



# **Department Review:**

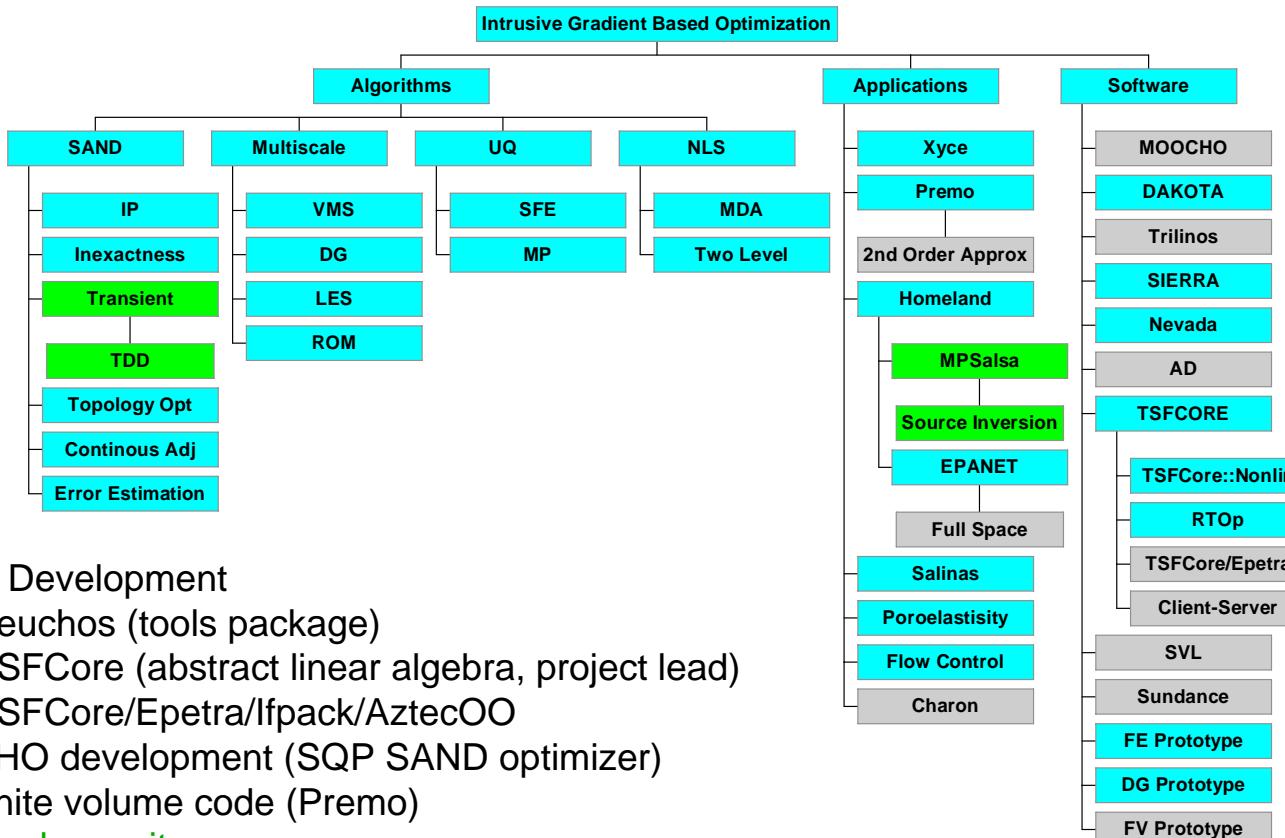
# **Intrusive Gradient-Based Optimization Part II**

**Roscoe A. Bartlett**

**9211, Optimization and Uncertainty Estimation**

**May 6, 2004**

# Intrusive Optimization Efforts

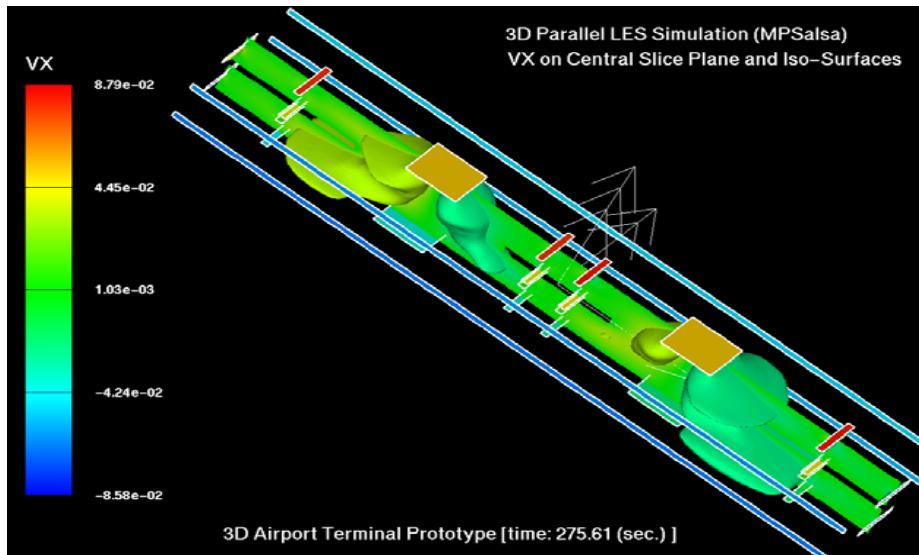


- Trilinos Development
  - Teuchos (tools package)
  - TSFCore (abstract linear algebra, project lead)
  - TSFCore/Epetra/Itpack/AztecOO
- MOOCHO development (SQP SAND optimizer)
- Euler finite volume code (Premo)
- **Homeland security**
  - Air chemical/biological source inversion
    - MOOCHO MPSalsa
    - Direct QP solver
- Transient optimization
  - Time domain decomposition (TDD)
  - TSFCore-based transient CG solver

## Collaborative Efforts

# Air Chem/Bio Source Inversion

## 3D Airport Terminal Model



## Collaborators:

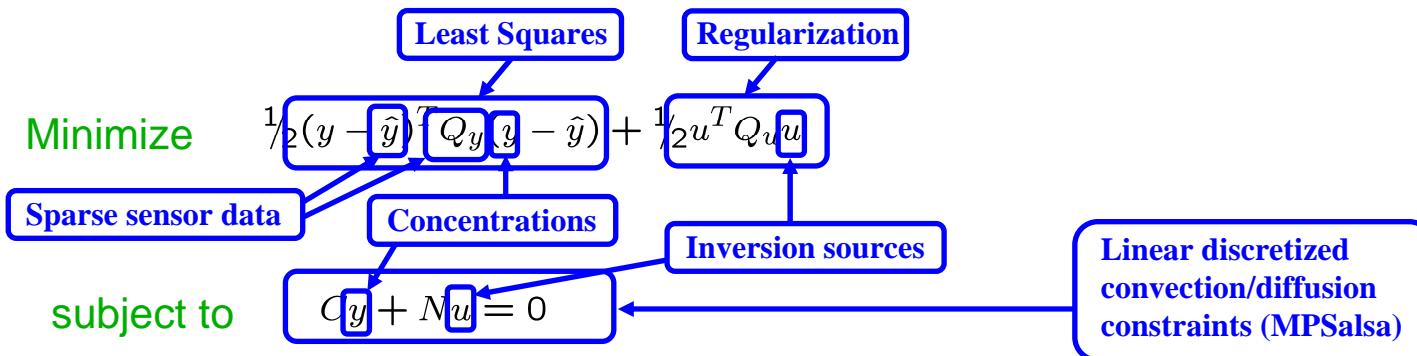
- Andy Salinger (9233)
- John Shadid (9233)
- Paul Lin (9233)
- Bart van Bloemen Waanders (9211)
- Roscoe A. Bartlett (9211)

## Requirements

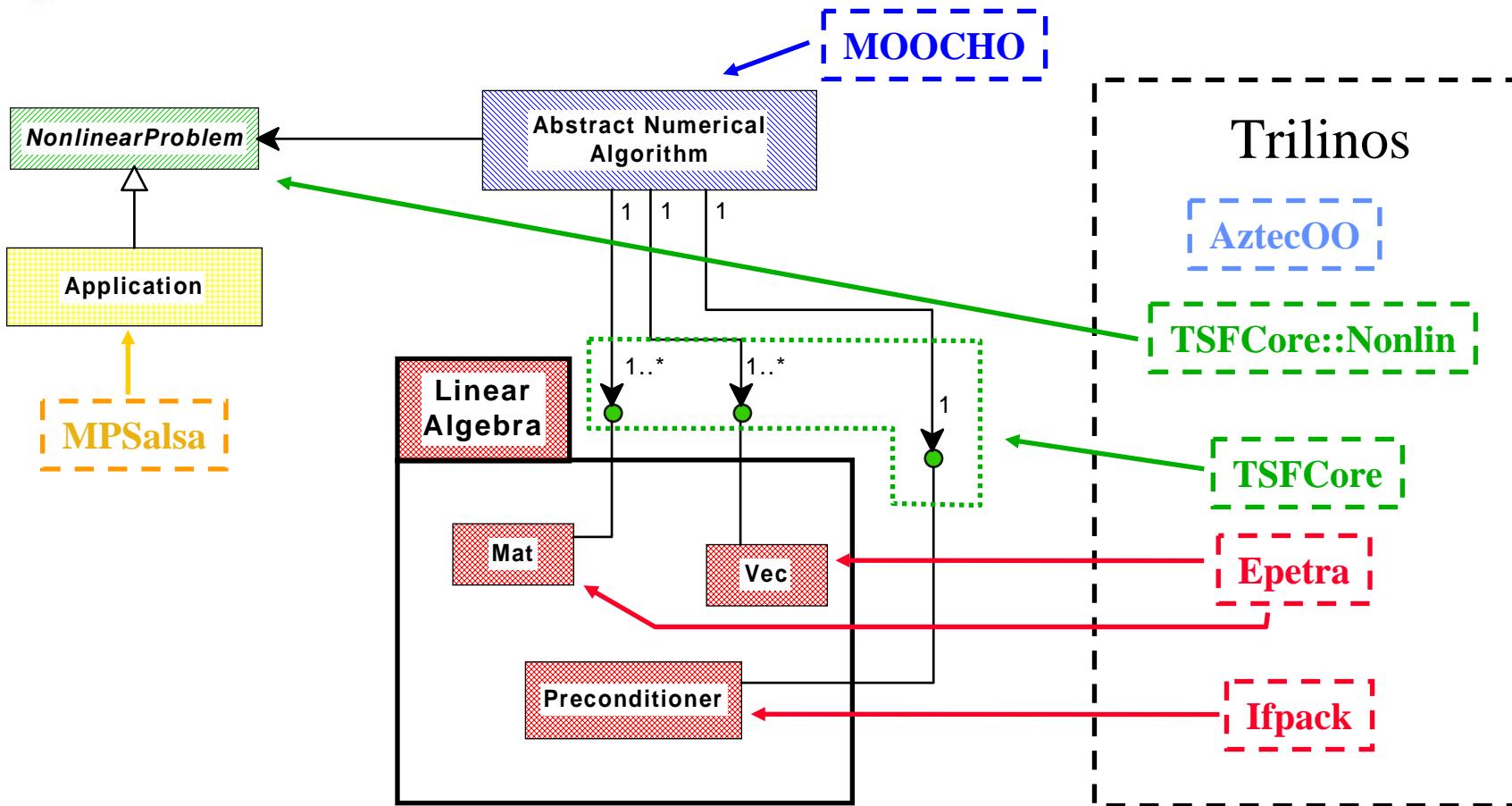
- Online inversion of 3D models (i.e. seconds!)

Real-time!

## Steady-State Source Inversion (Quadratic Program)



# MOOCHO / Trilinos / MPSalsa



- Developed infrastructure to enable parallel application codes for SAND optimization
- Solved CVD reactor problem (A. Salinger (9233)) with MPSalsa (48 processors)
- Solved (???) source inversion problem with quasi-Newton SQP algorithm in MOOCHO

# Specialized Direct QP Solver

## Reduced-space QP

$$\begin{aligned}\min \hat{f}(u) &= f(y(u), u) \\ &= \frac{1}{2}(y(u) - \hat{y})^T Q_y (y(u) - \hat{y}) + \frac{1}{2} u^T Q_u u\end{aligned}$$

where:

$$y(u) = y_0 + D(u - u_0)$$

$$D = -C^{-1}N$$

Linear elimination

- Direct sensitivity matrix
- Indep of attack scenario
  - Compute offline!

## Online solution of reduced-space QP:

$$\frac{\partial^2 \hat{f}}{\partial u^2} = D^T Q_y D + Q_u$$

$$\frac{\partial \hat{f}}{\partial u} = -D^T Q_y \hat{y}$$

$$\begin{aligned}\min \quad & \left( \frac{\partial \hat{f}}{\partial u} \right) u + \frac{1}{2} u^T \left( \frac{\partial^2 \hat{f}}{\partial u^2} \right) u \\ \text{s.t.} \quad & u \geq 0\end{aligned}$$

- Reduced Hessian
- Compute online
  - Factor online

- Reduced gradient
- Compute online

- Source inversion
- Compute online
  - BC QP solve

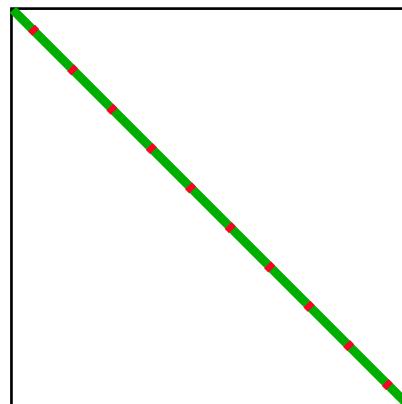
- Compressed Direct sensitivity matrix
- Indep of attack scenario
  - Compute offline!

- Reduced Hessian
- Compute online
  - Factor online

## Specialized online reduced-Hessian computation

- Example: 10 sensors

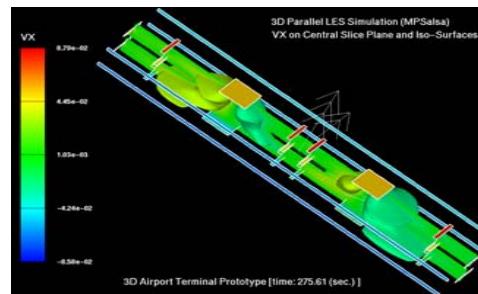
$$D^T Q_y D = \boxed{\text{Red}} \quad \boxed{\text{Red}} \quad \boxed{\text{Red}}$$



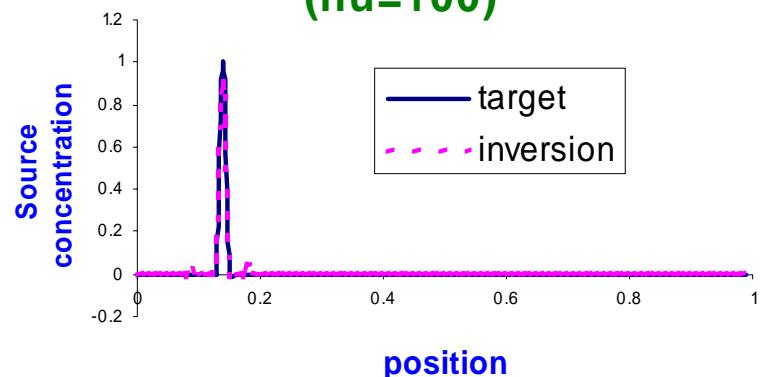
# Direct QP Solver : Results

## 3D Airport Terminal Model

- # nodes = 334184
- # state variables = 334184 (1 states per node)
- # inversion parameters = 100 and 500
- Platform : 4 processors on 2.0 GHz Linux cluster



## Source Inversion results (nu=100)



## Timing results

| Computation                       | nu=100    | nu=500     |
|-----------------------------------|-----------|------------|
| Compute direct sensitivity matrix | 28.03 min | 137.68 min |
| Compute reduced Hessian           | 0.016 sec | 0.14 sec   |
| Compute reduced gradient          | small     | small      |
| Solve for inversion parameters    | 0.038 sec | 2.25 sec   |

## Future Work

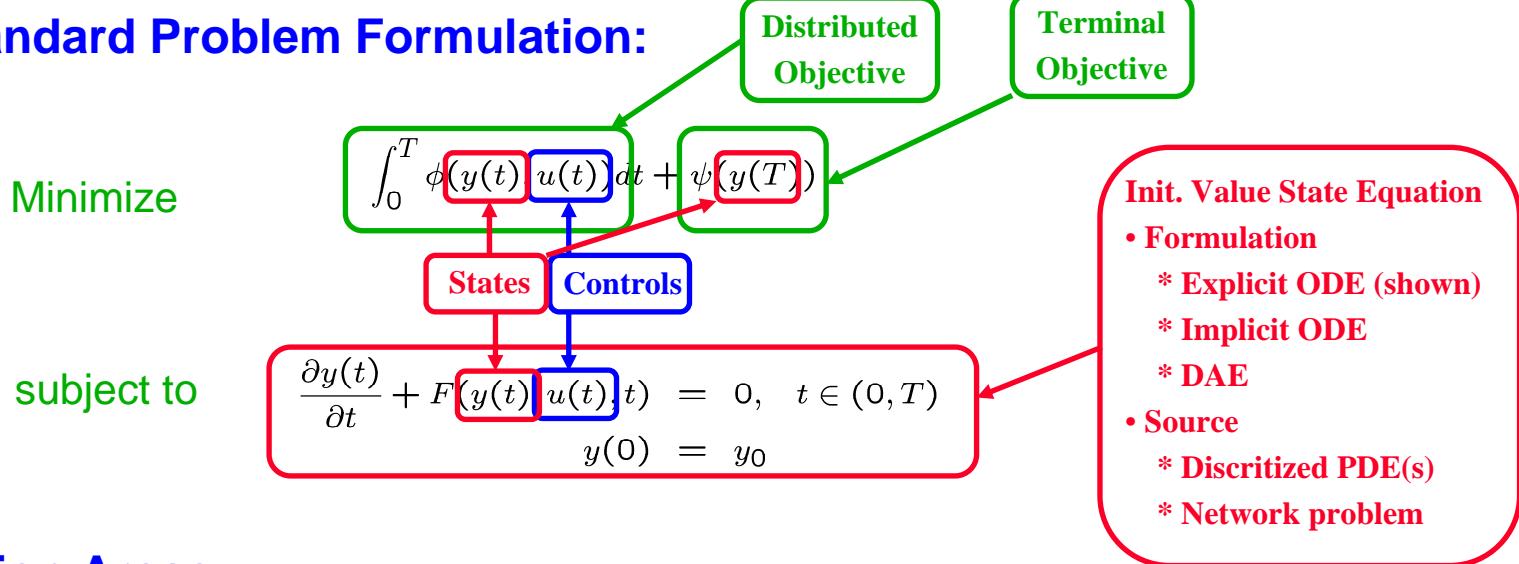
- Transient source inversion

### Online computations

- Solve of  $n_u = 100$  is less than 0.1 sec!
- Solve of  $n_u = 500$  is less than 3.0 sec!

# Large Scale Transient Optimization

## A Standard Problem Formulation:



## Application Areas:

- DHS
  - Chem/Bio Source Inversion
  - Chem/Bio Remediation (i.e. control)
- DP
  - Design
  - Control

## Solution methods for large $|U| > O(10^6)$

- Black box, direct methods ==> very expensive
- Adjoint methods ==> less expensive

## Collaborators:

- Matthias Heinkenschloss (Rice)
- **Roscoe A. Bartlett (9211)**
- Bart van Bloemen Waanders (9211)
- Scott Collis (9211)

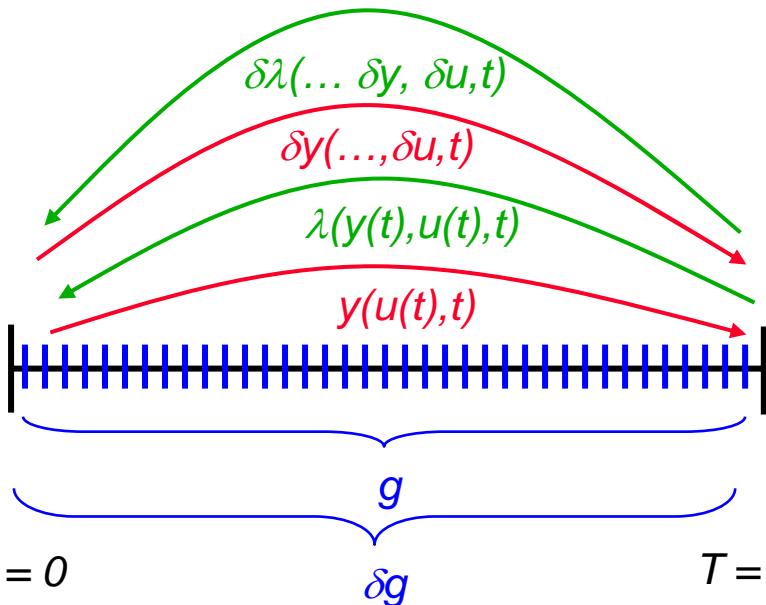
# Adjoint Solution Procedures

## Problem Formulation

$$\min \int_0^T \phi(y(t), u(t)) dt + \psi(y(T))$$

$$\text{s.t. } \frac{\partial y(t)}{\partial t} + F(y(t), u(t), t) = 0, \quad t \in (0, T) \\ y(0) = y_0$$

## Standard Integration Procedure



## First Order Optimality Conditions:

$$\frac{\partial y(t)}{\partial t} + F(y(t), u(t), t) = 0, \quad t \in (0, T) \\ y(0) = y_0$$

State

$$\frac{\partial \lambda(t)}{\partial t} - \left( \frac{\partial F}{\partial y} \right)^T \lambda(t) = \left( \frac{\partial \phi}{\partial y} \right)^T, \quad t \in (0, T) \\ \lambda(T) = - \left( \frac{\partial \psi}{\partial y} \right)^T$$

Adjoint

$$g(t) = \left( \frac{\partial \phi}{\partial u} \right)^T + \left( \frac{\partial F}{\partial u} \right)^T \lambda(t) = 0, \quad t \in (0, T)$$

Gradient

## Unconstrained Optimization

- Solve:  $g(u) = 0$ 
  - First-order methods, evaluate:  $u \rightarrow g(u)$ 
    - SD, Nonlinear CG, QN etc..
  - Second-order methods, CG Solve:  $\frac{\partial g}{\partial u} \delta u = -g$ 
    - Requires op:  $\frac{\partial g}{\partial u} \delta u \rightarrow \delta g$
    - Linearized state, adjoint, gradient**

Problem: Sequential in time!

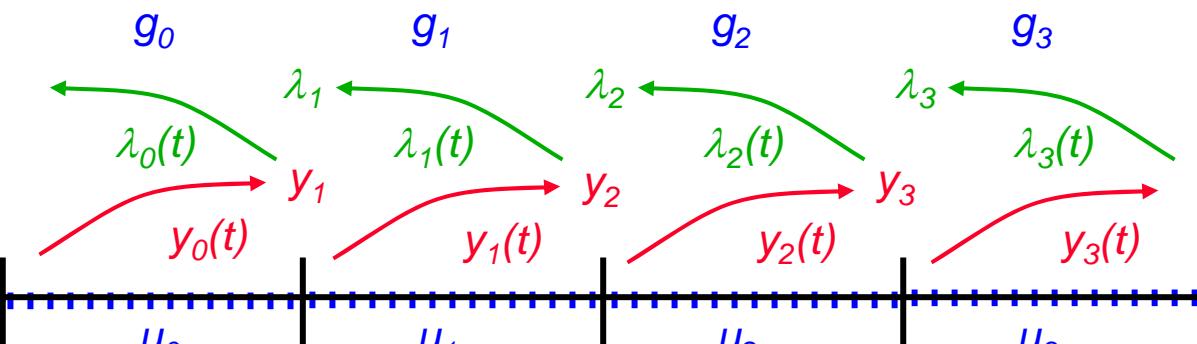
# Multiple Shooting : Parallel in time!

**Multiple Shooting KKT Conditions :  $G(x) = 0$**

- Example,  $N = 4$  parallel time domains

## Variables

$$x = \begin{Bmatrix} \vdots \\ y_{i+1} \\ u_i \\ \lambda_i \\ \vdots \end{Bmatrix} \quad \left. \begin{array}{l} \text{---} \\ i^{\text{th}} \text{ period} \end{array} \right\}$$



## Equations

Gradient:

$$g_0 = 0$$

Adjoint Cont:

$$\lambda_1 - \lambda_1(T_1) = 0 \quad \lambda_2 - \lambda_2(T_2) = 0 \quad \lambda_3 - \lambda_3(T_3) = 0$$

State Cont:

$$y_1 - y_0(T_1) = 0 \quad y_2 - y_1(T_2) = 0 \quad y_3 - y_2(T_3) = 0$$

$$\underbrace{G(x) = 0}_{t=0}$$

$$t = T_0$$

$$t = T_1$$

$$t = T_2$$

$$t = T_3$$

$$t = T_4$$

# Multiple Shooting Matrix-Free Algorithm

Solution of multiple-shooting KKT conditions using Newton's method for  $G(x) = 0$

Choose  $x_0$

**for**  $k = 0, 1, \dots$

If  $|G(x_k)| < \eta$  **Stop!**

Solve  $\frac{\partial G(x_k)}{\partial x} \delta x_k = -G(x_k)$

Choose  $\alpha$

$$x_{k+1} = x_k + \alpha \delta x_k$$

**end for**

Computation of residual  $G(x_k)$

- Computation of **state**, **adjoint** and **gradient** in parallel

Solution of the Newton system :  $\frac{\partial G}{\partial x} \delta x = -G$

- Using GMRES (using TSFCore version by H. Thornquist (9214))

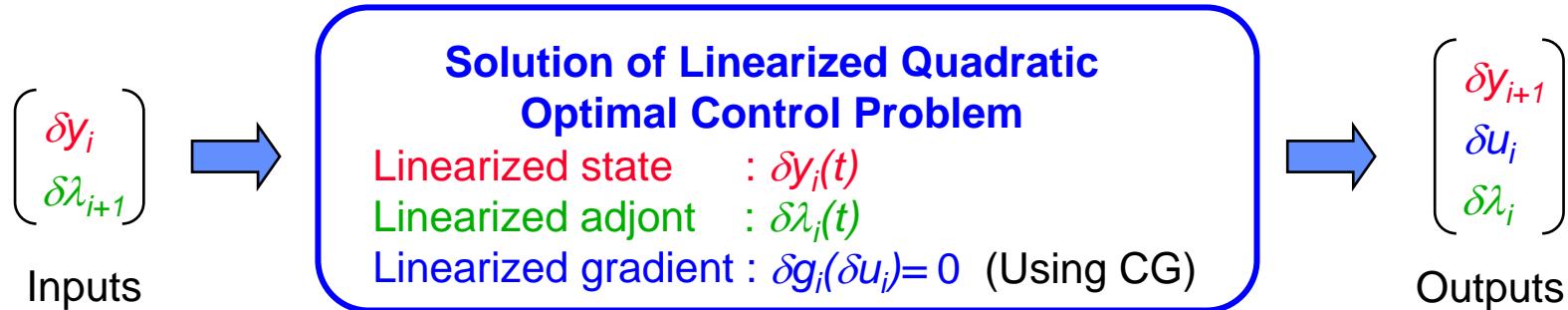
• Linear operator :  $\frac{\partial G}{\partial x} \delta x \Rightarrow \delta G$

- Computation of **linearized state**, **adjoint** and **gradient** in parallel

# Preconditioned Multiple Shooting

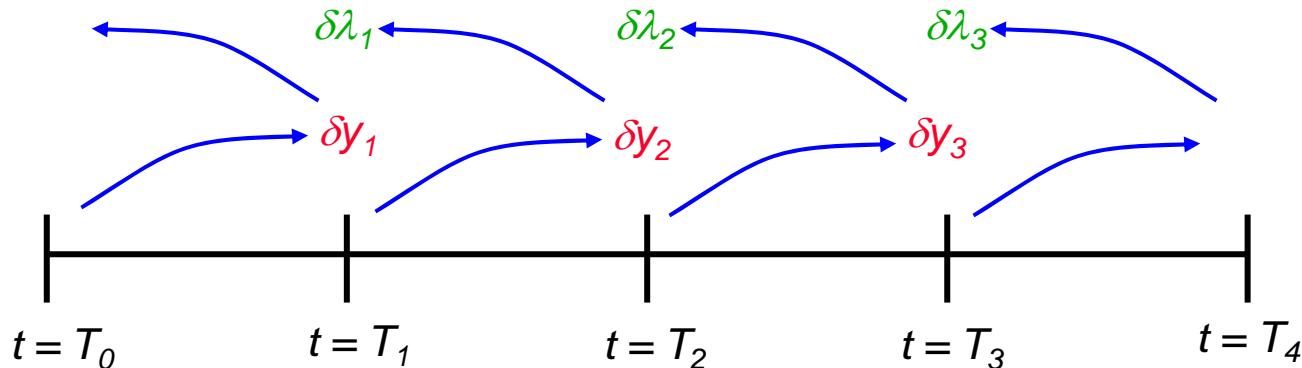
How do we build a preconditioner for  $\frac{\partial G}{\partial x}$  ?

- Basic preconditioner building block:



Block Gauss-Seidel Preconditioners :

- Backward GS:
- Forward GS:
- Fwd/Bwd GS



# Prec. Mult. Shoot. : Prelim. Results

## Linear Example : 1D Heat equation

- Simple quadratic objective
- Left and right boundary control
- $N_y = 40$  (linear) finite elements in space
- $N_t = 160$  total time steps (Crank-Nicolson)

## Observations

😊 Number of GMRES iterations constant with N for fwd/bwd GS preconditioner

😢 Current Gauss-Seidel preconditioners are not parallel

## Future Work:

- Find GS-like preconditioner that will:
  - Hold down outer GMRES iterations
  - Have small(er) serial component

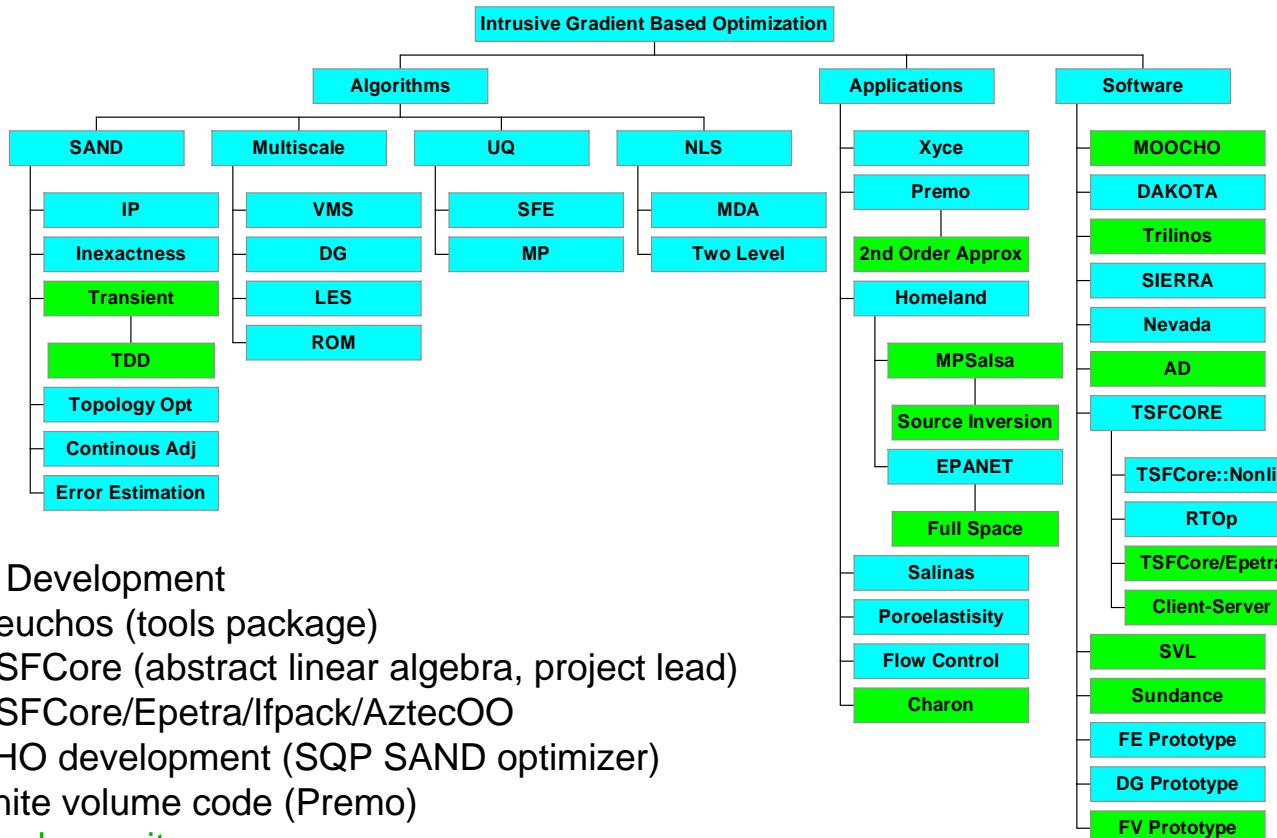
Number of GMRES iterations

| N  | No Prec. | Fwd GS | Bwd GS | Fwd/Bwd GS |
|----|----------|--------|--------|------------|
| 1  | 12       | 1      |        |            |
| 2  | 18       | 4      | 5      | 4          |
| 4  | 23       | 6      | 8      | 4          |
| 8  | 29       | 7      | 12     | 4          |
| 16 | 37       | 7      | 20     | 4          |
| 32 | 50       | 7      | 36     | 3          |

Total work

| N  | No Prec. | Fwd GS | Bwd GS | Fwd/Bwd GS |
|----|----------|--------|--------|------------|
| 1  | 7200     | 9440   |        |            |
| 2  | 10080    | 22080  | 28640  | 44400      |
| 4  | 12480    | 21280  | 28400  | 31320      |
| 8  | 15360    | 18480  | 28600  | 24360      |
| 16 | 19200    | 15030  | 34070  | 19980      |
| 32 | 25440    | 13575  | 51760  | 14960      |

# Summary



- Trilinos Development
  - Teuchos (tools package)
  - TSFCore (abstract linear algebra, project lead)
  - TSFCore/Epetra/Itpack/AztecOO
- MOOCHO development (SQP SAND optimizer)
- Euler finite volume code (Premo)
- Homeland security
  - Air chemical/biological source inversion
    - MOOCHO MPSalsa
    - Direct QP solver
- Transient optimization
  - Time domain decomposition (TDD)
  - TSFCore-based transient CG solver