SAND 2009-0463P

DAKOTA Training

Calibration

http://www.cs.sandia.gov/dakota

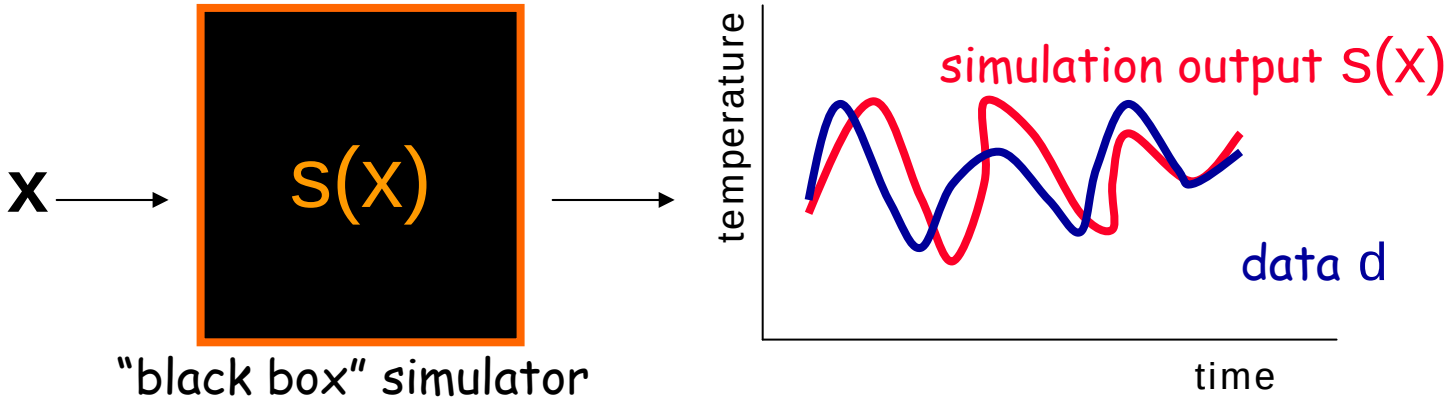
Learning goals:

- **Understand what calibration *is and is not* and why it is important**
- **Differentiate between optimization and least-squares calibration mathematical formulations**
- **Use various DAKOTA methods to perform model calibration to data**
- **How to run DAKOTA: specify (input deck) and run an analysis**

*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-AC04-94AL85000.

What is calibration?

$$f(x) = \sum_{i=1}^{n} (\underbrace{s_i(x)}_{\text{simulation output that depends on x}} - \underbrace{d_i}_{\text{given data}})^2$$



- **Calibration: Adjust model parameters x to maximize agreement with a set of experimental data.**
- **A.K.A. parameter estimation, parameter identification, systems identification, nonlinear least-squares, inverse problem.**

Why use calibration?

- **Ensure sufficient simulation code predictive capability**
- **Decrease the amount of info lost due to using a model instead of the "truth" (minimize discrepancy)**
- **Increased understanding of design space**
- **Find parameters yielding improved model robustness**
- **Calibration is not validation! Separate data should be used for calibration vs. validation.**

Nonlinear Least Squares

- **Calibration problems are often formulated to minimize the two norm of the error between the model and data: *minimize***

$$f(x) = \frac{1}{2} r(x)^T r(x) = \frac{1}{2}\left[s(x) - d\right]^T \left[s(x) - d\right] = \frac{1}{2}\sum_{i=1}^{n}\left(s_i(x) - d_i\right)^2$$

- **Example: `osborne1` analytic test problem, with *i = 1,…,33*:**

$$r_i(x) = \underbrace{\left(x_1 + x_2 e^{t_i x_4} + x_3 e^{t_i x_5}\right)}_{\text{model/simulation}} - \underbrace{d_i}_{\text{data}}; \quad t_i = -10(i-1)$$

- **A specialized class of optimization algorithms exploit this structure for efficient solution without second derivative information (more coming soon)**

DAKOTA Input: osborne1

```
method,
        nl2sol
        max_iterations =
        convergence_tolerance =
model,
        single
variables,
        continuous_design = 5
        cdv_initial_point        .5      1.5     -1      .01     .02
        cdv_lower_bounds  .3     0.7     -2      .001    .001
        cdv_upper_bounds  .6     1.8     0       .2      .23
        cdv_descriptor           'x1'    'x2'    'x3'    'x4'    'x5'
interface,
        system
          analysis_driver = './osborne1'
responses
        num_least_square_terms = 33
        analytic_gradients
        no_hessians
```

Method independent options

Run Dakota on osborne1

```
> cd nlls
> dakota –i osborne1.in

<<<<< Function evaluation summary: 27 total (26 new, 1 duplicate)
<<<<< Best parameters          =
                    3.7541004764e-01 cdv_1
                    1.9358463401e+00 cdv_2
                   -1.4646865611e+00 cdv_3
                    1.2867533504e-02 cdv_4
                    2.2122702031e-02 cdv_5
<<<<< Best residual norm = 7.3924926090e-03; 0.5 * norm
<<<<< Best residual terms =
                    2.5698266188e-03
                   -4.4759880011e-03
```

Using a Separate Data Source (d_i): osborne1b(b)

```
method nl2sol
  output silent
  convergence_tolerance = -1.

variables,
  continuous_design = 5
  initial_point .5 1.5 .01 -1 .02
  lower_bounds .2 1.0 .005 -1.5 .01
  upper_bounds .6 2.0 .012 1.5 .05

interface,
  system
  analysis_driver = ’./osborne1b’

responses,
  num_least_squares_terms = 33
  least_squares_data_file ’osborne1_y’
  analytic_gradients    # For finite differences, comment this
  # numerical_gradients # and uncomment this line.
  no_hessians
```

Least-squares Structure

- **When minimizing *f(x)* with gradient-based methods, can take advantage of the form of its derivatives:**

$$f(x) = \frac{1}{2} r(x)^T r(x) = \frac{1}{2}\left[s(x) - d\right]^T \left[s(x) - d\right]$$

$$\nabla f(x) = J(x)^T r(x); \quad J_{ij} = \frac{\partial r_i}{\partial x_j}$$

$$\nabla^2 f(x) = J^T J + \sum_{i=1}^{n} r_i(x) \nabla^2 r_i(x)$$

Algorithms vary in how they approximate this Hessian.

Hessian Approximations

$$\nabla^2 f(x) = J^T J + \sum_{i=1}^{n} r_i(x) \nabla^2 r_i(x)$$

Gauss-Newton: $J(x)^T J(x)$

Levenberg-Marquardt: $J(x)^T J(x) + \mu I$, with $\mu \geq 0$

$NL2SOL: J(x)^T J(x) + S,$

with $S = 0$ or $S =$ Quasi-Newton approximation to $\sum_{i=1}^{n} f_i(x) \nabla^2 f_i(x)$

DAKOTA Method Selection

Calibration Method	Step Control	Unconstrained	Bounds	Linear/ Nonlinear
`nl2sol`	trust region	X	X	
`nlssol`	line search	X	X	X
`optpp_g_newton`	trust region or line search	X	X	X

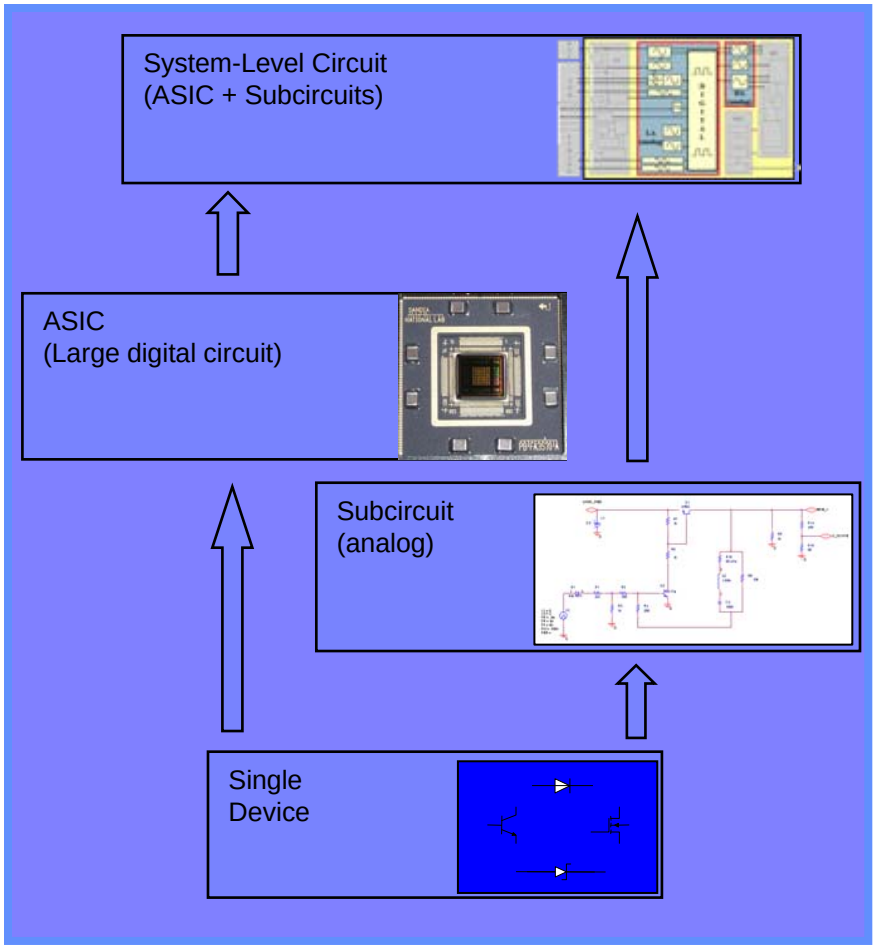
NL2SOL can handle highly nonlinear problems.

Confidence Intervals on Params

```
dakota lls.in

...
<<<<< Best parameters =
3.9975104529e-01 cdv_1
7.8306751279e-01 cdv_2
-1.1317783545e-01 cdv_3
...

Confidence Interval for cdv_1
is [ -5.2378467908e-01, 1.3232867697e+00 ]

Confidence Interval for cdv_2
is [ -9.4840422538e-01, 2.5145392510e+00 ]

Confidence Interval for cdv_3
is [ -1.5865346409e+00, 1.3601789700e+00 ]
```

Example: Electrical Application

- Radiation-aware electrical models
- Predict responses of electrical devices in hostile environments
- Building blocks of a large electrical system being examined hierarchically
- Access to code and model developers



Hierarchical Electrical Model

Least Squares Objective Function

$$\min \sum_{i=1}^{N} w_i(T_i) \sum_{t=1}^{T_i} \left(s_i(t;x) - e_i(t)\right)^2$$

N = number of tests

T_i = (relevant) number of experimental values for test i

$w_i(T_i)$ = weighting factor (depends on number of experimental points)

$S_i(t;x)$ = simulated value, calculated with parameters x, corresponding to experimental point t for experiment i

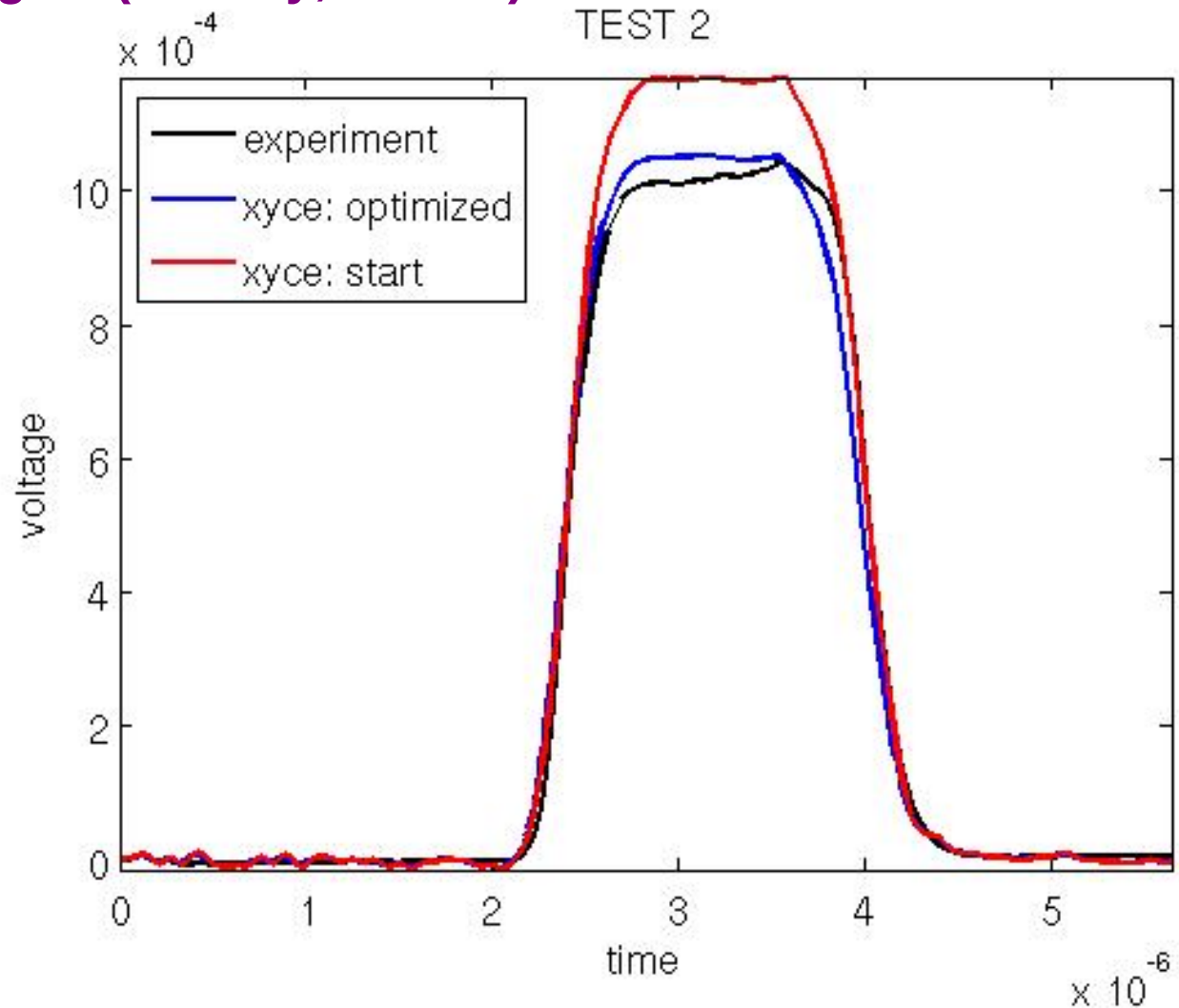
$e_i(t)$ = test value of point t in test i

Parameters

- Device model has ~30 parameters
- Parameters ranked by modeler
  - How much does the model rely on the parameter being chosen correctly?
  - How uncertain are we about the current value being used?
- Selected
  - 8 parameters for calibration
  - Either physical parameters or covering “missing” physics
- Modeler provided bounds and starting points by “hand tuning” process

Sample Signal Calibration

- **Calibration of 8 circuit parameters to match experimental signal (G. Gray, M. M-C)**



EXTRA SLIDES

Examples in nlls.tgz

```
gzip -dc nlls.tgz | tar xf -
```

gives directory nlls containing:

lls	**analysis driver compiled from lls.c**
lls.c	**source for lls**
lls.in	**DAKOTA input file using lls**
osborne1	**python script as analysis driver**
osborne1[ab]	**variations on osborne1 script**
osborne1*.in	**input files using osborne1***
osborne1	**y right-hand side file (data) for osborne1b and osborne1bb**

nl2sol method dependent options

```
{nl2sol}                                  \
[function_precision = <REAL>]             \
[absolute_conv_tol = <REAL>]              \
[x_conv_tol = <REAL>]                     \
[singular_conv_tol = <REAL>]              \
[singular_radius = <REAL>]                \
[false_conv_tol = <REAL>]                 \
[initial_trust_radius = <REAL>]           \
[covariance = <INTEGER>]                  \
[regression_diagnostics]                  \
```

Reference Manual Chapter 2

Circuit Simulator

- Written at Sandia to support electrical (circuit) design simulation
- Started with Berkeley SPICE 3f5
  - Mostly Algebraic/Differential Equations with Behavioral Model Options
  - Physics based models (instead of empirical based models)
  - Improvement over industry standard PSPICE capability
- Massively parallel code that allows simulation of large-scale complex system circuit model
  - Investigate circuit interactions
  - Simulate large digital components

Hutchinson, Keiter, Hoekstra, Rankin, Waters, Russo, Wix, Ballard, …