 0. \end{aligned} \quad (15.4.11)$$

We note that in the event that there is no incoming flux of toxin, $\alpha = 0$ and the first boundary integral in (15.4.11) vanishes.

15.4.2 The Source-Inversion Optimization Problem

We can now form the preliminary version of the optimization problem. Suppose that we have data c_i^* that is the concentration reading from sensor i . Then we seek the source $s(x)$ that creates a calculated concentration field that best matches this data. That is, we define

$$f_p(s, c) = \frac{1}{2} \sum_{i=1}^{N_s} (c(x_i) - c_i^*)^2,$$

where N_s is the number of sources. This is the L_2 measure of the discrepancy. We then write the first version of the optimization problem as

$$\begin{aligned} & \underset{c, s}{\text{minimize}} && f_p(c, s) \\ & \text{subject to:} && (15.4.11). \end{aligned} \quad (15.4.12)$$

There are several points that need to be addressed before we have a problem that we can attempt to solve.

First, there are several ways to consider for the handling of the source term in (15.4.11). One way is to assume a model for the sources. For example, we could assume a Gaussian model for source i , i.e.,

$$s_i(x) = \alpha_i e^{\beta_i(x-x_i)^2},$$

where α_i and β_i are constants and $x_i \in \Omega$ is the location. Then the source field is

$$s(x) = \sum_{i=1}^S s_i(x),$$

where S is the number of sources. Alternatively, we could consider source i to be a δ -function with a specified strength. In both of these cases, the number of sources is not known in advance and must be determined as part of the source inversion problem. Also, it is clear that some of the parameters enter the source term nonlinearly. The advantage of these models, however, is that there is a small number of parameters to estimate.

Another way to proceed is to leave $s(x)$ as simply a function over all of Ω to be determined. The two major advantages of this approach are that the function s enters (15.4.11) linearly and there is no need to know in advance how many sources there are. The disadvantage is that the number of parameters to be estimated is the number of nodes, a potentially large number. We believe, however, that the advantages of this approach outweigh the disadvantages and we thus choose this form for the source term.

Second, given the form of the source term, there is a need to regularize the problem. As posed, the problem is underdetermined; there are many source functions that will make $f_p = 0$, but they will be excessively oscillatory. We therefore need to smooth the solution since we expect s to be essentially zero except where real sources exist. Thus we use standard Tikhonov regularization, i.e., we modify the objective function f_p to be of the form

$$f(c, s) = f_p + \frac{1}{2} \sigma \int_{\Omega} (\nabla s)^2.$$

Our modified form of the optimization problem becomes

$$\begin{aligned} & \underset{c, s}{\text{minimize}} && f(c, s) \\ & \text{subject to:} && (15.4.11). \end{aligned} \tag{15.4.13}$$

As posed, problem (15.4.13) is an equality-constrained quadratic programming problem. It is well known that, when discretized, such quadratic programs (QP) can be solved by solving the linear system that arises from writing the first order necessary, or KKT, conditions. We point out that Sundance allows the creation of discrete functions defined on the mesh. Thus we can easily define the objective function $f(c, s)$ and then create the Lagrangian

$$L(c, s, \lambda) = f(c, s) + \int_{\Omega} \lambda h(c, s),$$

where λ is the Lagrange multiplier and $h(c, s)$ is the left hand side of (15.4.11). In Sundance we can define an expression that is the variation of $L(c, s, \lambda)$ with respect to its arguments, i.e., the first order conditions. We can then create a Sundance problem, specify a solver (and a preconditioner), and solve the system. This procedure takes fewer than twenty lines of code.

We need to consider, however, the possibility of adding additional constraints to problem (15.4.13). In particular, we know that both the source field and the concentration field must be nonnegative. In the case

of (15.4.11), it is easy to see that if $s \geq 0$ then it follows that $c \geq 0$. Thus the final form of the optimization problem that we consider is

$$\begin{aligned} & \underset{c, s}{\text{minimize}} && f(c, s) \\ & \text{subject to:} && (15.4.11) \\ & && s \geq 0. \end{aligned} \tag{15.4.14}$$

Sundance can also be used to create this problem. Let

$$a^t x + \frac{1}{2} x^t Q x \tag{15.4.15}$$

be the general form of a quadratic objective function where $a \in \mathcal{R}^n$ and Q is an $(n \times n)$ symmetric matrix. We create a Sundance problem for the function given by $f(c, s)$. Then by getting the linear operator for this problem, we have the matrix Q and by getting the right hand side we have a . Similarly, we can then get the matrix and right hand side associated with the Sundance problem corresponding to (15.4.11), thus obtaining the discretized linear equality constraints. It is then trivial in TSF to create the identity matrix (never actually formed) and vector of zeros corresponding to the constraints $s \geq 0$. These matrices and vectors can be passed to any appropriate QP solver.

We now make some remarks about problem (15.4.14) that motivate our choice of QP solver.

1. Recall that we are not interested in high accuracy solutions.
2. With this formulation, there will be many near-active constraints, i.e., many points with very small, but positive, function values; such problems often create severe computational difficulties.
3. We expect that our formulation, which uses the Tikhonov regularization term, will tend to smear the constraints. Thus we are not concerned with getting the correct active set.
4. Given that we do not care about getting the correct active set, we are certainly not interested in getting the Lagrange multipliers.
5. Since the equality constraint given by (15.4.11) will be a discretized PDE in the QP, we do not need to worry about satisfying these constraints to high accuracy.

The above observations lead us to consider the use of *O3D* to solve the inequality-constrained problem (15.4.14). *O3D* is a primal, interior-point method that can be controlled so that estimates of the multipliers are not computed. It operates on a QP with the objective function given by (15.4.15) and with only inequality constraints of the form

$$Ax + b \leq 0.$$

In essence, *O3D*, which stands for Optimizing over 3-Dimensional subspaces, is an iterative method that at each iteration creates a 3-dimensional subspace, solves the quadratic program restricted to that subspace, and moves 99% of the distance to the boundary in the direction implied by the solution. The three directions, p_i , all satisfy a linear system of the form

$$(A^t D^2 A + Q/\gamma) p_i = t_i, \quad i = 1, 2, 3$$

where

$$D = \text{diag} \{1/(Ax + b)_i\},$$

γ is a scalar computed at each iteration, and t_i is an appropriate right hand side. There are, of course, many other details; see [46] for a more complete description.

In the tests reported in the next section, we tried several formulations of the problem in an attempt to find an efficient strategy. The original strategy for handling equality constraints in *O3D* is to convert them to a pair of equality constraints and then to form a “big M phase 1 problem.” This problem has an artificial variable that is forced to zero as solution is neared. In the limit, this forces the two inequalities corresponding to an equality constraint to become equal, thus satisfying the equality constraint. Although this has often worked well in practice, it was not efficient enough for our purposes here; we thus looked for a more efficient formulation. Although not often recommended, the best formulation that we found was to create a penalty form of (15.4.14) so that we have a problem with only inequality constraints. This formulation allows us to control the tolerance to which the equality constraints will be satisfied. As noted above, it does not make sense to require satisfaction of the equality constraint to high accuracy since it is a finite element approximation to (15.4.11). The penalty form that we consider is given by

$$\begin{aligned} & \underset{c, s}{\text{minimize}} && f(c, s) + \frac{1}{2}\rho \|h(c, s)\|^2 \\ & \text{subject to:} && s \geq 0. \end{aligned} \tag{15.4.16}$$

We choose an increasing sequence ρ_i and solve (15.4.16) for ρ_i , $i = 1, \dots$, for a relaxed set of convergence criteria. After each solution, we check the value of $\|h(c, s)\|$. If the value is less than ϵ , where ϵ is a given tolerance, then we fix ρ at the current value and solve (15.4.16) to a tighter set of convergence criteria. We note that in this form it is easy to get a good interior starting approximation by solving the problem (15.4.13) for s (by solving one linear system) and then modifying any value of $s \leq 0$ to be some small positive value. As reported in the next section, this allowed convergence to acceptable accuracy in a very small number of iterations.

15.5 Numerical Results

To conduct our numerical tests, it was first necessary to solve the flow model developed in section 15.3. We therefore first describe a simple 2-dimensional room with one inlet and one outlet and display the resulting flow field. After commenting on some of the realistic features of this field, we then show the source-inversion results that were obtained when sources and sensors are placed in the room.

15.5.1 Sample 2-Dimensional Problem Geometry and Flow Field

As indicated above, a simple two-dimensional room was constructed with an inlet and an outlet as shown by the domain in Figure 15.1. The main part of the room is 15×20 units, with two small (2×2) inlet/outlet sections on the left and right, respectively. Various triangular meshes for this geometry were created with Shewchuk’s mesh-generating package, Triangle ([211], [212]), with two layers of additional nodes specified along the boundaries to better resolve the (turbulent) boundary-layer structure in the vicinity of

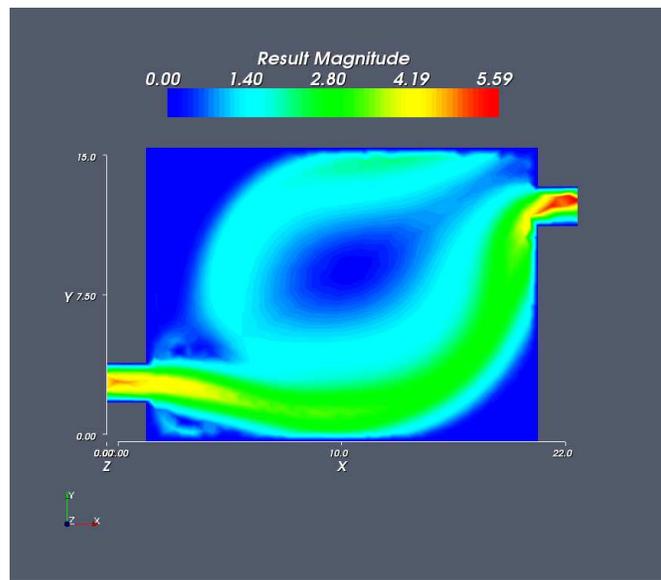


Figure 15.1. Calculated vector flow field for $Re = 10000$.

the walls. In all, the flow field used in the numerical calculations described below was generated by Sundance on a roughly 15,000-node mesh with approximately 31,000 triangles. This degree of mesh resolution enabled us to calculate, using the iterative procedure described in section 3, a flow field corresponding to fairly high Reynolds numbers (the source-inversion calculations described below corresponded to $Re = 10000$). Generally speaking, the ability to iterate to increasingly large Reynolds numbers required increasingly fine meshes.

As suggested in section 15.3.2, the specific procedure we successfully employed, given a sufficiently fine mesh, was to start with zero initial data for the velocity and pressure fields and iterate to a converged solution for $Re = 100$. That solution was then used as initial data for the $Re = 200$ problem, and so forth in increments of 100 in the Reynolds number. Since (15.3.24) represents a formal linearization of the flow-field model based on Taylor expansions of the nonlinear terms, roughly quadratic convergence was achieved at each step in the Reynolds-number loop. Typically, only four or five iterations were required to meet our fairly stringent convergence criteria, although somewhat more (about 10) were usually needed on the first $Re = 100$ step due to our starting the iteration from zero initial data. A convergence failure at a particular Reynolds number generally implied an insufficiently fine mesh, and could be remedied by redoing the calculation on a finer mesh.

The inlet velocity profile was specified to be Poiseuille flow with the peak inlet flow velocity at the center of the profile taken to be 4. The resulting vector flow field for $Re = 10000$ is shown in Figure 15.1. It is readily observed that the main flow direction is from the inlet to the outlet. A more careful look, however, reveals several recirculation zones and other structures that are indicative of a realistic flow field.

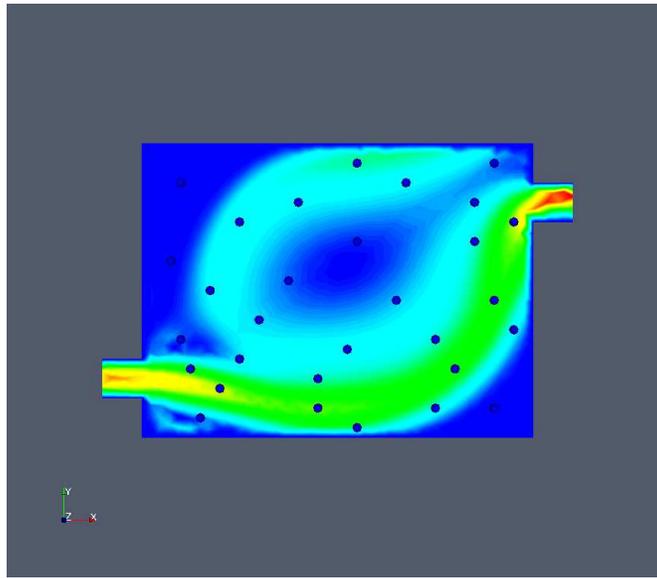


Figure 15.2. Flow field with 30 irregularly spaced sensors

15.5.2 Source Inversion Tests

To test our source-inversion procedure we first generate some data on the 2-D example above. We do this by assuming a Gaussian source as described in section 15.4 and solving the forward problem, i.e., we solve (15.4.11). In particular, we take our sources to be

$$s(x) = \alpha_i e^{\beta_i(x-x_i)^2}$$

where $\alpha_i = 20$ and $\beta_i = 2$ for all i . We then specify the location for a set of sensors in the room, get the concentration c at each sensor location, and modify it by a random value of $\pm 5\%$. (The value of 5% was chosen in consultation with one of our Sandia National Laboratory colleagues who is working on the design of the sensors, [78]). We investigated two patterns of the locations for the sensors. The first was a regular pattern of 9 sensor locations; the second was a hand-selected pattern of 30 locations shown in figure 15.2. This second pattern will be discussed further later.

The results for the 9-sensor source inversion problems are as follows. First, it was sometimes possible to achieve reasonable predictions, even without using the inequality constraints, but it was very easy to pick source locations for which the prediction of both the number and location of the sources was poor. Without using inequality constraints, we often observed negative values of the source field, but even when we used the inequality constraints, the results were still poor. We conclude from this that 9 sensors on a regular grid are not sufficient to reconstruct the sources for many source locations.

To determine an effective sensor arrangement, it is necessary to consider how the toxin is transported throughout the building and how this information is represented by the sensor readings. Here, we will look

at two different cases. First, suppose a source is located in a region of the building in which the air flow is minimal, e.g., a corner. Over time the toxin will accumulate and any sensor in that region will exhibit a high concentration reading while all other sensors return comparatively low readings, assuming only one source exists. Thus, given at least one sensor in this region, any solution would exhibit a source in the proper area of the building. Since any sensor in this location will eventually have a comparatively high concentration reading, we require only one sensor to adequately determine sources in these types of regions. For the second case, suppose we have a source in the main stream of the flow. Instead of accumulating around the source location, the toxin will be distributed throughout the building as dictated by the flow field. Thus, if the main stream of the flow contains too few sensors, as in the 9-sensor arrangement, many different source locations could result in the same set of concentration readings. Therefore, in the main stream of the flow it is necessary to place sensors based on both the direction and the magnitude of the flow.

Using this knowledge, we picked sensor locations by hand by conducting many experiments. We make no claims, however, that these are the optimal sensor locations, a topic to which we return in section 15.6. With these locations, we were able to locate any pair of sources with acceptable accuracy.

We can draw several conclusions from these results.

1. 30 sensors usually gets acceptable accuracy.
2. Adding inequality constraints sometimes helps significantly by reducing nonphysical oscillations of the source field and is not computationally too expensive. In fact, it generally took fewer than 10 iterations of $O3D$ to solve the problem to acceptable accuracy.
3. Experience and knowledge of the flow field helps to predict locations of the sources.
4. As expected by our use of the Tikhonov regularization, the sources are smeared, especially for sources placed in the main stream of the flow. Nevertheless, the peaks in the computed source field correspond reasonably well with the true source locations.
5. We could do better with more sensors, but we want to restrict the number due to practical considerations: In a real 3-dimensional building, we will not be able to have a high concentration of sensors with complete freedom of placement. On the other hand, the sources cannot be placed arbitrarily either.
6. The reconstructed concentration field is much better than the source field; this implies that we have an ill-conditioned problem.
7. We also ran these problems with 10% and 15% error with progressively worse, but not horrible, results.

15.5.3 Coarse Meshes

In the results reported above, we showed that we were, indeed, able to reconstruct the source or sources with acceptable accuracy. In this section, we consider the issue of doing this rapidly. In particular, we discuss the possibility of using a coarser mesh to reconstruct the source field than was necessary to

compute the flow field. Obviously, if a coarser mesh suffices, the time to solve the required linear systems will decrease.

Two possibilities present themselves for creating a coarser mesh. As noted above, we solved for the flow field on a fine, uniform mesh that was necessary to achieve convergence for the Reynolds number we desired. Thus, the first way to generate a coarser mesh is simply to regenerate the mesh with a larger value of the parameter that controls the size of the elements. Having done this, we can then interpolate the flow field from the solution on the fine mesh onto this coarser mesh and proceed with the source inversion. The second way to proceed is to try to generate a coarser mesh that is adapted to the flow field itself so that larger elements will appear in regions where the flow field is not changing rapidly and smaller elements where the flow field is rapidly varying. Fortunately, the meshing tool that we are using allows us to do this relatively easily. A few words about this tool are in order.

We have installed and used the Bidirectional Anisotropic Mesh Generator (BAMG) that is being developed at INRIA (see [111]). This code allows the generation of meshes over a 2-dimensional domain by specifying a number of parameters. These parameters control properties such as maximum and minimum edge length, the maximum number of triangles, etc., to generate uniform meshes. More interestingly, it also allows the specification of a “metric” field and will attempt to adapt the mesh to that field. Thus if we compute the flow field on a fine mesh and use this as the metric field, BAMG will produce a mesh that is adapted to this field. The other parameters are also used so that by specifying the minimum and maximum edge lengths along with this metric file, BAMG will produce a coarser mesh that is adapted to the flow field. BAMG will then compute a Lagrangian interpolation of the fine flow field values onto the coarse meshes. BAMG is currently being extended to 3-dimensions.

Using BAMG we were easily able to generate both the uniform and adapted coarse meshes necessary to run our source inversion method on these problems. We were (pleasantly) surprised by the amount of coarsening that was possible without noticing any degradation in the accuracy of the reconstructed sources. Indeed, we could not see any difference in the predicted source locations by reducing the number of triangles from the original 31,000 to approximately 1500, a factor of over 20! Recall that a grid of 1500 triangles would not be nearly fine enough to compute the flow field. Some representative results are shown in figures 15.3 – 15.7.

These figures show the results of using ≈ 1500 triangles, ≈ 1200 triangles, and ≈ 500 triangles both for adaptive and uniform meshes. The cases pictured generally show situations where there were some problems. In many of the cases not pictured here, we were able to obtain acceptable results to the lowest number of triangles. Overall, our experience suggests that the adaptive meshes yield better results for the coarsest meshes. There were certainly cases in which the adaptive meshes did not perform better. For example, figure 15.5 shows that the adaptive mesh predicts two sources at the coarsest level whereas the uniform mesh correctly predicts only one.

For some source configurations, it was not possible to get acceptable predictions at the coarsest levels for either mesh strategy; see, e.g., figure 15.7. In this figure, we note that the adaptive mesh was able to achieve acceptable results using approximately 1200 triangles whereas the uniform mesh incorrectly predicts three sources. In figure 15.6 we also see that the adaptive mesh performs better at the coarser levels.

We observe that sources placed in the main flow are much harder to locate due to the rapid dispersal of the

agent in this region of the flow field. It is also true that the predicted strengths in these regions is also much less than the true value. Again, we think that that the use of the Tikhonov regularization accounts for some of this.

Finally, we point out that the times required for doing the inversion are, as expected, substantially reduced by using these coarser meshes. Table 15.8 gives the the number of *O3D* iterations necessary to achieve the required accuracy and the number of seconds it took to run. Clearly the reduction of run-times by a factor of 40–100 will be significant.

15.6 Discussion and Conclusions

In this chapter we have developed a formulation for the source inversion problem that we have shown to be effective for locating sources in a steady-state environment. Although we have not stressed computational efficiency in our numerical experiments, we have been able to solve these problems relatively quickly, in terms of the number of iterations of *O3D*. Much work remains, however, to improve the efficiency of the linear solvers, the main source of computational costs in our runs. We have also shown that Sundance, along with TSF are powerful tools for the rapid prototyping and testing of various ideas and strategies.

In addition to continuing to work on the linear solver strategies, both in serial and in parallel, there are several other topics that need to be investigated.

1. As noted above, problem 15.4.14 uses a standard Tikhonov regularization scheme. Our computational results indicate that this tends to smear the source. Other possible regularizations can be considered, as in [18].
2. The problem of the optimal location of sensors is difficult. It is not at all clear what the objective should be. Our understanding is that different toxins dictate different attack strategies. Thus determining the location of the sensors must take that into account. We are pursuing ideas related to hierarchical control (see, e.g., [45]); this will be the subject of a future work. We are also considering ideas related to the mesh strategies noted above where sensors could be located in regions of rapid change of the flow field.

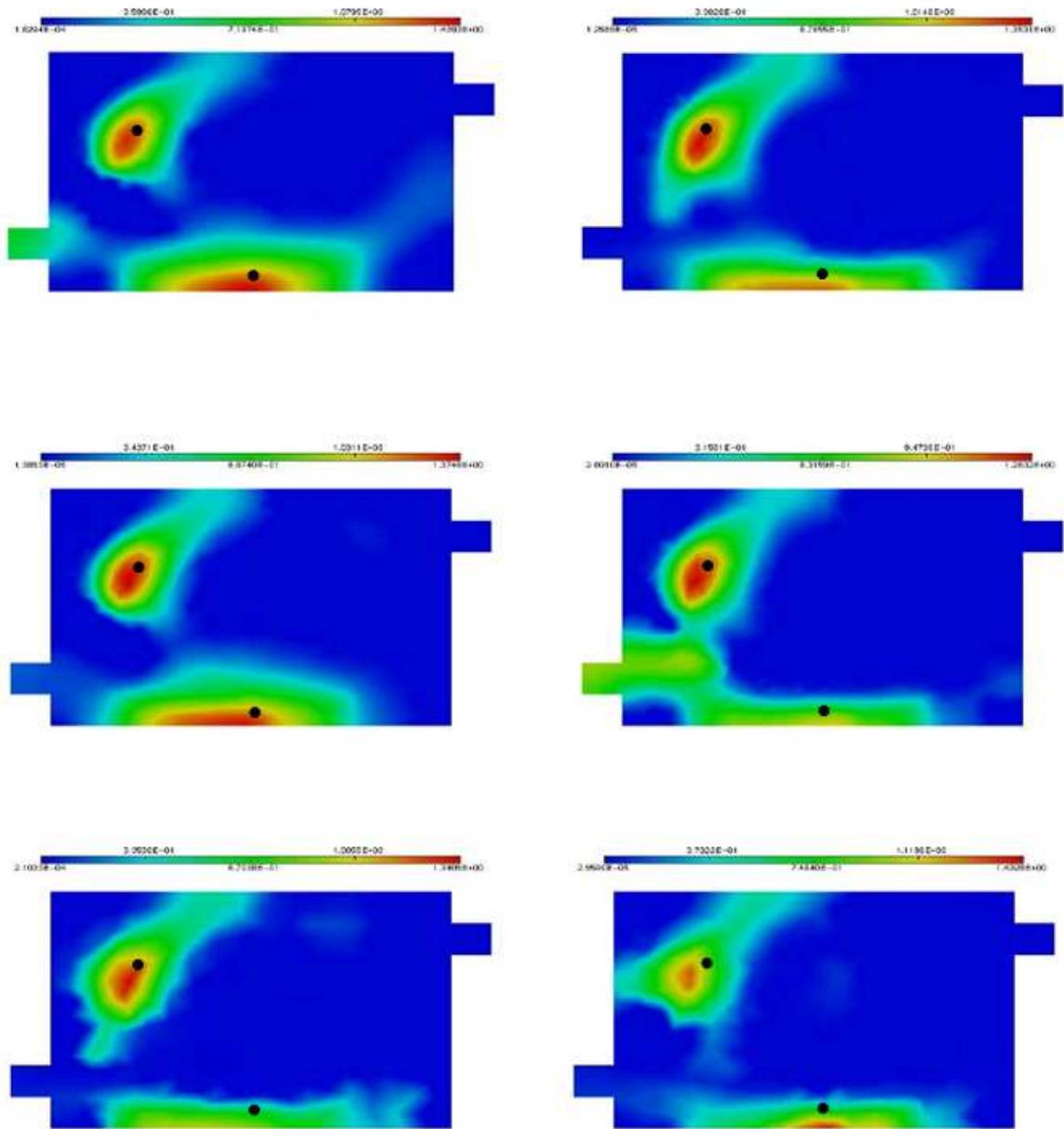


Figure 15.3. Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source locations are the points (10.0, 1.0) and (4.5, 10.0)

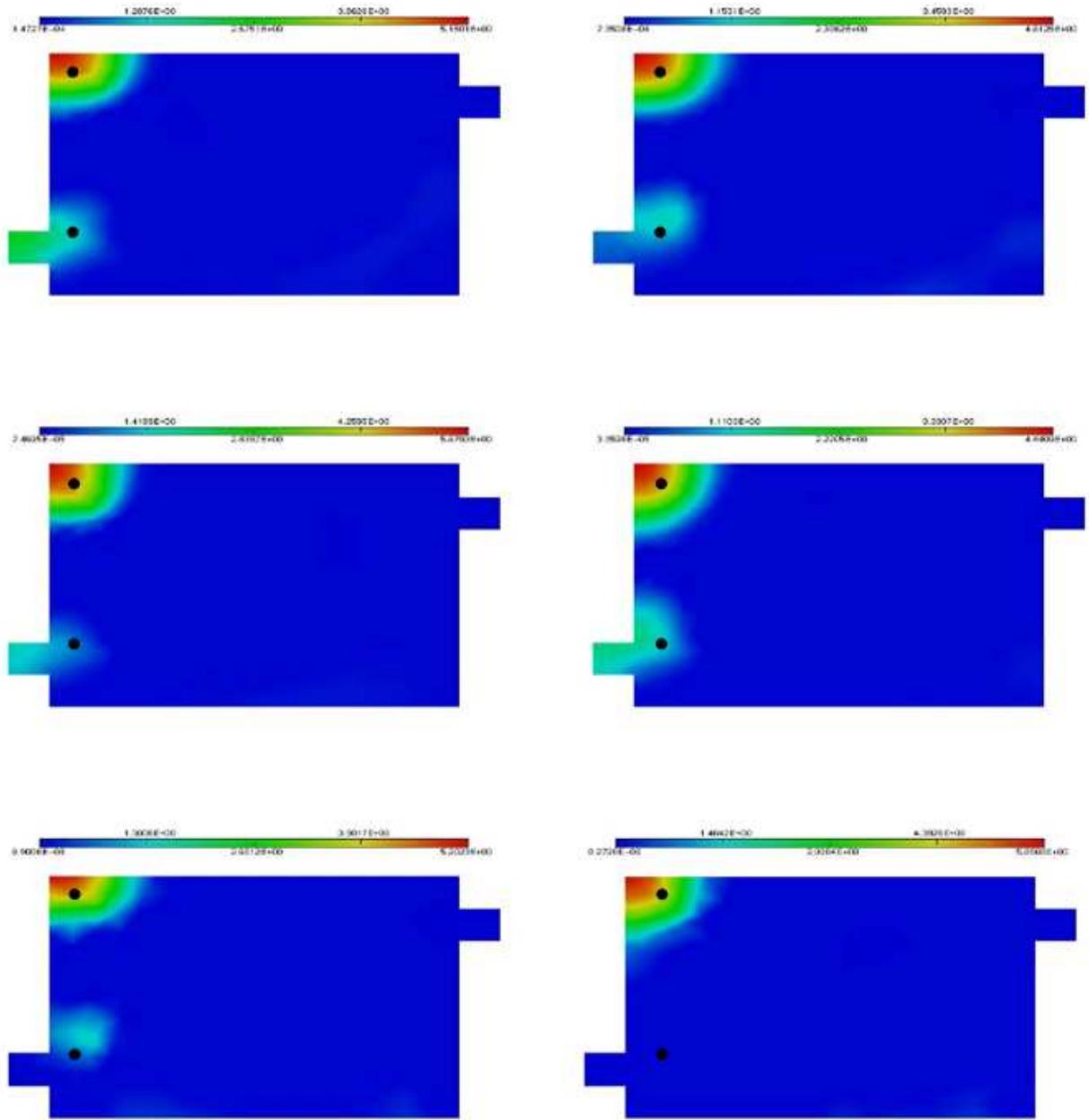


Figure 15.4. Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source locations are the points $(1.0, 14.0)$ and $(1.0, 4.0)$

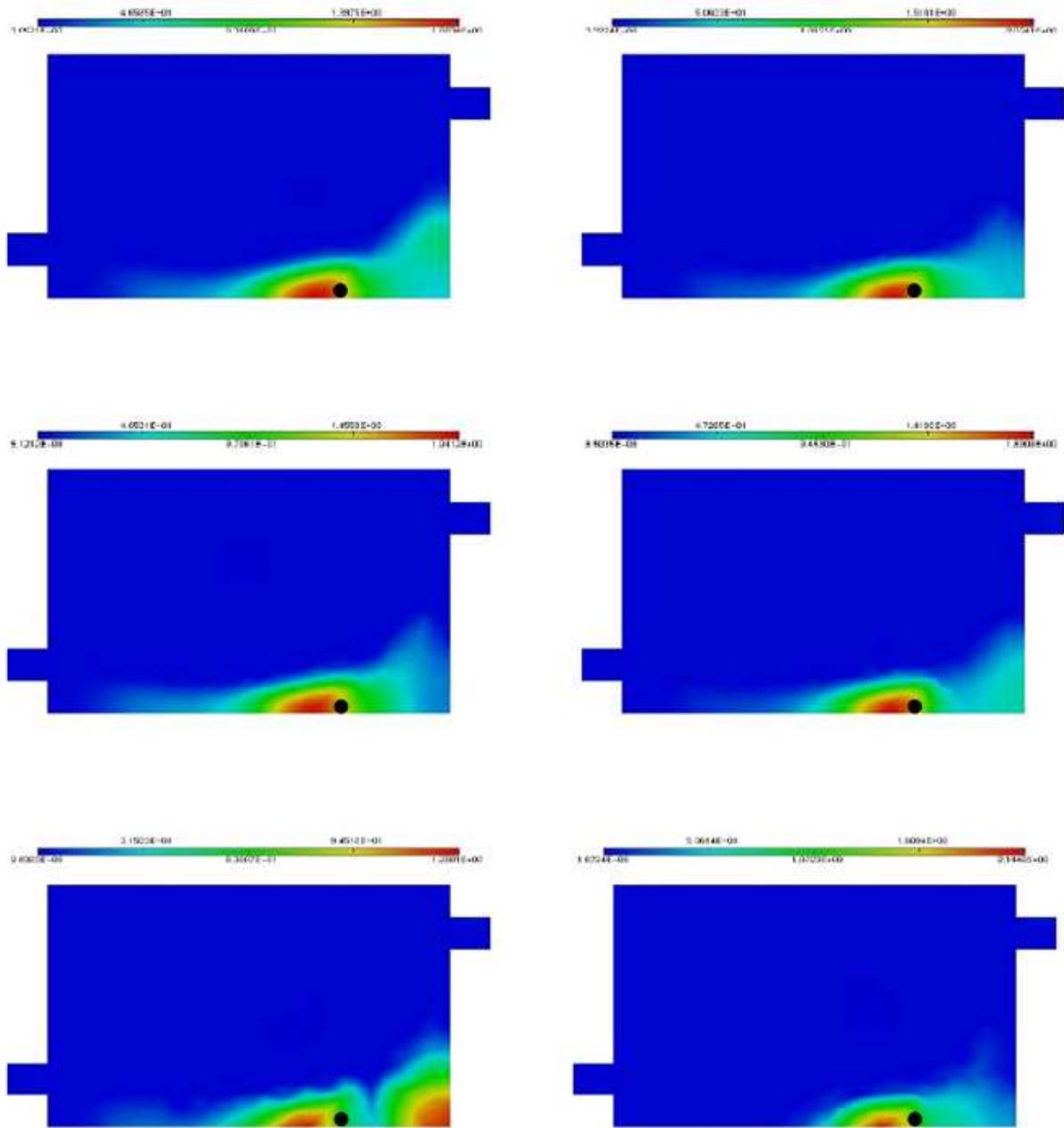


Figure 15.5. Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source location is the point (14.5, 0.5)

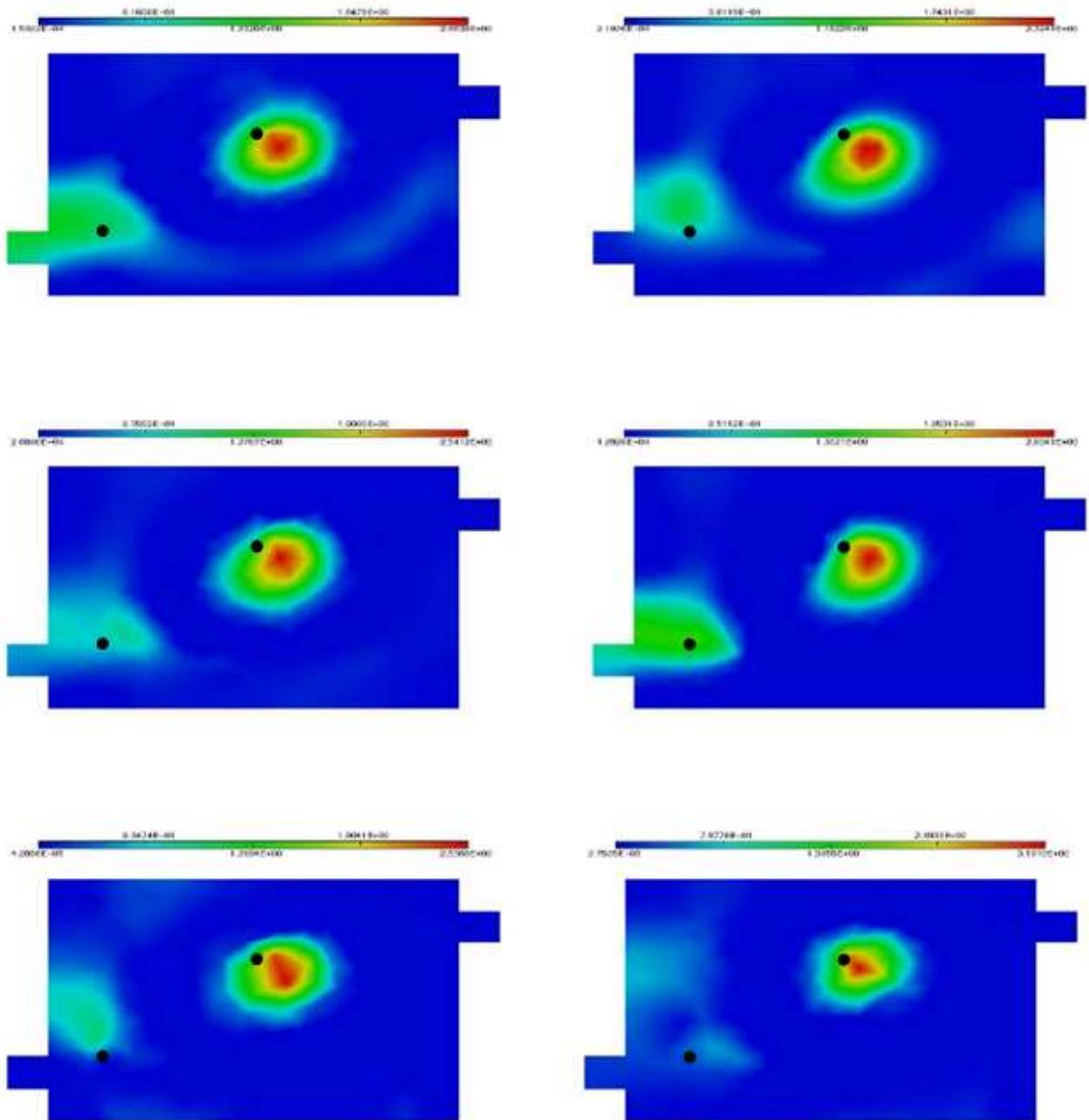


Figure 15.6. Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source locations are the points (3.0,4.0) and (10.0,8.0)

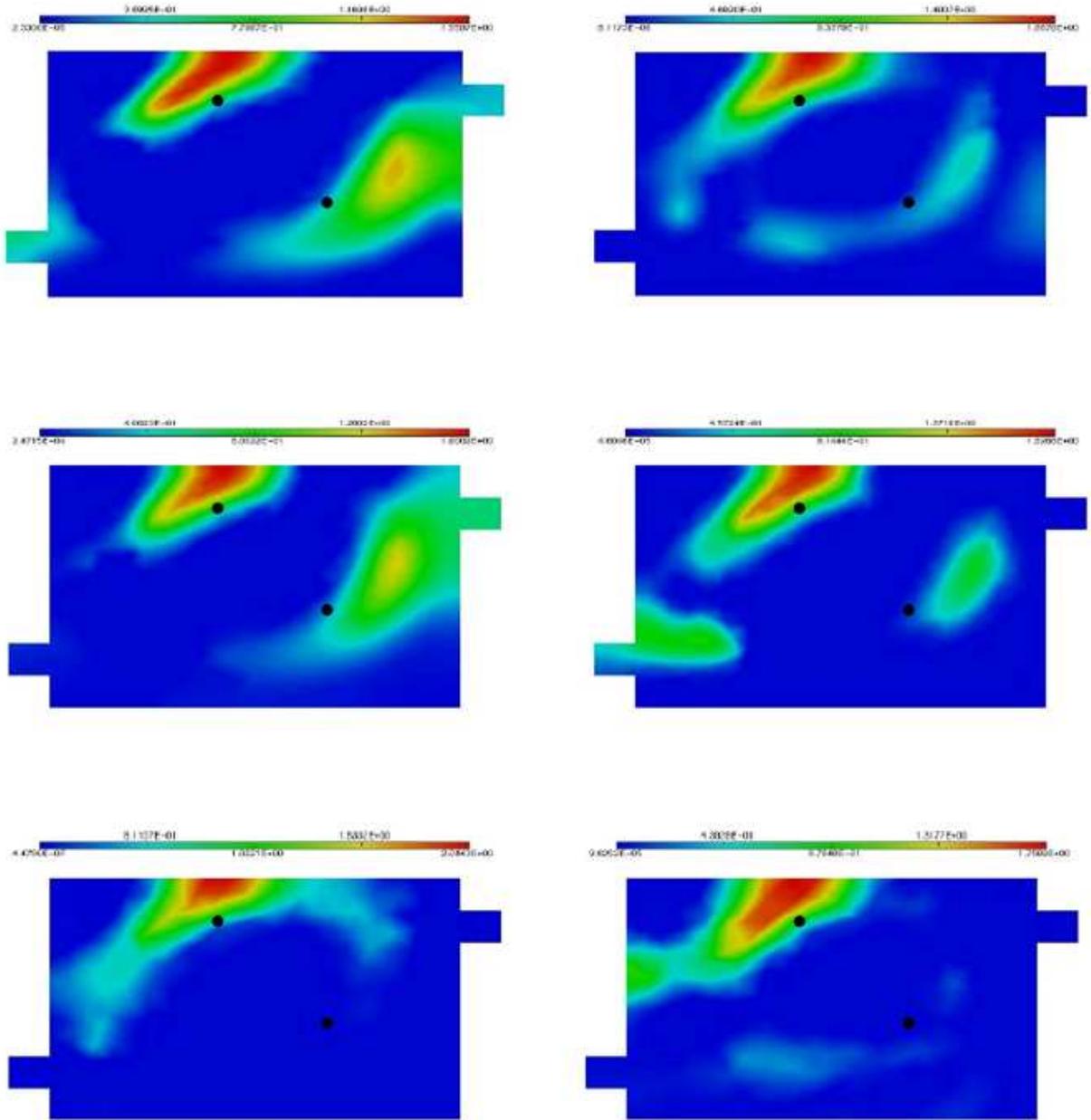


Figure 15.7. Source inversion with adaptive (left) and uniform (right) meshes. Top row, 1471 and 1488 triangles; middle row, 1172 and 1196 triangles; bottom row, 474 and 476 triangles. The true source locations are the points (6.0, 12.0) and (16.0, 6.0)

Approximate No. of Triangles	Uniform mesh		Adaptive mesh	
	<i>O3D</i> iterations	time	<i>O3D</i> iterations	time
31,000	6.27	385.77	–	–
1500	7.33	9.21	5.78	8.52
1200	6.88	7.13	5.84	6.58
500	7.78	3.00	8.09	3.05

Figure 15.8. Table giving run times and the number of *O3D* iterations at various levels of mesh coarsening

Chapter 16

Offline/Online QP Strategies for Real Time Inversion

Roscoe A. Bartlett, Andrew G. Salinger, John N. Shadid, Bart G. van Bloemen Waanders

16.1 Background

Here we describe an approach where large-scale direct sensitivities are exploited to solve an inversion problem of source terms on the domain boundary subject to transient convection diffusion by matching only a small set of observations. This formulation can be applied to a variety of problems such as identifying contamination events or controlling temperature in heating and ventilation systems. The goal of the inversion problem is to minimize the difference between the observations and predictions. Because the number of observations will typically be less than the possible discretization points on the boundary, the problem is ill-posed and needs to be regularized, which amounts to an additional penalty type term in the objective functional. Even though the regularization term causes a smoothing of the reconstruction, this formulation has proven to be effective in solving inversion problems [233, 18]. In our numerical experiments, a forward simulation with specified initial conditions and source terms is used to extract observations from a few locations on the boundary and then fed into the inversion problem as the target values.

Real-time performance is a critical goal in order for an inversion of contamination events or temperature control to have any practical use. A variety of approaches are discussed in this report to solve large-scale optimization problems as efficiently as possible, but here we focus on decomposing the optimization problem into offline and online calculations. The general idea is to recognize that a significant portions of the calculations do not change between successive optimization subproblems and can therefore be extracted as offline computations. What remains is then a subset of the overall computation and can be performed online with real-time performance. We describe the tailored solution of a special class of quadratic programming (QP) problems that arise in least-squares and inverse problems. The offline

computation involves the formation of a compressed state/source sensitivity matrix. This computation can be performed using a direct or adjoint approach offline. The online computation uses the precomputed compressed state/source sensitivity matrix to compute a reduced gradient and reduced Hessian which are then used to solve a reduced QP for the model parameters.

The proposed offline/online reduced QP method is demonstrated on a prototype problem where a contamination event is simulated in an airport terminal. The transport of the chemical is based on the steady-state convection-diffusion equations and the velocities are calculated through a steady-state, time-averaged Navier-Stokes model. A least-squares formulation attempts to reduce the difference between sparse concentration information and simulated values in an attempt to reconstruct the location and character of the contamination event. The main target application for this specialized QP method is the repeated online solution of inversion problems involving large-scale linear state constraints. The algorithmic strategy is presented in discrete and general form to emphasize the applicability of the formulation to other problems with similar characteristics.

16.1.1 Motivating Example Application: Source Inversion for Convection-Diffusion in an Airport Terminal

We describe a set of inversion problems involving a prototype model of an airport terminal. The goal of the inversion is to attempt to invert for the location and intensity of a chemical, biological or radiological release coming from within the facility given only sparse sensor data. In these scenarios, it is critical to be able to quickly (in a matter of seconds) invert for the source location of a possible terrorist attack so that remediation procedures can be quickly put into place. This problem is inherently transient and therefore a transient formulation is used and transient results are given.

The model used for the results in this section was that of an airport terminal shown in Figure 16.1. The governing equations are comprised of a finite-element discretization of the convection-diffusion equations. The velocity field is calculated (offline) through a Navier-Stokes solver with time-averaged turbulence modeling. The resulting velocity is field \mathbf{v} is based on inflow and outflow setting for a certain operating condition, which is assumed to be constant during the inversion calculation. The transient transport equation for contamination concentration $y(t)$, for $t \in [t_0, t_f]$, in a domain Ω is given by the standard convection-diffusion equation.

The infinite-dimensional inversion problem is formulated as follows:

$$\text{Minimize } \frac{1}{(t_f - t_0)\tilde{n}_s} \sum_{l=1}^P \sum_{i=1}^{\tilde{n}_s} \left(\frac{y(\mathbf{x}_i, t_l) - (\tilde{y}_l)_{(i)}}{|(\tilde{y}_l)_{(i)}| + \gamma} \right)^2$$

$$+ \left(\frac{\beta}{(t_f - t_0) \int_{\Gamma_N} d\Gamma_N} \right) \int_{t_0}^{t_f} \int_{\Gamma_N} u^2 d\Gamma_N dt \quad (16.1.1)$$

$$(16.1.2)$$

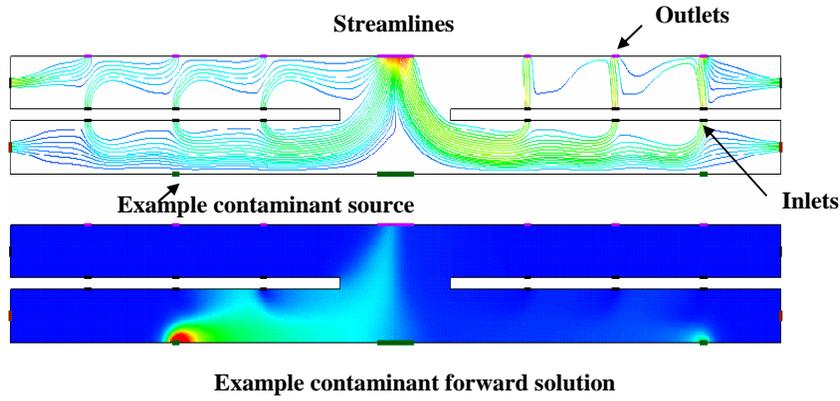


Figure 16.1. Two-D cross-section of the 3D airport terminal model. Also shown is an example flow pattern and contaminate release simulation.

$$\text{Subject to: } \frac{dy}{dt} - k\Delta y(t) + \mathbf{v} \cdot \nabla y(t) = 0, \text{ on } \Omega \text{ and } t \in [t_0, t_f], \quad (16.1.3)$$

$$y(t_0) = y_0 \text{ on } \Omega \quad (16.1.4)$$

$$\partial y(t)/\partial n = u(t) \text{ on } \Gamma_N \text{ and } t \in [t_0, t_f], \quad (16.1.5)$$

$$\partial y(t)/\partial n = 0 \text{ on } \Gamma_D \text{ and } t \in [t_0, t_f], \quad (16.1.6)$$

$$u(t) \geq 0 \text{ on } \Gamma_N \text{ and } t \in [t_0, t_f], \quad (16.1.7)$$

where $k > 0$ is the molecular diffusion (assumed constant), Γ_N the portion of the boundary where the source $u(t)$ is injected, Γ_D is the portion of the boundary where no source injection is allowed, and $\tilde{y}_l \in \mathbf{R}^{\tilde{y}}$ are the target values for the observed states at times t_l , for $l = 1 \dots P$.

In this model, possible source locations are allowed only on the bottom floor which is represented as Γ_N . The chosen spatial source discretization of the bottom floor neglects the possibility of the source emanating from other locations in the domain. While a more general spatial source discretization could have been used, the choice of using only the bottom floor was for simplicity and to limit the number of inversion parameters. In general, the entire perimeter of the facility could also be used as possible source contaminant locations and this would not greatly increase the number of inversion parameters in the discretized version of the problem.

In the following subsection, the general form of the QP problem is given and this is followed by an outline for the rest of the chapter.

16.1.2 Statement of the Problem: General QP Formulation

The offline and online calculation strategy is based on the recognition that our particular inversion problem can be formulated as a Quadratic Programming (QP) problem in which the objective (or cost) functional is quadratic and the constraints are linear. This class of problems can take advantage of standard solution methods [100, 160]. The form of the QP considered here is

$$\text{Minimize} \quad \frac{1}{2}(Cy - \hat{y})^T \hat{Q}_y (Cy - \hat{y}) + \frac{1}{2}u^T Q_u u \quad (16.1.8)$$

$$\text{Subject to} \quad Ay + Bu = 0 \quad (16.1.9)$$

$$u_L \leq u \leq u_U \quad (16.1.10)$$

where

$y \in \mathbf{R}^{n_y}$ are state variables,

$u \in \mathbf{R}^{n_u}$ are design (or inversion) variables,

$\hat{y} \in \mathbf{R}^{n_s}$ is some target vector for the observed state,

$u_L, u_U \in \mathbf{R}^{n_u}$ are simple bounds on u ,

$\hat{Q}_y \in \mathbf{R}^{n_s \times n_s}$ is a diagonal positive definite matrix,

$Q_u \in \mathbf{R}^{n_u \times n_u}$ is a positive definite matrix,

$A \in \mathbf{R}^{n_y \times n_y}$ is a nonsingular unsymmetric matrix known as the *state Jacobian*,

$B \in \mathbf{R}^{n_y \times n_u}$ is a rectangular matrix known as the *design Jacobian*, and

$C \in \mathbf{R}^{n_s \times n_y}$ is a rectangular matrix known as the *state observation matrix*.

In the context of an inversion problem, this formulation solves for the inversion variables u that give the best possible match of the linear state model (16.1.9) for y to some observation of the state variables $Cy \in \mathbf{R}^{n_s}$ compared to a target vector $\hat{y} \in \mathbf{R}^{n_s}$. Only incomplete information about the target state in \hat{y} is known and these are selected by the *state observation matrix* $C \in \mathbf{R}^{n_s \times n_y}$. In inversion problems, the number of state observations n_s is less than the number of inversion parameters n_u . This gives rise to an ill-posed inversion problem where an entire subspace of solutions for u exists. Some type of regularization is therefore added to the objective function in the form of the positive-definite matrix Q_u that results in a well-posed QP with a unique solution for u . For problems with many inversion parameters, enforcing physical bounds on the inversion parameters (for example, non-negative concentrations) in (16.1.10) results in much better inversion results in terms of quality of inversion. Instead of inverting for every location in a 3D domain, we consider only the boundary of the 3D domain and in addition we parametrize the domain so that we can control the final number of inversion parameters. We focus therefore on the solution of QPs where n_u and n_s are not excessively large (i.e. n_u and n_y are a few thousand or less).

In addition, to achieve real-time performance we recognize that the data in \hat{Q}_y and \hat{y} change from one QP to the next but not the model matrices A , B or C . We are also interested in using approaches that allow for a fast solution time for repeated QP subproblems. It should be noted that for most problems, even a single iterative solve with the full state Jacobian A (or its adjoint) can not be performed online in real-time. What is critical about this approach is that the state equation matrices A and B and the state observation matrix C are constant for all QPs that we consider. This allows one to perform perhaps some very expensive computations with A , B and C offline and then use these computed quantities in the repeated online solution of the QPs.

In the following section, the reduced form of the general QP problem is presented. This is followed by a general description of the offline/online decomposition approach in Section 16.3. A specialization of the general QP problem for linearly constrained transient inversion problems is then given in Section 16.4. Details of the discretization, general approach, and results for the transient airport terminal model are laid out in Section 16.5. Finally, conclusions and future work are described in Section 16.6 and Section 16.7.

16.2 Reduced QP problem

For the class of QPs considered here, it is useful to form a reduced QP where the state variables y and the linear equality constraints in (16.1.9) are eliminated from the problem. It is this linear elimination that allows a significant portion of the computation to be performed offline and therefore will be the key to the success of the repeated online solution of this class of QPs.

The elimination of the linear equality constraints in (16.1.9) gives rise to the following implicit state function

$$y(u) = Du \quad (16.2.11)$$

where

$$D = -A^{-1}B \in \mathbf{R}^{n_y \times n_u} \text{ is known as the direct sensitivity matrix.}$$

To obtain the reduced QP, $y(u)$ is substituted into (16.1.8) to obtain

$$\text{Minimize} \quad g^T u + \frac{1}{2} u^T G u \quad (16.2.12)$$

$$\text{Subject to} \quad u_L \leq u \leq u_U \quad (16.2.13)$$

where

$$g = -D^T C^T \hat{Q}_y \hat{y} \in \mathbf{R}^{n_u} \text{ is the reduced gradient, and}$$

$$G = D^T C^T Q_y C D + Q_u \in \mathbf{R}^{n_u \times n_u} \text{ is the reduced Hessian.}$$

It is this reduced QP formulation in (16.2.12)–(16.2.13) that is the focus of the offline/online decomposition.

16.3 Decomposition of Reduced QP into Offline and Online Subproblems

We consider a decomposition of the reduced QP in (16.2.12)–(16.2.13) into offline and online subproblems to achieve real-time performance. An important concept of the approach is to realize that not all the values of the direct sensitivity matrix D are required in the optimization calculation. In the inversion problem,

only the state concentration values at sensor locations need to be extracted, stored and then read in during the on-line computation and this is embodied in the matrix

$$\hat{D} = CD = -CA^{-1}B \in \mathbf{R}^{n_s \times n_u} \quad (16.3.14)$$

which is referred to as the *compressed direct sensitivity matrix* in the sequel. Storing \hat{D} avoids large storage requirements and IO costs during the online calculation process.

We provide a general overview of the decomposition algorithm and further define the individual calculations in subsequent sections. The general algorithm is as follows:

Algorithm 16.3.1. OFFLINE AND ONLINE COMPUTATIONAL PROCEDURE

1. *Offline Computations*
 - compute compressed direct sensitivity matrix \hat{D}
 2. *Online Computations*
 - read in compressed direct sensitivity matrix \hat{D}
 - for each online QP problem
 - read in compressed data \hat{Q}_y and \hat{y}
 - assemble reduced gradient g
 - assemble and factor reduced Hessian G
 - solve bound constrained QP (16.2.12)–(16.2.13)
-

The next subsection describes various approaches for the offline computation of the compressed direct sensitivity matrix \hat{D} . This is followed Subsection 16.3.2 by a discussion of how, given a precomputed matrix \hat{D} , the reduced QP subproblem in (16.2.12)–(16.2.13) is formed and solved.

16.3.1 Offline Subproblem

The offline subproblem involves the precomputation of the compressed direct sensitivity matrix $\hat{D} = -CA^{-1}B$. This matrix can be computed by either first solving the set of n_u simultaneous systems $A^{-1}B$ (known as the forward approach), or by first solving the set of n_s simultaneous systems $A^{-T}C^T$ (known as the adjoint approach). There are many factors to consider when selecting which of these two approaches to use. Note that for some challenging applications, the number of state variables n_y may be in the millions or more and solving systems involving A and A^T may require the use iterative linear solvers (such as GMRES) on a distributed-memory massively parallel processing (MPP) computer. However, $\hat{D} = -CA^{-1}B \in \mathbf{R}^{n_s \times n_u}$ will always be a relatively small matrix since we are only considering problems where n_u and n_s are a few thousand at most. Therefore, while the computation of \hat{D} may be able to effectively utilize a large MPP, the storage and use of \hat{D} can not and a simple single-processor or symmetric multi-processor (SMP) is sufficient for all of the online computations described a little later.

These two approaches, and the issues involved with each, are described in the following subsections.

16.3.1.1 Computing Compressed Sensitivities using the Forward Approach

The forward approach for computing the compressed direct sensitivity matrix $\hat{D} = -CA^{-1}B$ involves first solving the n_u simultaneous systems

$$D = -A^{-1}B \quad (16.3.15)$$

followed by simply multiplying $\hat{D} = CD$. This is potentially the most efficient approach for steady-state and transient problems when $n_u < n_s$. However, even when $n_u > n_s$, the forward approach may still be preferable since it only requires forward solves.

For a steady-state problem, solving $A^{-1}B$ requires a single block iterative linear solve with n_u simultaneous right-hand sides. For a transient problem, solving $A^{-1}B$ actually involves solving a set of n_u simultaneous linear ODEs from the initial time to the final time. For both the steady-state and transient problems, block linear solvers, such as block GMRES, may greatly reduce the cost of these expensive offline computations.

16.3.1.2 Computing Compressed Sensitivities using the Adjoint Approach

The adjoint approach for computing the compressed direct sensitivity matrix $\hat{D} = -CA^{-1}B$ involves first solving the n_s simultaneous adjoint systems

$$T = A^{-T}C^T \quad (16.3.16)$$

followed by simply multiplying $\hat{D} = -T^TB$. This is probably the more efficient approach when $n_s < n_u$. If an iterative linear solver such as GMRES is being used, then solving the adjoint system requires adjoint multi-vector products of the form A^TV . Note that there is no cheap way of approximating these adjoint products using directional finite differences.

For a steady-state problem, solving $A^{-T}C^T$ requires a single block adjoint iterative linear solve with n_s simultaneous right-hand sides. For a transient problem, a set of n_s simultaneous linear ODEs must be integrated from the final time backwards to the initial time. Since the adjoint sensitivity equations are linear and do not depend on the state, no transient state information needs to be stored and only local time-step information is needed. For both the steady-state and transient problems, block linear solvers, such as block GMRES, may greatly reduce the cost of these expensive offline adjoint computations.

16.3.2 Online Subproblem

The solution of the online problem first involves forming the reduced gradient g and the reduced Hessian G in (16.2.12) given the compressed data \hat{Q}_y , and \hat{y} , and the precomputed compressed direct sensitivity matrix \hat{D} . Once these quantities are formed, the reduced bound-constrained QP (16.2.12)–(16.2.13) is then solved. All of these online calculations are relatively computationally inexpensive (compared to the offline computation) and can therefore be performed on a single processor or multi-processor SMP shared-memory machine. On an SMP, all of the level-2 and level-3 BLAS and LAPACK operations performed can exploit more than one thread of execution and therefore provide some parallel speedup.

The computation of the reduced gradient $g = \hat{D}^T \hat{Q}_y \hat{y}$ is straightforward and relatively cheap. Computing g only requires an $O(n_s)$ diagonal scaling to form $\hat{Q}_y \hat{y}$ followed by an $O(n_u n_s)$ level-2 matrix-vector multiplication $g = \hat{D}^T (\hat{Q}_y \hat{y})$. The cost of this computation is in the noise of the CPU time of the other online computations.

The reduced Hessian $G = \hat{D}^T \hat{Q}_y \hat{D} + Q_u$ is formed online as follows. First, the diagonal matrix \hat{Q}_y is decomposed into

$$\hat{Q}_y = \hat{Q}_y^{1/2} \hat{Q}_y^{1/2} \quad (16.3.17)$$

by simply taking the square root of the diagonal of \hat{Q}_y (which gives real components since \hat{Q}_y is positive definite). This is only an $O(n_s)$ operation and is therefore very fast.

Next, the matrix

$$\hat{J} = \hat{Q}_y^{1/2} \hat{D} \quad (16.3.18)$$

is formed by simply scaling the rows of \hat{D} using the diagonal entries of $\hat{Q}_y^{1/2}$. This is an $O(n_s n_u)$ operation.

After \hat{J} is calculated, the $O(n_u (n_s)^2)$ level-3 BLAS operation

$$\hat{S} = \hat{J}^T \hat{J} \in \mathbf{R}^{n_u \times n_u} \quad (16.3.19)$$

is performed. For problems where $n_s \gg n_u$ (for example with least-squares problems), this computation is the dominant cost of the online solution of the reduced QP. For problems where $n_s < n_u$ (which is always the case in an ill-posed inversion problem for instance) then \hat{S} will be singular with at least $n_u - n_s$ zero eigenvalues and this computation will not be a dominant component of the online solution.

Once the matrix \hat{S} is formed, it is added to the matrix Q_u to form the reduced Hessian

$$G = \hat{S} + Q_u \quad (16.3.20)$$

which is an $O(n_u^2)$ operation. Note that since \hat{S} is positive semi-definite and Q_u is positive definite, then G is guaranteed to be positive definite. However, the conditioning of G may be very poor if \hat{S} is singular and dominates Q_u .

Finally, the active-set QP solver used to solve (16.2.12)–(16.2.13) requires many solves with G and this is facilitated by first performing an $O(n_u^3)$ Cholesky factorization of G . Of all of the online computations described up to this point, this $O(n_u^3)$ Cholesky factorization is by far the most expensive when $n_s \ll n_u$.

After the reduced gradient g has been formed and the reduced Hessian G has been calculated and factored, then the reduced QP subproblem (16.2.12)–(16.2.13) can be solved. There are many different approaches to solving such a QP but the one used in the results reported in Section 16.5 was QPSchur [29]. QPSchur solves convex QPs (that is, QPs where the Hessian is positive definite) using a dual active-set method that starts with the unconstrained minimizer and then adds violated inequality constraints until the solution is found. Each addition to the working set involves at least one $O(n_u^2)$ level-2 BLAS back-solve with the Cholesky factors of G . Therefore, the online cost of solving a QP with QPSchur with n_{act} active inequality constraints is at least $O(n_u^2 n_{act})$. Note that the number of QP iterations can easily exceed n_u and each of these iterations can only utilize level-2 BLAS. Therefore, even though the number of flops in the QP solve is still usually just $O(n_u^3)$, the actual runtime can be much higher than the $O(n_u^3)$ Cholesky factorization of G which can utilize more efficient level-3 BLAS. Hence, if n_{act} is moderately large, then QPSchur will dominate the online solve.

16.4 Linearly Constrained Transient Inversion Problems

In this section, we consider transient inversion problems where the state model satisfies the following linear ODEs

$$\tilde{M} \frac{d\tilde{y}(t)}{dt} + \tilde{A}\tilde{y}(t) + \tilde{B}\tilde{u}(t) = 0, \text{ on } t \in [t_0, t_f], \quad (16.4.21)$$

$$\tilde{y}(t_0) = \tilde{y}_0, \quad (16.4.22)$$

$$(16.4.23)$$

where $\tilde{y}(t) \in \mathbf{R}^{\tilde{n}_y}$ and $\tilde{u}(t) \in \mathbf{R}^{\tilde{n}_u}$ are the spatially-discretized state and source at time t and $\tilde{M} \in \mathbf{R}^{\tilde{n}_y \times \tilde{n}_y}$ is a nonsingular mass matrix. This form of the state equations is used by the example inversion problem described in Section 16.5. Here, we focus on the general structure of this problem and of the general solution approach.

In order to transform the infinite-dimensional temporal ODEs in (16.4.21)–(16.4.22) into the standard form of the linear equations in (16.1.9), temporal discretizations for the state $\tilde{y}(t)$ and the source parameters $\tilde{u}(t)$ must be selected.

The temporal discretization of the source parameters can take the general form

$$\tilde{u}(t) = \sum_{j=1}^M \phi_j(t) \tilde{u}_j, \quad (16.4.24)$$

where $\phi_j(t) \in \mathbf{R} \rightarrow \mathbf{R}$ are a set of $j = 1 \dots M$ basis functions and $\tilde{u}_j \in \mathbf{R}^{\tilde{n}_u}$ are a set of $j = 1 \dots M$ coefficient vectors. We discuss a particular selection of these basis functions in Section 16.4.2.

The temporal discretization for the state $\tilde{y}(t)$ and the ODEs in (16.4.21)–(16.4.22) can be chosen from a variety of possible options. Here, primarily for illustrative purposes, we consider the implicit backward Euler method which yields the set of coupled linear equations

$$\tilde{M} \left(\frac{\tilde{y}_{k+1} - \tilde{y}_k}{\Delta t_k} \right) + \tilde{A}\tilde{y}_{k+1} + \tilde{B}\tilde{u}(t_{k+1}) = 0, \text{ for } k = 0 \dots N - 1, \quad (16.4.25)$$

where \tilde{y}_0 is given, $\Delta t_k = t_{k+1} - t_k$ for $k = 0 \dots N - 1$, and $t_N = t_f$. Of course the equations in (16.4.25) are solved forward in time for \tilde{y}_{k+1} , for $k = 0 \dots N - 1$.

The temporal discretizations of the source $\tilde{u}(t)$ in (16.4.24) and for the state $\tilde{y}(t)$ and ODEs in (16.4.25) transform the infinite-dimensional temporal problem into a set of linear equations of the form $Ay + Bu = 0$

time steps $k(1), k(2), \dots, k(l), \dots, k(P)$ then the global matrix $C \in \mathbf{R}^{n_s \times n_y}$ can be represented as

$$C = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_l \\ \vdots \\ C_P \end{bmatrix} \quad (16.4.32)$$

where $C_l \in \mathbf{R}^{n_s \times \tilde{n}_y}$, for $l = 1 \dots P$, are block row-wise matrices with $k = 1 \dots N$ horizontal blocks, all of which are zero except for the $k(l)$ block which is equal to \tilde{C} which is shown as

$$C_l = \begin{bmatrix} 0 & \dots & 0 & \tilde{C} & 0 & \dots & 0 \end{bmatrix}_{k(l)} \quad (16.4.33)$$

for $l = 1 \dots P$. Above, the 0 blocks represent zero matrices of dimension $(\tilde{n}_s \times \tilde{n}_y)$. This global matrix C is really nothing more than a compact specification of how observations of the state are sampled at the various times from the global state vector in (16.4.26).

The compressed data is compactly represented as

$$\hat{Q}_y = \begin{bmatrix} \bar{Q}_{y,1} & & & & & & \\ & \bar{Q}_{y,2} & & & & & \\ & & \ddots & & & & \\ & & & \bar{Q}_{y,l} & & & \\ & & & & \ddots & & \\ & & & & & \bar{Q}_{y,P} & \end{bmatrix} \in \mathbf{R}^{P\tilde{n}_s \times P\tilde{n}_s} \quad (16.4.34)$$

$$\hat{y} = \begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_l \\ \vdots \\ \bar{y}_P \end{bmatrix} \in \mathbf{R}^{P\tilde{n}_s} \quad (16.4.35)$$

where $\bar{Q}_{y,l} \in \mathbf{R}^{\tilde{n}_y \times \tilde{n}_y}$ and $\bar{y}_l \in \mathbf{R}^{\tilde{n}_y}$ are the compressed objective function data for the time snapshot observations $l = 1 \dots P$ at the time steps $k(1), k(2), \dots, k(l), \dots, k(P)$. For the objective in (16.1.2) the diagonal Hessians $\hat{Q}_{y,l}$ have the diagonal elements

$$\text{diag} \{ \hat{Q}_{y,l} \}_{(i)} = \frac{1}{(t_f - t_0)\tilde{n}_y (|(\bar{y}_l)_{(i)}| + \gamma)}, \text{ for } i = 1 \dots \tilde{n}_y \quad (16.4.36)$$

for $l = 1 \dots P$. Note that $n_s = P\tilde{n}_s$ and therefore the total number of observations is the number of state samples in a time snapshot \tilde{n}_s multiplied by the number of snapshots P . This means that n_s can grow in two

ways by increasing the number of observations per time step and/or by increasing the number of observation snapshots.

It is worthwhile to consider the general block structure of the resulting compressed direct sensitivity matrix \hat{D} that arises from (16.4.28)–(16.4.29) and (16.4.32) (as given by $\hat{D} = CA^{-1}B$) which is

$$\hat{D} = \begin{bmatrix} \hat{D}_{1,1} & \hat{D}_{1,2} & \dots & \hat{D}_{1,j} & \dots & \hat{D}_{1,M} \\ \hat{D}_{2,1} & \hat{D}_{2,2} & \dots & \hat{D}_{2,j} & \dots & \hat{D}_{2,M} \\ \vdots & \vdots & & \vdots & & \vdots \\ \hat{D}_{l,1} & \hat{D}_{l,2} & \dots & \hat{D}_{l,j} & \dots & \hat{D}_{l,M} \\ \vdots & \vdots & & \vdots & & \vdots \\ \hat{D}_{P,1} & \hat{D}_{P,2} & \dots & \hat{D}_{P,j} & \dots & \hat{D}_{P,M} \end{bmatrix} \in \mathbf{R}^{P\tilde{n}_s \times M\tilde{n}_u} \quad (16.4.37)$$

In the above compressed direct sensitivity matrix, $\hat{D}_{l,j} \in \mathbf{R}^{\tilde{n}_s \times \tilde{n}_u}$, for $l = 1 \dots P$ and $j = 1 \dots M$, is the sensitivity of the observed state $\tilde{C}y_{k(l)}$ at the time $t_{k(l)}$ with respect to the source basis vector \tilde{u}_j . This interpretation of $\hat{D}_{l,j}$ is generalized somewhat in the next section.

16.4.1 Generalized Interpretation of \hat{D} , \hat{Q}_y and \hat{y}

From the standpoint of the online direct QP solver, all that is important are the objects \hat{D} , \hat{Q}_y and \hat{y} ; and that they be created in a logical and consistent manner. Therefore, the details of how the ODE is integrated, how the observations of the state are sampled, or how the source is discretized are unimportant to the online computation. From this slightly more abstract point of view, any time integration algorithm can be used to solve the ODE in (16.4.21)–(16.4.22) and the only requirement is that the states be sampled at specific times $t_1, t_2, \dots, t_l, \dots, t_P$ in order to generate \hat{D} given in (16.4.37). These times t_l , for $l = 1 \dots P$, must correspond to the same times for which target state observations \tilde{y}_l , for $l = 1 \dots P$, are given.

16.4.2 Piecewise-Constant Temporal Discretization of the Source $u(t)$ and the Specialized Computation of \hat{D}

Here we consider the special structure of the compressed direct sensitivity matrix \hat{D} in (16.4.37) using piece-wise linear basis functions $\phi_j(t)$, for $j = 1 \dots M$, for the source in (16.4.24) of the form

$$\phi_j(t) = \begin{cases} \frac{t-t_{j-1}}{t_j-t_{j-1}} & : \text{ for } t \in [t_{j-1}, t_j) \\ \frac{t-t_{j+1}}{t_j-t_{j+1}} & : \text{ for } t \in [t_j, t_{j+1}) \\ 0 & : \text{ otherwise} \end{cases} \quad (16.4.38)$$

which gives “hat” functions at each time point t_j . The significance of this choice for $\phi_j(t)$ is that the compressed direct sensitivity matrix \hat{D} in (16.4.37) takes on a block lower triangular-like form. To see this, consider the discretization show in Figure 16.2 with $M = 6$ temporal source nodes and $P = 3$ state sensor snapshot readings. This gives rise to the \hat{D} matrix shown in Figure 16.3. In this example, the source coefficients u_1 and u_2 can affect all of the sensor reading snapshots \hat{y}_1 , \hat{y}_2 , and \hat{y}_3 and therefore all of these

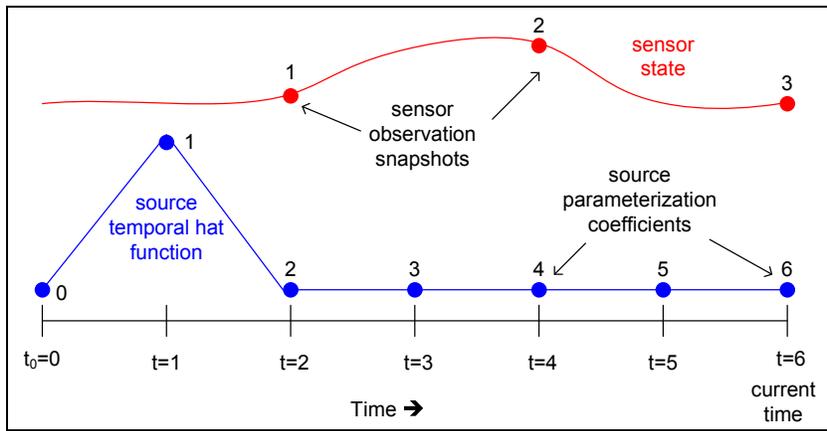


Figure 16.2. Source and state sensor profiles for a transient inversion problem for $P = 6$ source basis vectors with basis functions ϕ_j that span $M = 3$ sampling periods of the state and $\tilde{n}_u = 2\tilde{n}_s$ (i.e. twice as many source parameters that state observations in a time snapshot).

blocks are nonzero. However, the source coefficients u_3 and u_4 can only affect the sensor reading snapshots \hat{y}_2 and \hat{y}_3 and therefore the blocks (1,3) and (1,4) must be zero. Finally, the source coefficients u_5 and u_6 can only affect the sensor reading snapshot \hat{y}_3 and therefore, only the blocks (3,5) and (3,6) are nonzero in the last two block columns.

If the temporal source discretization nodes coincide with the state sensor snapshot times, as shown in Figure 16.2, then the full matrix \hat{D} can be computed at a fraction of the cost of the general case. To see how this is done, consider the effect that shifting a source profile in time has on the output state sensor readings as shown in Figure 16.4. With a zero initial condition for the state, shifting the source in time results in an identical shifting of the state in time. Because of this, sensitivities for source coefficients for earlier times can be shifted in time and used for sensitivities for source coefficients and sensor readings for later times. For example, in Figure 16.3, blocks (2,3) and (3,5) are the same as block (1,1), Likewise the blocks (2,4) and (3,6) are the same as block (1,2). Therefore, in this example, only the sensitivities with respect to the source coefficients u_1 and u_2 must be explicitly computing. The remaining nonzero blocks in \hat{D} are determined by shifting these initial blocks down and to the right. The reuse of block is shown using color coding in Figure 16.3. The exact nature of this mapping depends on the number temporal source nodes M and the number sensor snapshots P but the general concept should be clear.

16.5 Source Inversion for Convection-Diffusion in an Airport Terminal : Discretization and Results

In this section we describe the discretization, other numerical details, and some results for the transient inversion problem first introduced in Section 16.1.1.

16.5.1 Discretization and General Numerical Approach

The problem formulation used for the results shown later used a 2D spatial version of the airport terminal model. We have also experimented with a 3D version of the terminal model but we chose to use the 2D model to allow for better visualization of the inversion results. The basic mathematical approach and the properties of the inversion do not change when going from a 2D spatial model to a 3D spatial model.

The transient convection-diffusion equations in (16.4.21)–(16.4.22) were spatially discretized using a finite-element method implemented in MPSalsa and the resulting ODE was solved using a backward-Euler method with constant time steps. In addition, the time-averaged steady-state velocity field \mathbf{v} used in the convection-diffusion equations was computed using a turbulent finite-element discretization with MPSalsa. In all of the numerical experiments reported, a time horizon of 64 seconds was used with $n_t = 128$ evenly spaced time steps of size $\delta t = 0.5$ seconds. While a backward-Euler discretization only gives $O(\delta t)$ temporal accuracy, this is of little consequence with respect to this work. For the standpoint of the inversion problem, the model is the model and it is not critical if the model 100% approximates reality. The same discrete model is used to both generate the sensor data from the forward problem and to invert for the source locations.

The compressed direct sensitivity matrix \hat{D} was computed by running MPSalsa's time integrator. Because of the specialized structure of this problem, as described in Section 16.4.2, we generated all of \hat{D} by just explicitly computing n_x different sensitivities using the time integrator in MPSalsa. Each of these transient integrations (with 128 time steps) took about five minutes on a 2.5 MHz single processor linux machine. In our prototype implementation, we ran MPSalsa separately for each of the n_x sensitivities. Since the equations are linear, each sensitivity was computed by simply setting a single impulse source of magnitude 1.0 at the first time node for each of the n_x spatial nodes. The rest of \hat{D} was then formed by translation as described in Section 16.4.2 using perl scripts.

The source field along the floor of the 2D airport terminal model was spatially discretized using n_x evenly spaced source locations. The source was then temporally discretized using m_t/n_t evenly spaced piece-wise linear nodes. This gives $n_u = n_x \times m_t/n_t$ total discretized source parameters. The piece-wise linear source discretization was described in Section 16.4.2.

In all these inversion problems, it is assumed that the initial condition for the contaminant is zero at the beginning of the time horizon. This is a critical aspect and this requires that the time horizon be sufficiently long to allow for enough sensor data to be acquired to allow for a reasonable inversion result. A zero initial condition is critical in allowing for the specialized computation of the sensitivity matrix \hat{D} as described in Section 16.4.2.

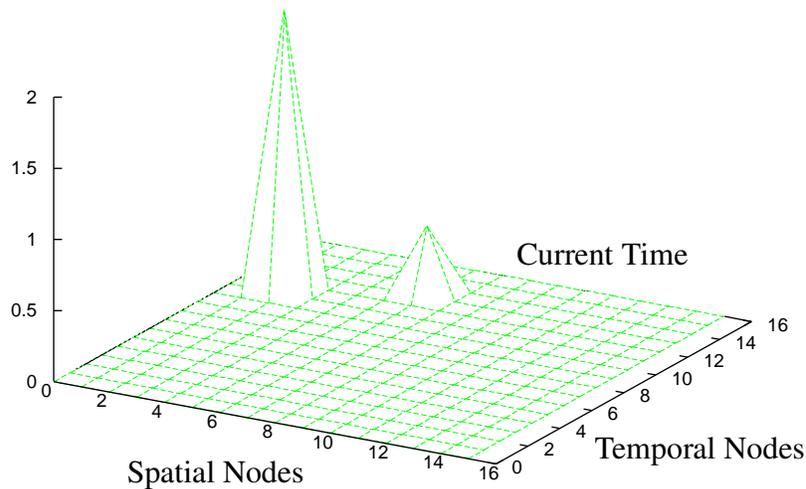


Figure 16.5. Two impulse releases, one at spatial node 2 with magnitude 2.0 and another 16 seconds later at spatial node 6 with magnitude 0.5.

16.5.2 Transient Inversion Results

Several different aspects of the inversion algorithm for this application are presented here. First, the general behavior of the inversion approach and the quality of the inversions are presented in Section 16.5.2.1. The impact of objective sensor scaling and bounds on the quality of the inversion is then investigated in Section 16.5.2.2. The computational expense of the online approach and a discussion of the impact of various aspects of the formulation and solution approaches is given in Section 16.5.2.4. Finally, issues of conditioning and regularization are covered in Section 16.5.2.4.

16.5.2.1 General Inversion Behavior and Quality

Here we consider the general quality of the inversion algorithms. In this section we consider a discretization of the source with 16 spatial nodes and 16 temporal nodes giving $n_u = 256$ total inversion parameters. The attach scenario is two impulse releases shown in Figure 16.5. Inverting for this attach scenario is difficult because of the relatively close spatial and temporal proximities of the first and second release and because the second release is much smaller than the first.

It is difficult to quantitatively define the criteria for a “good” inversion and the particular metric used should be determined by the final use of the result. Here we will use two different metrics for the quality of

an inversion: a quantitative measure and a qualitative measure.

The first measure of inversion quality used is the quantitative square-root of the integrated relative squared error

$$\text{relative error} = \left(\frac{\int_x \int_t (u - u^*)^2}{\max(u^*)} \right)^{1/2}. \quad (16.5.39)$$

The above integral is computed using the source discretization node-wise in space and time between the inverted source coefficients u and the true source coefficients u^* that was used to generate the sensor data in the forward problem. The form of the relative error computed in (16.5.39) insures that the maximum is near 1.0 for a complete mismatch and goes to zero as the inversion quality improves. This relative error will only be greater than 1.0 when the inverted magnitude of the inverted source u is greater than the true source u^* which is rare due to the presents of u^2 regularization.

The second inversion quality metric used is the qualitative “eye-ball” norm where one looks at the true source and the inverted source and then determines the quality of the match. The “eye-ball” norm is quite effectively is determining the location of the source(s) in the inversion result, even when there is little sensor data.

In the following subsection, the behavior of successive inversion solutions is investigated.

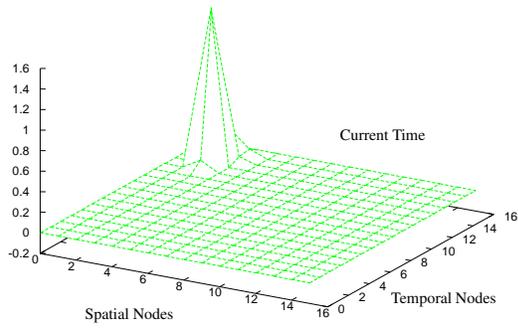
Simulation of Successive Inversion Solutions

It is interesting to observe the temporal nature of the inversion method in successive inversions. We simulated an attach scenario where we start solving inversion problems when the first sensor detects a harmful contaminate. In such a simulation, inversion problems are solved after each sensor snapshot reading. The first inversion problem is solved after 8 seconds with only one temporal sensor snapshot reading. The second inversion problem is solved after 16 seconds which uses data from two temporal sensor snapshot readings and so forth. Finally, at the end of 64 seconds the inversion problem is solved with eight sensor snapshot readings. Because each inversion problem acquires more data, we expect the quality of the inversion to improve as more time passes and that is what we see in our simulations as shown in Figure 16.6.

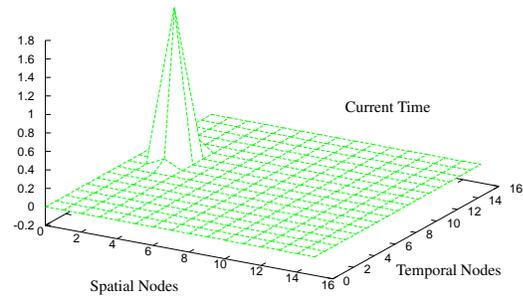
As shown in Figure 16.6, the quality of the inversion of the first impulse source becomes better and better as more sensor data is acquired. However, there is no indication of the presence second source until the very last $t = 64$ inversion run. The inability to “see” the second source is not a property of the inversion formulation but is determined by the number and location of the sensors as described in the next section.

Impact of Number of Sensors on Inversion Quality

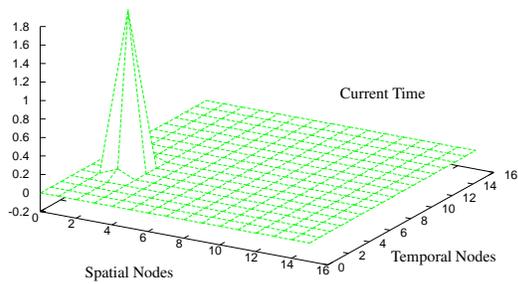
Here we consider the impact that the number and location of sensors has on the quality of the inversion result. In these experiments, we start with 128 randomly placed sensors in the spatial domain and then we



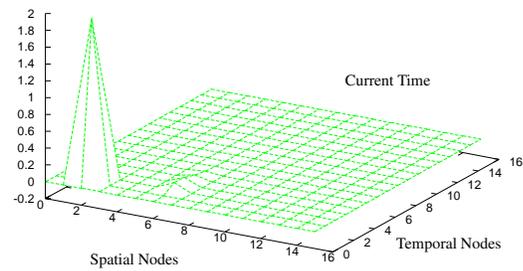
a) $t = 16$ seconds



b) $t = 32$ seconds



c) $t = 48$ seconds



d) $t = 64$ seconds

Figure 16.6. Progression of inversion results using 16 sensors with sensor snapshot readings every 8 seconds for the attach scenario shown in Figure 16.5.

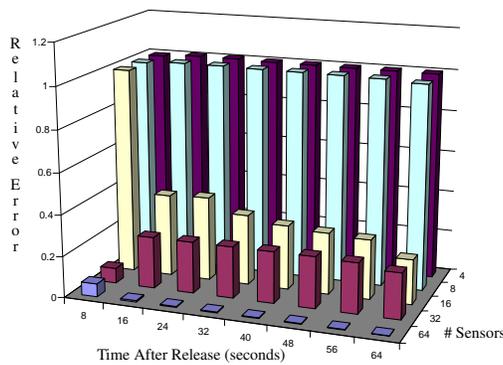
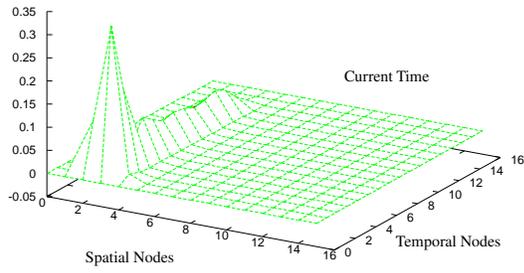


Figure 16.7. Comparison of relative errors (16.5.39) of the inversion result for the attach scenario shown in Figure 16.5. These relative errors are shown for different numbers of random sensors 4, 8, 16, 32, 64 and for times after initial release of 8 through 64 seconds. The number of sensors increases from back to front and the successive inversions progress from left to right.

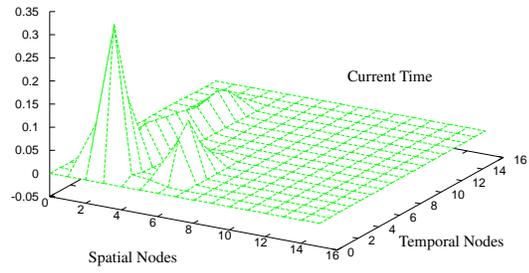
randomly sample sensors from this collection starting from the same seed each time until we reach the desired number of sensors. For example, when comparing 4 and 8 sensors, the set of 8 sensors strictly includes the set 4 sensors. Likewise, the set of 16 sensors strictly includes the set of 8 sensors and so on. Using this approach, we guarantee that the quality of the inversion can only improve by adding more sensors. Therefore adding more sensors can only improve the relative error as defined in (16.5.39) and this is seen in all of our results. A less careful approach of randomly sampling the sensors may result in different sensor placings and it can be that 4 sensors may give a better inversion result than 8 sensors, for instance, due to a more fortunate placement of the 4 sensors.

The quantitative relative error of the inversion results for different numbers of sensors in successive inversions over the 64 second time horizon are shown in Figure 16.7. Surface plots for the inverted source after 65 seconds from the first impulse release are displayed in Figure 16.8. Comparing the quantitative relative error in Figure 16.7 and the qualitative “eye-ball” norm between in Figure 16.5 (shifted in time) and Figure 16.8 yields may interesting observations.

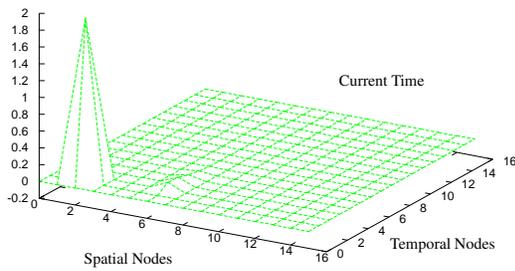
The relative error shown in Figure 16.7 uniformly decreases as sensors are added and as successive inversions are performed with more data. The only exception to this is the $t = 8$, 32-sensor case that has a lower relative error than the $t = 16$ second, 32-sensor case. This can be explained by the fact that at $t = 8$ seconds, the second impulse source had not quite occurred and therefore did not affect the relative error. However, in the $t = 16$ inversion, the second source had emerged but there was not as good of sensor data to identify it and therefore the relative error increased. In successive inversions though, the relative error decreased only minimally. What this shows is that in the 32-sensor configuration generated exceptional data for the first source but the data collected for the second source was much poorer. In fact, comparing the relative error between the 32-sensor and 64-sensor inversion at $t = 8$ seconds seems to suggest that the identification of the first source was as good in the 32-sensor case as in the 64-sensor case.



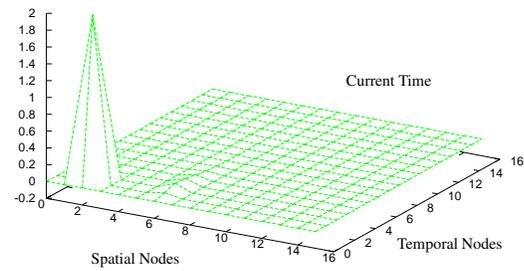
a) 4 Sensors



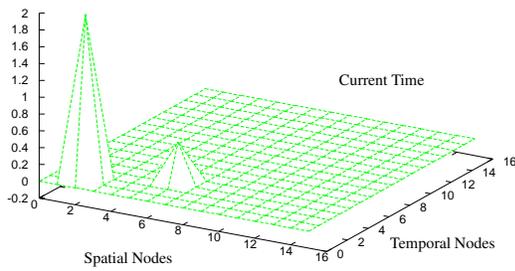
b) 8 Sensors



c) 16 Sensors



d) 32 Sensors



e) 64 Sensors

Figure 16.8. Comparison of the inversion results for the attach scenario shown in Figure 16.5 after $t = 64$ seconds for varying numbers of sensors.

For another interesting observation, note that while the relative error for the 4-sensor inversion is very close to 1.0 (i.e. very poor), the inversion result plotted in Figure 16.8(a) still does a reasonable job of showing the spatial location of the first, larger-magnitude source. However, with just 4 sensors, the temporal nature of the source can not be pinned down. The inversion result with just 8 sensors plotted in Figure 16.8(b) shows the spatial locations and the relative magnitudes of the two impulse sources remarkably well while the inversion results with 16 sensors and 32 sensors does not pick up the second, smaller source as well. Therefore, based on the metric of determining the spatial location of the sources one would conclude that the 8-sensor inversion out-performed the 16-sensor and 32-sensor inversions even though the quantitative relative error for the 8-sensor inversion is much worse. This can be explained in that in the 16-sensor and 32-sensor cases there was an extra sensor close to the first larger source and this dominated the other sensors that picked up the smaller second source. With these extra sensor gone in the 8-sensor case, the inversion formulation with the 8-sensors had more balanced data and resulted in a more balanced inversion than in the 16-sensor or 32-sensor cases.

The inversion quality really only becomes close to perfect in the 64-sensor inversion as seen in Figure 16.8(e). In the 64-sensor inversion, the inversion algorithm identifies the two sources the instant they occur and this is shown in the very low relative error bars in Figure 16.7.

These results seem to indicate that this flow field results in transport that is convection dominated. In a convection dominated flow, a released contaminant will follow the stream lines right out the exhaust ports to the facility and unless a sensor happens to be sitting close to that stream line, the contaminant will not be observed. On the other hand, diffusion dominated transport would result in sensors getting better and better readings as time goes on so we would expect the inversion result to improve dramatically as more temporal sensor snapshots are collected. Because our results show that the location of sensors is far more important than the number of snapshot readings, we can deduce that this flow is convection dominated.

16.5.2.2 Impact of Scaling and Bounds on Inversion Quality

Here we show why it is important to include scaling in the objective function in (16.1.2) as shown in (16.4.36) and impose the non-negativity bounds in (16.1.7).

Consider the single impulse source attack scenario shown in Figure 16.9. Figure 16.10 shows a comparison of the relative errors (16.5.39) between a) using no scaling and no bounds, b) using only bounds, c) using only scaling, and d) using bounds and scaling.

The qualitative difference in the quality of the inversion results can be seen in Figure 16.11 for the case with 32 sensors at time $t = 40$ seconds.

These results clearly demonstrate the importance of scaling the sensor observations and adding the bounds to the QP formulation. Without the objective scaling, the reduced Hessian G can be computed and factored offline. Without bounds, the QP can be solved with a single back-solve of the factors of G . Therefore, without bounds, the online problem would scale as $O((n_u)^2)$ and significantly finer discretizations of the source can be considered. The computational expense is investigated more in the next section.

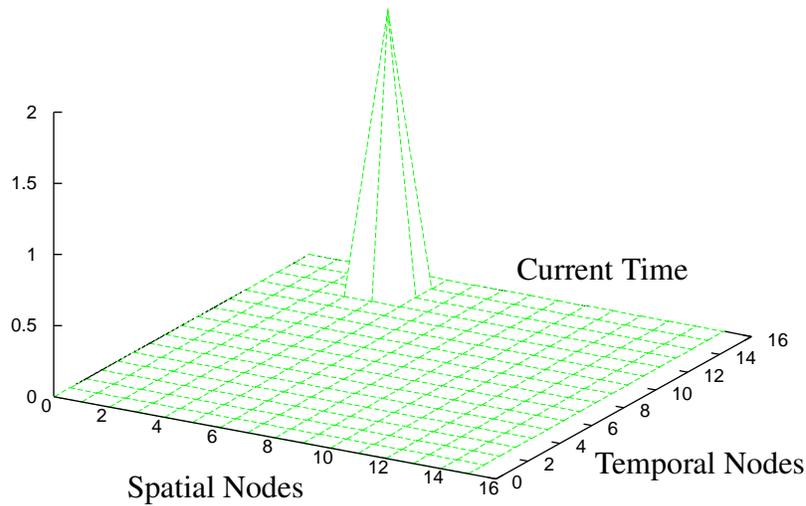


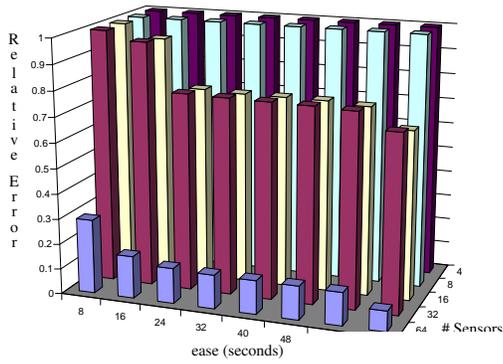
Figure 16.9. Spike release of magnitude 2 at spatial node 4 after 8 seconds.

16.5.2.3 Online Computational Expense and Inexact QP Solves

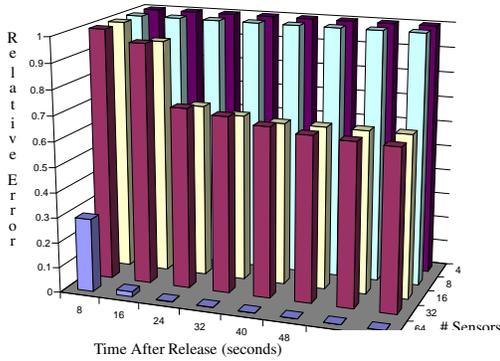
The proposed offline/online reduced-space approach described here was implemented in a mixed-language C++/Fortran program. The program was compiled with high optimizations with the gcc compiler suite version 3.3.4 using optimized BLAS for the dense matrix kernels and was run on a 2.8 MHz Intel Pentium IV machine running Linux. This implementation produced very fast inversion times for modest numbers of inversion parameters. For example, all of the $n_u = 256$ source discretization problems solved in less than 1.0 seconds with most times around 0.25 seconds. The downside of this approach, however, is that the online cost scales at least as bad as $(n_u)^3$. In this section, we break down the online computational costs of this approach, how they are affected by the use of scaling and bounds on the variables, and how to reduce those costs.

First we should note the offline costs for generating \hat{D} using the approach described in Section 16.5.1 where MPSalsa was run separately for each of the n_x spatial source nodes. Since each of these time integrations took about 5 minutes, computing sensitivities for the $n_x = 16$ case (for the $n_u = 256$ case) took about one hour twenty minutes. Likewise, computing sensitivities for the $n_x = 32$ (for the $n_u = 1024$ case) discretization required about two hours forty minutes. Note that a naive computation of \hat{D} for the $n_u = 1024$ case would require 1024 separate integration runs and would have taken about three and a half days!

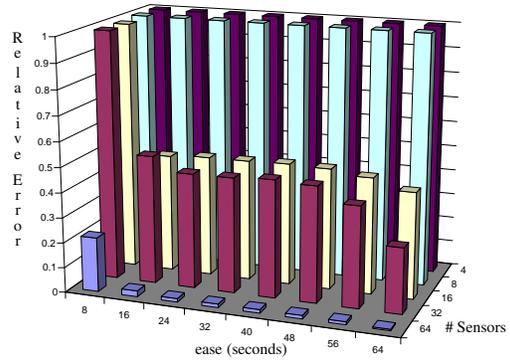
In this section, we consider a finer 32×32 spatial and temporal discretization of the source which gives $n_u = 1024$ total inversion parameters. Given a single impulse source, Figure 16.12 gives the relative errors and the CPU times for the online inversions for different numbers of sensors and for increasing time after



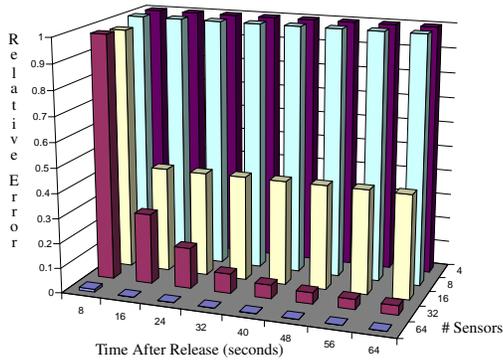
c) scaling only



b) With bounds, no scaling

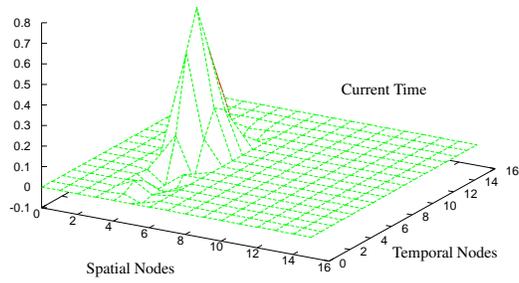


d) scaling, no bounds

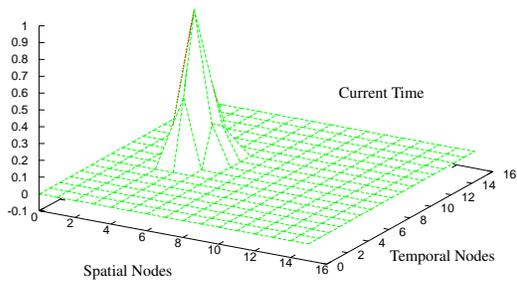


e) With scaling and bounds

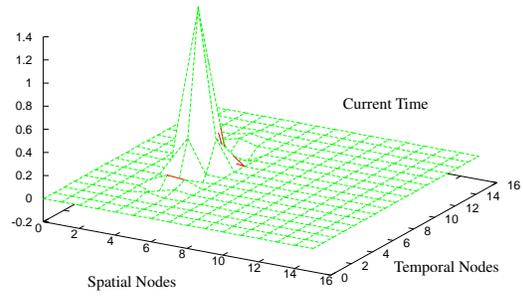
Figure 16.10. Comparison of relative errors (16.5.39) of the inversion result for using objective scaling and/or bounds for the source shown in Figure 16.9. These relative errors are shown for different numbers of random sensors 4, 8, 16, 32, 64 and for times after initial release of 8 through 64 seconds.



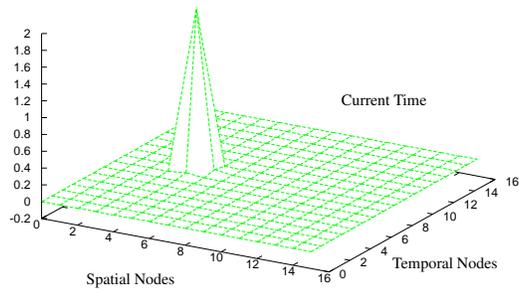
a) No scaling or bounds



b) With bounds, no scaling

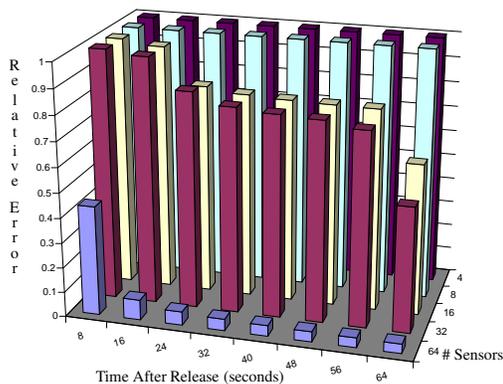


c) With scaling, no bounds

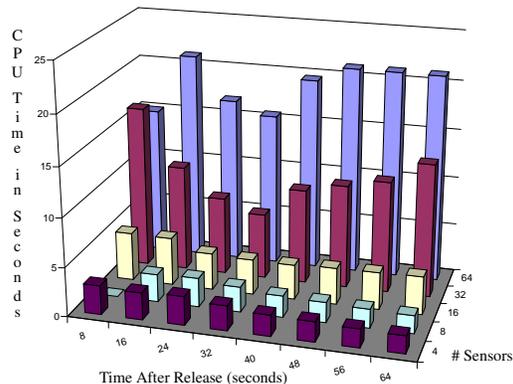


d) With scaling and bounds

Figure 16.11. Comparison of the inversion result for 32 sensors at time $t = 40$ seconds after the initial release at $t = 0$ seconds for the source shown in Figure 16.9. Note the different scales of the z-axis in each case!



a) Relative Errors (16.5.39)



b) CPU Times

Figure 16.12. Relative error (16.5.39) and CPU time for online inversions for a 32×32 , $n_u = 1024$ source discretization of an impulse release at spatial node 8. The relative errors and CPU times are shown for different numbers of random sensors 4, 8, 16, 32, 64 and for times after initial release of 8 through 64 seconds. **Note:** The order of the number of sensors is reversed between the two charts to improve readability.

release. Note that CPU time for the inversion in the 4-sensor case at $t = 8$ seconds is nearly zero since the numerical Cholesky factorization failed due the ill conditioning of the reduced Hessian G .

One interesting aspect to the results shown in Figure 16.12 is that it shows, in general, that the lower the relative error, the higher the CPU times are. To investigate this further, consider a breakdown of the CPU times for two different inversions which are shown in Table 16.1.

As shown in Table 16.1, while there is a slight increase in the CPU time for preprocessing the QP from 0.63 to 1.09 seconds when going from 4 sensors to 64 sensors, the bulk of the time is spend in the dual, active-set QP solver QPSchur. The reason that the QP preprocessing took longer from the 4-sensor to the 64-sensor case was the increase in the number of total sensor data from $n_s = 32$ to $n_s = 512$. The 4-sensor QP solve took only 2.92 seconds while the 64-sensor QP solve was cut short at the 20 seconds time limit (but was very close to optimality). The reason that the 64-sensor QP solve is much more expensive than the 4-sensor QP solve is most likely due to the fact that the 64-sensor QP has many more active degenerate¹ variable bounds at the solution than the 4-sensor QP. It is well known that both active-set and interior-point methods show degraded performance in the presence of large number of degenerate inequality constraints. The reason that the 64-sensor QP has many more degenerate bounds is that the excellent unconstrained solution has most of the source u parameters near zero away from impulse source. This can be seen, for example, in Figure 16.8(e).

¹Here we use the term *degenerate* to denote active inequality constraints that have very small Lagrange multiplier values.

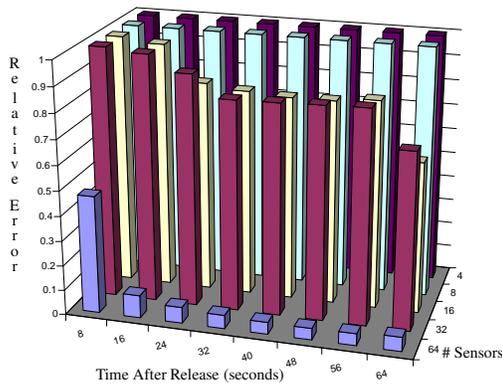
Category	CPU time (seconds)	
	4 sensors	64 sensors
Reading in \hat{D}	0.102	1.603
Reading in \hat{y} , \hat{Q}_y and other vector data	0.009	0.012
Computing $g = \hat{D}\hat{Q}_y\hat{y}$	0.000128	0.00134
Computing $(\hat{Q}_y)^{1/2}$	0.0025	0.0382
Computing $\hat{f} = (\hat{Q}_y)^{1/2}\hat{D}$	0.0026	0.0382
Computing $\hat{S} = \hat{f}^T\hat{f}$	0.048	0.613
Computing $G = \hat{S} + Q_u$	0.0001	0.0001
Factoring G	0.393	0.400
Total for QP preprocessing	0.63	1.09
Solving reduced QP (QPSchur)	2.29	20.01
Total CPU time	2.92	21.10

Table 16.1. Breakdown of CPU times for online inversion of a 32×32 , $n_u = 1024$ source discretization of an impulse release at spatial node 8 using 4 and 64 spatial sensors.

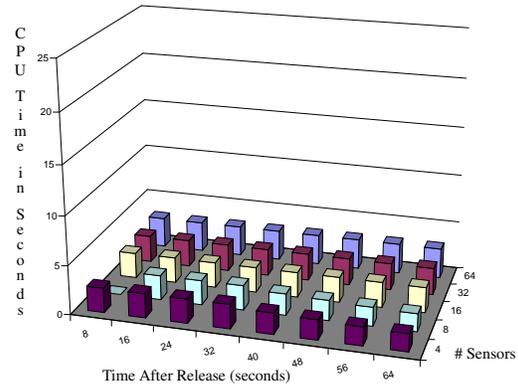
The fact that the QP solve becomes much more expensive as the quality and quantity of the sensor data improves seems to be a disturbing trend at first glance. However, note that as the quality and quantity of the sensor data improves, the need for imposing bounds on the solution is diminished as shown in Figure 16.10(a). Therefore, why not limit the CPU time that the QP is allowed to consume? QPSchur implements a dual, active-set algorithm that starts from an unconstrained solution and then adds violated variable bounds. Therefore, each iteration of the QPSchur algorithm will simply improve on the unconstrained solution and the iterations can be terminated at any time which will result in a reasonable solution for u . When the quality and quantity of the sensor data is low, limiting the QP solve time will not significantly affect the QP solve since the QP solve is fast anyway. Alternatively, when the quality and quantity of the sensor data is high, limiting the QP solve time will not affect the overall inversion quality since the unconstrained QP results in a high quality inversion anyway.

Figure 16.13 shows the relative errors and CPU times for limiting the QP solve time to 2.0 seconds. By comparing Figure 16.12(a) and Figure 16.13(a), one can see that the relative errors are almost indistinguishable but the CPU times in Figure 16.13(b) are all below about 3.0 seconds total.

By looking at the CPU times in Table 16.1 we can also now see the effects that scaling the objective and adding bounds have on the computational cost of solving the online inversion problem. If the objective scaling in (16.4.36) is not used (i.e. zero is substituted for $(\bar{y}_l)_{(i)}$ in (16.4.36)), then \hat{Q}_y is constant and the reduced Hessian G can be computed and factored online. By computing and factoring G offline, the online preprocessing time for the 64-sensor case would be reduced to about 0.013 seconds. Here we have ignored the CPU time to read in \hat{D} since this would be stored in core in successive QP solves. If bounds were



a) Relative Errors (16.5.39)



b) CPU Times

Figure 16.13. Relative error (16.5.39) and CPU time for online inversions for a 32×32 , $n_u = 1024$ source discretization of an impulse release at spatial node 8 with the QP solve limited to 2.0 seconds. Compare to Figure 16.12

eliminated from the formulation, then the QP can be solved as $u = -G^{-1}g$ with a single $O((n_u)^2)$ back-solve with the factors of G . This cost would be similar to the $O((n_u)^2)$ computation of $G = \hat{S} + Q_u$ which is about 0.0001 seconds. Therefore, without scaling the objective function and without imposing bounds, the total cost of solving the $n_u = 1024$ online problem would be reduced to about 0.013 seconds total. The dominant cost here would then be the 0.012 seconds for linear-time file I/O² reading in the vector data for the problem. The remaining $O(n_u n_s)$ and $O((n_u)^2)$ computations for $g = \hat{D}\hat{Q}_y \hat{y}$ and $u = -G^{-1}g$ would consume only about 0.001 seconds. Therefore, without scaling and bounds, one could potentially solve a $n_u = 1024 \left(\frac{1.0\text{sec}}{0.001\text{sec}}\right)^{1/2} \approx 32,000$ parameter problem in 1.0 second! Note that $n_u = 32,000$ corresponds to approximately a $32 \times 32 \times 32$ source discretization of a 3D spatial version of this problem where the 2D floor is discretized into 32×32 spatial source coefficients with 32 temporal coefficients. These results show that if there is sufficient sensor data, then the unscaled, unconstrained QP formulation yields quite good inversion results and therefore a fairly fine source discretization of the 3D problem can be considered.

16.5.2.4 Conditioning and Regularization

Here we discuss the issues of conditioning of the problem and the influence of regularization. First, note that the conditioning of the reduced Hessian G , which is indicative of whether the optimization problem is well posed or ill-posed, has no influence on the cost of computing or factoring G since a dense Cholesky

²In our prototype implementation we read ASCII text files to read in sensor data which requires may expensive string to double conversions. In a real implementation, the files would use binary data and the file I/O would be much faster in practice.

factorization is used. Because QPSchur uses a dual, active-set algorithm, the conditioning of G only weakly effects the cost of solving the bound-constrained reduced QP subproblem, and then only when there are significant numbers of degenerate bound inequalities. Therefore, the cost in solving the inversion problem is not so directly effected by the conditioning of the problem but the reduced Hessian must not be too ill-conditioned or the numerical Cholesky factorization will fail as happened in the 4-sensor case at $t = 8$ seconds as shown in Figure 16.13(b).

In each case shown in this chapter, the regularization term $\frac{\beta}{n_u} \sum_{i=1}^{n_u} u_i^2$ was used with $\beta = 1 \times 10^{-5}$. This form of the regularization, combined with the weighted matching terms meant that the effect of the regulation parameter β was approximately the same independent of the magnitude of the sensor readings or the total number of sensors n_s or the total number of source parameters n_u . With this value for the regularization, the condition number of G in the infinity norm $\kappa_\infty(G)$ varied from about 1×10^8 when there was abundant sensor data to 1×10^{16} when there was little sensor data. Experiments where done in which β was varied to determine its influence on the inversion. When β was decreased much below $\beta = 1 \times 10^{-5}$, more inversions failed due to extreme round-off in the Cholesky factorization. Increasing β some (say to $\beta = 1 \times 10^{-4}$) has little impact on the quality of the inversion but increasing it much more resulted in degraded inversion quality and resulted in increases in the relative error as computed by (16.5.39)

As stated earlier in Section 16.5.2.1, an increase in the number of temporal snapshots did not greatly improve the quality of the inversion since the flow was convection dominated. This can be seen very clearly in the conditioning of the reduced Hessian G . Consider the $n_u = 256$ source discretization problem described in Section 16.5.2.1. When only 4 sensors are used with only one sensor snapshot reading (i.e. $n_s = 32$ but $\hat{D}^T \hat{Q}_y \hat{D}$ is only rank 4 out of 256 rows and columns), the conditioning of the reduced Hessian is $\kappa_\infty(G) = 3.8 \times 10^{13}$. However, when 64 sensors are used and after eight sensor snapshot readings (i.e. $n_s = 512$ and $\hat{D}^T \hat{Q}_y \hat{D}$ is full rank in theory), the conditioning of G only improves to $\kappa_\infty(G) = 1.7 \times 10^{10}$. What this shows is that collecting more sensor snapshot readings does not really result in much new information and does not dramatically improve the quality of the inversion and this trend is seen in Figure 16.6, Figure 16.7, Figure 16.10, and Figure 16.12(a).

16.6 Summary and Conclusions

Here described is an approach for decomposing the repeated solution of a class of QP problems into offline and online subproblems. The general formulation for the class of QPs we consider encompasses several different types of inversion and least-squares problems with both steady-state and transient linear state models. Our method exploits the properties that each QP to be solved has the same linear state constraint matrices $A \in \mathbf{R}^{n_y \times n_y}$ and $B \in \mathbf{R}^{n_y \times n_u}$, and has a diagonal objective function Hessian $Q_y \in \mathbf{R}^{n_y \times n_y}$ with the same nonzero structure along the diagonal (which is represented as a mapping matrix $C \in \mathbf{R}^{n_s \times n_y}$). We exploit these properties by splitting the computation of the solution of the QP into offline and online components. This approach allows for the “real time” solution of a relatively broad class of these QPs using modest computing resources (at least for the online computation).

Offline, a compressed sensitivity matrix $\hat{D} = -CA^{-1}B \in \mathbf{R}^{n_s \times n_u}$ is computed that only depends on the state constraint matrices A and B , and the structure of the state objective Hessian matrix represented by C . The offline formation of \hat{D} can be very computationally expensive, can use either forward or adjoint linear

solver methods, and can be computed using large-scale iterative linear solvers on a distributed-memory super computer.

The cost of the offline computation of \hat{D} scales as $O(n_u)$ for the forward approach and as $O(n_s)$ for the adjoint approach. This means that for typical inversion problems where $n_u > n_s$ that adjoint methods may be preferable.

Each particular online QP can have a different compressed diagonal objective Hessian matrix $\hat{Q}_y \in \mathbf{R}^{n_s \times n_s}$, compressed target state vector $\hat{y} \in \mathbf{R}^{n_s}$, and regularization Hessian matrix $Q_u \in \mathbf{R}^{n_u \times n_u}$. Given this data, the reduced gradient $g = \hat{D}^T \hat{Q}_y \hat{y} \in \mathbf{R}^{n_s}$ and the reduced Hessian $G = \hat{D}^T \hat{Q}_y \hat{D} + Q_u \in \mathbf{R}^{n_s \times n_s}$ are formed online and then a bound-constrained reduced QP subproblem with n_u variables is solved.

The cost of the online computation scales in general as $O(n_u^3)$ and it independent of the size of the state space n_y . The online computations never require even a single linear solve involving the state Jacobian A or its adjoint A^T . Most notably, the online problem can be solved on a single processor machine.

The above described approach was demonstrated on a steady-state inversion problem involving a 2D airport terminal model where the state constraint matrices were produced using the CFD code MPSalsa. Inversion results were reported for $n_u = 256$ and $n_u = 1024$ inversion parameters. While the offline cost of computing the compressed sensitivity matrix \hat{D} was considerable (e.g. about one day on a linux machine for the $n_u = 1024$ case), the online costs were reduced to less than 3.0 seconds for the $n_u = 1024$ inversion on a single 2.5 MHz Linux workstation.

These results clearly demonstrate that repeatably solving large-scale QP problems of the type considered here in “real time” is very tractable using the proposed offline/online approach given current computer technology.

16.7 Recommendations and Future Work

Several different avenues to pursue to continue this work which are described below.

- The numerical experiments indicated that little new information was gained from accumulating more and more sensor snapshot readings at different times and that sensor number and location are much more important. This suggests that the best way to make use of a limited number of sensors is to move the sensors around in some predetermined way to cover the spatial domain. For example, by just moving 4 sensors around the same quantity and quality of sensor data could be accumulated in 16 snapshot readings has is given with 64 sensors. As long as the locations of the sensors was predetermined as a function of time, the compressed direct sensitivity matrix \hat{D} can still be computed offline.
- For a large number of inversion parameters n_u the $O(n_u^3)$ cost of solving the online reduced QP subproblem becomes prohibitive and threatens the “real time” capability to solve these problems. In this cases, more attention needs to be focused on the solution of the reduced QP subproblem.

Possible areas to pursue include:

- Exploiting the structure of \hat{D} : In our prototype implementation, we treated \hat{D} as a dense matrix but in fact this is a block lower triangular matrix as shown in Figure 16.3. Therefore, one could save approximately half the flops involving \hat{D} if a more specialized, structure-exploiting implementation involving \hat{D} was used.
- Using SMP BLAS: In our prototype implementation, we used just a single processor implementation of the BLAS. However, good SMP implementations of the BLAS are available and therefore one could see good parallel speedup to a moderate number of processors on an SMP machine.
- Using approximate reduced Hessians instead of recomputing them from scratch every time: This approach could be particularly effective when solving a sequence of related QPs for a transient inversion problem.
- Using warm-start estimates of the optimal active-set of inequality constraints from the last QP subproblem.
- Using a primal QP solver instead of the dual QP solver QPSchur. A primal QP solver may allow for better premature termination of the the QP algorithm that still results in sufficient quality inversion results.
- Using an interior-point algorithm to approximately solve the reduced QP. An interior-point method would require an $O(n_u^3)$ Cholesky factorization at each interior-point iteration but each of these factorizations would utilize level-3 BLAS for peak performance. Premature termination of the algorithm would guarantee that the bounds in (16.2.13) would not be violated.
- Solving a sequence of inversion problems with increasing refinements of the source discretization and use the initial guess of the active-set from a previous problem for the next finer problem: For example, the results from a 16×16 inversion could be used for the active-set guess for a 32×32 inversion. Then, a primal active-set QP solver like QPOPT may be able to take this initial guess for the active set and very rapidly find the optimal active set in $O(n_u(n_s)^2) + O((n_u - n_{\text{act}})^3)$ time. Here the key is that G need not be formed explicitly or factored.
- The computation of the compressed sensitivity matrix \hat{D} can consume considerable computational resources, especially for transient problems. While this computation does not impact the cost of the online computations it does represent a real cost that can be a hindrance if it is too expensive. The forward and adjoint approaches that solve for $D = A^{-1}B \in \mathbf{R}^{n_y \times n_u}$ and $T = A^{-T}C^T \in \mathbf{R}^{n_y \times n_s}$, respectively, can both take advantage of block linear solver methods such as block GMRES. The use of a block solver may reduce the considerable cost of the offline computations by an order of magnitude or more in some cases.

Chapter 17

Real-Time Identification of Airborne Contaminants

V. Akçelik ,G. Biros, A. Drăgănescu, O. Ghattas, J. Hill, B. van Bloemen Waanders

17.1 Background

Recent years have seen sustained progress in PDE solvers and large-scale optimization algorithms, along with a rapid rise in computing capability. Accompanying these advances has been a growing interest in simulation-based optimization in such diverse areas as aerodynamics, atmospheric and geosciences, the chemical process industry, the environment, homeland security, infrastructure, manufacturing, medicine and physics. Specialized optimization methods that are tailored to the structure of the PDE constraints in these problem classes are being developed to address the need for PDE-constrained optimization across many of the active applications areas.

One class of simulation-based optimization problems are *inverse problems*, in which the goal is to reconstruct the missing, uncertain, unknown, or errant data from sparse observations and measurements over a finite time interval. The reconstruction must be *model-based* such that the reconstructed input data is mapped to the observed measurements in a manner that is consistent with a PDE *forward* simulation model. Once an estimate of the input data has been constructed, it can be used to initialize forward simulations that predict future system behavior over an appropriate time horizon. The “observe–invert–predict” cycle is then repeated for the next time interval of observations, and so on.

Many applications require predictions to be issued rapidly. Some examples include hazard assessment, emergency response, treaty verification, structural health monitoring, image-driven surgery, weather forecasting, geophysical exploration, and closed-loop process control, to name just a few. Historically, this meant that simulation accuracy and resolution were sacrificed for speed. Where rapid prediction and response are mandated, the supporting simulations have had to revert from high-resolution, high-fidelity

three-dimensional PDE models back to simplified models such as lookup tables, algebraic models, or one- or two-dimensional PDEs.

However, in recent years it has become meaningful, and in many cases imperative, to contemplate simulations of complex physical systems that are *both* rapid and highly-resolved. This has been motivated by advances in sensing technologies, deployment of very high bandwidth networks, and the availability of terascale supercomputers. To capitalize on this emerging infrastructure, a central challenge facing computational scientists is the construction of robust scalable parallel algorithms for near-real time solution of the underlying data-driven inverse problems. Unfortunately, inverse problems are often much more difficult to solve than corresponding forward simulations, because inverse problems

- usually require *numerous* repeated forward simulations;
- are usually *ill-posed* despite the well-posedness of the forward problem; and
- are *boundary value problems in four-dimensional space-time*, despite the initial-value, time-marching character of the forward problem.

Nevertheless, there is a pervasive need in many application areas for *near-real time, high-fidelity inversion*. The development of scalable parallel algorithms for this task is the goal of this chapter.

Although our approach is general and widely applicable, we have chosen a specific driving application to instantiate and evaluate our algorithms and implementation. We focus on the localization of airborne contaminant releases in regional atmospheric transport models from sparse observations [18], in time scales short enough for predictions to be useful for hazard assessment, mitigation, and evacuation procedures. In particular, our goal is model-based rapid reconstruction — via solution of a large-scale inverse problem — of the unknown initial concentration of the airborne contaminant in a convection-diffusion transport model, from limited-time spatially-discrete measurements of the contaminant concentration, and from a velocity field as predicted, for example, by a mesoscopic weather model.

Mathematically, transport of the contaminant is described by the convection-diffusion equation

$$\begin{aligned} \frac{\partial u}{\partial t} - \mathbf{v}\Delta u + \mathbf{v} \cdot \nabla u &= 0 \quad \text{in } \Omega \times (0, T), \\ \mathbf{v}\nabla u \cdot \mathbf{n} &= 0 \quad \text{on } \Gamma \times (0, T), \\ u &= u_0 \quad \text{in } \Omega \times \{t = 0\}, \end{aligned} \tag{17.1.1}$$

where $u(x, t)$ is the contaminant concentration field, $u_0(x)$ is the initial concentration that, together with the velocity field $\mathbf{v}(x, t)$, drives the system, and \mathbf{v} is the diffusion coefficient. We seek to reconstruct the initial concentration u_0 from measurements of the concentration u over a short time horizon, taken at a small number of sensor locations throughout the domain. Then — using the just-reconstructed initial concentration — we can issue predictions of the longer-time transport of the contaminant plume throughout the region.

Inverse problems for convective-diffusive transport of this type arise in several settings: characterization of pollutants in the atmosphere, unintentional catastrophic accidents involving, for example, chemical plants, or intentional releases of hazardous chemical or biological agents. Although studies have been conducted

of the sensitivity of chemical concentration with respect to source terms or observer placement, [79, 122, 127, 152], very little work has been done on reconstruction of initial concentrations via solution of an inverse problem.

In Section 2 we formulate the inverse problem as an output least squares optimization problem with a convection-diffusion PDE constraint. First order optimality conditions produce a coupled system of partial differential-algebraic equations, which includes the initial value convection-diffusion PDE, the *terminal-value* adjoint convection-diffusion PDE, and an algebraic equation for the initial concentration. As mentioned above, this system is an ill-posed boundary value problem in 4D space-time, and typical problem sizes of interest present a significant challenge for rapid solution. To overcome the four-dimensionality of this system, we invoke a block elimination that reduces the system to one in just the (3D) spatially-discretized initial concentration variable u_0 . Unfortunately, the operator for this reduced system is non-local and cannot even be formed for the problems we target, even with petascale computing resources. Fortunately, the action of the reduced operator on a vector can be formed by solving a pair of forward/adjoint convection-diffusion PDEs, and the spectral character of the reduced operator (as for many inverse problems) guarantees that a Krylov method applied to this system converges in a mesh-independent number of iterations.

However, a constant number of iterations independent of mesh size is, by itself, not sufficient for real-time applications: *the constant itself must be reduced* so that no more than a few iterations — and hence forward/adjoint simulations — are needed to solve the inverse problem. This requires a scalable and effective preconditioner, which is particularly challenging because the inverse operator is never formed. In Section 3 we present a parallel multigrid preconditioner designed to reduce the number of Krylov iterations for inverse problems. Unlike PDE operators, inverse operators are compact, and their spectral properties are different from those of differential operators. As a result, standard multigrid smoothers are not applicable for inverse operators. Instead, special-purpose smoothers that are tailored to their spectral properties must be employed, and these are presented in Section 3. Section 4 provides a prototype inversion scenario: localization of the release of a contaminant in the Los Angeles harbor from short-term measurements of its transport by onshore winds, followed by longer-term prediction of the transport of the contaminant throughout the Greater LA Basin. We also provide results on the performance and scalability of our inversion algorithm. Our results demonstrate that due to high parallel and algorithmic efficiency, inverse problems with 17 million initial concentration unknowns, and 8.7 billion total space-time unknowns, can be solved in less than 30 minutes on 1024 processors of an Alphaserver EV68-based system. The time taken is just 18 times that of a *single* forward transport simulation. Moreover, inverse problems with 135 million initial concentration parameters — and 139 billion total space-time unknowns — are solved in less than 5 hours on the same number of processors.

Ultimately, our results demonstrate that, with careful attention to the design of scalable parallel algorithms, high-resolution inverse transport problems can be solved in “real time,” i.e. in time scales feasible for simulation-based hazard assessment and response. More generally, for inverse problems characterized by other classes of forward simulations, the turn-around time will, of course, depend on the complexity of the forward simulation. However, *our results indicate that model-based reconstruction of incomplete initial conditions can be achieved in a small multiple of the cost of the forward simulation, even when millions of uncertain parameters must be estimated.*

17.2 Formulation and optimality conditions

In this section we give details on the mathematical formulation of the inverse problem, its discretization, and the overall strategy for its numerical solution.

Given observations of the concentration $\{u_j^*\}_{j=1}^{N_s}$ at N_s locations $\{x_j\}_{j=1}^{N_s}$ inside a domain Ω , we wish to estimate the initial concentration $u_0(x)$ that leads to the closest reproduction of the observed concentrations. The inverse problem is formulated as a constrained, regularized, least squares optimization problem:

$$\begin{aligned} \min_{u, u_0} \mathcal{J}(u, u_0) &\stackrel{\text{def}}{=} \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T \int_{\Omega} (u - u^*)^2 \delta(x - x_j) dx dt + \frac{\beta}{2} \int_{\Omega} u_0^2 dx, \\ \text{subject to} \quad &\frac{\partial u}{\partial t} - \nu \Delta u + v \cdot \nabla u = 0 \quad \text{in } \Omega \times (0, T), \\ &\nu \nabla u \cdot n = 0 \quad \text{on } \Gamma \times (0, T), \\ &u = u_0 \quad \text{in } \Omega \times \{t = 0\}. \end{aligned} \tag{17.2.2}$$

The first term in the objective functional \mathcal{J} represents a least-squares misfit of predicted concentrations $u(x_j)$ with observed concentrations $u^*(x_j)$, where the delta function localizes the u and u^* fields to the sensor locations. The second term in \mathcal{J} , scaled by the constant $\beta/2$, is a regularization term that introduces well-posedness in the inverse problem. In the absence of the regularization term, the inverse problem would be ill-posed even if measurements are available at all points in space and time: small perturbations in measurements, e.g. due to sensor faultiness or data transmission, would produce exponentially large errors in the recovered solution. It should also be noted that, even though this regularization term **explicitly** controls only the size of u_0 , that is, its L^2 -norm, the resulting recovered solution will have controlled square variation as well (see [86]). It is also the case that we cannot hope to recover components of the initial concentration that are much more oscillatory than dictated by the spacing of the sensors. Therefore, oscillatory components of u_0 lie in the null space of the inverse operator and, in the absence of regularization, will appear as arbitrary noise in the reconstructed initial concentration field.

The constraints in the optimization problem (17.2.2) are the contaminant transport convection-diffusion equation, boundary condition, and initial condition. The transport of the pollutant is driven by the initial conditions, the diffusion, and the velocity field. In practice, the velocity field would be provided by a regional numerical weather prediction model such as MM5 [159]. For our present purposes, however, we are interested in assessing the real-time viability and algorithmic and parallel scalability of our inversion method. For simplicity, we employ a steady laminar incompressible Navier-Stokes solver to generate wind velocity fields over a terrain of interest.

The inverse problem then is to determine the initial concentration field $u_0(x)$, and the resulting space-time evolution of the concentration $u(x, t)$, by solving the optimization problem (17.2.2). First-order necessary conditions for optimality, the *KKT conditions*, may be derived by introducing a Lagrangian functional:

$$\begin{aligned} \mathcal{L}(u, u_0, p) &\stackrel{\text{def}}{=} \frac{1}{2} \sum_{j=1}^{N_s} \int_0^T \int_{\Omega} (u - u^*)^2 \delta(x - x_j) dx dt + \frac{\beta}{2} \int_{\Omega} u_0^2 dx \\ &+ \int_0^T \int_{\Omega} \left(p \frac{\partial u}{\partial t} + \nu \nabla u \cdot \nabla p + p v \cdot \nabla u \right) dx dt + \int_{\Omega} p(u - u_0) dx, \end{aligned} \tag{17.2.3}$$

in which the *adjoint concentration* $p(x, t)$ is used to enforce the convection-diffusion equation and initial condition. Requiring stationarity of the Lagrangian \mathcal{L} with respect to p , u , and u_0 , respectively yields the KKT conditions, which consist of:

The forward convection-diffusion problem

$$\begin{aligned} \frac{\partial u}{\partial t} - v\Delta u + v \cdot \nabla u &= 0 \quad \text{in } \Omega \times (0, T), \\ v\nabla u \cdot n &= 0 \quad \text{on } \Gamma \times (0, T), \\ u &= u_0 \quad \text{in } \Omega \times \{t = 0\}. \end{aligned} \quad (17.2.4)$$

The adjoint convection-diffusion problem

$$\begin{aligned} -\frac{\partial p}{\partial t} - v\Delta p - \nabla \cdot (pv) &= -\sum_{j=1}^{N_s} (u - u^*)\delta(x - x_j), \quad \text{in } \Omega \times (0, T) \\ (v\nabla p + vp) \cdot n &= 0, \quad \text{on } \Gamma \times (0, T), \\ p &= 0 \quad \text{in } \Omega \times \{t = T\}. \end{aligned} \quad (17.2.5)$$

The initial concentration equation

$$\beta u_0 - p|_{t=0} = 0 \quad \text{in } \Omega. \quad (17.2.6)$$

The equations in (17.2.4) are the original forward convection-diffusion transport problem for the contaminant field. The adjoint convection-diffusion problem (17.2.5) resembles the forward problem, but with some essential differences. First, it is a terminal value problem, the adjoint p is specified at the final time $t = T$, rather than an initial value problem. Second, the convection is directed backward along the streamlines. Finally, it is driven by a source term given by the negative of the misfit between the predicted and measured concentrations at sensor locations. The initial concentration equation (17.2.6) is, in the case of L^2 regularization, an algebraic equation. Together, (17.2.4), (17.2.5), and (17.2.6) furnish a coupled system of linear PDEs for (u, p, u_0) . The principal difficulty in solving this system is that, though the forward and adjoint transport problems are parabolic-hyperbolic problems, the KKT optimality system is a *coupled boundary value problem in 4D space-time*.

To simplify discussion of solution approaches, we introduce the operators A, T, B and R . Here, A denotes the forward transport operator and A^{-1} its inverse; T extends a spatial field at initial time into space-time; B is an observation operator that localizes space-time to points at which sensors are placed; R is the regularization operator (in the present case the identity); A^* is the adjoint transport operator and A^{-*} its inverse; and T^* restricts a space-time field to a spatial field at $t = 0$. With these definitions, we can write the KKT conditions in operator form:

$$\begin{bmatrix} B & 0 & A^* \\ 0 & \beta R & -T^* \\ A & -T & 0 \end{bmatrix} \begin{bmatrix} u \\ u_0 \\ p \end{bmatrix} = \begin{bmatrix} Bu^* \\ 0 \\ 0 \end{bmatrix} \quad (17.2.7)$$

Special-purpose Krylov solvers and parallel preconditioners can be very effective at solving discretized versions of optimality systems such as (17.2.7) for optimization problems constrained by *steady-state*

PDEs [43, 44]. Here, however, the 4D space-time nature of (17.2.7) presents prohibitive memory requirements for large scale problems. Solution of (17.2.7) in its *full-space* form is essentially intractable for such problems using present computing resources. Instead, we pursue a *reduced-space* method that amounts to a block elimination combined with a matrix-free Schur complement solver.

Eliminating the concentration u and forward transport equation (third row of (17.2.7)) and adjoint concentration p and adjoint transport equation (first row of (17.2.7)) from the KKT optimality system, we obtain the Schur complement system for the initial concentration u_0 :

$$Hu_0 = g, \quad (17.2.8)$$

where the *reduced Hessian* (or inverse) operator H is defined by

$$H \stackrel{\text{def}}{=} T^*A^{-*}BA^{-1}T + \beta R, \quad (17.2.9)$$

and the *reduced gradient* g is defined by

$$g \stackrel{\text{def}}{=} T^*A^{-*}Bu^*.$$

It is immediately clear that H is a symmetric and boundedly invertible operator. Thus, the optimization problem has a unique solution for non-vanishing β .

Notice that H is a non-local operator and its explicit construction is unfeasible. For example, for the problem with 139 billion space-time unknowns solved in Section 4, H is of dimension 135×10^6 by 135×10^6 . Thus, storing H would require about 10^{23} bytes of memory. Moreover, forming H would require 135 million solutions of forward convection-diffusion transport equations; on the 1024 processor Alpha system we used for the numerical experiments of Section 4, this would require over 400 years of computing time. While explicit formation of H and its singular value decomposition constitutes an attractive and popular approach for small-scale inverse problems [110], alternative approaches are essential for large-scale problems, and in particular those for which near real-time response is mandated.

Therefore, we opt to solve the discretized form of (17.2.8) using the preconditioned Conjugate Gradient (CG) method. H is never formed explicitly; instead, we compute $w = Hv$, the action of the reduced Hessian on a given spatial field v , in matrix-free fashion as follows. (i) Set $u_0 = v$ and solve the forward transport equation (17.2.4) to obtain the concentration evolution u . (ii) Compute the misfit between the measurements u^* and the predicted concentrations u at the sensor locations. (iii) Use this misfit as a source to solve the adjoint transport equation (17.2.5) backward in time to obtain $p|_{t=0}$, the adjoint at $t = 0$. (iv) Set $w = \beta v - p|_{t=0}$. Therefore, each application of the reduced Hessian requires solving two transport equations, one forward in time and one backward. Besides u_0 , we need to store the entire time history of u (if we have measurements over the entire time interval $(0, T)$) but *only* at the sensor locations. In contrast with full-space methods, we avoid storing the forward and adjoint concentration time histories. Thus, the memory requirements for the inverse problem (17.2.2) are similar to those of the forward problem (17.1.1).

The overall computational cost of the (unpreconditioned) CG method is the work per iteration — dominated by the two transport equations solutions — multiplied by the number of CG iterations. The latter depends on the condition number of the reduced Hessian. One can show that, for fixed β , H is a compact perturbation of the identity, and its discrete version, denoted by H_h (h represents the discretization parameter), has a mesh-independent condition number [85]. Furthermore, its spectrum has a small number of clusters; it collapses exponentially onto β . Therefore, if we use a Krylov method, such as CG, to solve

(17.2.8), the number of iterations for a specific relative residual reduction will also be mesh-independent. Thus the number of required CG iterations grows linearly with the number of space-time unknowns (optimal order), a fact that we have also verified numerically (see Section 17.4, and also [86]). Thus, mesh-independent convergence comes for free, with the constant mildly deteriorating with $\beta \rightarrow 0$. The constant also depends on the Peclet number, the length of the time horizon, the complexity of the velocity field, and the topography.

However, mesh-independence of CG iterations is by itself *not sufficient for problems requiring real-time inversion*. Although algorithmically optimal, the number of unpreconditioned CG iterations is often so large that the cost of solving the inverse problem is equivalent to many tens to hundreds of forward transport solutions, which precludes the use of high-resolution models in the real-time setting. Our goal therefore is to reduce the absolute number of iterations so that the cost is equivalent to a handful of forward transport solves. To achieve this goal, we must *reduce the constant in the complexity estimate*. The immediate idea is to precondition the reduced Hessian system to further decrease the number of CG iterations and, most importantly, reduce the overall wall-clock time. One challenge in constructing suitable preconditioners for the reduced Hessian is the impossibility of explicitly forming this operator for reasons stated above. For this reason, and due to their demonstrated success as preconditioners for second-kind integral operators [106, 126], we pursue multigrid preconditioners.

17.3 Multigrid Preconditioner

Multigrid methods have revolutionized scientific computing, especially for linear systems related to elliptic and parabolic partial differential equations. Such multigrid schemes, however, are not directly applicable to reduced Hessian operators for inverse problems. For this reason, there has been recent interest in developing specific multigrid-like methods for inverse problems (for example see [86, 109, 121, 126, 176]). The main difficulty lies in constructing a proper smoothing operator for the reduced Hessian operator H .

For our problem, the continuous reduced Hessian operator is spectrally equivalent to a Fredholm integral operator of the second kind. Multigrid solvers for such problems have been very successful [106]. The overall algorithm follows the standard multigrid hierarchy: pre-smoothing, restriction to a coarser grid, solution, prolongation back to the fine grid, correction, and post-smoothing. The key aspect is the smoother, which must address the spectrum of the reduced Hessian.

Classical solvers such as Jacobi and Gauss-Seidel work well as smoothers for elliptic PDE operators, where the large eigenvalues of the differential operator correspond to high frequency eigenvectors (see [56]). These methods rapidly eliminate oscillatory components of the numerical error, but are notoriously slow at eliminating the smooth components. The multigrid method can be used to effectively eliminate the smooth error components by iterating on coarser scales.

Unlike elliptic operators, however, the continuous reduced Hessian is a strongly smoothing, compact, and nonlocal operator (hence its discrete version H_h is represented by a dense matrix). Its eigenvector–eigenvalue correspondence is reversed, with large eigenvalues associated with smooth eigenvectors, and small eigenvalues associated with oscillatory eigenvectors. Neither Krylov-subspace methods, nor stationary methods such as Jacobi and Gauss-Seidel, act as smoothers for H_h ; in fact, in

addition to being expensive to apply, they act more as roughers, since the “high energy” (large eigenvalue) components, which correspond to smooth eigenvectors, are typically resolved first, leaving oscillatory components in the error.

The smoothing properties of the continuous reduced Hessian H , combined with its approximation of the discrete counterpart H_h , imply that by decreasing the mesh size, both H and the “high energy”, smooth, eigenvectors are increasingly well represented on the immediately coarser grid. For clarity we denote by H_h the discrete reduced Hessian at resolution h . If we assume that the “coarse” space V_{2h} is embedded into the “fine” space V_h (as is often the case with finite element discretizations), and we denote by $P_h : V_h \rightarrow V_{2h}$ the L^2 -orthogonal projection, then the action of the reduced Hessian H_h on the “coarse” space is well approximated by H_{2h} . If in addition we regard the orthogonal complement $W_{2h} = (I - P_h)V_h$ of V_{2h} in V_h as the space of high frequency functions (this is only approximately true; in fact W_{2h} actually has both smooth and oscillatory components [56]), then the strong smoothing properties of the continuous reduced Hessian imply that

$$H_h \approx \beta(I - P_h) + H_{2h}P_h . \quad (17.3.10)$$

which combined with orthogonality between P_h and $I - P_h$ suggests¹ the following two-level preconditioner M_h [86, 176]:

$$H_h^{-1} \approx M_h \stackrel{\text{def}}{=} \beta^{-1}(I - P_h) + (H_{2h})^{-1}P_h . \quad (17.3.11)$$

Note that the first part of the preconditioner, $\beta^{-1}(I - P_h)$, acts as a smoother, since it removes high frequency components from the residual. If V_h are finite element spaces, and we invert for the initial value given the entire final-time state, it has been shown in [85, 86] that, for h small enough

$$1 - C \frac{h^p}{\beta} \leq \frac{\langle M_h u, u \rangle}{\langle (H_h)^{-1} u, u \rangle} \leq 1 + C \frac{h^p}{\beta}, \quad \text{for all } u \in V_h, u \neq 0, \quad (17.3.12)$$

where $\langle \cdot, \cdot \rangle$ is the L^2 -inner product, p is the convergence order of the forward method ($p = 2$ for piecewise linear polynomials) and C is independent of h . The two-level preconditioner defined in (17.3.11) can be turned into a multi-level preconditioner, denoted by MLAS (**M**ulti-**L**evel **A**dditive **S**chwarz) with a \mathcal{W} -cycle structure, that satisfies (17.3.12) as well (see [86] for details). It should be noted that by simply replacing in (17.3.11) the call to $(H_{2h})^{-1}$ with a recursive call to the coarse-level preconditioner one **does not** obtain a multi-level preconditioner with optimal qualities. The statement (17.3.12) shows that the two-level preconditioner becomes increasingly effective at high resolution. In particular, this has an interesting consequence: the number of MLAS-preconditioned CG iterations **decreases** with increasing number of levels, assuming the base-level resolution is fine enough. This is precisely the fact that allows significant speed-ups over unpreconditioned CG.

There are a number of implementation issues to consider. At the coarsest level, the approximate inverse is replaced by an “exact” solve. The mesh size for the coarse level cannot be chosen arbitrarily, and is a function of the regularization parameter β and the compact part of H (see [85]). Since the reduced Hessian is not available explicitly (even at the coarsest scale) the exact coarse solve is performed by the CG solver. The orthogonal decomposition $(I - P_h)$ can be replaced by less expensive projection-like operators, such as

¹Assume that $\beta = 0$. If $v \in V_h$ is decomposed into a smooth $v_s \in V_{2h}$ and an oscillatory $w_o \in W_{2h}$ then $H_h v = H_h v_s + H_h w_o \approx H_{2h} v_s = H_{2h} P_h v$. Furthermore, we can write $H_h = (1 - P_h)H_h(1 - P_h) + P_h H_h(1 - P_h) + (1 - P_h)H_h P_h + P_h H_h P_h \approx P_h H_h P_h$. Then for $\beta \neq 0$, (17.3.10) follows easily.

the standard interpolation-restriction operators from classical multigrid theory. Our preconditioner then becomes

$$H_h^{-1} \approx M_h = \beta^{-1}(I - I_{2h}^h I_h^{2h}) + I_{2h}^h M_{2h} I_h^{2h}, \quad (17.3.13)$$

where $I_{2h}^h : V_{2h} \rightarrow V_h$ is the natural interpolation operator, and $I_h^{2h} : V_h \rightarrow V_{2h}$ is the full-weighting restriction operator defined by $I_h^{2h} = c(I_{2h}^h)^T$ with c chosen so that a constant function is restricted to itself. The operator $\beta^{-1}(I - I_{2h}^h I_h^{2h})$ acts as a smoother, replacing the more expensive $\beta^{-1}(I - P_h)$. Note that the smoothing operator is explicit, symmetric, and sparse; it acts only on initial concentrations, and therefore its application has negligible computational cost compared to the pair of forward/adjoint convective-diffusive transport solves at each CG iteration. Since we coarsen in both temporal and spatial dimensions at the same rate (in our implementation the time-stepping and spatial discretization methods have the same approximation order), the cost of one reduced Hessian–vector multiplication on level l is 2^4 times more expensive than that at the immediately coarser level $l - 1$. Therefore, by using just three levels, the cost of a reduced Hessian-vector product on the finest grid is 256 times the cost of a similar operation at the coarsest level. By using a simple \mathcal{V} -cycle strategy at the finest level we avoid computing the finest-level residual inside the preconditioner. As stated before, CG is used as a direct solver on the coarsest level. We employ the full multigrid framework (i.e. grid sequencing) to compute initial guesses for each level. In the next section, we discuss numerical results.

17.4 Implementation and Numerical Results

We demonstrate our inversion framework on a hypothetical atmospheric contamination event in the Greater Los Angeles Basin (GLAB) region. Using real topographical data and synthesized velocity fields, we conduct numerical experiments in which sparse observations are extracted from forward simulations and subsequently used in the inverse problem. Our implementation builds on PETSc [28] to manage parallel data structures, coordinate different grid resolutions in our multigrid preconditioner, interface with linear solvers and domain decomposition preconditioners, and utilize a range of software services. We first discuss discretization and geometry details and problem setup. We then briefly present numerical results for initial concentration inversions in the GLAB. Finally, we provide parallel and algorithmic scalability results on structured grids without topography.

In our actual implementation, we first discretize the optimization problem (17.2.2), and then write optimality conditions (as opposed to the writing the infinite-dimensional optimality conditions (17.2.7) and then discretizing; in the present case the two are not identical [14]). We employ Streamline Upwind Petrov-Galerkin (SUPG) finite elements [57] in space and a Crank-Nicolson discretization in time. For problems with high Peclet number, stabilized methods such as SUPG are more accurate than standard Galerkin on coarse meshes. We use a logically-rectangular topography-conforming isoparametric hexahedral finite element mesh on which piecewise-trilinear basis functions are defined. Since the Crank-Nicolson method is implicit, we “invert” the time-stepping operator using a restarted GMRES method, accelerated by an additive Schwarz domain decomposition preconditioner, both from the PETSc library.

17.4.1 Source inversion in the Greater Los Angeles Basin

Contaminant transport is modeled over a $360 \text{ km} \times 120 \text{ km} \times 5 \text{ km}$ region in the GLAB. Land surface elevations are obtained at 1 km spacing from the USGS Land Processes Distributed Active Archive Center (GTOPO30 digital elevation model).² The three-dimensional mesh is created from the surface elevations by inserting equally spaced grid points vertically from the surface grid to the top of the domain at 5 km.

To simulate a contamination event, an initial contaminant plume with a Gaussian concentration given by $20\exp(-0.04|x - x_c|)$ is centered at $x_c = (120 \text{ km}, 60 \text{ km}, 0 \text{ km})$. The plume is transported over the GLAB region by solving the convection-diffusion equation (17.1.1) with specified velocity field over a time horizon of 120 minutes. Sensor measurements are taken every 3 minutes to develop a time history from which to invert. For this contaminant, the diffusion coefficient is taken as $\nu = 0.05$. The regularization parameter is fixed at $\beta = 0.01$. The forward and inverse problems are solved on a mesh with $361 \times 121 \times 21$ grid points, representing 917,301 concentration unknowns at each time step. The time step is the same as the sensor recording rate, i.e. 3 minutes, for a total of 40 time steps. Therefore, there are about 74×10^6 total space-time variables in the KKT optimality system (17.2.7).

A velocity field v for the convection-diffusion equation is synthesized by solving the steady-state incompressible Navier-Stokes equations, $\rho(v \cdot \nabla)v + \nabla p - \mu\Delta v = 0$, $\nabla \cdot v = 0$, where p is the fluid pressure and (ρ, μ) are its density and viscosity. To simulate an onshore wind, an inflow Dirichlet boundary condition with $v_x = v_{\max}(z/(5.0 - z_{\text{surface}}))^{0.1}$ and zero for the other components is applied to the $x = 0$ plane, where v_{\max} is specified as 30 km/hr. Traction-free boundary conditions are applied to the outflow plane at $x = 360 \text{ km}$. Traction-free tangential and no-slip normal boundary conditions are applied to the remaining portions of the boundary. An SUPG-stabilized finite element method is employed to solve the Navier-Stokes equations using linear tetrahedral elements derived by subdividing the convection-diffusion hexahedral mesh.

We sample the concentrations from the forward transport simulations on a uniformly-spaced array of sensors, and use them as synthetic observations to drive the inverse problem. Figure 17.1 depicts inversion results for different sensor array densities, along with the actual initial concentration (labeled *Target* in the figure). One of the critical issues is to determine the number of sensors required to resolve the initial concentration. Quantified in Table 17.1, as the sensor density increases, the error between the actual initial concentration and predicted initial concentration is reduced. The relative L^2 norm error for the $21 \times 21 \times 21$ sensor array is 34%, but as can be observed in the final image, the initial concentration is localized very accurately. Recall that due to the non-vanishing regularization parameter and the fixed mesh size, we cannot expect to recover the initial concentration exactly.

The run-time on a modest number of Alphaserver processors (64) is 2.5 hours for the finest sensor array. As the sensor array becomes denser, the number of CG iterations increases, causing an increase in wall-clock time. With new information provided by the additional more finely-spaced sensors, one might expect a quicker inversion due to a less-poorly-posed problem. In fact, the opposite is true: richer information provided by more the finely-spaced sensors provides more energy to the oscillatory components of the residual; the CG solver thus must work harder to recover these solution components. (Note that in this set of experiments we have not used the multigrid preconditioner.)

²http://edcdaac.usgs.gov/gtopo30/dem_img.asp

Table 17.1. Sensitivity of the inversion quality to the sensor array density. The sensitivity of the inversion quality to the sensor density is reported in terms of the relative errors in both the L_2 and ∞ norms, the wall-clock time, and the number of CG iterations. As the number of sensors in each direction increases, the quality of the reconstruction of the initial concentration plume improves. The additional information resulting for a higher sensor density results in more CG iterations. In these simulations, the regularization parameter β was held constant at 0.01 and the diffusion coefficient was fixed at 0.05.

sensor array	$\ e\ _{L_2}^{relative}$	$\ e\ _{\infty}^{relative}$	time	iterations
$6 \times 6 \times 6$	0.793	0.980	2:01:47.11	377
$11 \times 11 \times 11$	0.495	0.870	2:20:02.65	437
$21 \times 21 \times 21$	0.339	0.805	2:33:11.72	484

We have also studied the effect of the regularization parameter, β , on the quality of the inversion, shown in Table 17.2. As β becomes smaller, the problem becomes more ill-posed. This is evident by the increase in both the wall-clock time and the number of CG iterations required for the inversion. However, as the regularization term approaches zero, the optimization problem becomes a simple least-squares misfit problem. The quality of the inversion improves, as is evident by the reduction in the relative errors.

For inversion quality, the inversion algorithm handles a convection-dominated transport problem better than a diffusion-dominated flow, as shown in Table 17.3. As the diffusion coefficient increases and the flow becomes more diffusion-dominant, the quality of the inversion is reduced. Intuitively, we expect this because as the contaminant dissipates by diffusion, no information is provided to the inversion algorithm concerning this dissipation. However, for convection-dominated flow, the inversion algorithm also has the velocity field as additional information.

Finally, the effect of white noise in the sensor readings was studied. In real-world situations noisy sensor readings are the result of uncalibrated or miscalibrated sensors, or malfunctioning sensors. Noisy data was simulated by adding a random amount, up to a maximum percentage of the measured contaminant at each sensor. It is evident from Table 17.4.1 that reasonably noisy data has little effect on the quality of the inversion or the number of CG iterations required to converge to a solution.

What is of ultimate interest is how successful the reconstructed initial field is in predicting the actual transport of the contaminant. Figure 17.2 compares the actual evolution (left) and predicted evolution (right) of the contaminant plume in time. It is evident from this figure that although the reconstructed concentration does not match the actual concentration exactly at $t = 0$, the difference between the two diminishes over time, due to the dissipative nature of the forward convection-diffusion problem.

Table 17.2. Sensitivity of the inversion quality to the choice of regularization parameter, β . The sensitivity of the inversion quality to the choice of regularization parameter is reported in terms of the relative errors in both the L_2 and ∞ norms, the wall-clock time, and the number of CG iterations. As the regularization parameter, β , is reduced, the quality of the inversion improves, when measured by the relative error. However, the problem becomes more ill-posed resulting in more CG iterations and a longer run-time. In these simulations, the an $11 \times 11 \times 11$ array of sensors was employed and the diffusion coefficient was fixed at 0.05.

β	$\ e\ _{L_2}^{relative}$	$\ e\ _{\infty}^{relative}$	time	iterations
1	0.852	0.980	1:10:15.81	212
0.1	0.633	0.940	1:14:00.97	211
0.01	0.495	0.870	2:20:10.29	437
0.001	0.435	0.835	5:24:35.65	1078

Table 17.3. Sensitivity of the inversion quality to the amount of diffusion. The sensitivity of the inversion quality to the amount of diffusion is reported in terms of the relative errors in both the L_2 and ∞ norms, the wall-clock time, and the number of CG iterations. As the transport problem becomes more diffusion dominant (i.e. the diffusion parameter increases), the quality of the inversion degrades, based on both the relative L_2 and infinity norms. In these simulations, the an $11 \times 11 \times 11$ array of sensors was employed and the regularization parameter was fixed at 0.01.

ν	$\ e\ _{L_2}^{relative}$	$\ e\ _{\infty}^{relative}$	time	iterations
0.05	0.495	0.870	2:18:39.94	437
0.10	0.554	0.910	0:51:35.30	140
0.20	0.618	0.935	1:07:38.93	195
0.40	0.673	0.955	1:33:44.02	275

Table 17.4. Sensitivity of the inversion quality to noise in the sensor readings. The sensitivity of the inversion quality to noise in the sensor measurements is reported in terms of the relative errors in both the L_2 and ∞ norms, the wall-clock time, and the number of CG iterations. Based on the relative errors, a small amount of random noise in the sensor readings does not significantly affect the inversion quality. In these simulations, the an $11 \times 11 \times 11$ array of sensors was employed, the diffusion coefficient was constant at 0.05, and the regularization parameter was fixed at 0.01.

Noise	$\ e\ _{L_2}^{relative}$	$\ e\ _{\infty}^{relative}$	time	iterations
0 %	0.495	0.870	2:20:10.29	437
5 %	0.497	0.870	2:40:24.02	506
10 %	0.500	0.875	3:01:52.31	581

17.4.2 Scalability of the Multigrid preconditioner

We next study the parallel and algorithmic scalability of the multigrid preconditioner. In all experiments, we use a regular grid with a constant unidirectional velocity field. This is an important simplification of the problem. Further tests are necessary for the case of more complex and time-dependent velocity fields. The corresponding Peclet number is 3. We take synthetic measurements on a $7 \times 7 \times 7$ sensor array. CG is terminated when the residual of (17.2.8) has been reduced by six orders of magnitude.

Table 17.5 presents fixed-size scalability results. The inverse problem is solved on a $257 \times 257 \times 257 \times 257$ grid, i.e. there are 17×10^6 inversion parameters in (17.2.8) and 4.3×10^9 total space-time unknowns in (17.2.7). Note that while the CG iterations are insensitive to the number of processors, the forward and adjoint transport simulations at each iteration rely on a single-level Schwarz domain decomposition preconditioner, whose effectiveness deteriorates with increasing number of processors. Thus, the efficiencies reported in the table reflect parallel as well as (forward) algorithmic scalability. The multigrid preconditioner incurs non-negligible overhead as the number of processors increases for fixed problem size, since the coarse subproblems are solved on ever larger numbers of processors. For example, on 1024 processors, the $65 \times 65 \times 65$ coarse grid solve has just 270 grid points per processor, which is far too few for a favorable computation-to-communication ratio.

On the other hand, the unpreconditioned CG iterations exhibit excellent parallel scalability since the forward and adjoint problems are solved on just the fine grids. Nevertheless, the multigrid preconditioner achieves a net speedup in wall-clock time, varying from a factor of 2.5 for 128 processors to 1.5 for 1024 processors. Most important, the inverse problem is solved in less than 29 minutes on 1024 processors. This is about 18 times the wall-clock time for solving a single forward transport problem.

Table 17.6 presents isogranular scalability results. Here the problem size ranges from 5.56×10^8 to 1.39×10^{11} total space-time unknowns, while the number of processors ranges from 16 to 1024. Because we refine in time as well as in space, and because the number of processors increases by a factor of 8 with

Table 17.5. Fixed size scalability of unpreconditioned and multigrid preconditioned inversion. Here the problem size is $257 \times 257 \times 257 \times 257$ for all cases. We use a three-level version of the multigrid preconditioner described in Section 17.3. The variables are distributed across the processors in space, whereas they are stored sequentially in time (as in a multicomponent PDE). Here *hours* is the wall-clock time, and η is the parallel efficiency inferred from the runtime. The unpreconditioned code scales extremely well since there is little overhead associated with its single-grid simulations. The multigrid preconditioner also scales reasonably well, but its performance deteriorates since the problem granularity at the coarser levels is significantly reduced. Nevertheless, wall-clock time is significantly reduced over the unpreconditioned case.

CPUs	no preconditioner		multigrid	
	hours	η	hours	η
128	5.65	1.00	2.22	1.00
512	1.41	1.00	0.76	0.73
1024	0.74	0.95	0.48	0.58

each refinement of the grid, the total number of space-time unknowns is not constant from row to row of the table; in fact it doubles. However, the number of grid points per processor does remain constant, and this is the number that dictates the computation to communication ratio. For ideal overall (i.e. algorithmic + parallel) scalability, we would thus expect wall-clock time to double with each refinement of the grid. Unpreconditioned CG becomes too expensive for the larger problems, and is unable to solve the largest problem in reasonable time. The multigrid preconditioned solver, on the other hand, exhibits very good overall scalability, with overall efficiency dropping to 95% on 128 processors and 86% on 1024 processors, compared to the 16 processor base case. From the fixed-size scalability studies in Table 17.5, we know that the parallel efficiency of the multigrid preconditioner drops on large numbers of processors due to the need to solve coarse problems. However, the isogranular scalability results of Table 17.6 indicate substantially better multigrid performance. What accounts for this? First, the constant number of grid points per processor keeps the processors relatively well-populated for the coarse problems. Second, the algorithmic efficacy of the multigrid preconditioner improves with decreasing mesh size (as predicted by (17.3.12)); the number of iterations drops from 8 to 5 over two successive doublings of mesh resolution. The largest problem exhibits a factor of 4.6 reduction in CG iterations relative to the unpreconditioned case (5 vs. 23). This improvement in algorithmic efficiency helps keep the overall efficiency high.

17.5 Conclusions

We have presented a methodology for solving large-scale inverse problems, determining the unknown initial condition data from sparse observations in a model-consistent manner. The methodology has been

Table 17.6. Isogranular scalability of unpreconditioned and multigrid preconditioned inversion. The spatial problem size per processor is fixed (stride of 8). Ideal speedup should result in doubling of wall-clock time. The multigrid preconditioner scales very well due to improving algorithmic efficiency (decreasing CG iterations) with increasing problem size. Unpreconditioned CG is not able to solve the largest problem in reasonable time.

grid size	problem size		CPUs	no preconditioner		multigrid	
	u_0	(u, p, u_0)		hours	iterations	hours	iterations
129^4	2.15E+6	5.56E+8	16	2.13	23	1.05	8
257^4	1.70E+7	8.75E+9	128	5.65	23	2.22	6
513^4	1.35E+8	1.39E+11	1024	—	—	4.89	5

instantiated in the context of inverse convection-diffusion transport problems. The main difficulty is that the optimality system is a boundary value problem in 4D space-time, even though the forward simulation problem is an initial value parabolic-hyperbolic problem. We have presented special-purpose parallel multigrid algorithms that exploit the spectral structure of the inverse operator. Experiments on problems of localizing airborne contaminant release from sparse observations in a regional atmospheric transport model demonstrate that:

- 17-million-parameter inversion can be effected at a cost of just 18 forward simulations;
- wall-clock time on 1024 Alphaserver EV68 processors for the 17-million parameter inversion case is just 29 minutes;
- the multigrid preconditioner reduces the number of iterations by as much as a factor of 4.6; and
- inverse problems with 135 million initial condition parameters and 139 billion total space-time unknowns are solved in less than 5 hours on 1024 processors at 86% overall (parallel + algorithmic) efficiency.

These results suggest that ultra-high resolution inversion for linear transport problems can be carried out sufficiently rapidly to enable simulation-based “real-time” hazard assessment. The next step is to assess scalability, performance, and real-time viability using complex wind velocity fields. Our long-term goal is to incorporate more sophisticated transport models into our current framework, including, for example, deposition, regional weather models, and more accurate terrain information.

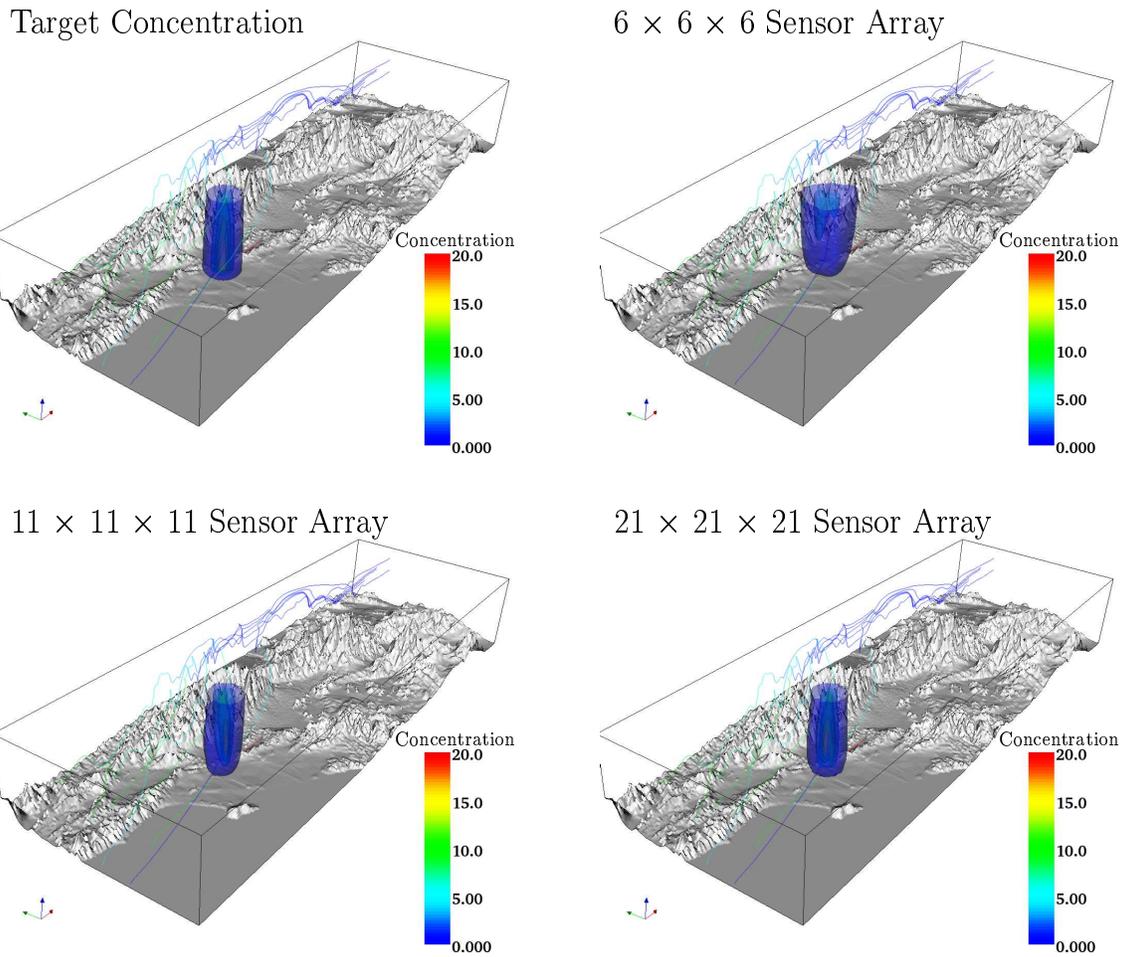


Figure 17.1. Sensitivity of the inversion result to the sensor array density. The target initial concentration is shown in the upper-left corner, and inversion results using successively-finer sensor arrays are shown in the subsequent images. As the number of sensors in each direction increases, the quality of the reconstruction of the initial concentration plume improves. $L^2(\Omega)$ norm relative errors are 0.79, 0.49, and 0.34 for the $6 \times 6 \times 6$, $11 \times 11 \times 11$, and $21 \times 21 \times 21$ sensor arrays, respectively. Inversion using the $21 \times 21 \times 21$ sensor array takes 2.5 hours on 64 processors of the Alphaserver EV68 system at the Pittsburgh Supercomputing Center. CG iterations are terminated when the norm of the residual of (17.2.8) is reduced by five orders of magnitude. Topographical elevation has been exaggerated for visualization purposes.

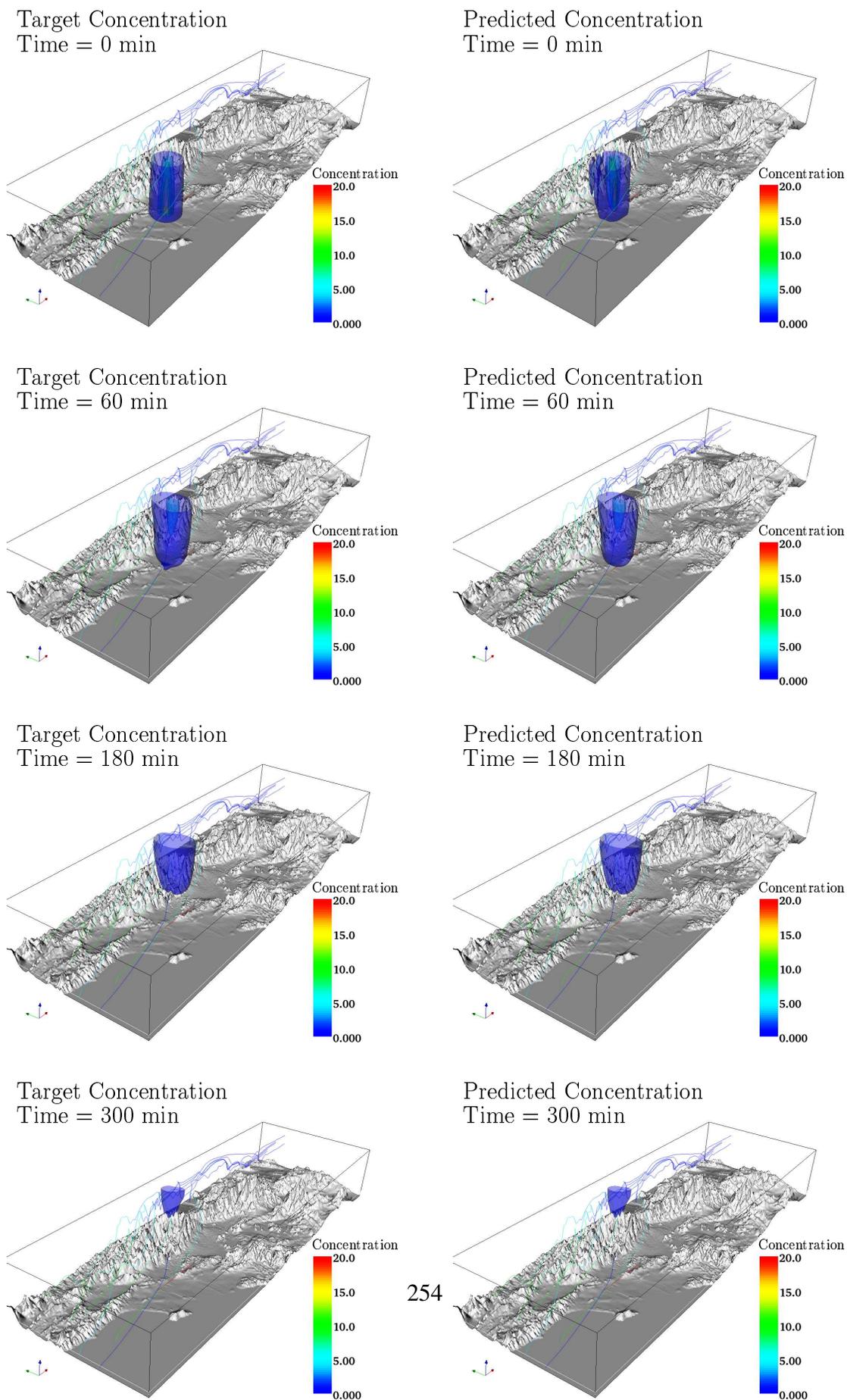


Figure 17.2. Illustration of the predictive capabilities of our inversion algorithm, using an $11 \times 11 \times 11$ sensor array. Forward transport of the central initial concentration is compared with forward

Chapter 18

An Optimization Framework for Goal-Oriented, Model-Based Reduction of Large-Scale Systems

K. Willcox (MIT), O. Ghattas (University of Texas), B. van Bloemen Waanders, B. Bader

18.1 Background

Model reduction is a powerful tool that permits the systematic generation of cost-efficient representations of large-scale systems that, for example, result from discretization of partial differential equations (PDEs). The task of determining these representations may be posed as an optimization problem: determine the reduced model that provides the optimal representation (with respect to some measure) of the large-scale system behavior. For very large systems, determination of the best reduced model via direct optimization has not been pursued, due to challenges in solving the resulting optimization problem. Instead, several reduction methods have been developed that trade off optimality for tractability, and these have been applied in many different settings with considerable success, including controls, fluid dynamics, structural dynamics, and circuit design. However, a number of open issues remain with these methods, including the reliability of reduction techniques, guarantees associated with the quality of the reduced models, and the generation of reduced models that are suitable for optimal design, optimal control and inverse problem applications.

Recent advances in scalable algorithms for large-scale optimization of systems governed by PDEs have led to solution of problems with millions of state and optimization variables [17, 16]. The problem of determining a reduced model can be cast in such a model-constrained optimization context. In particular, we consider a *goal-oriented* formulation in which the reduced model is chosen to optimally represent a particular output functional. Whereas other large-scale reduction methods, such as the proper orthogonal decomposition (POD), are purely data-driven and do not consider the underlying equations, our

model-based optimization approach enforces the reduced-order governing equations as constraints. This improves on a data-driven approach by bringing additional knowledge of the reduced-order governing equations into the construction of the basis.

Most large-scale model reduction frameworks are based on a projection approach, which can be described in general terms as follows. Consider the general linear, time-invariant (LTI) dynamical system

$$M\dot{u} + Ku = f, \quad (18.1.1)$$

$$g = Cu, \quad (18.1.2)$$

with initial condition

$$u(0) = u_0, \quad (18.1.3)$$

where $u(t) \in \mathbf{R}^N$ is the system state, $\dot{u}(t)$ is the derivative of $u(t)$ with respect to time, and the vector u_0 contains the specified initial state. In general, we are interested in systems of the form (18.1.1) that arise from spatial discretization of PDEs. In this case, the dimension of the system, N , is very large and the matrices $M \in \mathbf{R}^{N \times N}$ and $K \in \mathbf{R}^{N \times N}$ result from the spatial discretization of the underlying governing equations. The vector $f(t) \in \mathbf{R}^N$ defines the input to the system and the matrix $C \in \mathbf{R}^{Q \times N}$ defines the Q outputs of interest, which are contained in the output vector $g(t)$.

A reduced-order model of (18.1.1)–(18.1.3) can be derived by assuming that the state $u(t)$ is represented as a linear combination of m basis vectors

$$\hat{u} = \Phi\alpha \quad (18.1.4)$$

where $\hat{u}(t)$ is the reduced model approximation of the state $u(t)$ and $m \ll N$. The projection matrix $\Phi \in \mathbf{R}^{N \times m}$ contains as columns the basis vectors ϕ_i , i.e., $\Phi = [\phi_1 \ \phi_2 \ \cdots \ \phi_m]$, and the vector $\alpha(t) \in \mathbf{R}^m$ contains the corresponding modal amplitudes. This yields the reduced-order model with state $\alpha(t)$ and output $\hat{g}(t)$

$$\hat{M}\dot{\alpha} + \hat{K}\alpha = \hat{f}, \quad (18.1.5)$$

$$\hat{g} = \hat{C}\alpha, \quad (18.1.6)$$

$$\hat{M}\alpha(0) = \Phi^T M u_0, \quad (18.1.7)$$

where $\hat{M} = \Phi^T M \Phi$, $\hat{K} = \Phi^T K \Phi$, $\hat{f} = \Phi^T f$, and $\hat{C} = C \Phi$.

Projection-based model reduction techniques seek to find a basis Φ so that the reduced system (18.1.5)–(18.1.7) provides an accurate representation of the large-scale system (18.1.1)–(18.1.3) over the desired range of inputs. An optimal reduced model can be defined as one that minimizes the H-infinity norm of the difference between the reduced and original system transfer functions; however, no polynomial-time algorithm is known to achieve this goal. Algorithms such as optimal Hankel model reduction [15, 38, 132] and balanced truncation [157] have been used widely throughout the controls community to generate suboptimal reduced models with strong guarantees of quality. These algorithms can

be carried out in polynomial time; however, the computational requirements make them impractical for application to large systems such as those arising from the discretization of PDEs, for which system orders often exceed 10^4 .

Considerable effort has been applied in recent years towards development of algorithms that extend balanced truncation to large-scale LTI systems [145, 105]; however, efficient algorithms for very large systems remain a challenge. The proper orthogonal decomposition (POD) [214, 115] has emerged as a popular alternative for reduction of very large dynamical systems; however, it lacks the quality guarantees of methods such as balanced truncation.

Effective model reduction methods for optimal design, optimal control and inverse problem applications remain a challenge. Approaches developed for dynamical systems, such as POD and Krylov-based methods, have been applied in an optimization context [80, 113, 133]; however, the number of parameters in the optimization application was small. A key challenge that must be addressed in order to provide optimization-ready reduced-order models is the need for the reduced models to capture variation over a parametric input space, which, for many optimization applications, will be of high dimension. In recent work for steady-state problems, methods are presented for constructing reduced models that are of guaranteed quality over a range of inputs via the use of error estimates and adaptivity [171].

In this chapter, we formulate the problem of determining a projection basis using a goal-oriented, model-based optimization framework. The mathematical framework permits consideration of general dynamical systems with general parametric variations. The methodology is applicable to both linear and nonlinear systems and to systems with many input parameters. This chapter focuses on an initial presentation and demonstration of the methodology on a simple model problem that is linear and has a small number of inputs. We propose an efficient solution strategy that borrows concepts from the POD and employs recent methods for optimization of systems governed by PDEs to make the approach tractable for large-scale problems. The chapter is organized as follows. First, the general dynamical system framework with parametric variations is described. This is followed by a description of the goal-oriented basis optimization formulation and the proposed model reduction methodology. The approach is then demonstrated for a linear model problem that considers the unsteady two-dimensional heat equation with parametrically varying boundary control inputs. Finally, we present conclusions and directions for future research.

18.2 Dynamical System Framework

The standard LTI system framework is defined by (18.1.1)–(18.1.3). In this section, we present the more general case that includes parametric variation in the system.

18.2.1 Parametric input variations

We consider a finite set of instantiations of the governing equations (18.1.1)–(18.1.3) that could arise from variations in the coefficient matrices M and K , the input f , or the initial state u_0 . For example, where

(18.1.1)–(18.1.3) represent a spatially discretized PDE, these variations stem from changes in the domain shape, boundary conditions, coefficients, initial conditions, or sources of the underlying PDEs. The general dynamical system for S different instances is thus written

$$M^k \dot{u}^k + K^k u^k = f^k, \quad k = 1, \dots, S \quad (18.2.8)$$

$$u^k(0) = u_0^k, \quad k = 1, \dots, S \quad (18.2.9)$$

$$g^k = C^k u^k, \quad k = 1, \dots, S \quad (18.2.10)$$

where the superscript k denotes the k th instance of the system, which has corresponding state $u^k(t)$ and output $g^k(t)$.

Using the projection framework described in the previous section, a reduced-order model of (18.2.8)–(18.2.10) is obtained as

$$\hat{M}^k \dot{\alpha}^k + \hat{K}^k \alpha^k = \hat{f}^k, \quad k = 1, \dots, S \quad (18.2.11)$$

$$\hat{g}^k = \hat{C}^k \alpha^k, \quad k = 1, \dots, S \quad (18.2.12)$$

$$\hat{M}^k \alpha(0)^k = \Phi^T M^k u_0^k, \quad k = 1, \dots, S \quad (18.2.13)$$

where $\hat{M}^k = \Phi^T M^k \Phi$, $\hat{K}^k = \Phi^T K^k \Phi$, $\hat{f}^k = \Phi^T f^k$, and $\hat{C}^k = C^k \Phi$.

18.2.2 Proper orthogonal decomposition

POD is a widely used approach to determine the reduced basis Φ . POD can be applied efficiently to large systems using the method of snapshots [214] as follows. Consider the collection of “snapshots”, $u^k(t_j)$, $j = 1, \dots, T$, $k = 1, \dots, S$, where $u^k(t_j) \in \mathbf{R}^N$ is the solution of the governing equations (18.2.8) at time t_j for parameter instance k . T time instants are considered for each parameter instance, yielding a total of ST snapshots. We define the snapshot matrix $U \in \mathbf{R}^{N \times ST}$ as

$$U = [u^1(t_1) \ u^1(t_2) \ \dots \ u^1(t_T) \ u^2(t_1) \ \dots \ \dots \ u^S(t_T)] \quad (18.2.14)$$

and we will refer to the i th column of U as the i th snapshot, denoted by U_i .

The POD basis vectors are chosen to be the orthonormal set that maximizes the following cost [30]:

$$\phi = \arg \max_{\phi} \frac{\langle |u, \phi|^2 \rangle}{(\phi, \phi)}, \quad (18.2.15)$$

where (u, ϕ) denotes the scalar product of the basis vector with the field $u(t)$ evaluated over the domain, and $\langle \cdot \rangle$ represents a time-averaging operation. In the case of the discrete snapshots contained in U , (18.2.15) is maximized when the m basis vectors are chosen to be the first m left singular vectors of U . For a fixed basis size, the POD basis therefore minimizes the error between the original snapshots and their representation in the reduced space defined by

$$E = \sum_{k=1}^S \sum_{j=1}^T [u^k(t_j) - \tilde{u}^k(t_j)]^T [u^k(t_j) - \tilde{u}^k(t_j)], \quad (18.2.16)$$

where $\tilde{u}^k(t_j) = \Phi\Phi^T u^k(t_j)$. This error is equal to the sum of the singular values corresponding to those singular vectors not included in the basis

$$E = \sum_{i=m+1}^{ST} \sigma_i, \quad (18.2.17)$$

where σ_i is the i^{th} singular value of U .

The POD is an optimal basis in the sense that it minimizes the data reconstruction error given by (18.2.16); however, it is important to note that this optimality applies only to the representation of a known state solution $u^k(t_j)$ in the reduced space, i.e. \tilde{u} is computed as $\tilde{u}^k(t_j) = \Phi\Phi^T u^k(t_j)$, not by solution of the reduced model ($\tilde{u} \neq \hat{u}$). Therefore, the error expression does not apply to the resulting POD reduced-order model (18.1.5). In particular, the error expression yields no rigorous information regarding the accuracy of the solution of the reduced model and thus whether \hat{u} is a good approximation of u . Moreover, the POD basis does not account for the system outputs, although methods to augment the standard approach have been proposed that use adjoint information [138, 242]. In addition, because no information regarding the governing equations is included in the POD process, the POD basis does not properly reflect the fact that the snapshots $u^k(t_j)$ are associated with different parametric instances of the system.

In the following section we present an alternative method to determine the reduced-space basis. This method seeks to minimize an error similar in form to (18.2.16); however, we will improve upon the POD, first, by minimizing the error in the outputs (as opposed to states) and, second, by imposing additional constraints that $\hat{u}^k(t)$ should result from satisfying the reduced-order governing equations for each parameter instance k .

18.3 Optimized Reduced-Order Basis

18.3.1 Constrained optimization formulation for projection basis

We pose the problem of selecting the basis Φ as a goal-oriented optimization problem that seeks to minimize the difference between the full-space and reduced-order output solution over a selected set of inputs and the interval $[0, t_f)$, subject to satisfying the underlying governing equations. The problem of determining the optimal basis, $\Phi \in \mathbf{R}^{N \times m}$, can be written as

$$\begin{aligned} \min_{\Phi, \alpha} \mathcal{G} &= \frac{1}{2} \sum_{k=1}^S \int_0^{t_f} (g^k - \hat{g}^k)^T (g^k - \hat{g}^k) dt \\ &+ \frac{\beta}{2} \sum_{j=1}^m (1 - \phi_j^T \phi_j)^2 \end{aligned} \quad (18.3.18)$$

subject to

$$\Phi^T M^k \Phi \dot{\alpha}^k + \Phi^T K^k \Phi \alpha^k = \Phi^T f^k, \quad k = 1, \dots, S \quad (18.3.19)$$

$$\Phi^T M^k \Phi \alpha^k(0) = \Phi^T M^k u_0^k, \quad k = 1, \dots, S \quad (18.3.20)$$

$$\hat{g}^k = C^k \Phi \alpha^k, \quad k = 1, \dots, S. \quad (18.3.21)$$

In the case of a linear relationship between outputs and state as in (18.2.10), the objective function can be written

$$\begin{aligned} \mathcal{G} &= \frac{1}{2} \sum_{k=1}^S \int_0^{t_f} (u^k - \hat{u}^k)^T H^k (u^k - \hat{u}^k) dt \\ &+ \frac{\beta}{2} \sum_{j=1}^m (1 - \phi_j^T \phi_j)^2, \end{aligned} \quad (18.3.22)$$

where $H^k = C^{kT} C^k$ can be interpreted as a weighting matrix that defines the states relevant to the specified output. While the first term in the objective function (18.3.22) has similarities with that minimized by the POD, given by (18.2.16), there are two important distinctions to note. First, the goal-oriented nature of the formulation (18.3.22) focuses on reduction of the error for a particular output functional rather than for the general state vector. Second, through the optimization constraints (18.3.19)–(18.3.21), the general optimization approach requires satisfaction of the reduced-order governing equations to compute \hat{u} . The error minimized by the optimization approach is thus tied rigorously to the reduced-order model, whereas the POD is based purely on snapshot data. In both cases, however, the definition of the error is limited to a discrete set of observations.

The second term in (18.3.22) is a regularization term that penalizes the deviation of the length of the basis vectors from unity and β is a constant weighting. This regularization acts only in the null space of the projected Hessian matrix of the first term of (18.3.22). Therefore, the reduced output approximation, \hat{g} , is unaffected by the regularization term, yet the conditioning of the optimization problem is improved. Note, however, that there remains a null space of the projected Hessian matrix that admits arbitrary rotations of the basis vectors; the optimization method chosen to solve (18.3.18)–(18.3.21) should therefore be tolerant of singular projected Hessian matrices. It is also important to note that the optimization problem (18.3.18)–(18.3.21) is nonlinear and nonconvex; therefore, there is no guarantee that a purely local optimization method will converge to the global optimum. Therefore, generating the initial guess is very important; strategies to address this issue will be discussed in the next section.

18.3.2 Optimality conditions and the reduced gradient

The optimality conditions for the system (18.3.18)–(18.3.21) can be derived by defining the Lagrangian functional

$$\mathcal{L}(\Phi, \alpha^k, \lambda^k, \mu^k) = \frac{1}{2} \sum_{k=1}^S \int_0^{t_f} (u^k - \Phi \alpha^k)^T H^k (u^k - \Phi \alpha^k) dt$$

$$\begin{aligned}
& + \frac{\beta}{2} \sum_{j=1}^m (1 - \phi_j^T \phi_j)^2 \\
& + \sum_{k=1}^S \int_0^{t_f} \lambda^{kT} \left(\Phi^T M^k \Phi \dot{\alpha}^k + \Phi^T K^k \Phi \alpha^k - \Phi^T f^k \right) dt \\
& + \sum_{k=1}^S \mu^{kT} \left(\Phi^T M^k \Phi \alpha(0) - \Phi^T M^k u_0^k \right), \tag{18.3.23}
\end{aligned}$$

where $\lambda^k = \lambda^k(t) \in \mathbf{R}^m$ and $\mu^k \in \mathbf{R}^m$ are Lagrange multipliers (also known as adjoint state variables) that respectively enforce the state ODE system and initial conditions for the k th sample. The optimality system can be derived by taking variations of the Lagrangian with respect to the adjoint, state, and basis vector variables.

Setting the first variation of the Lagrangian with respect to λ^k to zero and arguing that the variation of λ^k is arbitrary in $[0, t_f)$, and setting the derivative of the Lagrangian with respect to μ^k to zero, simply recovers the state equation and initial conditions (18.3.19)–(18.3.20).

Setting the first variation of the Lagrangian with respect to the α^k to zero, and arguing that the variation of α^k is arbitrary in $[0, t_f)$, at $t = 0$, and at $t = t_f$, yields the adjoint equation, final condition and definition of μ

$$-\Phi^T M^k \Phi \dot{\lambda}^k + \Phi^T K^{kT} \Phi \lambda^k = \Phi^T H^k (u^k - \Phi \alpha^k), \tag{18.3.24}$$

$$\lambda^k(t_f) = 0, \quad k = 1, \dots, S \tag{18.3.25}$$

$$\mu^k = \lambda^k(0), \quad k = 1, \dots, S. \tag{18.3.26}$$

Note that, without loss of generality, M is assumed to be a symmetric matrix.

Taking the derivative of the Lagrangian with respect to the basis vector variables Φ yields the following matrix equation

$$\begin{aligned}
\delta\mathcal{L}_\Phi &= \sum_{k=1}^S \int_0^{t_f} H^k (\Phi\alpha^k - u^k) \alpha^{kT} dt \\
&\quad - 2\beta\Phi \text{diag}(1 - \phi_i^T \phi_i) \\
&\quad + \sum_{k=1}^S \int_0^{t_f} [M^k \Phi (\lambda^k \dot{\alpha}^{kT} + \dot{\alpha}^k \lambda^{kT}) \\
&\quad + K^{kT} \Phi \lambda^k \alpha^{kT} + K^k \Phi \alpha^k \lambda^{kT} - f^k \lambda^{kT}] dt \\
&\quad + \sum_{k=1}^S M^k \Phi \mu^k \alpha_0^T \\
&\quad + \sum_{k=1}^S M^k (\Phi \alpha_0^k - u_0^k) \mu^{kT} = 0.
\end{aligned} \tag{18.3.27}$$

The combined system (18.3.19)–(18.3.20), (18.3.24)–(18.3.26), and (18.3.27) represents the first-order Karush-Kuhn-Tucker optimality conditions for the optimization problem (18.3.18)–(18.3.21).

To solve the constrained optimization problem (18.3.18)–(18.3.21), we choose to eliminate the state variables α^k and state equations (18.3.19) and solve an equivalent unconstrained optimization problem in the Φ variables. Efficient solution of this unconstrained optimization problem requires a gradient-based method, which requires function and gradient evaluations. In this case, the gradient of the unconstrained objective with respect to Φ is given by $\delta\mathcal{L}_\Phi$ when the α^k satisfy the state equations and (λ^k, μ^k) satisfy the adjoint equations. The procedure to compute the gradient of \mathcal{G} for any value of Φ can therefore be summarized as follows. First, solve the state equations (18.3.19)–(18.3.20) to determine $\alpha^k(t)$. Second, solve the adjoint equations (18.3.24)–(18.3.26) to determine $\lambda^k(t)$ and μ^k . Finally, use the computed α^k , λ^k , and μ^k in (18.3.27) to determine the gradient.

18.3.3 Basis computation

The formulation defined by equations (18.3.18)–(18.3.21) provides a mathematical definition of the desired optimal basis; however, in practice this optimization problem may not be tractable for large-scale problems. First, we may not be able to afford storage of the entire time history for the full model, which leads us to adopt a snapshot-based approach. As in the POD, the time integrals in (18.3.18) are replaced by a summation over a finite number of discrete time instants. Our method therefore requires a priori computation of a set of high-fidelity solutions over a pre-determined set of time instants and input parameter values.

Second, even with this simplification, the number of optimization variables is equal to mN — the desired number of basis functions multiplied by the length of each basis vector — where for many applications $N \geq O(10^6)$. Therefore, it will be assumed that each basis vector can be represented as a linear

combination of snapshots:

$$\phi_j = \sum_{i=1}^{ST} \gamma_i^j U_i \quad j = 1, \dots, m \quad (18.3.28)$$

where the coefficients γ_i^j are the variables in the modified optimization problem. This assumption reduces the number of optimization variables from mN to mST , where, for large-scale applications, typically $ST \ll N$. As a consequence, neither the gradient computation nor the optimization step computation (which dominate the cost of an optimization iteration) scale with the full system size N . Moreover, using adjoints ensures that the gradient computation requires just $2S$ reduced model solutions per optimization iteration, as opposed to mST as with direct sensitivities. The assumption that the basis vectors can be represented as a linear combination of snapshots is motivated by the singular value decomposition (SVD) theory underlying the POD, for which the relation (18.3.28) is exact (this is equivalent to solving the inner versus the outer SVD problem). As will be demonstrated in the following results section, in the case of the optimal basis formulation, numerical experiments suggest that this formulation is still capable of yielding very good results.

Equation (18.3.28) can be written in matrix form as

$$\Phi = U\Gamma, \quad (18.3.29)$$

where γ_i^j is the ij th element of $\Gamma \in \mathbf{R}^{ST \times m}$. Gradients of the objective function with respect to Γ are related simply to gradients with respect to Φ by

$$\frac{\partial \mathcal{L}}{\partial \Gamma} = U^T \frac{\partial \mathcal{L}}{\partial \Phi}. \quad (18.3.30)$$

The modified optimization formulation offers no guarantees of convexity and the choice of initial guess for the basis is thus very important. In this chapter, we present two possible strategies. The first is to use the POD basis as an initial starting point. Since a snapshot set is required anyway, the additional cost of computing the POD basis is small. A second strategy is to employ continuation on the basis dimension. In this approach, the initial guess for the case of m basis vectors is chosen to be the solution of the optimization problem for $m - 1$ basis vectors plus an arbitrary m th vector. This iterative procedure can be initialized at any value $m \geq 1$ with the POD basis vectors as an initial guess on the first iteration.

18.4 Model Problem and Results

18.4.1 Model problem description

Results are presented for a model problem that considers the two-dimensional time-dependent heat equation. In this case, the PDE is given by

$$\frac{\partial \bar{u}}{\partial t} - k\nabla^2 \bar{u} = 0 \text{ in } \Omega \quad (18.4.31)$$

$$\bar{u} = \bar{u}_c \text{ on } \Gamma \quad (18.4.32)$$

$$\bar{u} = \bar{u}_0 \text{ in } \Omega \text{ for } t = 0 \quad (18.4.33)$$

where $\bar{u}(x, y, t)$ is the temperature field defined on the domain Ω , $\bar{u}_c(x, y)$ is the boundary control function (which is assumed to be constant in time) applied on the boundary Γ , and $\bar{u}_0(x, y)$ is the given initial temperature field. The output of interest is the temperature over a specified sub-region of the domain.

A modification of the finite element formulation from [19] is used to discretize the problem in space, yielding a dynamical system of the form (18.2.8)–(18.2.10), where $u^k(t)$ represents the spatially discretized temperature field corresponding to forcing input f^k , and $g^k(t)$ contains those elements of u^k that lie within the specified region of interest. Figure 18.1 shows the specific domain Ω that was used, which is discretized with triangular elements. Results will be shown for a discretization containing a total of $N = 480$ temperature unknowns. The specified initial condition is $u = 0$ at $t = 0$, and the selected time integration scheme is an implicit Euler method with a constant time step over the time interval $[0, T]$. The boundary control is applied on $\Gamma_c = \{(0, y) : 0 \leq y \leq 3\}$, i.e., Dirichlet control on the left boundary of the domain. Neumann boundary conditions are specified on $\Gamma_N = \{(3, y) : 1.5 \leq y \leq 3\}$ and remaining part of the boundary, Γ_D , is fixed with zero Dirichlet conditions.

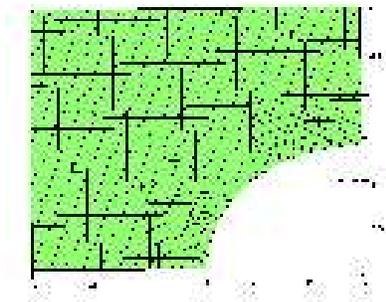


Figure 18.1. Problem domain and boundary conditions: Neumann on right side, Dirichlet on all other boundaries.

Snapshots were generated by performing a time simulation of the system under different forcing conditions. The forcing was generated by applying a temperature distribution along the boundary Γ_c in Figure 18.1. For the results presented here, the forcing functions considered were parameterized using sinusoidal distributions with successively higher spatial frequency. Snapshots were generated over $S = 5$ instances of the control parameter forcing with $T = 20$ time instants for each parameter instance. Using the optimization formulation (18.3.18)–(18.3.21), we seek to select the m basis functions that minimize the error defined by (18.3.18) while satisfying the reduced-order state equations for each control instance. The basis functions are assumed to be a linear combination of available snapshots, hence there are $mST = 100m$ design variables in the optimization problem. The state and adjoint equations, each consist of m uncoupled ODE systems of dimension $ST = 100$. An implicit backward Euler scheme is used to discretize the ODEs (note that the adjoint equations are marched backward in time). The resulting fully discrete systems are lower and upper tridiagonal for the state and adjoint, respectively, and thus can be solved very efficiently.

18.4.2 Optimized basis performance

For the first set of results, the output is defined to be the temperature over a strip of the domain in the region $0.5 < x < 1.0$, $0.5 < y < 2.5$, yielding an output vector of size $Q = 47$. To determine the goal-oriented basis, (18.3.18)–(18.3.21) were solved by using (18.3.27) to compute analytical gradients and employing an unconstrained optimization algorithm that uses a trust-region-based Newton method [70]. Figure 18.2 shows the resulting objective function values for bases ranging in size from $m = 1$ to $m = 10$. Figure 18.2 also shows the evaluation of (18.3.22) for the POD bases over this range of m . It can be seen clearly that the optimized basis outperforms the POD in all cases, particularly when m is small.

In order to provide a quantitative metric by which to judge the performance of the optimized basis, balanced truncation was applied to this problem. The problem was converted to standard LTI form by considering each parametric forcing function as an independent input. Figure 18.2 shows the evaluation of (18.3.22) for truncated balanced models of size $m = 1$ through $m = 10$. It can be seen that the optimized basis provides a substantial improvement over POD when both are compared to the results of balanced truncation. It is also important to note that balanced truncation uses both a left and a right projection basis, and thus has twice as many degrees of freedom as the goal-oriented optimized basis.

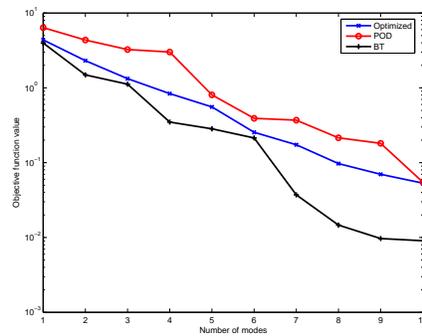


Figure 18.2. The error (18.3.22) versus number of modes for the goal-oriented optimized basis, the POD basis, and balanced truncation.

18.4.3 Comparison with POD

A significant advantage of the goal-oriented approach is that the basis can be optimized with respect to a particular output functional, whereas the POD seeks to minimize the reconstruction error over all states. Several different output definitions were considered in order to gain insight to the optimized basis results.

If the output considered is to minimize the error of state prediction over the entire domain, that is, $H^k = 1$ in (18.3.22), then the goal-oriented approach seeks to minimize the same error as the POD. However, it is important to note again the difference in the representation of the term \tilde{u}^j , which for POD is computed directly from the known solution u^j , i.e. $\tilde{u}^j = \Phi\Phi^T u^j$. In this sense the POD is a purely data-based method

Table 18.1. Comparison of optimization results. The objective function given by (18.3.22) is evaluated for the optimized basis (\mathcal{G}_{opt}) and the POD basis (\mathcal{G}_{pod}).

Min. error over	S	T	m	\mathcal{G}_{opt}	\mathcal{G}_{pod}
All states	5	20	5	28.9829	29.2762
$x = 0.625, y = 0.625$	3	20	5	2.9038e-3	0.01066
$0.5 < x, y < 1$	5	20	5	0.01282	0.1932
$0.5 < x, y < 1$	5	20	5	0.5555	0.8062

that does not account in any way for the underlying governing equations. In contrast, our method computes \hat{u}^j in (18.3.22) by requiring the solution to satisfy the governing equations in the reduced-order space.

Results for this case are shown in the first row of Table 18.1. Using the POD basis as an initial guess, the optimizer is able to make almost no improvement in the objective function. As shown in Table 18.1, the reduction in the error is just 1%. For different values of S , T and m , the POD basis is found to be almost optimal with respect to state reconstruction error for this example. Due to the symmetry properties of the system (M and K are symmetric matrices), any congruent basis transformation, such as the POD, is guaranteed to preserve the stability of the system. Thus it is to be expected that the POD should perform well on this heat conduction example. As the results show, the additional error from solution of the governing equations in the reduced space is not significant in this case. In more complicated examples where this error becomes significant, the optimized basis might be expected to provide an advantage over the POD even in terms of full state reconstruction. This is particularly true for non-symmetric systems, such as those representing the Euler equations, for which the POD basis can routinely produce unstable reduced-order models.

Table 18.1 shows the results for other outputs corresponding to various specified $x - y$ regions (and thus different weightings H in the objective function). Note that the POD basis is computed in the standard way and thus is insensitive to the choice of output functional. The values in the column \mathcal{G}_{pod} represent the standard POD basis evaluated using the criterion defined by (18.3.22) for each different instance of H (i.e. the metric \mathcal{G}_{pod} is case-dependent). It can be seen that by defining an output functional, the goal-oriented basis can yield substantial improvements in errors over the POD basis. It should be emphasized that our method does not simply “ignore” states that lie outside of the region of interest, since \hat{u}^j is computed by solving the reduced-order equations over the entire domain. Therefore the basis must represent *all* states – but the optimization formulation allows the basis energy to be focused appropriately to achieve the desired objective. One might draw conceptual parallels between this approach and a posteriori error estimates to manage grid adaptivity.

Figure 18.3 shows the output errors in the case of an output functional defined over the region $0.5 < x < 1, 0.5 < y < 1$. Each plot in the figure corresponds to one of the nine grid points that lie within the region of interest (for clarity, just four of the points are shown). The first $T = 20$ snapshots correspond to the first instance of control forcing, the second $T = 20$ correspond to the second instance, and so on. The figure shows that for almost every snapshot in the ensemble, the optimized basis results in a more accurate prediction of the temperature at the point of interest. In many cases, the error is reduced by almost an order of magnitude.

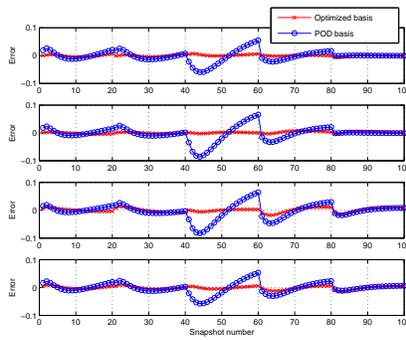


Figure 18.3. Error in temperature prediction for each snapshot using POD and optimized basis. The optimized basis was selected so as to minimize the error over the region $0.5 < x < 1, 0.5 < y < 1$. Errors are shown for four of the nine points contained within this region.

The reduced output errors shown in Figure 18.3 come at a cost. Figure 18.4 shows the norm of the errors computed over the entire domain for each snapshot. In order to reduce the errors at the specified points, the optimized basis yields less accurate predictions for other states. However, it is again important to note that this trade-off in accuracy is done in a systematic way using both the governing equations and the defined output functional. According to the optimization result, the larger errors observed in other areas of the domain are compatible with the task of reducing the error in the region of interest.

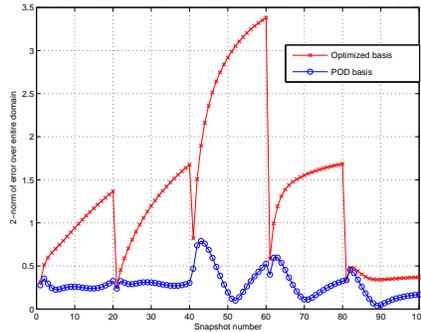


Figure 18.4. Norm of the error in temperature prediction over the entire domain for each snapshot using POD and optimized basis. The optimized basis was selected so as to minimize the error over the region $0.5 < x < 1, 0.5 < y < 1$.

Figure 18.5 shows an example solution for one particular snapshot. The snapshot chosen corresponds to the first time instant in the third control function (snapshot number 41 in Figures 18.3 and 18.4). The output of interest is the temperature over the region $0.5 < x < 1, 0.5 < y < 1$, indicated by the bold square in the

figure. As can be seen from Figures 18.3 and 18.4, this snapshot corresponds to a case where the output error is substantially reduced by the optimized basis. The reductions in output error are approximately an order of magnitude – for example, in the bottom plot in Figure 18.3 the error in the output for this snapshot is reduced in magnitude from -0.0209 to 0.0036. In contrast, the norm of the error over the domain is substantially increased from 0.4675 to 0.8201. By comparing the plots in Figure 18.5, this effect can be clearly seen. The optimized basis result indicates that the large errors near the control boundary are acceptable (in fact optimal) if one is concerned only with predicting the temperature within the indicated region.

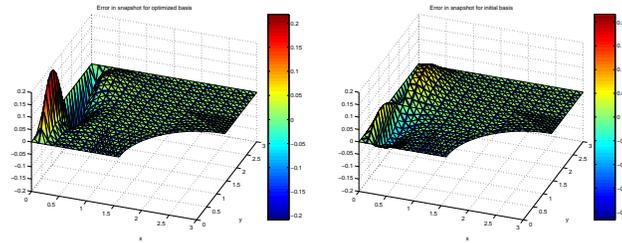


Figure 18.5. Reduced-order model temperature prediction for snapshot number 41 using optimized (left) and POD (right) basis. The optimized basis was selected so as to minimize the error over the region $0.5 < x < 1, 0.5 < y < 1$.

18.5 Conclusions

The goal-oriented, model-based optimization approach presented here provides a general framework for construction of reduced models, and is particularly applicable for optimal design, optimal control and inverse problems. The optimization approach provides significant advantages over the POD by allowing the projection basis to be targeted to output functionals, by providing a framework in which to consider multiple parameter instances, and by incorporating the reduced-order governing equations as constraints in the basis derivation.

APPENDIX I- The Regularized Eikonal Equation

We motivate and derive the weak form of the regularized eikonal equation used in the main body of Chapter 15 as follows. First, we illustrate ideas by considering a couple of (essentially) one-dimensional analytical examples; then, a weak form of the fully multidimensional problem is derived; finally, an iterative procedure is specified and illustrated using the flow geometry prescribed in Section 15.5.

We begin by considering the one-dimensional eikonal equation

$$\left(\frac{d\ell^*}{dx}\right)^2 = 1, \quad (.0.34)$$

subject to $\ell^* = 0$ at the boundaries $x = 0$ and $x = 1$. Solutions to (.0.34) are given by $\ell^* = \pm x$, which cannot solve the boundary conditions without introducing discontinuities in either ℓ^* or its derivative at $x = 1/2$.

For stable solution by finite elements in an enclosed region, the eikonal equation must be regularized. If we add a regularization term $-\epsilon d^2\ell^*/dx^2$, where ϵ is a small positive quantity, to the left hand side of Eq. (.0.34), the problem becomes

$$\left(\frac{d\ell^*}{dx}\right)^2 - \epsilon \frac{d^2\ell^*}{dx^2} = 1, \quad \ell^*(0) = \ell^*(1) = 0. \quad (.0.35)$$

The general solution of the differential equation for ℓ^* is obtained by first solving the first-order equation for $d\ell^*/dx$ and then performing a second integration to give $\ell^* = d - \epsilon \ln \cosh[(x - c)/\epsilon]$, where c and d are constants of integration. Applying the boundary conditions then determines ℓ^* uniquely as

$$\ell^*(x) = \epsilon \ln \left\{ \frac{\cosh[-1/(2\epsilon)]}{\cosh[(2x-1)/(2\epsilon)]} \right\} = -\epsilon \ln \left[\frac{e^{(2x-1)/(2\epsilon)} + e^{-(2x-1)/(2\epsilon)}}{e^{-1/(2\epsilon)} + e^{1/(2\epsilon)}} \right]. \quad (.0.36)$$

It is readily deduced that for $0 < \epsilon \ll 1$, ℓ^* has the asymptotic behavior $\ell^* \sim x$ for $0 \leq x < 1/2$ and $\ell^* \sim 1 - x$ for $1/2 < x \leq 1$. In the neighborhood of $x = 1/2$, we define the stretched coordinate $\eta = \epsilon^{-1}(x - 1/2)$, in terms of which $\ell^* \sim 1/2 - \epsilon \ln[2 \cosh \eta]$. In the asymptotic context, a composite approximation spanning both the outer regions ($0 \leq x < 1/2$ and $1/2 < x \leq 1$) and the inner zone $|x - 1/2| \sim O(\epsilon)$ is in fact given by the inner approximation written in terms of x . Hence, an asymptotic approximation for (.0.36), valid in the limit of small ϵ , is given by

$$\ell^*(x) \sim \frac{1}{2} - \epsilon \ln \left[2 \cosh \left(\frac{2x-1}{2\epsilon} \right) \right] = \frac{1}{2} - \epsilon \ln \left[e^{(2x-1)/(2\epsilon)} + e^{-(2x-1)/(2\epsilon)} \right]. \quad (.0.37)$$

We note that the maximum deviation in ℓ^* from $\sup\{x, 1-x\}$ occurs at $x = 1/2$, where $\ell^*(1/2) \sim 1/2 - \epsilon \ln 2$. The exact solution (.0.36) and its approximation (.0.37) are illustrated in Figure 1.

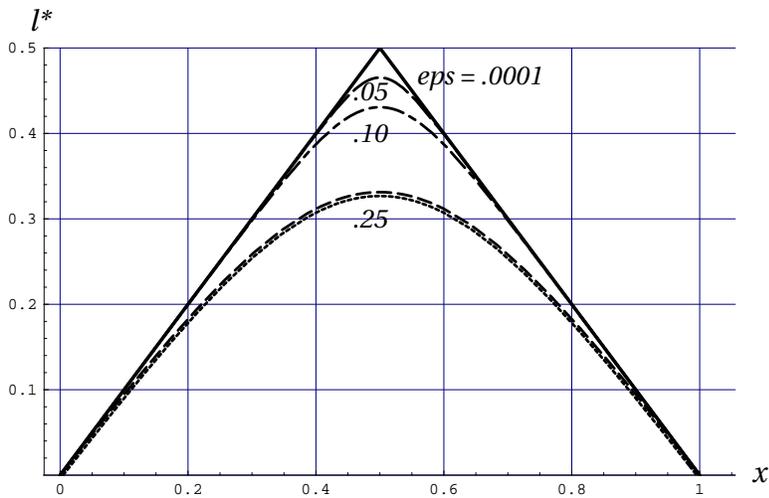


Figure 1. Solution of (.0.35) for several values of ϵ (“eps”). The two curves (dashed and dotted) for $\epsilon = .25$ correspond to the exact solution (.0.36) and its asymptotic representation (.0.37), respectively; the two expressions are virtually indistinguishable for smaller values of ϵ .

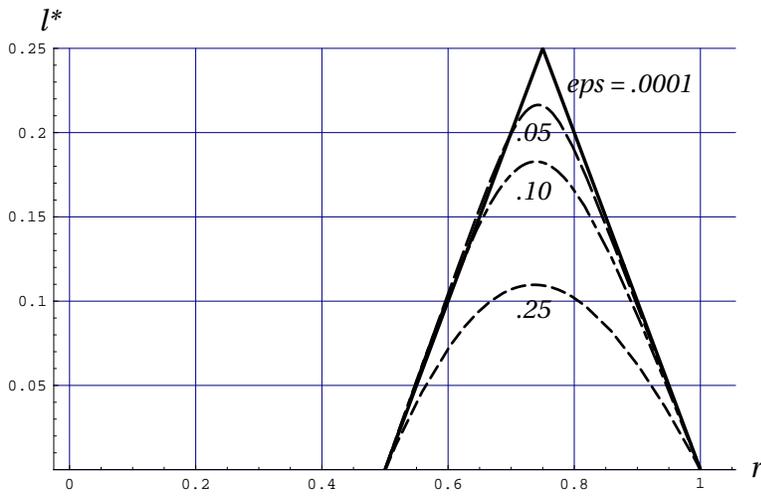


Figure 2. Solution of (.0.39) for $r_i = 1/2$, $r_0 = 1$, and several values of ϵ (“eps”).

The straightforward multidimensional generalization of (.0.35) used in the present work is given, on a domain Ω , by

$$(\vec{\nabla} \ell^*) \cdot (\vec{\nabla} \ell^*) - \epsilon \nabla^2 \ell^* = 1, \quad \ell^* = 0 \text{ on } \partial\Omega_w, \quad \vec{\nabla} \ell^* = 0 \text{ on } \partial\Omega_{L,o}. \quad (.0.38)$$

The solution of this problem yields a tent-like structure, with the aforementioned distance-representation property for ℓ^* , over the domain Ω . For example, if the domain is the rectangle $0 \leq x \leq 1$, $0 \leq y \leq L$, with $\ell^* = 0$ on $x = 0$ and $x = 1$, and $\partial \ell^* / \partial y = 0$ on $y = 0$ and $y = L$, the solution is simply (.0.36), independent of y . A more nontrivial example is the solution of (.0.38) in polar coordinates (r, ϑ) when $\partial\Omega_w$ is the ring $0 < r_i \leq r \leq r_0$ and the boundary conditions are $\ell^*(r_i) = \ell^*(r_0) = 0$. Due to angular symmetry, (.0.38) reduces to solving $(d\ell^*/dr)^2 - \epsilon r^{-1} (d/dr)(r d\ell^*/dr) = 1$, ultimately giving the final result

$$\ell^* = \epsilon \ln \left\{ \frac{I_0(r_i/\epsilon) K_0(r_0/\epsilon) - I_0(r_0/\epsilon) K_0(r_i/\epsilon)}{I_0(r/\epsilon) [K_0(r_0/\epsilon) - K_0(r_i/\epsilon)] - K_0(r/\epsilon) [I_0(r_0/\epsilon) - I_0(r_i/\epsilon)]} \right\}, \quad (.0.39)$$

where I_0 and K_0 are the zero-order modified Bessel functions. This solution is illustrated in Figure 2; its physical appearance in three dimensions is obtained by rotating the profiles about the ℓ^* -axis to yield the upper part of a torus-like structure.

As with the Navier-Stokes equations, the use of Sundance requires a weak form of the eikonal problem

(.0.38). We begin by writing

$$\int_{\Omega} r \left[(\vec{\nabla} \ell^*) \cdot (\vec{\nabla} \ell^*) - \varepsilon \nabla^2 \ell^* - 1 \right] d\Omega = 0, \quad (.0.40)$$

where r is the test function. Use of the identity $r \nabla^2 \ell^* = \vec{\nabla} \cdot (r \vec{\nabla} \ell^*) - (\vec{\nabla} r) \cdot (\vec{\nabla} \ell^*)$ and application of the divergence theorem allows (.0.40) to be expressed in terms of first derivatives as

$$\int_{\Omega} \left\{ r \left[(\vec{\nabla} \ell^*) \cdot (\vec{\nabla} \ell^*) - 1 \right] + \varepsilon (\vec{\nabla} r) \cdot (\vec{\nabla} \ell^*) \right\} d\Omega - \varepsilon \int_{\partial\Omega} r (\vec{\nabla} \ell^*) \cdot \vec{n} d(\partial\Omega) = 0, \quad (.0.41)$$

where \vec{n} is the unit normal to $\partial\Omega$.

Based on the boundary conditions expressed in (.0.38), the surface integral vanishes on the inlet and outlet portions of $\partial\Omega$. Approximating the wall boundary condition by

$$-(\vec{\nabla} \ell^*) \cdot \vec{n} \Big|_{\partial\Omega_w} = \hat{\varepsilon}^{-1} \ell^*, \quad \hat{\varepsilon} \ll 1, \quad (.0.42)$$

and taking the limit $\hat{\varepsilon} \rightarrow 0$ [independent of the ε in (.0.41)], we obtain, by the same argument advanced in Section 3.2, a replacement for the surface term according to

$$\int_{\Omega} \left\{ r \left[(\vec{\nabla} \ell^*) \cdot (\vec{\nabla} \ell^*) - 1 \right] + \varepsilon (\vec{\nabla} r) \cdot (\vec{\nabla} \ell^*) \right\} d\Omega + \varepsilon \int_{\partial\Omega_w} r \ell^* d(\partial\Omega) = 0. \quad (.0.43)$$

Equation (.0.43) is thus the final weak form of the problem (.0.38). We note that (.0.43) is decoupled from the larger problem, although the reverse is obviously not true since ℓ^* enters into (15.3.22) through the mixing length ℓ_j that appears in the expression for the turbulent stress tensor $\underline{\tau}$.

In order to actually solve (.0.43), which is nonlinear in the unknown function ℓ^* , it is necessary to adopt an iterative approach as discussed in Section 3.2. In particular, since we ultimately require a weak form that is linear with respect to the unknown function ℓ^* (as well as the test function r), we formally linearize the nonlinear term $(\vec{\nabla} \ell^*) \cdot (\vec{\nabla} \ell^*)$ about an initial guess ℓ_0^* by writing $\ell^* = \ell_0^* + \ell_1^*$, where the correction ℓ_1^* is presumed to be small. Consequently,

$$\begin{aligned} (\vec{\nabla} \ell^*) \cdot (\vec{\nabla} \ell^*) &= (\vec{\nabla} \ell_0^*) \cdot (\vec{\nabla} \ell_0^*) + 2(\vec{\nabla} \ell_1^*) \cdot (\vec{\nabla} \ell_0^*) + (\vec{\nabla} \ell_1^*) \cdot (\vec{\nabla} \ell_1^*) \\ &\approx (\vec{\nabla} \ell_0^*) \cdot (\vec{\nabla} \ell_0^*) + 2(\vec{\nabla} \ell_1^*) \cdot (\vec{\nabla} \ell_0^*) \\ &= (\vec{\nabla} \ell_0^*) \cdot (\vec{\nabla} \ell_0^*) + 2[\vec{\nabla}(\ell^* - \ell_0^*)] \cdot (\vec{\nabla} \ell_0^*) \\ &= (\vec{\nabla} \ell_0^*) \cdot (2\vec{\nabla} \ell^* - \vec{\nabla} \ell_0^*). \end{aligned} \quad (.0.44)$$

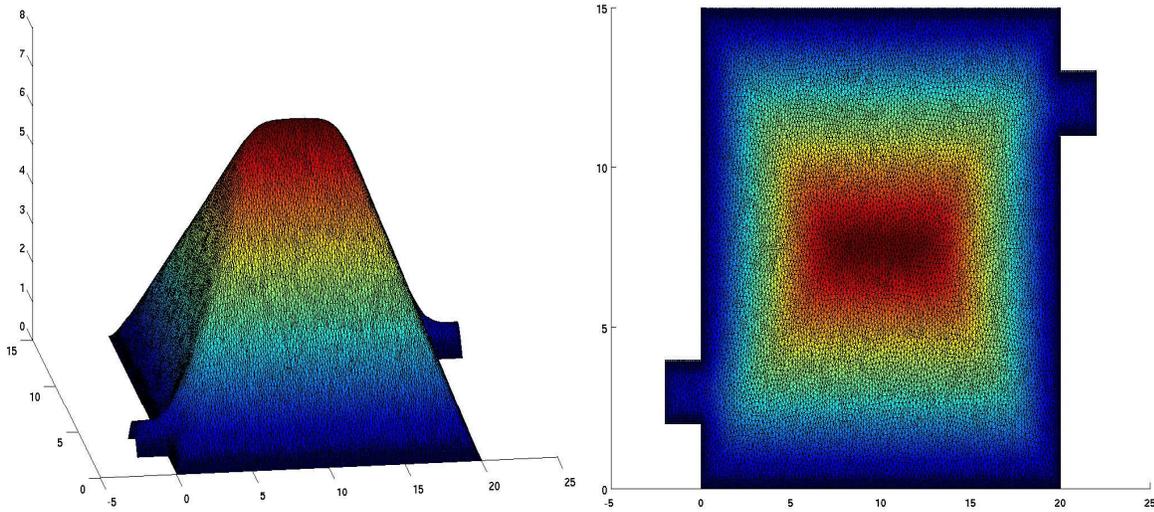


Figure 3. Angle and top views of the converged iterative solution of Eq. (.045) for $\varepsilon = .05$ on a sample domain Ω with inlet (lower left) and outlet (upper right) boundaries.

Using this linearized representation for $(\vec{\nabla} \ell^*) \cdot (\vec{\nabla} \ell^*)$ in (.043), we thus arrive at a functional iteration scheme that computes the successive approximation ℓ_{i+1}^* in terms of an initial guess ℓ_i^* according to

$$\int_{\Omega} \left\{ r \left[(\vec{\nabla} \ell_i^*) \cdot (2\vec{\nabla} \ell_{i+1}^* - \vec{\nabla} \ell_i^*) - 1 \right] + \varepsilon (\vec{\nabla} r) \cdot (\vec{\nabla} \ell_{i+1}^*) \right\} d\Omega + \varepsilon \int_{\partial\Omega} r \ell_{i+1}^* d(\partial\Omega) = 0. \quad (.045)$$

Equation (.045) is linear with respect to the unknown function ℓ_{i+1}^* and can be handled directly by Sundance. The sequence of approximations is expected to be convergent given even a crude starting guess ℓ_0^* for sufficiently large ε , since in that limit the original nonlinearity becomes a perturbation of an otherwise linear problem. An outer iteration scheme can then be used to generate sufficiently good starting guesses ℓ_0^* for the recursive algorithm (.045) with successively smaller values of ε . An example of such a calculation is illustrated in Figure 3.

References

- [1] *EPANET 2.0 User's Manual*.
- [2] *International Organization of Standards. Draft ISO Guide 73*.
- [3] *Introduction to Probability Theory and its Applications*. John Wiley and Sons, 1957.
- [4] *An Anatomy of Risk*. John Wiley & Sons, 1977.
- [5] *Hydraulics of Groundwater*. McGraw-Hill, New York, 1979.
- [6] *Spectral Analysis and Time Series*. Univariate Series, Academic Press, 1981.
- [7] *Reliability-Based Design in Civil Engineering*. Dover Publishing Inc., 1987.
- [8] *Ecological Risk Assessment*. Lewis Publishers, 1993.
- [9] *Stochastic Subsurface Hydrology*. Prentice Hall,, 1993.
- [10] *Risk assessment for management, a unified approach. Risk management handbook for environmental, health, and safety professional*. McGraw-Hill, 1996.
- [11] *Disaster planning and recovery: a guide for facility professionals*. Wiley, 1997.
- [12] *Managing the Risks of Organizational Accidents*. Ashgate, 1997.
- [13] *The Design and Evaluation of Physical Protection Systems*. Butterworth-Heinmann Publishers, 2001.
- [14] Feby Abraham, Marek Behr, and Matthias Heinkenschloss. The effect of stabilization in finite element methods for the optimal boundary control of the Oseen equations. *Finite Elements in Analysis and Design*, 41(3):229–251, 2004.

- [15] V.M. Adamjan, D.Z. Arov, and M.G. Krein. Analytic properties of Schmidt pairs for a Hankel operator and the generalized Schur-Takagi problem. *Math. USSR Sbornik*, 15:31–73, 1971.
- [16] V. Akcelik, J. Bielak, G. Biros, I. Epanomeritakis, A. Fernandez, O. Ghattas, E. Kim, J. Lopez, D. O’Hallaron, T. Tu, and J. Urbanic. Terascale forward and inverse earthquake modeling. In *Proceedings of SC2003*, 2003.
- [17] V. Akcelik, G. Biros, and O. Ghattas. Parallel multiscale Gauss-Newton-Krylov methods for inverse wave propagation. In *Proceedings of SC2002*, 2002.
- [18] V. Akcelik, G. Biros, O. Ghattas, K. R. Long, and B. van Bloemen Waanders. A variational finite element method for source inversion for convective-diffusive transport. *Finite Elements in Analysis and Design*, 39(8):683–705, 2003.
- [19] J. Albery, C. Carstensen, and S. Funken. Remarks around 50 lines of matlab: Short finite element implementation. *Numerical Algorithms*, 20:117–137, 1999.
- [20] S. Aliabadi, R. Baffour, J. Abedi, A. Ji, M. Watts, A. Fuller, K. Bota, N. Regina, and A. Johnson. Large scale simulations of contaminant dispersions in urban area. In *the 7th Annual George Mason University Workshop on Atmospheric Transport and Dispersion Modeling*.
- [21] S. Aliabadi, S. Tu, M. Watts, A. Ji, and A. Johnson. Integrated high performance computational tools for simulations of transport and diffusion of contaminants in urban areas. *International Journal of CFD*, 2005.
- [22] E.D. Anderson. *The MOSEK Optimization Toolbox for MATLAB Version 2.5, User’s Guide and Reference Manual*. World Wide Web, <http://www.mosek.com>, 1999–2002.
- [23] I.P. Androulakis, C.D. Maranas, and C.A. Floudas. α BB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7:337–363, 1995.
- [24] E.N. Antoun, J.E. Dykseen, and D.J. Hildebrand. Unidirectional flushing: a powerful tool. *Journal of the American Water Works Association*, 1999.
- [25] E. Asbeck and Y.Y. Haimes. The partitioned multiobjective risk method. *Large Scale Sys.*, 1984.
- [26] U. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential Algebraic Equations*. SIAM, Philadelphia, PA, 1998.
- [27] R. Bahadur, W. B. Samuels, W. Grayman, D. Amstutz, and J. Pickus. Pipelinenet: A model for monitoring introduced contaminants in a distribution system. In *Proc., World Water & Environmental Resources Congress 2003 and Related Symposia*. ASCE, 2003. CD-ROM.

- [28] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Lois Curfman McInnes, and Barry F. Smith. PETSc home page. <http://www.mcs.anl.gov/petsc>, 2001.
- [29] Roscoe A. Bartlett. *Object Oriented Approaches to Large-Scale Nonlinear Programming for Process Systems Engineering*. PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 2001.
- [30] G. Berkooz, P. Holmes, and J.L. Lumley. The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows. *Annual Review of Fluid Mechanics*, 25:539–575, 1993.
- [31] J. Berry, L. Fleischer, W. E. Hart, and C. A. Phillips. Sensor placement in municipal water networks. In P. Bizier and P. DeBarry, editors, *Proc. World Water and Environmental Resources Congress*. American Society of Civil Engineers, 2003.
- [32] J. Berry, L. Fleischer, W. E. Hart, C. A. Phillips, and J.-P. Watson. Sensor placement in municipal water networks. *J. Water Planning and Resources Management*, 131(3):237–243, May 2005.
- [33] J. Berry, L. Fleischer, W.E. Hart, C.A. Phillips, and J.P. Watson. Sensor placement in municipal water networks. *Journal of Water Resources Planning and Management*, 2005.
- [34] J. Berry, W. E. Hart, C. A. Phillips, and J. Uber. A general integer-programming-based framework for sensor placement in municipal water networks. In *Proc. World Water and Environment Resources Conference*, 2004.
- [35] Jonathan Berry, Lisa Fleischer, William Hart, and Cynthia Phillips. Optimal sensor placement in municipal water networks. In *Proceedings of EWRI*.
- [36] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming Series B*, 98:49–71, 2003.
- [37] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52:35–53, 2004.
- [38] M. Bettayeb, L.M. Silverman, and M.G. Safonov. Optimal Approximation of Continuous-Time Systems. *Proceedings of the 19th IEEE Conference on Decision and Control*, 1, December 1980.
- [39] L. T. Biegler, A. M. Cervantes, and A. Wächter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 57:575, 2002.
- [40] Lorenz T. Biegler, Omar Ghattas, Matthias Heinkenschloss, and Bart van Bloemen Waanders, editors. *Large-Scale PDE-Constrained Optimization*, volume 30 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Heidelberg, 2003. to appear.

- [41] Lorenz T. Biegler, Ignacio E. Grossmann, and Arthur W. Westerberg. *Systematic Methods of Chemical Process Design*. Prentice Hall International Series in the Physical and Chemical Engineering Sciences. Prentice Hall PTR, Upper Saddle River, New Jersey 07458, 1997.
- [42] R. B. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport Phenomena*. John Wiley & Sons, New York, 1960.
- [43] George Biros and Omar Ghattas. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: The Krylov-Schur solver. *SIAM Journal on Scientific Computing*, 2005. In press.
- [44] George Biros and Omar Ghattas. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: The Lagrange Newton solver, and its application to optimal control of steady viscous flows. *SIAM Journal on Scientific Computing*, 2005. In press.
- [45] P. T. Boggs, A. J. Kearsley, and J. W. Tolle. Hierarchical control of a linear diffusion equation. *in press*, 2002.
- [46] Paul T. Boggs, Paul D. Domich, and Janet E. Rogers. An interior-point method for general large scale quadratic programming problems. *Annals of Operations Research*, 62:419–437, 1996.
- [47] Paul T. Boggs, Anthony J. Kearsley, and Jon W. Tolle. A global convergence analysis of an algorithm for large scale nonlinear programming problems. *SIAM Journal on Optimization*, 9(4):833–862, 1999.
- [48] Paul T. Boggs, Anthony J. Kearsley, and Jon W. Tolle. A practical algorithm for general large scale nonlinear optimization problems. *SIAM Journal on Optimization*, 9(3):755–778, 1999.
- [49] M. Born and E. Wolf. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, Cambridge, 1999.
- [50] R. Borum, R. Fein, B. Vossekuil, and J. Berglund. Threat assessment: Defining an approach for evaluating risk of targeted violence. *Behavioral Sciences and the Law*, 1999.
- [51] J. Boussinesq. Essai sur la theorie des eaux courantes. *Memoirs Acad. des Sciences*, 23, 1872.
- [52] James Bramble. *Multigrid methods*, volume 294 of *Pitman Research Notes in Mathematics Series*. Longman Scientific, 1993.
- [53] M.M. Polycarpou J.G. Uber Z. Wang F. Shang M. Brdys. Feedback control of water quality. *I.E.E.E. Control Systems Magazine*, 2002.

- [54] M. Brezina. *Robust iterative solvers on unstructured meshes*. PhD thesis, University of Colorado at Denver, 1997.
- [55] M. Brezina and P. Vaněk. A black-box iterative solver based on a two-level Schwarz method. *Computing*, 63:233–263, 1999.
- [56] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.
- [57] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32:199–259, 1982.
- [58] M. Brown. Urban dispersion - challenges for fast response modeling. In *5th AMS Urban Env. Conf.*
- [59] P.N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comp.*, 11:450–481, 1990.
- [60] S.G. Buchberger, J.T. Carter, Y.H. Lee, and T.G. Schade. Random demands, travel times, and water quality in deadends. Technical report, American Water Works Association Research Foundation, 2003.
- [61] S.G. Buchberger, J.T. Carter, and Y.Lee. Travel time in dead-end mains. In *Proceedings ASCE 25th Annual Conference on Water Resources Planning and Management*.
- [62] S.G. Buchberger, Carter J.T., Y. Lee, and T.G Schade. Random demands, travel times and water quality in deadends.
- [63] S.G. Buchberger and T.G. Schade. Poisson rectangular pulse model for residential water use. In *Proceedings XXVII IAHR Congress*.
- [64] S.G. Buchberger and G.J. Wells. Intensity, duration and frequency of residential water demands. *Journal of Water Resources Planning and Management, ASCE*, 1996.
- [65] S.G. Buchberger and L. Wu. A model for instantaneous residential water demands. *Journal of Hydraulic Engineering, ASCE*, 1995.
- [66] R. Carr, H. J. Greenberg, W. E. Hart, G. Konjevod, E. Lauer, H. Lin, T. Morrison, and C. A. Phillips. Robust optimization of contaminant sensor placement for community water systems. *Mathematical Programming B*. (to appear).
- [67] T. Cebeci and J. Cousteix. *Modeling and Computation of Boundary-Layer Flows*. Horizons Publishing/Springer-Verlag, Long Beach/Berlin, 1999.

- [68] A. Cervantes and L. T. Biegler. Optimization strategies for dynamic systems. In C. Floudas, P. Pardalos (Eds.), *Encyclopedia of Optimization*, 2003.
- [69] E. Chow. An unstructured multigrid method based on geometric smoothness. *Numer. Linear Algebra Appl.*, 10:401–421, 2003.
- [70] T.F. Coleman and Y. Li. An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6:418–445, 1996.
- [71] S. Scott Collis. Monitoring unresolved scales in multiscale turbulence modeling. 13(6):1800–1806, 2001.
- [72] S. Scott Collis. The DG/VMS method for unified turbulence simulation. *AIAA Paper 2002-3124*, 2002.
- [73] S. Scott Collis. Discontinuous Galerkin methods for turbulence simulation. In *Proceedings of the 2002 Center for Turbulence Research Summer Program*, pages 155–167, 2002.
- [74] S. Scott Collis and Kaveh Ghayour. Discontinuous Galerkin for compressible DNS. *ASME paper number FEDSM2003-45632*, 2003.
- [75] S. Scott Collis and Srinivas Ramakrishnan. The local variational multiscale method. In *Proceedings of the Third MIT Conference on Computational Fluid and Solid Dynamics*, June 2005.
- [76] J. Cooper. Soft markov chain relations for modeling organizational behavior. *Risk Decision and Policy*, 2004.
- [77] E. B. Cummings, S. K. Griffiths, and R. H. Nilson. Applied microfluidic physics LDRD final report. Technical report, Sandia National Laboratories, 2002.
- [78] Eric Cummings. Private communication. 2003.
- [79] D. N. Daescu and G. R. Carmichael. An adjoint sensitivity method for the adaptive location of the observations in air quality modeling. *Journal of Atmospheric Sciences*, 60(2):434–450, 2003.
- [80] L. Daniel, O.C. Siong, L.S. Chay, K.H. Lee, and J. White. Multiparameter moment matching model reduction approach for generating geometrically parameterized interconnect performance models. *Trans. on Computer Aided Design of Integrated Circuits*, 23(5):678–693, May 2004.
- [81] J.J. Danneels and R.E. Finley. Assessing the vulnerabilities of u.s. drinking water systems. *J. of Contemporary Water Research and Education*, 2004.

- [82] G. B. Dantzig. Linear programming under uncertainty. *Management Science*, 1(3, 4):197–206, 1955.
- [83] J. W. Demmel, J. R. Gilbert, and X. S. Li. SuperLU users' guide. Technical Report LBNL-44289, Lawrence Berkeley National Laboratory, October 2003.
- [84] K. Devine, E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science and Engineering*, 4(2):90–97, 2002.
- [85] Andrei Drăgănescu. *Two investigations in numerical analysis: Monotonicity preserving finite element methods, and multigrid methods for inverse parabolic problems*. PhD thesis, University of Chicago, August 2004.
- [86] Andrei Drăgănescu and Todd F. Dupont. Optimal order multi-level preconditioners for regularized ill-posed problems, 2005. To appear.
- [87] S. J. Owen ed. Cubit 9.0 users manual. Sandia Report SAND94-1100 Rev. 4/2002, Sandia National Laboratories, June 2004.
- [88] S.C. Eisenstat and H.F. Walker. Globally convergent inexact Newton methods. *SIAM J. Optimization*, 4:393–422, 1994.
- [89] B.C. Ezell, J.V. Farr, and I. Wiese. Infrastructure risk analysis model (a). *J. Infrastruct. Syst.*, 2000.
- [90] B.C. Ezell, J.V. Farr, and I. Wiese. Infrastructure risk analysis model (b). *J. Infrastruct. Syst.*, 2000.
- [91] Vossekul B. Fein R.A. Protective intelligence threat assessment investigations: A guide for state and local law enforcement officials. Technical report, Department of Justice, 1998.
- [92] Scott Ferson, Cliff A. Joslyn, Jon C. Helton, William L. Oberkampf, and Kari Sentz. Challenge problems: Uncertainty in system response given uncertain parameters. *Reliability Engineering and System Safety*, 85:11–19, 2004.
- [93] AWWA Research Foundation and US Environmental Protection Agency. Actual and threatened security events at water utilities. Technical report, AWWA and EPA, 2003.
- [94] AWWA Research Foundation and Sandia National Laboratories. Risk assessment methodology for water utilities. Technical report, AWWA and Sandia National Laboratories, 2002.
- [95] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. The Scientific Press (now an imprint of Boyd & Fraser Publishing Co.), Danvers, MA, USA, 1993.

- [96] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. The Scientific Press (now an imprint of Boyd & Fraser Publishing Co.), Danvers, MA, USA, 1993.
- [97] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, 2nd edition, 2002.
- [98] A. Gadgil, Christian Lobscheid, Marc Abadie, and Elizabeth Finlayson. Indoor pollutant mixing time in an isothermal closed room: an investigation using cfd. *Atmospheric Environment Journal*, 2003.
- [99] T. B. Gatski. Turbulent flows: Model equations and solution methodology. In R. Peyret, editor, *Computational Fluid Mechanics*, London, 1996. Academic Press.
- [100] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, Orlando, San Diego, New-York, . . ., 1981.
- [101] H.J. Greenberg. *Mathematical Programming Glossary*. World Wide Web, <http://www.cudenver.edu/~hgreenbe/glossary/>, 1996–2004.
- [102] P. M. Gresho and R. L. Sani. *Incompressible Flow and the Finite Element Method*. John Wiley & Sons, New York, 2000.
- [103] N.S. Grigg. Water utility security: Multiple hazards and multiple barriers. *J. Infrastruct. Syst.*, 2003.
- [104] W.D. Gropp, D.K. Kaushik, D.E. Keyes, and B.F. Smith. High-performance parallel implicit CFD. *Parallel Computing*, 27:337–362, 2001.
- [105] S. Gugercin and A. Antoulas. A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77:748–766, 2004.
- [106] Wolfgang Hackbusch. *Multigrid methods and applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1985.
- [107] Y.Y. Haimes. Hierarchical holographic modeling. *IEEE Trans. On Sys., Man and Cybernetics*,, 1981.
- [108] Y.Y. Haimes, N.C. Matalas, J.H. Lambert, B.A Jackson, and J.F.R Fellows. Reducing vulnerability of water supply systems to attack. *J. Infrastruct. Syst.*, 1998.
- [109] Martin Hanke and Curtis R. Vogel. Two-level preconditioners for regularized inverse problems. I. Theory. *Numerische Mathematik*, 83(3):385–402, 1999.

- [110] Per Christian Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, 1997.
- [111] Frédéric Hecht. *BAMG: Bidimensional Anisotropic Mesh Generator*, October 1998. <http://www-rocq1.inria.fr/gamma/cdrom/www/bamg/eng.htm>.
- [112] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.
- [113] S. Hinze, M. and Volkwein. Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control. Technical Report SFB609-Preprint-27-2004, Technische Universitat Dresden, 2004.
- [114] F. O. Hoffman and J. S. Hammonds. Propagation of uncertainty in risk assessments: The need to distinguish between uncertainty due to lack of knowledge and uncertainty due to variability. *Risk Analysis*, 14(5):707–712, 1994.
- [115] P. Holmes, J.L. Lumley, and G. Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, Cambridge, UK, 1996.
- [116] Thomas J. R. Hughes, Luca Mazzei, and Kenneth E. Jansen. Large eddy simulation and the variational multiscale method. *Comp Vis Sci*, 3:47–59, 2000.
- [117] T.J.R. Hughes, L.P. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics: V. circumventing the Babuska-Brezzi condition: a stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Comp. Meth. AMech. and Eng.*, 59:85–99, 1986.
- [118] S.A. Hutchinson, L. Prevost, J.N. Shadid, C. Tong, and R.S. Tuminaro. Aztec user’s guide version 2.0. Sandia National Laboratories Technical Report SAND 99-8801J, 1999.
- [119] L. Jenkins, T. Kelley, C.T. Miller, and C.E. Kees. An aggregation-based domain decomposition preconditioner for groundwater flow. Technical Report TR00–13, Department of Mathematics, North Carolina State University, 2000.
- [120] S.A. Hutchinson G.L. Hennigan K.D Devine A.G. Salinger J.N. Shadid H.K. Moffat. Mpsalsa a finite element computer program for reacting flow problems part 1 - theoretical development. Technical Report SAND95-2752, Sandia National Laboratories, 1995.
- [121] Barbara Kaltenbacher. V-cycle convergence of some multigrid methods for ill-posed problems. *Mathematics of Computation*, 72(244):1711–1730, 2003.

- [122] T. Kaminski and M. Heimann. A coarse grid three-dimensional global inverse model of the atmospheric transport i. Adjoint model and Jacobian matrix. *Journal on Geophysics*, 104(D15):18535–18663, 1999.
- [123] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. Army HPC Research Center Technical Report 95-064, 1995.
- [124] P. Kastner-Klein and J.V. Clark. Modeling of flow and dispersion characteristics in typical building configurations with the fast-response model quic. In *5th AMS Urban Env. Conf.*
- [125] A. Kessler, A. Ostfeld, and G. Sinai. Detecting accidental contaminations in municipal water networks. *Journal of Water Resources Planning and Management*, 124(4):192–198, 1998.
- [126] J. Thomas King. Multilevel algorithms for ill-posed problems. *Numerische Mathematik*, 61(3):311–334, 1992.
- [127] G. Kontarev. Adjoint equation technique applied to meteorological problems. Technical Report 21, European Center for Medium Range Weather Forecasts, Reading, England, 1980.
- [128] B. Kosovic and S.T Chan. Integrated urban dispersion modeling capability. In *Symposium on Planning, NowCasting and Forecasting in the Urban Zone 84th AMS Annual Meeting*.
- [129] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, Norwell, MA, 1996.
- [130] A. Kumar, M. L. Kansal, and G. Arora. Discussion of ‘detecting accidental contaminations in municipal water networks’. *Journal of Water Resources Planning and Management*, 125(4):308–310, 1999.
- [131] A. Kumar, M.L. Kansal, and G. Arora. Discussion of “Detecting accidental contaminations in municipal water networks”. *Journal of Water Resources Planning and Management*, 124(4):308–310, September/October 1998.
- [132] S-Y. Kung and D.W. Lin. Optimal Hankel-norm model reductions: Multivariable systems. *IEEE Transactions on Automatic Control*, AC-26(1):832–52, August 1981.
- [133] K. Kunisch and S. Volkwein. Control of Burgers’ equation by reduced order approach using proper orthogonal decomposition. *Journal of Optimization Theory and Applications*, 102:345–371, 1999.
- [134] C. D. Laird, L. T. Biegler, B. van Bloemen Waanders, and R. A. Bartlett. Time dependent contaminant source determination for municipal water networks using large scale optimization. *Journal of Water Resources Planning and Management*, to appear, 2004.

- [135] C. D. Laird, L. T. Biegler, B. van Bloemen Waanders, and R. A. Bartlett. Real-time, large scale optimization of water network systems using a subdomain approach. *bad reference*, to appear, 2005.
- [136] C.D. Laird, L.T. Biegler, B.G. van Bloemen Waanders, and R.A. Bartlett. Contamination source determination for water networks. *A.S.C.E. Journal of Water Resources Planning and Management*, March/April 2005.
- [137] C.D. Laird, L.T. Biegler, B.G. van Bloemen Waanders, and R.A. Bartlett. Contamination source determination for water networks. *Journal of Water Resources and Planning*, 2005.
- [138] S. Lall, J.E. Marsden, and S. Glavaski. A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal on Robust and Nonlinear Control*, 12(5):519–535, 2002.
- [139] P. K. Lamm. Inverse problems and ill-posedness. *Inverse Problems in Engineering, Theory and Practice, Americal Society of Mechanical Engineers*, pages 1–10, 1993.
- [140] C. Lasser and A. Toselli. An overlapping domain decomposition preconditioner for a class of discontinuous Galerkin approximations of advection-diffusion problems. *Mathematics of Computation*, (72):1215–1238, 2003.
- [141] B. H. Lee and R. A. Deininger. Optimal locations of monitoring stations in water distribution system. *Journal of Environmental Engineering*, 118(1):4–16, 1992.
- [142] B. H. Lee, R. A. Deininger, and R. M. Clark. Locating monitoring stations in water distribution systems. *Journal, Am. Water Works Assoc.*, pages 60–66, 1991.
- [143] M. Lesieur and O. Métais. New trends in large-eddy simulations of turbulence. 28:45–82, 1996.
- [144] S. Leyffer. Integrating sqp and branch-and-bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18:295–305, 2001.
- [145] J. Li and J. White. Low rank solution of Lyapunov equations. *SIAM Journal on Matrix Analysis and Applications*, 24(1):260–280, 2002.
- [146] X.S. Li. An overview of SuperLU: Algorithms, implementation, and user interface. Technical Report LBNL-53848, 2003.
- [147] C. P. Liou and J. R. Kroon. Modeling the propogation of water-borne substances in distribution networks. *Journal of American Water Works Assoc.*, 79(11):54–58, 1987.
- [148] K. R. Long. Sundance users manual. Computer science and mathematics research department, Sandia National Laboratories, 2004, to appear.

- [149] Kevin R. Long. Sundance: A rapid prototyping tool for parallel pde-constrained optimization. In L. Biegler, O. Ghattas, M. Heinkenschloss, and B. van BloemenWaanders, editors, *Large-Scale, PDE-Constrained Optimization*, volume 30 of *Lecture Notes in Computational Science and Engineering*, Heidelberg, 2003. Springer-Verlag.
- [150] D.M. Lorenzetti. Computational aspects of nodal multizone airflow systems. *Buildings and Environment*, 2002.
- [151] C. Malivert. An algorithm for bilinear fractional problems. Technical Report 1998-03, Université de Limoges, Cedex, France, 1998.
- [152] G.I. Marchuk and V.V. Penenko. Study of the sensitivity of discrete models of atmospheric and oceanic dynamics. *Atmospheric and Oceanic Physics*, 15(11):785–789, 1980.
- [153] S.A. McKenna, S. Buchberger, and V.C. Tidwell. Examining the effects of variability in short time scale demands on solute transport.
- [154] R. Michel, C. Quémard, and R. Durant. Application d’un schéma de longueur de mélange à l’études des couches limites d’équilibre. ONERA Note Technique 154, ONERA, 1969.
- [155] P.S. Mirchandani and R.L. Francis, editors. *Discrete Location Theory*. John Wiley and Sons, 1990.
- [156] Parviz Moin and Krishnan Mahesh. Direct numerical simulation: A tool in turbulence research. *Annual Reviews in Fluid Mechanics*, 30:539–78, 1998.
- [157] B.C. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, AC-26(1):17–31, August 1981.
- [158] NARAC. Operational responses. <http://narac.llnl.gov/operational.html>.
- [159] The MM5 community mesoscale model. <http://www.mmm.ucar.edu/mm5>.
- [160] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Verlag, 2000.
- [161] William L. Oberkampf, Sharon M. DeLand, Brian M. Rutherford, Kathleen V. Diegert, and Kenneth F. Alvin. Error and uncertainty in modeling and simulation. *Reliability Engineering and System Safety*, 75:333–357, 2002.
- [162] A. Ostfeld and A. Kessler. Protecting urban water distribution systems against accidental hazards intrusions. In *Proceedings IWA Second Conference*. IWA, 2001. CD-ROM.
- [163] A. Ostfeld and E. Salomons. An early warning detection system (ewds) for drinking water distribution systems security. *Proceedings of the EWRI Conference, Philadelphia*, 2003.

- [164] A. Ostfeld and E. Salomons. Optimal layout of early warning detection stations for water distribution systems security. *Journal of Water Resources Planning and Management*, 130(5):377–385, 2004.
- [165] E.R. Pardyjak, N. L. Bagal, and M. J. Brown. Improved velocity deficit parameterizations for a fast response urban wind model. In *AMS Conf. on Urban Zone*.
- [166] U. Piomelli and E. Balaras. Wall-layer models for large eddy simulations. 34:349–374, 2002.
- [167] F. Shang J. Uber M. Polycarpou. A particle backtracking algorithm for water distribution systems analysis. *A.S.C.E. Journal of Environmental Eng*, 2002.
- [168] S. B. Pope. *Turbulent Flows*. Cambridge University Press, Cambridge, 2000.
- [169] L. Prandtl. Bericht über untersuchungen zur ausgebildeten turbulenz. *ZAMM. Z. angew. Math. Mech.*, 5:136–139, 1925.
- [170] M. Propato, O. Piller, and J. Uber. A sensor location model to detect contaminations in water distribution networks. In *Proc. World Water and Environmental Resources Congress*. American Society of Civil Engineers, 2005.
- [171] C. Prud’homme, D. Rovas, K. Veroy, Y. Maday, A.T. Patera, and G. Turinici. Reliable real-time solution of parameterized partial differential equations: Reduced-basis output bound methods. *Journal of Fluids Engineering*, 124:70–80, 2002.
- [172] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, Oxford, 1999.
- [173] Srinivas Ramakrishnan. *Local Variational Multiscale Method for Turbulence Simulation*. PhD thesis, Rice University, 2004.
- [174] M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88, 2004.
- [175] C. ReVelle and R. Swain. Central facilities location. *Geographical Analysis*, 2:30–42, 1970.
- [176] Andreas Rieder. A wavelet multilevel method for ill-posed problems stabilized by Tikhonov regularization. *Numerische Mathematik*, 75(4):501–522, 1997.
- [177] Marielba Rojas. *A Large-Scale Trust-Region Approach to the Regularization of Discrete Ill-Posed Problems*. PhD thesis, Rice University, May 1998.

- [178] W. C. Rooney and L. T. Biegler. Optimal process design with model parameter uncertainty and process variability. *AIChE Journal*, 49(2):438, 2003.
- [179] L. A. Rossman. The EPANET programmer's toolkit for analysis of water distribution systems. In *Proceedings of the Annual Water Resources Planning and Management Conference*, 1999. Available at <http://www.epanet.gov/ORD/NRMRL/wswrd/epanet.html>.
- [180] L. A. Rossman. *EPANET User's Manual*. Risk Reduction Engineering Laboratory, U.S. EPA, Cincinnati, 2000.
- [181] L. A. Rossman and P. F. Boulos. Numerical methods for modeling water quality in distribution systems: A comparison. *Journal of Water Resources Planning and Management*, 122(2):137–146, March/April 1996.
- [182] L. A. Rossman, P. F. Boulos, and T. Altman. Discrete volume-element method for network water-quality models. *Journal of Water Resources Planning and Management*, 119(5), September/October 1993.
- [183] L.A. Rossman and P.F. Boulos. Numerical methods for modeling water quality in distribution systems: A comparison. *Journal of Water Resources PLanning and Management*, 1996.
- [184] M. Tryby D. Boccelli J. Uber L. Rossman. A facility location model for booster disinfection of water supply networks. *A.S.C.E. Journal of Water Resources Planning and Manag.*, 2002.
- [185] W. D. Rowe. Understanding uncertainty. *Risk Analysis*, 14(5):743–750, 1994.
- [186] J.W. Ruge and K. Stüben. Algebraic multigrid. in *Multigrid Methods*, Vol.3, *Frontiers in Applied Mathematics* 73-130, 1987.
- [187] H. S. Ryoo and N. V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers Chemical Engineering*, 19(5), 1995.
- [188] Y. Saad. ILUT: a dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994.
- [189] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.
- [190] N.V. Sahinidis and M. Tawarmalani. *BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual*, 2005.
- [191] M. Sala. *Domain Decomposition Preconditioners: Theoretical Properties, Application to the Compressible Euler Equations, Parallel Aspects*. PhD thesis, EPFL, Lausanne, Switzerland, 2003.

- [192] M. Sala. Amesos 2.0 reference guide. Technical Report SAND2004-4820, Sandia National Laboratories, September 2004.
- [193] M. Sala and L. Formaggia. Algebraic coarse grid operators for domain decomposition based preconditioners. In P. Wilders, A. Ecer, J. Periaux, N. Satofuka, and P. Fox, editors, *Parallel Computational Fluid Dynamics – Practice and Theory*, pages 119–126. Elsevier Science, The Netherlands, 2002.
- [194] M. Sala, J. Hu, and R. Tuminaro. ML 3.1 smoothed aggregation user’s guide. Technical Report SAND2004-4819, Sandia National Laboratories, September 2004.
- [195] M. Sala, J. Shadid, and R. Tuminaro. A field-of-value approach for smoothed aggregation preconditioners. Technical Report SAND2005–3333, Sandia National Laboratories, May 2005.
- [196] A.G. Salinger, J.N. Shadid, H. K. Moffat, S.A. Hutchinson, G.L. Hennigan, and K.D. Devine. Parallel reacting flow calculations for chemical vapor deposition reactor design.
- [197] M.A. Scharfenaker. Water suppliers, usepa, congress addressing terrorism threat. *Journal of AWWA*, 2002.
- [198] H. Schlichting and K. Gersten. *Boundary Layer Theory*. Springer-Verlag, Berlin, 2000.
- [199] R. Sextro and D. Lorenzetti. Modeling the spread of anthrax in buildings. In *Indoor Air 2002 Conference*.
- [200] Buchbeger S.G. and Y. Lee. Evidence supporting the poisson pulse hypothesis for residential water demands. In *Proceedings CCWI: International Conference on Computing and Control for the Water Industry*.
- [201] Buchberger S.G., J.T. Carter, Y. Lee, and T.G. Schade. Demands, travel times and water quality in deadends. Technical report, AwwaRF Project, 2003.
- [202] Buchberger S.G. and G.J. Wells. Intensity, duration and frequency of residential water demands. *Journal of Water Resources Planning and Management ASCE*, 1996.
- [203] Buchberger S.G. and L. Wu. A model for instantaneous residential water demands. *Journal of Hydraulic Engineering, ASCE*, 1995.
- [204] J. N. Shadid. A fully-coupled newton-krylov solution method for parallel unstructured finite element fluid flow, heat and mass transfer simulations. *Int. J. CFD*, 1999.
- [205] J. N. Shadid, S. A. Hutchinson, G. L. Hennigan, H. K. Moffat, K. D. Devine, and A. G. Salinger. Efficient parallel computation of unstructured finite element reacting flow solutions. *Parallel Computing*, 1997.

- [206] J.N. Shadid. A fully-coupled Newton-Krylov solution method for parallel unstructured finite element fluid flow, heat and mass transfer simulations. *Int. J. CFD*, 12:199–211, 1999.
- [207] J.N. Shadid, S.A. Hutchinson, G.L. Hennigan, H.K. Moffet, K.D. Devine, and A.G. Salinger. Efficient parallel computation of unstructured finite element reacting flow solutions. *Parallel Computing*, 23:1307–1325, 1997.
- [208] J.N. Shadid, A.G. Salinger, R.C. Schmidt, S.A. Hutchinson, G.L. Hennigan, K.D. Devine, and H.K. Moffat. MPSalsa: A finite element computer program for reacting flow problems part 1: Theoretical development. Technical Report SAND98-2864, Sandia National Laboratories, Albuquerque NM, 87185, January. 1999.
- [209] J.N. Shadid, R.S. Tuminaro, K.D. Devine, G.L. Hennigan, and P.T. Lin. Performance of fully-coupled domain decomposition preconditioners for finite element transport/reaction simulations. *Journ Comp Phys*, 205(1):24–47, 2005.
- [210] F. Shang, J. G. Uber, and M. M. Polycarpou. Particle backtracking algorithm for water distribution systems analysis. *Journal of Environmental Engineering*, pages 441–450, May 2002.
- [211] J. R. Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In M. C. Lin and D. Manocha, editors, *First Workshop on Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, 1996. From the First ACM Workshop on Applied Computational Geometry.
- [212] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22:21–74, 2002.
- [213] J. Siegel and W. Nazaroff. Predicting particle deposition on hvac heat exchangers. *Atmospheric Environment Journal*, 2003.
- [214] L. Sirovich. Turbulence and the dynamics of coherent structures. Part 1 : Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, October 1987.
- [215] B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel methods for elliptic partial differential equations*. Cambridge University Press, 1996.
- [216] M. Sohn, Pamela Reynolds, Ashok Gadgil, and Rich Sextro. Rapidly locating sources and predicting contaminant dispersion in buildings. In *Proceedings Indoor Air 2002 Conference*.
- [217] G. Sugiyama. National atmospheric release advisory center (narc), model development and evaluation. In *9th Int. Conf On Harmonisation within Atmospheric Dispersion Modelling for Regularatory Purposes*.

- [218] M. Tawarmalani and N.V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.
- [219] G.I. Taylor. The dispersion of matter in turbulent flow through a pipe.
- [220] G.I. Taylor. Dispersion of soluble matter in solvent flowing slowly through a tube.
- [221] T.E. Tezduyar. Stabilized finite element formulations for incompressible flow calculations. *Advances in App. Mech.*, 28:1–44, 1992.
- [222] V.C. Tidwell, J.A. Cooper, and C.J. Silva. Markov latent effects modeling for threat assessment of water distribution systems. *Journal of Water Resources Planning and Management*, 2005.
- [223] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, 2001.
- [224] R.S. Tuminaro, C.H. Tong, J.N. Shadid, K.D.Devine, and D.M. Day. On a multilevel preconditioning module for unstructured mesh Krylov solvers: two-level Schwarz. *Comm. Num. Method. Eng.*, 18:383–389, 2002.
- [225] V.G. Tzatchkov, A.A. Aldama, and F.I. Arreguin. Advection-dispersion-reaction modeling in water distribution networks. *Journal of Water Resources Planning and Management*, 2002.
- [226] J. Uber, R. Janke, and P. Murray, R.and Meyer. Greedy hueristic methods for locating water quality sensors.
- [227] M.L. Zierolf M.M. Polycarpou J.G. Uber. Development and auto-calibration of an input-output model of chlorine transport in drinking water distribution systems. In *I.E.E.E. Trans. on Control Systems Technology*.
- [228] B. van Bloemen Waanders, R. Bartlett, K. Long, P. Boggs, and A. Salinger. Large scale non-linear programming for pde constrained optimization. Technical Report SAND2002-3198, Sandia National Laboratories, 2002.
- [229] B. van Bloemen Waanders, R.A. Bartlett, K. Long, P.T. Boggs, and A. Salinger. Large scale nonlinear programming for pde constrained optimization. *SAND Report, SAND2002-3198*, 2002.
- [230] B. G. van Bloemen Waanders, R. A. Bartlett, L. T. Biegler, and C. D. Laird. Nonlinear programming strategies for source detection of municipal water networks. *Presented at EWRI Conference, Philadelphia, 2003*.
- [231] B. G. van Bloemen Waanders, R. A. Bartlett, L. T. Biegler, and C. D. Laird. Nonlinear programming strategies for source detection of municipal water networks. *Presented at EWRI Conference, Philadelphia, 2003*.

- [232] B.G. van Bloemen Waanders, R.A. Bartlett, L.T. Biegler, and C.D. C.D. Laird. Nonlinear programming strategies for source detection of municipal water networks.
- [233] B.G. van Bloemen Waanders, R.A. Bartlett, K.R. Long, P.T. Boggs, and A.G. Salinger. Large scale non-linear programming for pde constrained optimization. Technical report, Sandia National Laboratories, 2002.
- [234] P. Vanek, M. Brezina, and J. Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88:559–579, 2001.
- [235] C. R. Vogel. *Computational Methods for Inverse Problems*. SIAM, Frontiers in Applied Mathematics, 2002.
- [236] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Accepted for publication in Mathematical Programming*, to appear March 2004.
- [237] Andreas Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, January 2002.
- [238] T.M. Walski, E.D. Brill, J. Gessler, I.C. Goulter, R.M. Jeppson, K. Lansley, Lee Han-Lin, J.C. Liebman, L. Mays, D.R. Morgan, and L. Ormsbee. Battle of the network models: Epilogue. *Journal of Water Resources Planning and Management*, 113(2):191–203, 1987.
- [239] J.-P. Watson, H. J. Greenberg, and W. E. Hart. A multiple-objective analysis of sensor placement optimization in water networks. In *Proc. World Water and Environment Resources Conference*, 2004.
- [240] J-P. Watson, H. J. Greenberg, and W. E. Hart. A multiple-objective analysis of sensor placement optimization in water networks. In *Proceedings of the World Water and Environment Resources Congress*. American Society of Civil Engineers, 2004.
- [241] D. C. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, La Cañada, CA, 2000.
- [242] K.E. Willcox and J. Peraire. Balanced model reduction via the proper orthogonal decomposition. *AIAA Journal*, 40(11):2323–30, November 2002.
- [243] M. Williams and T. Yamada. A microcomputer-based forecasting model: potential application for emergency response plans and air quality studies. *Journal of Air Water Management Association*, 1990.
- [244] H. Yaman, O. E. Karasan, and M. C. Pinar. The robust minimum spanning tree problem with interval data. *Operations Research Letters*, 29:31–40, 2001.

- [245] H. Yaman, O. E. Karasan, and M. C. Pinar. Restricted robust optimization for maximization over uniform matroid with interval data uncertainty. Technical report, Bilkent University Technical Report, 2004.
- [246] M. L. Zierolf, M. M. Polycarpou, and J. G. Uber. Development and autocalibration of an input-output model of chlorine transport in drinking water distribution systems. *IEEE Transactions on Control Systems Technology*, 6(4):543–553, July 1998.

DISTRIBUTION:

- | | |
|---|---|
| <p>1 Dan Quintana, Tucson Water, P.O. Box 27210, Tucson, AZ 85726-7210</p> <p>1 Omar Ghattas, University of Texas, 1 University Station, Austin 78712</p> <p>1 Volkan Akcelik, CMU, 5000 Forbes Ave., Pittsburgh, PA 15213-3890</p> <p>1 Lorenz Biegler, CMU, 5000 Forbes Ave., Pittsburgh, PA 15213-3890</p> <p>1 Roger Bilisoly, CCSU, 1615 Stanley Street, New Britain, CT 06050</p> <p>1 Carl Laird, CMU, 5000 Forbes Ave., Pittsburgh, PA 15213-3890</p> <p>1 James Uber, University of Cincinnati, Dept. of Civil & Env. Eng., 741 Baldwin Hall; PO Box 210071, Cincinnati OH 45221-0071</p> <p>1 Steve Buchberger, University of Cincinnati, Dept. of Civil & Env. Eng., 741 Baldwin Hall; PO Box 210071, Cincinnati OH 45221-0071</p> <p>1 Zhiwei Li, University of Cincinnati, Dept. of Civil & Env. Eng., 741 Baldwin Hall; PO Box 210071, Cincinnati OH 45221-0071</p> | <p>1 Harvey Greenberg, Mathematics Department - Campus Box 170, University of Colorado at Denver, PO Box 173364, Denver, CO 80217-3364</p> <p>1 Goran Konjevod, Department of Computer Science and Engineering, Arizona State University, P.O. Box 878809, AZ 85287 - 8809</p> <p>1 Henry Lin, Mathematics Department - Campus Box 170, University of Colorado at Denver, PO Box 173364, Denver, CO 80217-3364 Department of Electrical Engineering and Computer Sciences, 253 Cory Hall, Berkeley, CA 94720-1770</p> <p>1 Tod Morrison, Mathematics Department - Campus Box 170, University of Colorado at Denver, PO Box 173364, Denver, CO 80217-3364</p> <p>1 Rob Janke, Environmental Protection Agency, Office of Research and Development Cincinnati, OH 45268</p> <p>1 Regan Murray, Environmental Protection Agency, Office of Research and Development Cincinnati, OH 45268</p> |
|---|---|

1 Sariah R. Bujanda, Department of Earth and Planetary Sciences, MSCO3-2040, University of New Mexico, Albuquerque, NM87131-0001

1 Glenn Hammond, Pacific Northwest Laboratories, P.O. Box 999, Richland, WA 99352

1 Karen Willcox, Department of Aeronautics & Astronautics Massachusetts Institute of Technology, 77 Massachusetts Ave Room 37-447, Cambridge, MA 02139

1 Marzio Sala ETHZ Computational Laboratory, Building CAB, Room F84, Universitaetstrasse 6, ETH Zentrum, 8092 Zuerich

1 MS 0847
William Camp, 1400

1 MS 0847
David Womble, 1400

1 MS 0310
Daniel Rintoul, 1410

1 MS 0847
Jennifer Nelson, 1430

1 MS 0847
Sudip Dosanjh, 1420

1 MS 1231
Billy Marshall, 8000

1 MS 9004
Jill Hruby, 8100

1 MS 9004
Duane Lindner, 8120

1 MS 9004
Patricia Falcone, 8110

1 MS 9201
Susanna Gordon, 8114

1 MS 9155
Howard Hirano, 8122

1 MS 1158
Martin Pilch, 1522

1 MS 0847
Scott Mitchell, 1411

5 MS 0847
Bart van Bloemen Waanders, 1411

1 MS 0847
Roscoe Barlett, 1411

1 MS 0847
Tim Trucano, 1411

1 MS 0847
Mike Eldred, 1411

1 MS 0847
Scott Mitchell, 1411

1 MS 0847
Andrei Draganescu, 1411

1 MS 0847
Judy Hill, 1411

1 MS 0847
Steve Thomas, 1411

1 MS 0847
David Gay, 1411

1 MS 9159
Heidi Ammerlahn, 8962

1 MS 9159
Kevin Long, 8962

1 MS 9159
Paul Boggs, 8962

1 MS 9051
Youssef Marzouk, 8351

1 MS 0701
Peter Davies, 6100

1 MS 0701
John Merson, 6110

1 MS 0847
Ray Finley, 6113

1 MS 0847
Sean McKenna, 6115

1 MS 0847
Chad Peyton, 6115

1 MS 0847
Lane Yarrington, 6115

1 MS 0847
Vincent Tidwell, 6115

1 MS 0615
Arlin Cooper, 6252

1 MS 1110
Scott Collis, 1414

1 MS 9159
Ray Tuminaro, 1414

1 MS 0316
Scott Hutchinson, 1437

1 MS 0316
John Shadid, 1437

1 MS 0316
Paul Lin, 1437

1 MS 1110
Andy Salinger, 1416

1 MS 0316
Brett Bader, 1416

1 MS 1110
Susan Rountree, 1415

1 MS 1110
William Hart, 1415

1 MS 1110
Cynthia Phillips, 1415

1 MS 1110
Jean-Paul Watson, 1415

1 MS 1110
Jon Berry, 1415

1 MS 1110
Bruce Hendrikson, 1415

1 MS 1110
Robert Carr, 1415

1 MS 1110
Vitus Lueng, 1415

1 MS 9159
Heidi Ammerlahn, 8962

1 MS 9159
Paul Boggs, 8962

1 MS 9159
Kevin Long, 8962

1 MS 9159
Stephen Margolis, 8962

1 MS 0836
Richard Griffith, 1517

1 MS 0945
Consueloj Silva, 10863

1 MS 9018
Central Technical Files, 8940-2

2 MS 0899
Technical Library, 4916

2 MS 123
LDRD Office, 1011