Slides of this lecture are available at

http://www.columbia.edu/~kd2112/IceSheets09.pdf

Scalable Implicit Methods, SciDAC, and Ice Sheet Modeling

David Keyes

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

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

Caveats

- My HPC colleagues can now go to get coffee
  - Nothing particularly new in this talk
- My new geophysics/glaciologist/climatologist colleagues may at first think the talk has limited perspective
  - In fact, it is of *very broad* perspective
  - The feel of limited perspective is related to my relative newcomer status to ice sheets
  - No one is naïve enough to think that all PDEs are the same; ice sheet modeling will have unique difficulties, which we in the enabling technologies of computational science read as unique opportunities
  - We know that we have a lot to learn before we present to your colleagues at geophysical or climate meetings, but this is an internal, working meeting among new colleagues who are getting acquainted
- This talk is designed to acquaint with a particular SciDAC center, which is representative of a wealth of others
  - We are all working on software for general purposes that is customizable under a relatively stable interface to particular purposes

Another caveat

- **The number of slides in this talk exceeds my time limit**
  - **I will skip many of them, but I wanted to leave you with a document with more detail for later exploration**
  - **A review that captures the spirit of this talk is also available:**
    - **D. A. Knoll , D. E. Keyes, Jacobian-free Newton-Krylov methods: a survey of approaches and applications, Journal of Computational Physics, v.193 n.2, p.357-397, 2004**

"I have only made this letter longer because I have not had the time to make it shorter."

Blaise Pascal (1623-1662), *Lettres provinciales*.

Going implicit?

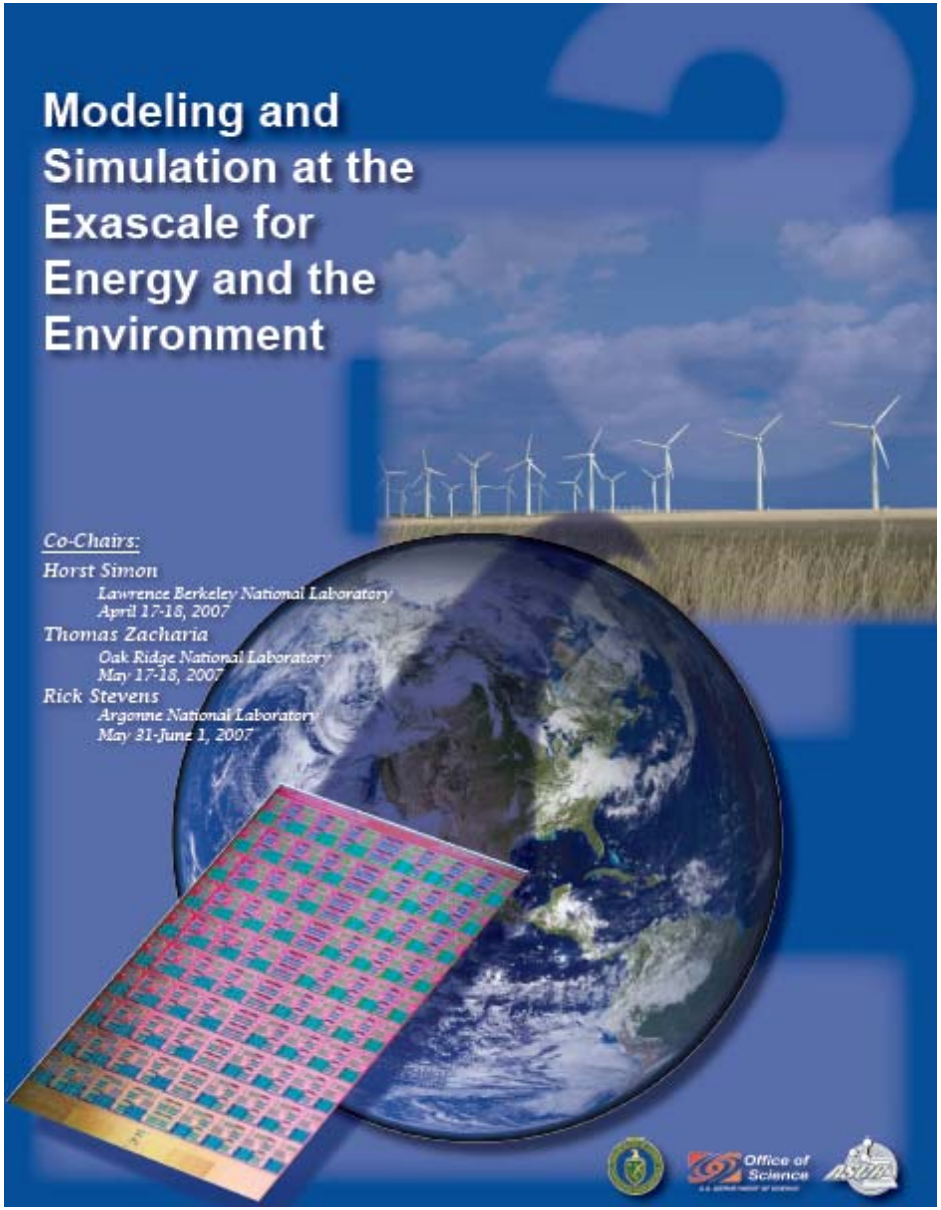
- **Why you *would*, if you *could* :**
  1. **multiscale problems with good scale separation**
  2. **coupled problems ("multiphysics")**
  3. **problems with uncertain or controllable inputs (optimization: design, control, inversion)**
- **You *can*, so you *should* !**
  1. **optimal and scalable 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

New York Blue at BNL (#45 on the Top 500)

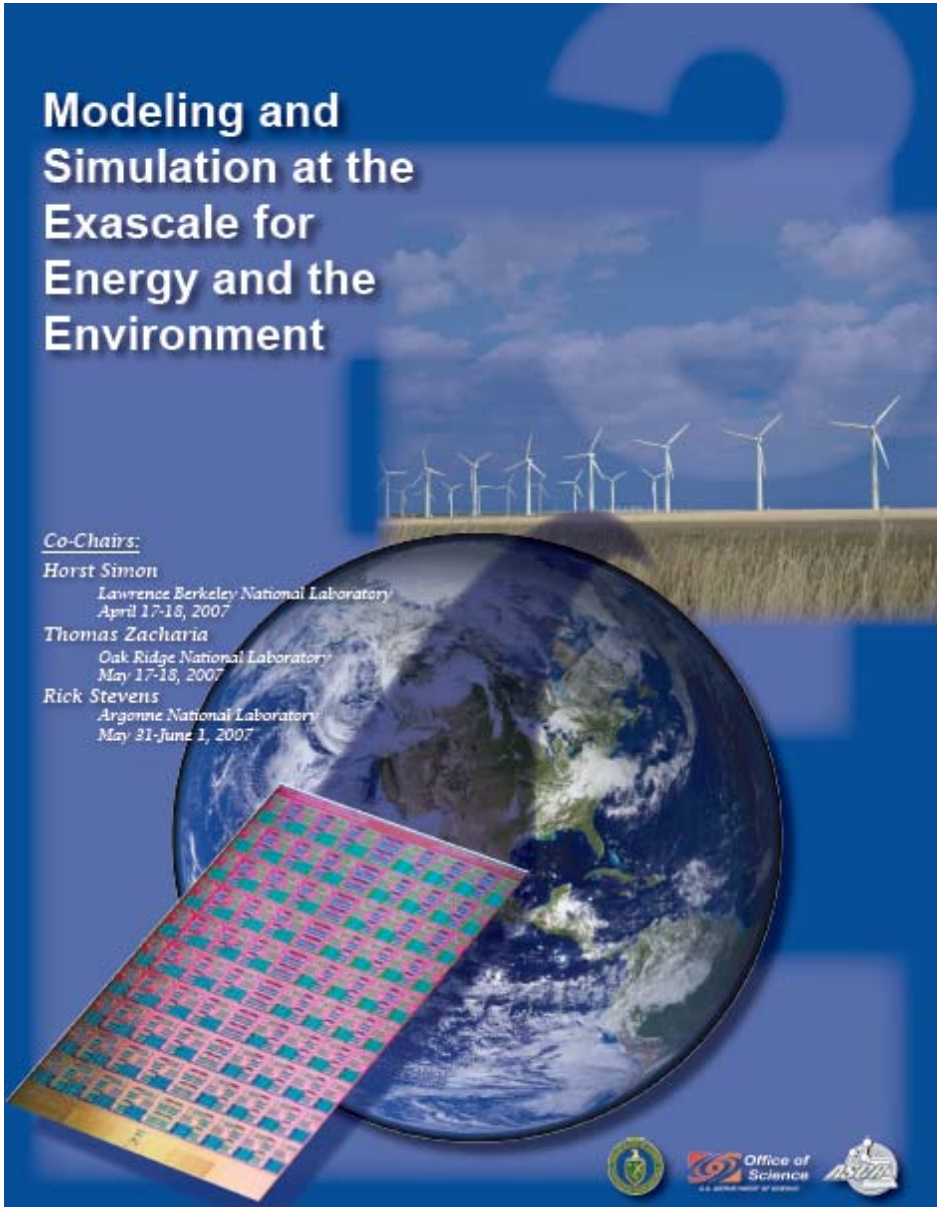
Recent "E^3" report highlights limitations of explicit methods



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."

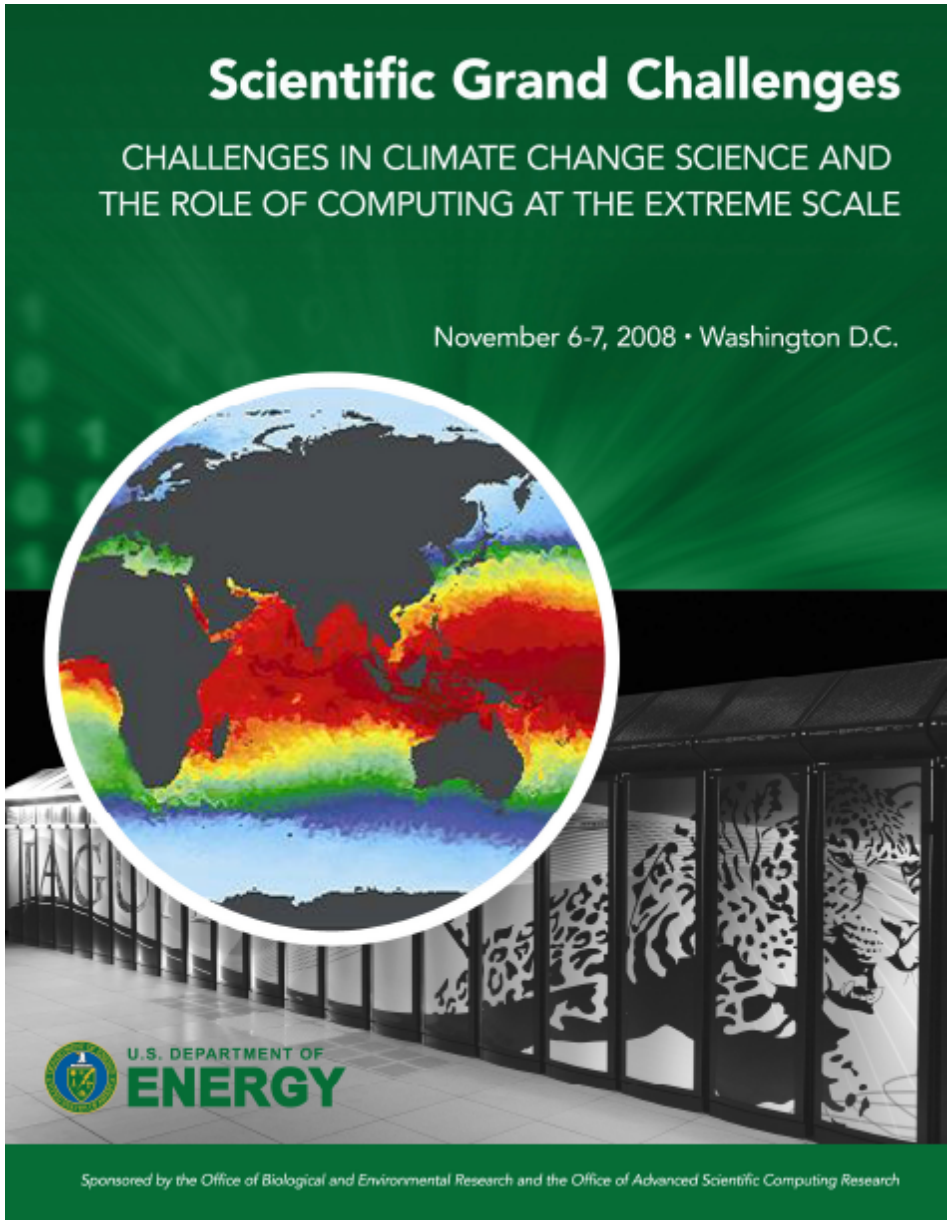
Recent E^3 report highlights opportunities for implicit methods



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."

First out of the six new "extreme scale" reports identifies implicitness as a priority



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 …*"

Scalable algorithms and ice sheet modeling

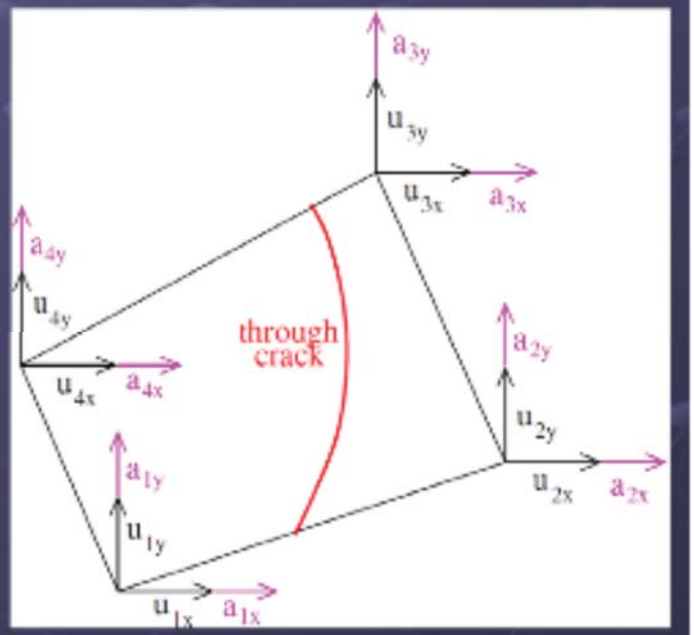
Advantages of XFEM

provides modeling flexibility and can be used to

1. Predict fracture and collapse of ice sheets
2. Predict ice calving
3. Explain accelerated ice sheet flow

Drawback of XFEM

adds degrees of freedom and accurate modeling will quickly result in *Billion of Unknowns*

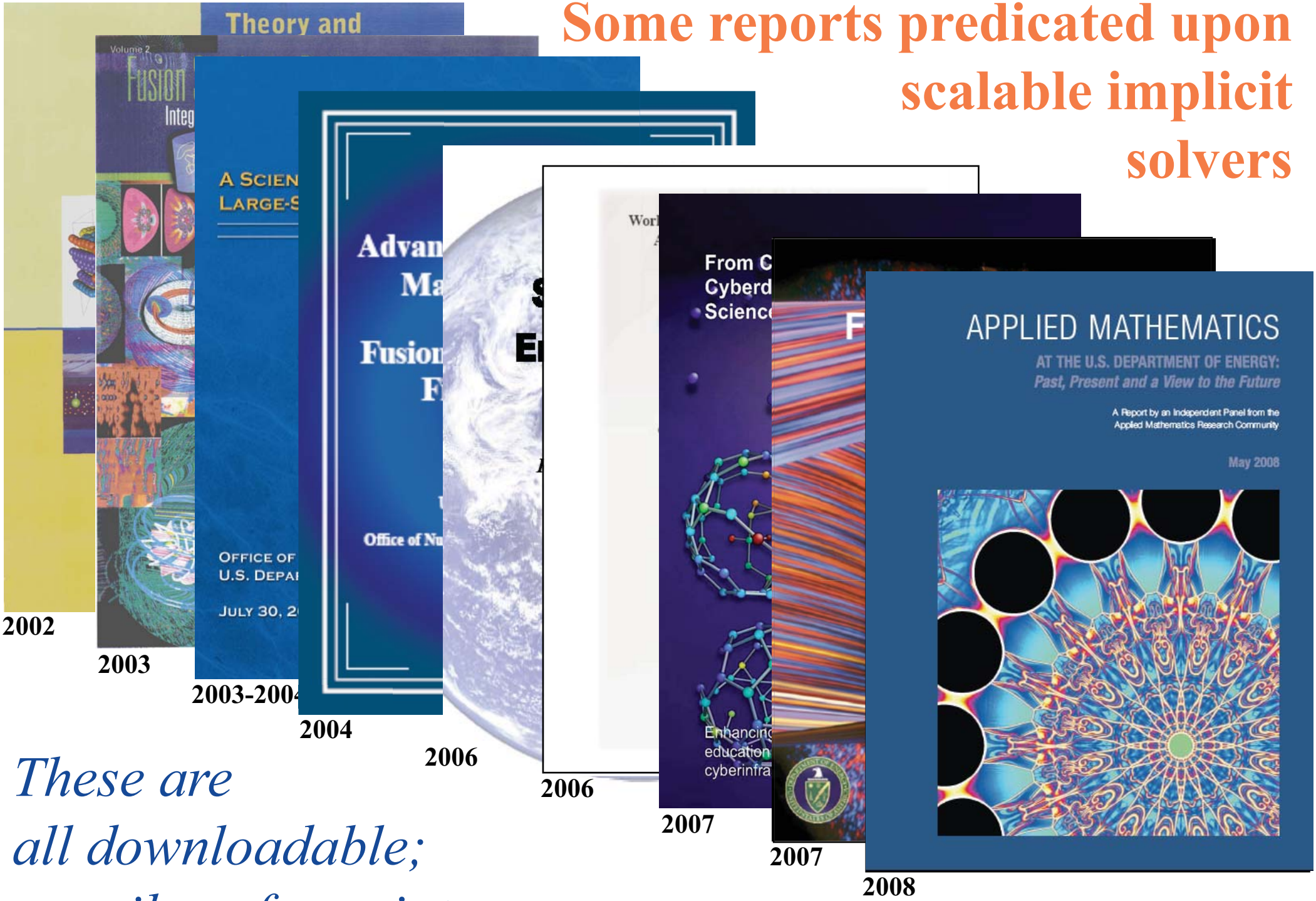


Need Efficient Parallel Solver

Strong ties with SciDAC TOPS program: Toward Optimal Petascale Simulations

c/o Haim Waisman

Some reports predicated upon scalable implicit solvers



2002

2003

2003-2004

2004

2006

2006

2007

2007

2008

These are all downloadable; e-mail me for pointers

Plan of presentation

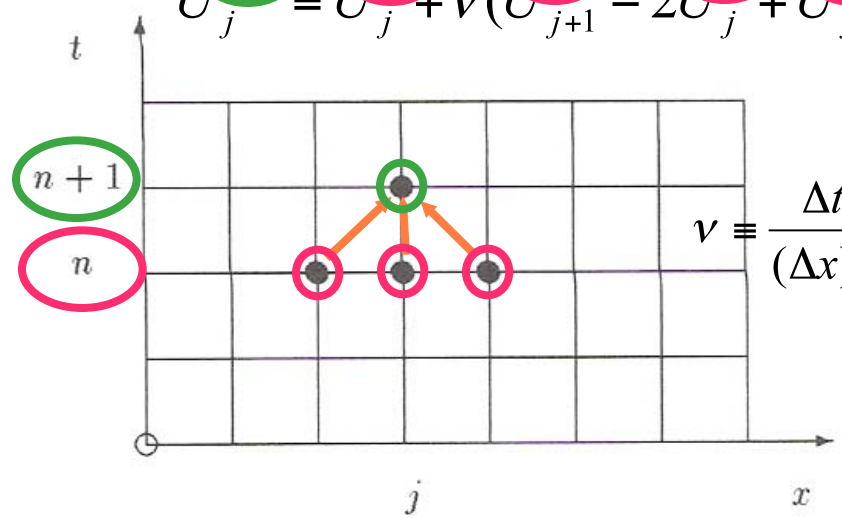
- **Motivations for implicit solvers**
  - **multi-scale, multi-physics, multi-solve (sensitivity, stability, uncertainty quantification, design, control, inversion)**
  - **one-dimensional model problems, linear and nonlinear**
- **State-of-the-art for large-scale nonlinearly implicit solvers**
  - **brief look at algorithms and software**
  - **intuition about how they scale**
- **An illustrative story from the trenches**
  - **an undergraduate semester project "gone Broadway"**

"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**

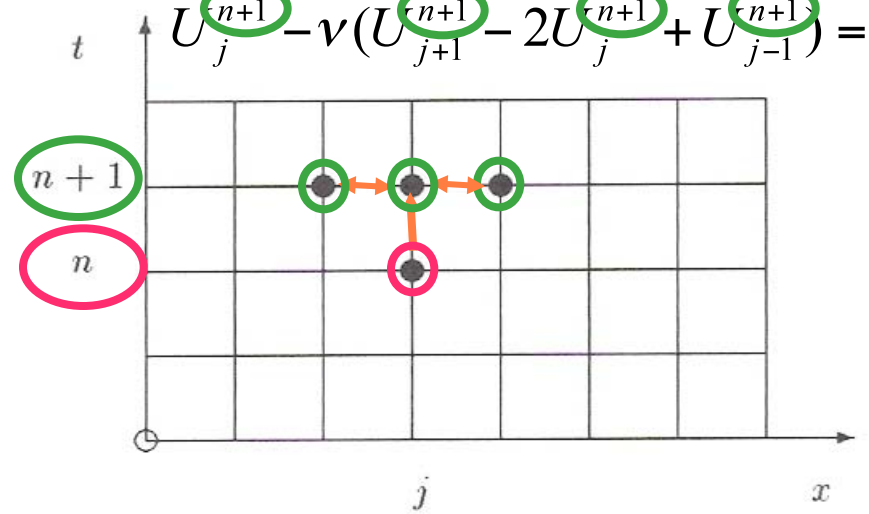
$$U_j^{n+1} = U_j^n + \nu(U_{j+1}^n - 2U_j^n + U_{j-1}^n)$$



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

- ***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

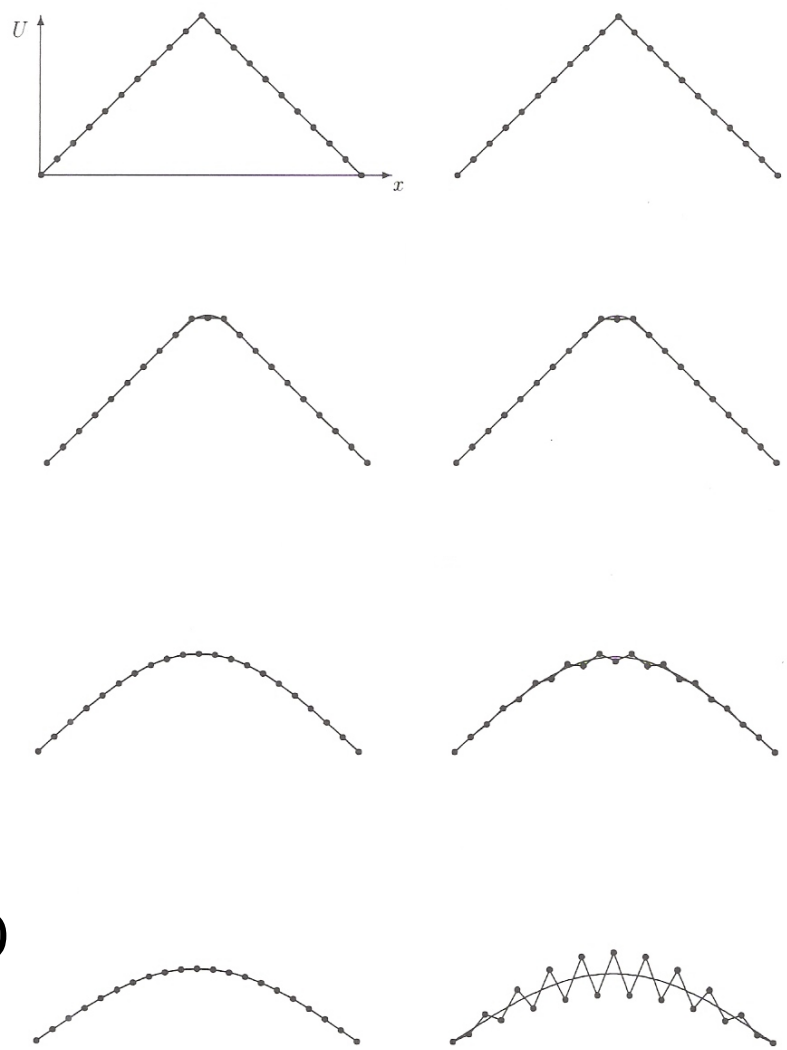
$\Delta t = 0.0012$ $\Delta t = 0.0013$

initial data

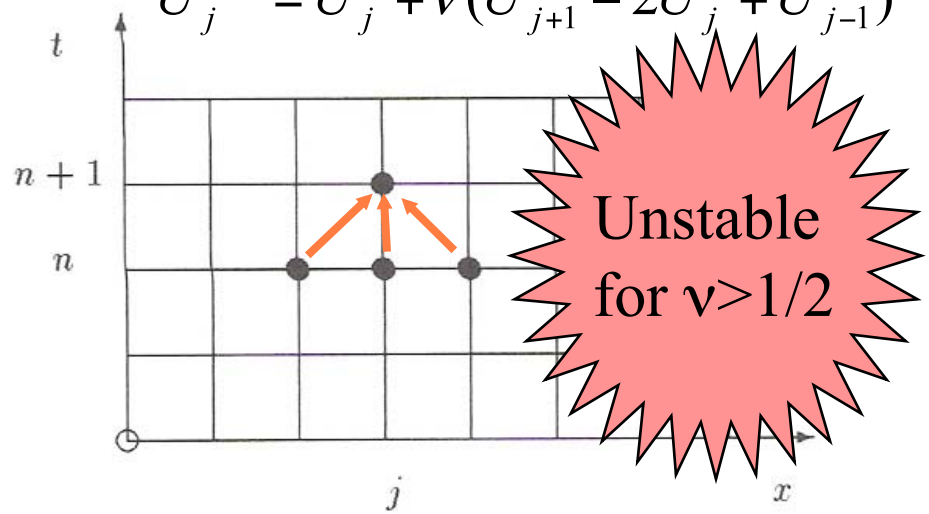
after 1 step

after 25 steps

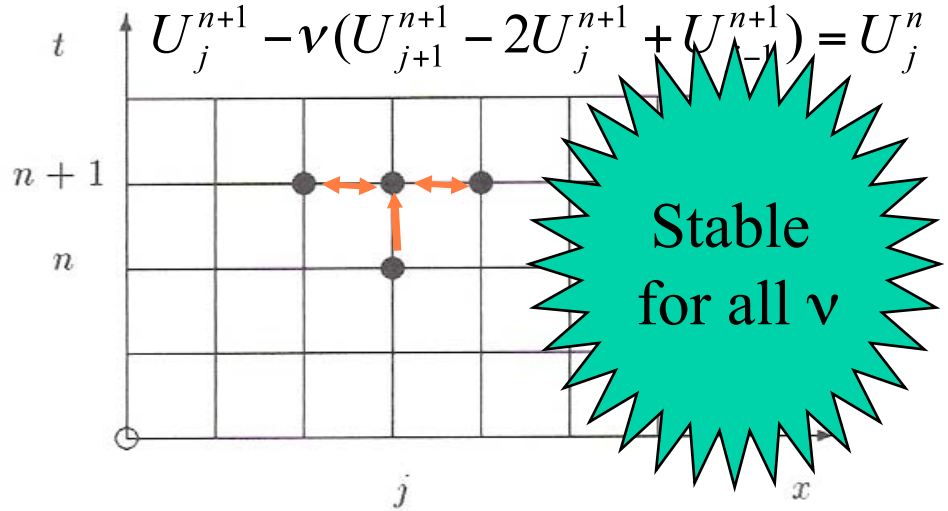
after 50 steps



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



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



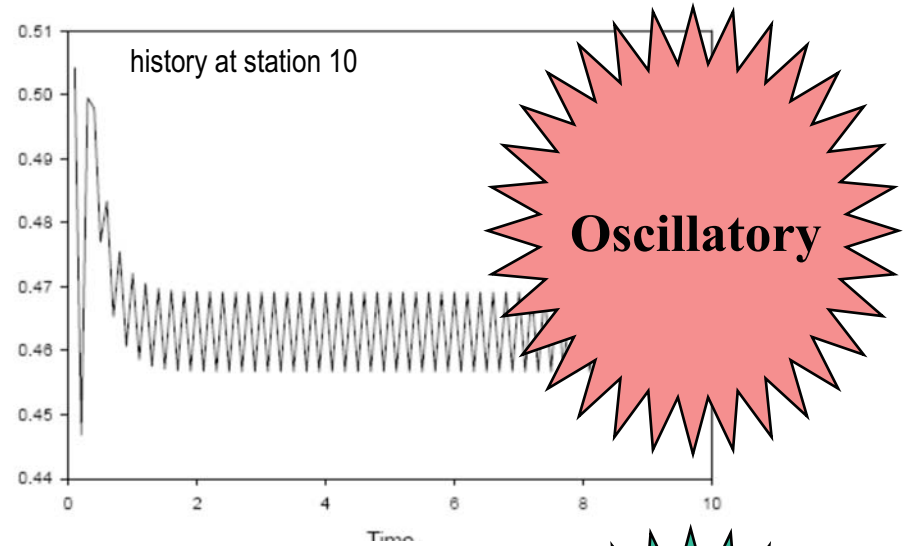
c/o K. Morton & D. Mayers, 2005

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

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

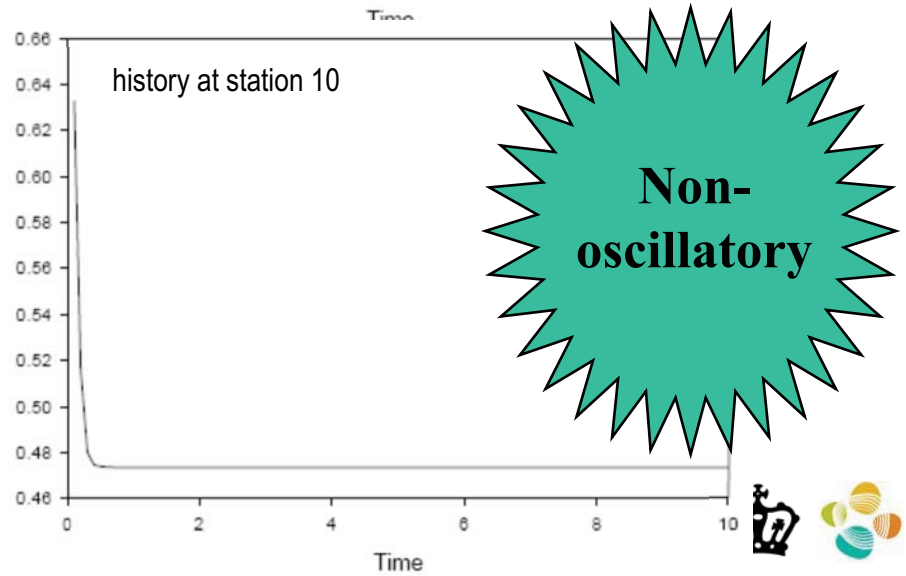
Linearly implicit, nonlinearly explicit:

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

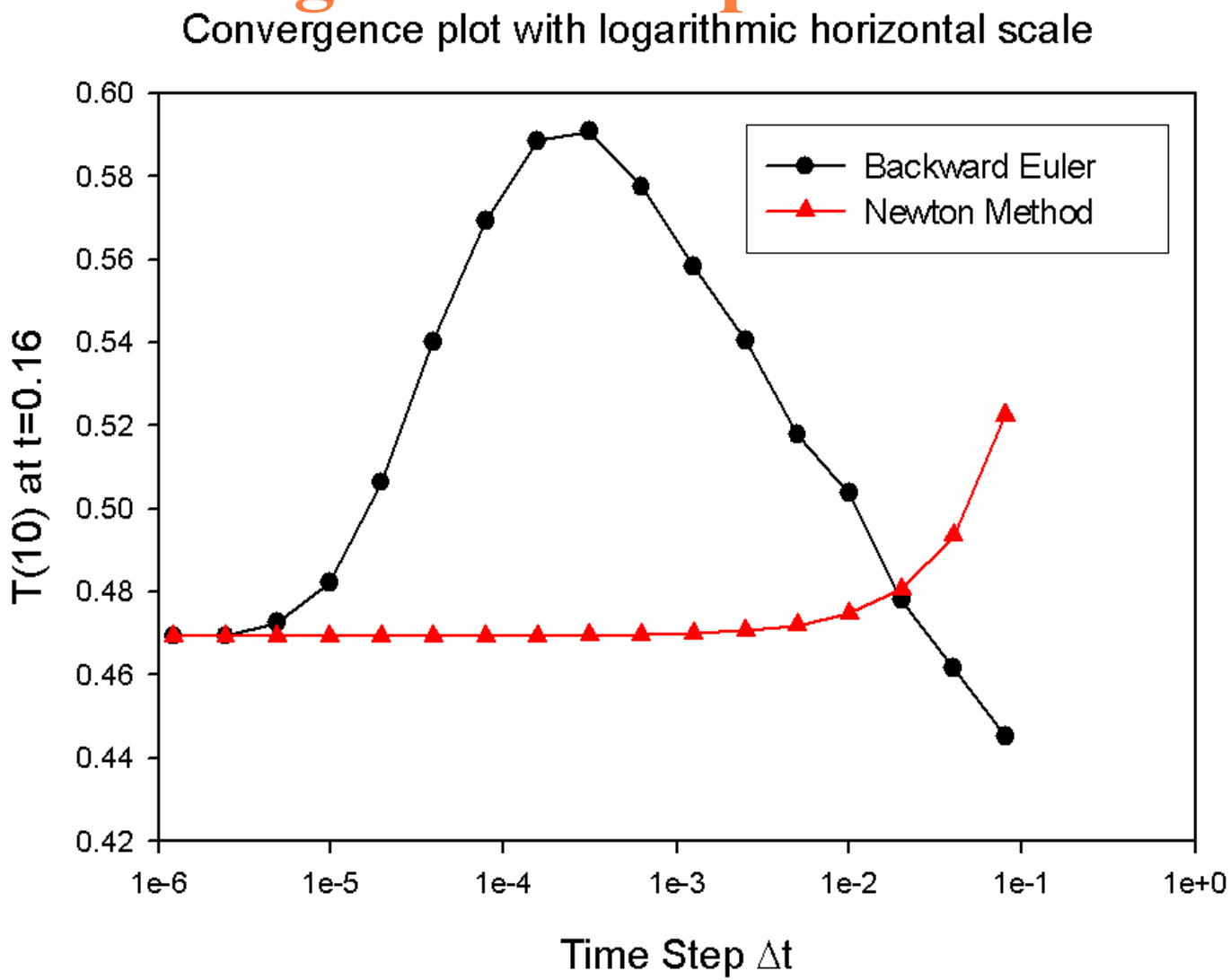


Linearly *and* nonlinearly implicit:

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



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

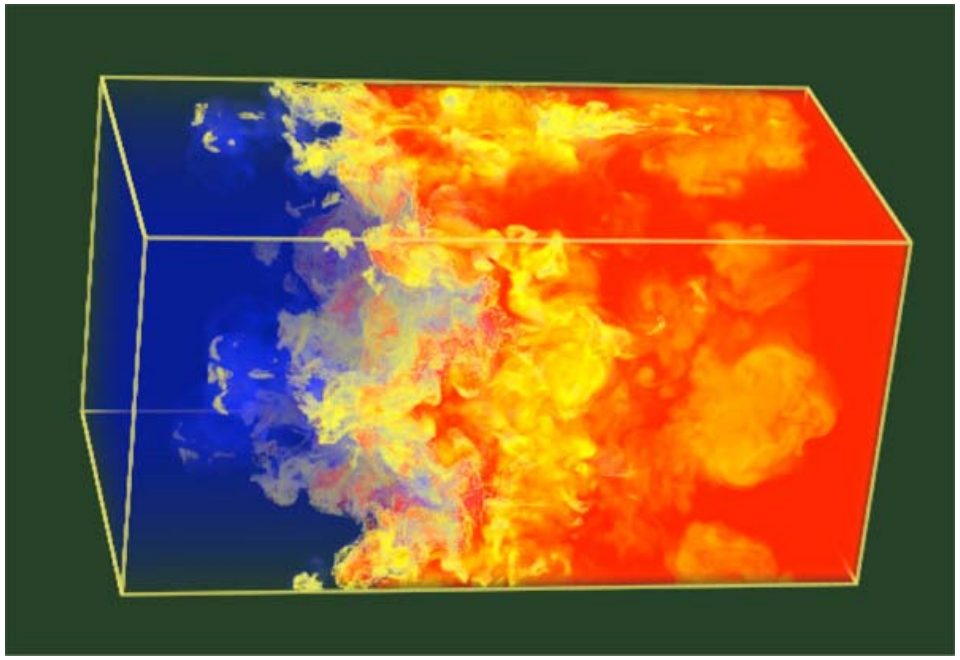
However – implicit methods can be unruly and expensive

	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

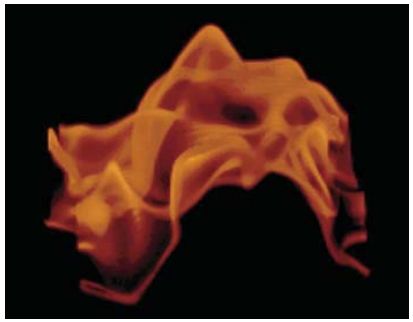
- **Multiple spatial scales**
  - **interfaces, fronts, layers**
  - **thin relative to domain size, $\delta << L$**
- **Multiple temporal scales**
  - **fast waves**
  - **small transit times relative to convection or diffusion, $\tau << T$**

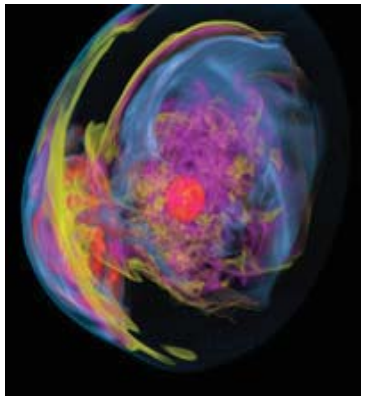


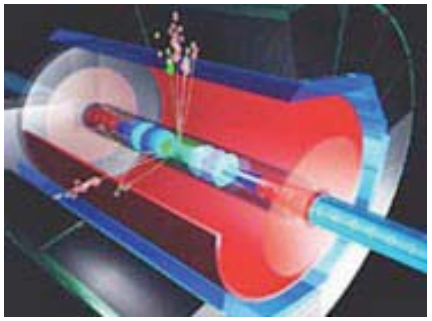
Richtmyer-Meshkov instability, c/o A. Mirin, LLNL

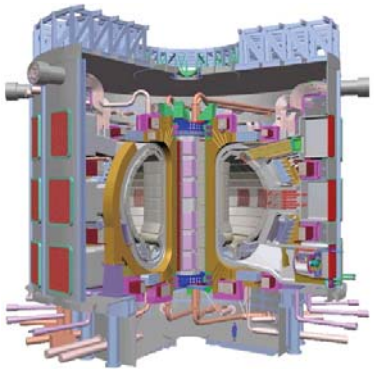
- **Analyst must *isolate dynamics of interest* and *model the rest* in a system that can be discretized over more modest range of scales**
- **Often involves filtering of high frequency modes, quasi-equilibrium assumptions, etc.**
- **May lead to infinitely "stiff" subsystem requiring implicit treatment**

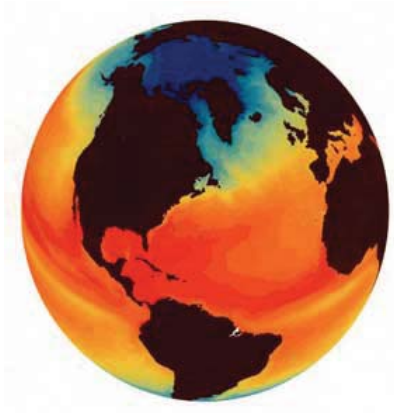
e.g., DOE's SciDAC* portfolio is multiscale



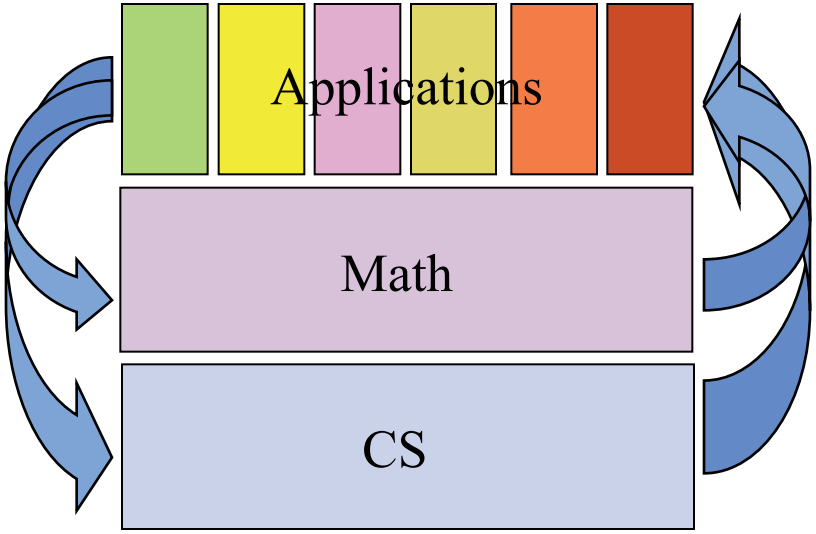








Many applications drive



Common technologies respond

* Scientific Discovery through Advanced Computing

Examples of scale-separated features of multiscale problems

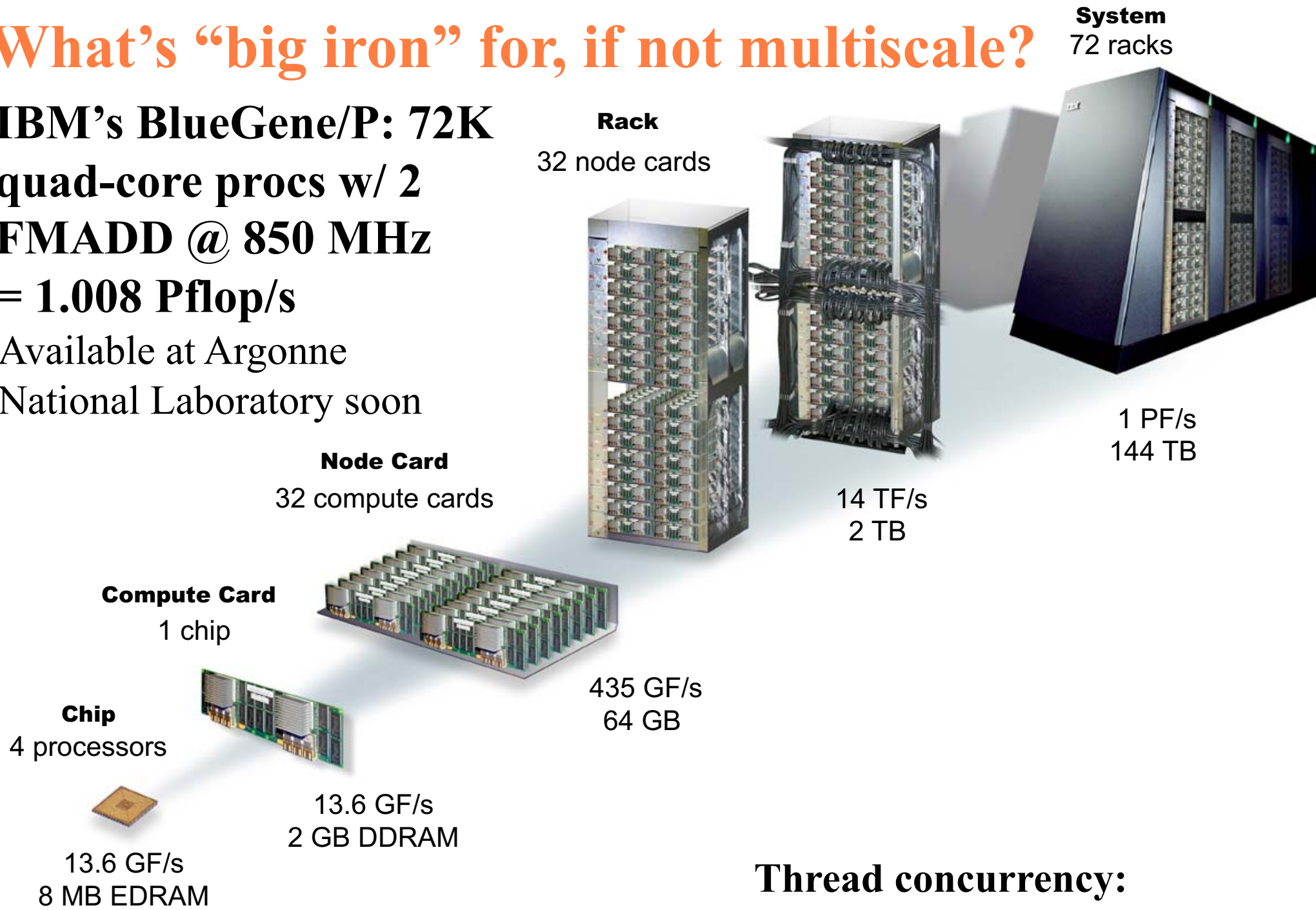
- Gravity surface waves in global climate
- Alfvén waves in tokamaks
- Acoustic waves in aerodynamics
- Fast transients in detailed kinetics chemical reaction
- Bond vibrations in protein folding (?)

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 = 1.008 Pflop/s

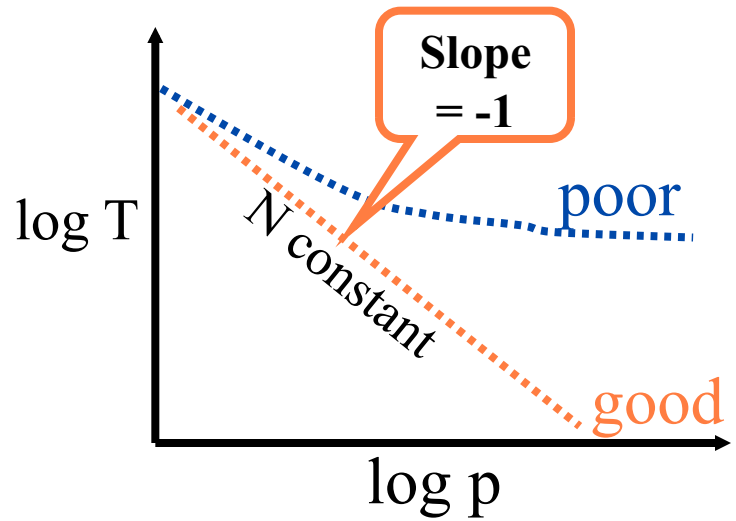
Available at Argonne National Laboratory soon



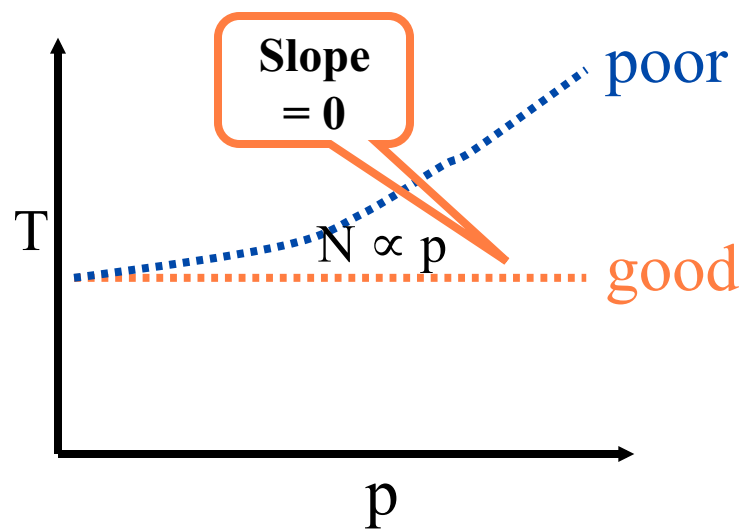
Thread concurrency: 288K (or 294,912) processors

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"**

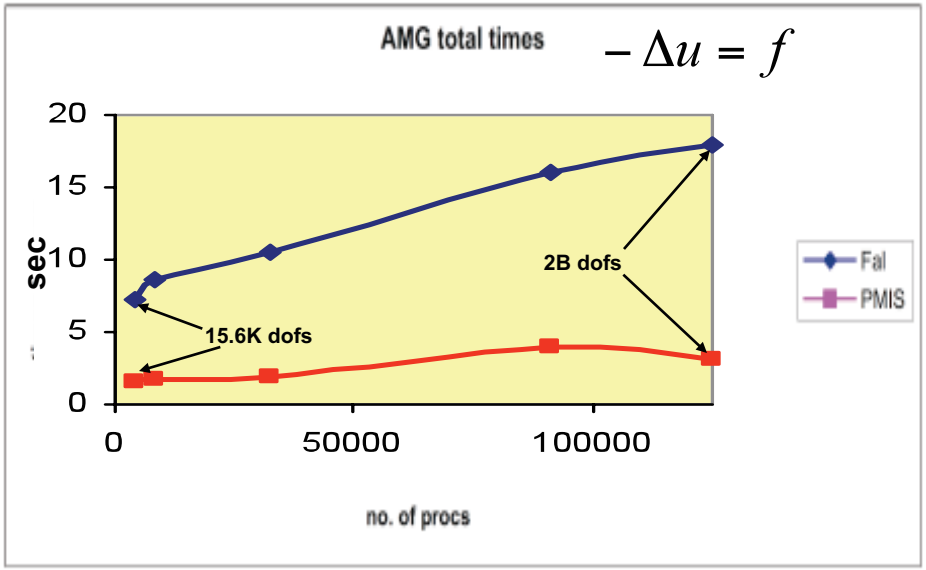


- **"Weak scaling" (memory bound)**
  - **execution time remains constant, as problem size and processor number are increased in proportion**
  - ***fixed size problem per processor***
  - **also known as "Gustafson scaling"**



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

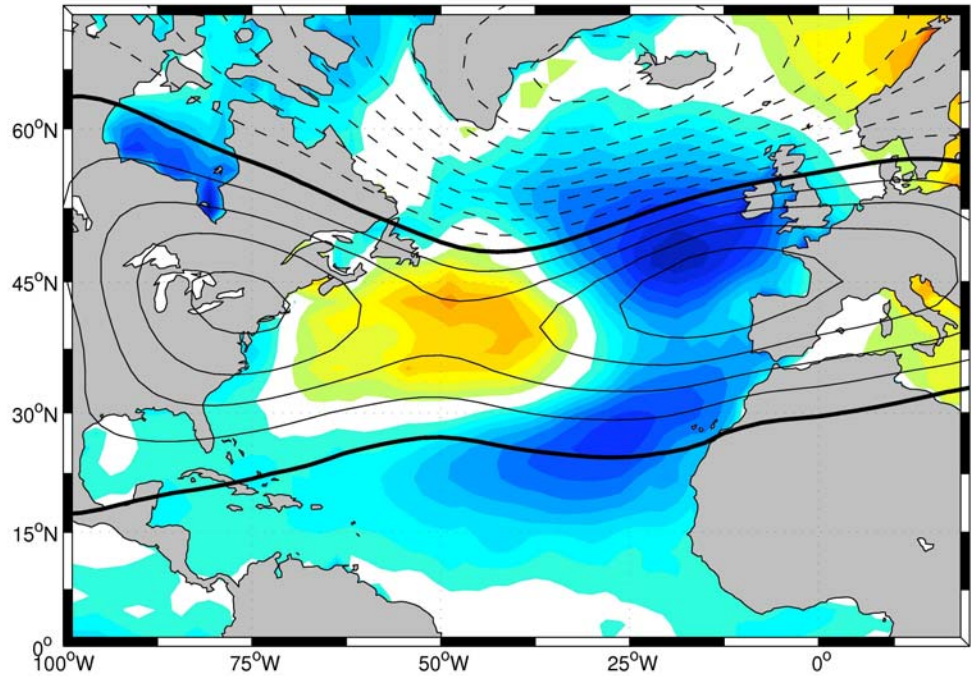
*assuming dynamics needs to be followed only on coarse scales

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

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

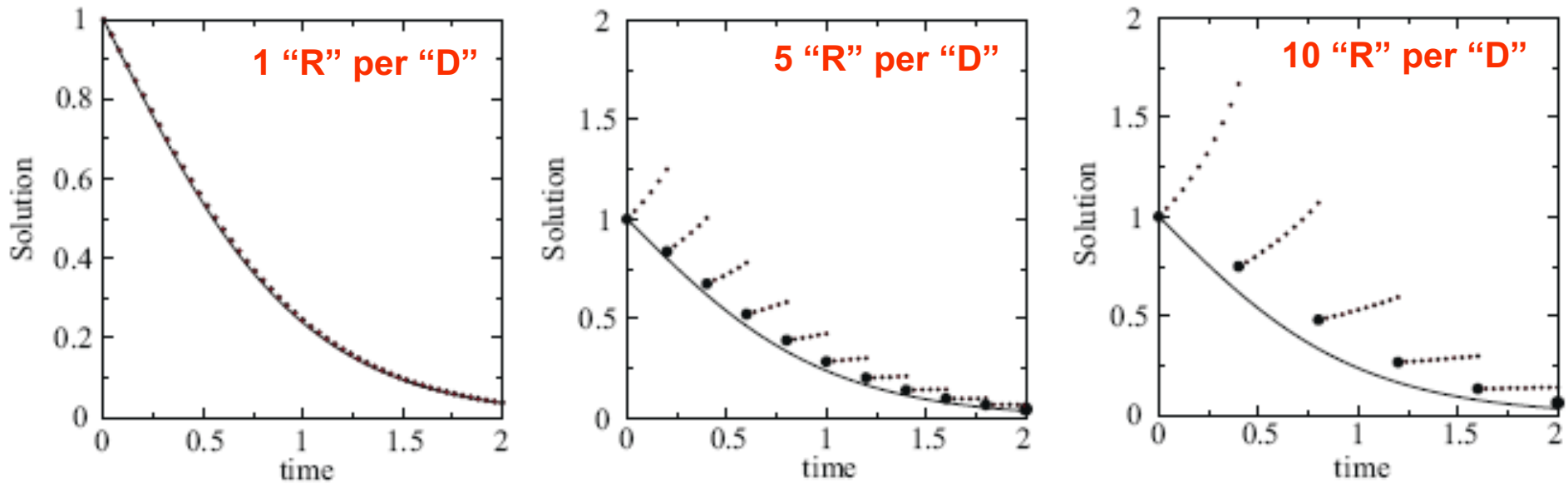
- **Numerical approx.** $U_k \approx u(t_k), \quad t_k = k\Delta t, \quad k = 0, 1, \ldots$
- **Phase 1 ("R")** $\dot{u}_R = u_R^2, \quad u_R(t_k) = U_k, \quad t_k < t \le t_{k+1}$
- **Phase 2 ("D")** $\dot{u}_D = -\lambda u_D, \quad u_D(t_k) = u_R(t_{k+1}), \quad t_k < t \le t_{k+1}$
- **Overall advance**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

- **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**



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

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

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

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

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

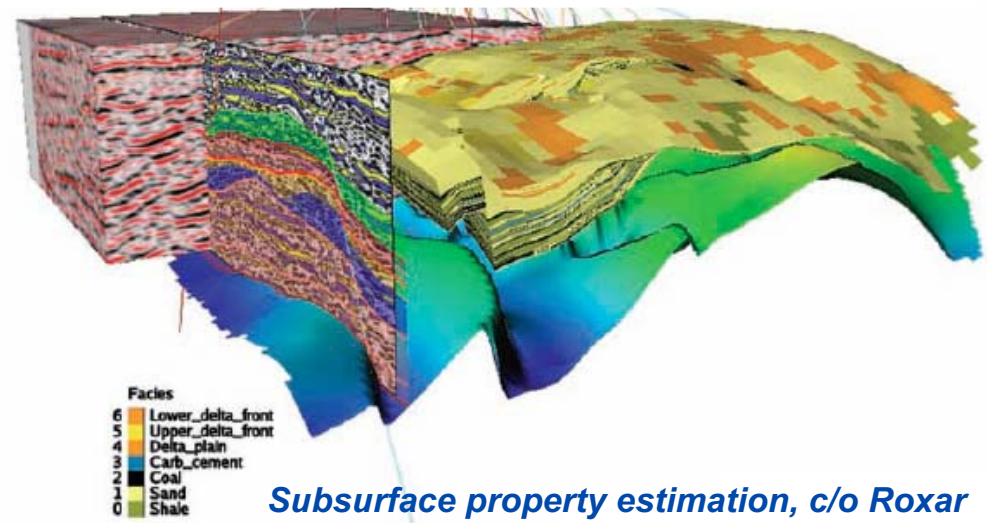
This is a prototype for a reaction-diffusion PDE

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

- **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 …** $\ell(u_k) = \langle \ell, u_k \rangle =$

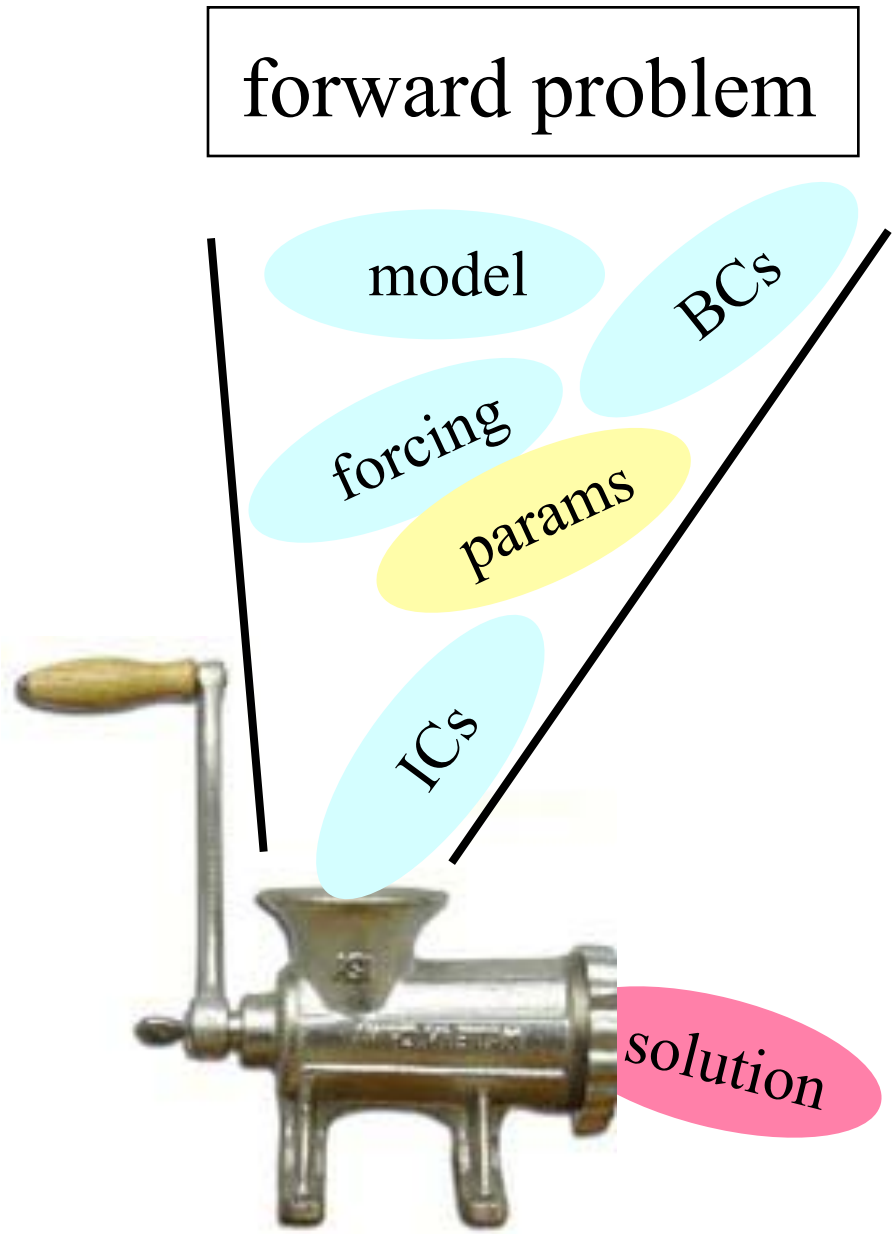
$$\langle L^* v, u_k \rangle = \langle v, Lu_k \rangle =$$

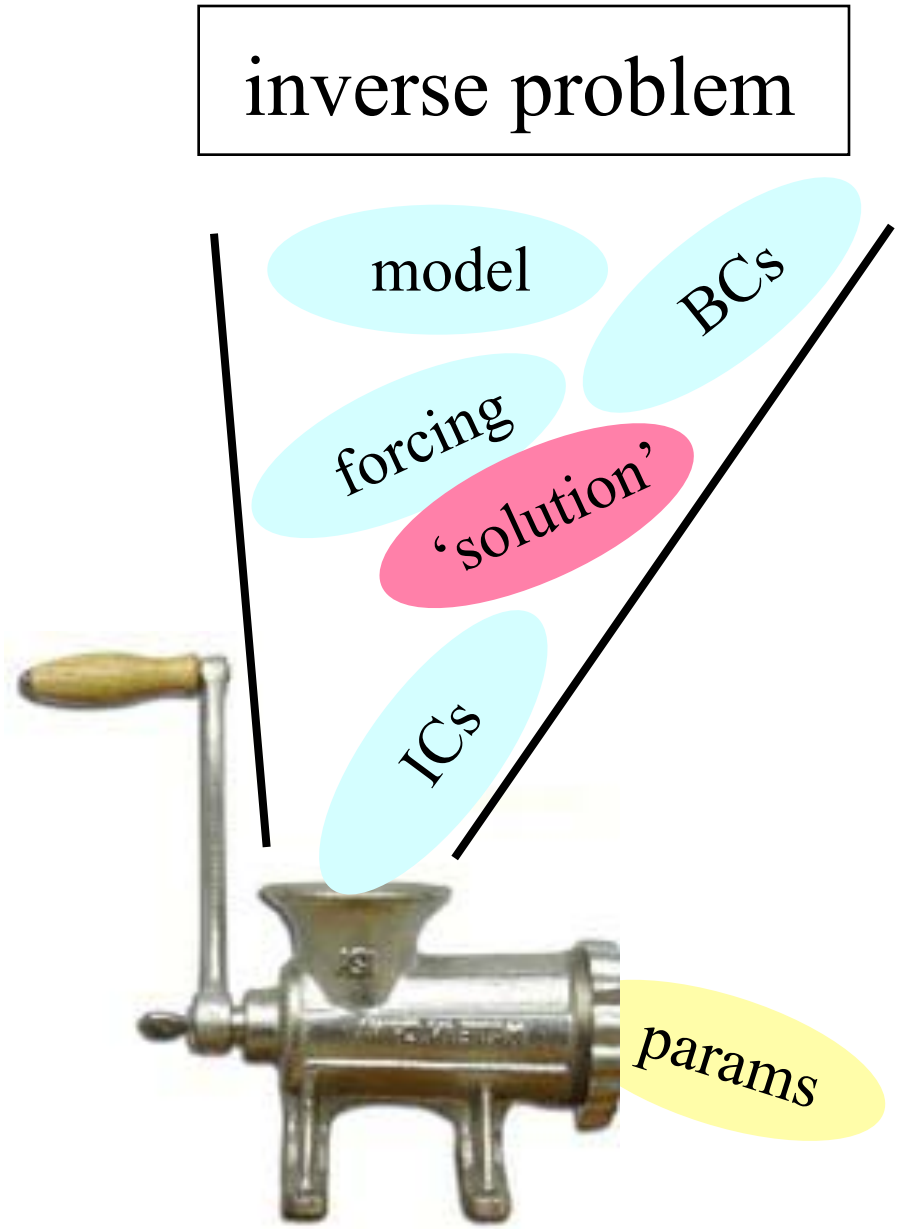
… reduces to an inner product for each forcing f ! $\langle v, f_k \rangle$

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
- Many types of generalization to nonlinear operators are 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





+ regularization

Significance 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**

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

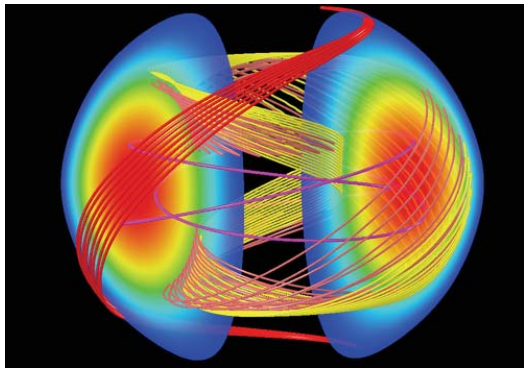
$$g = -g_u + J_u^T J_x^{-T} g_x - (J_u^T J_x^{-T} W_{xx} - W_{ux}) J_x^{-1} f$$

- **Then**

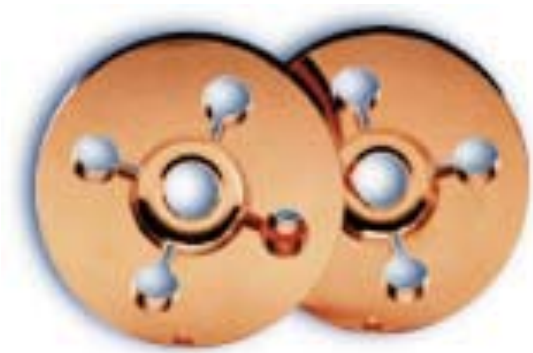
$$J_x \delta x = -f - J_u \delta u$$

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

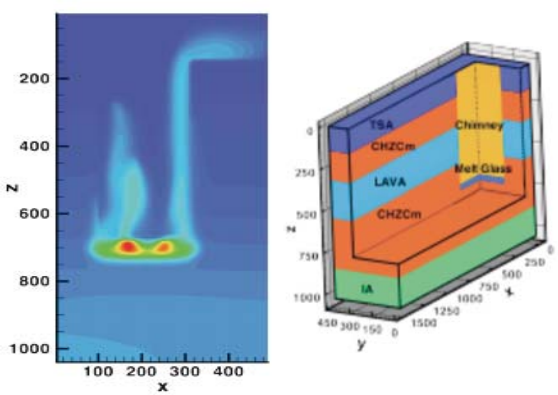
Applications requiring scalable solvers – conventional and progressive



- **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

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 (~$32M/10 years)

Argonne NATIONAL LABORATORY

TOPS

Towards Optimal Petascale Simulations

Lawrence Livermore National Laboratory

BERKELEY LAB

Sandia National Laboratories

Columbia University

University of Colorado

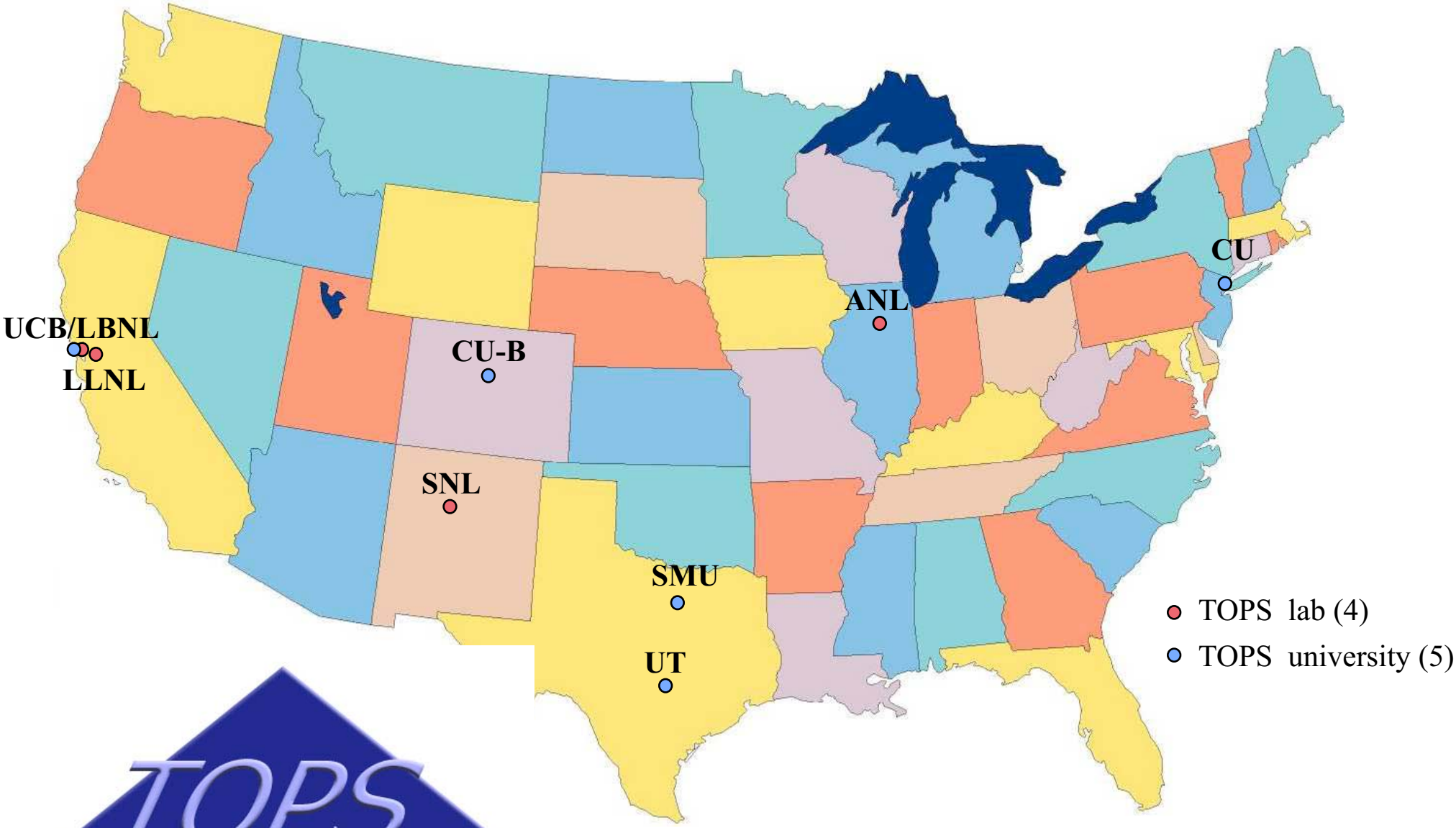
Berkeley University of California

The Institute for Computational Engineering and Sciences ICES

University of Texas

Southern Methodist University

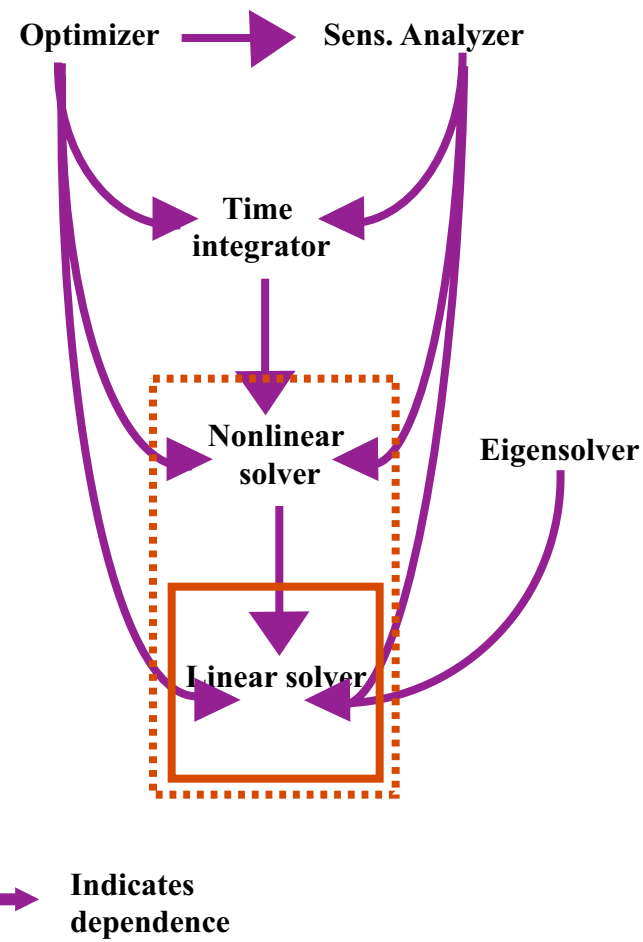
TOPS institutions



TOPS
Towards Optimal Petascale Simulations

TOPS is building a toolchain of proven solver components that interoperate

- We aim to 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, ScaLAPACK, 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.

Adams

Baker

Cai

Demmel

Falgout

Ghattas

Heroux

Hu

Kaushik

Keyes

Knepley

Li

Manteuffel

McCormick

McInnes

Moré

Munson

Ng

Reynolds

Rouson

Salinger

Smith

Woodward

C. Yang

U. Yang

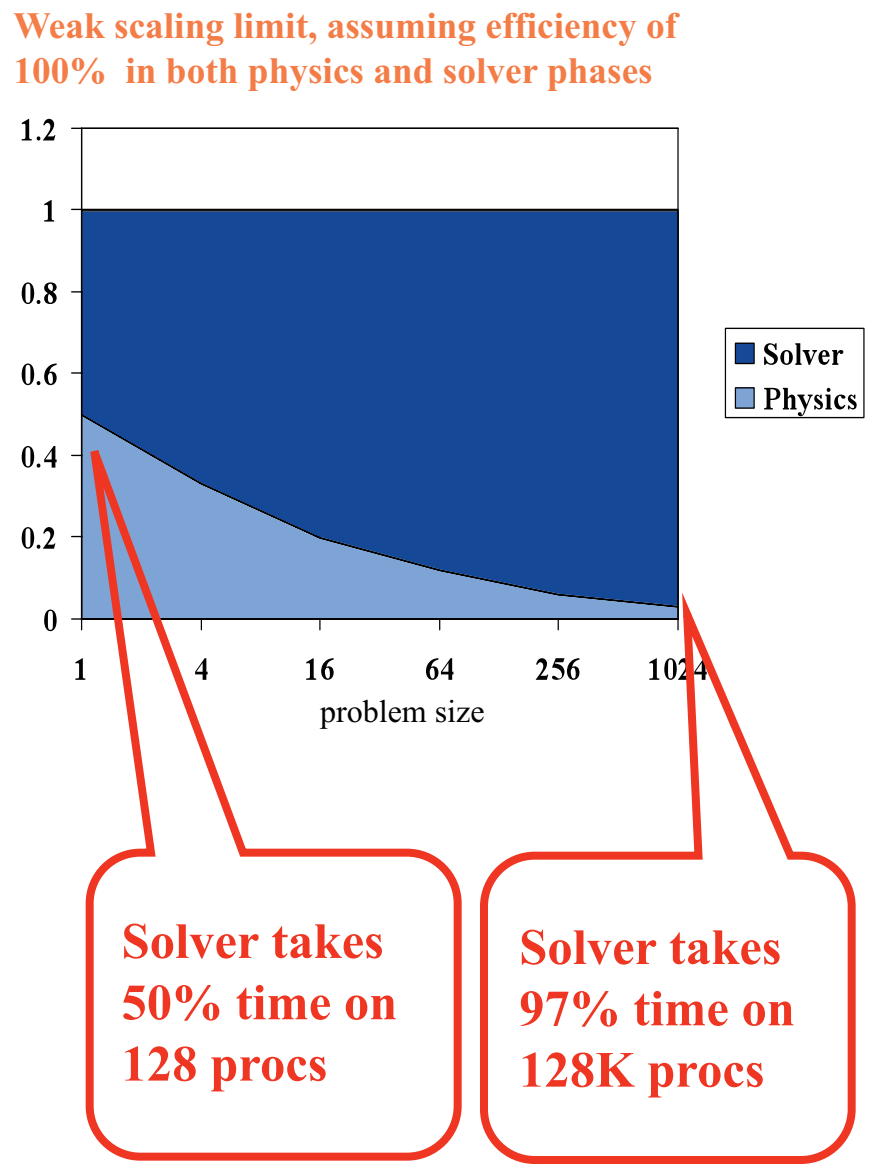
Zhang

Faces of TOPS

TOPS
Terascale Optimal PDE Simulations

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

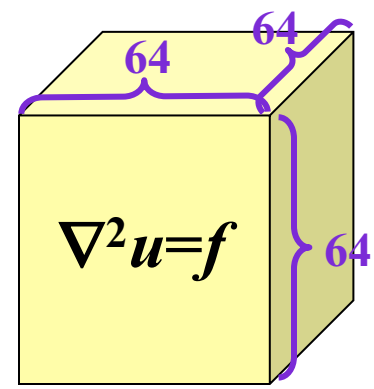
- **Given, for example:**
  - **a "physics" phase that scales as $O(N)$**
  - **a "solver" phase that scales as $O(N^{3/2})$**
  - **computation is almost all solver after several doublings**
- **Most applications groups have not yet "felt" this curve in their gut**
  - **as users actually get into queues with more than 4K processors, this will change**



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

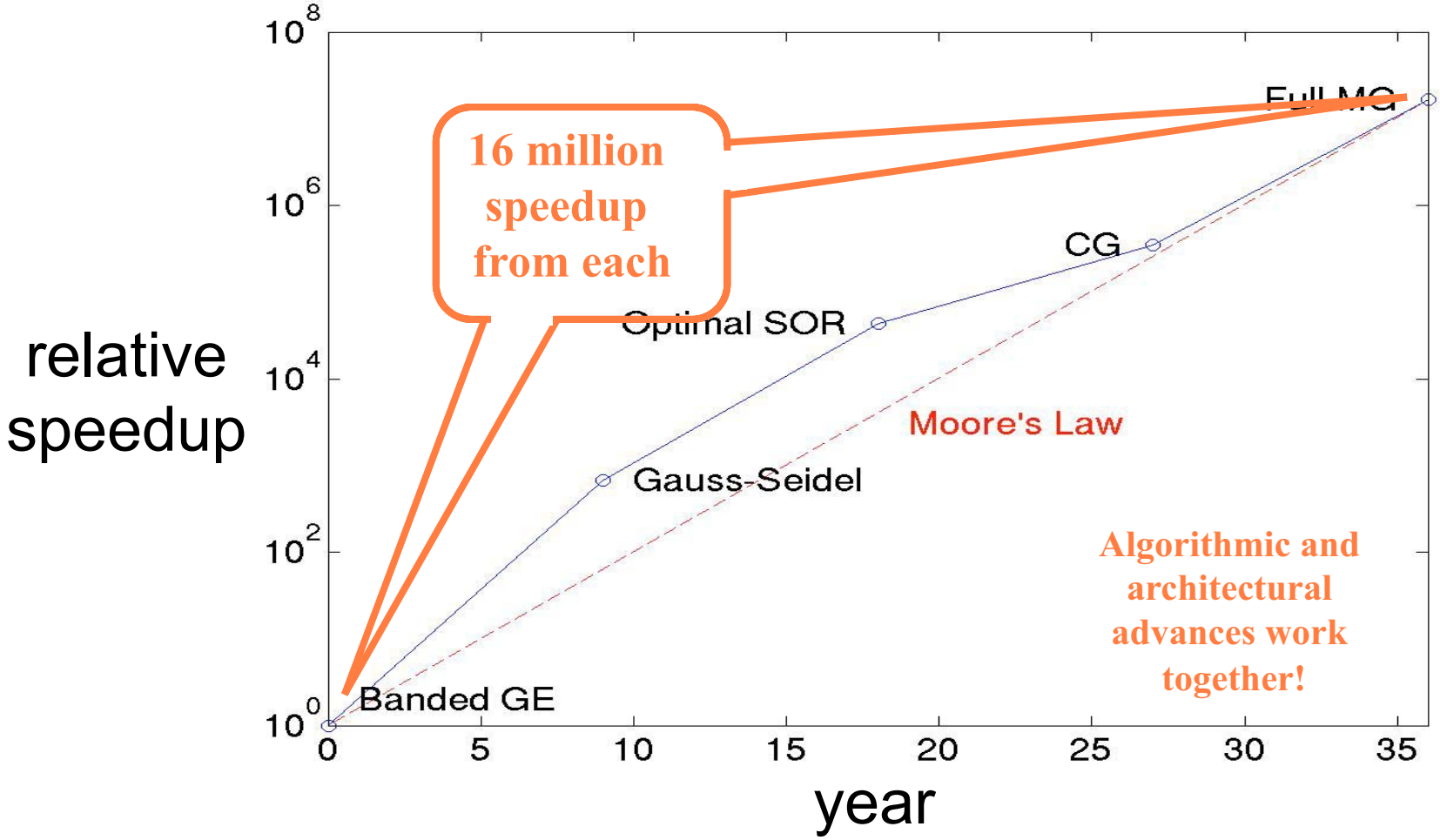


- If n=64, this implies an overall reduction in flops of ~ 16 million*

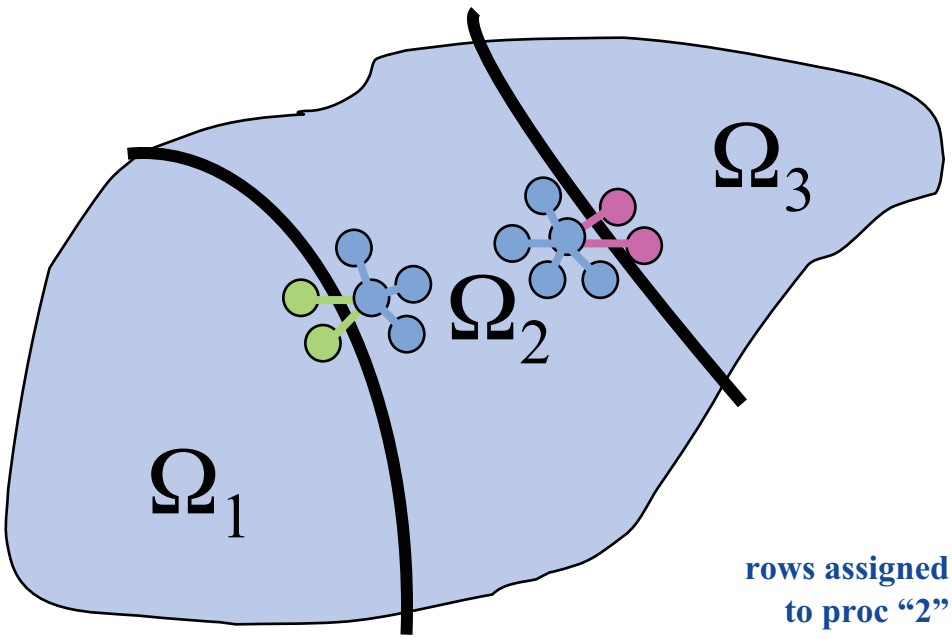
*Six months is reduced to 1 second

Algorithms and Moore's Law

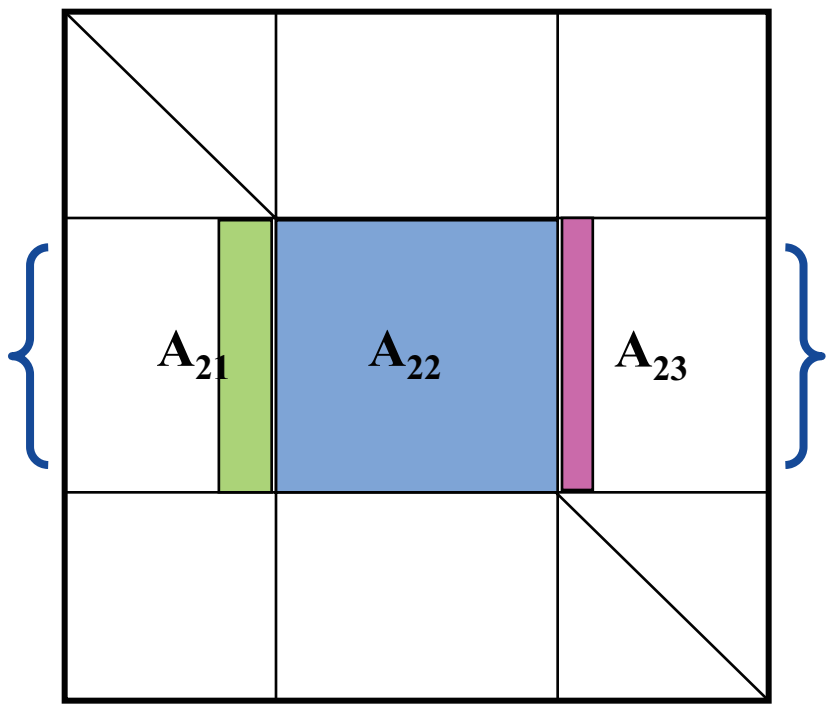
- This advance took place over a span of about 36 years, or 24 doubling times for Moore's Law
- $2^{24} \approx 16$ million $\Rightarrow$ the same as the factor from algorithms alone!



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



rows assigned to proc "2"

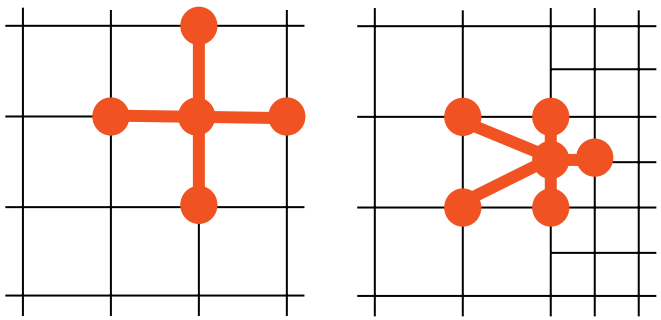


Partitioning of the grid induces block structure on the system matrix (Jacobian)

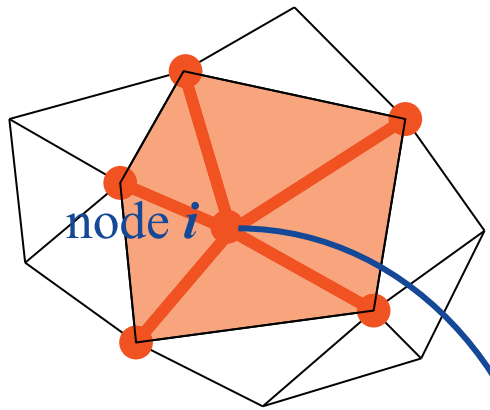
Computation scales with *area*; communication scales with *perimeter*; ratio *fixed* in weak scaling

Domain decomposition relevant to any local stencil formulation

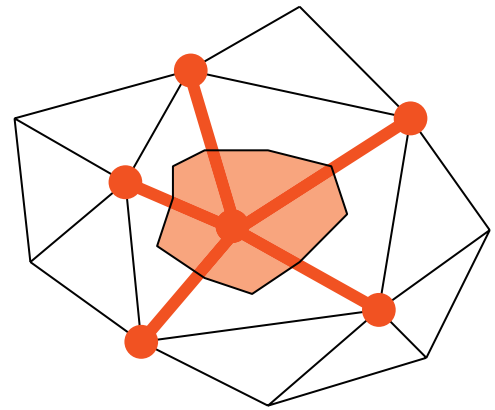
finite differences



finite elements

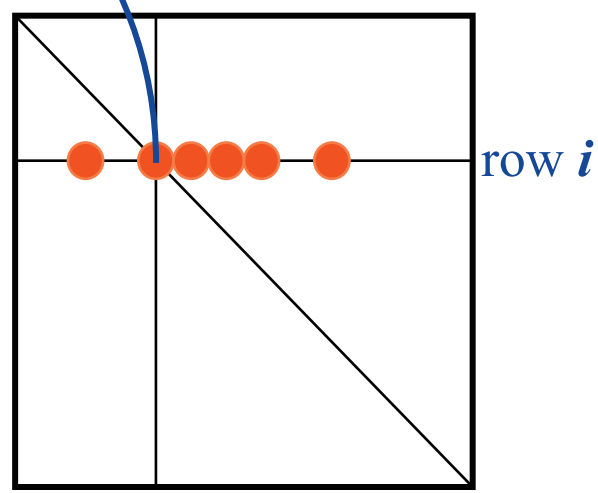


finite volumes



- **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**

J=



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"

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

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

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

Newton

nonlinear solver

asymptotically quadratic

Krylov

accelerator

spectrally adaptive

Schwarz

preconditioner

parallelizable

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

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

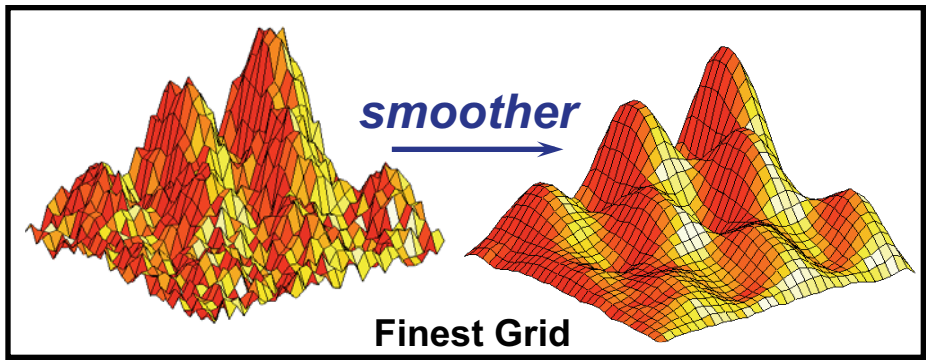
$$x \leftarrow x + B^{-1}(b - Ax)$$

- **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**

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

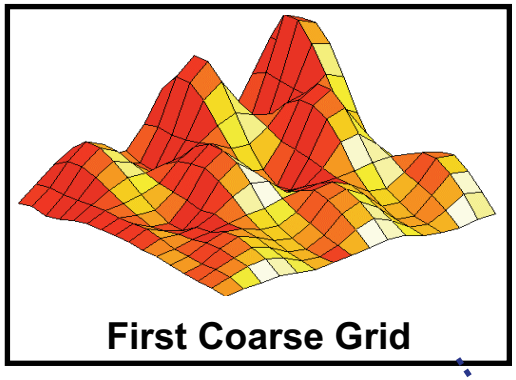
- **Scale recurrence, e.g., with** $B_2^{-1} = R^T (RAR^T)^{-1} R$ **, leads to *multilevel methods***
- **Optimal polynomials of** $(B^{-1}A)$ **lead to various *preconditioned Krylov methods***

"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)

Prolongation
transfer from coarse to fine grid

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

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*

Carl Jacobi

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

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

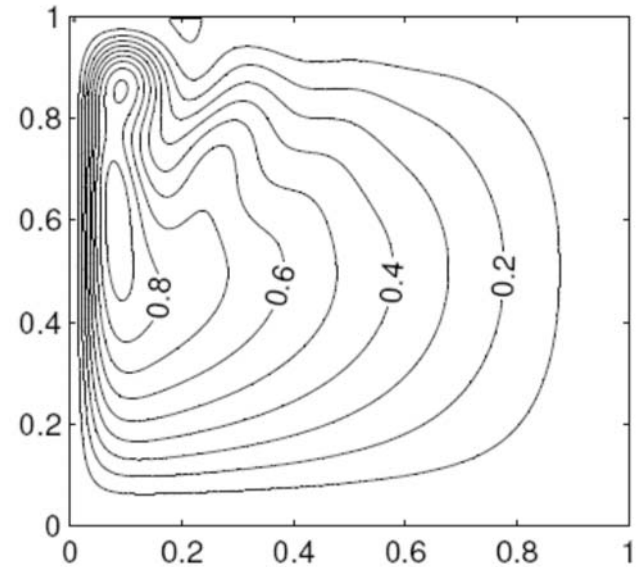
$$M : F(u^k) \mapsto \delta u$$

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

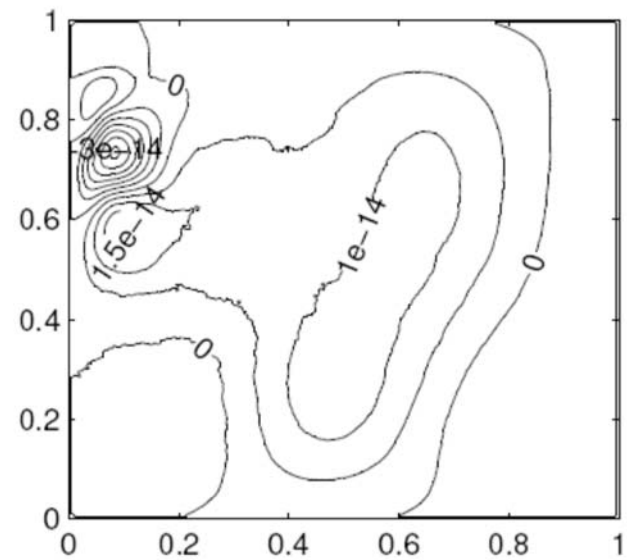
- M is, in effect, a preconditioner and can be applied directly within a Jacobian-free Newton context
- This is the "physics-based preconditioning" strategy discussed in the E^3 report

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!**

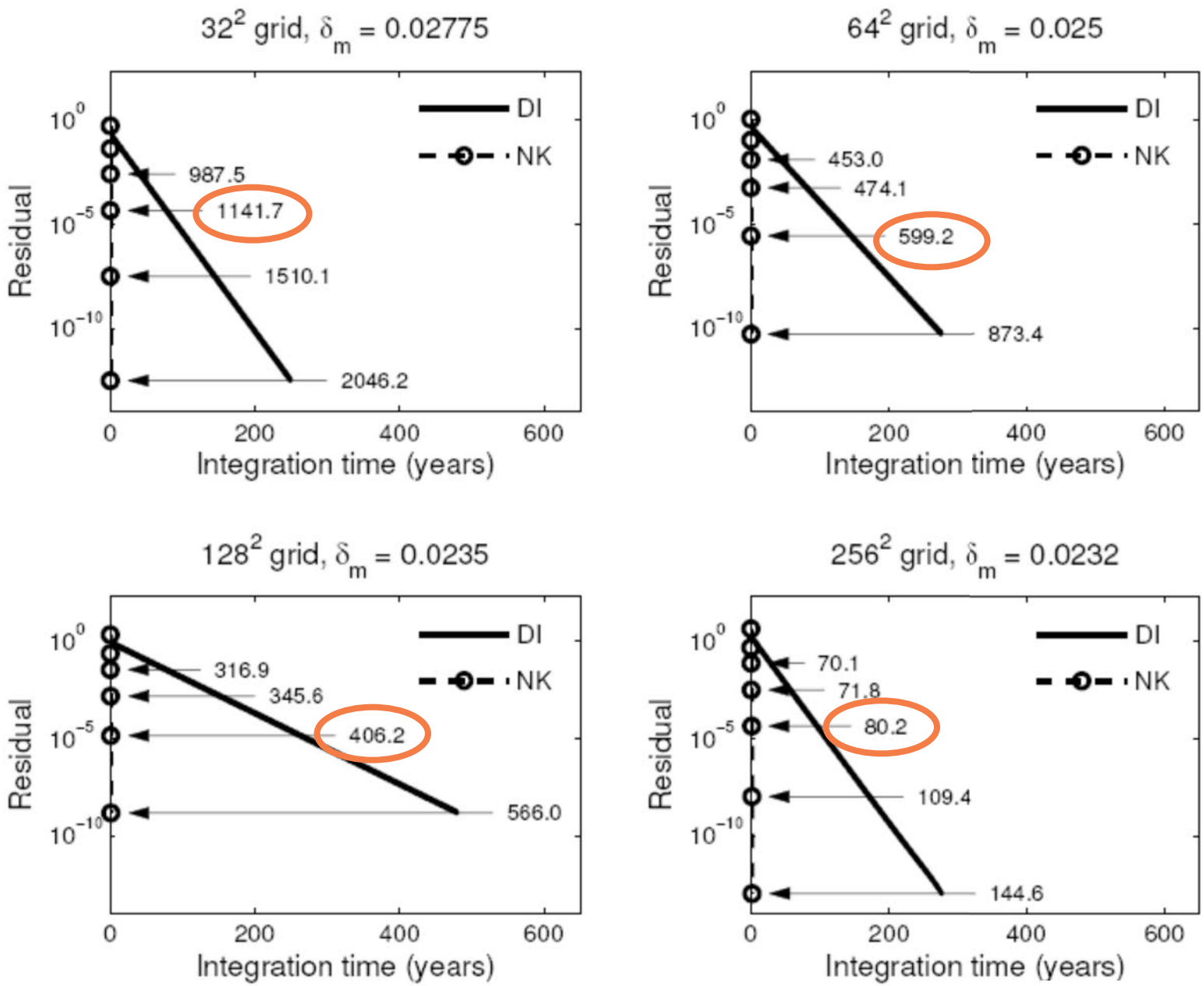


converged streamfunction



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

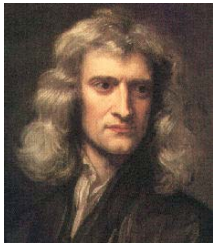
c/o T. Merlis (Columbia'05, *now* Caltech, Dept. Environmental Science & Engineering)

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**
  - **control of experiments**

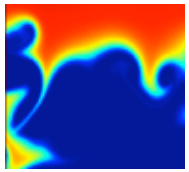
A “perfect storm” for scientific simulation

(dates are symbolic)



1686

scientific models

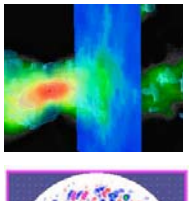


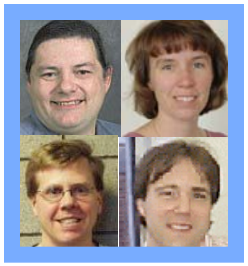
1947

numerical algorithms

1976

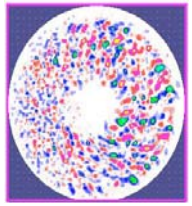
computer architecture

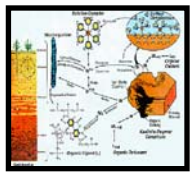




1992

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**

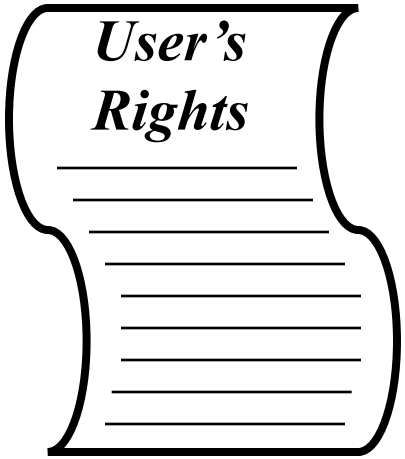
  with access to detailed profiling information

- **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**



SciDAC's computational math "centers"

- **Interoperable Tools for Advanced Petascale Simulations (ITAPS)**

  PI: ***L. Freitag-Diachin, LLNL***

  For complex domain geometry
- **Algorithmic and Software Framework for Partial Differential Equations (APDEC)**

  PI: ***P. Colella, LBNL***

  For solution adaptivity
- **Combinatorial Scientific Computing and Petascale Simulation (CSCAPES)**

  PI: ***A. Pothen, Purdue U***

  For partitioning and ordering
- **Towards Optimal Petascale Simulations (TOPS)**

  PI: ***D. Keyes, Columbia U***

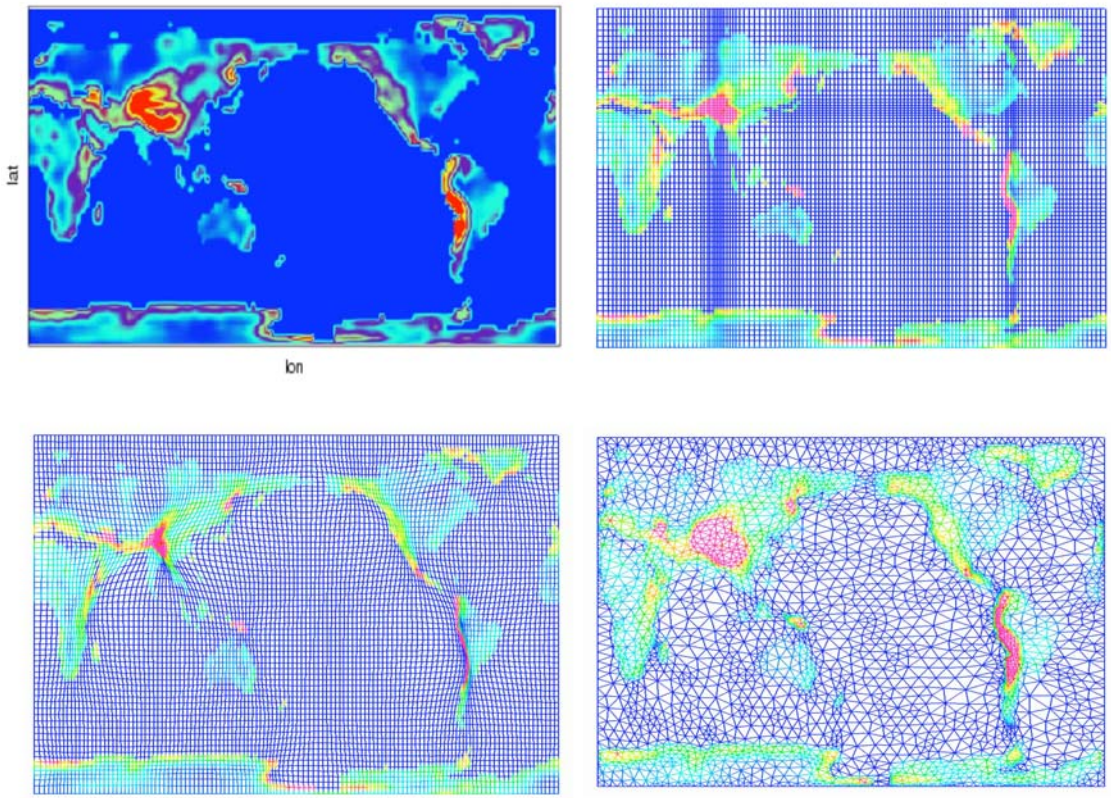
  For scalable solution

See: www.scidac.gov/math/math.html

ITAPS

Interoperable Tools for Advanced Petascale Simulations

Develop framework for use of multiple mesh and discretization strategies within a single PDE simulation. Focus on high-quality hybrid mesh generation for representing complex and evolving domains, high-order discretization techniques, and adaptive strategies for automatically optimizing a mesh to follow moving fronts or to capture important solution features.

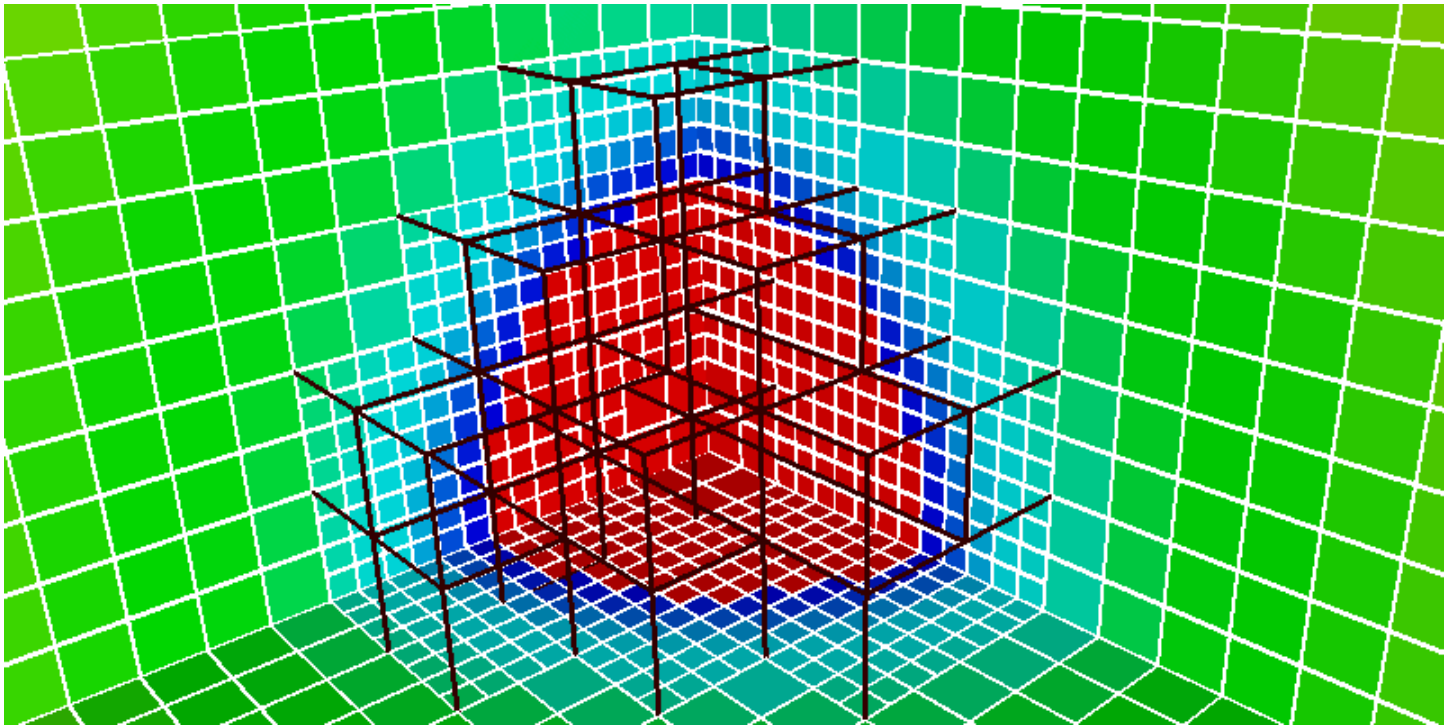


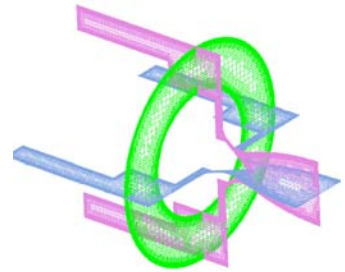
c/o L. Freitag, LLNL

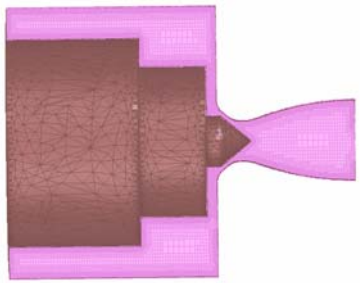
APDEC

Algorithmic and Software Framework for PDEs

Develop framework for PDE simulation based on locally structured grid methods, including adaptive meshes for problems with multiple length scales; embedded boundary and overset grid methods for complex geometries; efficient and accurate methods for particle and hybrid particle/mesh simulations.





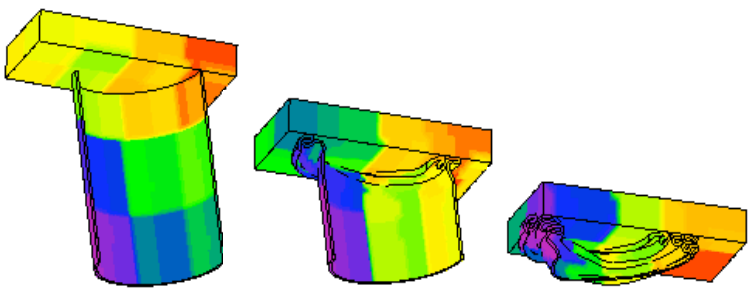


c/o P. Colella, LBNL

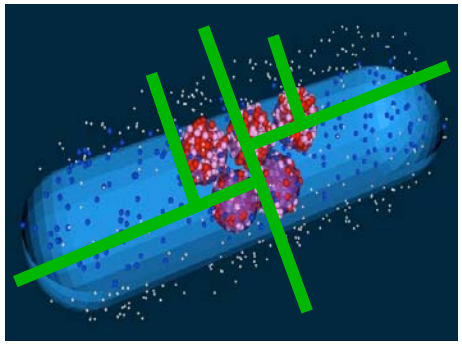
CSCAPES

Combinatorial Scientific Computing and Petascale Simulation

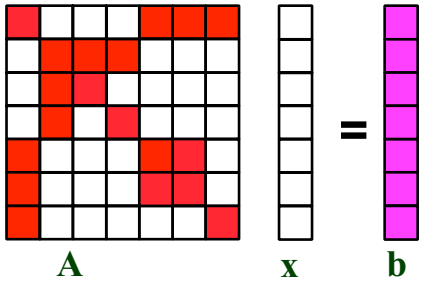
Develop toolkit of partitioners, dynamic load balancers, advanced sparse matrix reordering routines, and automatic differentiation procedures, generalizing currently available graph-based algorithms to hypergraphs



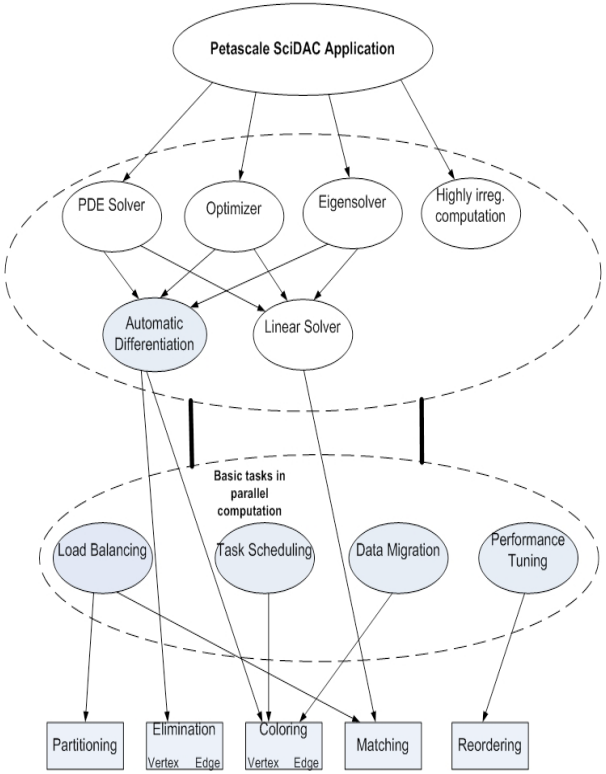
Contact detection



Particle Simulations



Linear solvers & preconditioners



c/o A. Pothen, Purdue

EOF