SAND2011-1425C

Multiphysics Coupling via LIME:

Lightweight Integrating Multiphysics Environment

Russell W. Hooper*,

Roger Pawlowski, Kenneth Belcourt & Rodney Schmidt

Sandia National Labs

Mar. 1, 2011

SIAM CSE 2011,

Reno, NV

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

CASL: The Consortium for Advanced Simulation of Light Water Reactors

A DOE Energy Innovation Hub for Modeling and Simulation of Nuclear Reactors

Leverage

- Current state-of-the-art neutronics, thermal-fluid, structural, and fuel performance applications
- Existing systems and safety analysis simulation tools

Develop

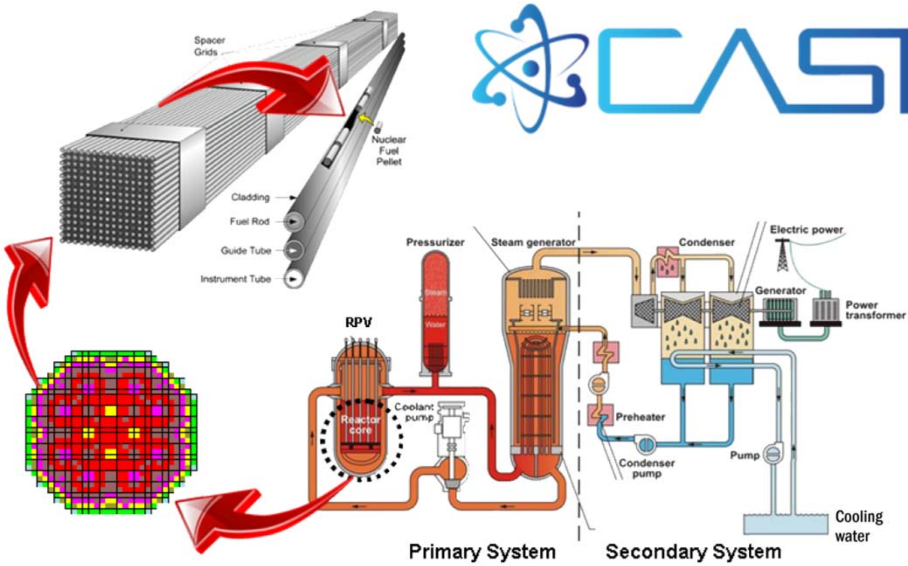
- New requirements-driven physical models
- Efficient, tightly-coupled multi-scale/multi-physics algorithms and software with quantifiable accuracy
- Improved systems and safety analysis tools
- UQ framework

Deliver

- An unprecedented predictive simulation tool for simulation of physical reactors
- Architected for platform portability ranging from desktops to DOE's leadership-class and advanced architecture systems (large user base)
- Rigorous Verification & Validation against existing reactors and data

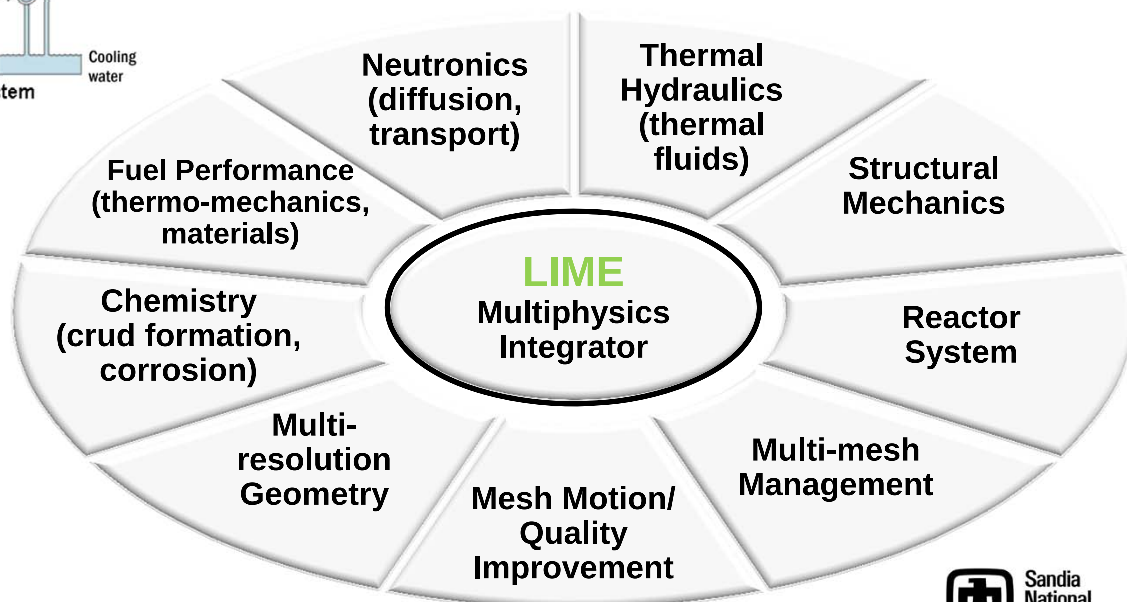
Sandia National Laboratories

CASL vision: Create a virtual reactor (VR) for predictive simulation of LWRs

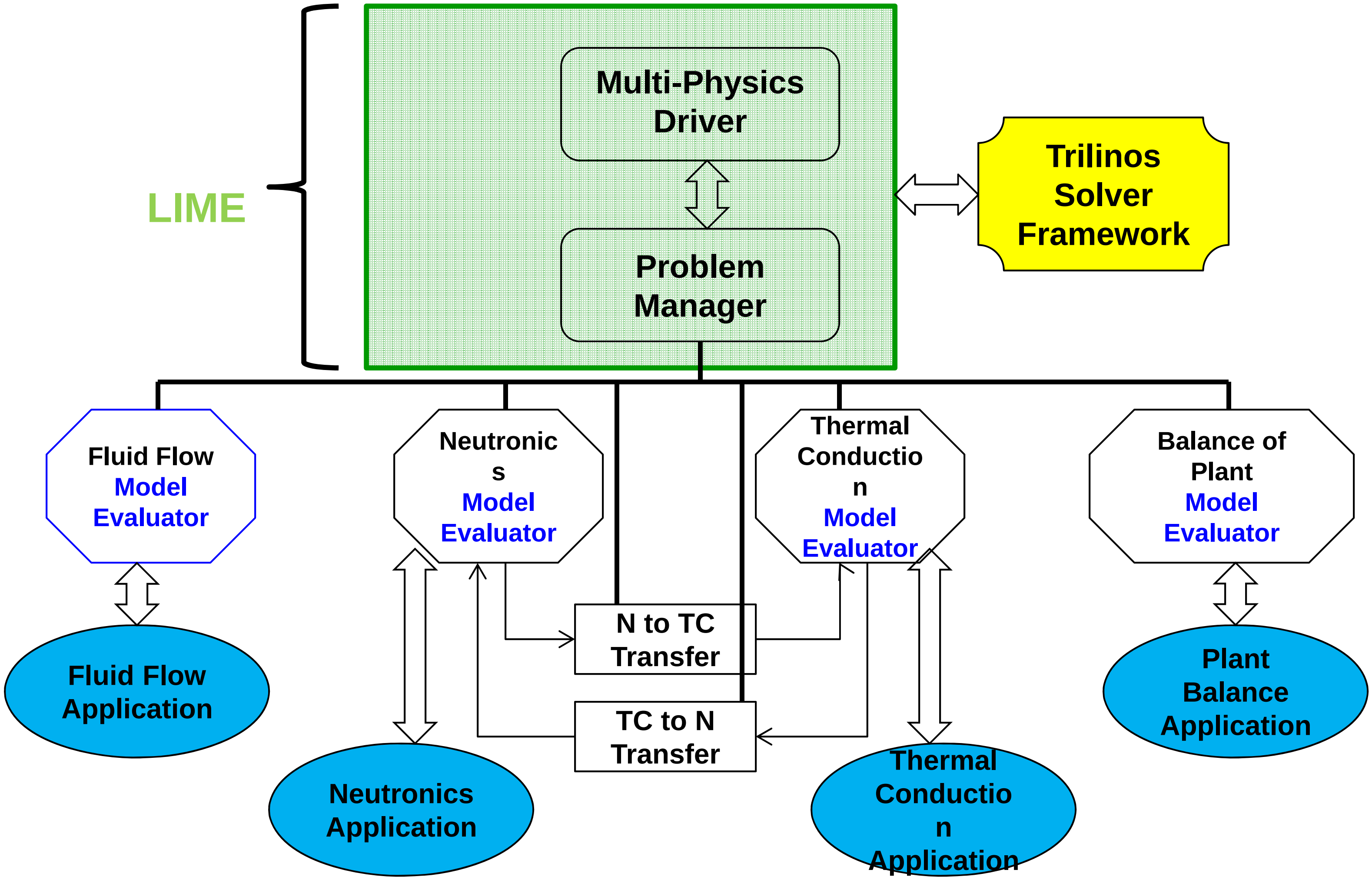


- Multi-scale: temporal and spatial
- Dynamic components: plug-and-play models and codes
- Tractable pre- & post-simulation complexity via friendly user-interface

- Efficient, tightly-coupled multi-scale/multi-physics algorithms and software with quantifiable accuracy
- Portability ranging from desktops to DOE's leadership-class and advanced architecture systems
- Works within UQ (Uncertainty Quantification) framework



LIME Multi-Physics



Model Evaluator:
Interface to Application Codes

- **Each code is wrapped so the Problem Manager can link to it (i.e. like a library).**

Model Evaluator interface allows an application to be treated as a flexible subroutine

InArgs
$\mathbf{x}$ $\dot{\mathbf{x}}$ $\mathbf{p}$
OutArgs
$\mathbf{R}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{p})$ $\frac{\partial \mathbf{R}}{\partial \dot{\mathbf{x}}}$ $\frac{\partial \mathbf{R}}{\partial \mathbf{x}}$ $\frac{\partial \mathbf{R}}{\partial \mathbf{p}}$ $\mathbf{R}(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{p}) = 0$

Very Simple Example

$$r_1(x_1, x_2) = 2x_1 - x_2 + k - 7 = 0$$

$$r_2(x_1, x_2) = x_1 + 2x_2 - 2k + 9 = 0$$

$$\mathbf{R}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}(k) = \mathbf{0}$$

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}(k)$$

$$k = 3, \quad \mathbf{x} = (x_1, x_2) = (1, -2)$$

Very Simple Example

$$r_1\left(x_1, x_2\right) = 2x_1 - x_2 + k - 7 = 0$$

$$r_2\left(x_1, x_2\right) = x_1 + 2x_2 - 2k + 9 = 0$$

Replace constant with value from another model

$$r_1\left(x_1, x_2, \tilde{x}_3\right) = 2x_1 - x_2 + \tilde{x}_3 - 7 = 0$$

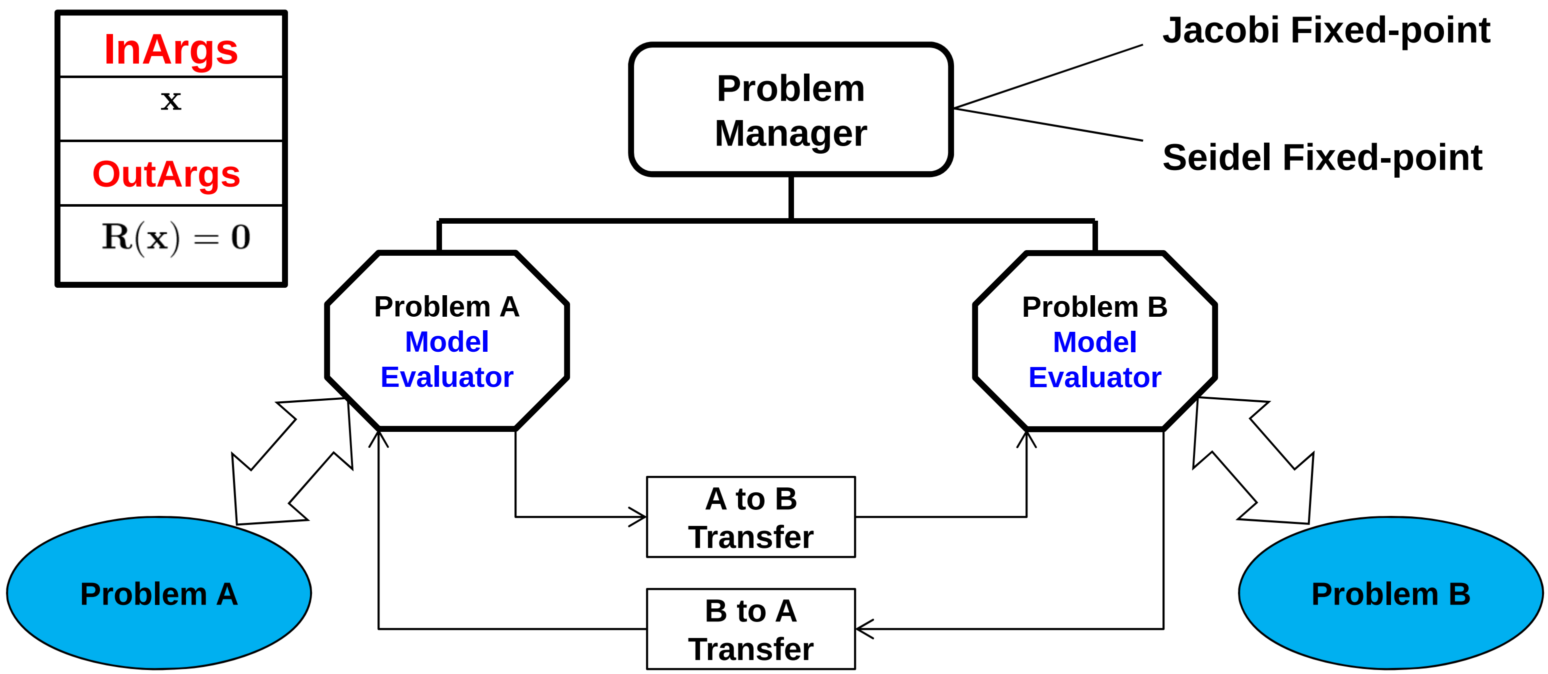
$$r_2\left(x_1, x_2, \tilde{x}_3\right) = x_1 + 2x_2 - 2\tilde{x}_3 + 9 = 0$$

$$r_3\left(\tilde{x}_1, \tilde{x}_2, x_3\right) = \left(\tilde{x}_1\right)^2 + \tilde{x}_2 + 2x_3 - 5 = 0$$

Very Simple Example

$$\mathbf{R}_A\left(\mathbf{x}_A, \tilde{\mathbf{x}}_B\right) = \mathbf{A}\mathbf{x}_A - \mathbf{b}_A(\tilde{\mathbf{x}}_B) = \mathbf{0}$$

$$\mathbf{R}_B\left(\mathbf{x}_B, \tilde{\mathbf{x}}_A\right) = \mathbf{B}\mathbf{x}_B - \mathbf{b}_B(\tilde{\mathbf{x}}_A) = \mathbf{0}$$



Very Simple Example

$$r_1(x_1, x_2) = 2x_1 - x_2 + k - 7 = 0$$

$$r_2(x_1, x_2) = x_1 + 2x_2 - 2k + 9 = 0$$

Replace constant with value from another model

$$r_1(x_1, x_2, \tilde{x}_3) = 2x_1 - x_2 + \tilde{x}_3 - 7 = 0$$

$$r_2(x_1, x_2, \tilde{x}_3) = x_1 + 2x_2 - 2\tilde{x}_3 + 9 = 0$$

$$r_3(\tilde{x}_1, \tilde{x}_2, x_3) = (\tilde{x}_1)^2 + \tilde{x}_2 + 2x_3 - 5 = 0$$

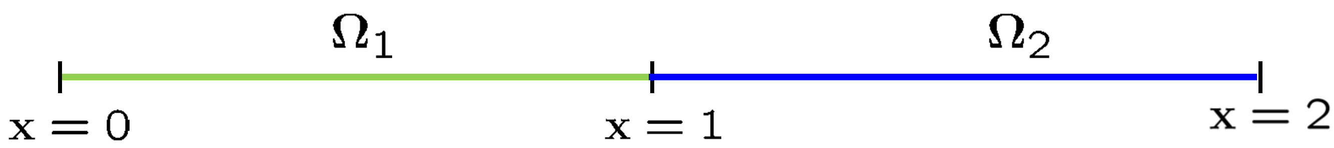
Fixed-Point does not converge !

Sandia National Laboratories

A Better Example

1D Conjugate Heat Transfer

Yeckel et al., IJNME, v 67, n 12, 2006.



$$T|_{x=0} = T_0 \qquad \frac{d^2T}{dx^2} - c\frac{dT}{dx} = 0 \qquad \kappa\frac{d^2T}{dx^2} = 0 \qquad T|_{x=2} = T_2$$

$$q|_{x=1^-} = -\left.\frac{dT}{dx}\right|_{x=1^-}$$

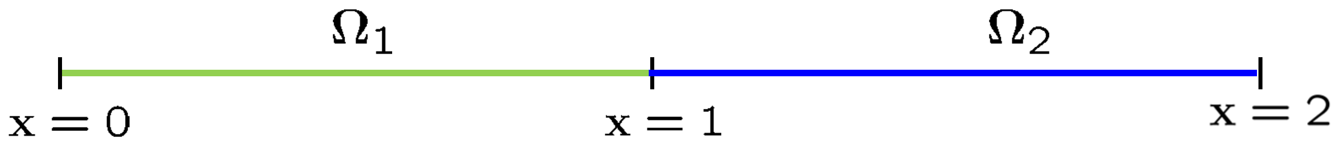
$$q|_{x=1^+} = -\kappa\left.\frac{dT}{dx}\right|_{x=1^+} + \underline{R}\left(T^4|_{x=1^+} - T_2^4\right)$$

$$T|_{x=1^-} = T|_{x=1^+}$$

A Better Example

1D Conjugate Heat Transfer

Yeckel et al., IJNME, v 67, n 12, 2006.



$$\frac{d^2T}{dx^2} - c\frac{dT}{dx} = 0 \qquad \kappa\frac{d^2T}{dx^2} = 0$$

$$q|_{x=1^-} = -\left.\frac{dT}{dx}\right|_{x=1^-}$$

$$q|_{x=1^+} = -\kappa\left.\frac{dT}{dx}\right|_{x=1^+} + \underline{R}\left(T^4|_{x=1^+} - T_2^4\right)$$

$$\underline{\alpha}[\![q]\!] + (1-\alpha)[\![T]\!] = 0 \quad \text{at} \quad x = 1^-$$

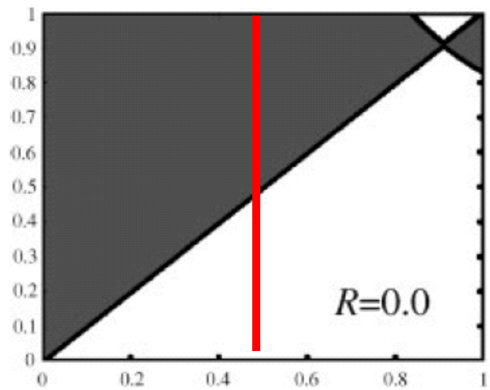
$$\underline{\beta}[\![q]\!] + (1-\beta)[\![T]\!] = 0 \quad \text{at} \quad x = 1^+$$

Sandia National Laboratories

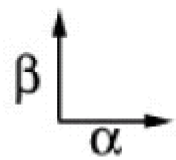
A Better Example

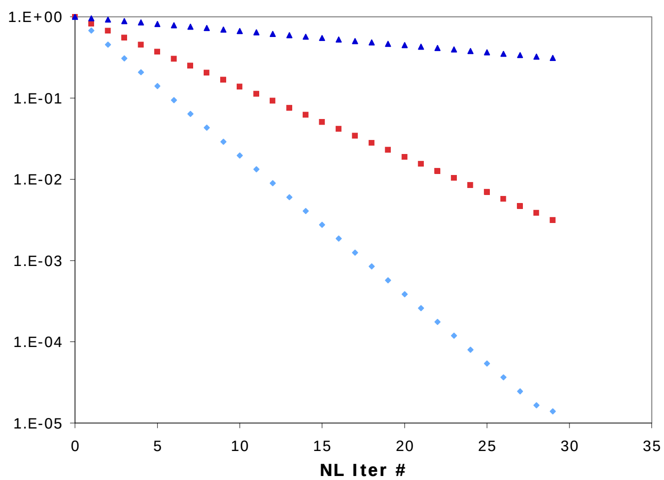
1D Conjugate Heat Transfer

Convergence and convergence rate of loose-coupling can be strongly problem-dependent.



Shaded regions indicate failure



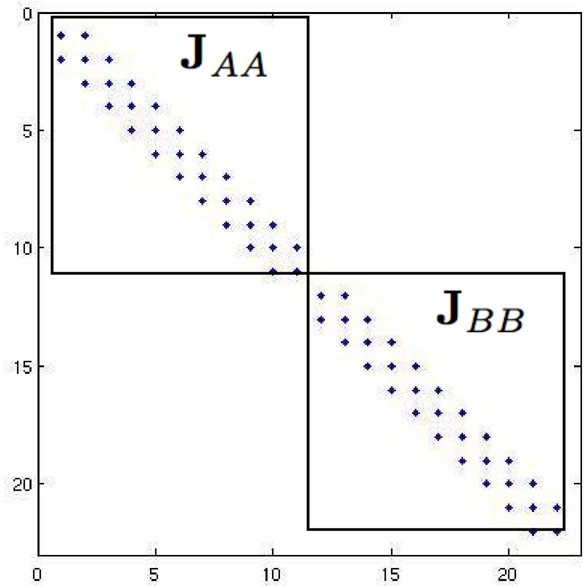


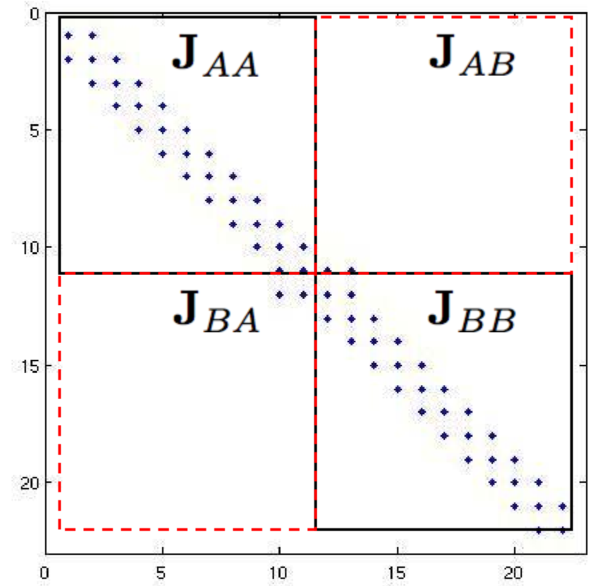
α	β	Fixed-pt Iters, R=0
0.50	0.40	33
0.50	0.45	60
0.50	0.49	253
0.50	0.60	N/A

A Better Example

1D Conjugate Heat Transfer

So close, yet so far away





$$\tilde{\mathbf{J}}\left(\mathbf{x}^k\right)\Delta\mathbf{x}^{k+1} = -\mathbf{R}\left(\mathbf{x}^k\right)$$

$$\mathbf{E}^k \equiv \mathbf{x}^* - \mathbf{x}^k$$

$$\mathbf{0} = \mathbf{R}\left(\mathbf{x}^*\right) = \mathbf{R}\left(\mathbf{x}^k\right) + \mathbf{J}\left(\mathbf{x}^k\right)\mathbf{E}^k + O\left(\left|\mathbf{E}^k\right|^2\right)$$

$$\Delta\mathbf{x}^{k+1} = \left[\tilde{\mathbf{J}}^{-1}\mathbf{J}\right]\mathbf{E}^k + O\left(\left|\mathbf{E}^k\right|^2\right)$$

$$\mathbf{E}^{k+1} = \left[\mathbf{I} - \tilde{\mathbf{J}}^{-1}\mathbf{J}\right]\mathbf{E}^k + O\left(\left|\mathbf{E}^k\right|^2\right)$$

$$\mathbf{E}^{k+1} = \mathbf{G}\left(\mathbf{x}^k\right)\mathbf{E}^k + O\left(\left|\mathbf{E}^k\right|^2\right)$$

Jacobian-Free Newton-Krylov

Matrix-Free Newton-Krylov :

$$\mathbf{R}(\mathbf{x} + \epsilon \mathbf{p}) = \mathbf{R}(\mathbf{x}) + \mathbf{J}(\mathbf{x} + \epsilon \mathbf{p} - \mathbf{x}) + O(||\epsilon \mathbf{p}||^2)$$

$$\mathbf{J}\mathbf{p} \approx \frac{\mathbf{R}(\mathbf{x} + \epsilon \mathbf{p}) - \mathbf{R}(\mathbf{x})}{\epsilon}$$

Jacobian-Free Newton-Krylov with Preconditioning:

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{p} \approx \frac{\mathbf{R}(\mathbf{x} + \epsilon \mathbf{M}^{-1}\mathbf{p}) - \mathbf{R}(\mathbf{x})}{\epsilon}$$

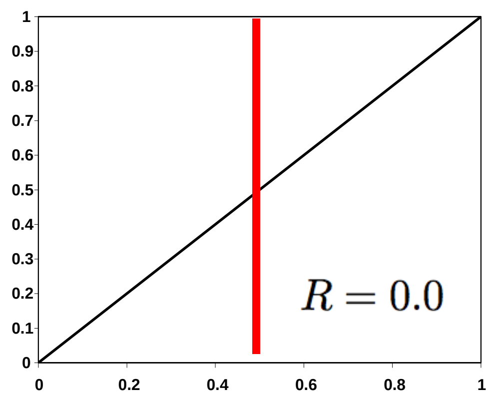
JFNK relaxes interface requirements to nearly that needed for fixed-point.

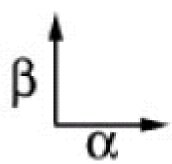
Sandia National Laboratories

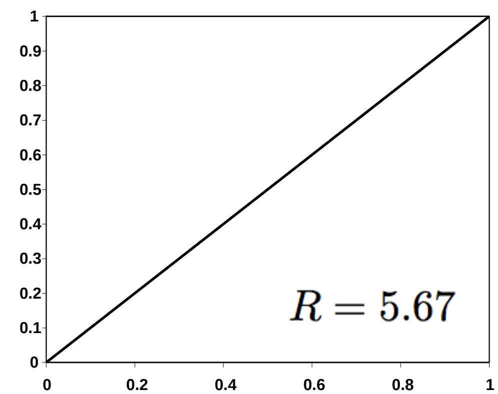
A Better Example
1D Conjugate Heat Transfer

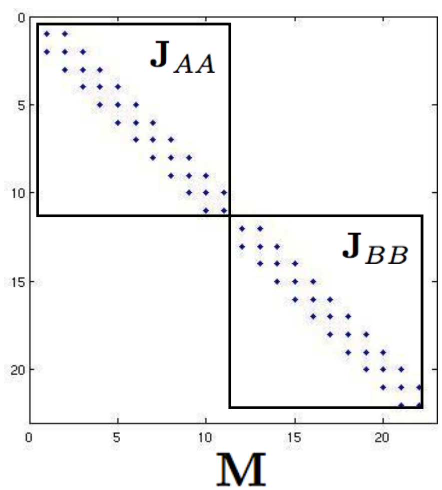
Convergence and convergence rate of loose-coupling can be strongly problem-dependent.

Newton-Based (JFNK) coupling dramatically improves both.





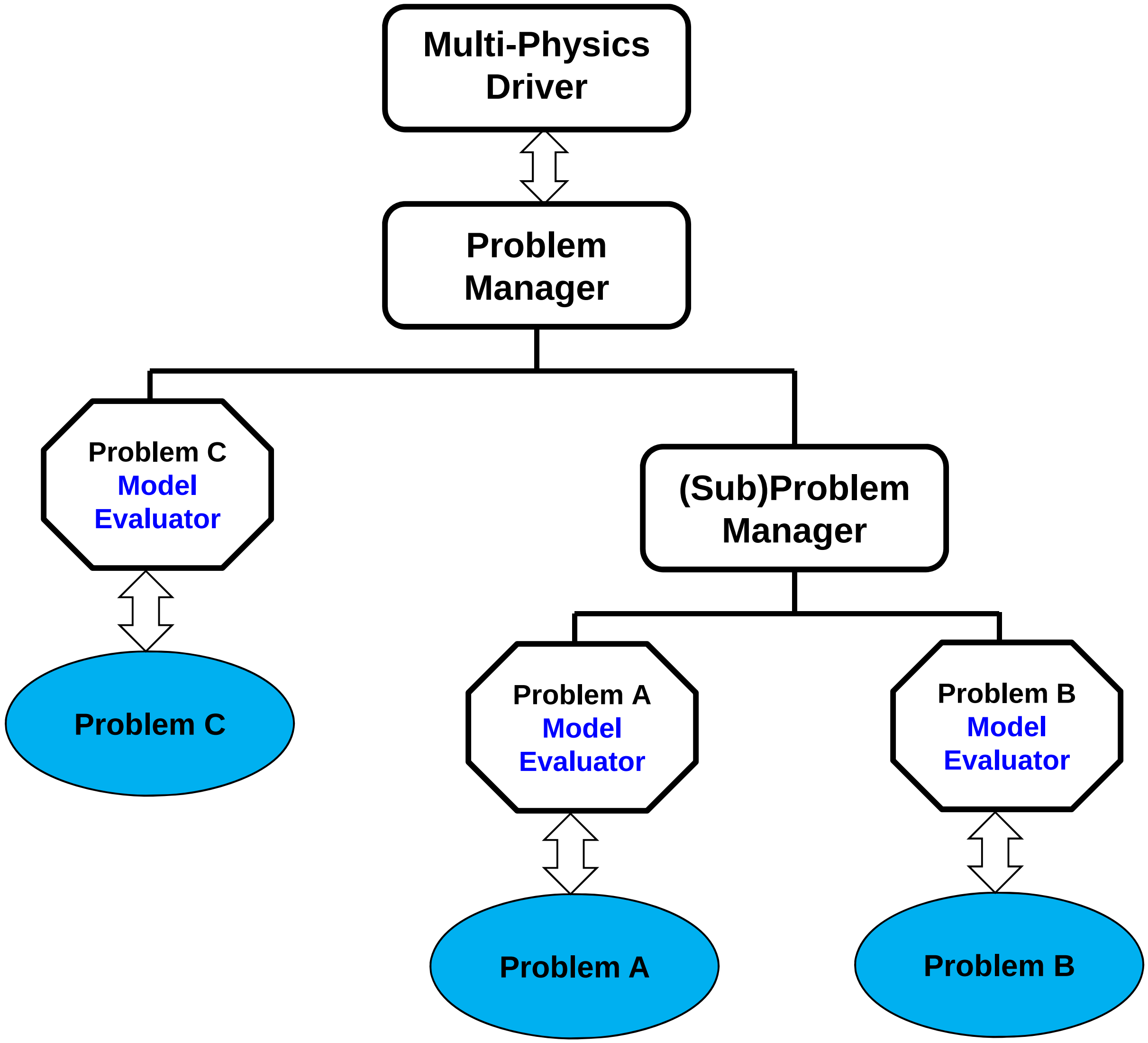




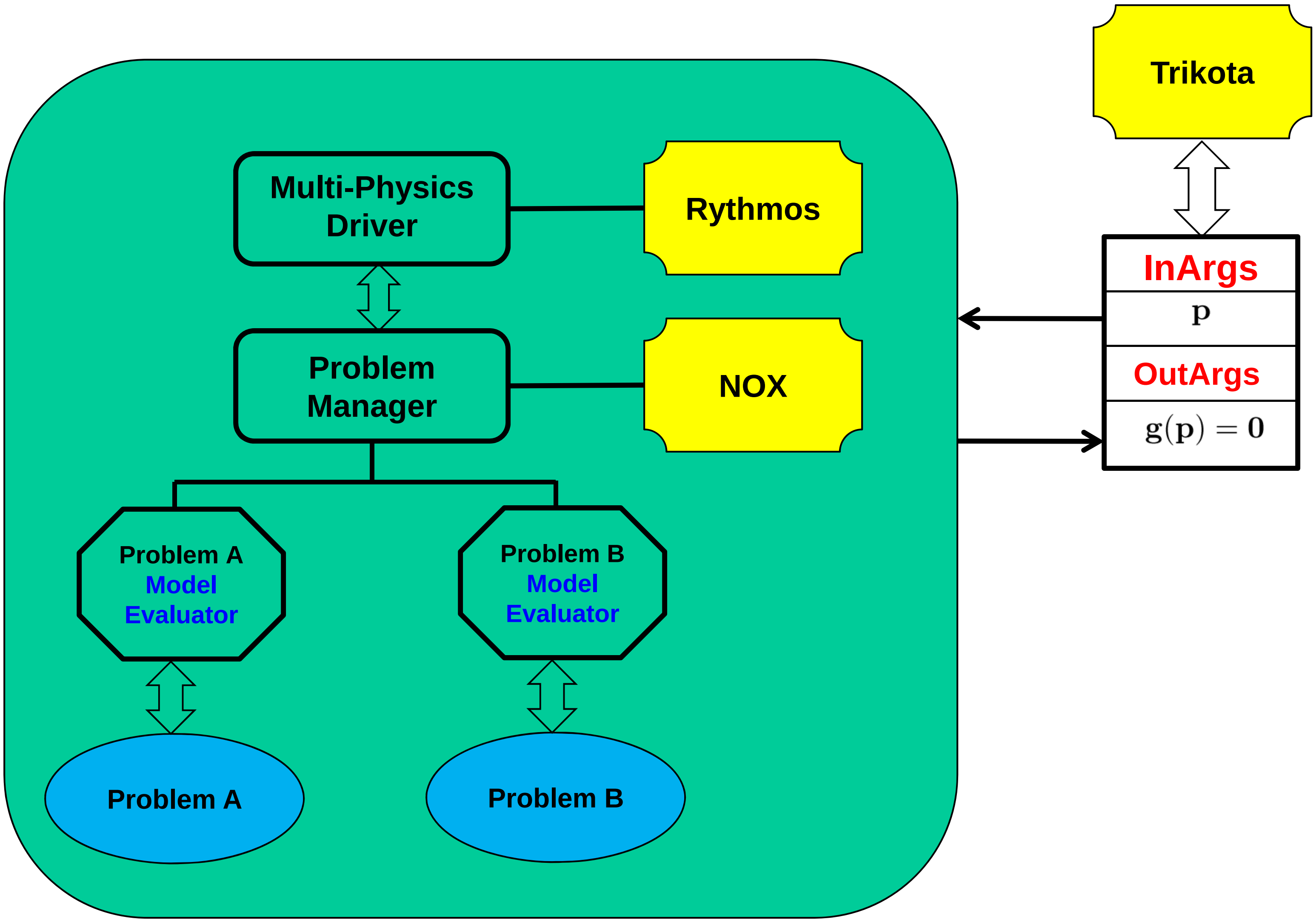
$$\mathbf{J}\mathbf{M}^{-1}\mathbf{p} \approx \frac{\mathbf{R}(\mathbf{x} + \epsilon\mathbf{M}^{-1}\mathbf{p}) - \mathbf{R}(\mathbf{x})}{\epsilon}$$

α	β	Fixed-pt Iters, R=0	JFNK Iters, R=0	JFNK Iters, R=5.67
0.50	0.40	33	1	3
0.50	0.45	60	1	3
0.50	0.49	253	1	3
0.50	0.60	N/A	1	3

Hierarchical Support



Meta-Solver Support



“Brusselator” Problem

Transient Thermal Diffusion with Source:

$$\frac{\partial T}{\partial t} = D_1 \frac{\partial^2 T}{\partial x^2} + \alpha - (1+\beta)T + CT^2$$

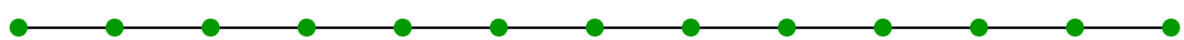
Transient Species Diffusion with Source:

$$\frac{\partial C}{\partial t} = D_2 \frac{\partial^2 C}{\partial x^2} + \alpha + \beta T - CT^2$$

Adjustable Parameter

$$D_1 = D_2 = 0.025, \alpha = 0.6, \beta = 2.0$$

$T_L = \alpha$ $\quad$ $T_R = \alpha$

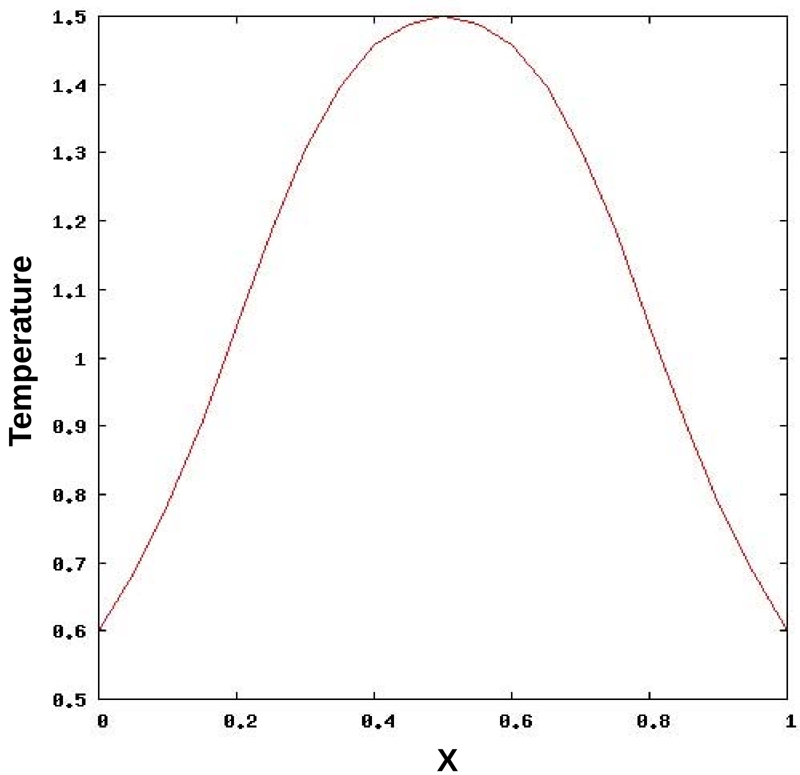


$C_L = \frac{\beta}{\alpha}$ $\quad$ $C_R = \frac{\beta}{\alpha}$

Sandia National Laboratories

Brusselator Problem

$$g(p) \equiv \min_{p}[1.5 - T_{max}(p)]^2$$



$$p_0 = D_1 = D_2 = 0.01$$

k	p_k	g_k
0	0.01	0.112
1	0.001984	2.01e-4
2	0.00196913	1.53e-7
3	0.00196912	4.50e-11

Conclusions

- **LIME provides a very lightweight connection between physics applications and solver algorithms**
- **Increasing richness of algorithms can be enabled by exposing more application data/functionality through each Model Evaluator as needed or desired**
- **LIME is an independent project but integrates seamlessly into the Trilinos software environment**
- **LIME will eventually be publicly available**
- **Current and future energy needs of the USA may benefit greatly from multi-physics modeling/simulation enabled to large extent by LIME**

End

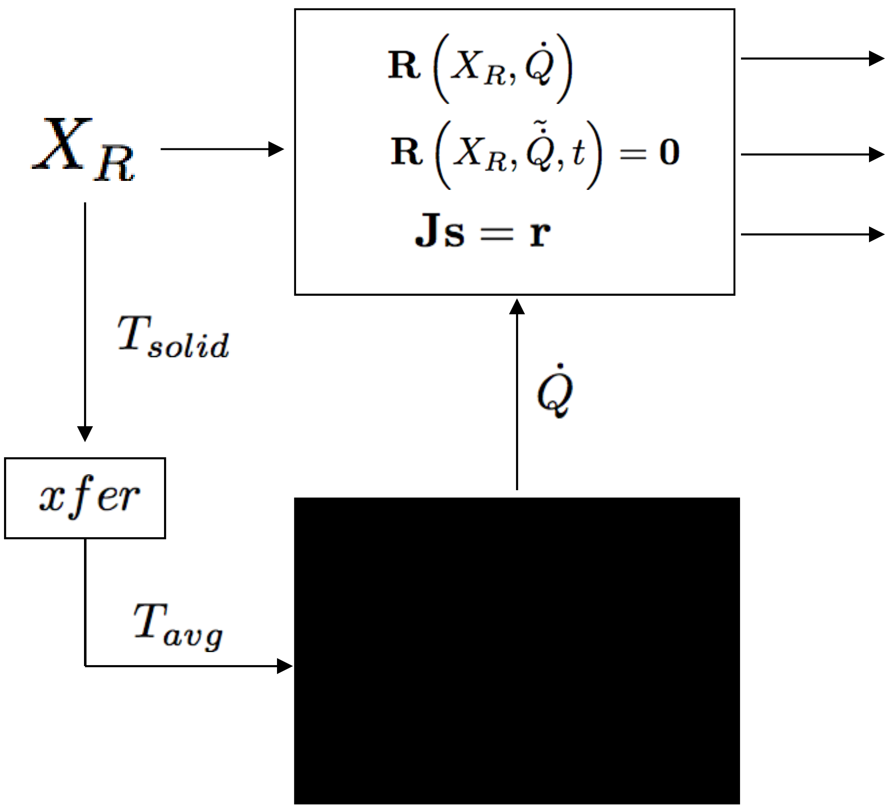
&

Backup Slides

Relaxing Requirements, Part 3
(Response Only Model Evaluator)

- **Fluid-Thermal Code** - $X_R \equiv (u, v, w, P, T_{fluid}, T_{solid})$

- **Neutronics** - $X_N \equiv (?)$



$$\mathbf{R}_C \equiv \mathbf{R}\left(X_R, \dot{Q}(X_R)\right) = \mathbf{0}$$

$$\mathbf{Jp} \approx \frac{\mathbf{R}_C(X_R + \epsilon \mathbf{p}) - \mathbf{R}_C(X_R)}{\epsilon}$$

Sandia National Laboratories

Typical Starting Point

```
call fillRHS ( xVec, RHSvec )
norm = TwoNorm( RHSvec )
while ( norm > tol )
  call fillJacobian ( xVec, Mat )
  call linSolver ( Mat, solnVec, RHSvec)
  call daxpy( xVec, 1.0, solnVec, 1.0)
  call fillRHS ( xVec, RHSvec )
  norm = TwoNorm( RHSvec )
end while
```

$$R(T) = 0$$

```
subroutine linSolver ( Mat, solnVec, RHSvec )
      …………….
return

subroutine fillRHS ( xVec, RHSvec )
      …………….
return

subroutine fillJacobian ( xVec, Mat )
      ……………….
return
```

Typical Starting Point

```
call fillRHS ( xVec, RHSvec )
norm = TwoNorm( RHSvec )
while ( norm > tol )
  call fillJacobian ( xVec, Mat )
  call linSolver ( Mat, solnVec, RHSvec)
  call daxpy( xVec, 1.0, solnVec, 1.0)
  call fillRHS ( xVec, RHSvec )
  norm = TwoNorm( RHSvec )
end while
```

LIME Problem Manager

$$R(T) = 0$$

```
subroutine linSolver ( Mat, solnVec, RHSvec )
      …………….
return

subroutine fillRHS ( xVec, RHSvec )
      …………….
return

subroutine fillJacobian ( xVec, Mat )
      ……………….
return
```

USER

Relaxing Requirements
(Avoiding Jacobian pain)

Finite Difference Approximation :

$$J_{ij} = \frac{\partial F_i}{\partial x_j} \approx \frac{F_i(\mathbf{x} + \epsilon \mathbf{e}_j) - F_i(\mathbf{x})}{\epsilon} \qquad Cost \sim O(N^3)$$

Relaxing Requirements
(Avoiding Jacobian pain)

Finite Difference Approximation :

$$J_{ij} = \frac{\partial F_i}{\partial x_j} \approx \frac{F_i(\mathbf{x} + \epsilon \mathbf{e}_j) - F_i(\mathbf{x})}{\epsilon} \qquad Cost \sim O(N^3)$$

Finite Differences with Coloring : $\quad Cost \sim O(N^2)$

$$\begin{array}{cccccc} \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{1} & \mathbf{2} & \mathbf{3} \end{array}$$

$$\begin{bmatrix} \times & \times & 0 & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 & 0 \\ 0 & \times & \times & \times & 0 & 0 \\ 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{bmatrix}$$

$$C_1 = \{1,4\}, \quad C_2 = \{2,5\}, \quad C_3 = \{3,6\}$$

$$\mathbf{c}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{c}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Graph partitioning from Isoroppia or EpetraExt in Trilinos.