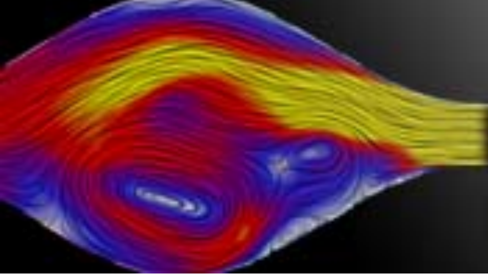


Sir James Lighthill
Distinguished Lectureship
in Mathematical Sciences

Tallahassee,
Florida

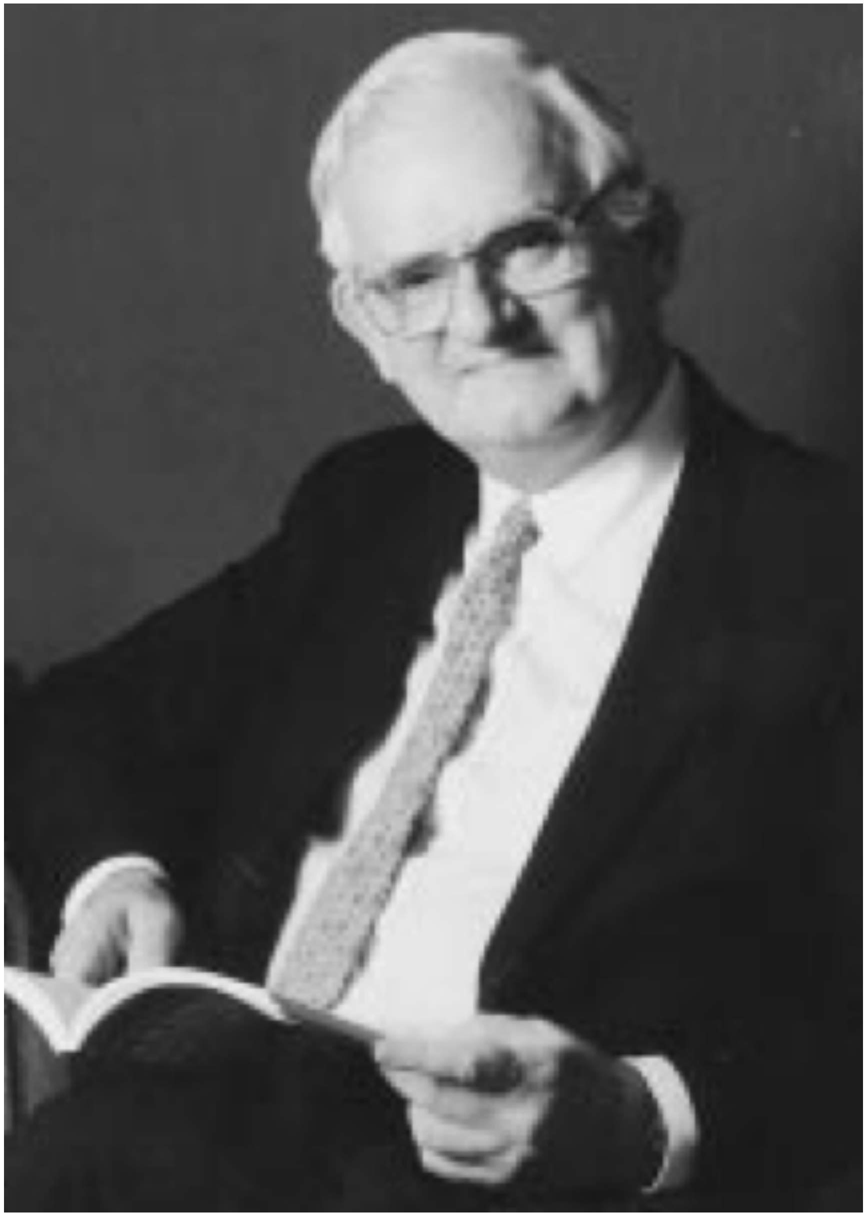
A nonlinearly implicit manifesto

David Keyes

Towards Optimal Petascale Simulations (TOPS), SciDAC Program, U.S. DOE

Mathematical and Computer Sciences & Engineering, KAUST
Applied Physics & Applied Mathematics, Columbia University

In Memoriam



23 Jan 1924 – 17 July 1998

"James Lighthill was acknowledged throughout the world as one of the great mathematical scientists of this century. He was the prototypical applied mathematician, immersing himself thoroughly in the essence and even the detail of every engineering, physical, or biological problem he was seeking to illuminate with mathematical description, formulating a sequence of clear mathematical problems and attacking them with a formidable range of techniques completely mastered, or adapted to the particular need, or newly created for the purpose, and then finally returning to the original problem with understanding, predictions, and advice for action."

(from the David Crighton bio in *AMS Notices*)

Plan of series

- Theme: role of mathematics in Computational Science & Engineering, specifically large-scale simulation
- Our philosophy has been to look at the scientific opportunity of large-scale simulation from three perspectives, concentrating one lecture on each
  - Applications, Architectures, Algorithms
- FSU Lighthill lectures are presumed neither cumulative nor exclusive
  - Individuals may attend any *one* without prerequisite
  - Individuals invited to attend all *three* (Engineering, Mathematics, Public)
- This requires a modicum of audience patience for either
  - Delegation (individual lectures not completely self-contained)
  - Repetition (lectures have some overlap)

Plan of mathematics presentation

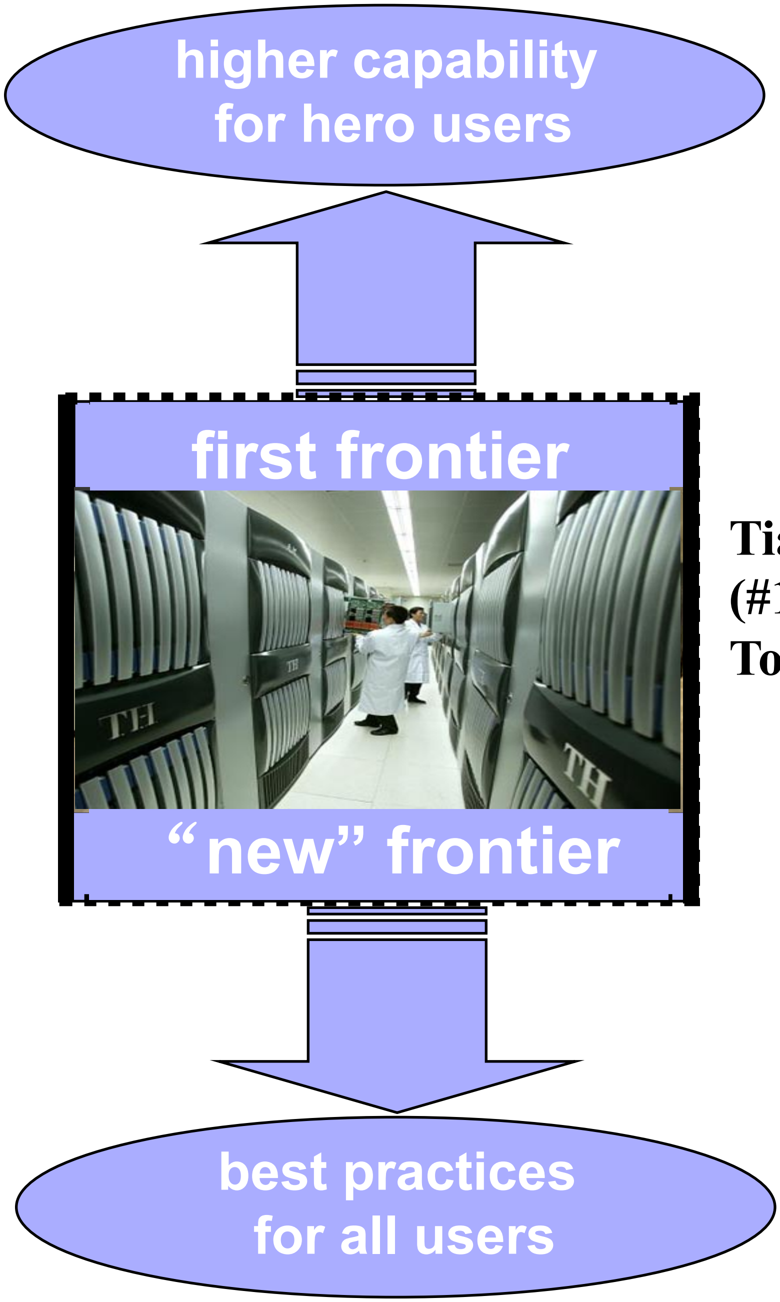
- **Motivations for implicit solvers**
  - **trends: multi-scale, multi-physics, multi-solve (sensitivity, stability, uncertainty quantification, design, control, inversion)**
  - **understanding: one-dimensional model problems, linear and nonlinear**
- **State-of-the-art for large-scale nonlinearly implicit solvers (at least in the DOE ☺)**
  - **brief look at algorithmic prototype: Newton-Krylov-Schwarz**
  - **intuition about how it scales (up to the petascale, at least)**
- **Illustration: “stories from the trenches”**
  - **an undergraduate semester project “gone Broadway”**
  - **community code simulations supporting international magnetically confined fusion energy program (ITER reactor)**

Going implicit?

- Why we *would*, if we *could* :
  1. multiscale problems with good scale separation
  2. coupled problems (“multiphysics”)
  3. problems with uncertain or controllable inputs (optimization: design, control, inversion)
- We *can*, so we *should* !
  1. optimal and scalable algorithms known
  2. freely available software
  3. reasonable learning curve that harvests legacy code

Current focus on *Jacobian-free* implicit methods

- Jacobian a steep price, in terms of coding
  - very valuable to have, but not necessary
  - approximations thereto often sufficient
  - meanwhile, automatic differentiation tools are lowering the threshold
- *Two* stories to track in supercomputing
  - raise the peak capability
  - lower the entry threshold



higher capability for hero users

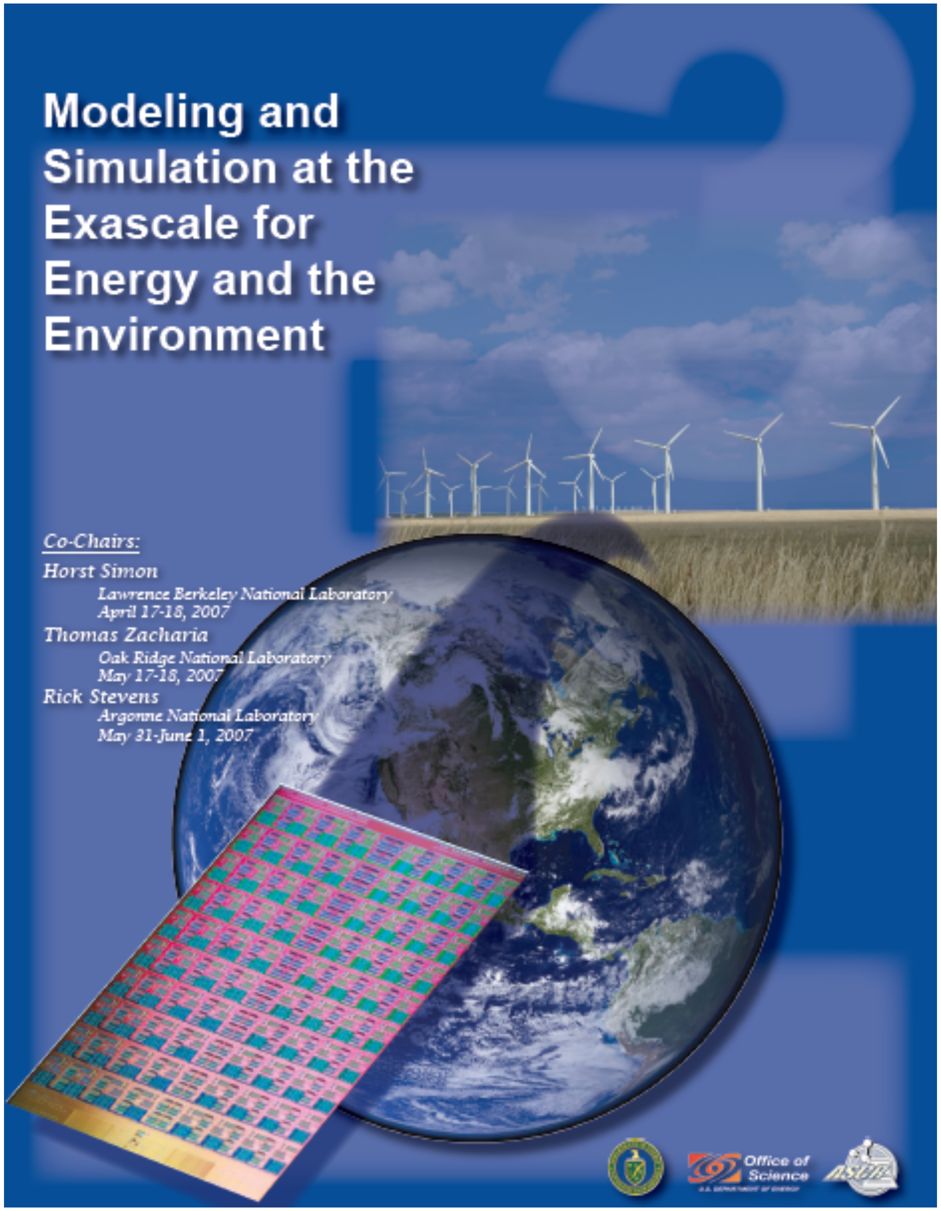
first frontier

"new" frontier

best practices for all users

Tianhe 1A (#1 on the Top 500)

Recent "E^3" report highlights limitations of explicit methods



Modeling and Simulation at the Exascale for Energy and the Environment

Co-Chairs:

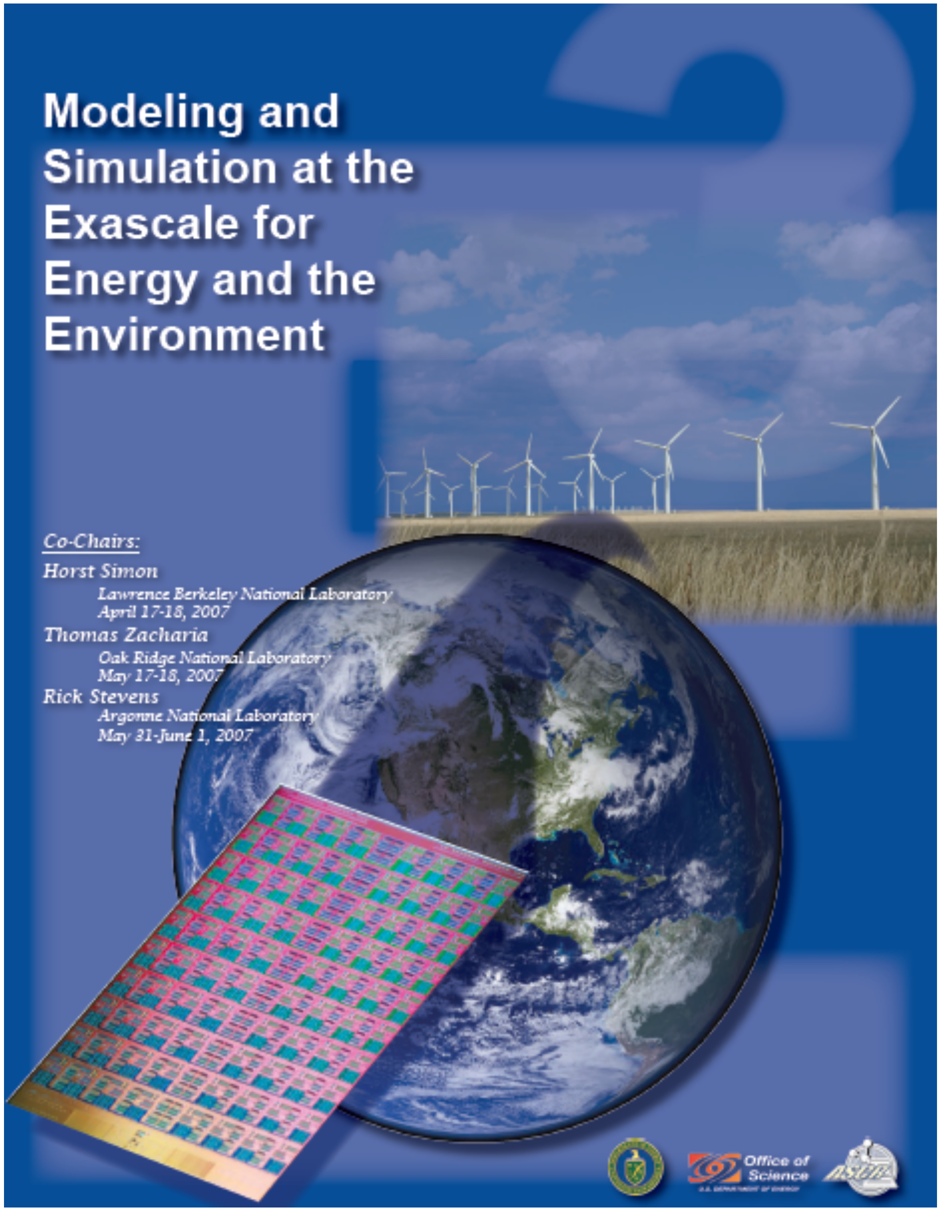
Horst Simon
Lawrence Berkeley National Laboratory
April 17-18, 2007

Thomas Zacharia
Oak Ridge National Laboratory
May 17-18, 2007

Rick Stevens
Argonne National Laboratory
May 31-June 1, 2007

"The dominant computational solution strategy over the past 30 years has been the use of *first-order-accurate operator-splitting, semi-implicit and explicit time integration methods, and decoupled nonlinear solution strategies*. Such methods have not provided the stability properties needed to perform accurate simulations over the dynamical time-scales of interest. Moreover, in most cases, numerical errors and means for controlling such errors are understood heuristically at best."

Recent E^3 report highlights opportunities for implicit methods



Modeling and Simulation at the Exascale for Energy and the Environment

Co-Chairs:

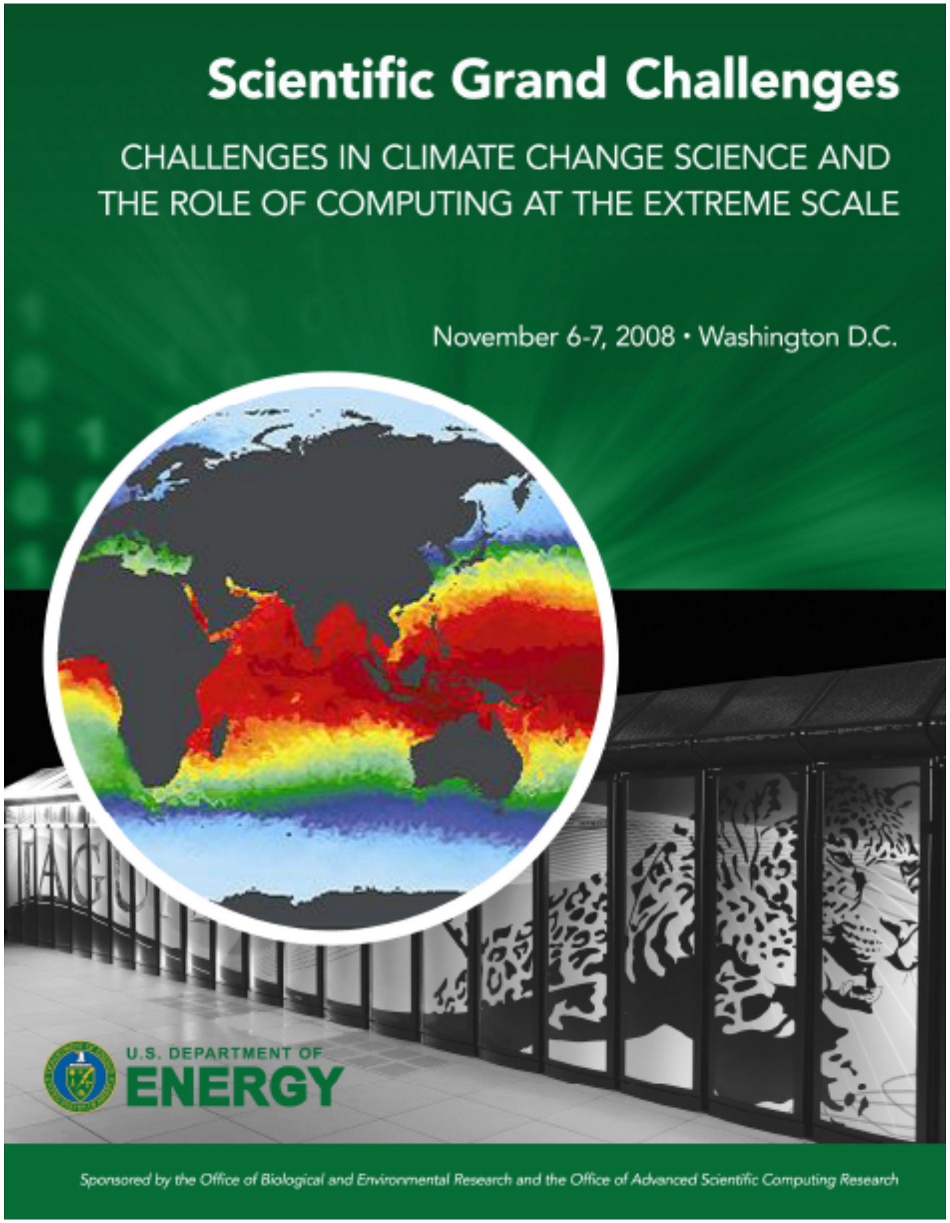
Horst Simon
Lawrence Berkeley National Laboratory
April 17-18, 2007

Thomas Zacharia
Oak Ridge National Laboratory
May 17-18, 2007

Rick Stevens
Argonne National Laboratory
May 31-June 1, 2007

“Research in *linear and nonlinear solvers* remains a critical focus area because the solvers provide the foundation for more advanced solution methods. In fact, as modeling becomes more sophisticated to include, increasingly, optimization, uncertainty quantification, perturbation analysis, and more, the speed and robustness of the linear and nonlinear solvers will directly determine the scope of feasible problems to be solved.”

Many of the eight new "extreme scale" reports identifies implicitness as a priority*



Scientific Grand Challenges

CHALLENGES IN CLIMATE CHANGE SCIENCE AND THE ROLE OF COMPUTING AT THE EXTREME SCALE

November 6-7, 2008 • Washington D.C.

U.S. DEPARTMENT OF ENERGY

Sponsored by the Office of Biological and Environmental Research and the Office of Advanced Scientific Computing Research

2009

"The following priority research direction [was] identified: develop scalable algorithms for non-hydrostatic atmospheric dynamics with quasi-uniform grids, *implicit formulations*, and adaptive and multiscale and multiphysics coupling… Improvements in scalability alone will not be sufficient to obtain the needed throughput (the time it takes to complete a climate simulation). *Obtaining the needed level of throughput will also require incorporating as much implicitness as possible …*"

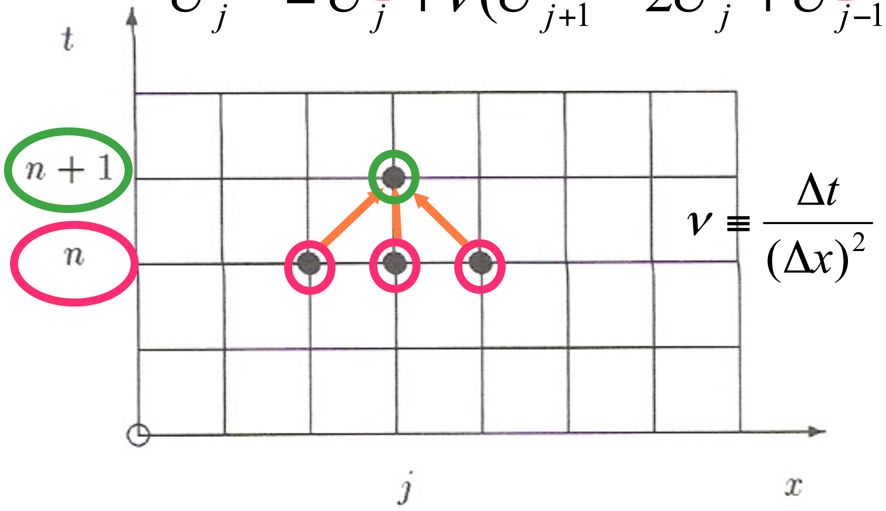
*Extreme scale fusion report is even more forceful

"Explicit" versus "implicit"

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

- ***Explicit*** **methods evaluate a function of state data at prior time, to update each component of the current state independently**
  - **equivalent to matrix-vector multiplication, in linear problems**

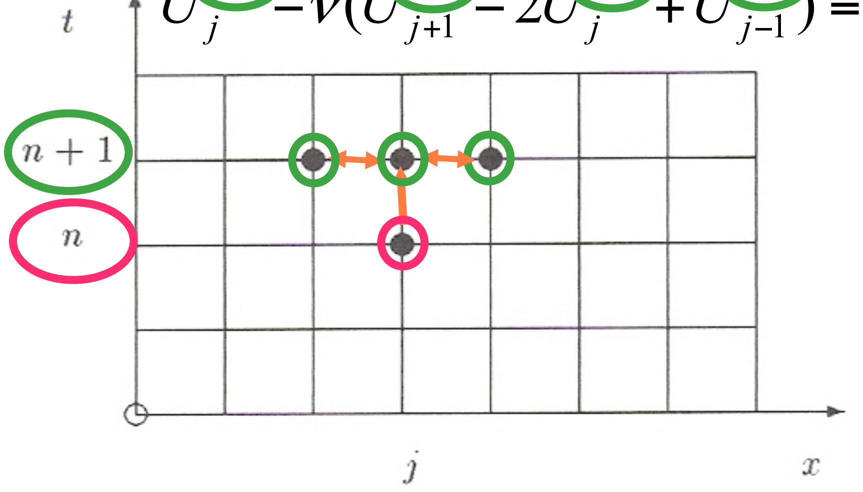
$$U_j^{n+1} = U_j^n + \nu(U_{j+1}^n - 2U_j^n + U_{j-1}^n)$$



$$\nu \equiv \frac{\Delta t}{(\Delta x)^2}$$

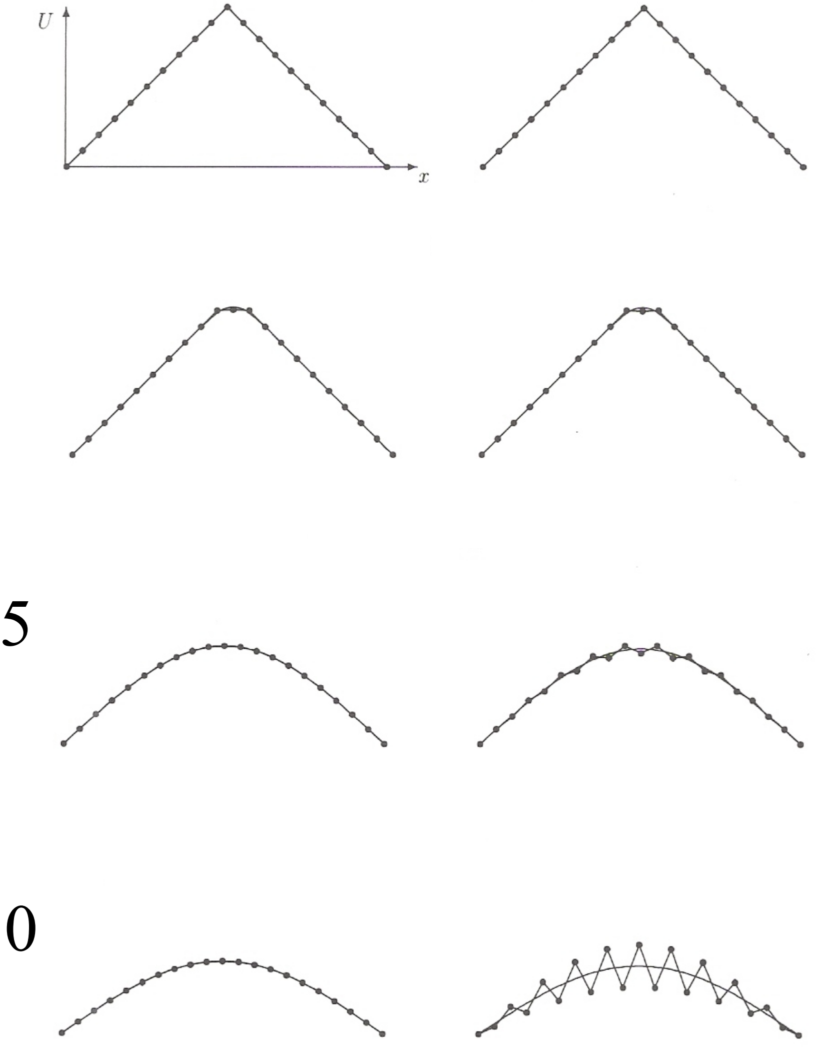
- ***Implicit*** **methods solve a function of state data at the current time, to update all components simultaneously**
  - **equivalent to inverting a matrix, in linear problems**

$$U_j^{n+1} - \nu(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}) = U_j^n$$



Explicit methods can be unstable – linear example

$\Delta t = 0.0012$ $\Delta t = 0.0013$



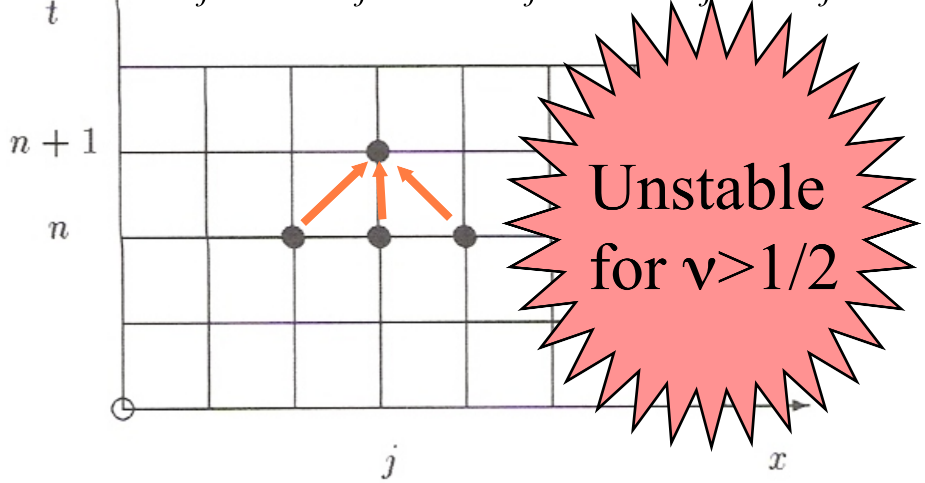
initial data

after 1 step

after 25 steps

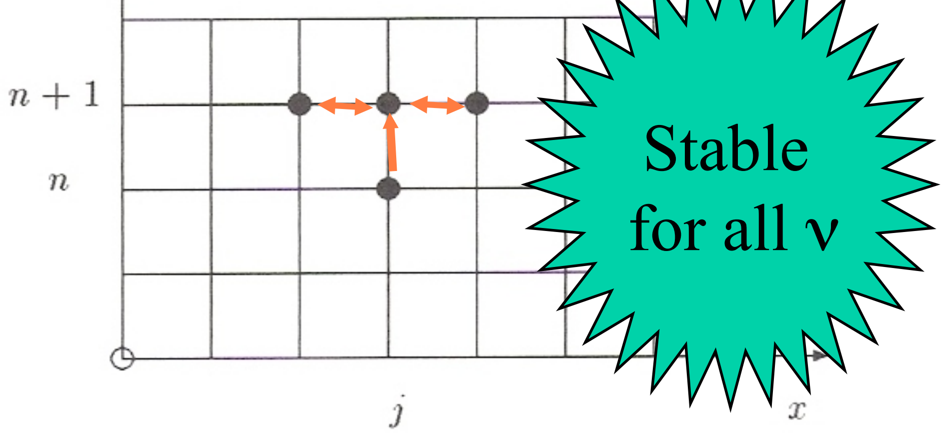
after 50 steps

$$U_j^{n+1} = U_j^n + \nu(U_{j+1}^n - 2U_j^n + U_{j-1}^n)$$



Unstable for ν>1/2

$$U_j^{n+1} - \nu(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}) = U_j^n$$



Stable for all ν

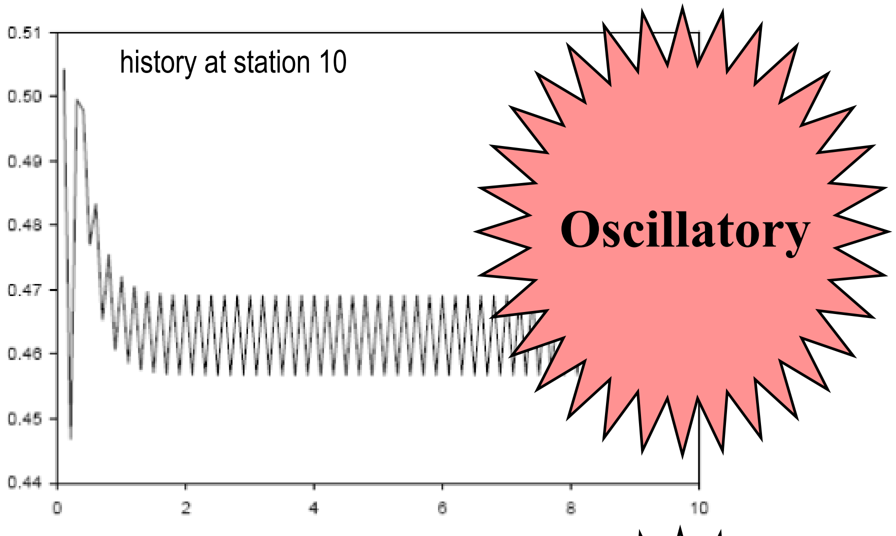
c/o K. Morton & D. Mayers, 2005

Explicit methods can be unphysically oscillatory – nonlinear example ("profile stiffness")

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(\alpha(x,u_x)\frac{\partial u}{\partial x}\right); \quad \alpha \equiv x\kappa(u_x); \quad \kappa(s) \equiv \begin{cases} \kappa_0, & |s| \le s_{crit} \\ \kappa_0 + \kappa_1(|s| - s_{crit})^{1/2}, & |s| > s_{crit} \end{cases}$$

Linearly implicit, nonlinearly explicit:

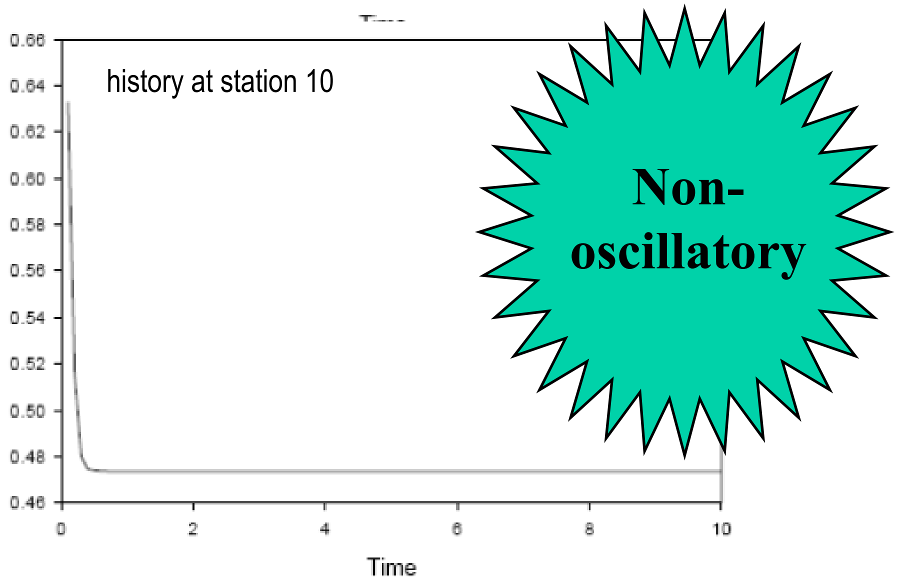
$$U_j^{n+1} - \nu[\boxed{\alpha^n_{j+1/2}}(U_{j+1}^{n+1} - U_j^{n+1}) - \boxed{\alpha^n_{j-1/2}}(U_j^{n+1} - U_{j-1}^{n+1})] = U_j^n$$



Oscillatory

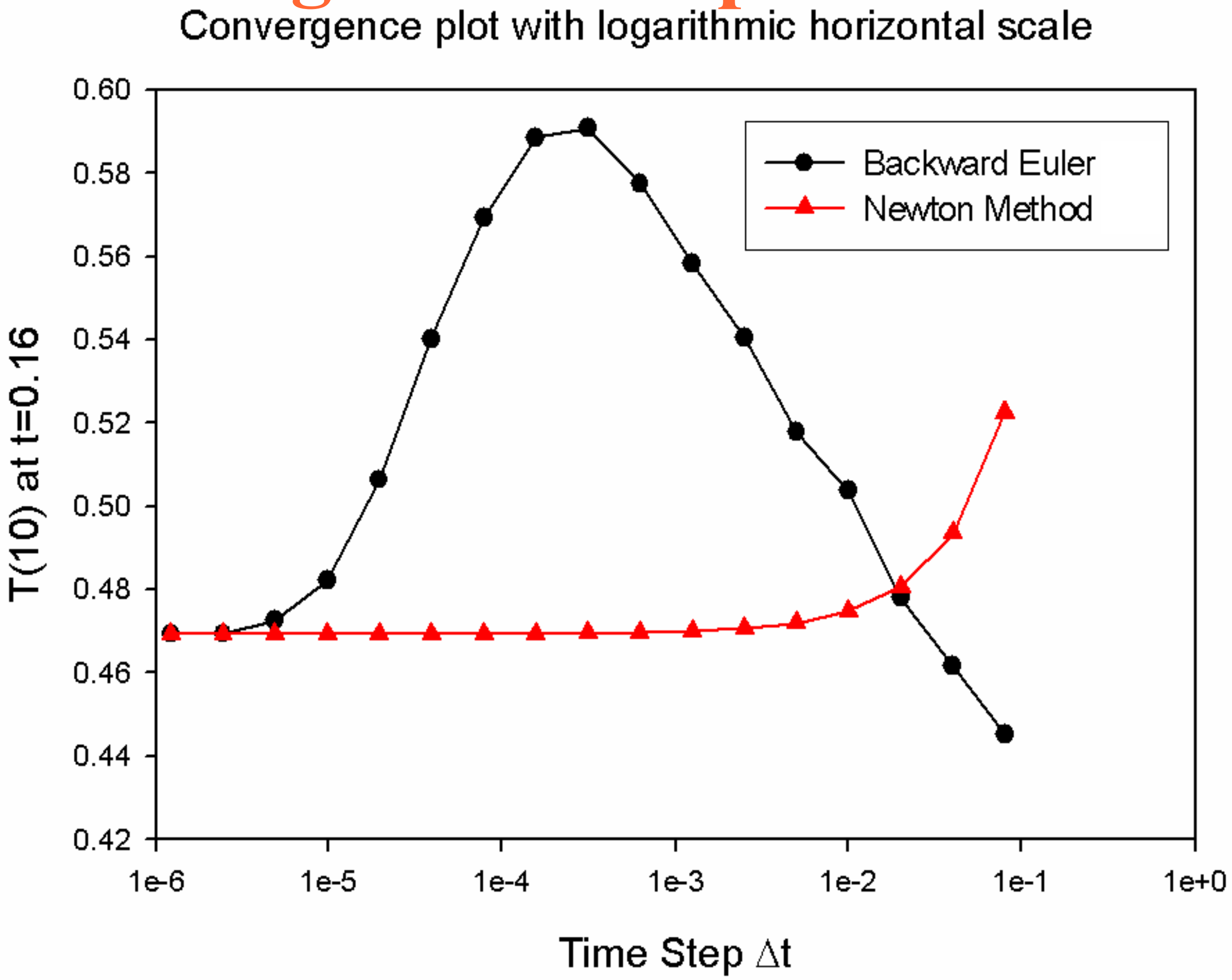
Linearly *and* nonlinearly implicit:

$$U_j^{n+1} - \nu[\boxed{\alpha^{n+1}_{j+1/2}}(U_{j+1}^{n+1} - U_j^{n+1}) - \boxed{\alpha^{n+1}_{j-1/2}}(U_j^{n+1} - U_{j-1}^{n+1})] = U_j^n$$



Non-oscillatory

Timesteps for equivalent accuracy – GLF23 with gradient-dependent diffusivity



Example from fusion collaboration: for sufficiently small timestep, the nonlinearly implicit and linearly implicit with lagged diffusivity converge on the same result, but the nonlinear implicit permits timesteps 10^4 times larger with same accuracy

c/o Steve Jardin, PPPL

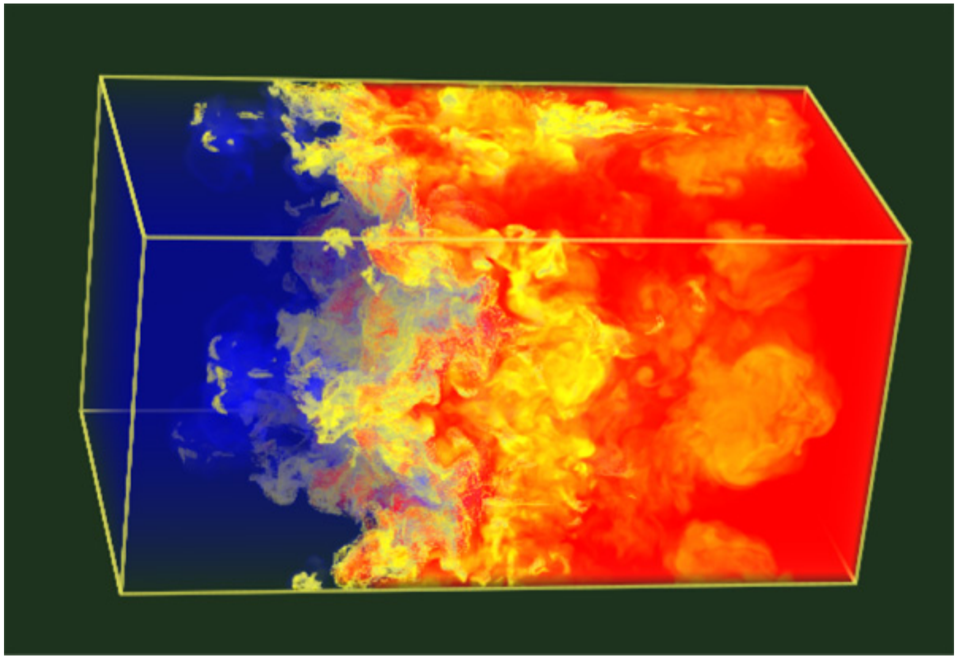
However – implicit methods can be unruly and expensive

	Explicit	Naïve Implicit
Reliability	robust when stable	uncertain
Performance	predictable	data-dependent
Concurrency	$O(N)$	limited
Synchronization	once per step	many times per step
Communication	nearest neighbor*	global, in principle
Workspace	$O(N)$	$O(N^{w})$, e.g., w=5/3
Complexity	$O(N)$	$O(N^{c})$, e.g., c=7/3

* plus the estimation of the stable step size

Motivation #1: Many simulation opportunities are multiscale

- Multiple spatial scales
  - interfaces, fronts, layers
  - thin relative to domain size, $\delta << L$
- Multiple temporal scales
  - fast waves
  - small transit times relative to convection or diffusion, $\tau << T$



Richtmeyer-Meshkov instability, c/o A. Mirin, LLNL

- Analyst must *isolate dynamics of interest* and *model the rest* in a system that can be discretized over more modest range of scales
- Often involves filtering of high frequency modes, quasi-equilibrium assumptions, etc.
- May lead to infinitely "stiff" subsystem requiring implicit treatment

Examples of scale-separated features of multiscale problems

- Gravity surface waves in global climate
- Alfvén waves in tokamaks
- Acoustic waves in aerodynamics
- Fast transients in detailed kinetics chemical reaction
- Bond vibrations in protein folding (?)

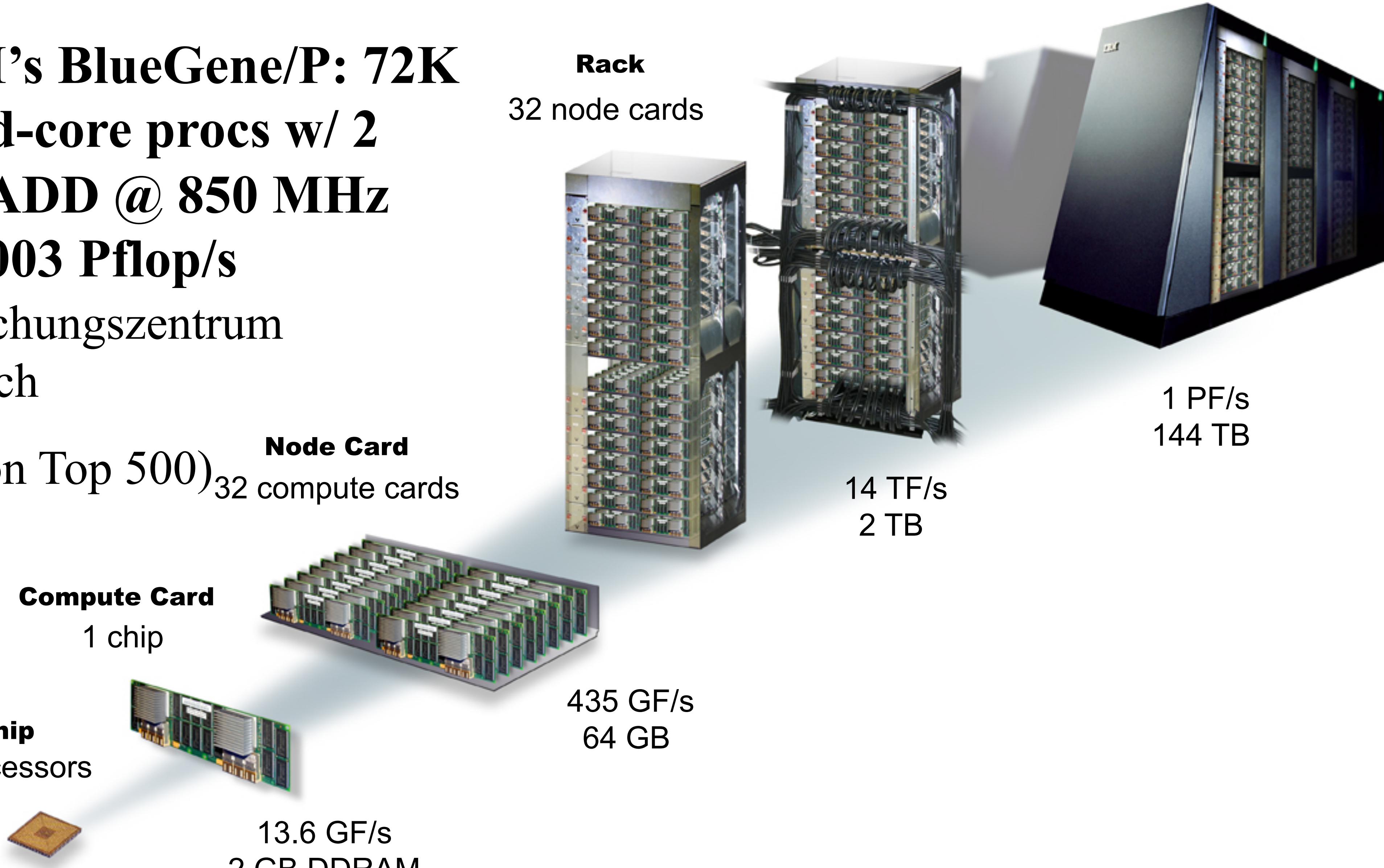
Explicit methods are restricted to marching out the long-scale dynamics on short scales. Implicit methods can "step over" or "filter out" with equilibrium assumptions the dynamically irrelevant short scales, ignoring stability bounds. (Accuracy bounds must still be satisfied; for long time steps, one can use high-order temporal integration schemes!)

What's "big iron" for, if not multiscale?

IBM's BlueGene/P: 72K quad-core procs w/ 2 FMADD @ 850 MHz = 1.003 Pflop/s

Forschungszentrum Jeulich

(#9 on Top 500)



Chip
4 processors

13.6 GF/s
8 MB EDRAM

Compute Card
1 chip

13.6 GF/s
2 GB DDRAM

Node Card
32 compute cards

435 GF/s
64 GB

Rack
32 node cards

14 TF/s
2 TB

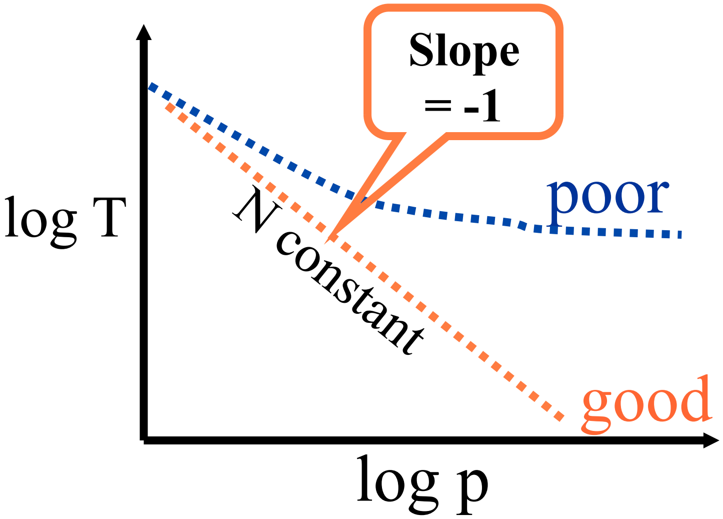
System
72 racks

1 PF/s
144 TB

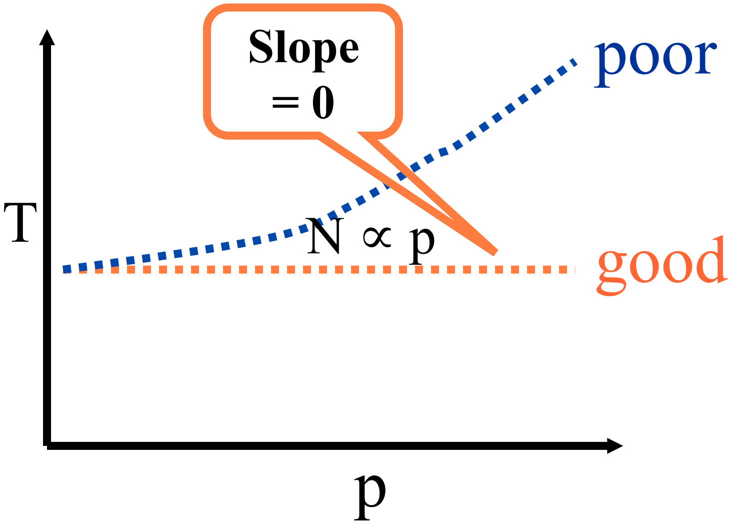
Thread concurrency: 288K (or 294,912) processors

Review: two definitions of scalability

- **"Strong scaling"**
  - **execution time (*T*) decreases in inverse proportion to the number of processors (*p*)**
  - ***fixed size problem* (*N*) *overall***
  - **often instead graphed as reciprocal, "speedup"**

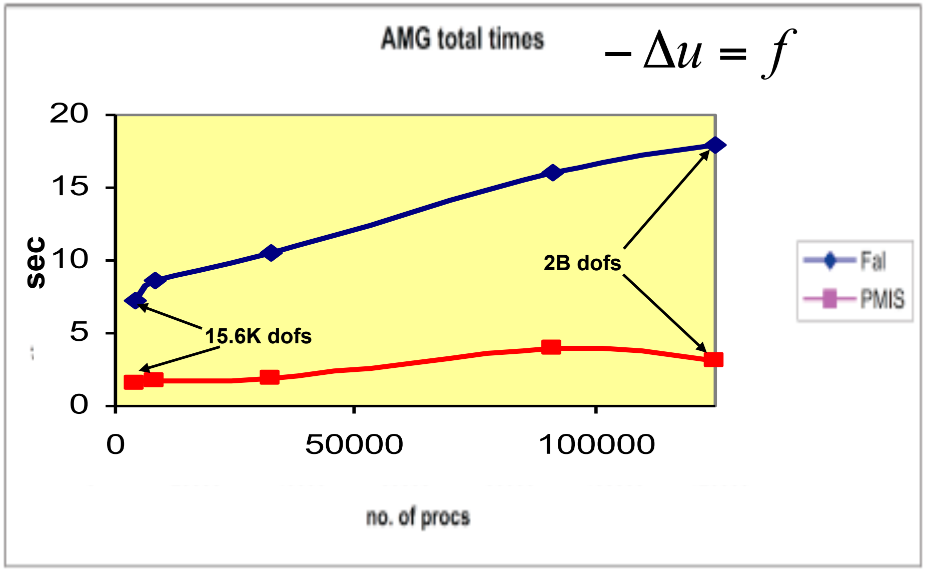


- **"Weak scaling" (memory bound)**
  - **execution time remains constant, as problem size and processor number are increased in proportion**
  - ***fixed size problem per processor***
  - **also known as "Gustafson scaling"**



Solvers *are* scaling: algebraic multigrid (AMG) on BG/L (hypre)

- Algebraic multigrid a key algorithmic technology
  - Discrete operator defined for finest grid by the application, itself, *and* for many recursively derived levels with successively fewer degrees of freedom, for solver purposes
  - Unlike geometric multigrid, AMG not restricted to problems with “natural” coarsenings derived from grid alone
- Optimality (cost per cycle) intimately tied to the ability to coarsen aggressively
- Convergence scalability (number of cycles) and parallel efficiency also sensitive to rate of coarsening
- While much research and development remains, multigrid will clearly be practical at BG/P-scale concurrency



7-pt Laplacian, total execution time, AMG-CG, total problem size ~2 billion

Figure shows weak scaling result for AMG out to 120K processors, with one 25×25×25block per processor (up to 1.875B dofs)

c/o U. M. Yang, LLNL

Explicit methods do not weak scale!

- 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

*assuming dynamics needs to be followed only on coarse scales

d-dimensional domain, length scale L

$d+1$-dimensional space-time, time scale T

h computational mesh cell size

τ computational time step size

$\tau=O(h^\alpha)$ stability 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

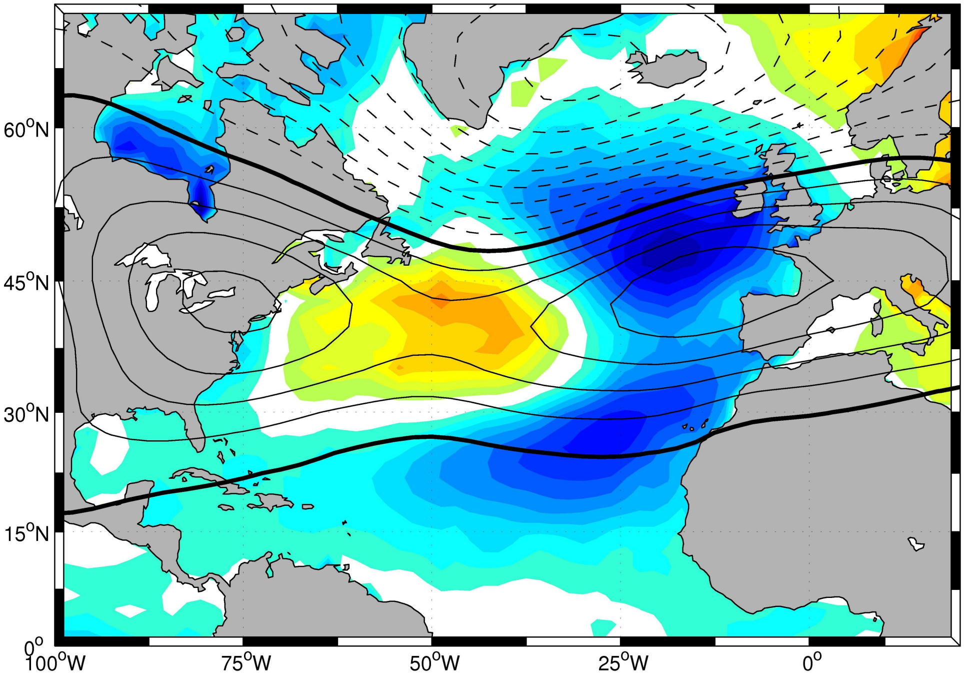
$\propto (T/\tau)(PS)/P \propto T\, S^{1+\alpha/d} P^{\alpha/d}$

(since $\tau \propto h^\alpha \propto 1/n^\alpha = 1/N^{\alpha/d} = 1/(PS)^{\alpha/d}$)

“blackboard”

Motivation #2: Many simulation opportunities are multiphysics

- **Interfacial coupling**
  - Ocean-atmosphere coupling in climate
  - Core-edge coupling in tokamaks
  - Fluid-structure vibrations in aerodynamics
  - Boundary layer-bulk phenomena in fluids
  - Surface-bulk phenomena in solids



SST Anomalies, c/o A. Czaja, MIT

- **Bulk-bulk coupling**
  - Radiation-hydrodynamics
  - Magneto-hydrodynamics

- **Coupled systems may admit destabilizing modes not present in either system alone**

Operator splitting can destabilize multiphysics

- **Model problem** $\dot{u} = -\lambda u + u^2, \quad u(0) = u_0, \quad t > 0$
- **Exact solution** $u(t) = \dfrac{u_0 \exp(-\lambda t)}{1 + \frac{u_0}{\lambda}(\exp(-\lambda t) - 1)}$

Well defined for all time if $\lambda > u_0$

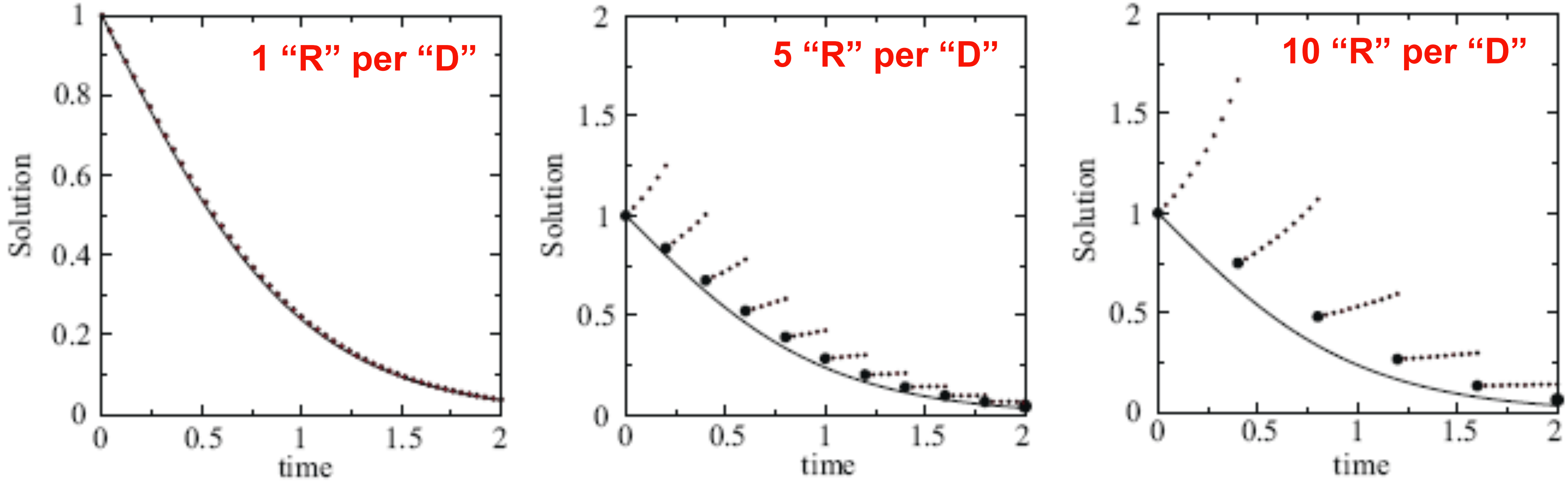
- **Numerical approx.** $U_k \approx u(t_k), \quad t_k = k\Delta t, \quad k = 0, 1, \ldots$
- **Phase 1 ("R")** $\dot{u}_R = u_R^2, \quad u_R(t_k) = U_k, \quad t_k < t \le t_{k+1}$
- **Phase 2 ("D")** $\dot{u}_D = -\lambda u_D, \quad u_D(t_k) = u_R(t_{k+1}), \quad t_k < t \le t_{k+1}$
- **Overall advance** $U_{k+1} = u_D(t_{k+1})$

- **Phase 1 solution** $u_R(t) = \dfrac{U_k}{1 - U_k(t - t_k)}$
- **Phase 2 solution** $u_D(t) = u_R(t_{k+1}) \exp(-\lambda(t - t_k))$
- **Overall advance** $U_{k+1} = U_k \dfrac{\exp(-\lambda \Delta t)}{1 - U_k \Delta t}$

Can blow up in finite time!

Operator splitting can destabilize multiphysics

- Example from *Estep et al.* (2007), $\lambda = 2,\ u_0 = 1$
- 50 time steps, phase 1 subcycled inside phase 2



$$\dot{u} + \lambda u = u^2, \quad u(0) = u_0, \quad t > 0$$

$$u(t) = \frac{u_0 \exp(-\lambda t)}{1 + \frac{u_0}{\lambda}(\exp(-\lambda t) - 1)}$$

$$u_R(t) = \frac{U_k}{1 - U_k(t - t_k)}$$

$$u_D(t) = u_R(t_{k+1}) \exp(-\lambda(t - t_k))$$

$$U_{k+1} = \frac{U_k}{1 - U_k \Delta t} \exp(-\lambda\, \Delta t)$$

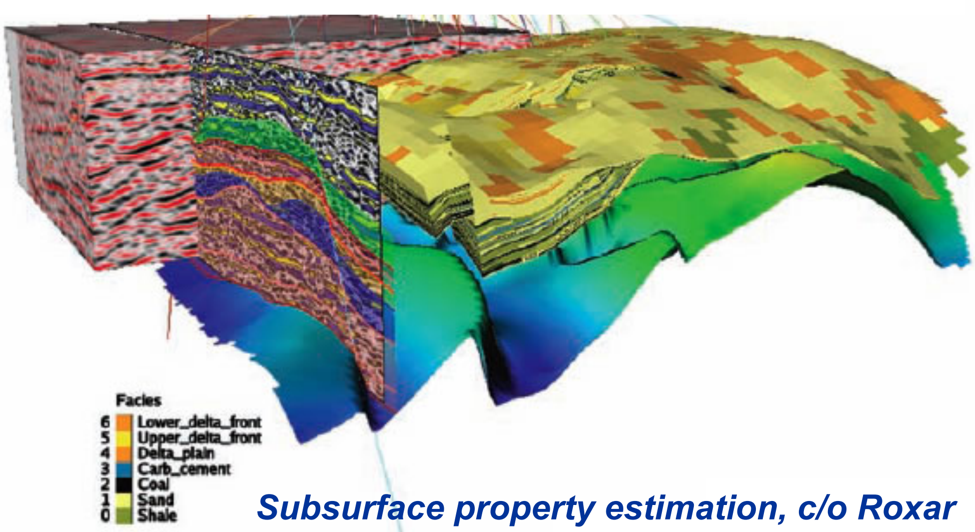
This is a prototype for a reaction-diffusion PDE

$$u_t - au_{xx} = u^2, \quad u(0,x) = u_0(x), \quad t > 0$$

- Diffusive time-scale is constant in time (for each wave number), whereas reactive time-scale changes with solution magnitude
- Besides opening the possibility of finite-time blow-up for a problem that is well defined for all time, operator splitting leaves a first-order error, independent of integration errors for the two phases
- Splitting a single equation is just the simplest example
- Other types of multiphysics (multiple equations in one domain, multiple domains) similarly treatable (see *D. Estep, et al.* 2007)

Motivation #3:
Many simulation opportunities face uncertainty

- Climate prediction
- Subsurface contaminant transport or petroleum recovery, and seismology
- Medical imaging
- Stellar dynamics, e.g., supernovae
- Nondestructive evaluation of structures



Subsurface property estimation, c/o Roxar

- Uncertainty can be in
  - constitutive laws
  - initial conditions
  - boundary conditions

- Sensitivity, optimization, parameter estimation, boundary control require the ability to apply the inverse action of the Jacobian or its adjoint – available in all Newton-like implicit methods

Adjoints "probe" uncertain problems efficiently

- **"Forward" operator equation** $L u_k = f_k$, k samples
- **Desired functional of solution** $\ell(u) = \langle \ell, u \rangle$
- **Define adjoint operator** $L^* v = g$

$$\langle L^* v, u \rangle \equiv \langle v, Lu \rangle$$

- **If we can solve for *v* given ℓ** $L^* v = \ell$
- **Then desired output …** $\ell(u_k) = \langle \ell, u_k \rangle =$

$$\langle L^* v, u_k \rangle = \langle v, L u_k \rangle =$$

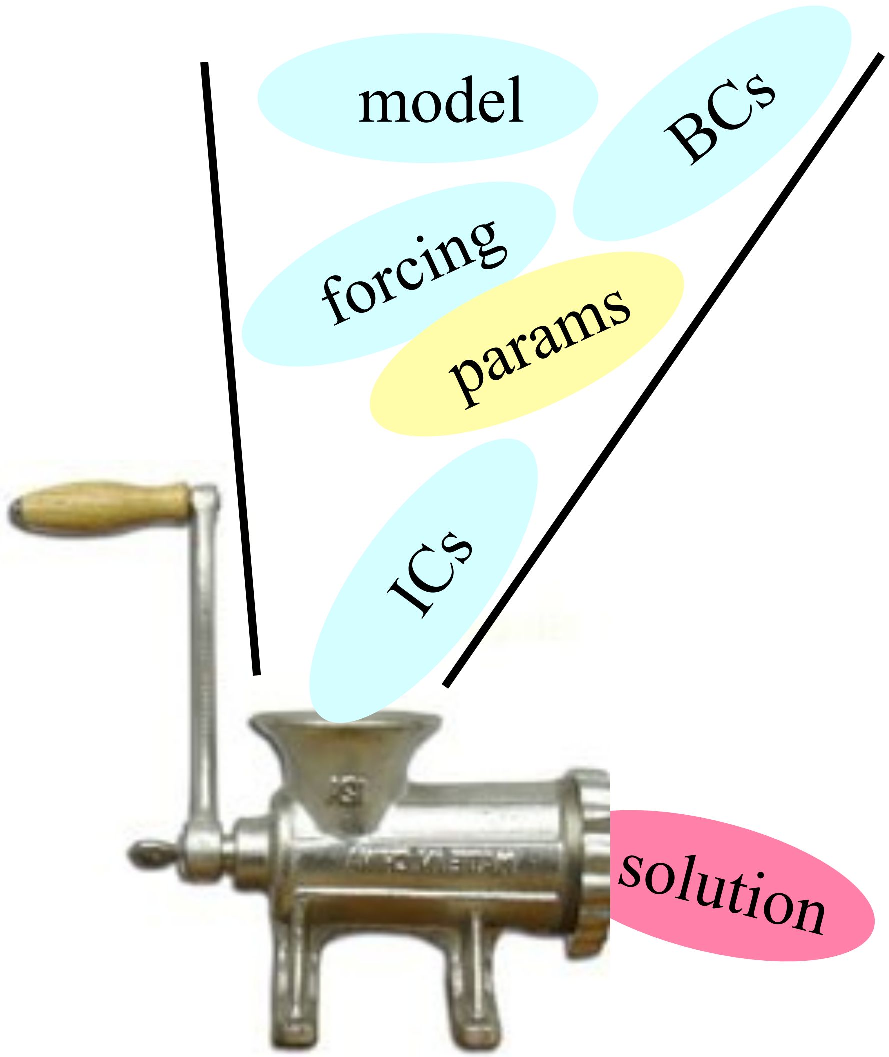
… reduces to an inner product for each forcing f ! $\langle v, f_k \rangle$

Significance and nonlinear generalizations

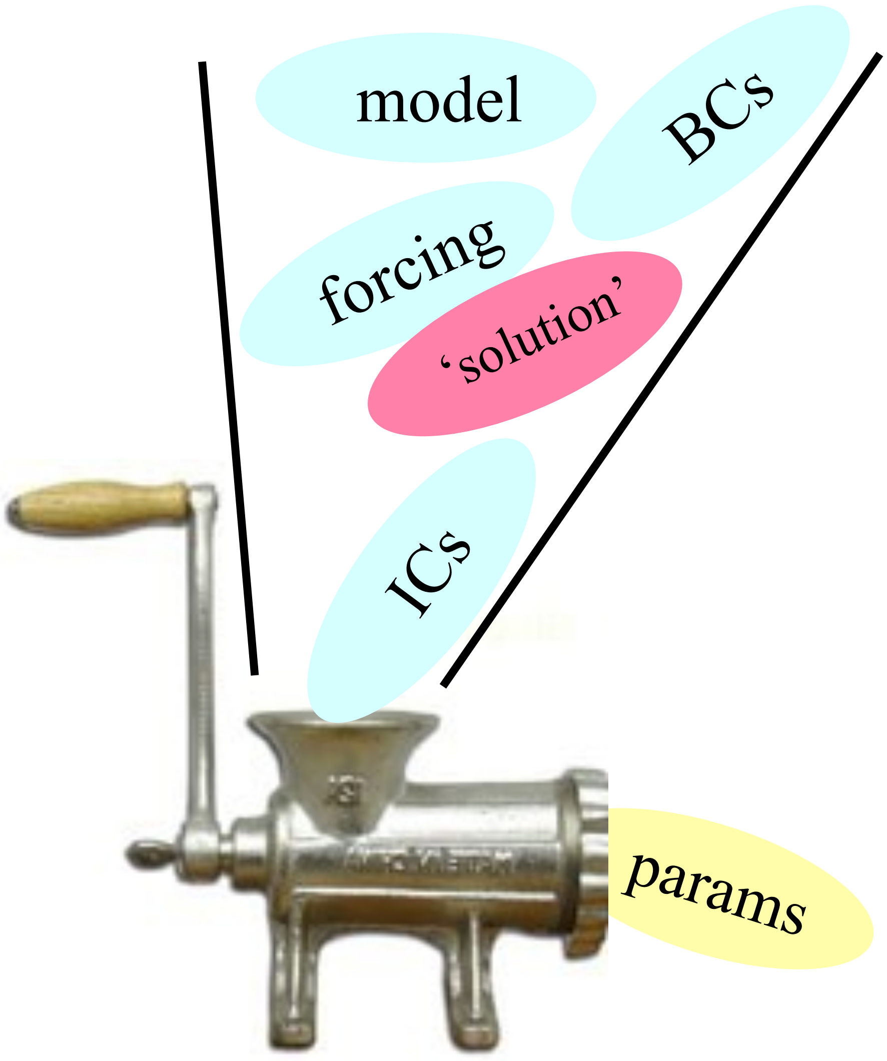
- For *one* solution of the adjoint problem (per output functional desired) one can evaluate *many* outputs per input to the forward problem
  - at a cost of one inner product each
- Otherwise, one would have to solve the forward problem for each input
- Generalization to nonlinear operators is possible, involving local linearizations
- Only price to be paid in coding (ability to solve with linearized adjoint) is often already included in the price paid to take the forward problem implicit
  - Caveat: shortcuts for solving with L not always available for L^*

Forward vs. inverse problems

forward problem



inverse problem



+ regularization

Significance of inverse problems for implicit methods

- Inverse problems can be formulated as PDE-constrained optimization problems
  - objective function (mismatch of model output and “true” output)
  - equality constraints (PDE)
  - possible inequality constraints, in addition
- Cast as nonlinear rootfinding problem
  - Form (augmented) Lagrangian
  - Take gradient of Lagrangian with respect to design variables, state variables, and Lagrange multipliers
  - Obtain large nonlinear rootfinding problem
- Solving with Newton requires Jacobian of gradient, or Hessian of Lagrangian
  - Major blocks are Jacobian of PDE system and its adjoint ☑

Constrained optimization w/Lagrangian

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

$$\Phi_x(x,u) + \lambda^T f_x(x,u) = 0$$

$$\Phi_u(x,u) + \lambda^T f_u(x,u) = 0$$

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

Newton reduced SQP

- Applying Newton’s method leads to the KKT system for states x , designs u , and multipliers λ

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

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

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

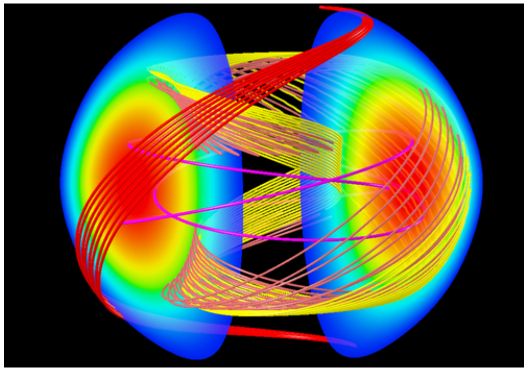
$$g = -g_u + J_u^T J_x^{-T} g_x - (J_u^T J_x^{-T} W_{xx} - W_{ux}) J_x^{-1} f$$

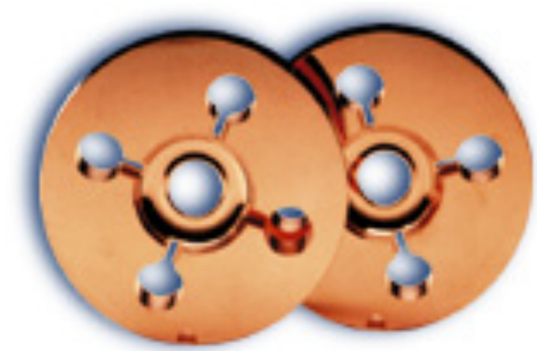
- Then

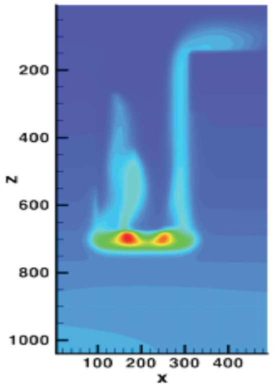
$$J_x \delta x = -f - J_u \delta u$$

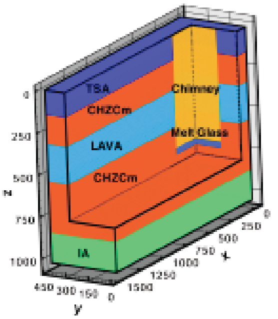
$$J_x^T \delta\lambda = -g_x - W_{xx}\delta x - W_{ux}^T \delta u$$

Applications requiring scalable solvers – conventional and progressive









- **Magnetically confined fusion**
  - **Poisson problems**
  - ***nonlinear coupling of multiple physics codes***
- **Accelerator design**
  - **Maxwell eigenproblems**
  - ***shape optimization subject to PDE constraints***
- **Porous media flow**
  - **div-grad Darcy problems**
  - ***parameter estimation***

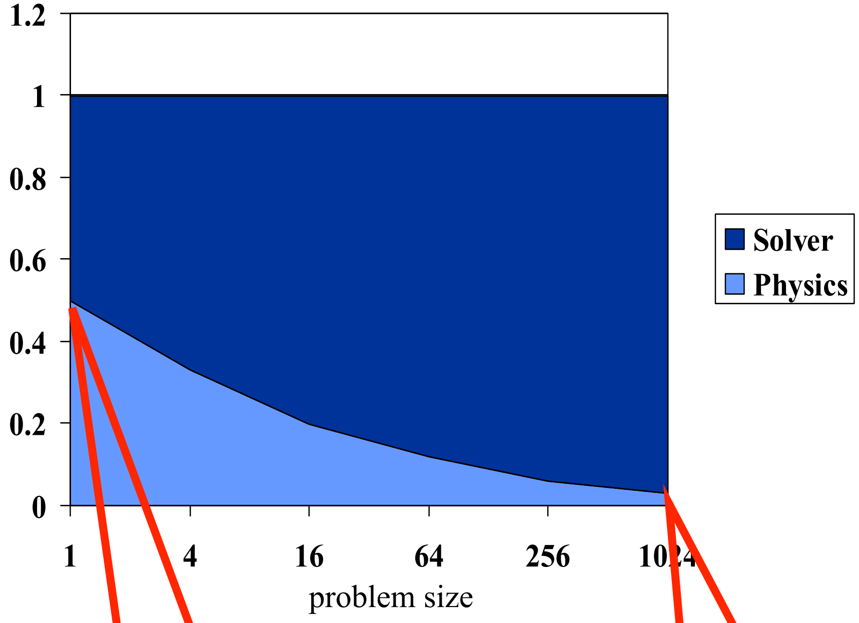
presenting symptoms

actual ailments

It’s *all* about algorithms (at the petascale)

- 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
- Most applications groups have not yet “felt” this curve in their gut
  - as users actually get into queues with more than 4K processors, this will change

Weak scaling limit, assuming efficiency of 100% in both physics and solver phases



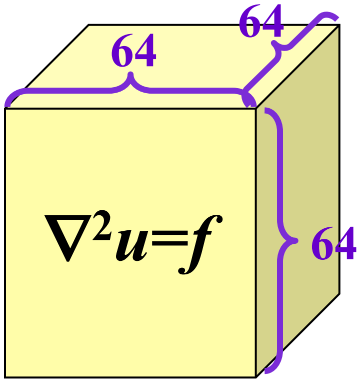
Solver takes 50% time on 128 procs

Solver takes 97% time on 128K procs

Reminder: solvers evolve underneath "$Ax = b$"

- Advances in algorithmic efficiency rival advances in hardware architecture
- Consider Poisson's equation on a cube of size $N=n^3$

Year	*Method*	*Reference*	*Storage*	*Flops*
1947	GE (banded)	Von Neumann & Goldstine	n^5	n^7
1950	Optimal SOR	Young	n^3	$n^4 \log n$
1971	CG-MILU	Reid	n^3	$n^{3.5} \log n$
1984	Full MG	Brandt	n^3	n^3

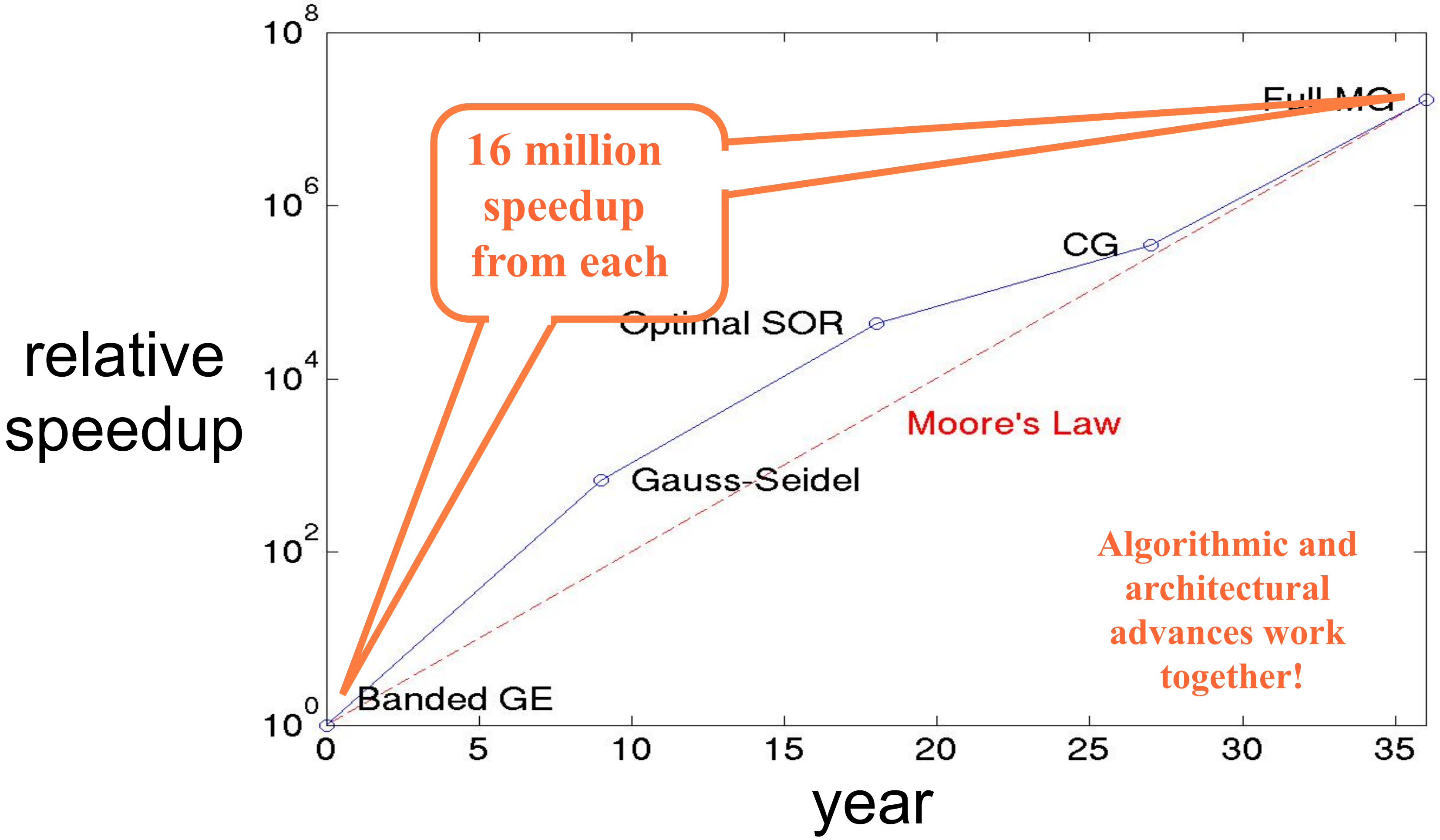


- If n=64, this implies an overall reduction in flops of ~ 16 million*

*Six months is reduced to 1 second

Algorithms and Moore’s Law

- This advance took place over a span of about 36 years, or 24 doubling times for Moore’s Law
- $2^{24} \approx$ 16 million ⇒ the same as the factor from algorithms alone!

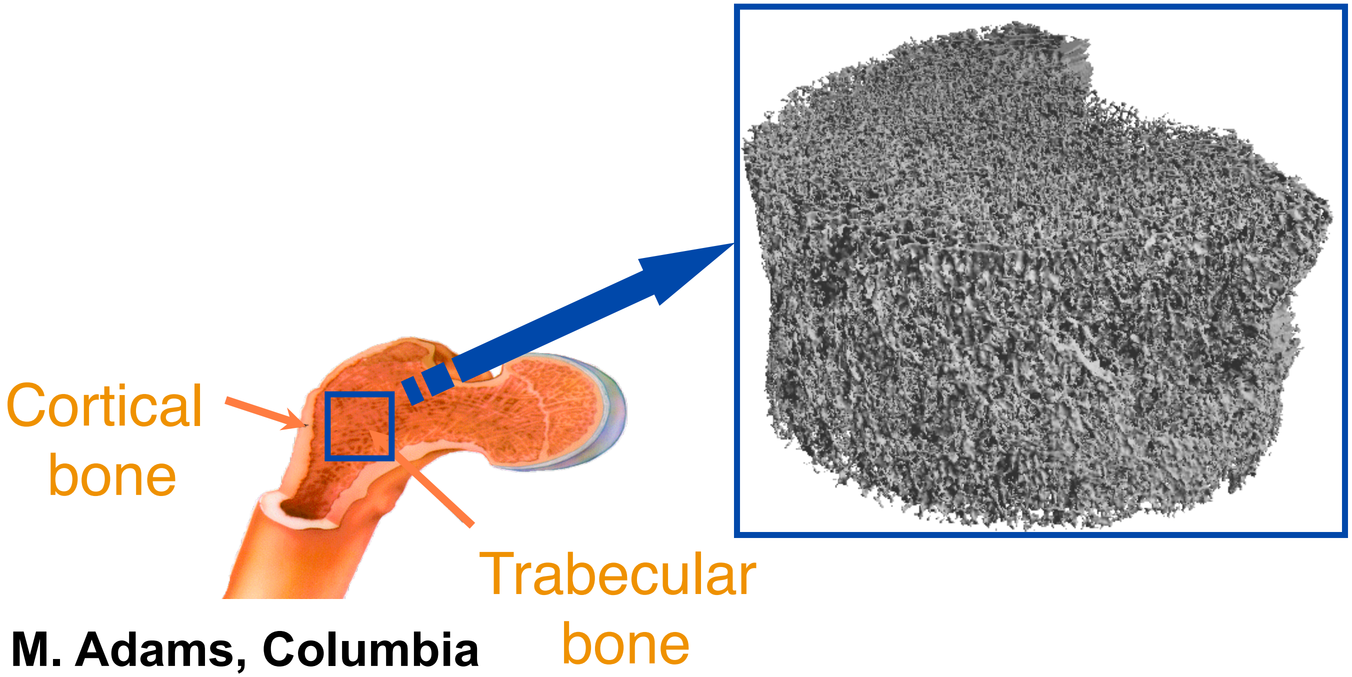


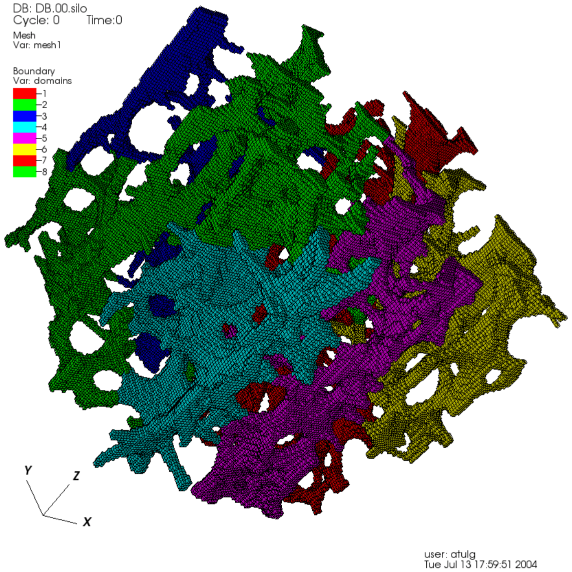
16 million speedup from each

Algorithmic and architectural advances work together!

Implicit methods can scale!
2004 Gordon Bell "special" prize

- **2004 Bell Prize in "special category" went to an implicit, unstructured grid bone mechanics simulation**
  - **0.5 Tflop/s sustained on 4 thousand procs of ASCI White**
  - **large-deformation analysis**
  - **in production in bone mechanics lab**





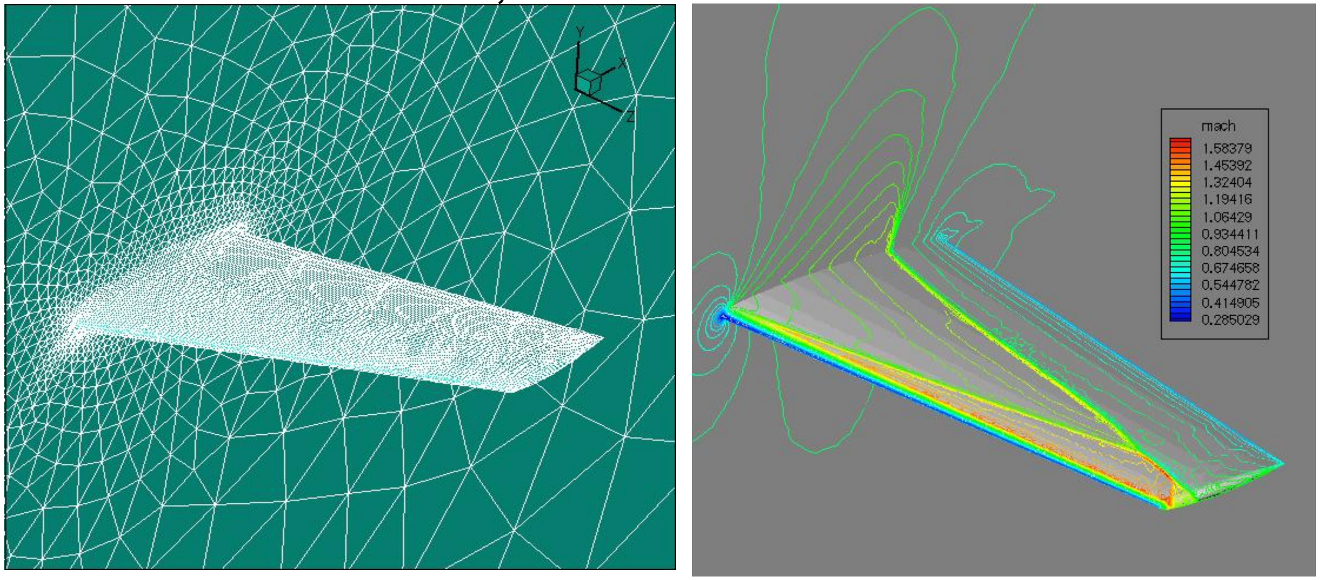
c/o M. Adams, Columbia

TOPS
Terascale Optimal PDE Simulations

Implicit methods can scale!
1999 Gordon Bell "special" prize

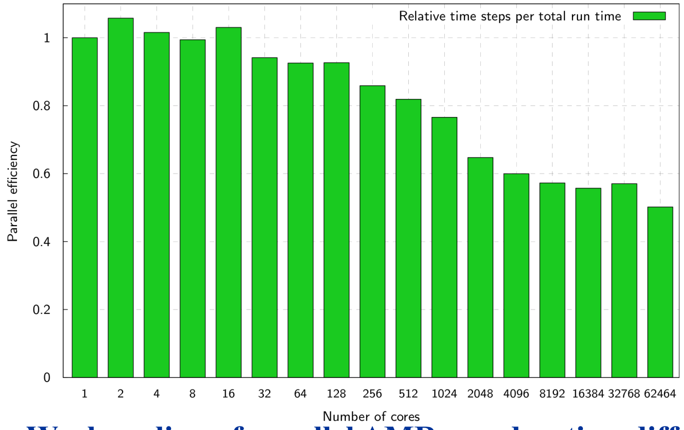
- **1999 Bell Prize in "special category" went to implicit, unstructured grid aerodynamics problems**
  - **0.23 Tflop/s sustained on 3 thousand processors of Intel's ASCI Red**
  - **11 million degrees of freedom**
  - **incompressible and compressible Euler flow**
  - **employed in NASA analysis/design missions**

Transonic "Lambda" Shock, Mach contours on surfaces



Supported in part by DOE SciDAC TOPS, NSF PetaApps/Earth Sciences, and DOE NNSA PSAAP

2008 Gordon Bell finalist: implicit methods *must* scale!



Weak scaling of parallel AMR on advection-diffusion problem with ~131,000 elements per core. Results demonstrate 50% parallel efficiency over a range of 1 to 62,464 cores on TACC's Sun/AMD *Ranger* system. Largest problem has ~7.9B finite elements. AMR imposes <10% overhead over a static mesh solver.

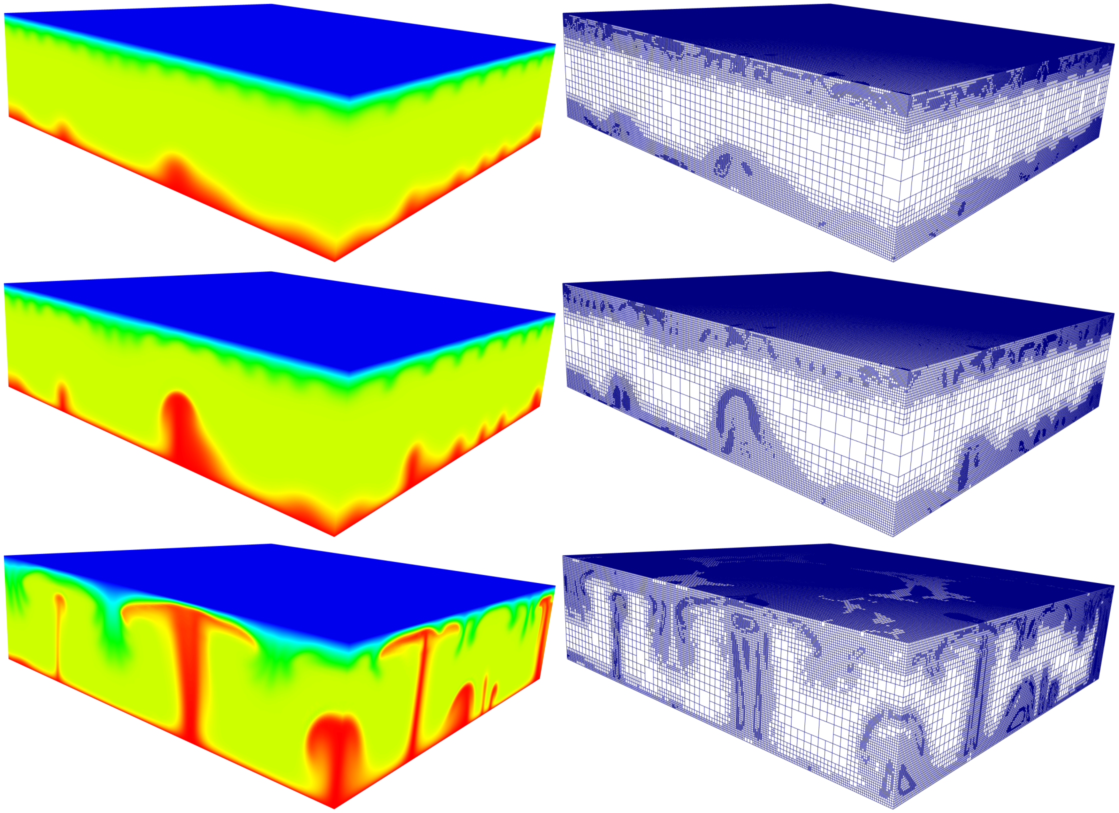
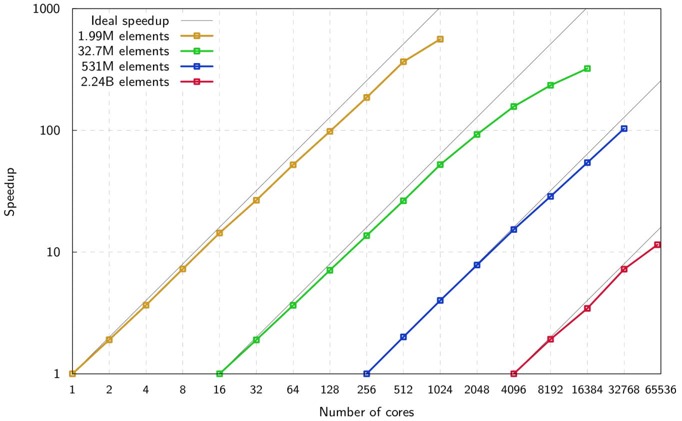


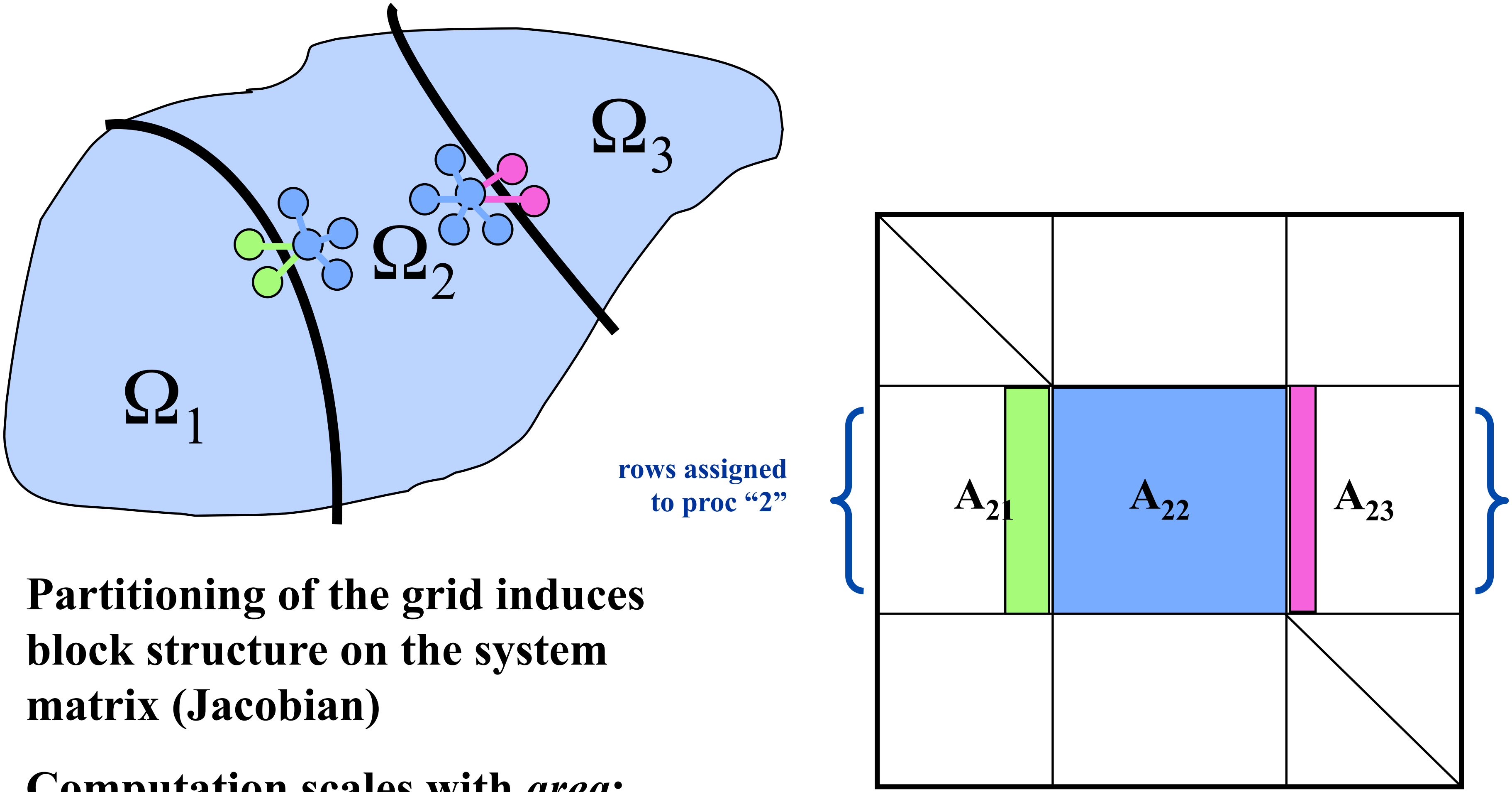
Illustration of parallel dynamic AMR for problem of modeling convection in Earth's mantle. The figure shows snapshots of the thermal field at three time instants (left column) and corresponding adapted meshes (right column). The mesh resolves the rising plumes as well as the instabilities at the top layer. The elements span levels 4 to 9 in octree depth.



Strong scaling of parallel AMR on advection-diffusion problem for small (yellow), medium (green), large (blue), and very large (red) problems sizes. Blue curve demonstrates nearly ideal strong scaling for 0.5B element problem over a range of 256 to 32,768 cores on *Ranger*.

c/o O. Ghattas (UTexas) *et al.*

SPMD parallelism w/domain decomposition puts off limitation of Amdahl in weak scaling

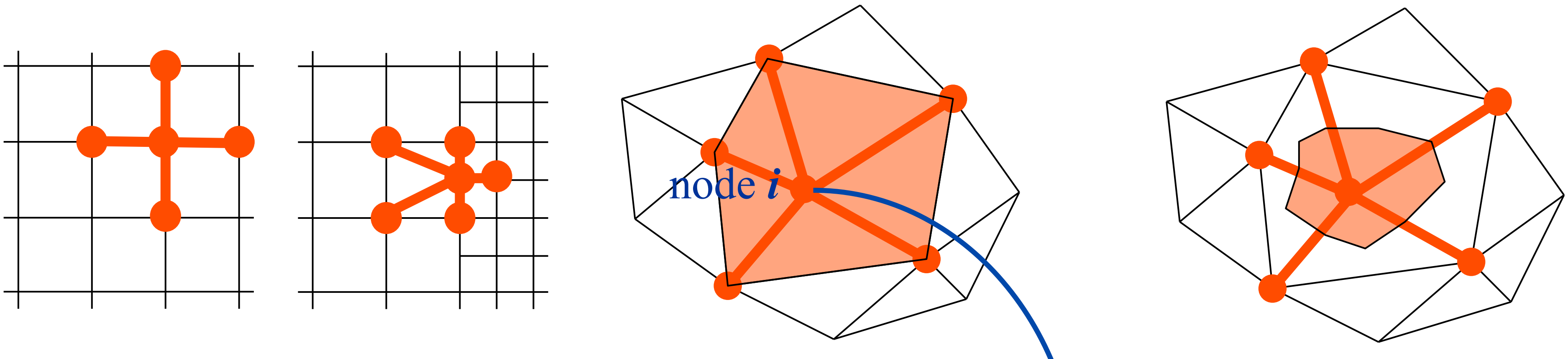


Partitioning of the grid induces block structure on the system matrix (Jacobian)

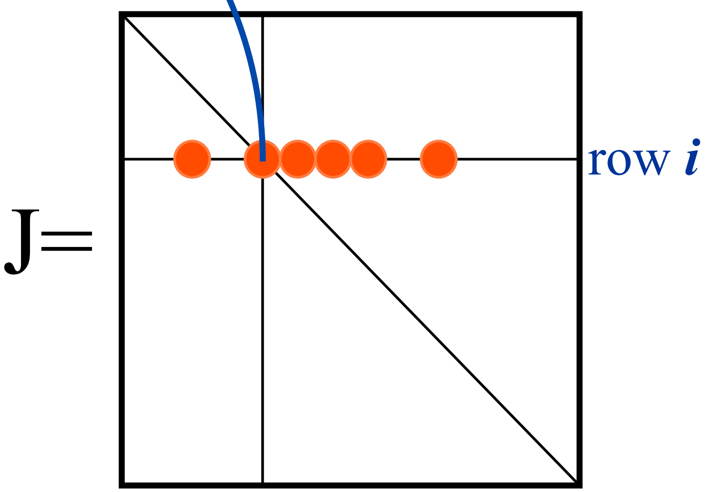
Computation scales with *area*; communication scales with *perimeter*; ratio *fixed* in weak scaling

DD relevant to any local stencil formulation

finite differences   finite elements   finite volumes



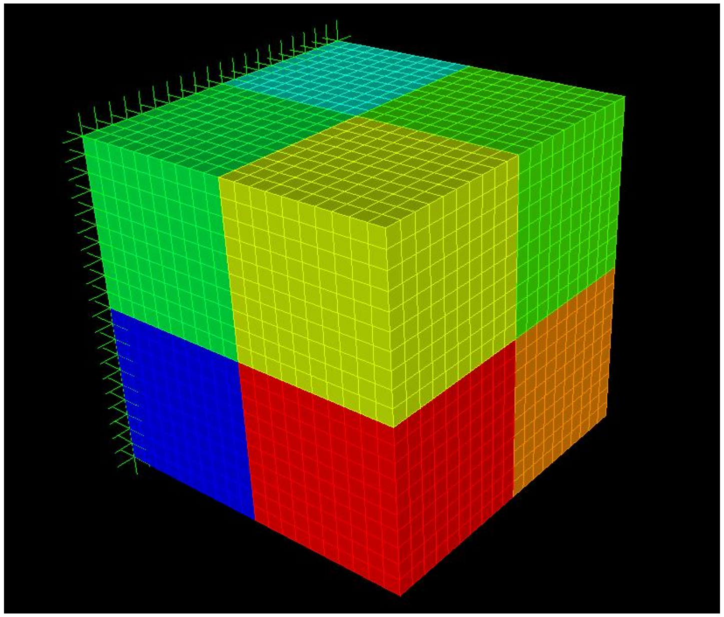
- **lead to sparse Jacobian matrices**
- **however, the inverses are generally dense; even the factors suffer unacceptable fill-in in 3D**
- **want to solve in subdomains only, and use to precondition full sparse problem**



Estimating scalability of stencil computations

- **Given complexity estimates of the leading terms of:**
  - the concurrent computation (per iteration phase)
  - the concurrent communication
  - the synchronization frequency
- **And a bulk synchronous model of the architecture including:**
  - internode communication (network topology and protocol reflecting horizontal memory structure)
  - on-node computation (effective performance parameters including vertical memory structure)
- **One can estimate optimal concurrency and optimal execution time**
  - on per-iteration basis, or overall (by taking into account any granularity-dependent convergence rate)
  - simply differentiate time estimate in terms of *(N,P)* with respect to *P*, equate to zero and solve for *P* in terms of *N*

Estimating 3D stencil costs (per iteration)



- grid points in each direction *n*, total work *N=O(n*^{3}*)*
- processors in each direction *p*, total procs *P=O(p*^{3}*)*
- memory per node requirements *O(N/P)*

- concurrent execution time per iteration $A\, n^3/p^3$
- grid points on side of each processor subdomain *n/p*
- Concurrent neighbor commun. time per iteration $B\, n^2/p^2$
- cost of global reductions in each iteration $C \log p$ or $C\, p^{(1/d)}$
  - *C* includes synchronization frequency
- same dimensionless units for measuring *A, B, C*
  - e.g., cost of scalar floating point multiply-add

3D stencil computation illustration

Rich local network, tree-based global reductions

- **total wall-clock time per iteration**

$$T(n,p) = A\frac{n^3}{p^3} + B\frac{n^2}{p^2} + C\log p$$

- **for optimal p, $\frac{\partial T}{\partial p} = 0$, or** $-3A\frac{n^3}{p^4} - 2B\frac{n^2}{p^3} + \frac{C}{p} = 0,$

or (with $\theta \equiv \frac{32B^3}{243A^2C}$ **),**

$$p_{opt} = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1+(1-\sqrt{\theta})\right]^{1/3} + \left[1-(1-\sqrt{\theta})\right]^{1/3}\right)\cdot n$$

- **without "speeddown,"** p **can grow with** n
- **in the limit as** $B/C \to 0$

$$p_{opt} = \left(\frac{3A}{C}\right)^{1/3}\cdot n$$

3D stencil computation illustration

Rich local network, tree-based global reductions

- **optimal running time**

$$T(n, p_{opt}(n)) = \frac{A}{\rho^3} + \frac{B}{\rho^2} + C\log(\rho n),$$

where

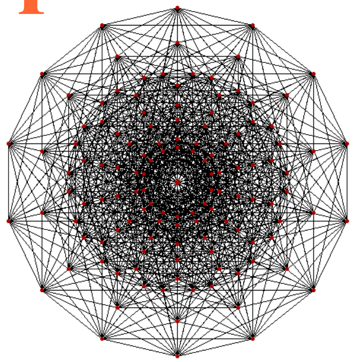
$$\rho = \left(\frac{3A}{2C}\right)^{1/3}\left(\left[1 + (1 - \sqrt{\theta})\right]^{1/3} + \left[1 - (1 - \sqrt{\theta})\right]^{1/3}\right)$$

- **limit of infinite neighbor bandwidth, zero neighbor latency ($B \rightarrow 0$)**

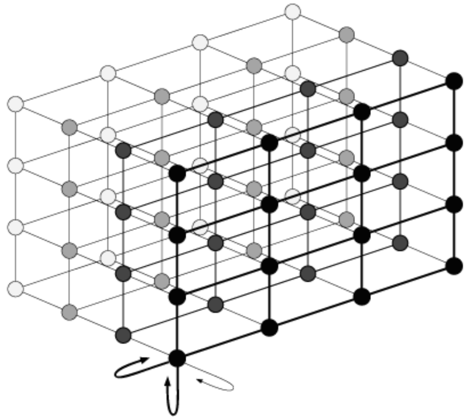
$$T(n, p_{opt}(n)) = C\left[\log n + \frac{1}{3}\log\frac{A}{C} + const.\right]$$

(This analysis is on a per iteration basis; complete analysis multiplies this cost by an iteration count estimate that generally depends on *n* and *p*.)

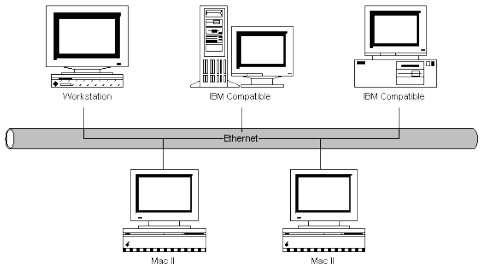
Scalability results for DD stencil computations



- With tree-based (logarithmic) global reductions and scalable nearest neighbor hardware:
  - optimal number of processors scales *linearly* with problem size



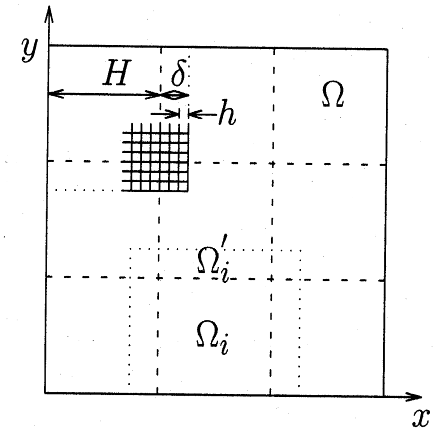
- With 3D torus-based global reductions and scalable nearest neighbor hardware:
  - optimal number of processors scales as *three-fourths* power of problem size (almost "scalable")



- With common network bus (heavy contention):
  - optimal number of processors scales as *one-fourth* power of problem size (not "scalable")

Factoring convergence rate into estimates

- Krylov-Schwarz iterative methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system
- In terms of ***N*** and ***P***, where for ***d***-dimensional isotropic problems, $N=h^{-d}$ and $P=H^{-d}$, for mesh parameter ***h*** and subdomain diameter ***H***, iteration counts may be estimated as follows:



Preconditioning Type	in 2D	in 3D
Point Jacobi	$O(N^{1/2})$	$O(N^{1/3})$
Domain Jacobi (δ=0)	$O((NP)^{1/4})$	$O((NP)^{1/6})$
1-level Additive Schwarz	$O(P^{1/2})$	$O(P^{1/3})$
2-level Additive Schwarz	$O(1)$	$O(1)$

Where do these results come from?

- Point Jacobi result is well known (see any book on the numerical analysis of elliptic problems)
- Subdomain Jacobi result has interesting history
  - Was derived independently from functional analysis, linear algebra, and graph theory
- Schwarz theory is neatly and abstractly summarized in Section 5.2 Smith, Bjorstad & Gropp (1996) and Chapter 2 of Toselli & Widlund (2004)
  - condition number, $\kappa \leq \omega\, [1+\rho(\mathcal{E})]\, C_0^2$
  - C_0^2 is a splitting constant for the subspaces of the decomposition
  - $\rho(\mathcal{E})$ is a measure of the orthogonality of the subspaces
  - ω is a measure of the approximation properties of the subspace solvers (can be unity for exact subdomain solves)
  - These properties are estimated for different subspaces, different operators, and different subspace solvers and the “crank” is turned

Comments on the Schwarz results

- **Original basic Schwarz estimates were for:**
  - *self-adjoint* **elliptic operators**
  - *positive definite* **operators**
  - *exact* **subdomain solves,** A_i^{-1}
  - *two-way* **overlapping with** R_i, R_i^T
  - *generous* **overlap,** $\delta=O(H)$ **(original 2-level result was** $O(1+H/\delta)$**)**
- **Subsequently extended to (within limits):**
  - *nonself-adjointness* **(e.g, convection)**
  - *indefiniteness* **(e.g., wave Helmholtz)**
  - *inexact* **subdomain solves**
  - *one-way* **overlap communication ("restricted additive Schwarz")**
  - *small* **overlap**

Comments on the Schwarz results, cont.

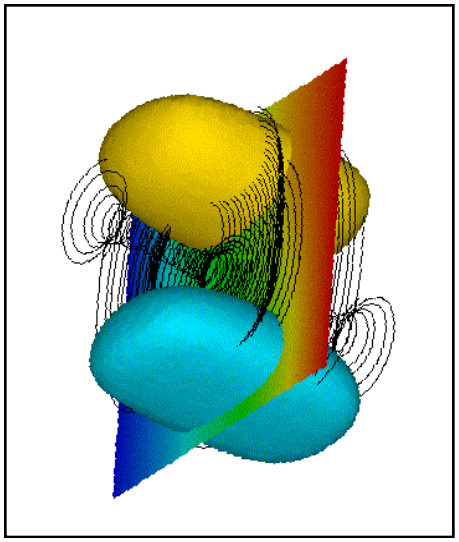
- **2-level theory requires "sufficiently fine" coarse mesh**
  - **However, coarse space need *not* be nested in the fine space or in the decomposition into subdomains**
- **Practice is better than one has any right to expect**

> "In theory, theory and practice are the same ...
> In practice they're not!"
> — *Yogi Berra*

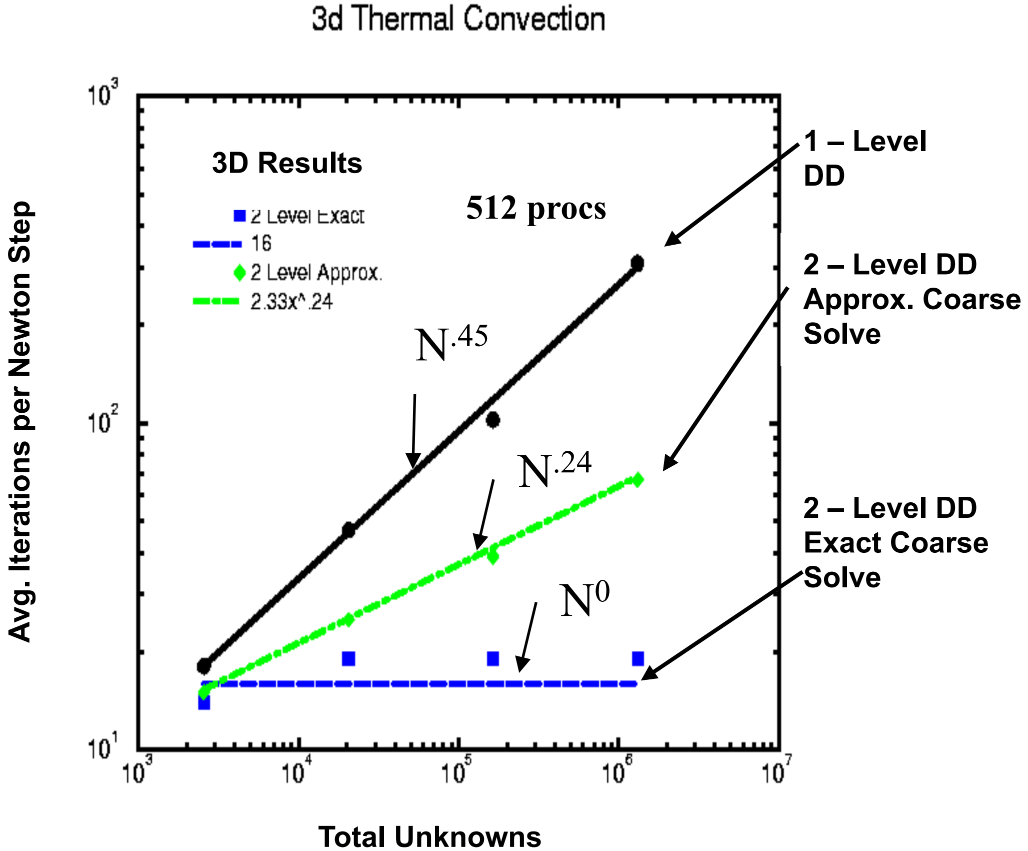
- **Wave Helmholtz (e.g., acoustics) is delicate at high frequency:**
  - **standard Schwarz Dirichlet boundary conditions can lead to undamped resonances within subdomains,** $u_\Gamma = 0$
  - **remedy involves Robin-type transmission boundary conditions on subdomain boundaries,** $(u + \alpha\, \partial u / \partial n)_\Gamma = 0$

Illustration of 1-level vs. 2-level tradeoff

Thermal Convection Problem (Ra = 1000)



Temperature iso-lines on slice plane, velocity iso-surfaces and streamlines in 3D



c/o J. Shadid and R. Tuminaro

Newton-Krylov solver with Aztec non-restarted GMRES with 1-level domain decomposition preconditioner, ILUT subdomain solver, and ML 2-level DD with Gauss-Seidel subdomain solver. Coarse Solver: “Exact” = SuperLU (1 proc), “Approx” = one step of ILU (8 proc. in parallel)

“Unreasonable effectiveness” of Schwarz

- **When does the sum of partial inverses equal the inverse of the sums? When the decomposition is right! Let $\{r_i\}$ be a complete set of orthonormal row eigenvectors for A :** $r_i A = a_i r_i$ **or** $a_i = r_i A r_i^T$

  Then

  $$A = \Sigma_i r_i^T a_i r_i$$

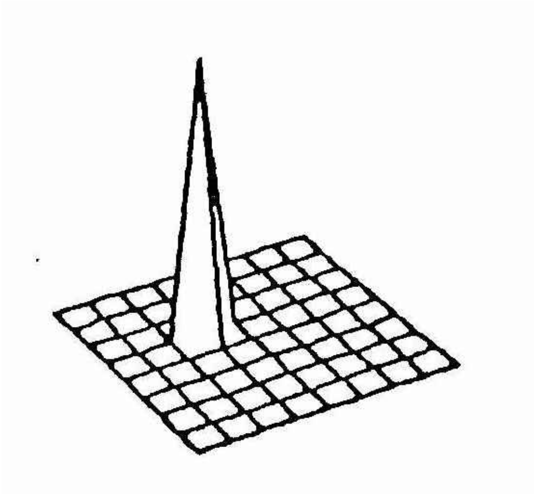
  and

  $$A^{-1} = \Sigma_i r_i^T a_i^{-1} r_i = \Sigma_i r_i^T (r_i A r_i^T)^{-1} r_i$$

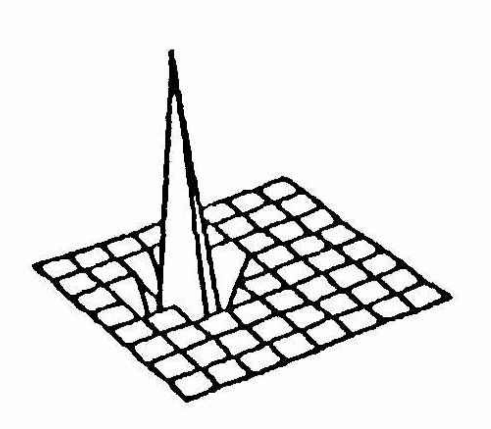
  — the Schwarz formula!

- **Good decompositions are a compromise between conditioning and parallel complexity, in practice**

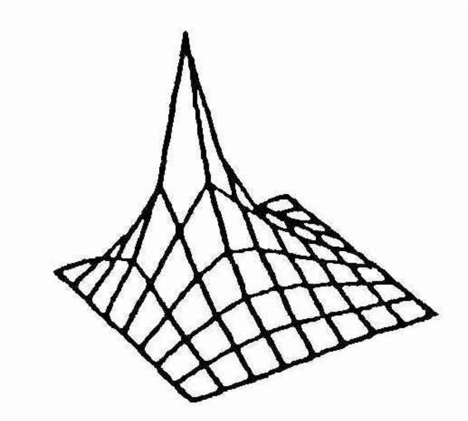
“Unreasonable effectiveness” of Schwarz, cont.



Delta function, $\delta(x)$



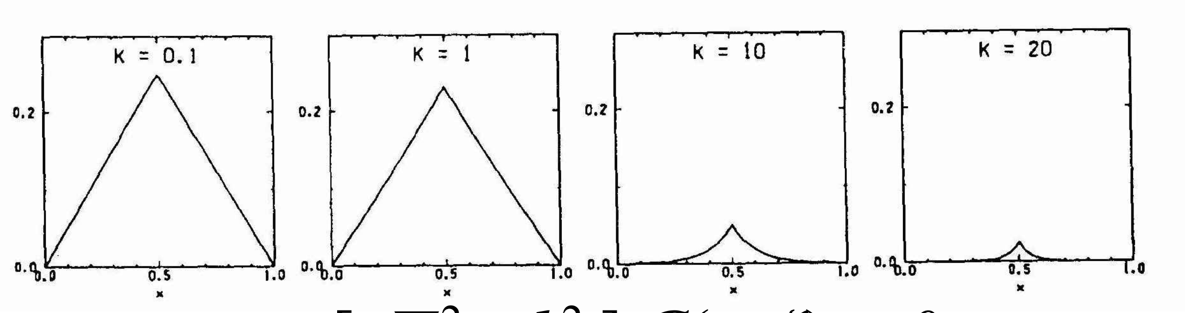
$A\ \delta(x)$



$A^{-1}\ \delta(x)$

- **Forward Poisson operator is localized and sparse**
- **Inverse operator is *locally concentrated,* but dense**
- **A coarse grid is necessary (and sufficient, for good conditioning) to represent the coupling between a field point and its forcing coming from nonlocal regions**

“Unreasonable effectiveness” of Schwarz, cont.



$$[\ -\nabla^2 + \boldsymbol{k}^2\ ]\ \boldsymbol{G}(\boldsymbol{x}, \xi) = 0$$

- **Green’s functions for the “good Helmholtz” operator on the unit interval, shown with four increasing diagonal shifts, for ξ = 0.5**
- **It is intuitively clear why the diagonally dominant case is easy to precondition without a coarse grid**
- **This corresponds to the implicitly differenced parabolic system, and arises commonly in practice**

There is no "scalable" without "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 (more practically and almost as good) linearly times a polylog term**
- **For iterative methods, this means that the product of 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, for instance, permits this with its log-diameter hardware global reduction**
- **Number of iterations comes from condition number for linear iterative methods; Newton's superlinear convergence is important for nonlinear iterations**

Why optimal algorithms?

- The more powerful the computer, the *greater* the importance of optimality
  - though the counter argument is often employed ☹
- Example:
  - Suppose *Alg1* solves a problem in time $C\,N^2$, where N is the input size
  - Suppose *Alg2* solves the same problem in time $C\,N \log_2 N$
  - Suppose *Alg1* and *Alg2* parallelize *perfectly* on a machine of 1,000,000 processors
- In constant time (compared to serial), *Alg1* can run a problem 1,000 X larger, whereas *Alg2* can run a problem nearly 65,000 X larger

Components of scalable solvers for PDEs

- Subspace solvers
  - **elementary smoothers**
  - **incomplete factorizations**
  - **full direct factorizations**

alone unscalable: either too many iterations or too much fill-in

- Global linear preconditioners
  - **Schwarz and Schur methods**
  - **multigrid**

opt. combins. of subspace solvers

- Linear accelerators
  - **Krylov methods**

mat-vec algs.

- Nonlinear rootfinders
  - **Newton-like methods**

vec-vec algs.
+ linear solves

Newton-Krylov-Schwarz: a PDE applications "workhorse"

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

$$u = u_c + \lambda\,\delta u$$

Newton

nonlinear solver

asymptotically quadratic

$$J\delta u = -F$$

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

Krylov

accelerator

spectrally adaptive

$$M^{-1}J\delta u = -M^{-1}F$$

$$M^{-1} = \sum_i R_i^T (R_i J R_i^T)^{-1} R_i$$

Schwarz

preconditioner

parallelizable

"Secret sauce" #1: iterative correction w/ each step *O*(*N*)

- The most basic idea in iterative methods for $Ax = b$

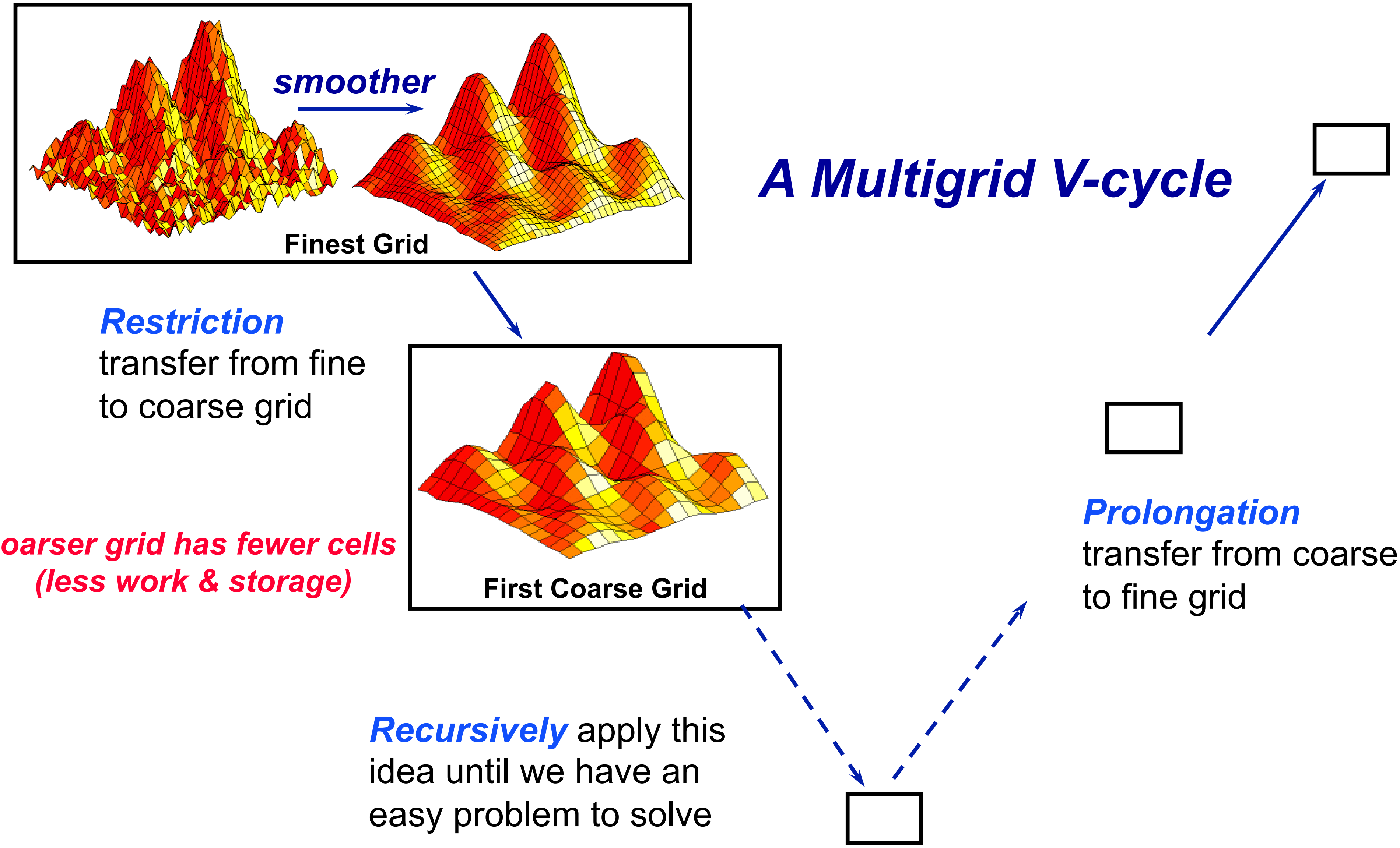
$$x \leftarrow x + B^{-1}(b - Ax)$$

- Evaluate residual accurately, but solve approximately, where B^{-1} is an approximate inverse to A
- A sequence of complementary solves can be used, e.g., with B_1 first and then B_2 one has

$$x \leftarrow x + [B_1^{-1} + B_2^{-1} - B_2^{-1} A B_1^{-1}](b - Ax)$$

- Scale recurrence, e.g., with $B_2^{-1} = R^T (RAR^T)^{-1} R$, leads to *multilevel methods*
- Optimal polynomials of $(B^{-1}A)$ lead to various *preconditioned Krylov methods*

“Secret sauce” #2:
treat each error component in optimal subspace



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

c/o R. Falgout, LLNL

"Secret sauce" #3: skip the Jacobian

- In the Jacobian-Free Newton-Krylov (JFNK) method for $F(u) = 0$, 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)

Carl Jacobi

Secret sauce #4: use the user's solver to precondition

- Almost any code to solve $F(u) = 0$ computes a residual and invokes some process to compute an update to u based on the residual
- Defines a weakly converging nonlinearly method

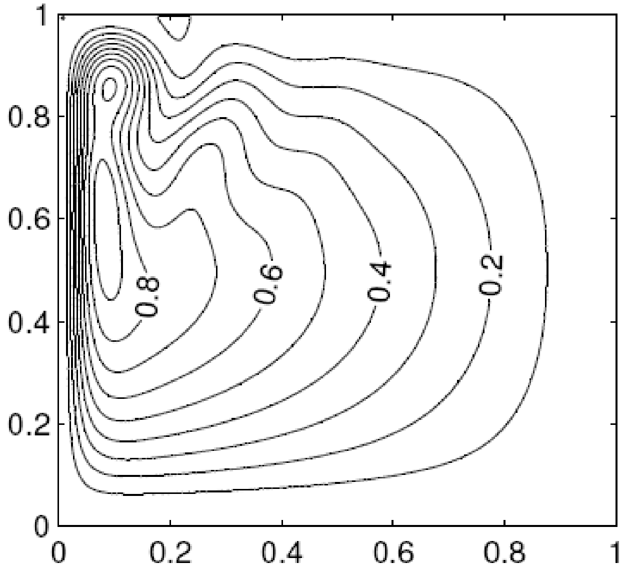
$$M : F(u^k) \mapsto \delta u$$

$$u^{k+1} \leftarrow u^k + \delta u$$

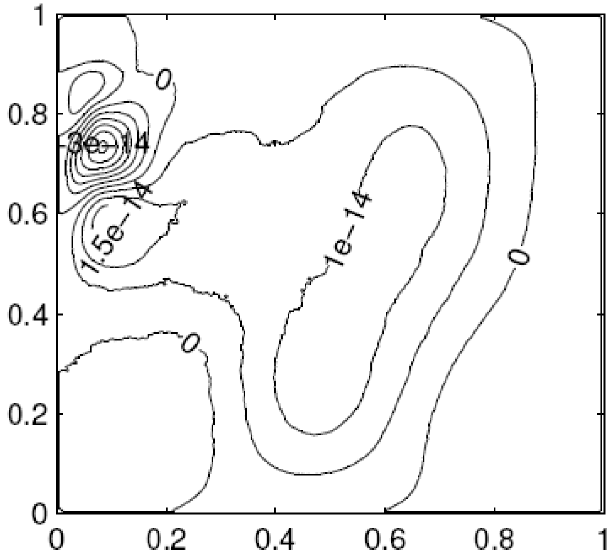
- M is, in effect, a preconditioner and can be applied directly within a Jacobian-free Newton context
- This is the "physics-based preconditioning" strategy discussed in the E^3 report

Example: fast spin-up of ocean circulation model using Jacobian-free Newton-Krylov

- State vector, $u(t)$
- Propagation operator (this is any code) $\Phi(u,t)$: $u(t) = \Phi(u(0),t)$
  - here, single-layer quasi-geostrophic ocean forced by surface Ekman pumping, damped with biharmonic hyperviscosity
- *Task*: find state u that repeats every period T (assumed known)
- *Difficulty*: direct integration (DI) to find steady state may require thousands of years of physical time
- *Innovation*: pose as Jacobian-free NK rootfinding problem, $F(u) = 0$, where $F(u) \equiv u - \Phi(u(0),T)$
  - Jacobian is dense, would never think of forming!

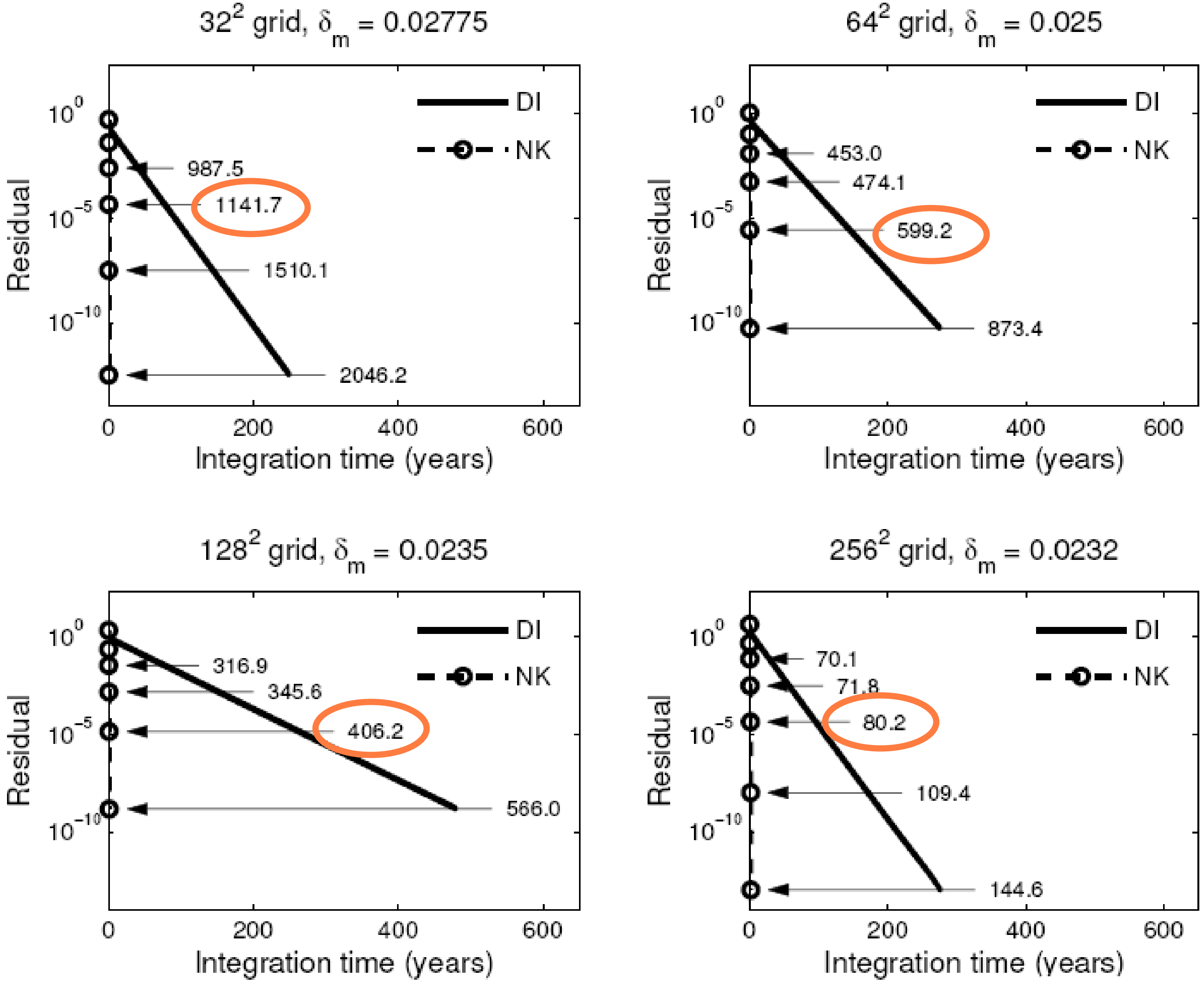


converged streamfunction



difference between DI and NK (10^{-14})

Example: fast spin-up of ocean circulation model using Jacobian-free Newton-Krylov



2-3 orders of magnitude speedup of Jacobian-free NK relative to Direct Integration (DI)

OGCM: Helfrich-Holland integrator

Implemented in PETSc as undergraduate research project

c/o T. Merlis (Columbia'05, *now* Caltech, Dept. Environmental Science & Engineering)

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 by automatic differentiation, so that the actual Jacobian elements are *never explicitly needed* (except for preconditioning)

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

How to accommodate preconditioning

- Krylov iteration is expensive in memory and in function evaluations, so subspace dimension ***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 blocks decomposed for parallelism (Schwarz)**
  - **Jacobian of lower-order discretization**
  - **Jacobian with "lagged" values for expensive terms**
  - **Jacobian stored in lower precision**
  - **Jacobian of related discretization**
  - **operator-split Jacobians**
  - **physics-based preconditioning**

Nonlinear Schwarz methods

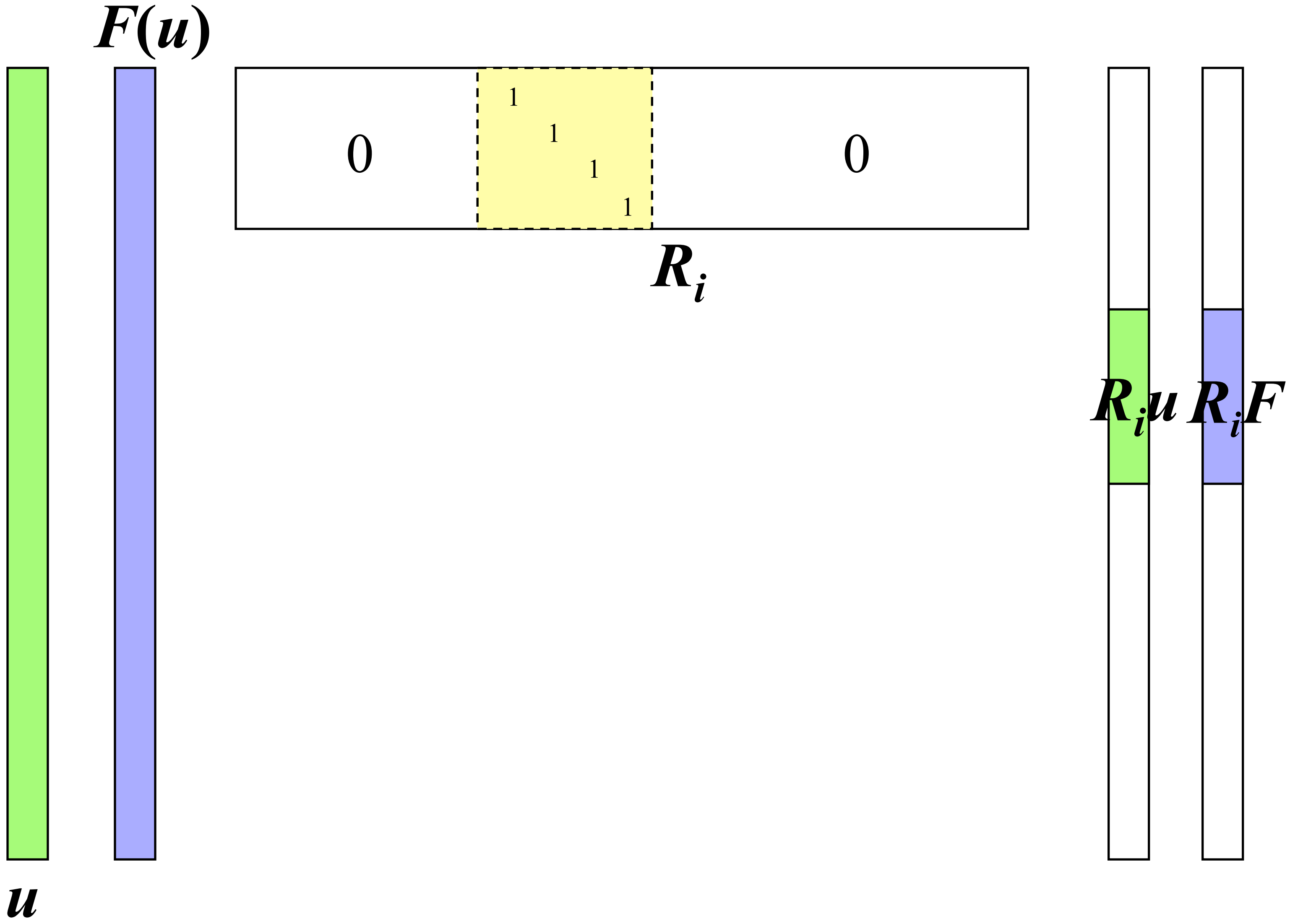
- Nonlinear Schwarz replaces linear solves with Newton in *inner* and *outer* iterations
- It replaces $F(u)=0$ with a new nonlinear system possessing the same root, $\Phi(u)=0$
- Define a correction $\delta_i(u)$ to the i^{th} partition (e.g., subdomain) of the solution vector by solving the following local nonlinear system:

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

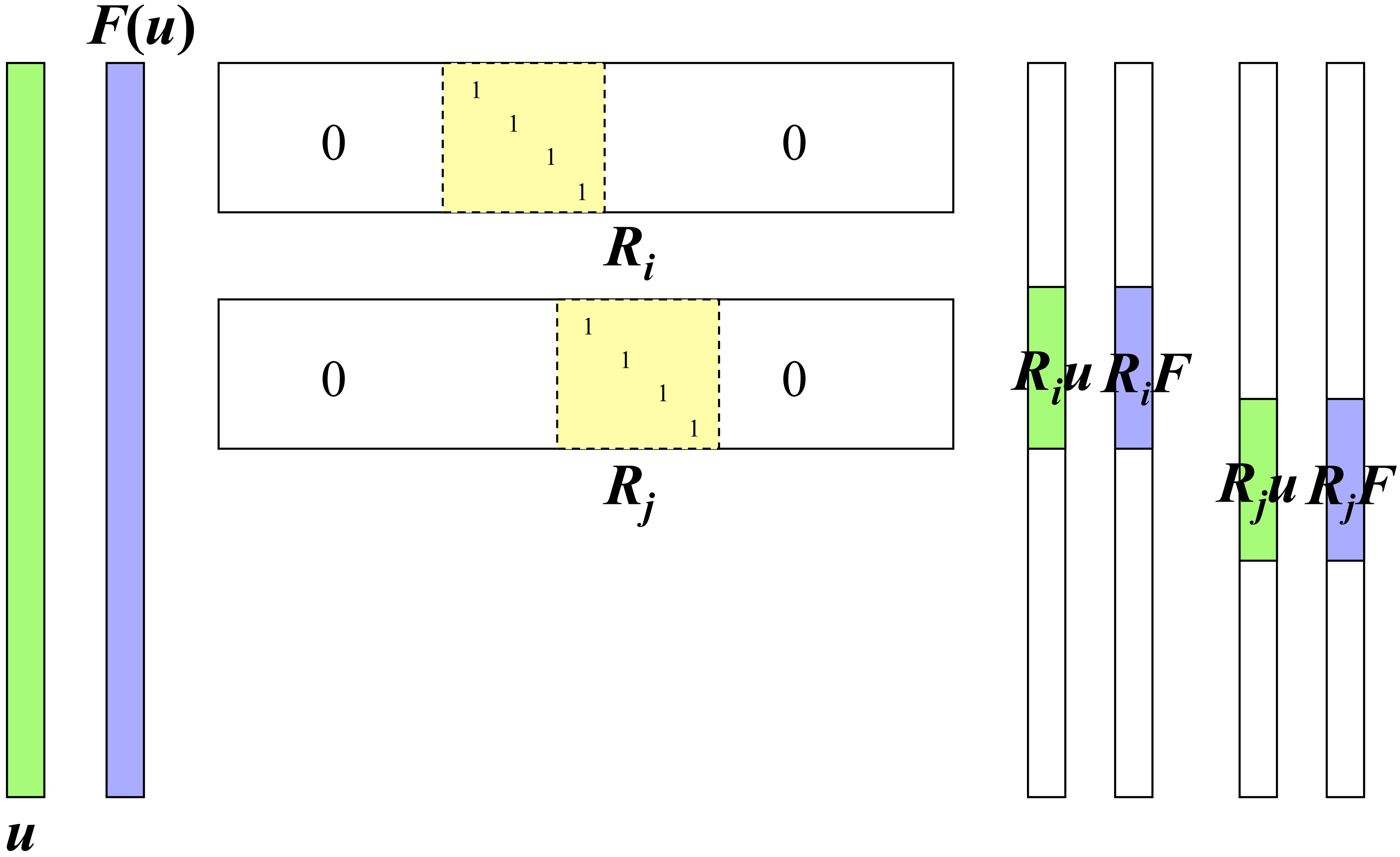
where $\delta_i(u) \in \Re^n$ is nonzero only in the components of the i^{th} partition

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

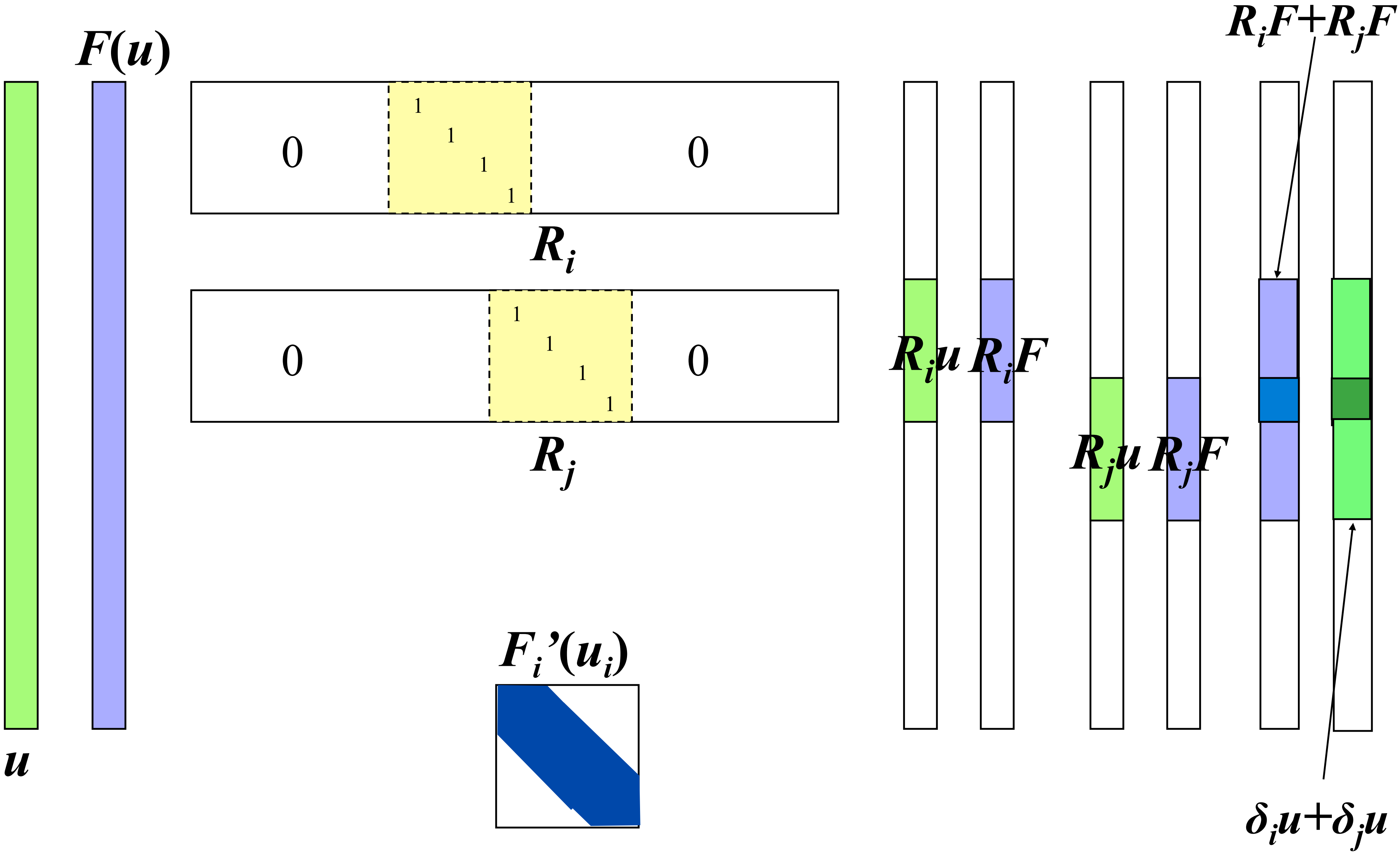
Nonlinear Schwarz – picture



Nonlinear Schwarz – picture



Nonlinear Schwarz – picture



Nonlinear Schwarz, cont.

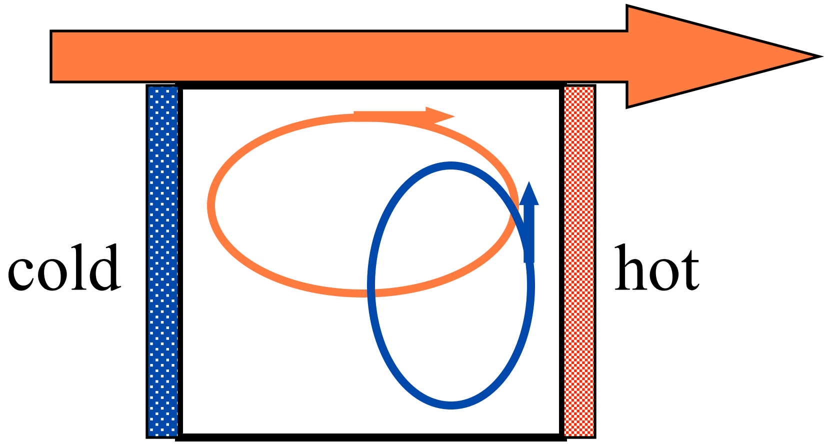
- It is simple to prove that if the Jacobian of $F(u)$ is nonsingular in a neighborhood of the desired root then $\Phi(u)=0$ and $F(u)=0$ have the same unique root
- To lead to a Jacobian-free Newton-Krylov algorithm we need to be able to evaluate for any $u,v\in\Re^n$:
  - The residual $\Phi(u)=\sum_i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

Driven cavity in velocity-vorticity coords



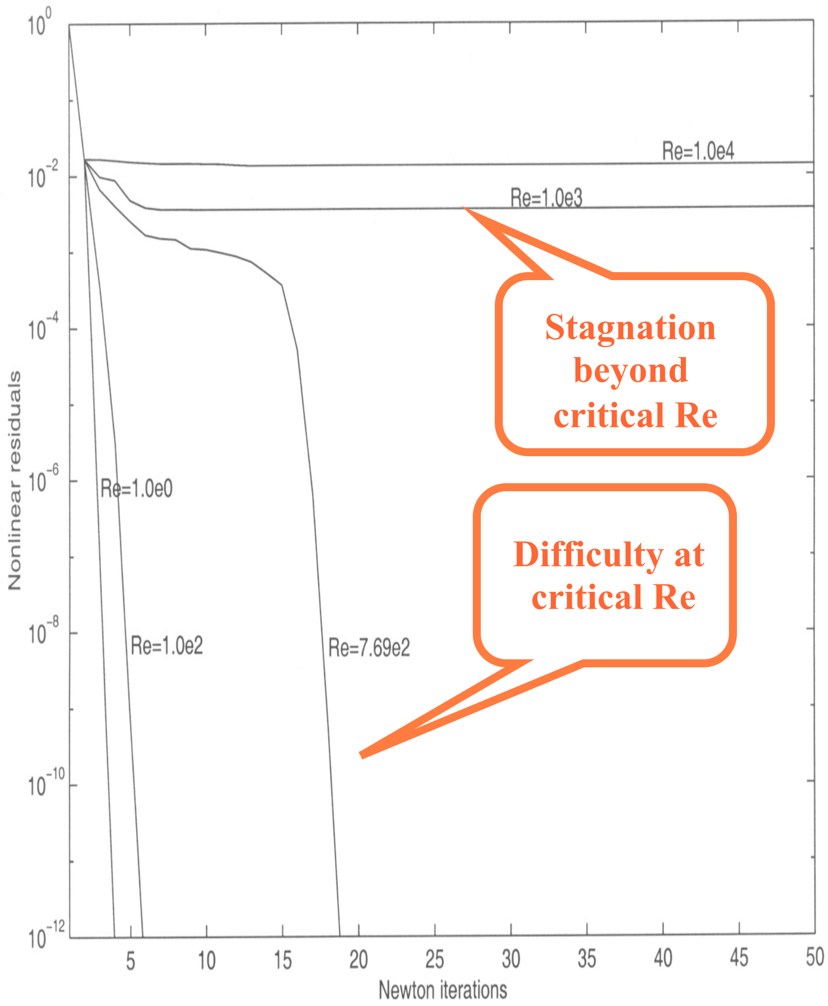
x-velocity $$-\nabla^2 u - \frac{\partial\omega}{\partial y} = 0$$

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

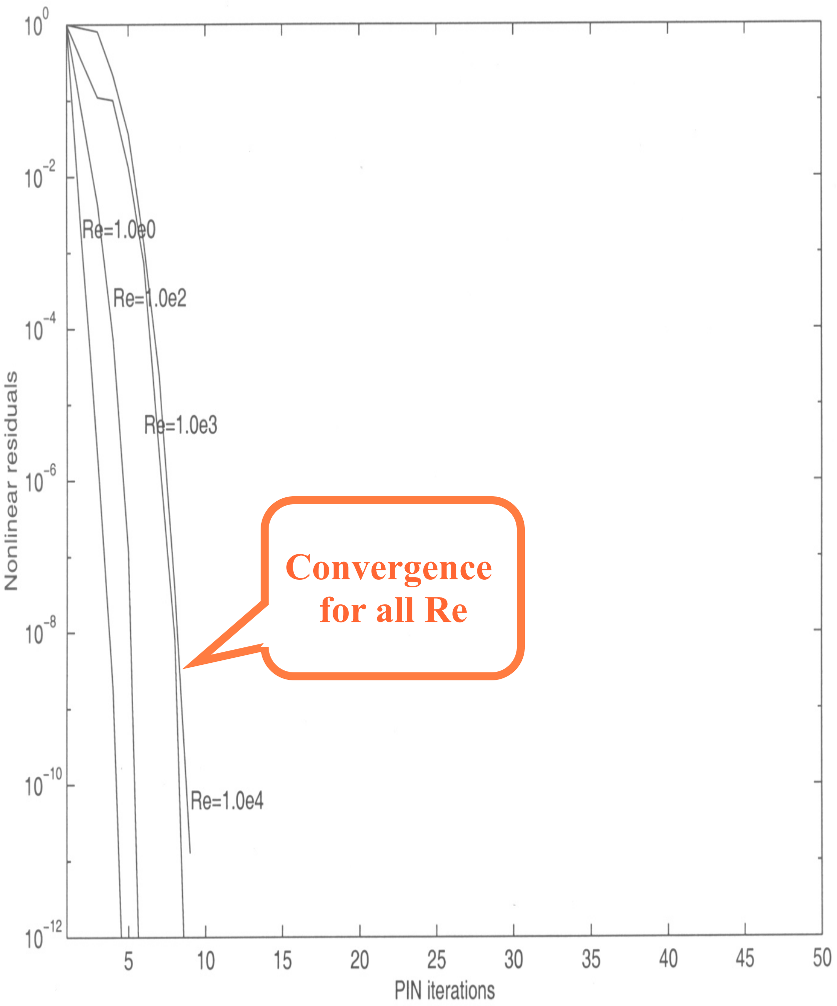
Experimental example of nonlinear Schwarz



Stagnation beyond critical Re

Difficulty at critical Re

Vanilla Newton’s method



Convergence for all Re

Nonlinear Schwarz

Multiphysics coupling

- Domain decomposition is fundamentally algebraic
- It does not care (except in the preconditioner) whether the coupled subproblems are from different subdomains, or different equations sets defined over a common domain
- Hence domain decomposition methods suggest methods for attacking multiphysics problems
- Consider (in the next three slides) the two standard means of solving multiphysics problems, by nested elimination and by block nonlinear Gauss-Seidel, and then nonlinear Schwarz

Multiphysics coupling: partial elimination

- Consider system $F(u) = 0$ partitioned by physics as

$$\begin{cases} F_1(u_1, u_2) = 0 \\ F_2(u_1, u_2) = 0 \end{cases}$$

- Can formally solve for u_1 in $F_1(u_1, u_2) = 0$

$$u_1 \equiv G(u_2)$$

- Then second equation is $F_2(G(u_2), u_2) = 0$
- Jacobian

$$\frac{dF_2}{du_2} = \frac{\partial F_2}{\partial u_1}\frac{\partial G}{\partial u_2} + \frac{\partial F_2}{\partial u_2}$$

can be applied to a vector in matrix-free manner

Multiphysics coupling: nonlinear GS

- **In previous notation, given initial iterate** $\left\{ u_1^0, u_2^0 \right\}$
- **For *k=1, 2, …,* until convergence, do**
  - Solve for *v* in $F_1(v, u_2^{k-1}) = 0$
  - Solve for *w* in $F_2(v, w) = 0$
- **Then**

$$\left\{ u_1^k, u_2^k \right\} = \left\{ v, w \right\}$$

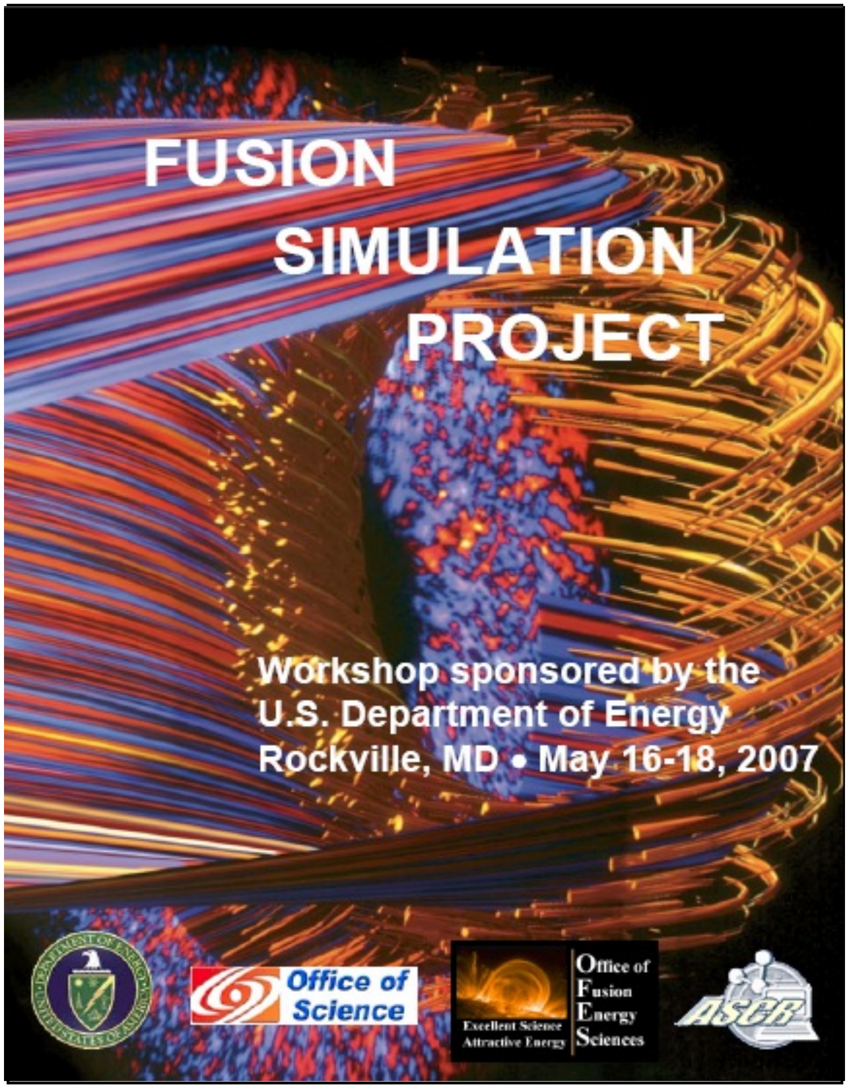
Multiphysics coupling: nonlinear Schwarz

- **Given initial iterate** $\left\{u_1^0, u_2^0\right\}$
- **For** *k=1, 2, …,* **until convergence, do**
  - Define $G_1(u_1,u_2) \equiv \delta u_1$ by $F_1(u_1^{k-1} + \delta u_1, u_2^{k-1}) = 0$
  - Define $G_2(u_1,u_2) \equiv \delta u_2$ by $F_2(u_1^{k-1}, u_2^{k-1} + \delta u_2) = 0$
- **Then solve** $\begin{cases} G_1(u,v) = 0 \\ G_2(u,v) = 0 \end{cases}$ **in matrix-free manner**
- **Jacobian:**

$$\begin{bmatrix} \frac{\partial G_1}{\partial u} & \frac{\partial G_1}{\partial v} \\ \frac{\partial G_2}{\partial u} & \frac{\partial G_2}{\partial v} \end{bmatrix} \approx \begin{bmatrix} I & \left(\frac{\partial F_1}{\partial u}\right)^{-1} \frac{\partial F_1}{\partial v} \\ \left(\frac{\partial F_2}{\partial v}\right)^{-1} \frac{\partial F_2}{\partial u} & I \end{bmatrix}$$

- **Finally** $\left\{u_1^k, u_2^k\right\} = \left\{v, w\right\}$

SciDAC’s Fusion Simulation Project: support of the international fusion program

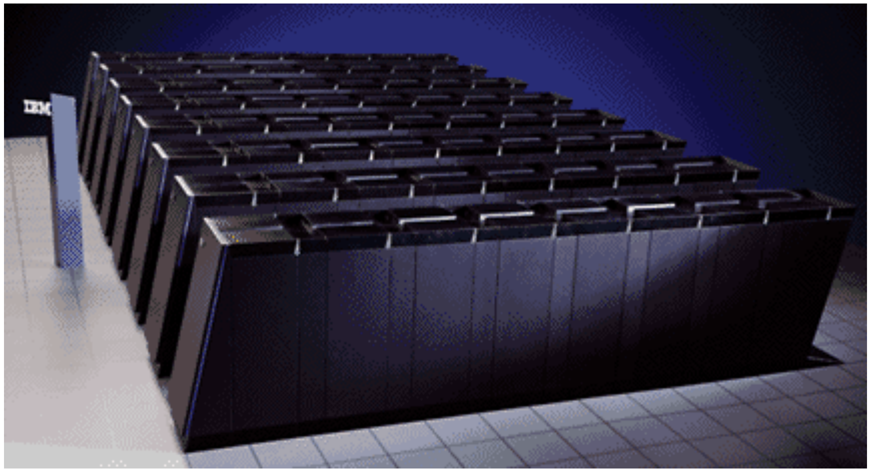


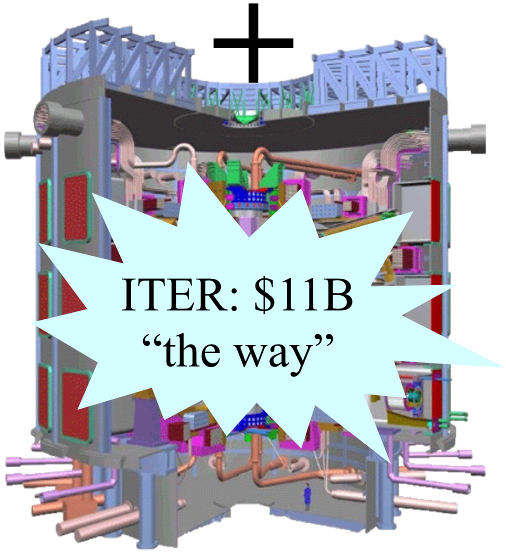
FUSION SIMULATION PROJECT

Workshop sponsored by the U.S. Department of Energy
Rockville, MD • May 16-18, 2007

Office of Science
Excellent Science Attractive Energy
Office of Fusion Energy Sciences
ASCR

J. Fusion Energy 28: 1-59 (2007)





ITER: $11B
“the way”

Fusion by 2017; criticality by 2022

“Big Iron” meets “Big Copper”

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."

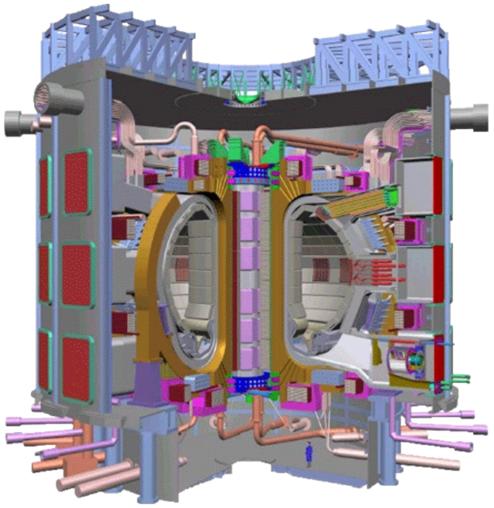
Optimizing the algorithms used, in other words, could lower the computing power required for some ITER simulations by an astounding nine orders of magnitude. Even more exciting, those newly feasible simulations would be at the petascale—ready to run on the petaflop/s supercomputers widely expected within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

			Small tokamak	Large tokamak	Huge tokamak
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 processo speed and efficiency
- 1.5 orders: nc ncy

Software: 9

- 1 ord i
  - Same fewer elements
- 1 ng
  - Les es
- 4 ord
  - Zones requ ing of ITER volume and resolution require ents a fro 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})!

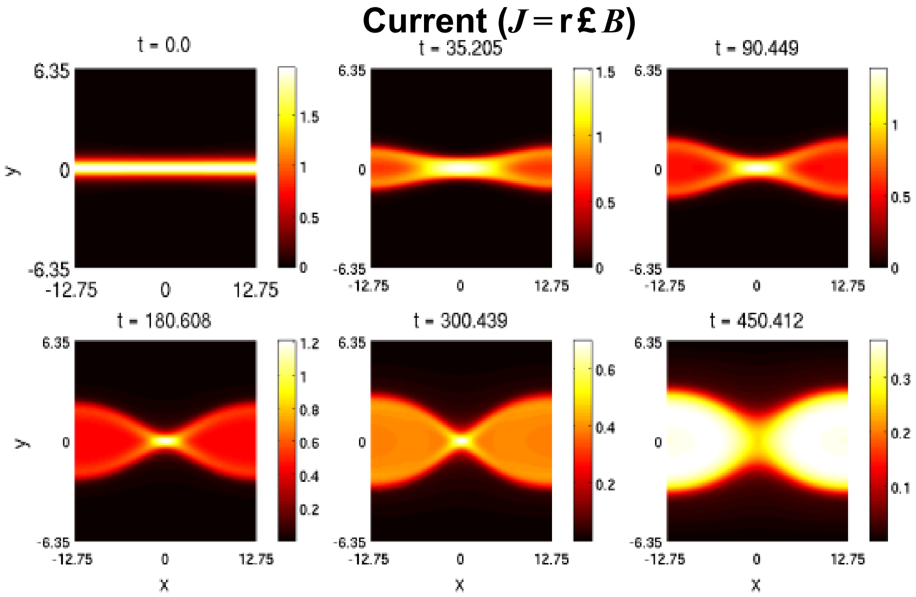
Comments on ITER simulation roadmap

- Increased processor speed
  - 10 years is 6.5 Moore doubling times
- Increased concurrency
  - BG/L is already 2^{17} procs, MHD now routinely at ca. 2^{12}
- Higher-order discretizations
  - low-order preconditioning of high-order discretizations
- Flux-surface following gridding
  - in SciDAC, this is ITAPS; evolve mesh to approximately follow flux surfaces
- Adaptive gridding
  - in SciDAC, this is APDEC; Cartesian AMR
- Implicit solvers
  - in SciDAC, this is TOPS; Newton-Krylov w/multigrid preconditioning

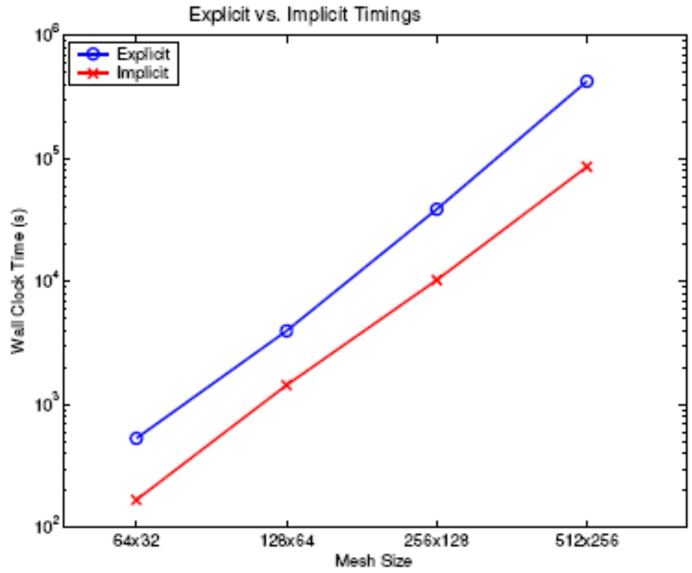
Resistive MHD prototype implicit solver

- ***Magnetic reconnection*: the breaking and reconnecting of oppositely directed magnetic field lines in a plasma, replacing hot plasma core with cool plasma, halting the fusion process**

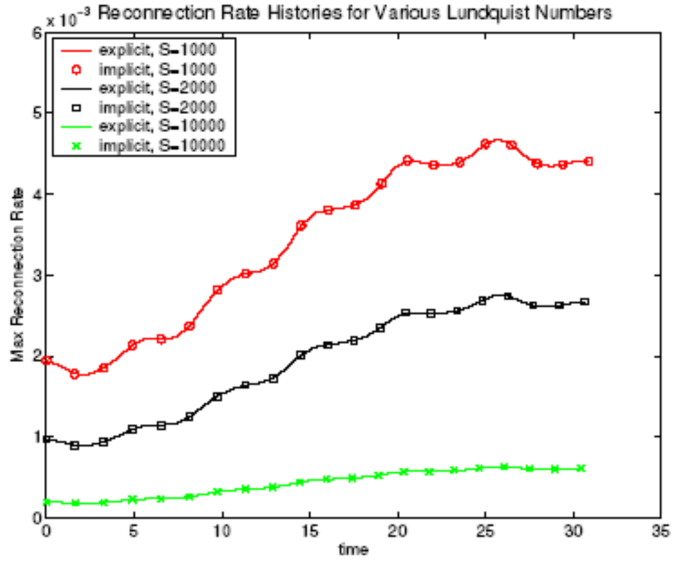
- **Replace explicit updates with implicit Newton-Krylov from SUNDIALS with factor of ~5× in execution time**



Current ($J = r£B$)



Explicit vs. Implicit Timings



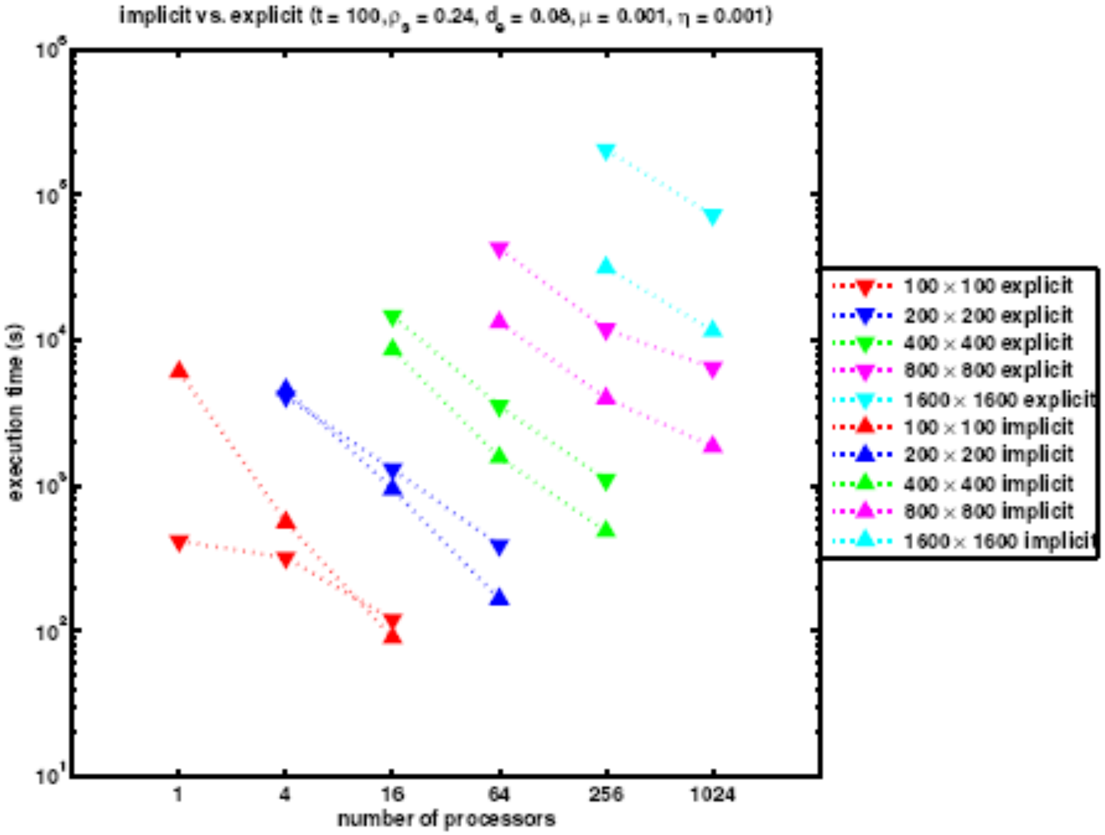
Reconnection Rate Histories for Various Lundquist Numbers

J. Brin et al., "Geospace Environmental Modeling (GEM) magnetic reconnection challenge," **J. Geophys. Res.** 106 (2001) 3715-3719.

c/o D. Reynolds, UCSD

Resistive MHD: implicit solver, ex #2

explicit/implicit execution time comparison



- *Magnetic reconnection*: previous example was compressible – primitive variable; this example is incompressible – streamfunction/vorticity

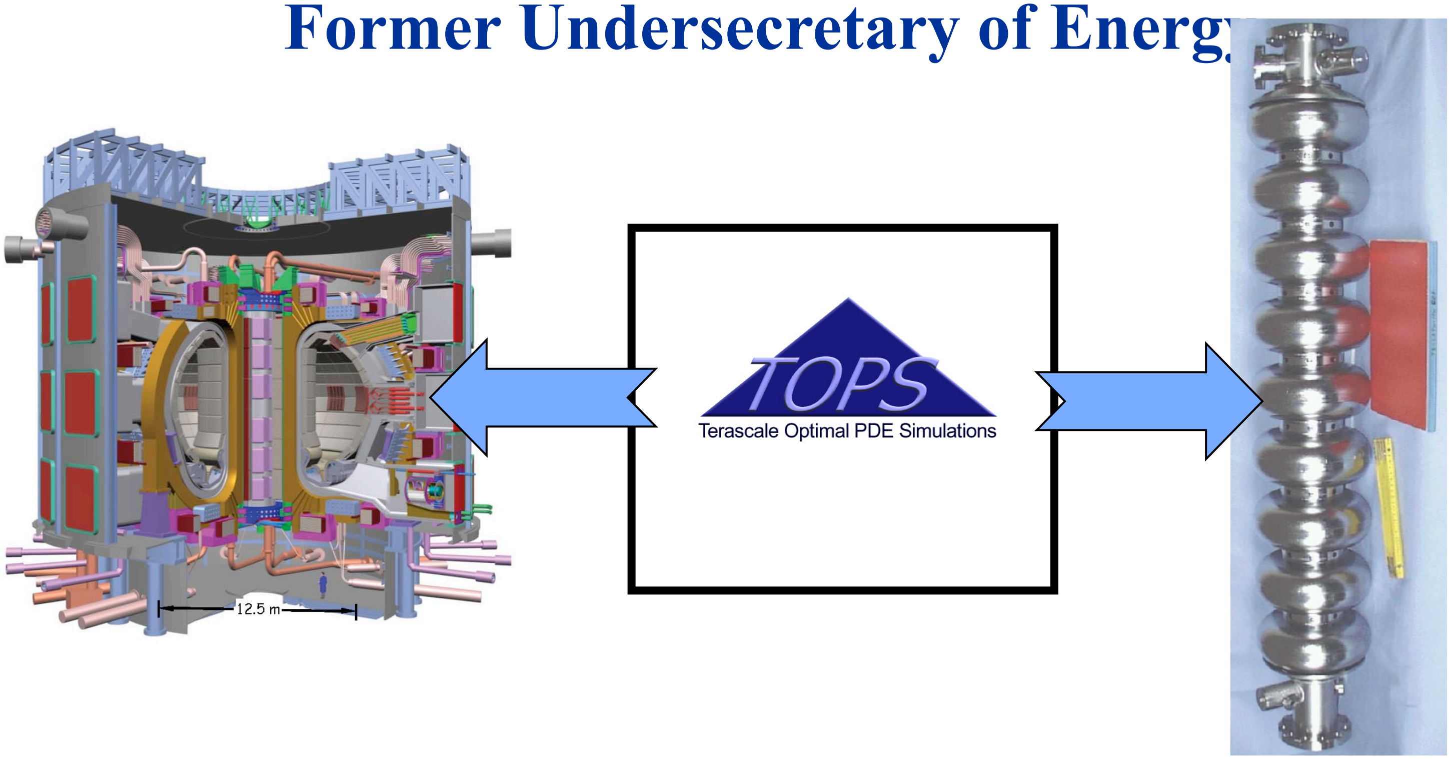
- Replace explicit updates with implicit Newton-Krylov from **PETSc** with factor of ~5× in speedup

c/o F. Dobrian, ODU

"What changed were simulations that showed that the new ITER design will, in fact, be capable of achieving and sustaining burning plasma."

– Ray Orbach,

Former Undersecretary of Energy



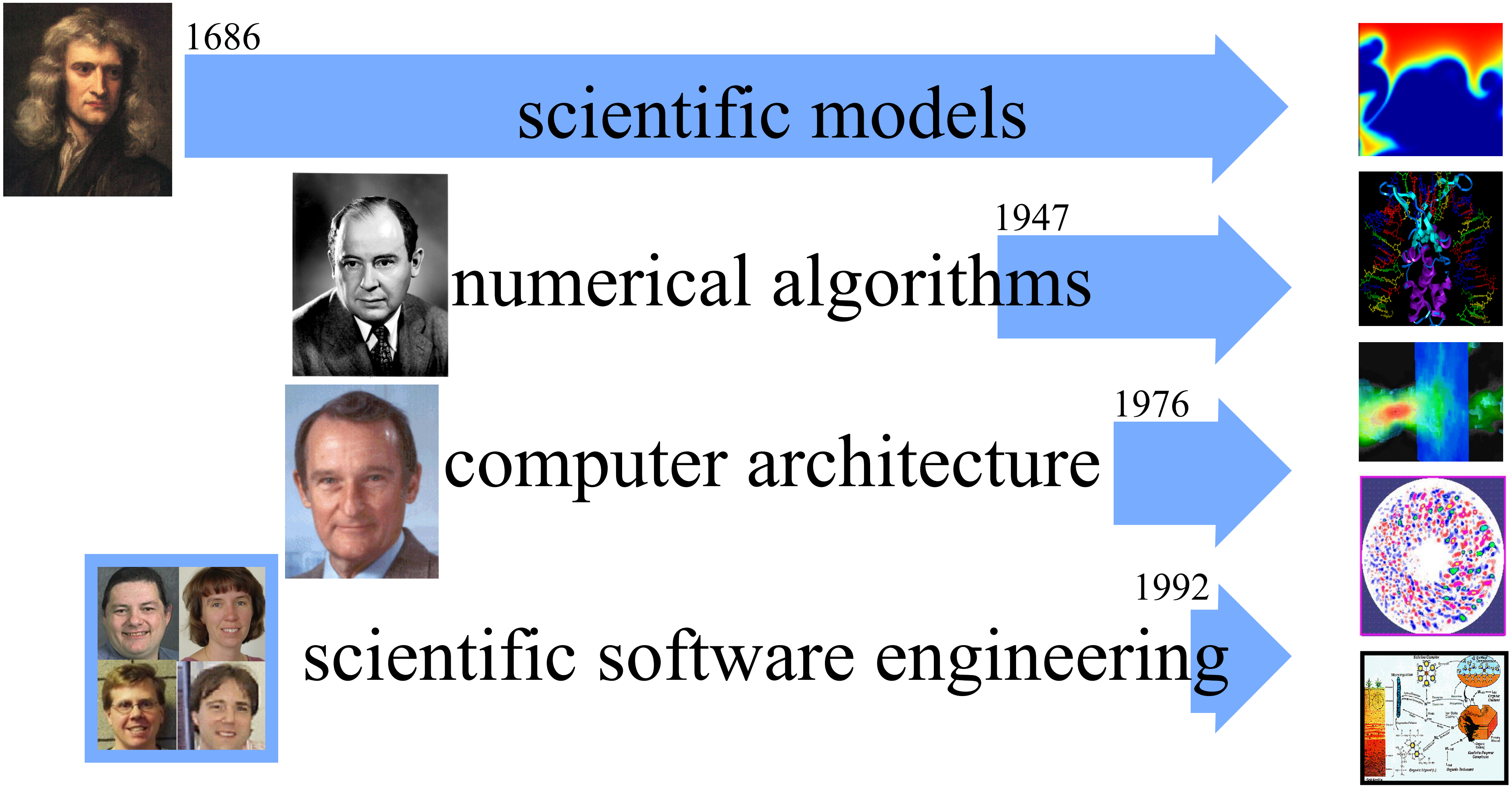
The U.S. role in multi-billion-dollar international projects will increasingly depend upon large-scale simulation, as exemplified by the 2003 Congressional testimony of Ray Orbach, above.

TOPS' wishlist for MHD collaborations — "Asymptopia"

- **Engage at a higher-level than *Ax=b***
  - **Newton-Krylov-Schwarz/MG on coupled nonlinear system**
- **Sensitivity analyses**
  - **validation studies**
- **Stability analyses**
  - **"routine" outer loop on steady-state solutions**
- **Optimization**
  - **parameter identification**
  - **design of facilities (accelerators, tokamaks, power plants, etc.)**
  - **control of experiments**

A “perfect storm” for scientific simulation

(dates are symbolic)



1686

scientific models

1947

numerical algorithms

1976

computer architecture

1992

scientific software engineering

TOPS dreams that users will…

- Understand range of algorithmic options w/ tradeoffs

  e.g., memory *vs.* time, comp. *vs.* comm., inner iteration work *vs.* outer

- Try all reasonable options "easily"

  without recoding or extensive recompilation

- Know how their solvers are performing

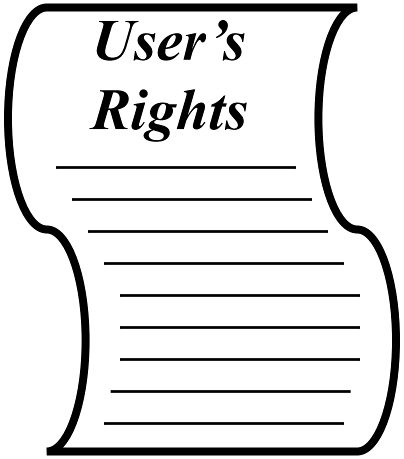
  with access to detailed profiling information

- Intelligently drive solver research

  e.g., publish joint papers with algorithm researchers

- Simulate *truly new physics* free from solver limits

  e.g., finer meshes, complex coupling, full nonlinearity



“Co-authors” of this manifesto

- 1982 William Gropp, UIUC
- 1984 Mitchell Smooke, Yale
- 1984 Tony Chan, UCLA
- 1989 Xiao-Chuan Cai, CU-Boulder
- 1990 Barry Smith, Argonne
- 1991 David Young, Boeing
- 1992 Dana Knoll, Idaho Nat Lab
- 1992 M. Driss Tidriri, Iowa State
- 1993 V. Venkatakrishnan, Boeing
- 1993 Dimitri Mavriplis, UWyoming
- 1995 C. Timothy Kelley, NCSU
- 1995 Omar Ghattas, UT-Austin
- 1995 Lois C. McInnes, Argonne
- 1996 Dinesh Kaushik, Argonne
- 1997 John Shadid, Sandia
- 1997 Kyle Anderson, UT-C
- 1997 Carol Woodward, LLNL
- 2001 Florin Dobrian, ODU
- 2002 Daniel Reynolds, UCSD
- 2006 Yuan He, Columbia

(with the year in which we began substantive collaborations)

EOF