Scalable Solver Strategies for MHD

David Keyes
Columbia University
& TOPS SciDAC Project

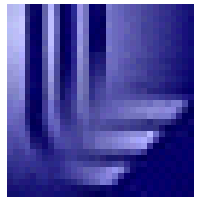
Outline

- **TOPS project "elevator speech"**
  - **five slides of propaganda collected by OASCR**
  - **TOPS renewed for a second five years, 2006-2011**
  - **TOPS refining its application plans currently**
  - ***must* target the petascale (2 platforms available by late 2008)**
- **Scalable solvers for PDEs**
  - **definition of scalable**
  - **two families of techniques that will *not* scale, and why**
  - **a family of techniques that will**
- **Comments on the "Jardin-Keyes roadmap" for MHD simulations at ITER scale**
- **TOPS wishlist for MHD collaborations**

The TOPS Center for Enabling Technology spans 4 labs & 5 universities

Our mission: Enable scientists and engineers to take full advantage of petascale hardware by overcoming the scalability bottlenecks traditional solvers impose, and assist them to move beyond "one-off" simulations to validation and optimization

Argonne NATIONAL LABORATORY

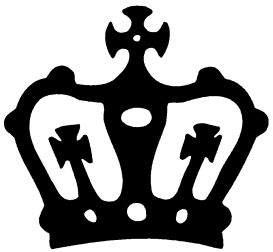


Lawrence Livermore National Laboratory

TOPS

Towards Optimal Petascale Simulations

Sandia National Laboratories



Columbia University

University of Colorado

Berkeley
University of California

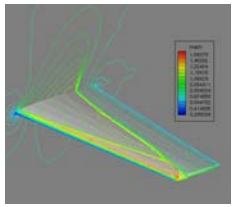
University of Texas

UCSD

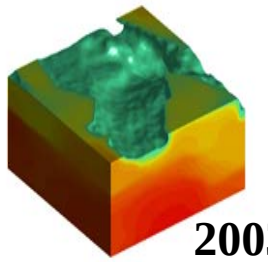
University of California at San Diego

Impact: TOPS software has a strong track record of taking applications to the architectural leading edge

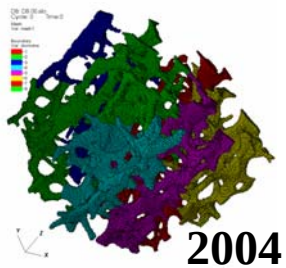
- **TOPS is at the heart of three Gordon Bell "Special" Prizes**



1999 fluids

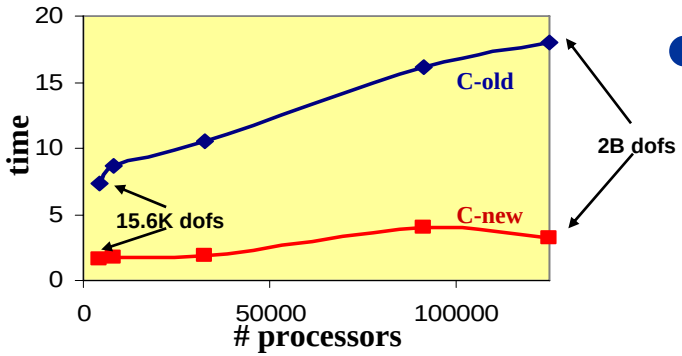


2003 seismic



2004 mechanics

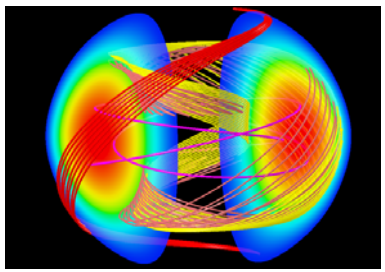
After new coarsening algorithm (red), nearly flat scaled speedup for Algebraic Multigrid



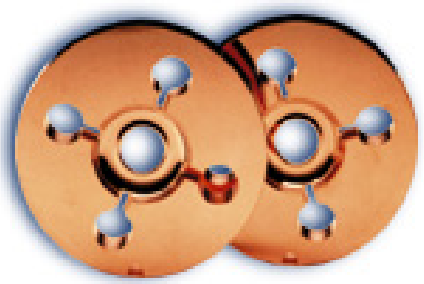
- **Scales to the edge of BlueGene/L (131,072 processors, 2B unknowns)**

- **Powered numerous applications achievements in SciDAC-1**

~5X speedup of plasma fusion code through linear solver replacement – like providing "next generation" computer

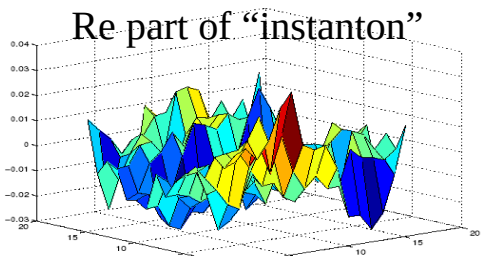


magneto-hydro-dynamics

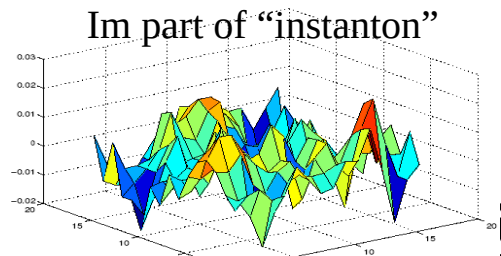


accelerator design

Prototype shape optimization capability



QCD

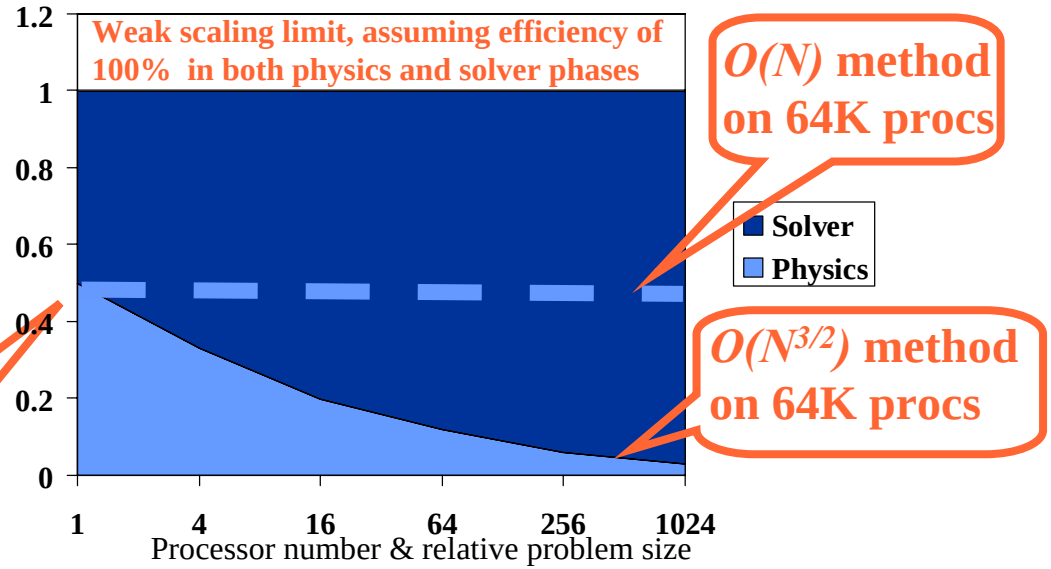


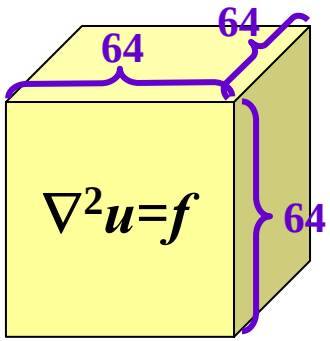
Robust solution algorithm for zero quark mass, fine lattices

Why TOPS is needed: new algorithms solve problems that new architectures cannot address

Given, for example:

- a "physics" phase that scales as *O(N)*
- a "solver" phase that scales as $O(N^{3/2})$
- computation is almost all solver after several doublings
- Optimal *O(N)* solver saves the computational cycles for the physics



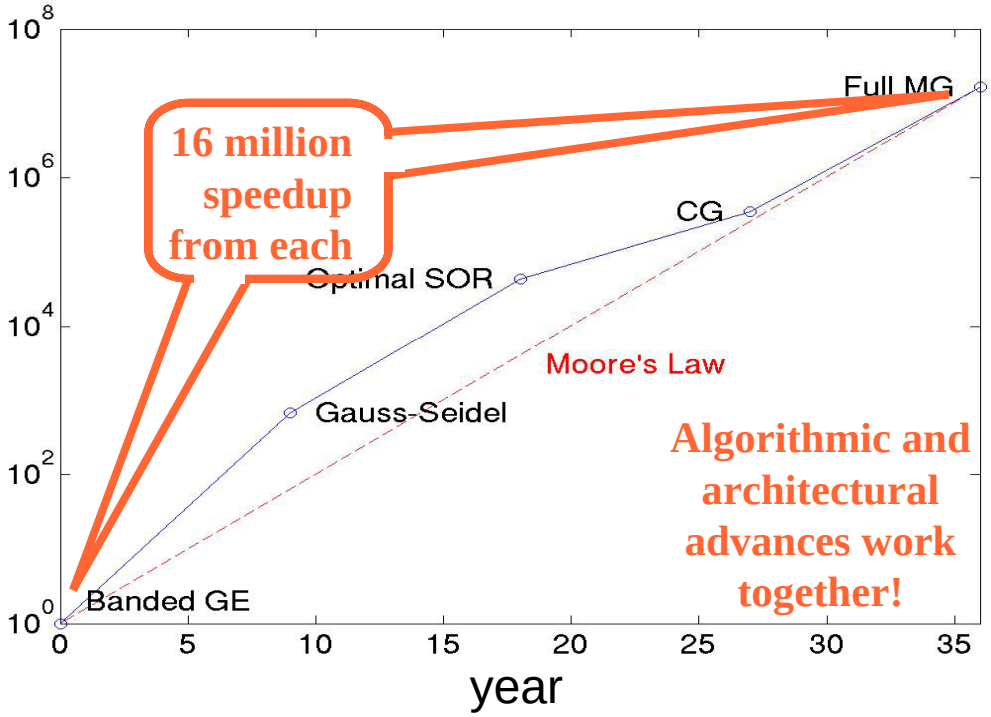


Consider, for example:

- Poisson's equation in a 3D domain
- Solve by "best method available" over a span of 1948 to 1984 (36 years)

Compare with Moore's Law:

- Over 36 years, processor architecture goes through 24 "doubling periods"
- Algorithms produce an equal factor of speedup on a small problem; much *more* on a larger problem



SciDAC-2 applications needing scalable solvers

- **Accelerator design**
  - Maxwell eigenproblems
  - shape optimization subject to PDE constraints
- **Plasma fusion**
  - Poisson problems
  - coupled nonlinear systems within a single “physics” domain (e.g., MHD)
  - nonlinear coupling of multiple physics codes
- **Porous media flow**
  - div-grad Darcy problems
- **Quantum chromodynamics**
  - Dirac operator inversions
- **Quantum chemistry**
  - generalized eigenproblems
- **Physicists want to concentrate on physics instead of solvers**
  - express solver tasks at a level of mathematical abstraction
  - exploit state-of-the-art solvers as these evolve under the interface
  - run *same code* on laptops (on travel), low-cost unmetered clusters (at work), and on unique shared national resources
- **Ordered goals for TOPS (need them all, in this order)**
  - usability and robustness
  - portability
  - algorithmic efficiency (optimality) and implementation efficiency (within a processor and in parallel)
  - algorithmic optimality and software stability

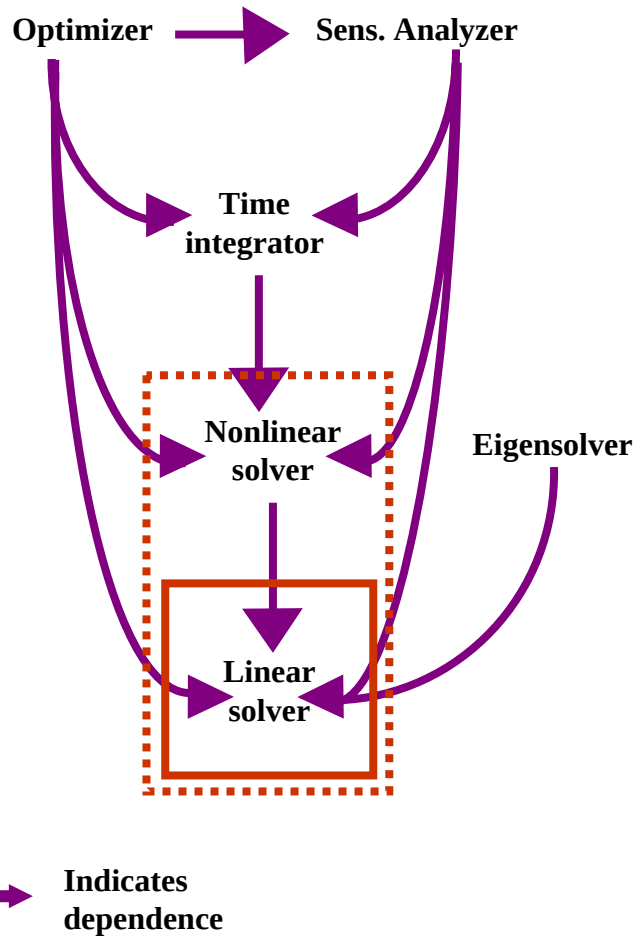
TOPS is building a toolchain of proven solver components that interoperate

- We carry users from "one-off" *solutions* to the full scientific agenda of *sensitivity, stability,* and *optimization* (from heroic *point studies* to systematic *parametric studies*) all in one software suite
- TOPS solvers are nested, from applications-hardened linear solvers outward, leveraging common distributed data structures
- Communication and performance-oriented details are hidden so users deal with mathematical objects throughout
- TOPS features these trusted packages, whose functional dependences are illustrated (right)*:

  Hypre, PETSc, SUNDIALS, SuperLU, TAO, Trilinos

  These are in use and actively debugged in dozens of high-performance computing environments, in dozens of applications domains, by thousands of user groups around the world

  * See also the webpages for each code



Some TOPS personnel relevant to MHD efforts

Adams

Ghattas

So far:

0.0 FTE

for MHD collaborations

Heroux

McCormick

Moré

Ng

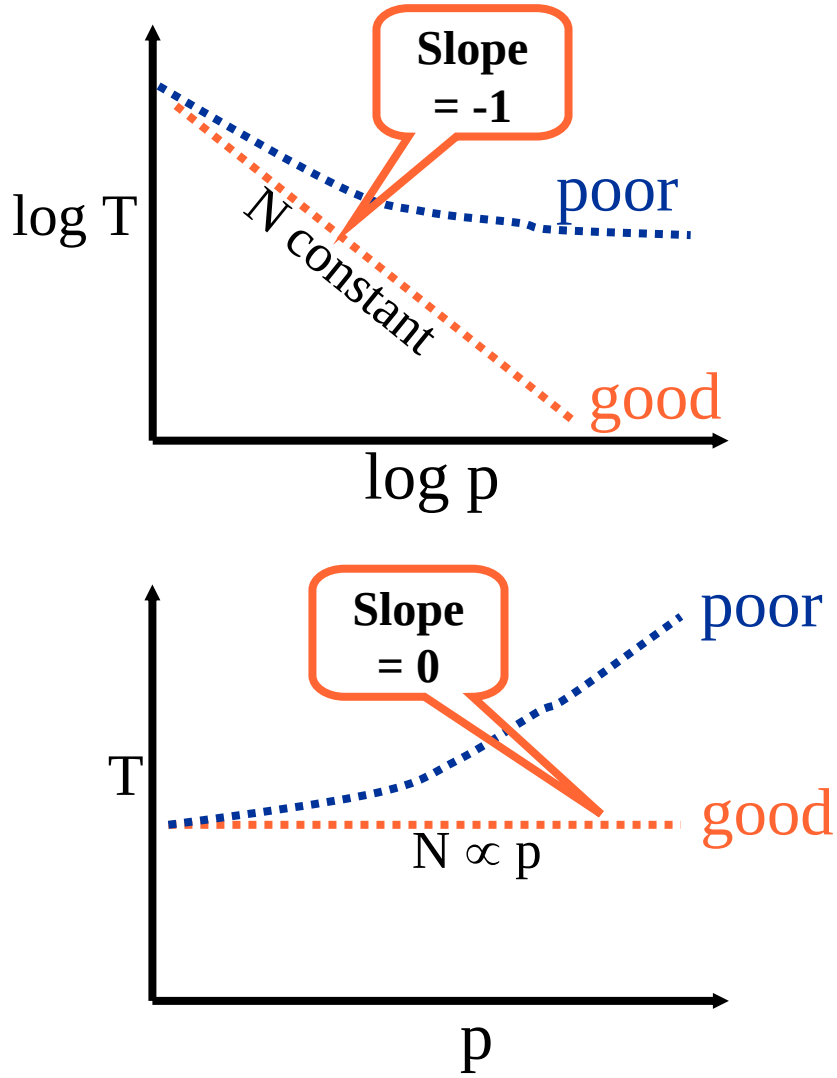
Reynolds

Smith

Woodward

Review: two definitions of scalability

- **"Strong scaling"**
  - **execution time decreases in inverse proportion to the number of processors**
  - ***fixed size problem overall***
  - **often instead graphed as reciprocal, "speedup"**
- **"Weak scaling"**
  - **execution time remains constant, as problem size and processor number are increased in proportion**
  - ***fixed size problem per processor***
  - **also known as "Gustafson scaling"**



Contraindications of scalability

- **Fixed problem size**
  - **Amdahl-type constraints**
    - **“fully resolved” discrete problems (protein folding, network problems)**
    - **“sufficiently resolved” problems from the continuum**
- **Scalable problem size**
  - **Resolution-limited progress in “long time” integration**
    - **explicit schemes for time-dependent PDEs**
    - **suboptimal iterative relaxations schemes for equilibrium PDEs**
  - **Nonuniformity of threads**
    - **adaptive schemes**
    - **multiphase computations (e.g, particle and field)**

Amdahl's Law (1967)

- Fundamental limit to strong scaling due to small overheads
- Speedup asymptotically independent of number of processors available
- Analyze by binning code segments by degree of exploitable concurrency and dividing by available processors, up to limit
- Illustration for just two bins:
  - fraction f_1 of work that is purely sequential
  - fraction (1-f_1) of work that is arbitrarily concurrent
- Wall clock time for ***p*** processors $\propto f_1 + (1 - f_1)/p$
- Speedup $= 1/[f_1 + (1 - f_1)/p]$
  - for f_1=0.01
- Applies to any performance enhancement, not just parallelism

p	1	10	100	1000	10000
S	1.0	9.2	50.3	91.0	99.0

Resolution-limited progress (weak scaling)

- **Illustrate for CFL-limited explicit time stepping**
- **Parallel wall clock time**

$$\propto T\, S^{1+\alpha/d}\, P^{\alpha/d}$$

- **Example: explicit wave problem in 3D ($\alpha=1$, $d=3$)**

Domain	$10^3\times 10^3\times 10^3$	$10^4\times 10^4\times 10^4$	$10^5\times 10^5\times 10^5$
Exe. time	**1 day**	**10 days**	**3 months**

- **Example: explicit diffusion problem in 2D ($\alpha=2$, $d=2$)**

Domain	$10^3\times 10^3$	$10^4\times 10^4$	$10^5\times 10^5$
Exe. time	**1 day**	**3 months**	**27 years**

d-dimensional domain, length scale L
$d+1$-dimensional space-time, time scale T
h mesh cell size
τ time step size
$\tau=O(h^\alpha)$ bound on time step
$n=L/h$ number of mesh cells in each dim
$N=n^d$ number of mesh cells overall
$M=T/\tau$ number of time steps overall
$O(N)$ total work to perform one time step
$O(MN)$ total work to solve problem
P number of processors
S storage per processor
PS total storage on all processors $(=N)$
$O(MN/P)$ parallel wall clock time
$\propto (T/\tau)(PS)/P \propto T\, S^{1+\alpha/d}\, P^{\alpha/d}$
(since $\tau \propto h^\alpha \propto 1/n^\alpha = 1/N^{\alpha/d} = 1/(PS)^{\alpha/d}$ **)**

"Scalable" includes "optimal"

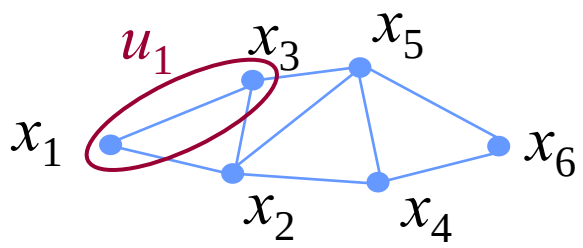
- "Optimal" for a theoretical numerical analyst means a method whose floating point complexity grows at most linearly in the data of the problem, N, or linearly times a polylog term
- For iterative methods, this means that both the *cost per iteration* and the *number of iterations* must be $O(N \log^p N)$
- Cost per iteration must include communication cost as processor count increases in weak scaling, $P \propto N$
  - BlueGene permits this with its log-diameter global reduction
- Number of iterations comes from condition number for linear iterative methods; Newton's superlinear convergence is important for nonlinear iterations

Scalable solvers for PDEs

- **Linear preconditioners**
  - **Domain-decomposition methods**
    - **Schwarz (DD by projection)**
    - **Schur (DD by partition and elimination)**
    - **Schwarz-Schur hybrids**
  - **Multigrid**
- **Linear accelerators**
  - **Krylov methods**
- **Nonlinear rootfinders**
  - **Newton-like methods**
- **Hybrids (nonlinear Schwarz, FAS multigrid) and implications for multiphysics coupling**

Digression for notation's sake

- **We need a convenient notation for mapping vectors (representing discrete samples of a continuous field) from full domain to subdomain and back**



- **Let R_i be a Boolean operator that extracts the elements of the i^{th} subdomain from the global vector**

$$R_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$R_1 u = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} \equiv u_1$$

- **Then R_i^T maps the elements of the i^{th} subdomain back into the global vector, padding with zeros**

$$R_1^T = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

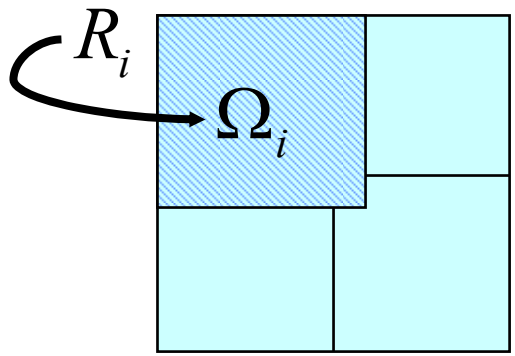
$$R_1^T u_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_1 \\ 0 \\ x_3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

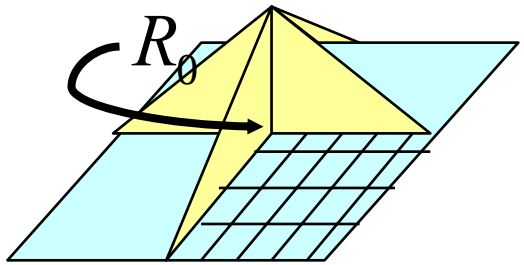
Schwarz domain decomposition method

- **Consider restriction and extension operators for subdomains,** R_i, R_i^T, **and for possible coarse grid,** R_0, R_0^T
- **Replace discretized** $Au = f$ **with**

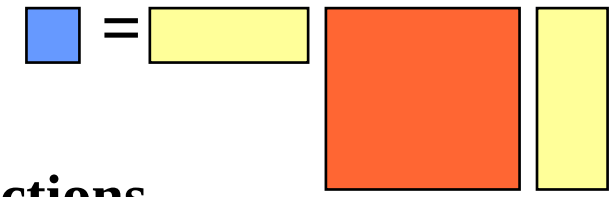
$$B^{-1}Au = B^{-1}f$$

$$B^{-1} = R_0^T A_0^{-1} R_0 + \sum_i R_i^T A_i^{-1} R_i$$





$$A_i = R_i A R_i^T$$



- **Solve by a Krylov method**
- **Matrix-vector multiplies with**
  - **parallelism on each subdomain**
  - **nearest-neighbor exchanges, global reductions**
  - **possible small global system (not needed for parabolic case)**

Schwarz formula (projections)

- **If A is an operator on a space V and R_i are restrictions into (possibly overlapping) subspaces of V, V_i, such that $V=\cup V_i$**
- **Then for a good approximation, B^{-1}, to A^{-1}:**

$$B^{-1} = \sum_i R_i^T (R_i A R_i^T)^{-1} R_i + R_0^T (R_0 A R_0^T)^{-1} R_0$$

or

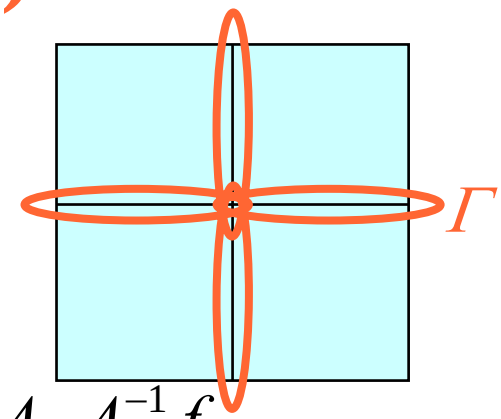
$$B^{-1} = \sum_i R_i^T A_i^{-1} R_i + R_0^T A_0^{-1} R_0$$

- **Then** $\kappa(B^{-1}A) = C$

where *C* is independent of *H* and *h* (resp. *P* and *N*)

Schur formula (partitions)

- **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:**

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

- **The full system matrix factors:**

$$A = \begin{bmatrix} A_{ii} & 0 \\ A_{\Gamma i} & I \end{bmatrix} \begin{bmatrix} I & A_{ii}^{-1} A_{i\Gamma} \\ 0 & S \end{bmatrix}$$

- **Then for a good approximation, B^{-1}, to A^{-1}:**

$$B^{-1} = \begin{bmatrix} I & \tilde{A}_{ii}^{-1} A_{i\Gamma} \\ 0 & M \end{bmatrix}^{-1} \begin{bmatrix} \tilde{A}_{ii} & 0 \\ A_{\Gamma i} & I \end{bmatrix}^{-1}$$

where M is S, or any good approximation to it

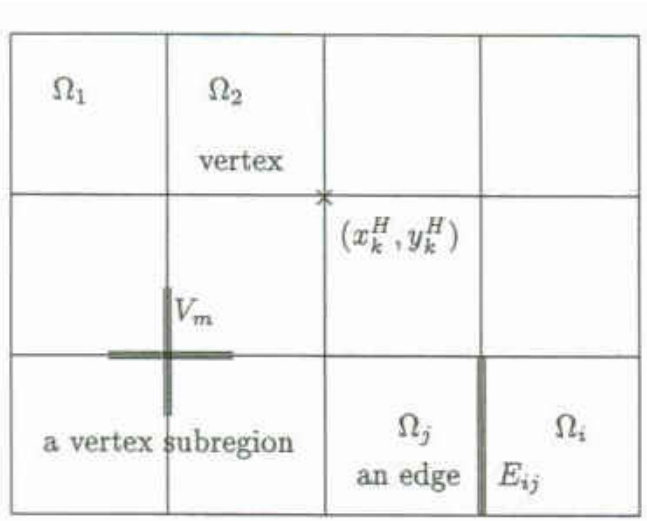
Schwarz-on-Schur

- **Since S and M may be complicated, we can further decompose the multisegmented interface into simple edges and a vertex block, preconditioned separately:**

$$M^{-1} = \sum_i R_{E_i}^T S_{E_i E_i}^{-1} R_{E_i} + R_H^T A_H^{-1} R_H$$

then

$$\kappa(M^{-1}S) = C(1+\log^2(Hh^{-1}))$$



where C is independent of H and h (but may still retain dependencies on other "bad" parameters, such as jumps in the diffusion coefficients)

Operator projection ideas not limited to DD

- In the abstract, Schwarz theory is about polynomials of projection operators
- Appropriate for other types of preconditioners, too
- Suppose we have two preconditioners, each of which is effective on part of the problem, and we use them sequentially

$$u \leftarrow u + B_1^{-1}(f - Au)$$

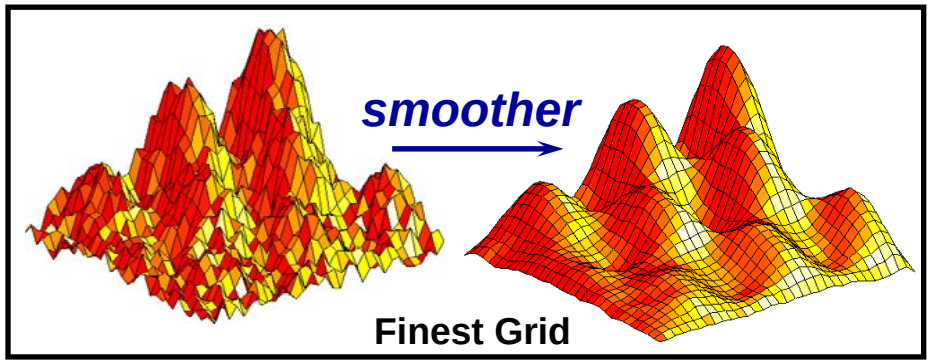
$$u \leftarrow u + B_2^{-1}(f - Au)$$

- This leads to a multiplicative scheme:

$$B^{-1} = B_2^{-1} + (I - B_2^{-1}A)B_1^{-1}$$

- This is the form of a standard two-level multigrid scheme in which B_1 is a "smoother" and B_2 handles the complementary modes: $B_2^{-1} = R^T A_C^{-1} R\,; \quad A_C = RAR^T$

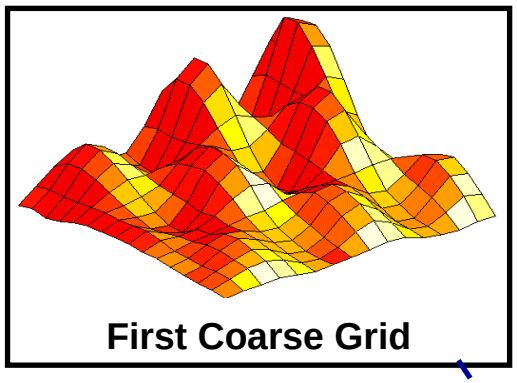
For multigrid, one recurs on this…



A Multigrid V-cycle

Restriction
transfer from fine to coarse grid

coarser grid has fewer cells (less work & storage)



Recursively apply this idea until we have an easy problem to solve

Prolongation
transfer from coarse to fine grid

Algebraic multigrid

- **For Poisson, there is a correspondence between the hard-to-smooth error modes and wavenumber, leading to the classification of "fine" (easy to smooth) and "coarse" (hard to smooth, near null-space)**
- **For more general operators, this *geometrical* correspondence is broken; the "coarse" space is whatever is complementary to the readily smoothable space and is found *algebraically*, in an operator-sensitive way (anisotropy, inhomogeneity, etc.)**
- **This freedom from geometry is liberating, since problems on unstructured meshes are readily accommodated**
- **Near null-space modes may now, however, be *dense* to represent, unlike in Poisson (okay, if just a few of them)**
- **Identifying the coarse space may defy heuristics and need to be found *adaptively* (see SIAM Review 47:317-346 (2005))**

Krylov accelerators

- Given $Ax = b$, $A \in \Re^{n \times n}$ and iterate x^0, we wish to generate a basis $V = \{v_1, v_2, ..., v_k\} \in \Re^{n \times k}$ for x ($x \approx Vy$) and a set of coefficients $\{y_1, y_2, ..., y_k\}$ such that x^k is a best fit in the sense that $y \in \Re^k$ minimizes $\| AVy - b \|$
- Krylov methods are algebraic Petrov-Galerkin methods that define a complementary "test" basis $W = \{w_1, w_2, ..., w_k\} \in \Re^{n \times k}$ so that $W^T (AVy - b) = 0$ may be solved for y
- In practice $k << n$ and the bases are grown from seed vector $r^0 = Ax^0 - b$ via recursive multiplication by A and Gram-Schmidt

Onward to nonlinearity

- **Linear versus nonlinear problems**
  - **Solving linear systems often constitutes 90% of the running time of a large PDE simulation**
  - **The nonlinearity is often a fairly straightforward outer loop, in that it introduces no new types of messages or synchronizations not present in Krylov-Schwarz, and has overall many fewer synchronizations than the preconditioned Krylov method or other linear solver inside it**
- **We can wrap Newton, Picard, fixed-point or other iterations outside, linearize, and apply what we know**
- **We consider both Newton-outside and Newton-inside methods**

Newton-Krylov-Schur-Schwarz: a solver "workhorse"

$$F(u) \approx F(u_c) + F'(u_c)\delta u = 0$$
$$u = u_c + \lambda\, \delta u$$

$$J\delta u = -F$$
$$\delta u = \underset{x \in V \equiv \{F, JF, J^2F, \cdots\}}{\operatorname{argmin}} \{Jx + F\}$$

$$B^{-1} J\delta u = -B^{-1} F$$
$$B^{-1} = \left(\begin{bmatrix} \tilde{A}_{ii} & 0 \\ A_{\Gamma i} & I \end{bmatrix} \begin{bmatrix} I & \tilde{A}_{ii}^{-1} A_{i\Gamma} \\ 0 & M \end{bmatrix} \right)^{-1}$$
$$\tilde{A}^{-1} = \sum_i R_i^T (R_i A R_i^T)^{-1} R_i$$

Newton

nonlinear solver

asymptotically quadratic

Krylov

accelerator

spectrally adaptive

Schur

preconditioner

parallelizable by structure

Schwarz

preconditioner

parallelizable by domain

Newton-like iteration

- Given $F(u) = 0, F : \Re^n \rightarrow \Re^n$ 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 $\delta u^k = u^{k+1} - u^k, k = 0,1,2,...$

- 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 $\| F(u^k + \lambda \delta u^k) \|$

Newton-Krylov-Schwarz

Newton loop

```
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
```

Krylov loop

Nonlinear Schwarz preconditioning

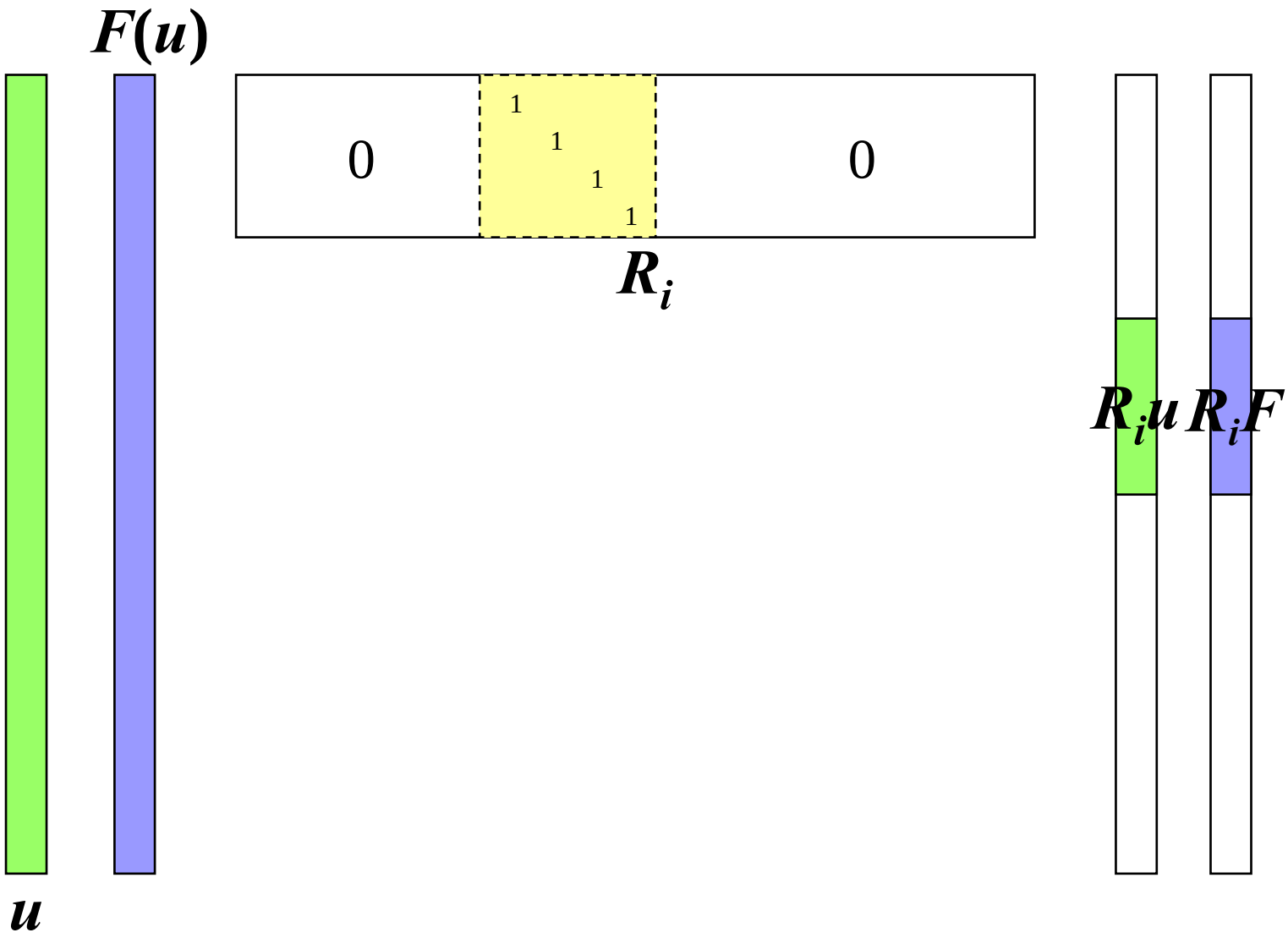
- **Nonlinear Schwarz has Newton both *inside* and *outside* and is fundamentally Jacobian-free**
- **It replaces** $F(u) = 0$ **with a new nonlinear system possessing the same root,** $\Phi(u) = 0$
- **Define a correction** $\delta_i(u)$ **to the** i^{th} **partition (e.g., subdomain) of the solution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

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

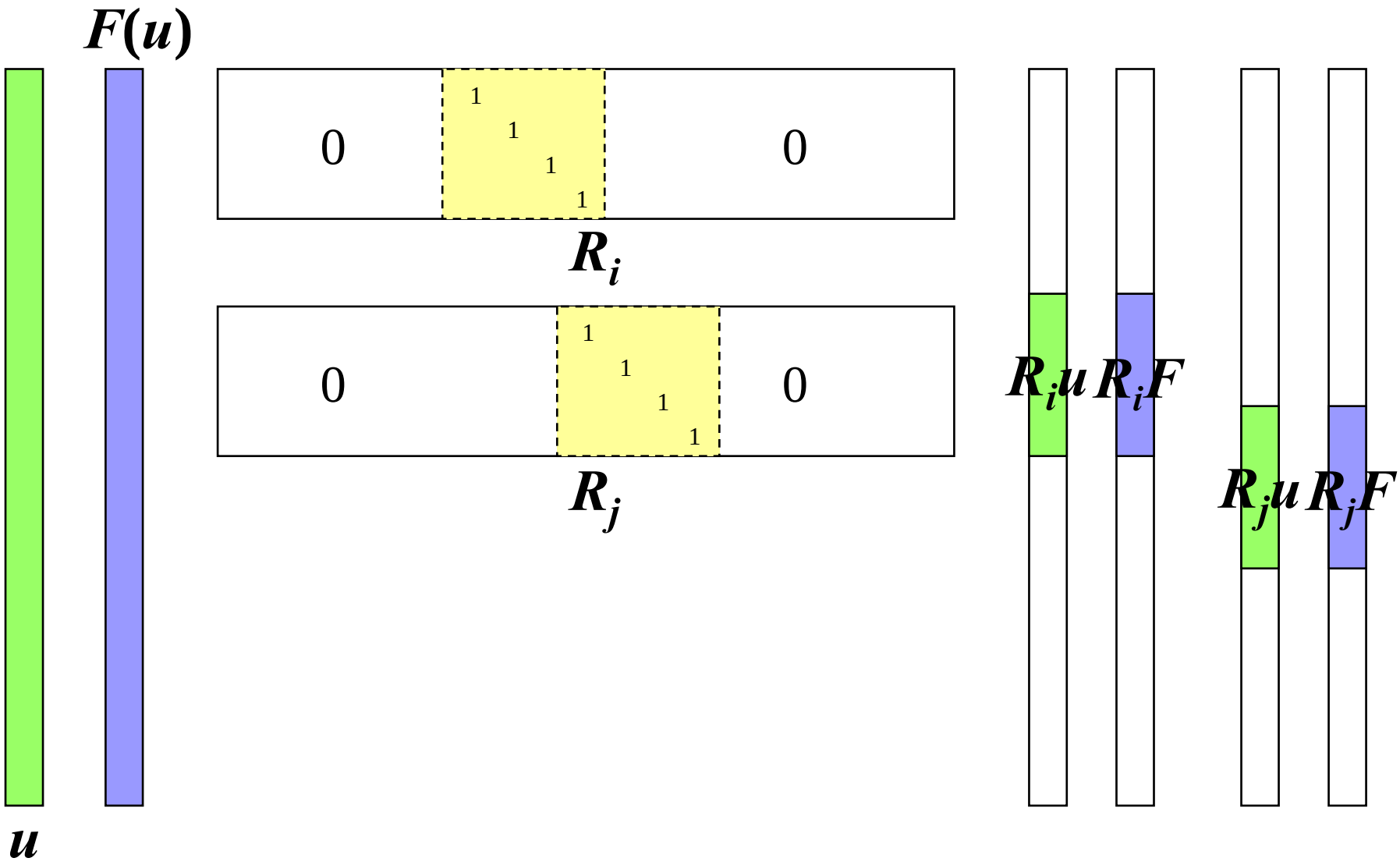
where $\delta_i(u) \in \Re^n$ **is nonzero only in the components of the** i^{th} **partition**

- **Then sum the corrections:** $\Phi(u) = \sum_i \delta_i(u)$ **to get an implicit function of** u

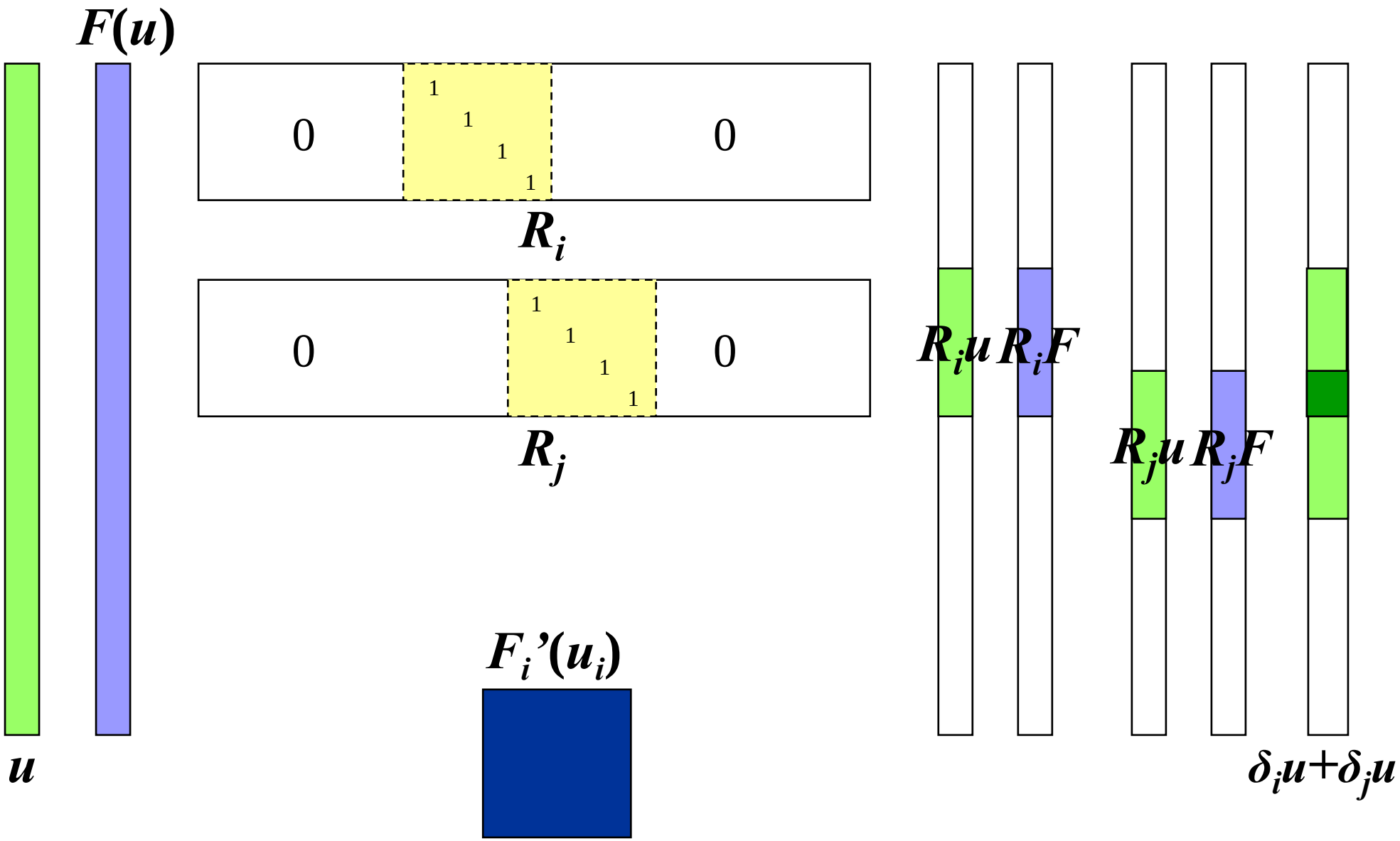
Nonlinear Schwarz – picture



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 $F(u)$!

Full Approximation Scheme (FAS) multigrid

- In Newton-Krylov-Schwarz, the linearization is global, then the global linear problem is additively domain decomposed into local subdomains
- In nonlinear Schwarz, the nonlinear problem is additively domain decomposed *directly* into local subdomains
- FAS multigrid is like nonlinear Schwarz, except that the subspaces are global and multiscale rather than local and multidomain; and they are handled multiplicatively rather than additively
- Historically, FAS has not had a good software engineering model, since the user must provide nonlinear residual evaluations of arbitrary subsets of the global problem
- TOPS is working on this, using macros and inlining (and techniques borrowed from automatic differentiation); FAS should become available in PETSc-3 during TOPS2

From *SIAM News*, Volume 39, Number 7, September 2006

Taking on the ITER Challenge, Scientists Look to Innovative Algorithms, Petascale Computers

By Michelle Sipics

The promise of fusion as a clean, self-sustaining and essentially limitless energy source has become a mantra for the age, held out by many scientists as a possible solution to the world's energy crisis and a way to reduce the amounts of greenhouse gases released into the atmosphere by more conventional sources of energy. If self-sustaining fusion reactions can be realized and maintained long enough to produce electricity, the technology could potentially revolutionize energy generation and use.

ITER, initially short for International Thermonuclear Experimental Reactor, is now the official, non-acronymic name (meaning "the way" in Latin) of what is undoubtedly the largest undertaking of its kind. Started as a collaboration between four major parties in 1985, ITER has evolved into a seven-party project that finally found a physical home last year, when it was announced that the ITER fusion reactor would be built in Cadarache, in southern France. (The participants are the European Union, Russia, Japan, China, India, South Korea, and the United States.) In May, the seven initialed an agreement documenting the negotiated terms for the construction, operation, and decommissioning of the ITER tokamak, signifying another milestone for both the project itself and its eventual goal of using fusion to facilitate large-scale energy generation for the world.

Problems remain, however—notably the years, and perhaps decades, of progress needed to attain such a goal. In fact, even *simulating* the proposed ITER tokamak is currently out of reach. But according to David Keyes, a computational mathematician at Columbia University and acting director of the Institute for Scientific Computing Research (ISCR) at Lawrence Livermore National Laboratory, the ability to perform such simulations may be drawing closer.

Hardware 3, Software 9

"Fusion scientists have been making useful characterizations about plasma fusion devices, physics, operating regimes and the like for over 50 years," Keyes says. "However, to simulate the dynamics of ITER for a typical experimental 'shot' over scales of interest with today's most commonly used algorithmic technologies would require approximately 10^{24} floating-point operations." That sounds bleak, given the 280.6 Tflop/s (10^{12} flops/s) benchmark performance of the IBM BlueGene/L at Lawrence Livermore National Laboratory—as of June the fastest supercomputer in the world. But Keyes is optimistic: "We expect that with proper algorithmic ingenuity, we can reduce this to 10^{15} flops."

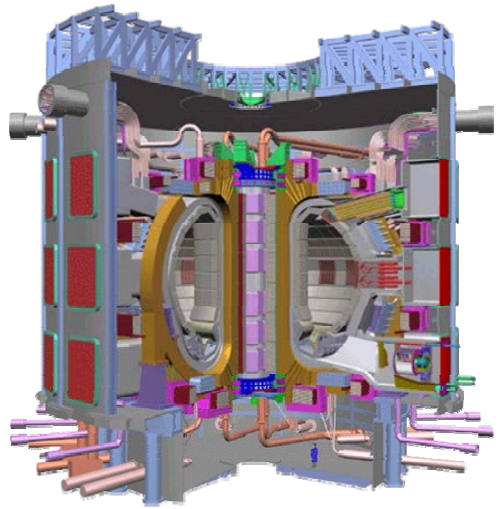
Optimizing the algorithms used, in other words, could lower the computing power required for some ITER simulations by an astounding nine orders of magnitude. Even more exciting, those newly feasible simulations would be at the petascale—ready to run on the petaflop/s supercomputers widely expected within a few years.

The ingenuity envisioned by Keyes even has a roadmap. Together with Stephen Jardin of the Princeton Plasma Physics Laboratory, Keyes developed a breakdown that explains where as many as 12 orders of magnitude of speedup will come from over the next decade: 1.5 from increased parallelism, 1.5 from greater processor speed and efficiency, four from adaptive gridding, one from higher-order elements, one from field-line following coordinates, and three from implicit algorithms.

Scaling fusion simulations up to ITER

name	symbol	units	CDX-U	DIII-D	ITER
Field	B_0	Tesla	0.22	1	5.3
Minor radius	a	meters	.22	.67	2
Temp.	T_e	keV	0.1	2.0	8.
Lundquist no.	S		1×10^{4}	7×10^{6}	5×10^{8}
Mode growth time	$\tau_A S^{1/2}$	s	2×10^{-4}	9×10^{-3}	7×10^{-2}
Layer thickness	$aS^{-1/2}$	m	2×10^{-3}	2×10^{-4}	8×10^{-5}
zones	$N_R\times N_\theta\times N_\phi$		3×10^{6}	5×10^{10}	3×10^{13}
CFL timestep	$\Delta X/V_A$ (Explicit)	s	2×10^{-9}	8×10^{-11}	7×10^{-12}
Space-time pts			6×10^{12}	1×10^{20}	6×10^{24}

10^{12} needed (explicit uniform baseline)



International Thermonuclear Experimental Reactor

2017 – first experiments, in Cadaraches, France

c/o S. Jardin, PPPL

Where to find 12 orders of magnitude in 10 years?

Hardware: 3

- 1.5 orders: increase processor speed and efficiency
- 1.5 orders: [illegible]ncy

Software: 9

- 1 ord[illegible]
  - Same [illegible] fewer elements
- 1 [illegible]ng
  - Les[illegible]s
- 4 ord[illegible]
  - Zones requ[illegible] of ITER volume and resolution require[illegible]ents a[illegible] fro[illegible]em are ~10^2 less severe
- 3 orders: implicit solvers
  - Mode growth time 9 orders longer than Alfven-limited CFL

Algorithmic improvements bring yottascale (10^{24}) calculation down to petascale (10^{15})!

Comments on JK roadmap

- **increased processor speed**
  - **10 years is 6.5 Moore doubling times**
- **increased concurrency**
  - **BG/L is already 2^{17} procs, MHD now at ca. 2^{12}**
- **higher-order discretizations**
  - **low-order FE preconditioning of high-order discretizations (Orszag, Fischer, Manteuffel, etc.)**
- **flux-surface following gridding**
  - **evolve mesh to approximately follow flux surfaces**
- **adaptive gridding**
  - **within SciDAC, this is APDEC; we will team to make it *implicit***
- **implicit solvers**
  - **we propose Newton-like fully implicit, with Krylov/MG innards**

IMPLICIT METHODS

- Partially implicit methods
  - Treat fastest time scales implicitly
  - Time step still limited by waves
- Semi-implicit methods
  - Treat linearized ideal MHD operator implicitly
  - Time step limited by advection
  - Many iterations
- Fully implicit methods
  - Newton-Krylov treatment of full nonlinear equations
  - Arbitrary time step
  - Still a research project

c/o Dalton Schnack, 2002

TOPS' wishlist for MHD collaborations — "Asymptopia"

- **Engage at a higher-level than *Ax=b***
  - **Newton-Krylov-Schwarz/MG on coupled nonlinear system**
- **Sensitivity analyses**
- **Optimization techniques**
  - **design of apparati**
  - **control of experiments**