

Physics-based Preconditioners and Optimization

David E. Keyes

Department of Applied Physics & Applied Mathematics

Columbia University

Institute for Scientific Computing Research

Lawrence Livermore National Laboratory

Recall Newton methods

- Given $F(u) = 0$, $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and iterate u^0 we wish to pick u^{k+1} such that

$$F(u^{k+1}) \approx F(u^k) + F'(u^k) \mathbf{d}u^k = 0$$

where $\mathbf{d}u^k = u^{k+1} - u^k$, $k = 0, 1, 2, \dots$

- Neglecting higher-order terms, we get

$$\mathbf{d}u^k = -[J(u^k)]^{-1} F(u^k)$$

where $J = F'(u^k)$ is the Jacobian matrix, generally large, sparse, and ill-conditioned for PDEs

- In practice, require $\|F(u^k) + J(u^k) \mathbf{d}u^k\| < \epsilon$
- In practice, set $u^{k+1} = u^k + \mathbf{l} \mathbf{d}u^k$ where \mathbf{l} is selected to minimize $\|F(u^k + \mathbf{l} \mathbf{d}u^k)\|$



Jacobian-free Newton-Krylov

- In the Jacobian-Free Newton-Krylov (JFNK) method, a Krylov method solves the linear Newton correction equation, requiring Jacobian-vector products
- These are approximated by the Fréchet derivatives

$$J(u)v \approx \frac{1}{\mathbf{e}} [F(u + \mathbf{e}v) - F(u)]$$

(where \mathbf{e} is chosen with a fine balance between approximation and floating point rounding error) or automatic differentiation, so that the actual Jacobian elements are *never explicitly needed*

- One builds the Krylov space on a true $F'(u)$ (to within numerical approximation)



Recall idea of preconditioning

- Krylov iteration is expensive in memory and in function evaluations, so k must be kept small in practice, through preconditioning the Jacobian with an approximate inverse, so that the product matrix has low condition number in

$$(B^{-1}A)x = B^{-1}b$$

- Given the ability to apply the action of B^{-1} to a vector, preconditioning can be done on either the left, as above, or the right, as in, e.g., for matrix-free:

$$JB^{-1}v \approx \frac{1}{e} [F(u + eB^{-1}v) - F(u)]$$



Philosophy of Jacobian-free NK

- To *evaluate* the linear residual, we use the true $F'(u)$, giving a true Newton step and asymptotic quadratic Newton convergence
- To *precondition* the linear residual, we do anything convenient that uses understanding of the dominant physics/mathematics in the system and respects the limitations of the parallel computer architecture and the cost of various operations:
 - Jacobian of lower-order discretization
 - Jacobian with “lagged” values for expensive terms
 - Jacobian stored in lower precision
 - Jacobian blocks decomposed for parallelism
 - Jacobian of related discretization
 - operator-split Jacobians
 - physics-based preconditioning



Using Jacobian of related discretization

- To precondition a variable coefficient operator, such as $\nabla \cdot (\mathbf{a} \nabla \bullet)$, use $\bar{\mathbf{a}} \nabla^2$, based on a constant coefficient average
- Brown & Saad (1980) showed that, because of the availability of fast solvers, it may even be acceptable to use $-\nabla^2$ to precondition something like

$$-\nabla^2(\bullet) + u \frac{\partial(\bullet)}{\partial x} + v \frac{\partial(\bullet)}{\partial y}$$



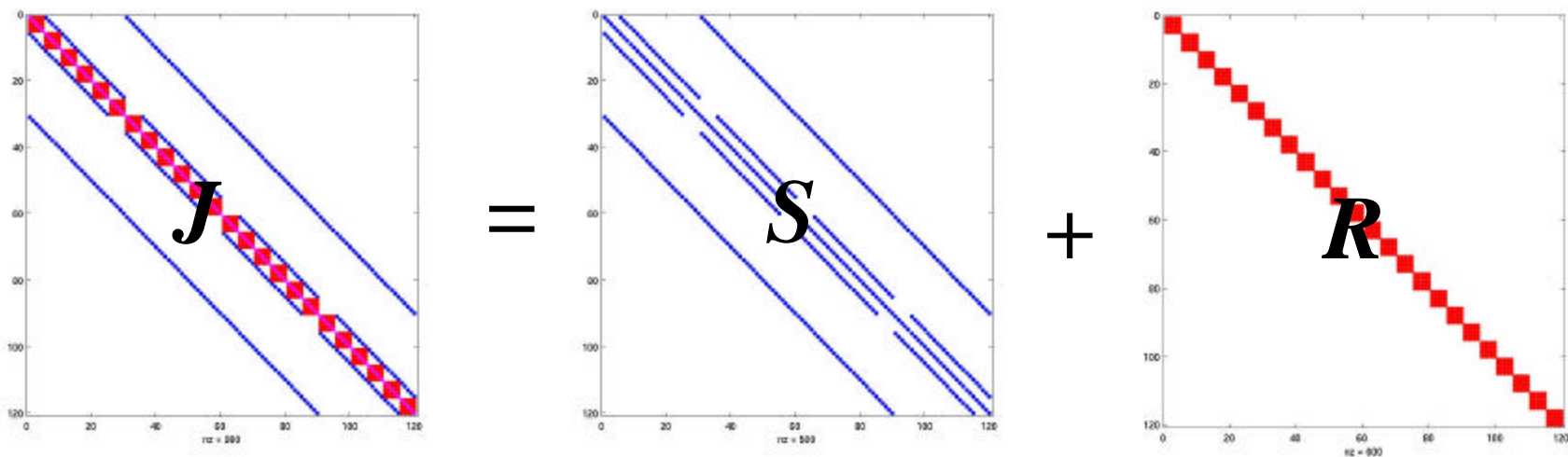
Operator-split preconditioning

- Subcomponents of a PDE operator often have special structure that can be exploited if they are treated separately
- Algebraically, this is just a generalization of Schwarz, by term instead of by subdomain
- Suppose $J = \boxed{t^{-1}I + S + R}$ and a preconditioner is to be constructed, where $I + tS$ and $I + tR$ are each “easy” to invert
- Form a preconditioned vector from u as follows:
$$(t^{-1}I + R)^{-1} (I + tS)^{-1} u$$
- Equivalent to replacing J with $\boxed{t^{-1}I + S + R} + tSR$
- First-order splitting error, yet *often used as a solver!*



Operator-split preconditioning, cont.

- Suppose S is convection-diffusion and R is reaction, among a collection of fields stored as gridfunctions
- On a small regular 2D grid with a five-point stencil:



- R is trivially invertible in block diagonal form
- S is invertible with one multilevel solve per field



Operator-split preconditioning, cont.

- **Preconditioners assembled from just the “strong” elements of the Jacobian, alternating the source term and the diffusion term operators, are competitive in convergence rates with block-ILU on the Jacobian**
 - **particularly, since the decoupled scalar diffusion systems are amenable to simple multigrid treatment – not as trivial for the coupled system**
- **The decoupled preconditioners store many fewer elements and significantly reduce memory bandwidth requirements and are expected to be much faster per iteration when carefully implemented**
- **See “alternative block factorization” by Bank et al.; incorporated into SciDAC TSI solver by D’Azevedo**



Physics-based preconditioning

- In Newton iteration, one seeks to obtain a correction (“delta”) to solution, by inverting the Jacobian matrix on (the negative of) the nonlinear residual:

$$\mathbf{d}u^k = -[J(u^k)]^{-1} F(u^k)$$

- A typical operator-split code also derives a “delta” to the solution, by some implicitly defined means, through a series of implicit and explicit substeps

$$F(u^k) \mapsto \mathbf{d}u^k$$

- This implicitly defined mapping from residual to “delta” is a *natural* preconditioner
- Software must accommodate this!



Physics-based Preconditioning

- We consider a standard “dynamical core,” the shallow-water wave splitting algorithm, *as a solver*
- Leaves a first-order in time splitting error
- In the Jacobian-free Newton-Krylov framework, this solver, which maps a residual into a correction, can be regarded *as a preconditioner*
- The true Jacobian is never formed yet the time-implicit nonlinear residual at each time step can be made as small as needed for nonlinear consistency in long time integrations



Example: shallow water equations

- Continuity (*)

$$\frac{\partial}{\partial t} \times \frac{\partial \mathbf{f}}{\partial t} + \frac{\partial (u \mathbf{f})}{\partial x} = 0$$

- Momentum (**)

$$\frac{\partial}{\partial x} \times \frac{\partial (u \mathbf{f})}{\partial t} + \frac{\partial (u^2 \mathbf{f})}{\partial x} + g \mathbf{f} \frac{\partial \mathbf{f}}{\partial x} = 0$$

- These equations admit a *fast gravity wave*, as can be seen by cross differentiating, e.g., (*) by t and (**) by x , and subtracting:

$$\frac{\partial^2 \mathbf{f}}{\partial t^2} - g \mathbf{f} \frac{\partial^2 \mathbf{f}}{\partial x^2} = \text{other terms}$$



1D shallow water equations, cont.

- Wave equation for geopotential:

$$\frac{\partial^2 \mathbf{f}}{\partial t^2} - g \mathbf{f} \frac{\partial^2 \mathbf{f}}{\partial x^2} = \text{other terms}$$

- Gravity wave speed $\sqrt{g \mathbf{f}}$
- Typically $\sqrt{g \mathbf{f}} \gg u$, but stability restrictions would require timesteps based on the Courant-Friedrichs-Levy (CFL) criterion for the fastest wave, for an explicit method
- One can solve *fully implicitly*, or one can filter out the gravity wave by solving *semi-implicitly*



1D shallow water equations, cont.

- Continuity (*)

$$\frac{\boxed{f^{n+1}} - f^n}{t} + \frac{\partial (u f)^{n+1}}{\partial x} = 0$$

- Momentum (**)

$$\frac{\boxed{(u f)^{n+1}} - (u f)^n}{t} + \frac{\partial (u^2 f)^n}{\partial x} + g f^n \frac{\partial \boxed{f^{n+1}}}{\partial x} = 0$$

- Solving (**) for $\boxed{(u f)^{n+1}}$ and substituting into (*),

$$\boxed{f^{n+1}} - g t^2 \frac{\partial}{\partial x} \left(f^n \frac{\partial \boxed{f^{n+1}}}{\partial x} \right) = f^n + \frac{\partial S^n}{\partial x}$$

where

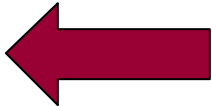
$$S^n = (u f)^n - t \frac{\partial (u^2 f)^n}{\partial x}$$



1D shallow water equations, cont.

- After the parabolic equation is spatially discretized and solved for \mathbf{f}^{n+1} , then $(u \mathbf{f})^{n+1}$ can be found from

$$(u \mathbf{f})^{n+1} = -t g \mathbf{f}^n \frac{\partial \mathbf{f}^{n+1}}{\partial x} + S^n$$

- *One scalar parabolic solve and one scalar explicit update* replace an *implicit hyperbolic system*
- This *semi-implicit* operator splitting is foundational to multiple scales problems in geophysical modeling
- Similar tricks are employed in aerodynamics (*sound waves*), MHD (*multiple Alfvén waves*), reacting flows (*fast kinetics*), etc.
- Temporal truncation error remains due to the lagging of the advection in (**)  To be dealt with shortly



1D Shallow water preconditioning

- Define continuity residual for each timestep:

$$R_f \equiv \frac{\boxed{f^{n+1}} - f^n}{t} + \frac{\partial \boxed{(u f)^{n+1}}}{\partial x}$$

- Define momentum residual for each timestep:

$$R_u f \equiv \frac{\boxed{(u f)^{n+1}} - (u f)^n}{t} + \frac{\partial (u^2 f)^n}{\partial x} + g f^n \frac{\partial \boxed{f^{n+1}}}{\partial x}$$

- Continuity delta-form (*):

$$\frac{\boxed{df}}{t} + \frac{\partial [\boxed{d (u f)}]}{\partial x} = - R_f$$

- Momentum delta form (**):

$$\frac{\boxed{d (u f)}}{t} + g f^n \frac{\partial [\boxed{df}]}{\partial x} = - R_u f$$



1D Shallow water preconditioning, cont.

- Solving (**) for $\mathbf{d}(\mathbf{u} \mathbf{f})$ and substituting into (*),

$$d\mathbf{f} - g t^2 \frac{\partial}{\partial x} (\mathbf{f}^n \frac{\partial [\mathbf{d} \mathbf{f}]}{\partial x}) = -R_{-}\mathbf{f} + t^2 \frac{\partial}{\partial x} (R_{-}\mathbf{u} \mathbf{f})$$

- After this parabolic equation is solved for $d\mathbf{f}$, we have

$$\mathbf{d}(\mathbf{u} \mathbf{f}) = -t g \mathbf{f}^n \frac{\partial [d\mathbf{f}]}{\partial x} - R_{-}\mathbf{u} \mathbf{f}$$

- This completes the application of the preconditioner to one Newton-Krylov iteration at one timestep
- Of course, the parabolic solve need not be done exactly; one sweep of multigrid can be used
- See paper by Mousseau et al. (2002) for impressive results for longtime weather integration



Physics-based preconditioning update

- So far, physics-based preconditioning has been applied to several codes at Los Alamos, in an effort led by D. Knoll
- Summarized in new J. Comp. Phys. paper by Knoll & Keyes (2002, under review)
- PETSc's "shell preconditioner" is ideal for inserting physics-based preconditioners, and PETSc's solvers underneath are ideal building blocks, but there are not yet examples in the public release



SciDAC philosophy on PDEs

- **Solution of a system of PDEs is rarely a goal in itself**
 - PDEs are solved to derive various outputs from specified inputs
 - actual goal is characterization of a response surface or a design or control strategy
 - together with analysis, sensitivities and stability are often desired

⇒ **Tools for PDE solution should also support these related desires**



PDE-constrained optimization

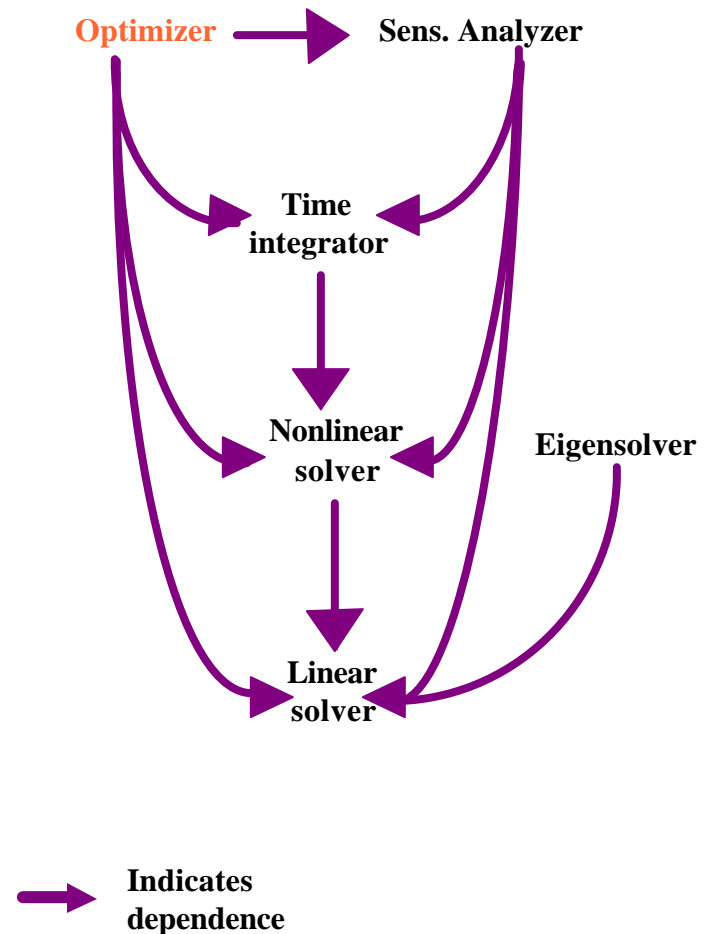
- **PDE-constrained optimization: a relatively new horizon**
 - ... for large-scale PDE solution
 - ◆ next step after reducing to practice parallel implicit solvers for coupled systems of (steady-state) PDEs
 - ◆ now “routine” to solve systems of PDEs with millions of DOFs on thousands of processors
 - ... for constrained optimization
 - ◆ complexity of a single projection to the constraint manifold for million-DOF PDE is too expensive for an inner loop of traditional RSQP method
 - ◆ must devise new “all-at-once” algorithms that seek “exact” feasibility only at optimality
- **Our approach starts from the iterative PDE solver side**



Optimizers

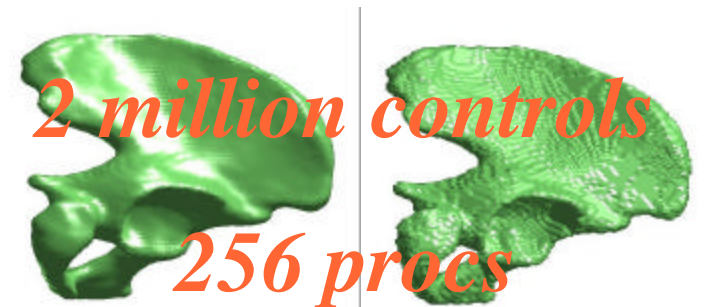
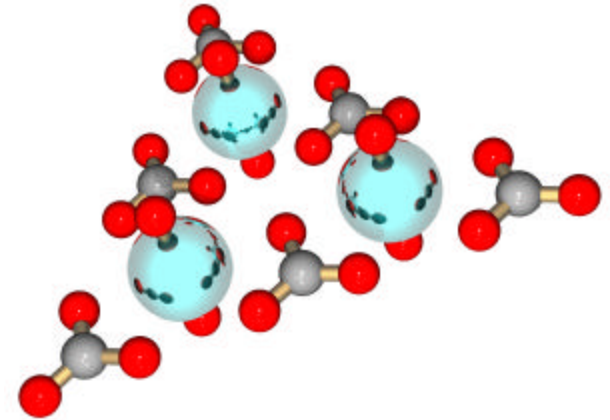
$$\min_u \mathbf{f}(x, u) \text{ s.t. } F(x, u) = 0, u \geq \mathbf{C}$$

- Many simulations are properly posed as optimization problems, but this may not always be recognized
- Unconstrained or bound-constrained applications use **TAO**
- PDE-constrained problems use **Veltisto**
- Both are built on **PETSc** solvers (and **Hypre** preconditioners)
- **TAO** makes heavy use of **AD**, freeing user from much coding
- **Veltisto**, based on **RSQP**, switches as soon as possible to an “all-at-once” method and minimizes the number of PDE solution “work units”



Recent optimization progress

- **Unconstrained or bound-constrained optimization**
 - **TAO-PETSc** used in quantum chemistry energy minimization
- **PDE-constrained optimization**
 - **Veltisto-PETSc** used in flow control application, to straighten out wingtip vortex by wing surface blowing and suction; performs full optimization in the time of just five N-S solves
- **“Best technical paper” at SC2002** went to our SciDAC colleagues at CMU (Ghattas, PI, speaking here Thursday morning):
 - Inverse wave propagation employed to infer hidden geometry

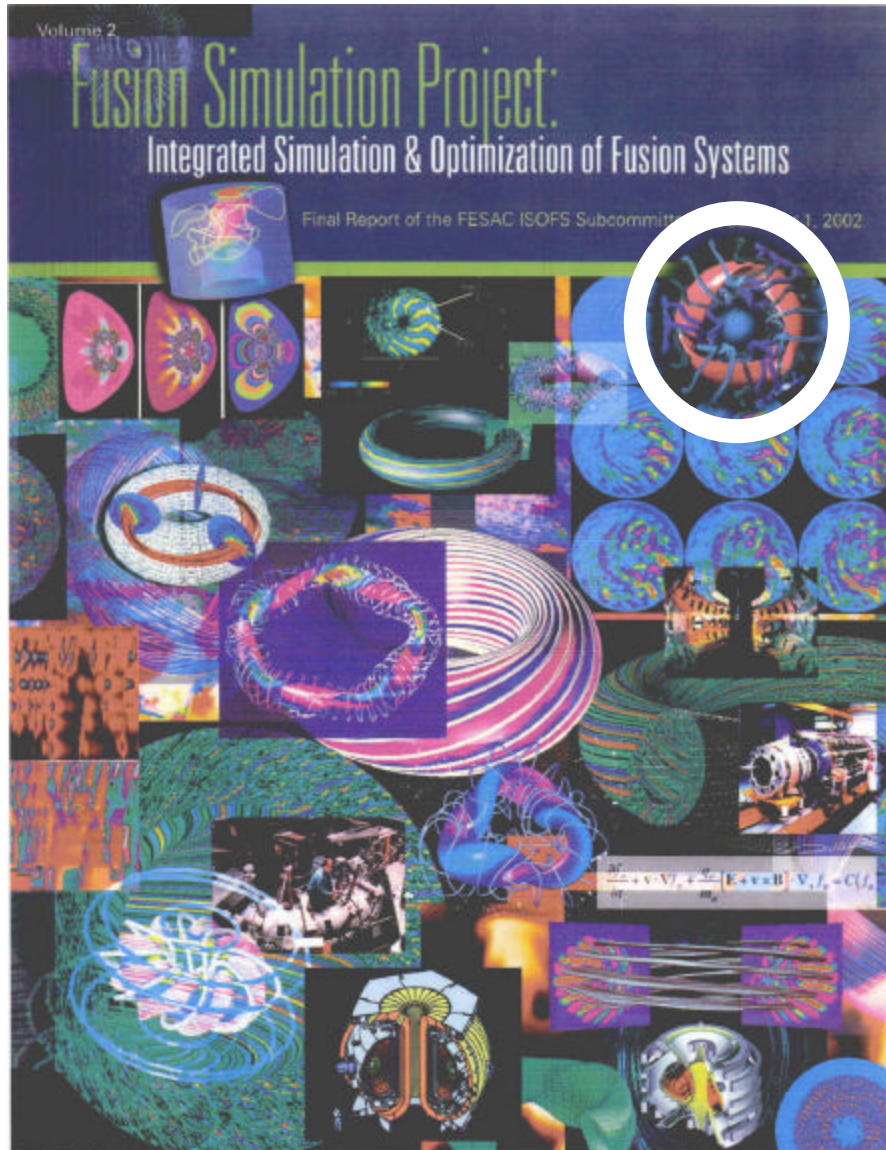


Motivating examples for simulation-based optimization

- Stellarator design
- Materials design and molecular structure determination
- Source inversion



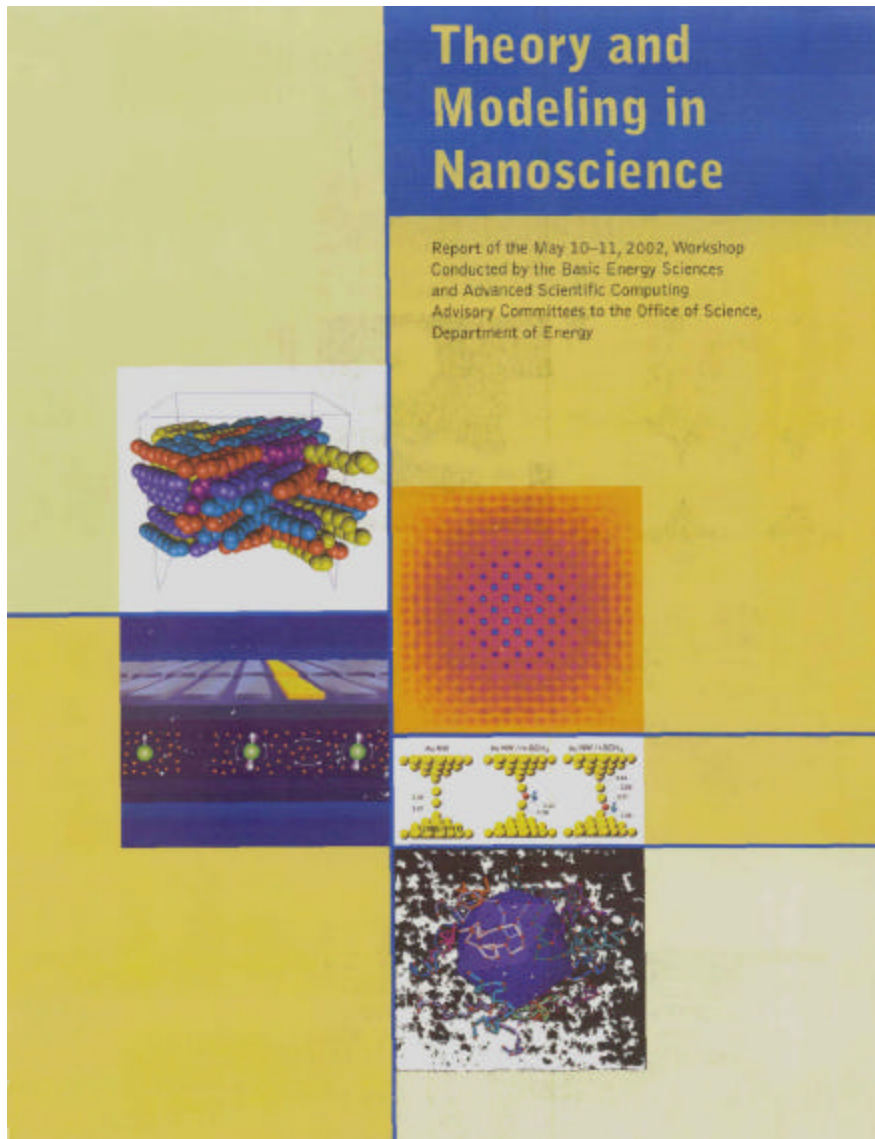
Stellarator design



- Dec 2002 report to DOE
- Multiphysics simulation for shape optimization of magnetic confinement fusion devices featured as a *key technology* for fusion energy
- Currently genetic algorithms used to optimize single-physics subsystems, e.g., magnetic flux surfaces of the plasma, magnetic coil shape of the controls – many analysis runs



Molecular-level materials design



- Jul 2002 report to DOE
- Optimization methods frequently invoked by nanoscientists as a *pressing need* – though there is much work ahead to make this concrete mathematically, except for ...
- ... Multilevel optimization for energy minimization in molecular structure determination – well along in protein folding, etc.



Source inversion

SAND REPORT

SAND2002-3198
Unlimited Release
October 2002

Large Scale Non-Linear Programming for PDE Constrained Optimization.

Bart van Bloemen Waanders, Roscoe Bartlett, Kevin Long, Paul Boggs,
Andrew Salinger
Sandia National Laboratories

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 under Contract DE-AC04-84A185000.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories

- **Oct 2002 Sandia LDRD final report**
- **Model source inversion problem solved by multilevel optimization, using Sundance/rSQP++**
- **Simultaneous Analysis and Design (SAND) framework exploited to save an order of magnitude in execution time, relative to blackbox methods**



Constrained optimization w/Lagrangian

- Consider Newton's method for solving the nonlinear rootfinding problem derived from the necessary conditions for constrained optimization
- Constraint $c(x, u) = 0 ; x \in \mathfrak{R}^N ; u \in \mathfrak{R}^M ; c \in \mathfrak{R}^N$
- Objective $\min_u f(x, u) ; f \in \mathfrak{R}$
- Lagrangian $f(x, u) + \mathbf{I}^T c(x, u) ; \mathbf{I} \in \mathfrak{R}^N$
- Form the gradient of the Lagrangian with respect to each of x , u , and λ :

$$f_x(x, u) + \mathbf{I}^T c_x(x, u) = 0$$

$$f_u(x, u) + \mathbf{I}^T c_u(x, u) = 0$$

$$c(x, u) = 0$$



Newton on first-order conditions

- Equality constrained optimization leads to the KKT system for states x , designs u , and multipliers l

$$\begin{bmatrix} W_{xx} & W_{ux}^T & J_x^T \\ W_{ux} & W_{uu} & J_u^T \\ J_x & J_u & 0 \end{bmatrix} \begin{bmatrix} dx \\ du \\ dl \end{bmatrix} = - \begin{bmatrix} g_x \\ g_u \\ c \end{bmatrix}$$

- Newton Reduced SQP solves the Schur complement system $H du = g$, where H is the reduced Hessian

$$H = W_{uu} - J_u^T J_x^{-T} W_{ux}^T - (J_u^T J_x^{-T} W_{xx} - W_{ux}) J_x^{-1} J_u$$

$$g = -g_u + J_u^T J_x^{-T} g_x - (J_u^T J_x^{-T} W_{xx} - W_{ux}) J_x^{-1} c$$

- Then

$$J_x dx = -c - J_u du$$

$$J_x^T dl = -g_x - W_{xx} dx - W_{ux}^T du$$



RSQP when constraints are PDEs

- **Problems**

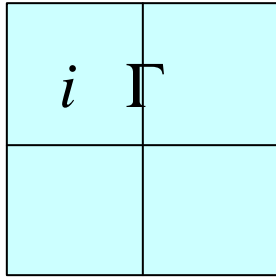
- ➡ ■ J_x is the Jacobian of a PDE \Rightarrow huge!
- ➡ ■ W_{ab} involve Hessians of objective and constraints \Rightarrow second derivatives and huge
- ➡ ■ H is unreasonable to form, store, or invert

- **Proposed solution: Schwarz inside Schur!**

- ➡ ■ form approximate inverse action of state Jacobian and its transpose in parallel by Schwarz/multilevel methods
- ➡ ■ form forward action of Hessians by automatic differentiation; exact action needed only on vectors (JFNK)
- ➡ ■ do not eliminate exactly; use Schur preconditioning on *full* system



Schur preconditioning from DD theory

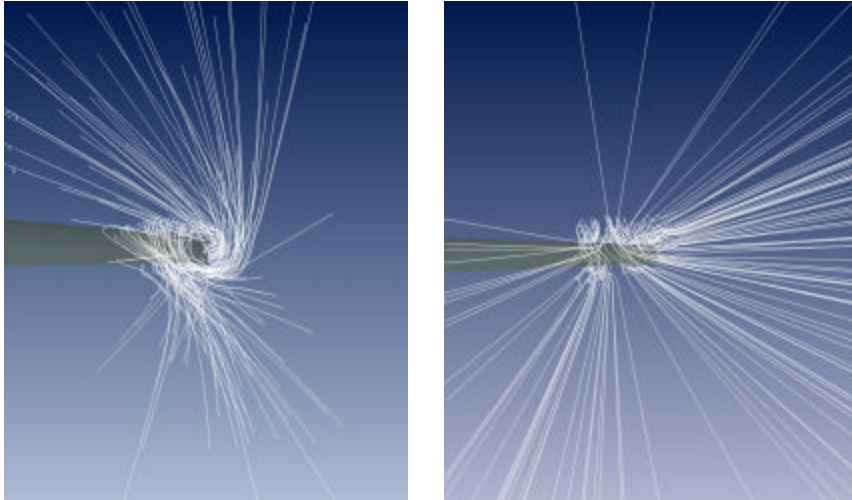
- Given a partition $\begin{bmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} u_i \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_i \\ f_\Gamma \end{bmatrix}$ 
- Condense:

$$S u_\Gamma = g \quad S \equiv A_{\Gamma\Gamma} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma} \quad g \equiv f_\Gamma - A_{\Gamma i} A_{ii}^{-1} f_i$$
- Let M be a good preconditioner for S
- Then $\begin{bmatrix} A_{ii} & 0 \\ A_{\Gamma i} & I \end{bmatrix} \begin{bmatrix} I & A_{ii}^{-1} A_{i\Gamma} \\ 0 & M \end{bmatrix}$ is a preconditioner for A
- Moreover, solves with A_{ii} may be approximate if all degrees of freedom are retained (e.g., a single V-cycle)
- Algebraic analogy from constrained optimization: “ i ” is state-like, “ Γ ” is decision-like

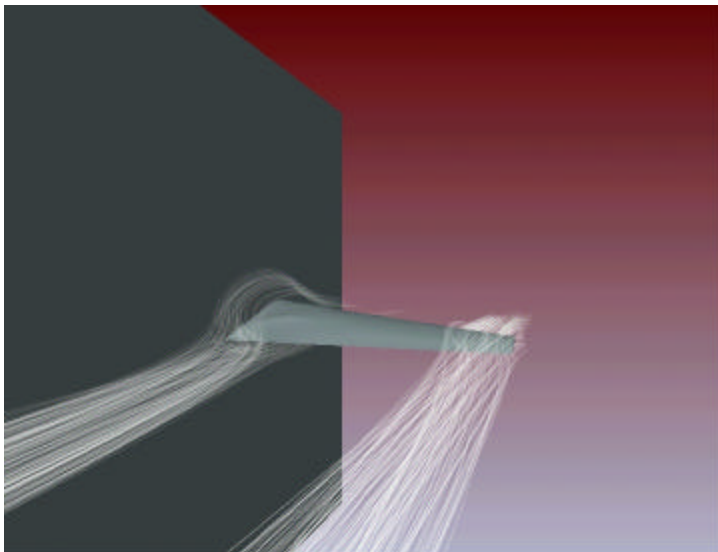


PDE-constrained Optimization

Lagrange-Newton-Krylov-Schur implemented in Veltisto/PETSc

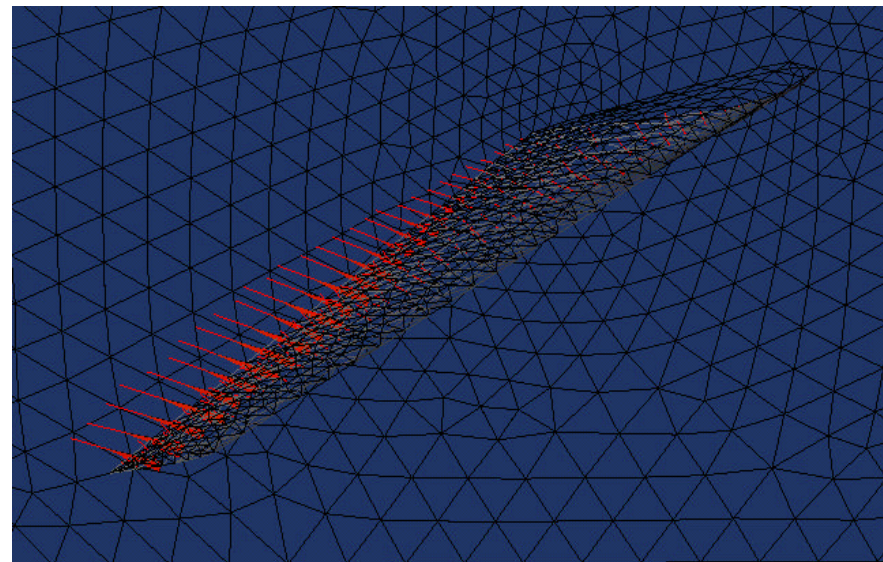


wing tip vortices, no control (l); optimal control (r)



- **Optimal control of laminar viscous flow**

- optimization variables are surface suction/injection
- objective is minimum drag
- 700,000 states; 4,000 controls
- 128 Cray T3E processors
- ~5 hrs for optimal solution (~1 hr for analysis)



optimal boundary controls shown as velocity vectors

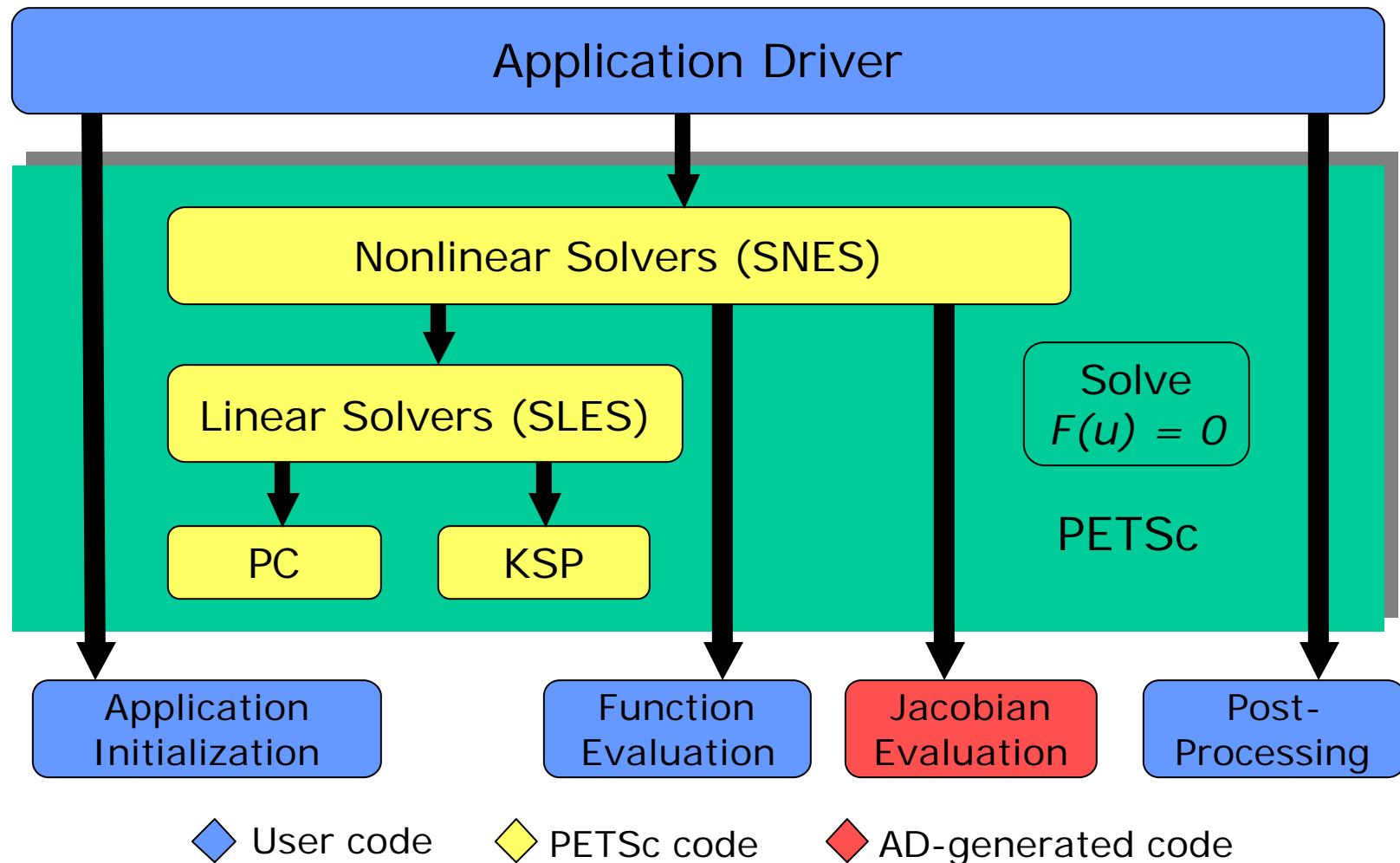
c/o G. Biros and O. Ghattas

www.cs.nyu.edu/~biros/veltisto/

Carnegie Mellon



Nonlinear PDE solution w/PETSc

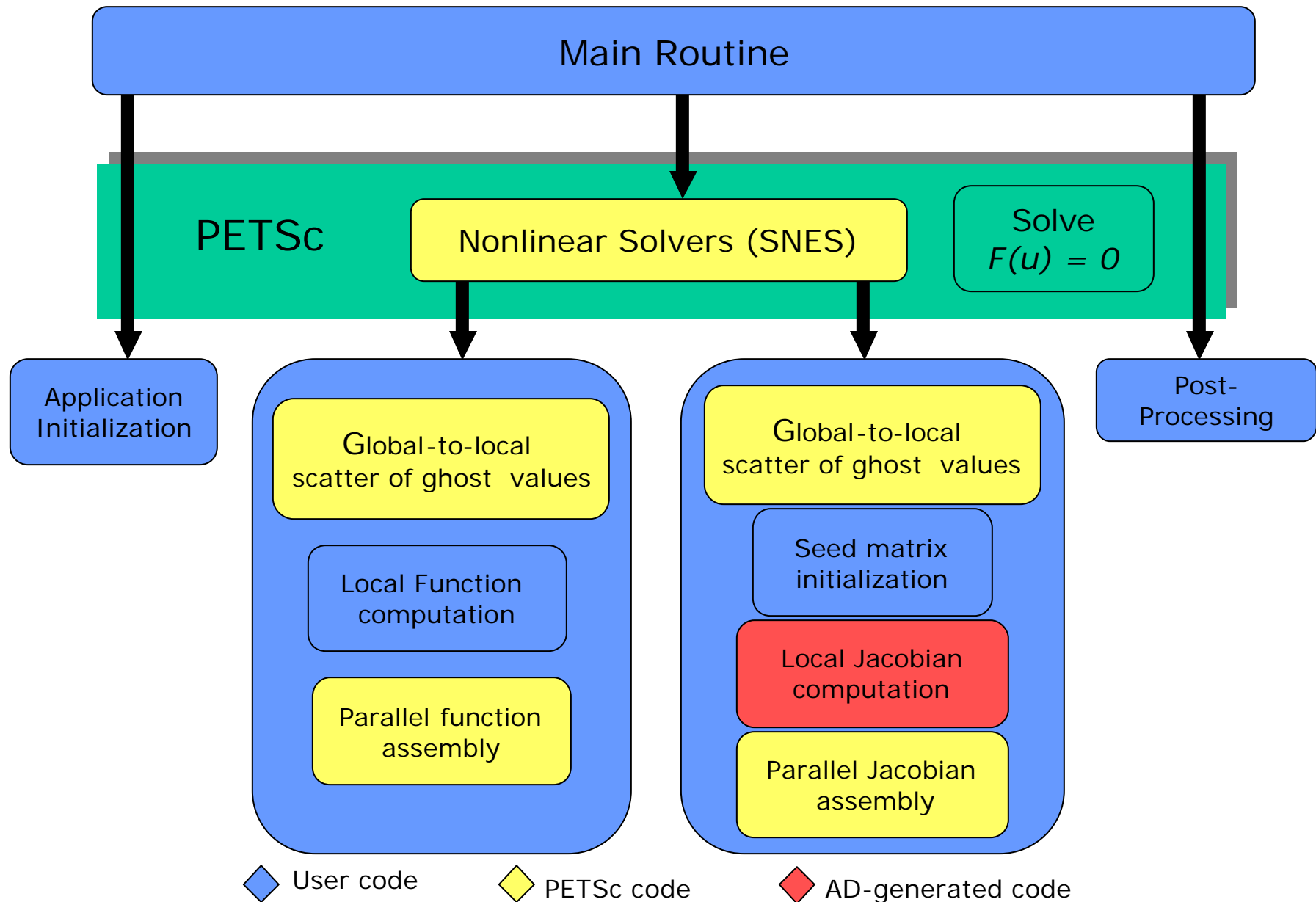


- **Automatic Differentiation (AD):** a technology for automatically augmenting computer programs, including arbitrarily complex simulations, with statements for the computation of derivatives, also known as sensitivities.

- **AD Collaborators:** P. Hovland and B. Norris (<http://www.mcs.anl.gov/autodiff>)

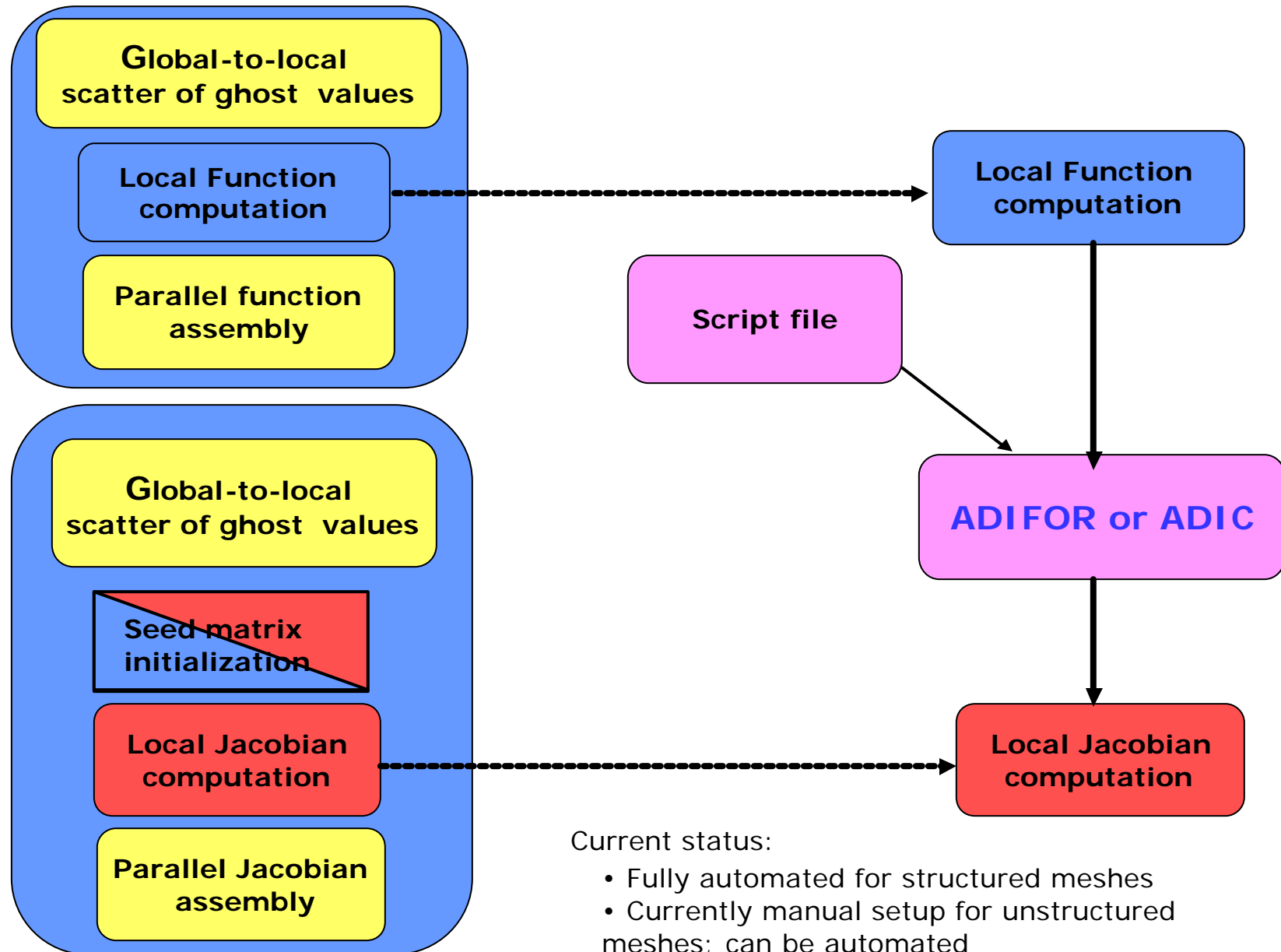


Zoom on user routine structure



Using AD with PETSc:

Creation of AD Jacobian from function



Parameter identification model

- **Nonlinear diffusion PDE BVP:** $\nabla \cdot (\mathbf{a}(x) T^b \nabla T) = 0$
- **Parameters to be identified:** $\mathbf{a}(x)$, b
- **Dirichlet conditions in x , homogeneous Neumann in all other dimensions (so solution has 1D character but arbitrarily large parallel test cases can be set up)**
- **Objective:** $\Phi = ||T(x) - \bar{T}(x)||^2$ where $\bar{T}(x)$ is synthetic data specified from *a priori* solution with given $\mathbf{a}(x)$ piecewise constant, $b=2.5$ (Brisk-Spitzer approximation for radiation diffusion)



Progress to date (Samyono thesis)

- Parallel implementation using PETSc's “shell preconditioner” functionality to build the block factored KKT preconditioner recursively
- Solution method: LNKS with Schwarz preconditioning of the PDE Jacobian blocks, ILU on Schwarz subdomains
- ADIC generates Jacobian blocks from user functions
- Newton-like convergence for PDE analysis
- Rougher, but monotone, convergence for parameter identification problems

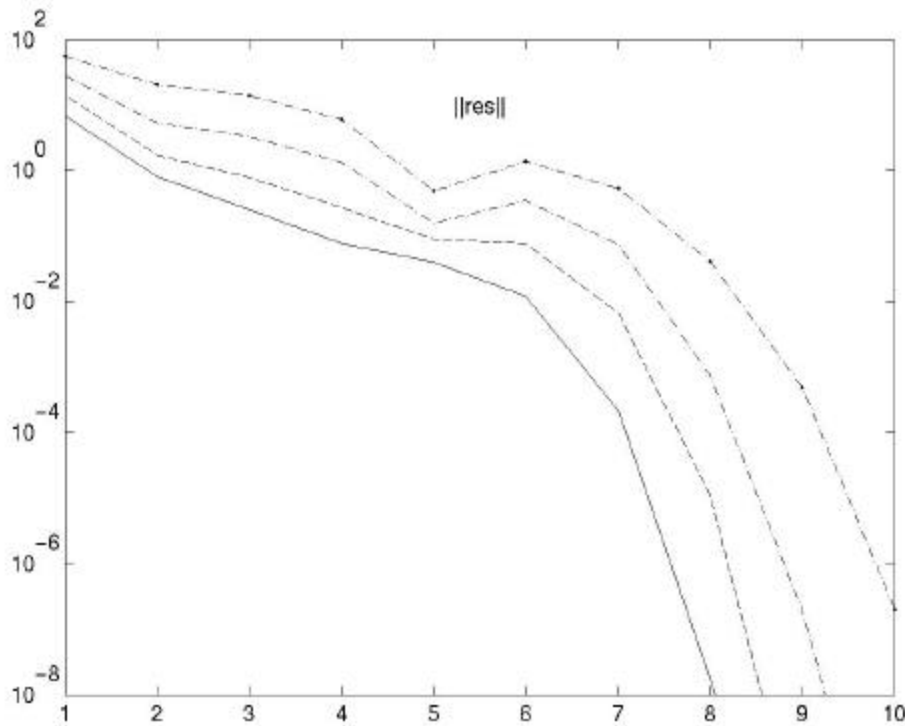


Implementation

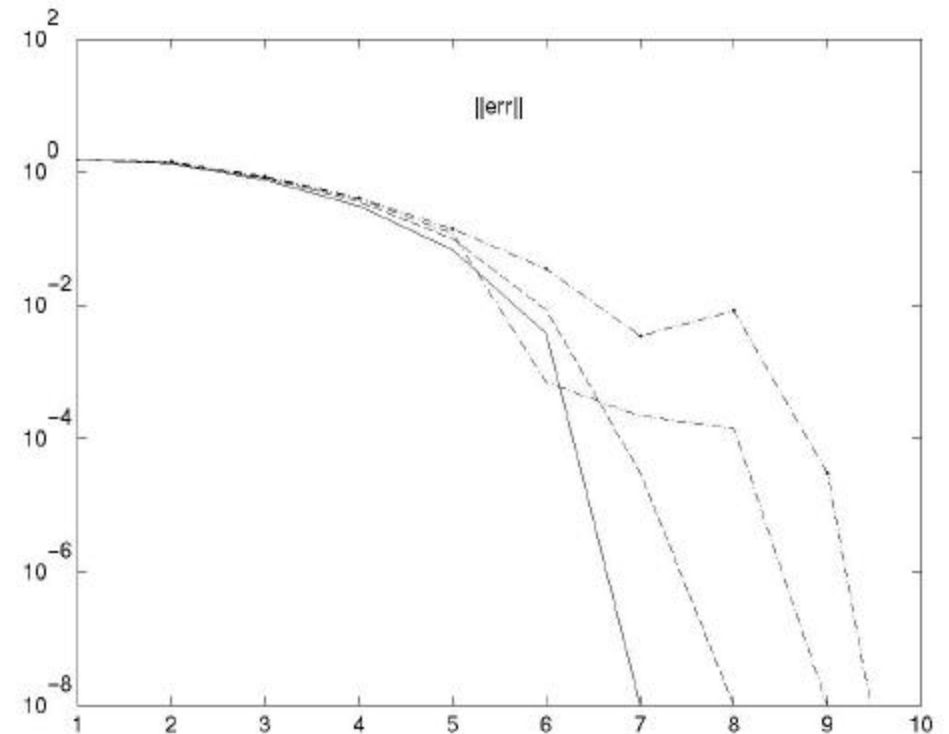
- PETSc's “shell preconditioner” functionality used to build the block factored KKT preconditioner recursively
- Solution method: LNKS with Schwarz preconditioning of the PDE Jacobian blocks, ILU on Schwarz subdomains
- MPI-based parallelization
- ADIC generates Jacobian blocks from user functions
- Illustrative results (next slide) fix $a(x)$ and identify exponent b *only*, while uniform mesh density is refined in 2D; have also identified $a(x)$ throughout full domain
- Newton-like, mesh-independent convergence for overall residual



Illustrative results



**2-norm of residual of full system
 $F(T(x), b, l(x))$ vs. iteration**



$\|b - b^*\|$ vs. iteration

[**solid**: 25x25 2Dmesh, **dash**: 50x50, **dot-dash**: 100x100, **dot-dot-dash**: 200x200]



Related URLs

- **Personal homepage: papers, talks, etc.**

<http://www.math.odu.edu/~keyes>

- **SciDAC initiative**

<http://www.science.doe.gov/scidac>

- **TOPS project**

<http://www.math.odu.edu/~keyes/scidac>

- **PETSc project**

<http://www.mcs.anl.gov/petsc>

- **Hypre project**

<http://www.llnl.gov/CASC/hypre>

- **ASCI platforms**

<http://www.llnl.gov/asci/platforms>



Bibliography

- *Jacobian-Free Newton-Krylov Methods: Approaches and Applications*, Knoll & Keyes, 2002, in review for J. Comp. Phys.
- *Nonlinearly Preconditioned Inexact Newton Algorithms*, Cai & Keyes, 2002, to appear in SIAM J. Sci. Comp.
- *High Performance Parallel Implicit CFD*, Gropp, Kaushik, Keyes & Smith, 2001, Parallel Computing 27:337-362
- *Four Horizons for Enhancing the Performance of Parallel Simulations based on Partial Differential Equations*, Keyes, 2000, Lect. Notes Comp. Sci., Springer, 1900:1-17
- *Globalized Newton-Krylov-Schwarz Algorithms and Software for Parallel CFD*, Gropp, Keyes, McInnes & Tidriri, 2000, Int. J. High Performance Computing Applications 14:102-136
- *Achieving High Sustained Performance in an Unstructured Mesh CFD Application*, Anderson, Gropp, Kaushik, Keyes & Smith, 1999, Proceedings of SC'99
- *Prospects for CFD on Petaflops Systems*, Keyes, Kaushik & Smith, 1999, in “Parallel Solution of Partial Differential Equations,” Springer, pp. 247-278
- *How Scalable is Domain Decomposition in Practice?*, Keyes, 1998, in “Proceedings of the 11th Intl. Conf. on Domain Decomposition Methods,” Domain Decomposition Press, pp. 286-297

