TOPS

Terascale Optimal PDE Simulations

http://www.tops-scidac.org

Advanced Solver Algorithms and Physics-based Preconditioners

SciDAC

Scientific Discovery through Advanced Computing

David E. Keyes, Columbia University

Recall Newton methods

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

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

  where $\delta u^k = u^{k+1} - u^k,\ k = 0, 1, 2, \ldots$
- **Neglecting higher-order terms, we get**

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

  where $J = F'(u^k)$ **is the Jacobian matrix, generally large, sparse, and ill-conditioned for PDEs**
- **In practice, require** $\| F(u^k) + J(u^k)\delta u^k \| < \varepsilon$
- **In practice, set** $u^{k+1} = u^k + \lambda \delta u^k$ **where** λ **is selected to minimize** $\| F(u^k + \lambda \delta u^k) \|$

Nonlinear Robustness

- **Problem:**
  - attempts to handle nonlinear problems with nonlinear implicit methods often encounter stagnation failure of Newton away from the neighborhood of the desired root
- **Algebraic solutions:**
  - linesearch and trust-region methods
  - "forcing terms"
- **Physics-based solutions:**
  - mesh sequencing
  - continuation (homotopy) methods for directly addressing this through the physics, e.g., pseudo-transient continuation
  - transform system to be solved so that neglected curvature terms of multivariate Taylor expansion truncated for Newton's method are smaller (nonlinear Schwarz)

Standard robustness features

- **PETSc** contains in its nonlinear solver library some standard algebraic robustness devices for nonlinear rootfinding from Dennis & Schnabel, 1983
- Line search
  - try to ensure that $F(u)$ is strictly monotonically decreasing
  - parameterize reduction of $|F(u+\lambda\delta u)|$ along Newton step δu
  - solve scalar minimization problem for λ
- Trust region
  - define a region about the current iterate within which we trust a model of the residual
  - approximately minimize the model of the residual within the region (again with low-dimensional parameterization of convex combination of descent direction and Newton direction)
  - shrink or expand trust region according to history

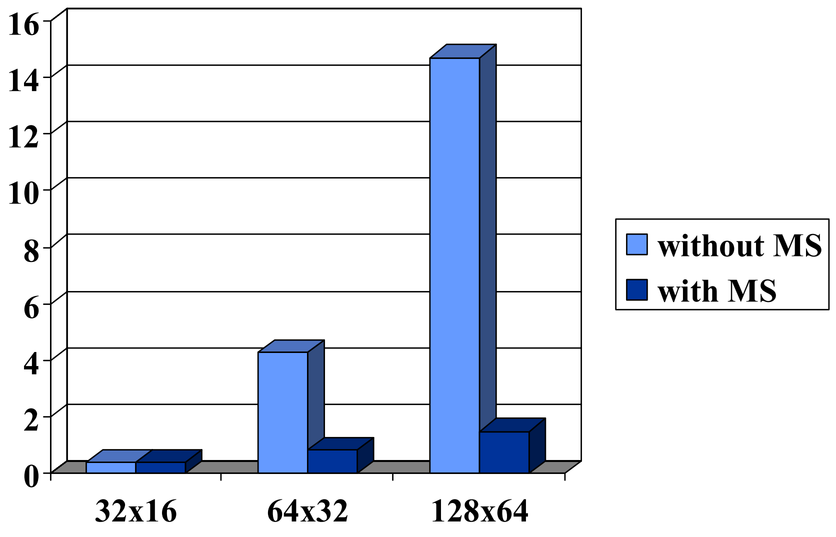
Standard robustness features

- **PETSc** contains in its nonlinear solver library standard algebraic robustness devices for nonlinear rootfinding from Eisenstat & Walker (1996)
  - EW'96 contains three heuristics for the accuracy with which a Newton step should be solved
  - relies intrinsically on iterative solution of the Newton correction equation
  - tolerance for linear residual ("forcing factor") computed based on norms easily obtained as by-products of the rootfinding computation – little additional expense
  - tolerance tightens dynamically as residual norm decreases during the computation
  - "oversolving" not only wastes execution time, but may be less robust, since early Newton directions are not reliable

Mesh sequencing

- Technique for robustifying nonlinear rootfinding for problems based on continuum approximation
- Relies on several levels of refinement from coarse to fine
- Theory exists showing (for nonlinear elliptic problems) that, asymptotically, the root on a coarser mesh, appropriately interpolated onto a finer mesh, lies in the domain of convergence of Newton's method on the finer grid

Execution times for 8-equation 2D BVP steady-state coupled edge plasma/Navier-Stokes problem, from Knoll & McHugh (SIAM J. Sci. Comput., 1999)



Time-implicit Newton-Krylov-Schwarz

For accommodation of unsteady problems, and nonlinear robustness in steady ones, NKS iteration is wrapped in time-stepping:

Pseudo-time loop

NKS loop

```
for (l = 0; l < n_time; l++) {
   select time step
   for (k = 0; k < n_Newton; k++) {
      compute nonlinear residual and Jacobian
      for (j = 0; j < n_Krylov; j++) {
         forall (i = 0; i < n_Precon ; i++) {
               solve subdomain problems concurrently
         } // End of loop over subdomains
         perform Jacobian-vector product
         enforce Krylov basis conditions
         update optimal coefficients
         check linear convergence
      } // End of linear solver
      perform DAXPY update
      check nonlinear convergence
   } // End of nonlinear loop
} // End of time-step loop
```

Pseudo-transient continuation (Ψtc)

- **Solve *F(u)=0* through a series of problems derived from method of lines model**

$$f^{\ell}(u) = \frac{u - u^{\ell-1}}{\tau^{\ell}} + F(u) = 0, \;\; \ell = 1,2,\cdots \qquad (*)$$

- **τ^{ℓ} is advanced from $\tau^{0} << 1$ to ∞ as $\ell \rightarrow \infty$ so that u^{ℓ} approaches the root**
- **With initial iterate for u^{ℓ} as $u^{\ell-1}$, the first Newton correction for (*) is**

$$u^{\ell} = u^{\ell-1} - [\frac{1}{\tau^{\ell}} I + F'(u^{\ell-1})]^{-1} F(u^{\ell-1})$$

- **Note that ||*F(u)*|| can climb hills during Ψtc**
- **Can subcycle inside physical timestepping**

Algorithmic tuning - continuation parameters

- "Switched Evolution-Relaxation" (SER) heuristic

$$N_{CFL}^{l} = N_{CFL}^{0} \left(\frac{\left\| f(u^{0}) \right\|}{\left\| f(u^{l-1}) \right\|} \right)^{p}$$

- Analysis in SIAM papers by Kelley & Keyes (1999 for parabolized, 2002 for mixed elliptic/parabolized)
- Parameters of interest:
  - initial CFL number
  - exponent in the Power Law
    - = 1 normally
    - > 1 for first-order discretization (1.5)
    - < 1 at outset of second-order discretization (0.75)
  - switch-over ratio between first-order and second-order

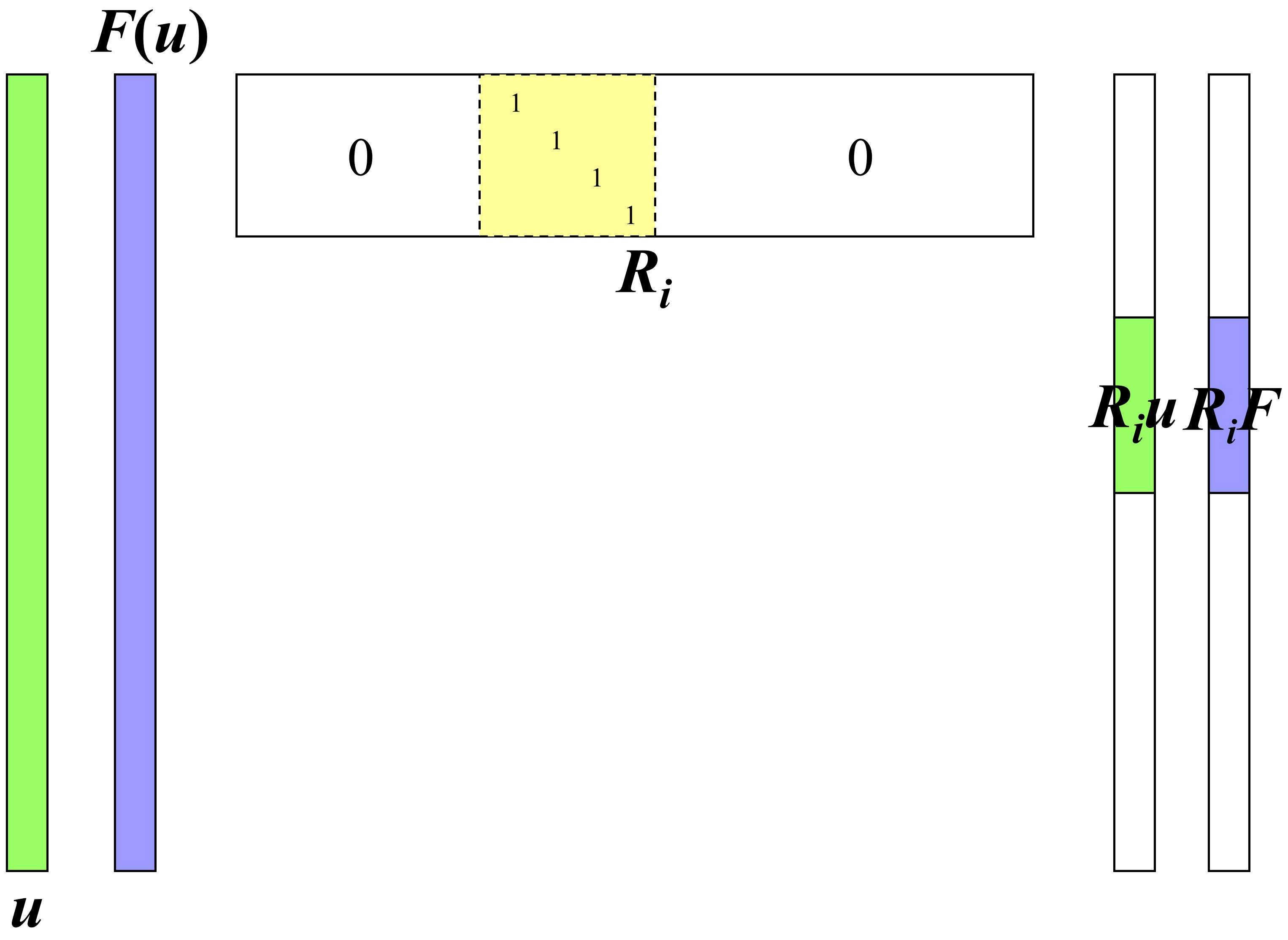
Nonlinear Schwarz preconditioning

- Nonlinear Schwarz has Newton both *inside* and *outside* and is fundamentally Jacobian-free
- It replaces $F(u) = 0$ with a new nonlinear system possessing the same root, $\Phi(u) = 0$
- Define a correction $\delta_i(u)$ to the i^{th} partition (e.g., subdomain) of the solution vector by solving the following local nonlinear system:

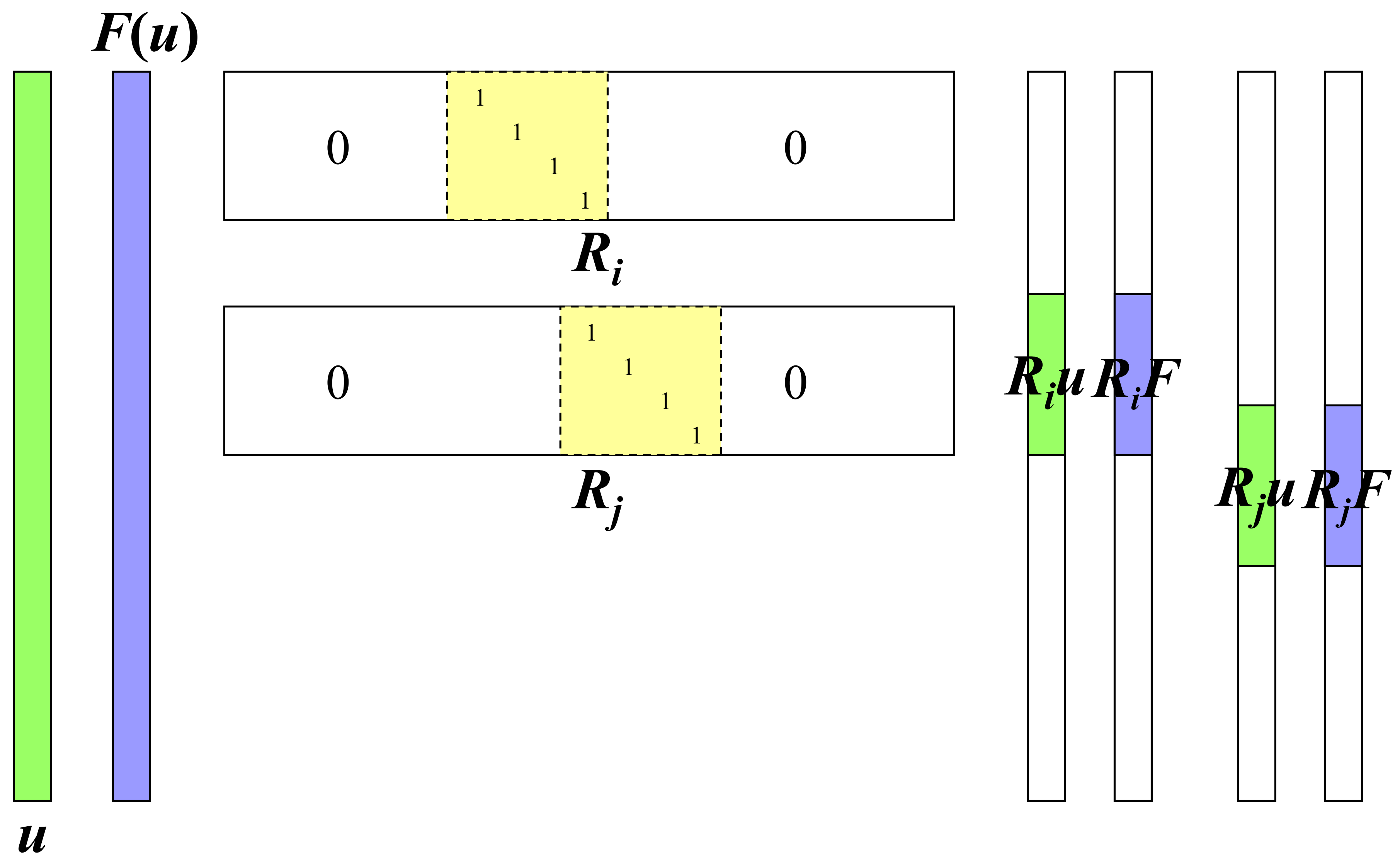
$$R_i F(u + \delta_i(u)) = 0$$

  where $\delta_i(u) \in \Re^n$ is nonzero only in the components of the i^{th} partition
- Then sum the corrections: $\Phi(u) = \sum_i \delta_i(u)$

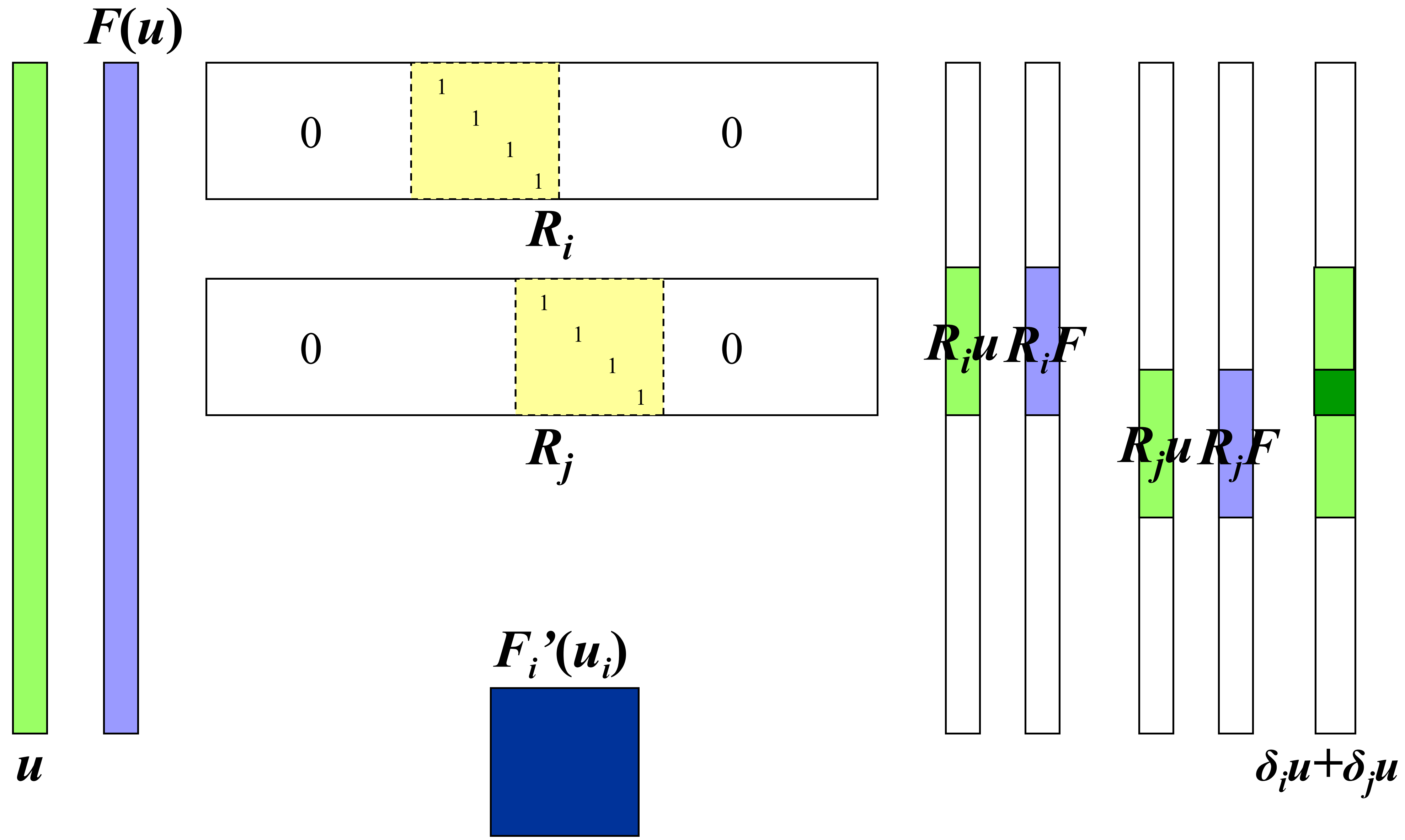
Nonlinear Schwarz – picture



Nonlinear Schwarz – picture



Nonlinear Schwarz – picture



Nonlinear Schwarz, cont.

- **It is simple to prove that if the Jacobian of $F(u)$ is nonsingular in a neighborhood of the desired root then $\Phi(u)=0$ and $F(u)=0$ have the same unique root**
- **To lead to a Jacobian-free Newton-Krylov algorithm we need to be able to evaluate for any $u, v \in \Re^n$:**
  - **The residual** $\Phi(u) = \sum_i \delta_i(u)$
  - **The Jacobian-vector product** $\Phi(u)' v$
- **Remarkably, (Cai-Keyes, 2000) it can be shown that**

$$\Phi'(u)v \approx \sum_i (R_i^T J_i^{-1} R_i) Jv$$

where $J = F'(u)$ **and** $J_i = R_i J R_i^T$

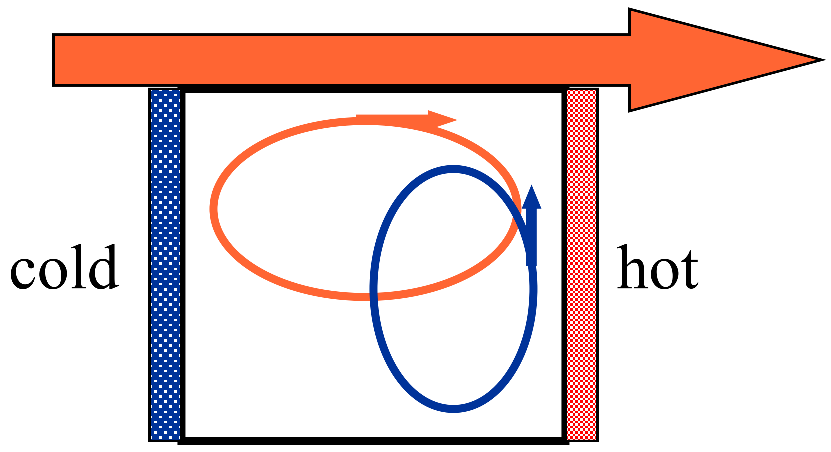
- **All required actions are available in terms of** $F(u)$ **!**

Nonlinear Schwarz, cont.

Discussion:

- **After the linear version of additive Schwarz was invented, it was realized that it was algebraically in the same class of methods as additive multigrid, though linear MG is usually done multiplicatively**
- **After nonlinear Schwarz was invented, it was realized that it was algebraically in the same class of methods as nonlinear multigrid (full approximation scheme MG), though nonlinear multigrid is usually done multiplicatively**

Driven cavity in velocity-vorticity coords



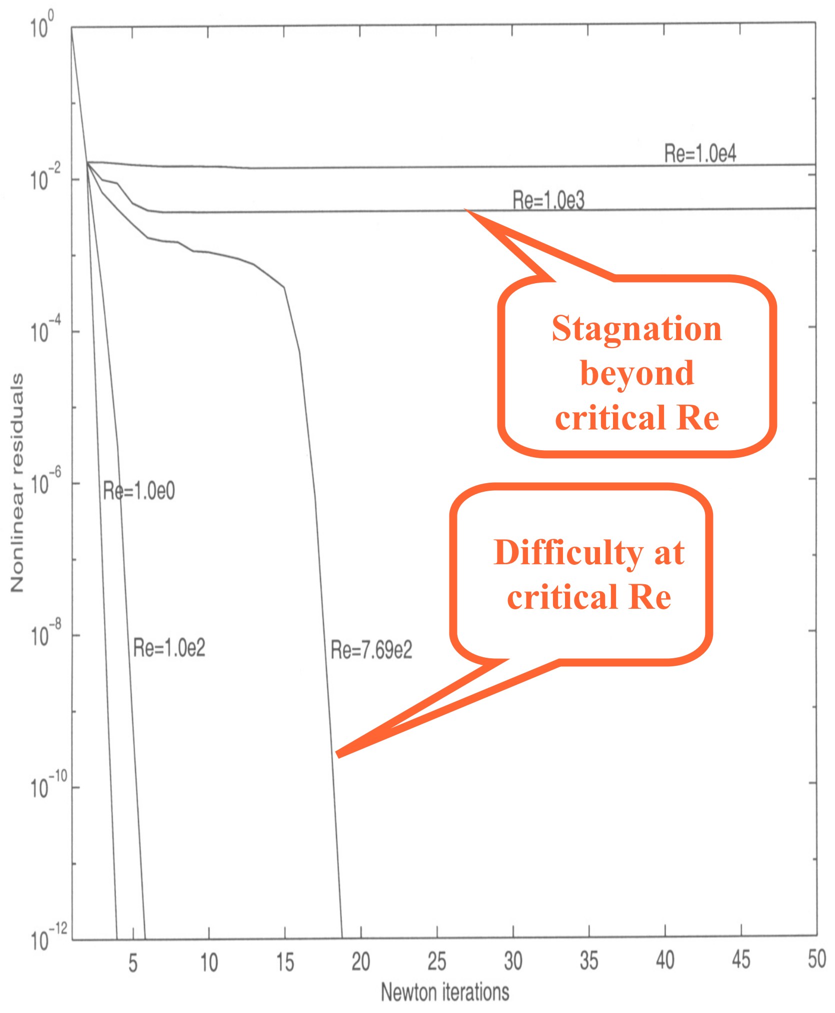
x-velocity $$-\nabla^2 u - \frac{\partial \omega}{\partial y} = 0$$

y-velocity $$-\nabla^2 v + \frac{\partial \omega}{\partial x} = 0$$

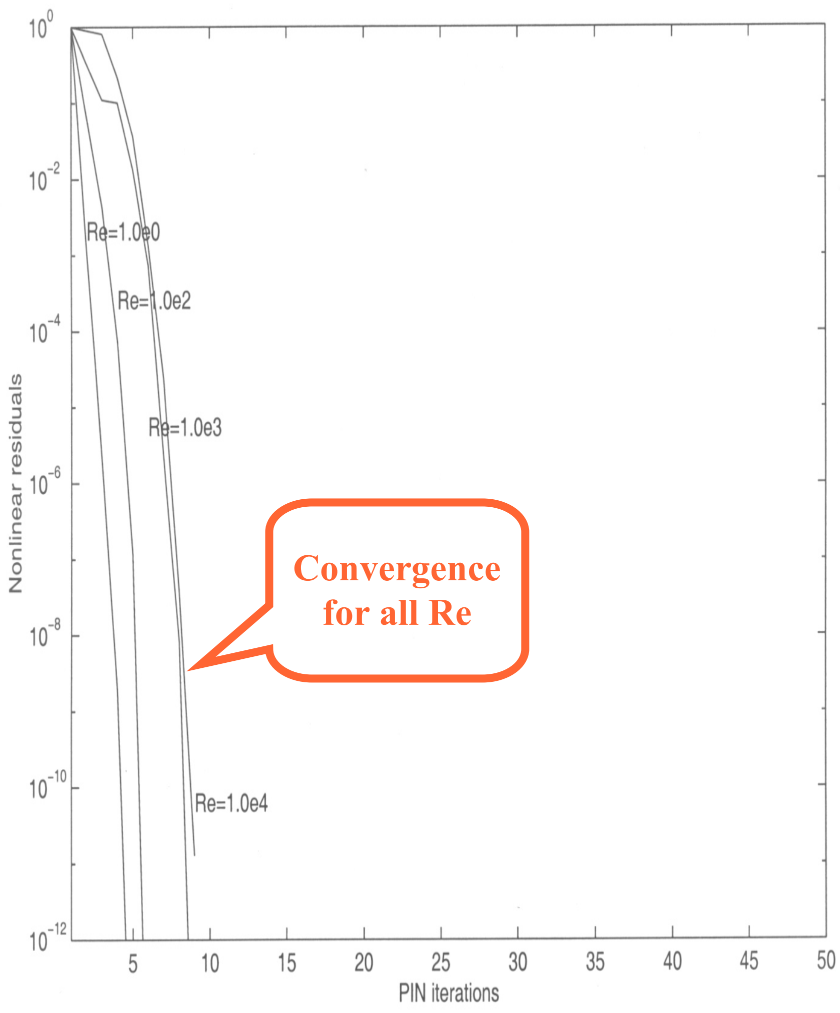
vorticity $$-\nabla^2 \omega + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} - \mathrm{Gr}\frac{\partial T}{\partial x} = 0$$

internal energy $$-\nabla^2 T + \mathrm{Pr}\left(u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y}\right) = 0$$

Experimental example of nonlinear Schwarz



Newton's method



Additive Schwarz Preconditioned Inexact Newton (ASPIN)

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**

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

(where ε 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:

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

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 lower order discretization

- Orszag popularized the use of linear finite element discretizations as preconditioners for high-order spectral element discretizations in the 1970s; both approach the same continuous operator
- It is common in CFD to employ first-order upwinded convective operators as approximate inversions for higher-order operators:
  - better factorization stability
  - smaller matrix bandwidth and complexity
- With Jacobian-free NK, we can have the best of both worlds – a stable factorization/cheap solve *and* a true Jacobian step

Using Jacobian with lagged terms

- Newton-chord methods (e.g., papers by Smooke et al.) “freeze” the Jacobian matrices:
  - saves Jacobian evaluation and factorization, which can be up to 90% of the running time of the code in some apps
  - however, nonlinear convergence degrades to linear rate
- In Jacobian-free NK, we can “freeze” some or all of the terms in the Jacobian preconditioner, while always accessing the action of the true Jacobian for the Krylov matrix-vector multiply:
  - still saves Jacobian work
  - maintains asymptotically quadratic rate for nonlinear convergence
- See (Knoll-Keyes ’03) for example with coupled edge plasma and Navier-Stokes, showing five-fold improvement over full Newton with constantly refreshed Jacobian on LHS, versus JFNK with preconditioner refreshed once each ten timesteps

Using Jacobian with lower precision elements

- Memory bandwidth is the critical architectural parameter for sparse linear algebra computations
- Storing the preconditioner elements in single precision effectively doubles memory bandwidth (and potentially halves runtime) for this critical phase
- We still form the Jacobian-vector product with full precision and "zero-pad" the preconditioner elements back to full length in the arithmetic unit, so the numerical quality of the Krylov subspace does not degrade

Memory BW bottleneck revealed via precision reduction

Execution times for unstructured NKS Euler Simulation on Origin 2000: double precision matrices versus single precision preconditioner

Number of Processors	Computational Phase			
	Linear Solve		Overall	
	Double	Single	Double	Single
16	223s	136s	746s	657s
32	117s	67s	373s	331s
64	60s	34s	205s	181s
120	31s	16s	122s	106s

Note that times are nearly halved, along with precision, for the BW-limited linear solve phase, indicating that the BW can be at least doubled before hitting the next bottleneck!

Using Jacobian of related discretization

- To precondition a variable coefficient operator, such as $\nabla\cdot(\alpha\nabla\,\bullet)$, use $\bar{\alpha}\nabla^2$, 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

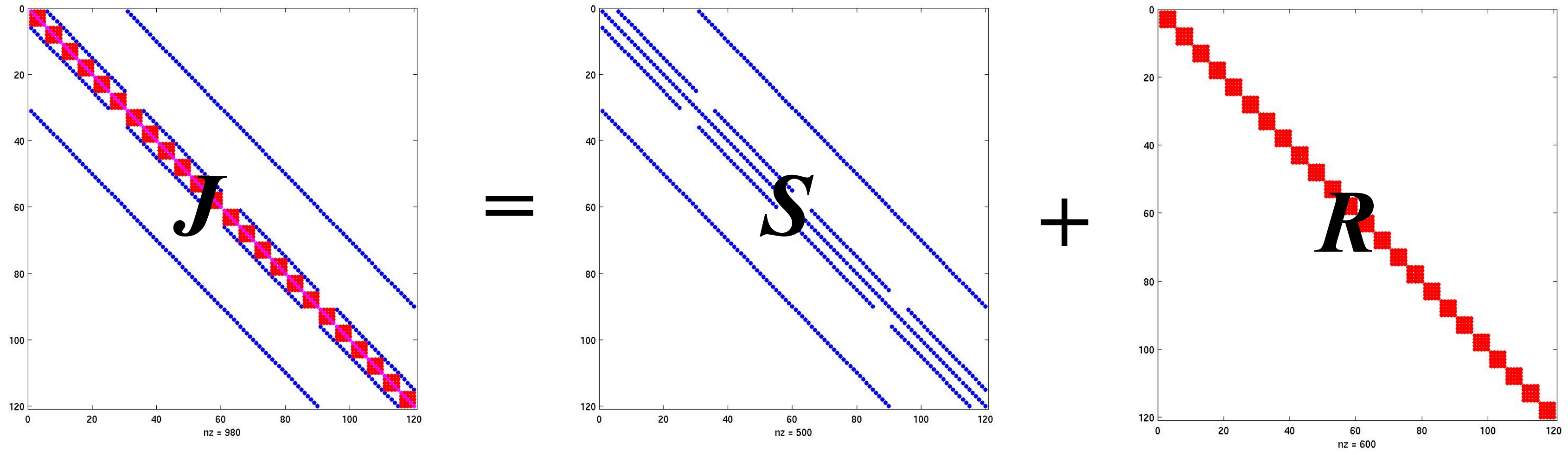
- Subcomponents of a PDE operator often have special structure that can be exploited if they are treated separately
- Algebraically, this is just a generalization of Schwarz, by term instead of by subdomain
- Suppose $J = \boxed{\tau^{-1}I + S + R}$ and a preconditioner is to be constructed, where $I + \tau S$ and $I + \tau R$ are each "easy" to invert
- Form a preconditioned vector from u as follows:

$$(\tau^{-1}I + R)^{-1}(I + \tau S)^{-1}u$$

- Equivalent to replacing J with $\boxed{\tau^{-1}I + S + R} + \tau SR$
- 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:**

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

- **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**

$$F(u^k) \mapsto \delta u^k$$

- **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 (*)**

$$\frac{\partial}{\partial t}\times \qquad \frac{\partial \phi}{\partial t}+\frac{\partial (u\phi)}{\partial x}=0$$

- **Momentum (**)**

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

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

$$\frac{\partial^2\phi}{\partial t^2}-g\phi\frac{\partial^2\phi}{\partial x^2}=other \quad terms$$

1D shallow water equations, cont.

- **Wave equation for geopotential:**

$$\frac{\partial^2 \phi}{\partial t^2} - g\phi \frac{\partial^2 \phi}{\partial x^2} = other \quad terms$$

- **Gravity wave speed** $\sqrt{g\phi}$
- **Typically** $\sqrt{g\phi} >> u$ **, 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 (*)**

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

- **Momentum (**)**

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

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

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

where

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

1D shallow water equations, cont.

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

$$(u\phi)^{n+1} = -\tau g \phi^{n} \frac{\partial \phi^{n+1}}{\partial x} + S^{n}$$

- *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:**

$$R\_\phi \equiv \frac{\boxed{\phi^{n+1}} - \phi^{n}}{\tau} + \frac{\partial \boxed{(u\phi)^{n+1}}}{\partial x}$$

- **Define momentum residual for each timestep:**

$$R\_u\phi \equiv \frac{\boxed{(u\phi)^{n+1}} - (u\phi)^{n}}{\tau} + \frac{\partial (u^2\phi)^{n}}{\partial x} + g\phi^{n} \frac{\boxed{\partial \phi^{n+1}}}{\partial x}$$

- **Continuity delta-form (*):**

$$\frac{\boxed{\delta\phi}}{\tau} + \frac{\partial [\boxed{\delta(u\phi)}]}{\partial x} = -R\_\phi$$

- **Momentum delta form (**):**

$$\frac{\boxed{\delta(u\phi)}}{\tau} + g\phi^{n} \frac{\partial [\boxed{\delta\phi}]}{\partial x} = -R\_u\phi$$

1D Shallow water preconditioning, cont.

- **Solving (**) for $\delta(u\phi)$ and substituting into (*),**

$$\delta\phi - g\tau^2 \frac{\partial}{\partial x}\left(\phi^n \frac{\partial[\delta\phi]}{\partial x}\right) = -R\_\phi + \tau^2 \frac{\partial}{\partial x}(R\_u\phi)$$

- **After this parabolic equation is solved for $\delta\phi$, we have**

$$\delta(u\phi) = -\tau g \phi^n \frac{\partial[\delta\phi]}{\partial x} - R\_u\phi$$

- **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 (2003)
- **PETSc**'s "shell preconditioner" is ideal for inserting physics-based preconditioners, and **PETSc**'s solvers underneath are ideal building blocks

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

$\Rightarrow$ Tools for PDE solution should also support these related desires

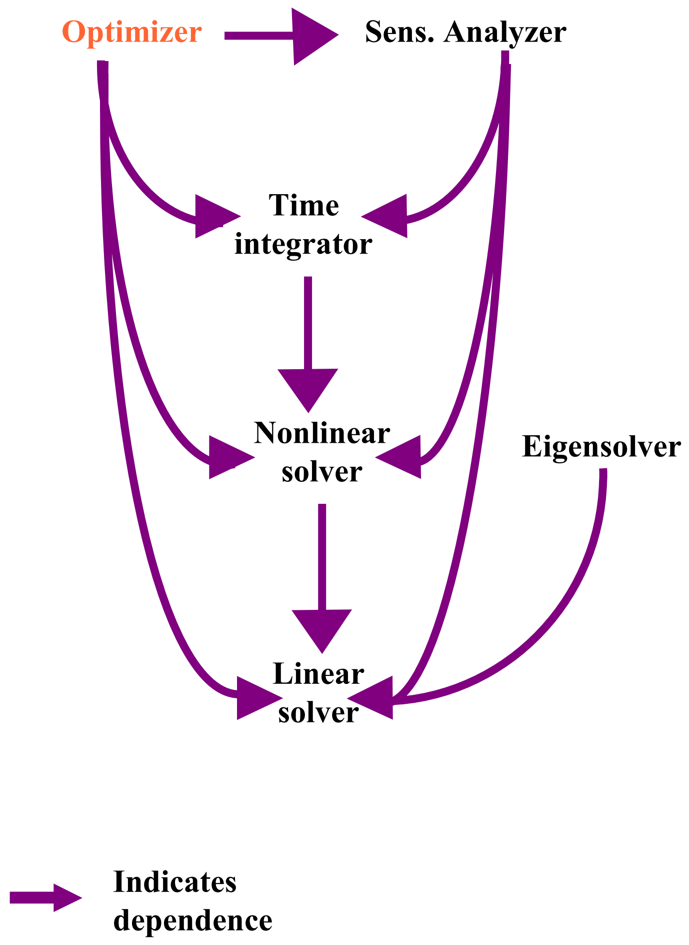
PDE-constrained optimization

- **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

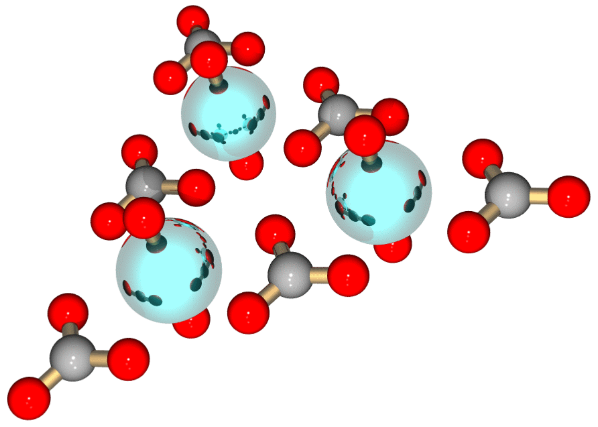
$$\min_{u} \phi(x,u)\ s.t.\ F(x,u)=0,\ u\geq 0$$

- 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"

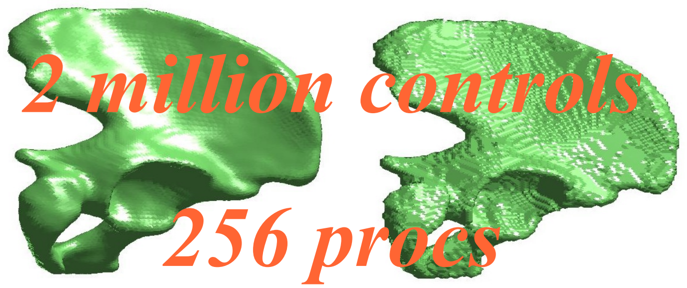


Recent optimization progress (recap)

- 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 sunction; performs full optimization in the time of just five N-S solves
- "Best technical paper" at SC2002 went to our SciDAC colleagues at CMU:
  - Inverse wave propagation employed to infer hidden geometry



4000 controls
128 procs



Constrained optimization w/Lagrangian

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

$$f_x(x,u)+\lambda^T c_x(x,u)=0$$
$$f_u(x,u)+\lambda^T c_u(x,u)=0$$
$$c(x,u)=0$$

Newton on first-order conditions

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

$$\begin{bmatrix} W_{xx} & W_{ux}^T & J_x^T \\ W_{ux} & W_{uu} & J_u^T \\ J_x & J_u & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta\lambda \end{bmatrix} = - \begin{bmatrix} g_x \\ g_u \\ c \end{bmatrix}$$

- Newton Reduced SQP solves the Schur complement system $H\,\delta u = g$, 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

$$J_x \delta x = -c - J_u \delta u$$

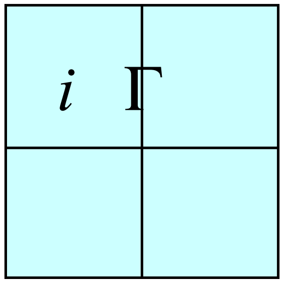
$$J_x^T \delta\lambda = -g_x - W_{xx}\delta x - W_{ux}^T \delta u$$

RSQP when constraints are PDEs

- **Problems**
  - J_x **is the Jacobian of a PDE $\Rightarrow$ huge!**
  - $W_{\alpha\beta}$ **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** $\begin{bmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma} \end{bmatrix}\begin{bmatrix} u_i \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_i \\ f_\Gamma \end{bmatrix}$



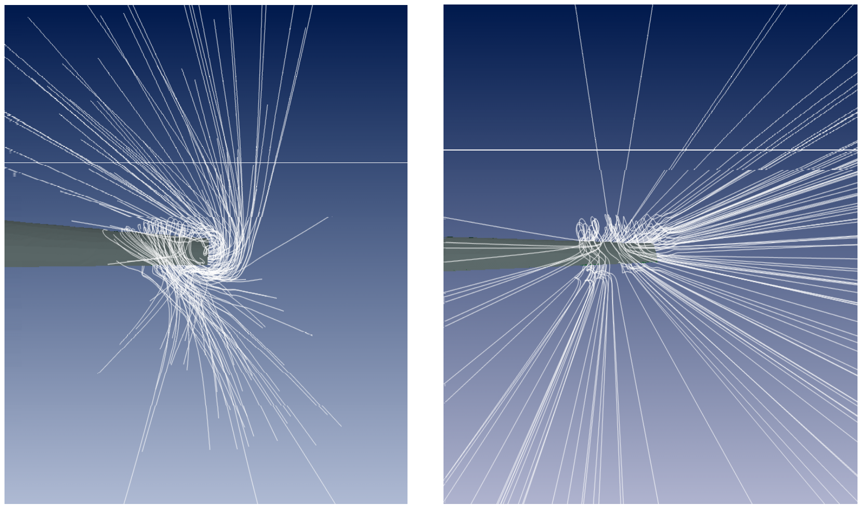
- **Condense:**

$$Su_\Gamma = g \qquad S \equiv A_{\Gamma\Gamma} - A_{\Gamma i}A_{ii}^{-1}A_{i\Gamma} \quad g \equiv f_\Gamma - A_{\Gamma i}A_{ii}^{-1}f_i$$

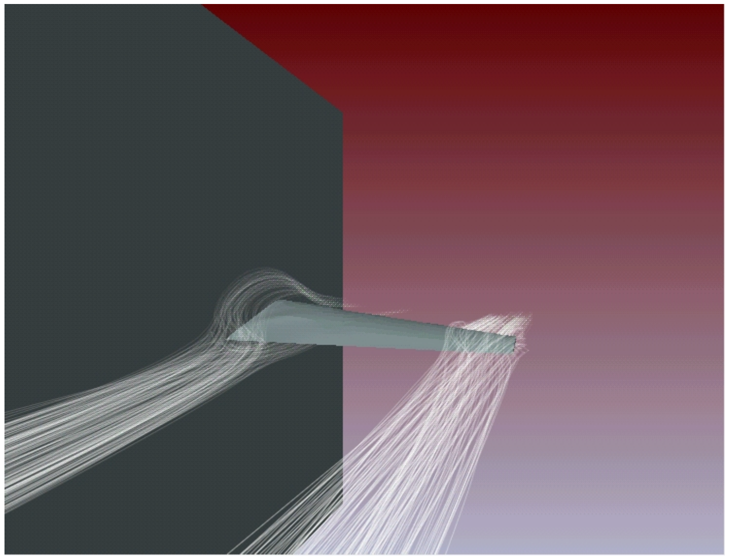
- **Let** M **be a good preconditioner for** S
- **Then** $\begin{bmatrix} A_{ii} & 0 \\ A_{\Gamma i} & I \end{bmatrix}\begin{bmatrix} I & A_{ii}^{-1}A_{i\Gamma} \\ 0 & M \end{bmatrix}$ **is a preconditioner for** A
- **Moreover, solves with** A_{ii} **may be approximate if all degrees of freedom are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

PDE-constrained Optimization

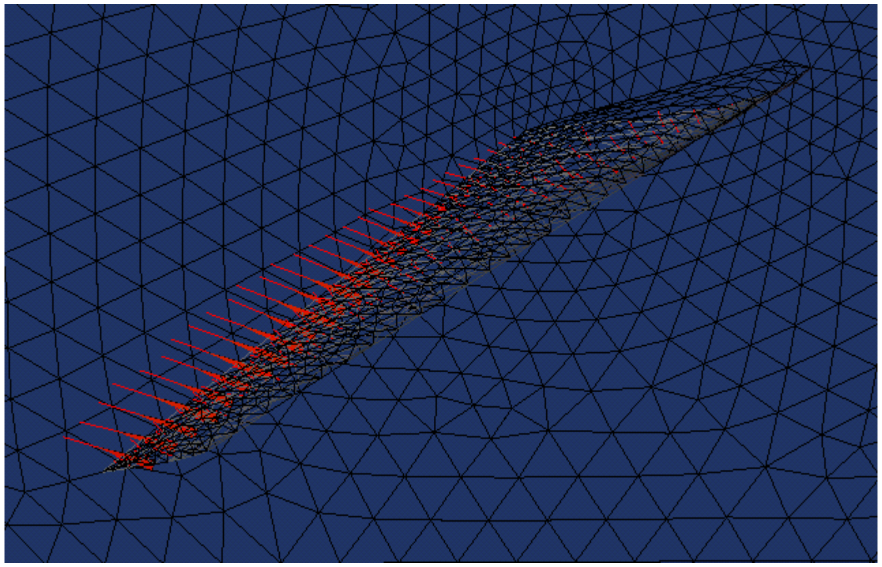
Lagrange-Newton-Krylov-Schur implemented in Veltisto/PETSc



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)**



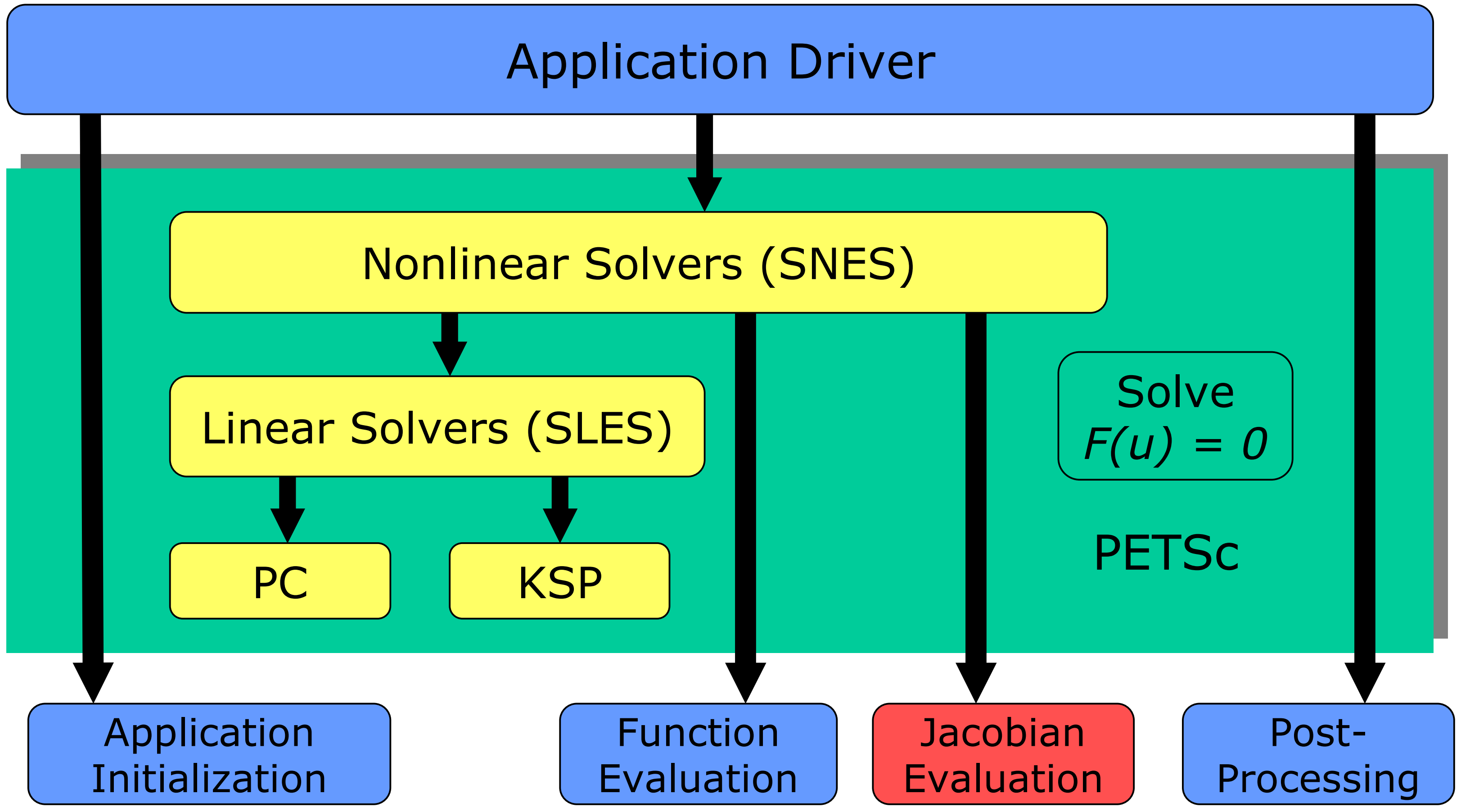
optimal boundary controls shown as velocity vectors

c/o G. Biros and O. Ghattas

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

Carnegie Mellon

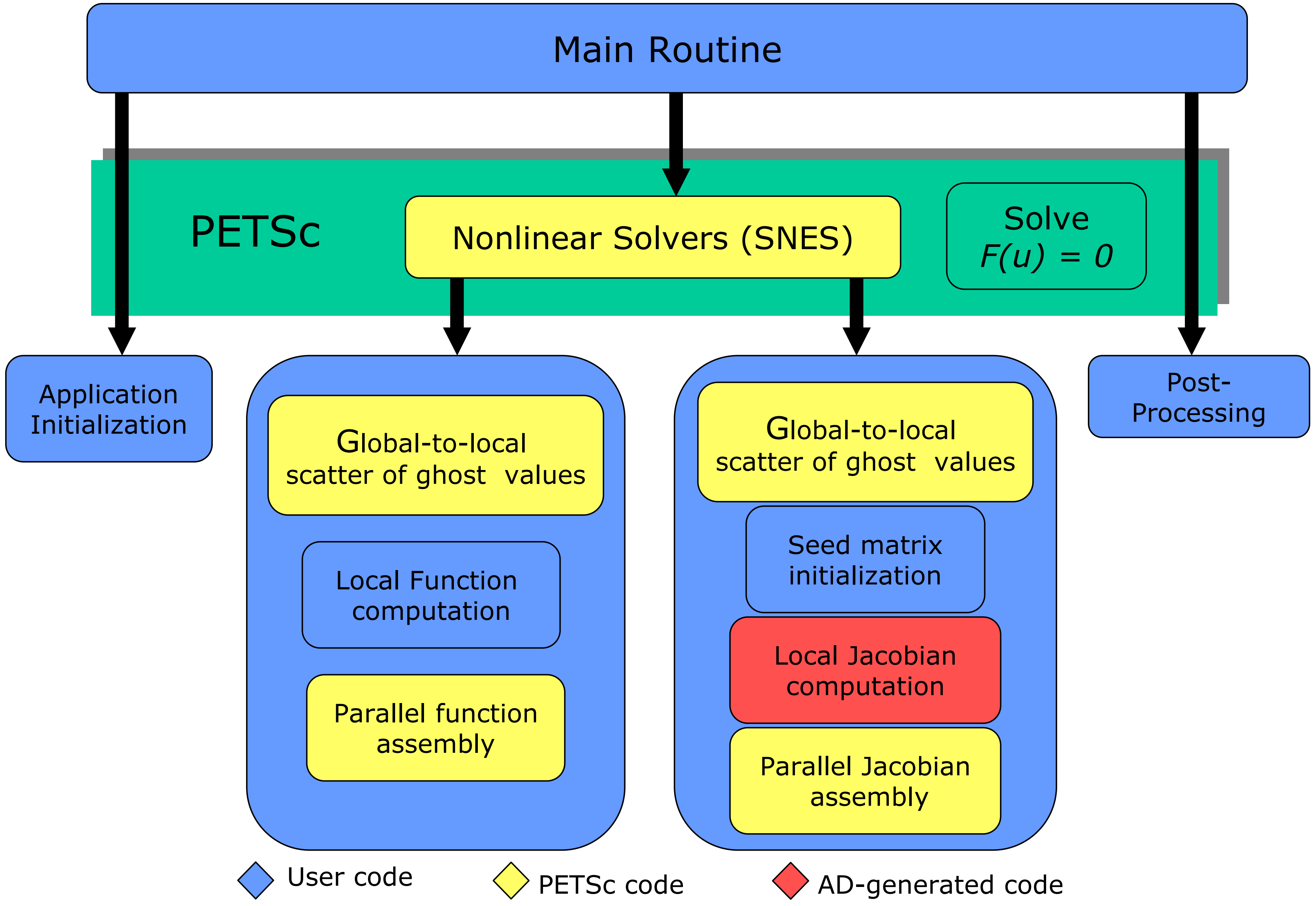
Nonlinear PDE solution w/PETSc



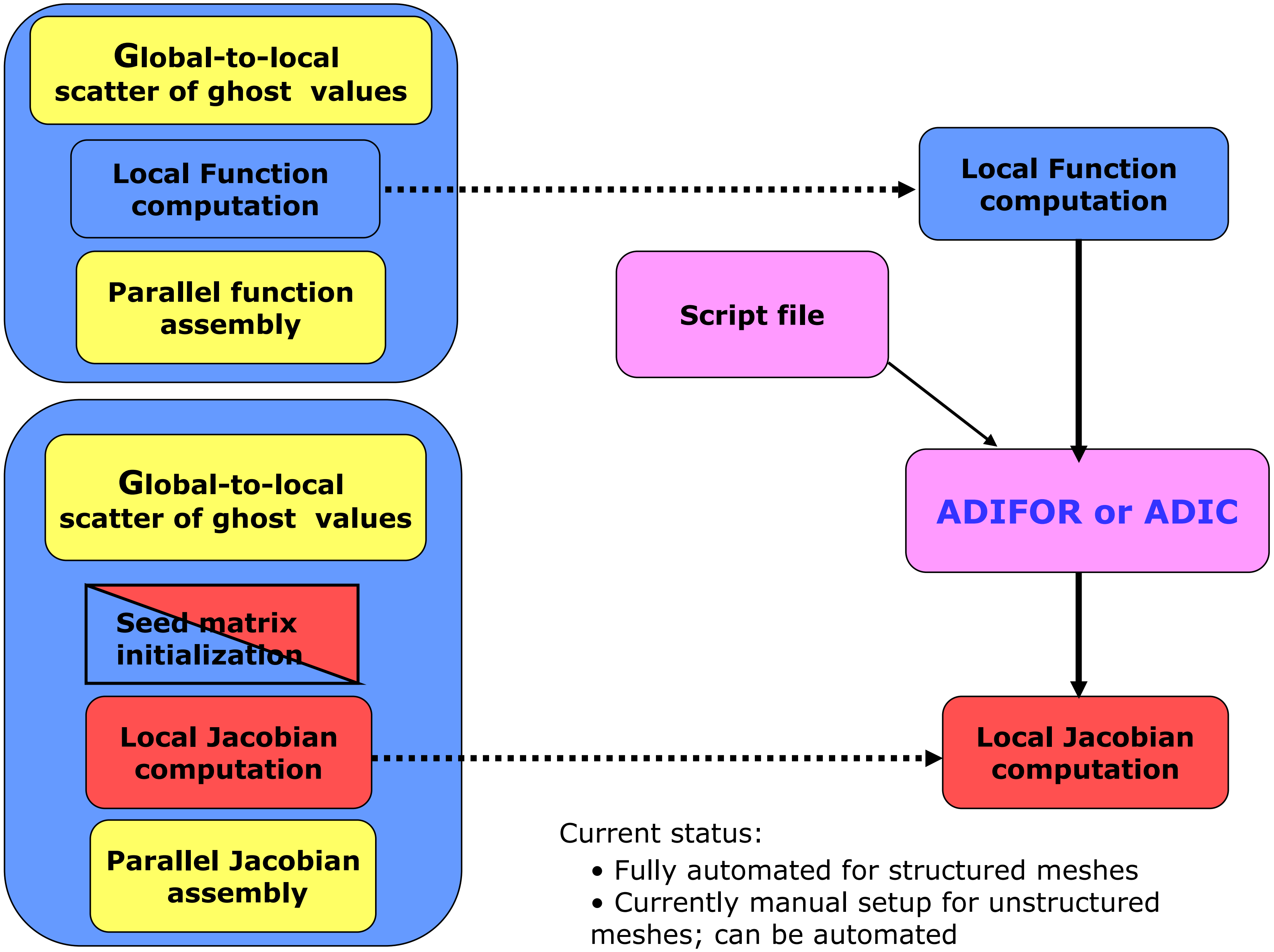
🔷 User code 🔶 PETSc code 🔶 AD-generated code

- **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


Main Routine
PETSc
Nonlinear Solvers (SNES)
Solve F(u) = 0
Application Initialization
Global-to-local scatter of ghost values
Local Function computation
Parallel function assembly
Global-to-local scatter of ghost values
Seed matrix initialization
Local Jacobian computation
Parallel Jacobian assembly
Post-Processing
User code
PETSc code
AD-generated code

Using AD with PETSc: Creation of AD Jacobian from function


Global-to-local scatter of ghost values
Local Function computation
Parallel function assembly
Global-to-local scatter of ghost values
Seed matrix initialization
Local Jacobian computation
Parallel Jacobian assembly
Local Function computation
Script file
ADIFOR or ADIC
Local Jacobian computation

Current status:

- Fully automated for structured meshes
- Currently manual setup for unstructured meshes; can be automated

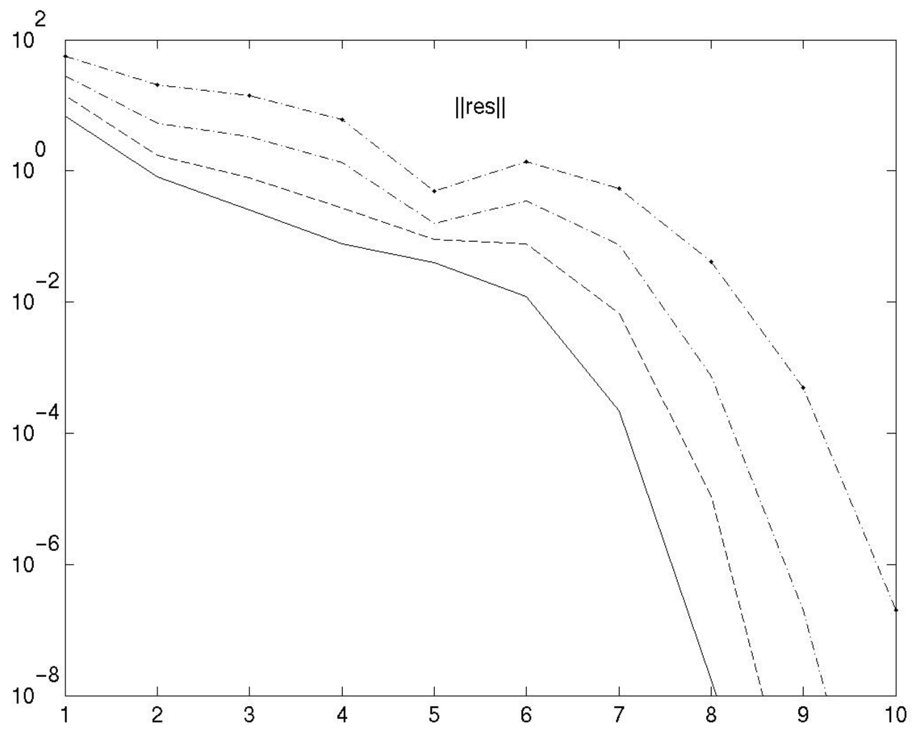
Parameter identification model

- **Nonlinear diffusion PDE BVP:** $\nabla \bullet (\alpha(x) T^{\beta} \nabla T) = 0$
- **Parameters to be identified:** $\alpha(x)$, β
- **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:** $\Phi = \| T(x) - \overline{T}(x) \|^2$ **where** $\overline{T}(x)$ **is synthetic data specified from *a priori* solution with given** $\alpha(x)$ **piecewise constant,** $\beta = 2.5$ **(Brisk-Spitzer approximation for radiation diffusion)**

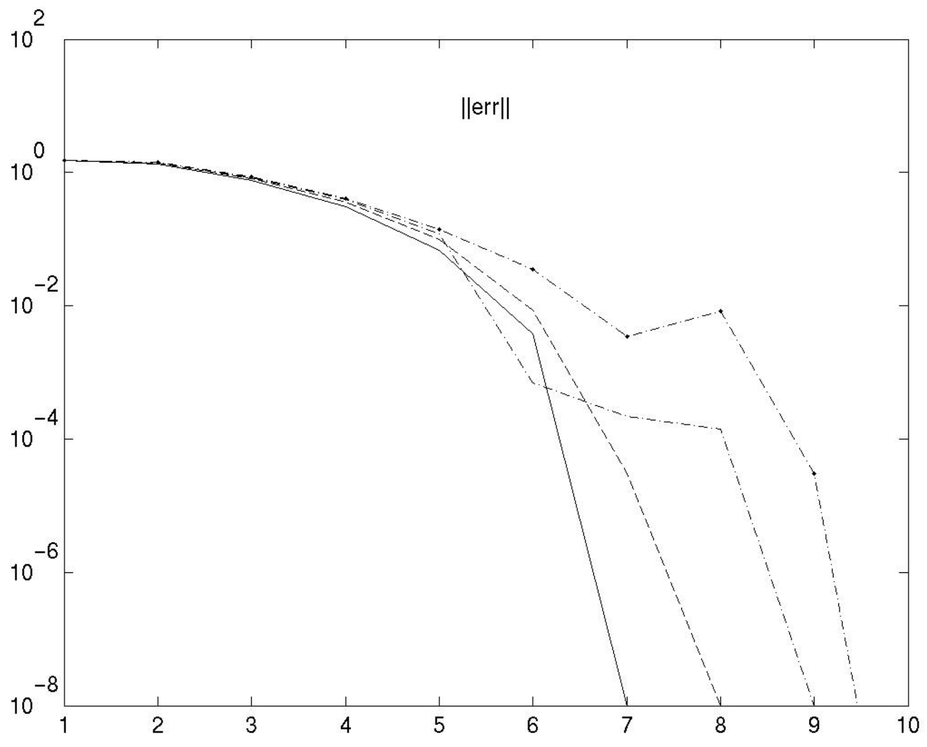
Implementation

- **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 $\alpha(x)$ and identify exponent β *only,* while uniform mesh density is refined in 2D; have also identified $\alpha(x)$ throughout full domain**
- **Newton-like, mesh-independent convergence for overall residual**

Illustrative results



2-norm of residual of full system $F(T(x),\beta,\lambda(x))$ vs. iteration



$|\beta - \beta^{*}|$ vs. iteration

[**solid**: 25×25 2Dmesh, **dash**: 50×50, **dot-dash**: 100×100, **dot-dot-dash**:200×200]

Closing

- TOPS is providing interoperable combinations of the tools in the ACTS Toolkit described so far in the program (**PETSc, TAO, SuperLU, Hypre**) and others
- However, TOPS does not pretend that the best "solutions" will be prepackaged and available across simple abstract interfaces
- Rather, TOPS hopes to provide multilayered access to solver building blocks that can be assembled by users to build more sophisticated solvers

"Knowing what is big and what is small is more important than being able to solve partial differential equations." **– S. Ulam**

TOPS

Terascale Optimal PDE Simulations

http://www.tops-scidac.org