ModelEvaluator

Scalable, Extendable Interface Between Embedded Nonlinear Analysis Algorithms and Applications

Roscoe A. Bartlett
Department of Optimization & Uncertainty Estimation

Sandia National Laboratories

April 3rd, 2008

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

Overview of Nonlinear Solver and Analysis Capabilities

Sandia National Laboratories

Trilinos Nonlinear Solver and Analysis Packages

Trilinos Packages

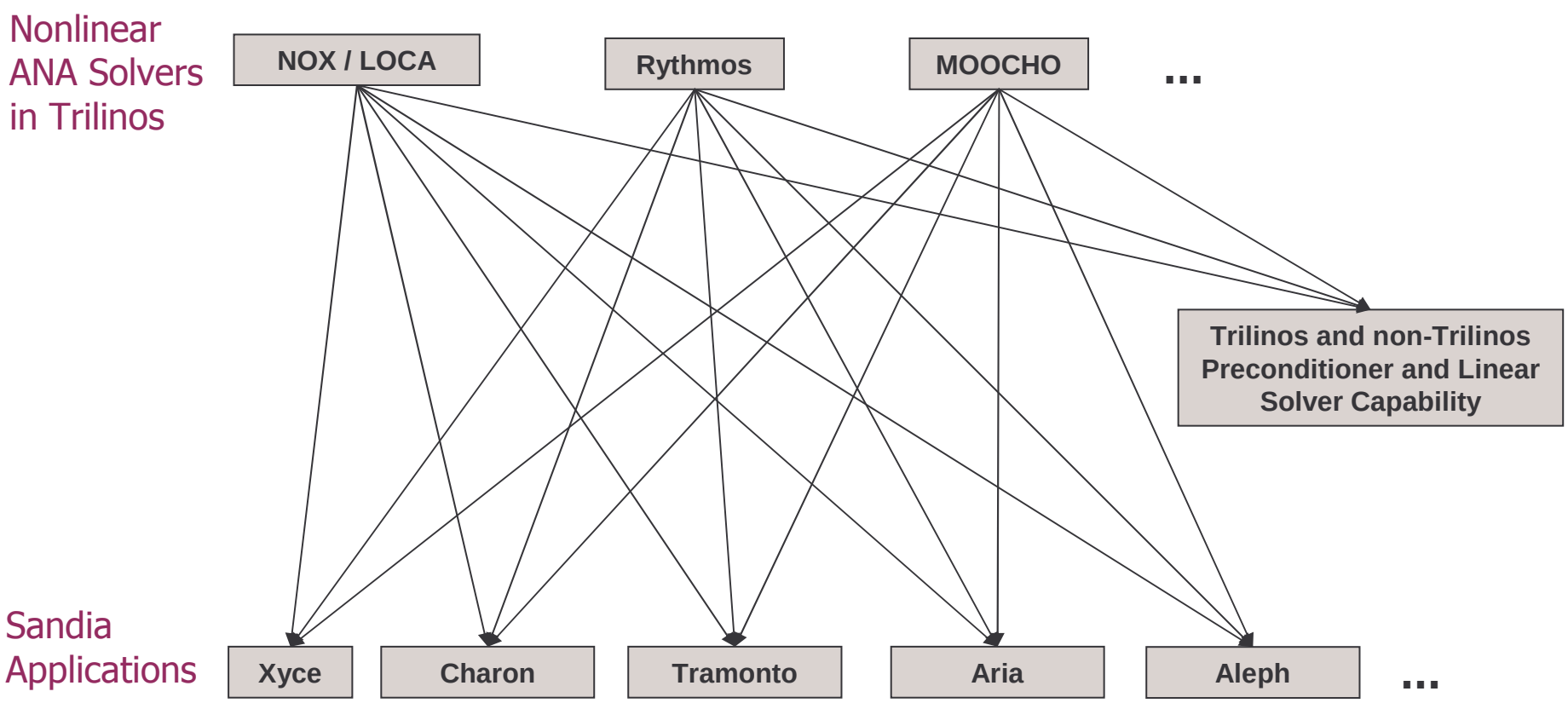
- Nonlinear Problems: Given nonlinear operator $f(x,p) \in \mathbf{R}^{n+m} \rightarrow \mathbf{R}^n$
  - Nonlinear equations: Solve $f(x) = 0$ for $x \in \mathbf{R}^n$ — NOX
  - Stability analysis: For $f(x,p) = 0$ find space $p \in \mathcal{P}$ such that $\frac{\partial f}{\partial x}$ is singular — LOCA
- Transient Nonlinear Problems:
  - DAEs/ODEs: Solve $f(\dot{x}(t), x(t), t) = 0, t \in [0,T],\ x(0) = x_0,\ \dot{x}(0) = x'_0$ for $x(t) \in \mathbf{R}^n, t \in [0,T]$ — Rythmos
  - ODE/DAE Sensitivities …
- Optimization Problems:
  - Unconstrained: Find $p \in \mathbf{R}^m$ that minimizes $g(p)$
  - Constrained: Find $x \in \mathbf{R}^n$ and $p \in \mathbf{R}^m$ that: minimizes $g(x,p)$ such that $f(x,p) = 0$ — MOOCHO

Sandia National Laboratories

ModelEvaluator Overview

Sandia National Laboratories

Nonlinear Algorithms and Applications : Everyone for Themselves?



Key Point

- BAD

Sandia National Laboratories

Overview of Nonlinear Model Evaluator Interface

Motivation: An interface for nonlinear problems is needed that will support a variety of different types of problems

- Nonlinear equations (and senstitivities)
- Stability analysis and continuation
- Explicit ODEs (and sensitivities)
- DAEs and implicit ODEs (and sensitivities)
- Unconstrained optimization
- Constrained optimization
- Uncertainty quantification
- …

as well as different combinations of problem types such as:

- Uncertainty in transient simulations
- Stability of an optimum under uncertainty of a transient problem

Key Point

The number of combinations of different problem types is large and trying to statically type all of the combinations is not realistic

Approach: Develop a single, scalable interface to address all of these problems

- (Some) Input arguments:
  - State and differential state: $x \in \mathcal{X}$ and $\dot{x} = \frac{dx}{dt} \in \mathcal{X}$
  - Parameter sub-vectors: $p_l \in \mathcal{P}_l$ for $l = 0 \ldots N_p - 1$
  - Time (differential): $t \in \mathbf{R}$
- (Some) Output functions:
  - State function: $(\dot{x}, x, \{p_l\}, t) \Rightarrow f \in \mathcal{F}$
  - Auxiliary response functions: $(\dot{x}, x, \{p_l\}, t) \Rightarrow g_j \in \mathcal{G}_j$, for $j = 0 \ldots N_g - 1$
  - State/state derivative operator (LinearOpWithSolve): $(\dot{x}, x, \{p_l\}, t) \Rightarrow W = \alpha \frac{\partial f}{\partial \dot{x}} + \beta \frac{\partial f}{\partial x}$

Key Point

All inputs and outputs are optional and the model evaluator object itself decides which ones are accepted.

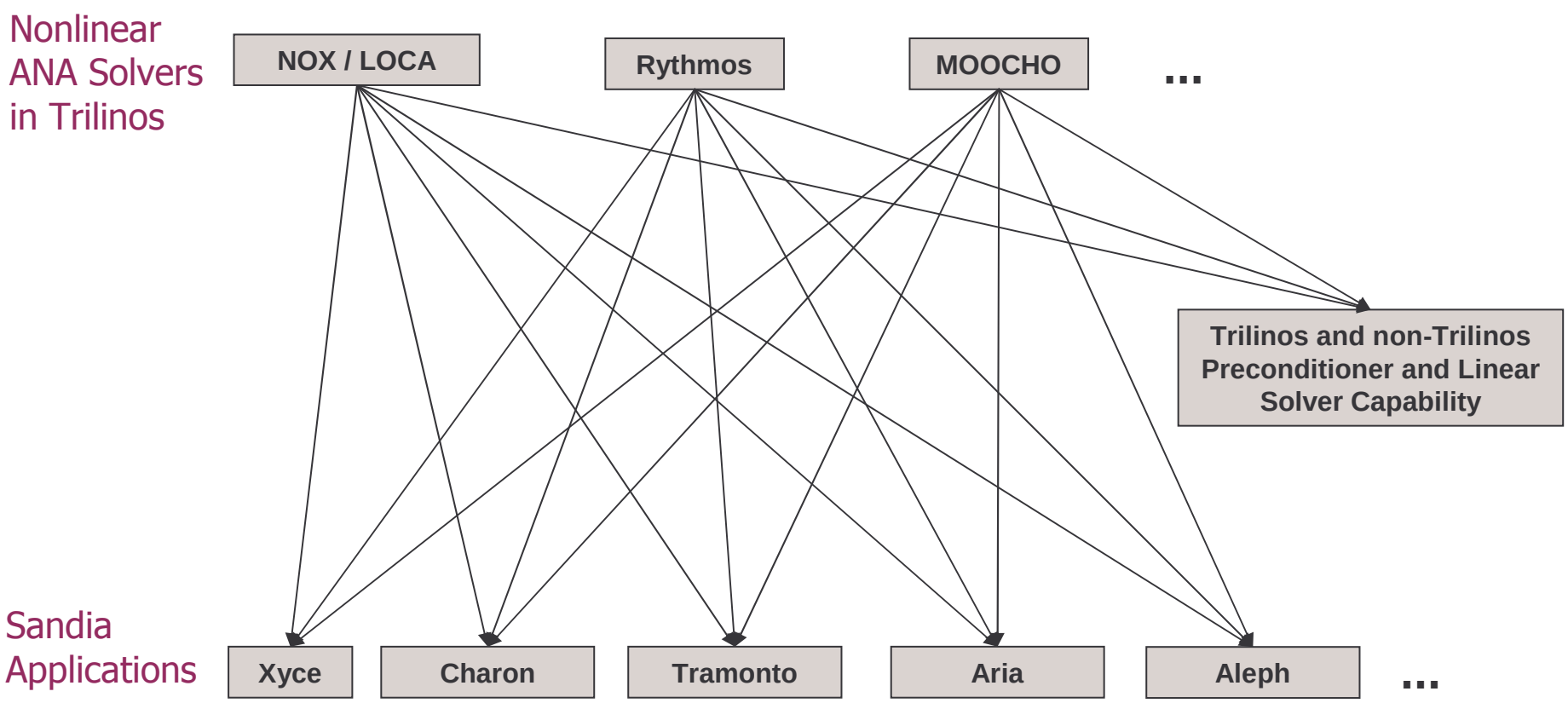
Sandia National Laboratories

Some Nonlinear Problems Supported by the ModelEvaluator

Nonlinear equations:	Solve $f(x) = 0$ for $x \in \mathbf{R}^n$
Stability analysis:	For $f(x,p) = 0$ find space $p \in \mathcal{P}$ such that $\frac{\partial f}{\partial x}$ is singular
Explicit ODEs:	Solve $\dot{x} = f(x,t) = 0, t \in [0,T]$, $x(0) = x_0$, for $x(t) \in \mathbf{R}^n, t \in [0,T]$
DAEs/Implicit ODEs:	Solve $f(\dot{x}(t), x(t), t) = 0, t \in [0,T]$, $x(0) = x_0$, $\dot{x}(0) = x'_0$ for $x(t) \in \mathbf{R}^n, t \in [0,T]$
Explicit ODE Forward Sensitivities:	Find $\frac{\partial x}{\partial p}(t)$ such that: $\dot{x} = f(x,p,t) = 0, t \in [0,T]$, $x(0) = x_0$, for $x(t) \in \mathbf{R}^n, t \in [0,T]$
DAE/Implicit ODE Forward Sensitivities:	Find $\frac{\partial x}{\partial p}(t)$ such that: $f(\dot{x}(t), x(t), p, t) = 0, t \in [0,T]$, $x(0) = x_0$, $\dot{x}(0) = x'_0$, for $x(t) \in \mathbf{R}^n, t \in [0,T]$
Unconstrained Optimization:	Find $p \in \mathbf{R}^m$ that minimizes $g(p)$
Constrained Optimization:	Find $x \in \mathbf{R}^n$ and $p \in \mathbf{R}^m$ that: minimizes $g(x,p)$ such that $f(x,p) = 0$
ODE Constrained Optimization:	Find $x(t) \in \mathbf{R}^n$ in $t \in [0,T]$ and $p \in \mathbf{R}^m$ that: minimizes $\int_0^T g(x(t),p)$ such that $\dot{x} = f(x(t),p,t) = 0$, on $t \in [0,T]$ where $x(0) = x_0$

Sandia National Laboratories

Nonlinear Algorithms and Applications : Everyone for Themselves?



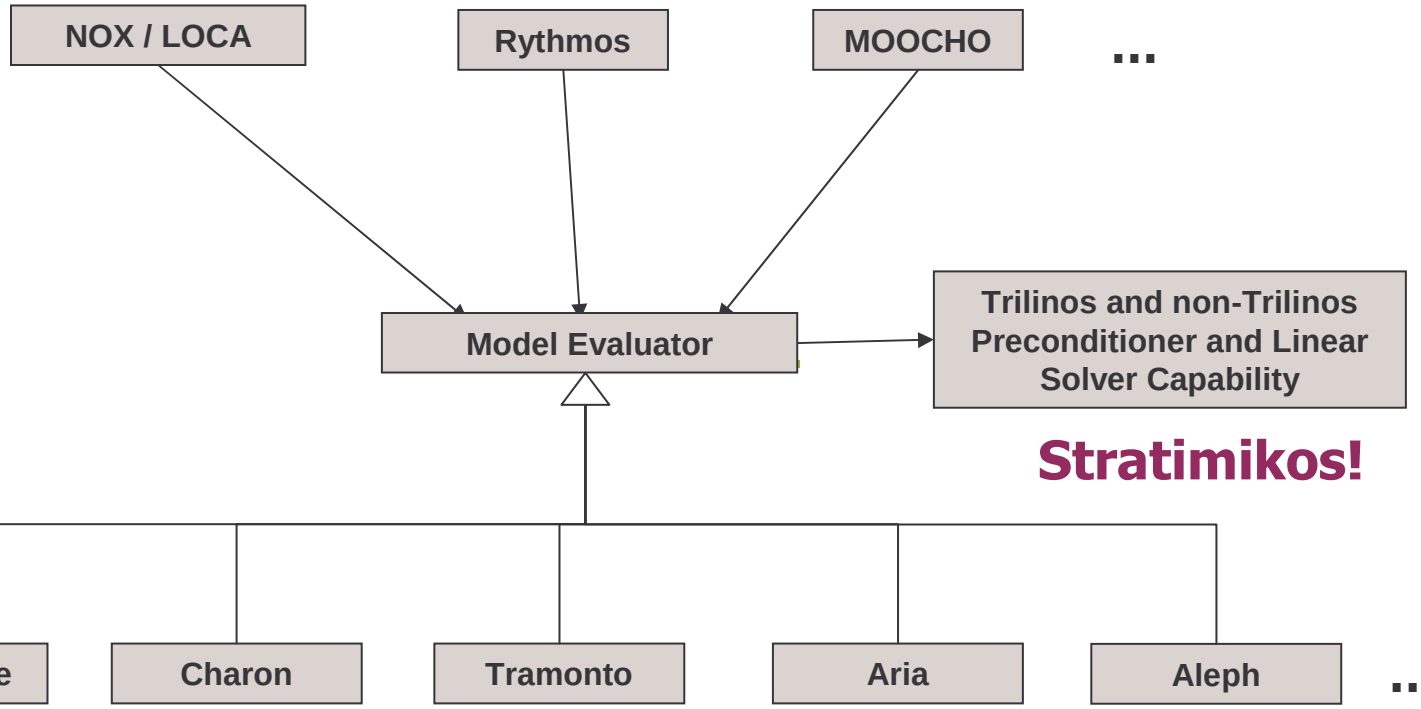
Key Point

- BAD

Sandia National Laboratories

Nonlinear Algorithms and Applications : Thyra & Model Evaluator!

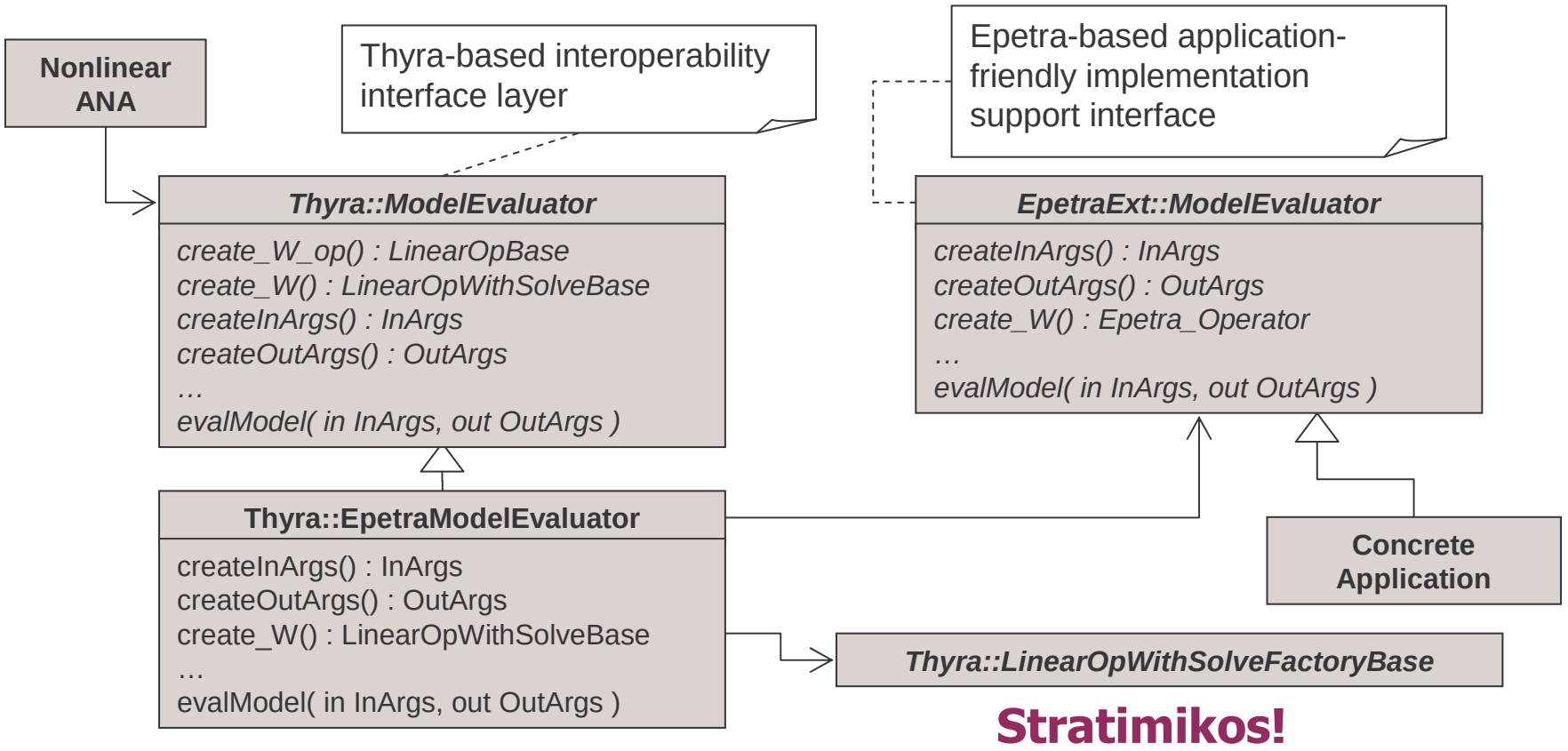
Nonlinear ANA Solvers in Trilinos



Key Points

- Provide single interface from nonlinear ANAs to applications
- Provide single interface for applications to implement to access nonlinear ANAs
- Provides shared, uniform access to linear solver capabilities
- Once an application implements support for one ANA, support for other ANAs can quickly follow

Model Evaluator : Thyra and EpetraExt Versions



- Thyra::ModelEvaluator and EpetraExt::ModelEvaluator are near mirror copies of each other.
- Thyra::EpetraModelEvaluator is fully general adapter class that can use any linear solver through a Thyra::LinearOpWithSolveFactoryBase object it is configured with
- Stateless model that allows for efficient multiple shared calculations (e.g. automatic differentiation)
- Adding input and output arguments involves
  - Modifying only the Thyra::ModelEvaluator, EpetraExt::ModelEvaluator, Thyra::EpetraModelEvaluator, and a few other auxiliary classes and tools => O(1) overhead to add new arguments!
  - Only recompilation of Nonlinear ANA and Concrete Application code

Simple Example EpetraExt::ModelEvaluator Subclass

Sandia National Laboratories

Example EpetraExt::ModelEvaluator Application Implementation

```
/** \brief Simple example ModelEvaluator subclass for a 2x2 set of
 * parameterized nonlinear equations.
 *
 * The equations modeled are:
 \verbatim

    f[0] =       x[0]      + x[1]*x[1] - p[0];
    f[1] = d * ( x[0]*x[0] - x[1]      - p[1] );

 \endverbatim
 */
class EpetraModelEval2DSim : public EpetraExt::ModelEvaluator {
public:
  EpetraModelEval2DSim(...);
  /** \name Overridden from EpetraExt::ModelEvaluator . */
  //@{
  RCP<const Epetra_Map>      get_x_map() const;
  RCP<const Epetra_Map>      get_f_map() const;
  RCP<const Epetra_Vector>   get_x_init() const;
  RCP<Epetra_Operator>       create_W() const;
  InArgs      createInArgs() const;
  OutArgs     createOutArgs() const;
  void evalModel( const InArgs& inArgs, const OutArgs& outArgs ) const;
  //@}

private:
  ...
};
```

Complete nonlinear equations example in:

Trilinos/packages/epetraext/example/model_evaluator/2dsim/

Sandia National Laboratories

Example EpetraExt::ModelEvaluator Application Implementation

```
EpetraExt::ModelEvaluator::InArgs EpetraModelEval2DSim::createInArgs() const
{
  InArgsSetup inArgs;
  inArgs.setModelEvalDescription(this->description());
  inArgs.setSupports(IN_ARG_x,true);
  return inArgs;
}

EpetraExt::ModelEvaluator::OutArgs EpetraModelEval2DSim::createOutArgs() const
{
  OutArgsSetup outArgs;
  outArgs.setModelEvalDescription(this->description());
  outArgs.setSupports(OUT_ARG_f,true);
  outArgs.setSupports(OUT_ARG_W,true);
  outArgs.set_W_properties(
    DerivativeProperties(DERIV_LINEARITY_NONCONST,DERIV_RANK_FULL,true)
    );
  return outArgs;
}

void EpetraModelEval2DSim::evalModel( const InArgs& inArgs, const OutArgs& outArgs ) const
{
  const Epetra_Vector &x = *inArgs.get_x();
  RCP<Epetra_Vector> f_out = outArgs.get_f();
  RCP<Epetra_Operator> W_out = outArgs.get_W();
  if (!is_null(f_out)) {
    ...
  }
  if (!is_null(W_out)) {
    ...
  }
}
```

Key Point

From looking at example code, there is not even a hint that other input and output arguments exist!

Sandia National Laboratories

Vertical Integration Milestone:

ModelEvaluator in Action!

Sandia National Laboratories

Summary of Trilinos Vertical Integration Project (Milestone)

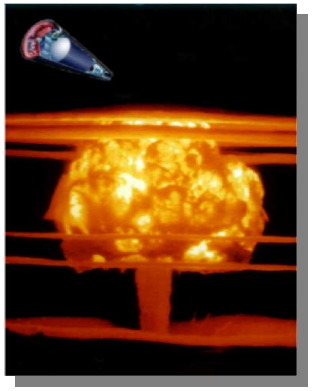
- **Goal:** Vertically integrate Trilinos solver algorithms in Trilinos to build new predictive embedded analysis capabilities
  - **Impact:** Vertically integrated 10+ Trilinos algorithm packages
- **Goal:** Demonstrate on relevant production applications
  - **Impact:** Solved steady-state parameter estimation problems and transient sensitivities on semiconductor devices in Charon
  - **Impact:** Solved Eigen problems on MHD problem in Charon
- **Added Goal:** Explore refined models of collaboration between production application developers and algorithm researchers.
  - **Impact:** Closer collaboration between application and algorithm developers yielding better algo and app R&D

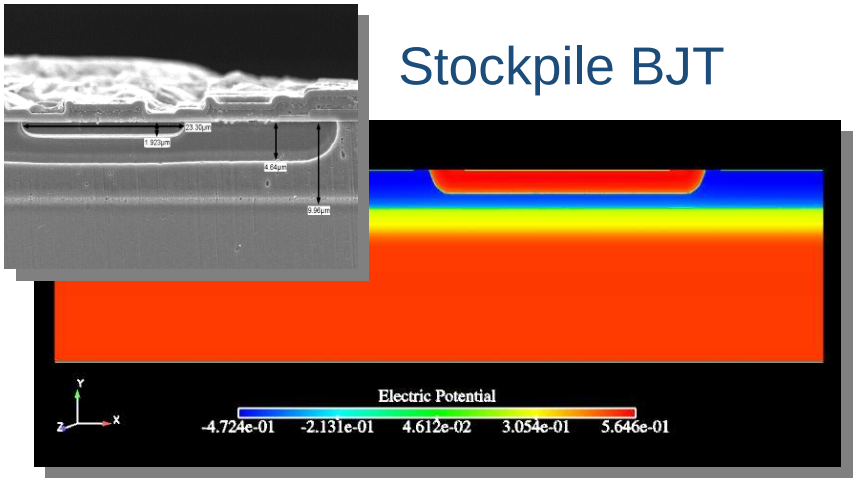
Bartlett, Roscoe, Scott Collis, Todd Coffey, David Day, Mike Heroux, Rob Hoekstra, Russell Hooper, Roger Pawlowski, Eric Phipps, Denis Ridzal, Andy Salinger, Heidi Thornquist, and Jim Willenbring. *ASC Vertical Integration Milestone*. SAND2007-5839, Sandia National Laboratories, 2007 [http://www.cs.sandia.gov/~rabartl/publications.html]

Sandia National Laboratories

QASPR

Qualification of electronic devices in hostile environments

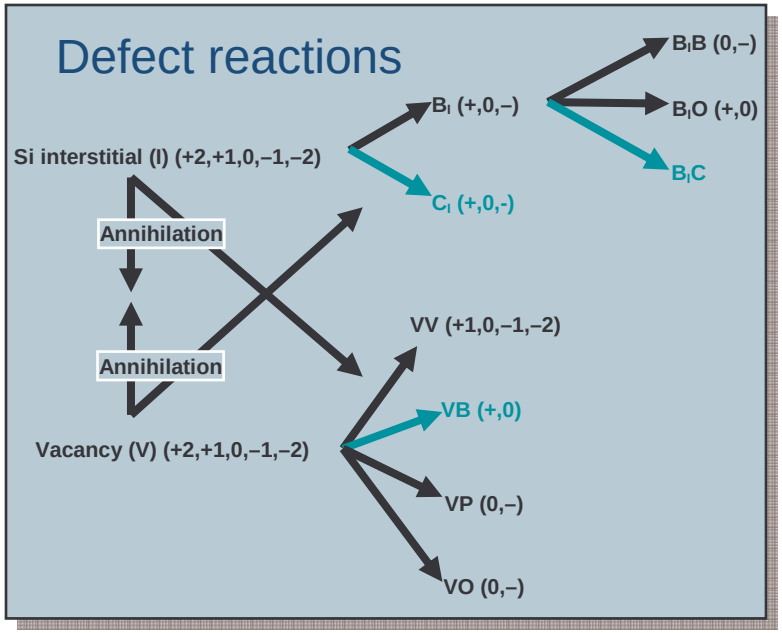


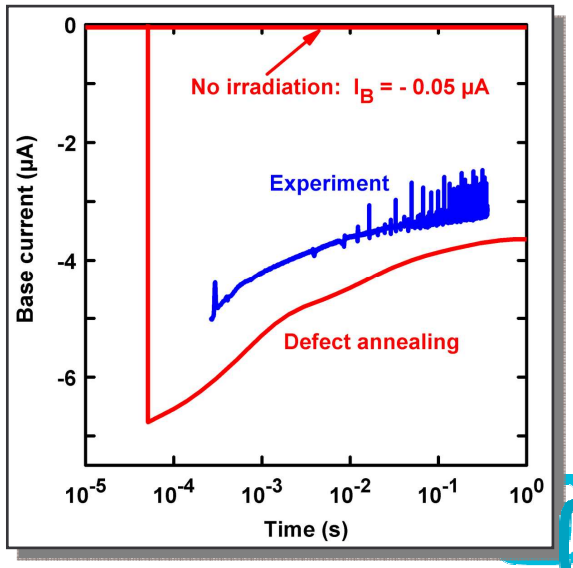


Stockpile BJT

CHARON

PDE semiconductor device simulation

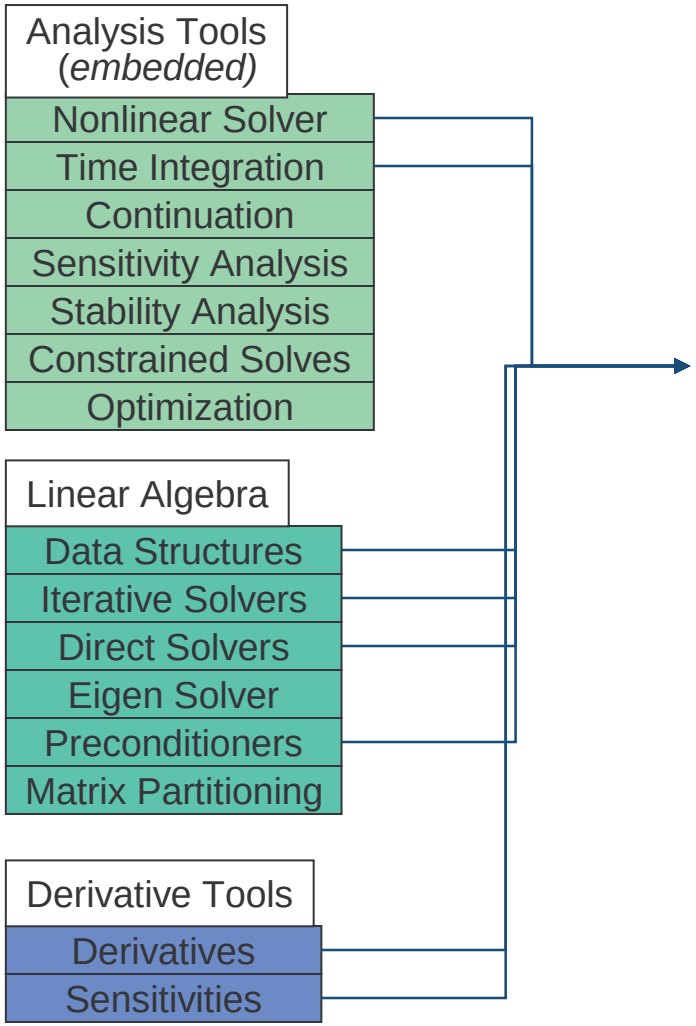


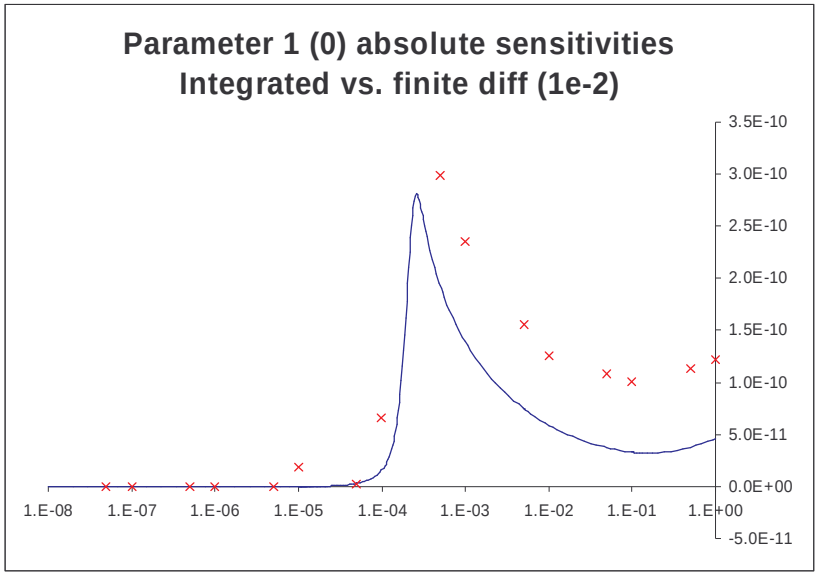


Sandia National Laboratories

Vertical Integrations of Trilinos Capabilities: Example 1

Trilinos Capabilities



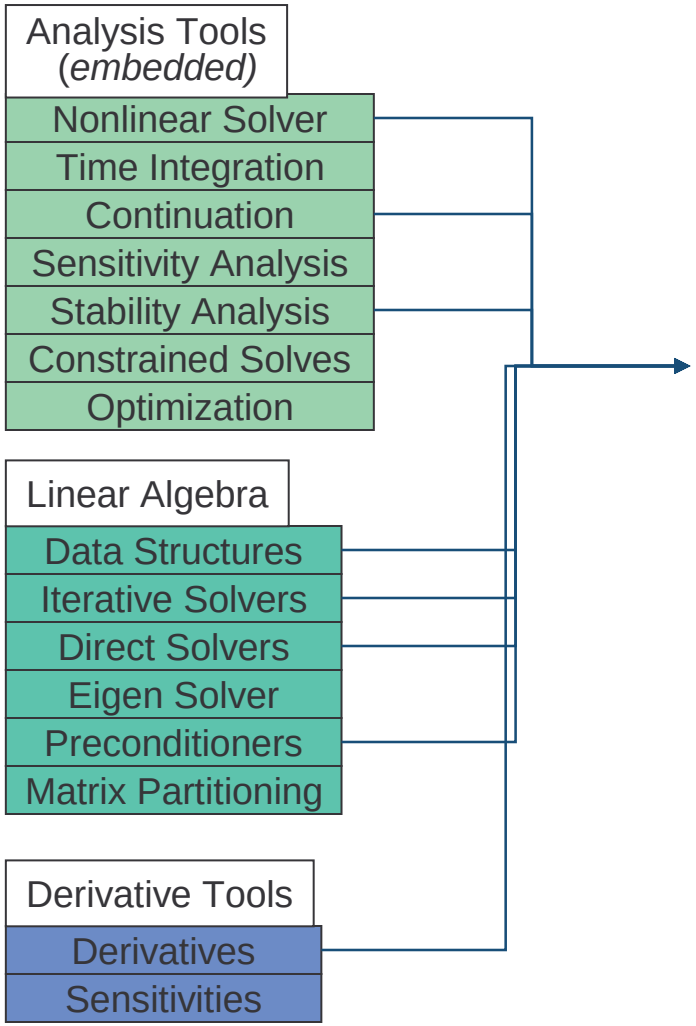


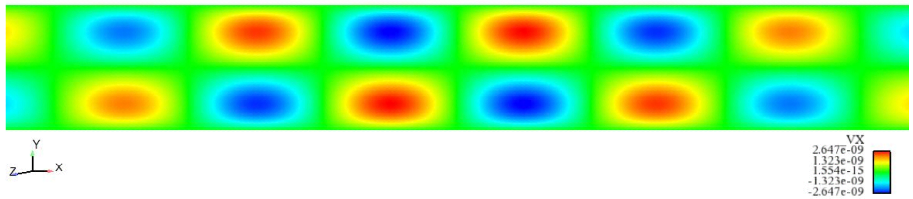
Transient sensitivity analysis of a 2n2222 BJT in Charon *w/AD+Rythmos: 14x faster than FD*

Sandia National Laboratories

Vertical Integrations of Trilinos Capabilities: Example 2

Trilinos Capabilities



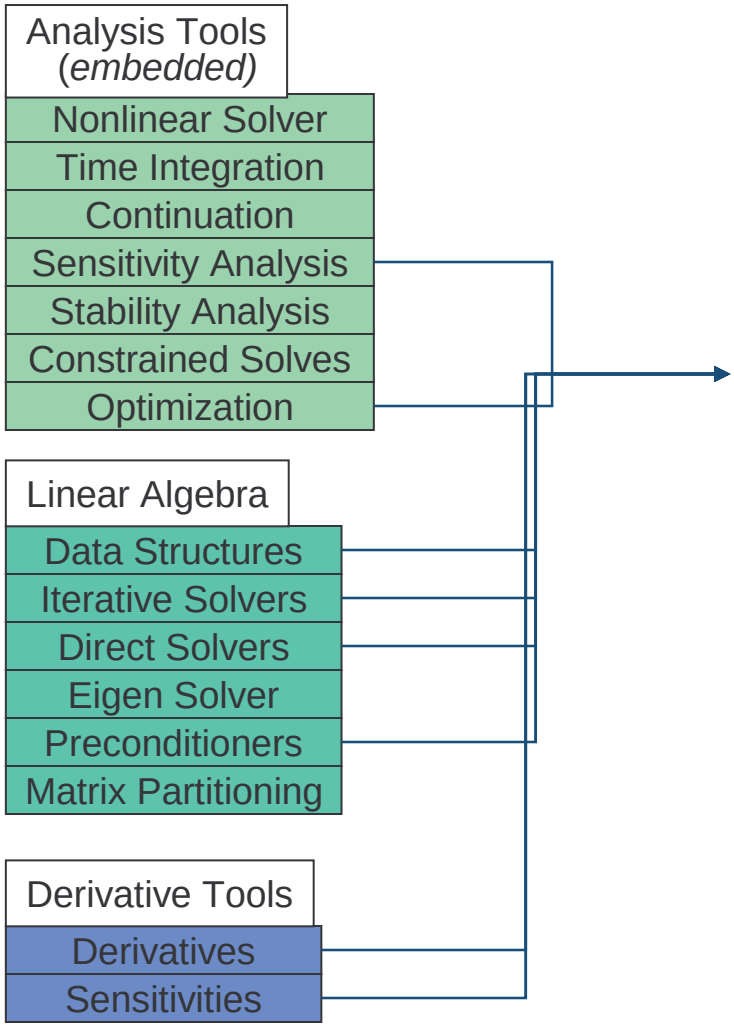


Destabilizing eigen-vector for heated fluid in magnetic field

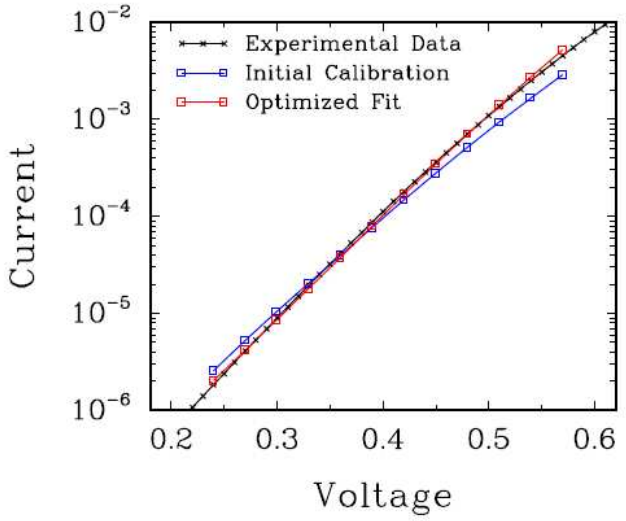
Sandia National Laboratories

Vertical Integrations of Trilinos Capabilities: Example 3

Trilinos Capabilities



$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}||g(x,p) - g^*||_2^2 + \tfrac{1}{2}\beta||p||_2^2 \\ \text{subject to} \quad & f(x,p) = 0 \end{aligned}$$



Steady-Sate Parameter Estimation Problem using 2n2222 BJT in Charon *MOOCHO + AD*

Sandia National Laboratories

Derivative Support and Sensitivity Analysis

Sandia National Laboratories

Model Derivatives

First Derivatives

- State function state sensitivities:

$$W = \alpha \frac{\partial f}{\partial \dot{x}} + \beta \frac{\partial f}{\partial x}$$

[LinearOpWithSolveBase or LinearOpBase]

- State function parameter sensitivities:

$$\frac{\partial f}{\partial p_l}, \text{ for } l = 0 \ldots N_p - 1$$

[LinearOpBase or MultiVectorBase]

- Auxiliary function state sensitivities:

$$\frac{\partial g_j}{\partial x}, \text{ for } j = 0 \ldots N_g - 1$$

[LinearOpBase or MultiVectorBase^2]

- Auxiliary function parameter sensitivities:

$$\frac{\partial g_j}{\partial p_l}, \text{ for } j = 0 \ldots N_g - 1, \; l = 0 \ldots N_p - 1$$

[LinearOpBase or MultiVectorBase^2]

Use Cases:

- Steady-state and transient sensitivity computations
- Optimization
- Multi-physics coupling
- …

Key Point

Derivative class used to store one of these three forms of a derivative

Sandia National Laboratories

Forward/Direct and Adjoint Sensitivities

Steady-state constrained response:

$$g(x,p) \text{ s.t. } f(x,p) = 0$$

Reduced response function:

$$\hat{g}(p) = g(x(p),p)$$

Reduced Sensitivities:

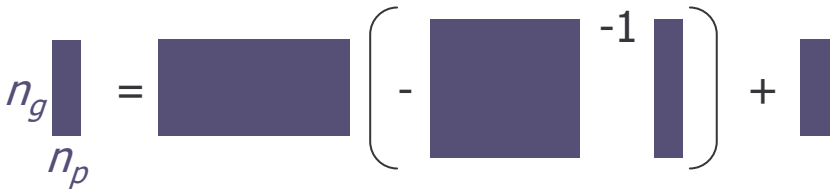
$$\frac{\partial \hat{g}}{\partial p} = \frac{\partial g}{\partial x}\frac{\partial x}{\partial p} + \frac{\partial g}{\partial p} \quad \text{where:} \quad \frac{\partial x}{\partial p} = -\frac{\partial f}{\partial x}^{-1}\frac{\partial f}{\partial p}$$

Key Point

The form of the derivatives you need depends on whether you are doing direct or adjoint sensitivities

Forward/Direct Sensitivities (n_g large, n_p small)

$$\frac{\partial \hat{g}}{\partial p} = \frac{\partial g}{\partial x}\left(-\frac{\partial f}{\partial x}^{-1}\frac{\partial f}{\partial p}\right) + \frac{\partial g}{\partial p}$$

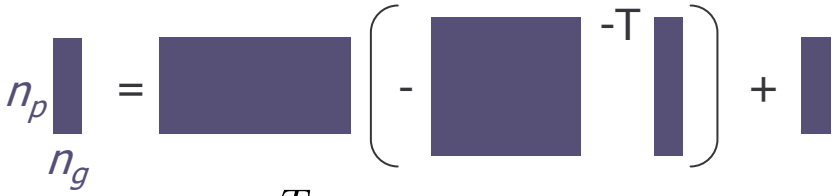


$\frac{\partial \hat{g}}{\partial p}$ MV | $\frac{\partial g}{\partial x}$ LO | $\frac{\partial f}{\partial x}$ $\frac{\partial f}{\partial p}$ LOWS MV | $\frac{\partial g}{\partial p}$ MV

MV

Adjoint Sensitivities (n_g small, n_p large)

$$\frac{\partial \hat{g}^T}{\partial p} = \frac{\partial f^T}{\partial p}\left(-\frac{\partial f}{\partial x}^{-T}\frac{\partial g}{\partial x}^T\right) + \frac{\partial g^T}{\partial p}$$



$\frac{\partial \hat{g}^T}{\partial p}$ MV | $\frac{\partial f^T}{\partial p}$ LO | $\frac{\partial f}{\partial x}$ $\frac{\partial g}{\partial x}^T$ LOWS MV | $\frac{\partial g^T}{\partial p}$ MV

MV

Transient Sensitivities: State Equations with Responses

Fully Implicit ODE/DAE State Equations

$$\begin{aligned} f\left(\dot{x}(t), x(t), p, t\right) &= 0,\ t \in \left[t_0, t_f\right] \\ x(0) &= x_0(p) \\ \dot{x}(0) &= \dot{x}_0(p) \end{aligned}$$

Composite Response Function

$$d(x,p) = \int_{t_0}^{t_f} g(\dot{x}(t), x(t), p, t)dt + h(\dot{x}(t_f), x(t_f), p)$$

Reduced Composite Response Function

$$\hat{d}(p,v) = \int_{t_0}^{t_f} \hat{g}(p,t)dt + \hat{h}(p)$$

where:

$$\hat{g}(p,t) = g(\dot{x}(p,t), x(p,t), p, t))$$

$$\hat{h}(p) = h(\dot{x}(p,t_f), x(p,t_f), p))$$

http://www.cs.sandia.gov/~rabartl/TransientSensitivitiesDerivation2007.pdf

Sandia National Laboratories

Transient Sensitivities: Forward Sensitivity Method

Forward Sensitivity Equations:

$$\begin{aligned} \frac{\partial f}{\partial \dot{x}}\left(\frac{\partial \dot{x}}{\partial p}\right)+\frac{\partial f}{\partial x}\left(\frac{\partial x}{\partial p}\right)+\frac{\partial f}{\partial p} &= 0,\ t\in\left[t_0,t_f\right] \\ \frac{\partial x(t_0)}{\partial p} &= \frac{\partial x_0}{\partial p} \\ \frac{\partial \dot{x}(t_0)}{\partial p} &= \frac{\partial \dot{x}_0}{\partial p} \end{aligned}$$

Forward Reduced Gradient

$$\frac{\partial \hat{d}}{\partial p}=\int_{t_0}^{t_f}\left(\frac{\partial g}{\partial \dot{x}}\frac{\partial \dot{x}}{\partial p}+\frac{\partial g}{\partial x}\frac{\partial x}{\partial p}+\frac{\partial g}{\partial p}\right)dt+\left(\frac{\partial h}{\partial \dot{x}}\frac{\partial \dot{x}}{\partial p}+\frac{\partial h}{\partial x}\frac{\partial x}{\partial p}+\frac{\partial h}{\partial p}\right)\Bigg|_{t=t_f}$$

Forward Sensitivity Methods:

- The forward sensitivities d(x)/d(p) are integrated right along with the forward state equation
- Can be solved using explicit or implicti time integration mehtods
- $O(n_p)$ extra storage and computation per time step
- Reuse of forward solver integrator infrastructure, Jacobian/preconditioner storage

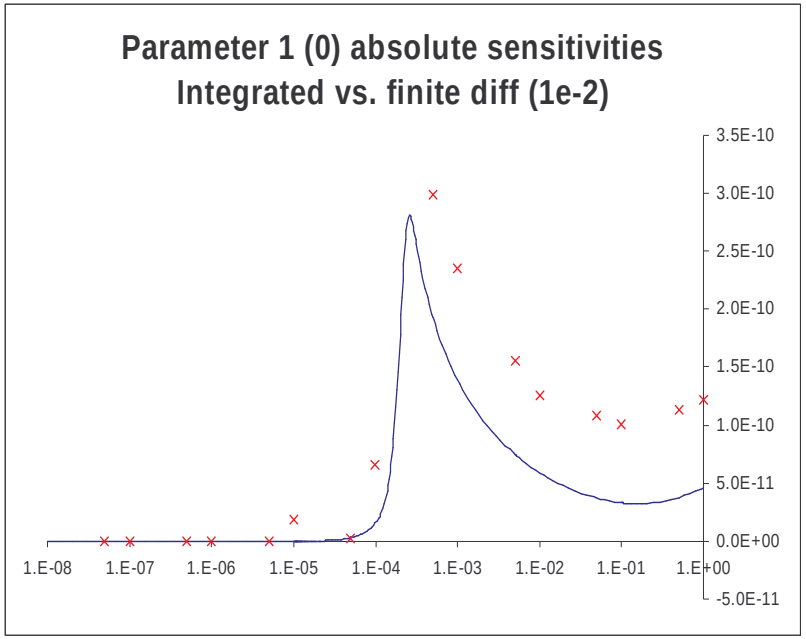
http://www.cs.sandia.gov/~rabartl/TransientSensitivitiesDerivation2007.pdf

Sandia National Laboratories

Forward Transient Sensitivities with Charon/Rythmos

QASPR transient current sensitivities w.r.t. reaction parameters for an irradiated semiconductor device modeled with Charon

- Embedded sensitivities with AD/Sacado (Phipps) & Rythmos
- Finite differences (steplen=1e-2) (optimal steplen=1e-1)
- Embedded sensitivities vs. finite diff.
  - Much more accurate and robust!
  - 10x faster for 40 parameters!



Bartlett, Roscoe, Scott Collis, Todd Coffey, David Day, Mike Heroux, Rob Hoekstra, Russell Hooper, Roger Pawlowski, Eric Phipps, Denis Ridzal, Andy Salinger, Heidi Thornquist, and Jim Willenbring. *ASC Vertical Integration Milestone*. SAND2007-5839, Sandia National Laboratories, 2007 [http://www.cs.sandia.gov/~rabartl/publications.html]

Sandia National Laboratories

ModelEvaluator Software Summary

- Motivation for Unified ModelEvaluator Approach to Nonlinear Problems
  - Large overlap in commonality between requirements for many nonlinear abstract numerical algorithms (ANAs).
  - Incremental support for sensitivities, optimization, and UQ
  - Mixed problem types will become more and more common and must be easy to support

- Properties of ModelEvalutor Software
  - Strong Typing of Input/Object Types but Weak Typing of Problem Formulation
  - Designed for Augmentation
  - Incremental Development of Model Types and Capabilities
  - Self-Describing Models => Smart Algorithms
  - Independence/Multiplicity of Input and Output Objects

- ANAs already using or can use ModelEvaluator
  - NOX (nonlinear equations)
  - LOCA (stability analysis, continuation)
  - Rythmos (explicit ODEs, implicit ODEs, DAEs)
  - MOOCHO (constrained optimization, unconstrained optimization, nonlinear equations)
  - Aristos (full space, trust-region optimization)

Sandia National Laboratories