David Keyes
Extreme Computing Research Center
King Abdullah University of Science and Technology
12 January 2022

Updating the Nonlinearly Implicit Manifesto

What has changed in time integration?

- Multi-partitioned RK
- Multirate/infinitesimal RK
- Partitioned Rosenbrock
- Multirate/infinitesimal Rosenbrock
- Splitting schemes
- Acceleration via surrogates
- Symplectic, split Hamiltonians
- …

(from Sandu, 2022)

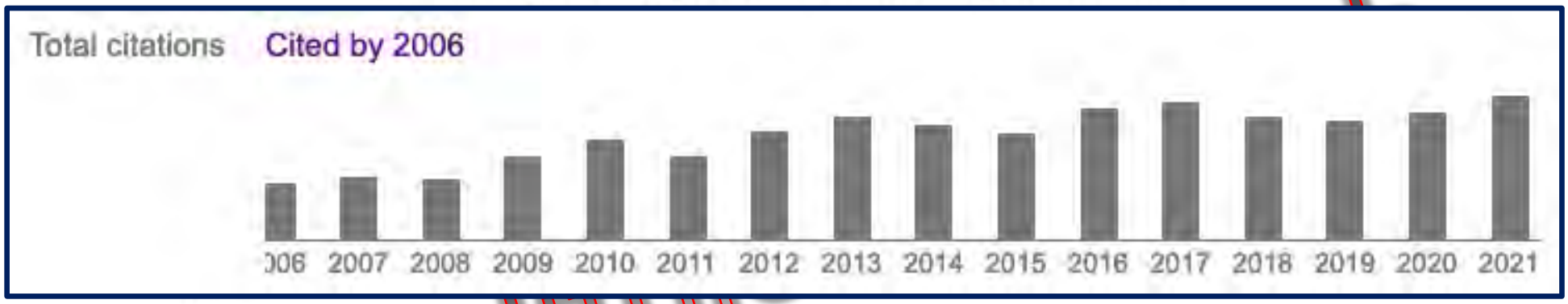
(from Constantinescu, 2022)

- **Many tasty new apples on the tree and delicious new entrees at the picnic!**
- **A "Cambrian explosion" of operator splitting methods that exploit natural separations on the 2x2 state space of { stiff | nonstiff } and { linear | nonlinear } terms in order to achieve solutions of desired temporal accuracy at lower cost relative to fully implicit solvers**

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 193 (2004) 357–397
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Review

Jacobian-free Newton–Krylov methods: a survey of approaches and applications



Jacobian-free Newton–Krylov (JFNK) methods are synergistic combinations of Newton-type methods for superlinearly convergent solution of nonlinear equations and Krylov subspace methods for solving the Newton correction equation. The link between the two methods is the Jacobian-vector product, which may be probed approximately without forming and storing the elements of the true Jacobian, through a variety of means. Various approximations to the Jacobian matrix may still be required for preconditioning the resulting Krylov iteration. As with Krylov methods for linear problems, successful application of the JFNK method to any given problem is dependent on adequate preconditioning. JFNK has potential for application throughout problems governed by nonlinear partial differential equations and integro-differential equations. In this survey paper, we place JFNK in context with other nonlinear solution algorithms for both boundary value problems (BVPs) and initial value problems (IVPs). We provide an overview of the mechanics of JFNK and attempt to illustrate the wide variety of preconditioning options available. It is emphasized that JFNK can be wrapped (as an accelerator) around another nonlinear fixed point method (interpreted as a preconditioning process, potentially with significant code reuse). The aim of this paper is not to trace fully the evolution of JFNK, nor to provide proofs of accuracy or optimal convergence for all of the constituent methods, but rather to present the reader with a perspective on how JFNK may be applicable to applications of interest and to provide sources of further practical information.

"The goal of high performance computing is not to solve model problems faster, but to solve complex problems *at all*."

Bill Gropp

"Complex" attributes, still within the realm of well-posed problems with well-defined, in principle computable solutions, given enough resolution and computational capability may include:

- nonlinearity
- high dimensionality
- multiple scales in space and time
- multiple coupled physical phenomena
- multiple field components
- wild inhomogeneity
- strong anisotropy
- etc.

"We want methods that *shoot the problem in the head*."

Bob Brown

In other words, robustness and general applicability for user productivity before all else, including efficiency, elegance, etc.

“The advantage of unconditional stability may be worth it, even at the cost of larger coefficients in the error term.”

David Shirokoff

“Full implicitness provides comfort, theory, and stability… Linear implicitness should do the job; then move beyond as needed.”

Giovanni Russo

“We’re seeking the least implicitness that delivers the stability we need… Implicitness steps over ‘inconsequential’ fast components (acoustic waves; Alfvén waves).”

Emil Constantinescu

What is wrong with fully implicit?

- **Complexity of code and user interface**
  - relative to fully explicit
- **Computational cost**
  - storage, operations, synchronizing communication
  - if fastest stiff scales need resolving anyway, fully implicit is not relevant
  - need 1 to 2 orders of magnitude of scale separation between 'inconsequential' fast phenomena and phenomena of interest to cover the generation of the Krylov subspace on each Newton step, depending on the Jacobian conditioning and the preconditioning
- **Robustness**
  - assumes you can trust Newton to converge
- **Temporal accuracy**
  - traditionally associated with first-order backward Euler

What is good about fully implicit?

- **Complexity of code and user interface**
  - relative to operator split methods that use both implicit and explicit
  - no worries if proper operator splits evolve dynamically
- **Computational cost**
  - allows integration on time scales of interest, sometimes stepping over many orders of magnitude of fast scales
  - full Jacobians not needed
- **Robustness**
  - unconditional stability
  - globalization methods for Newton have come a long way
- **Temporal accuracy**
  - no operator splitting error
  - BDF2 should often be the default, at the cost just one extra state vector and negligible extra operations or synchronizing communication
  - other fully implicit higher order schemes possibly relevant, at the cost of further back-vectors of state or RHS

“I’m a believer in ‘co-simulation’. We must learn to work directly from users’ codes.”

Ray Spiteri

“We use a preconditioner based on our favorite part and simply leave the other terms unpreconditioned… Whatever the code has it in already – use it!”

Dan Reynolds

“General Linear Method filters can deliver higher order with just backward Euler, without changing the user’s code.”

Sigal Gottlieb

“The name of the game is to minimize the computational cost for a given accuracy.”

Ray Spiteri

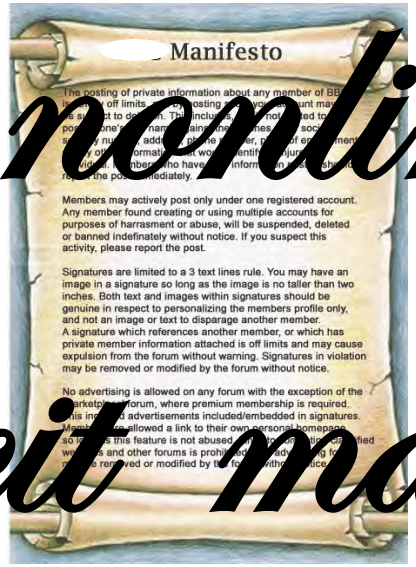
“The name of the game is to ask the user to write their code exactly once, no matter what algorithm they use.”

Matt Knepley

Flow of presentation

- **Blast thru original nonlinearly implicit manifesto (2007)**
  - ***multi-scale*** **– avoid stability timestep limit when it is tighter than the accuracy limit for the phenomena of interest**
  - ***multi-physics*** **– avoid first-order splitting error**
  - ***multi-solve*** **– inverse Jacobian action needed, anyway**
- **Back off from apparent claims of universal applicability**
  - **perhaps excessive hubris about Jacobian-free Newton-Krylov schemes**
  - **benefit of unconditional stability may be expensive compared to split schemes that compute the same outputs of interest as accurately**
- **Finally, reassert its utility**
  - **efficient implicit solvers still needed in ImEx and other tasty new apples, even if just for part of the operator**
  - **our toolkits contain ever improving globalizations for Newton's method (esp. thru nonlinear preconditioning)**
  - **likewise, ever improving approximate linear solvers**

A nonlinearly implicit manifesto*



David Keyes
Applied Physics & Applied Mathematics
Columbia University
&
Towards Optimal Petascale Simulations (TOPS) Center
U.S. DOE SciDAC Program

*a public declaration of principles and intentions

SC07

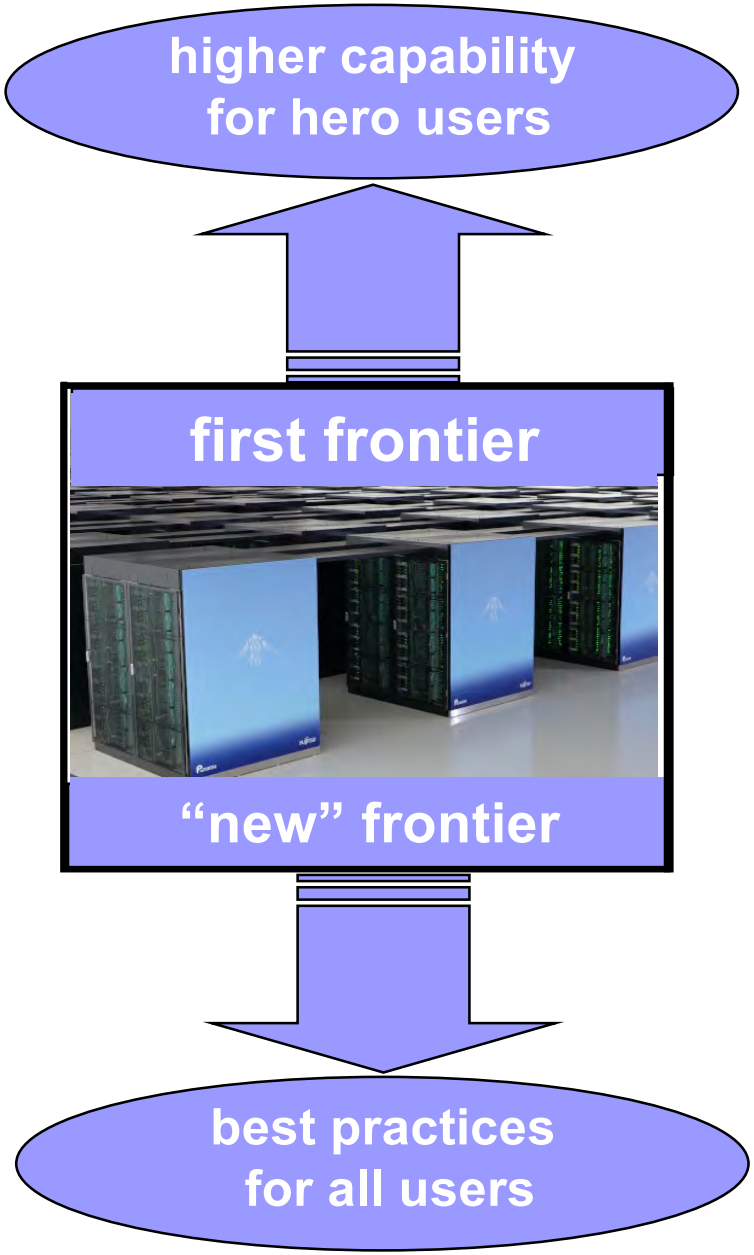
Going implicit?

- **Why you *would*, if you *could***
  - **for multi-scale problems with wide scale separation, where interest is in following the slower scales**
  - **for coupled (or "multiphysics") problems**
  - **for problems with controllable or uncertain parameters or fields (or "outer loop" problems): optimization, design, control, inversion, assimilation**
  - **for cranking up temporal order with BDF (at least to 2)**
- **You *can*, so you *should***
  - **optimal and scalable algorithms known**
  - **freely available software available**
  - **reasonable learning curve that harvests legacy code**

SC07

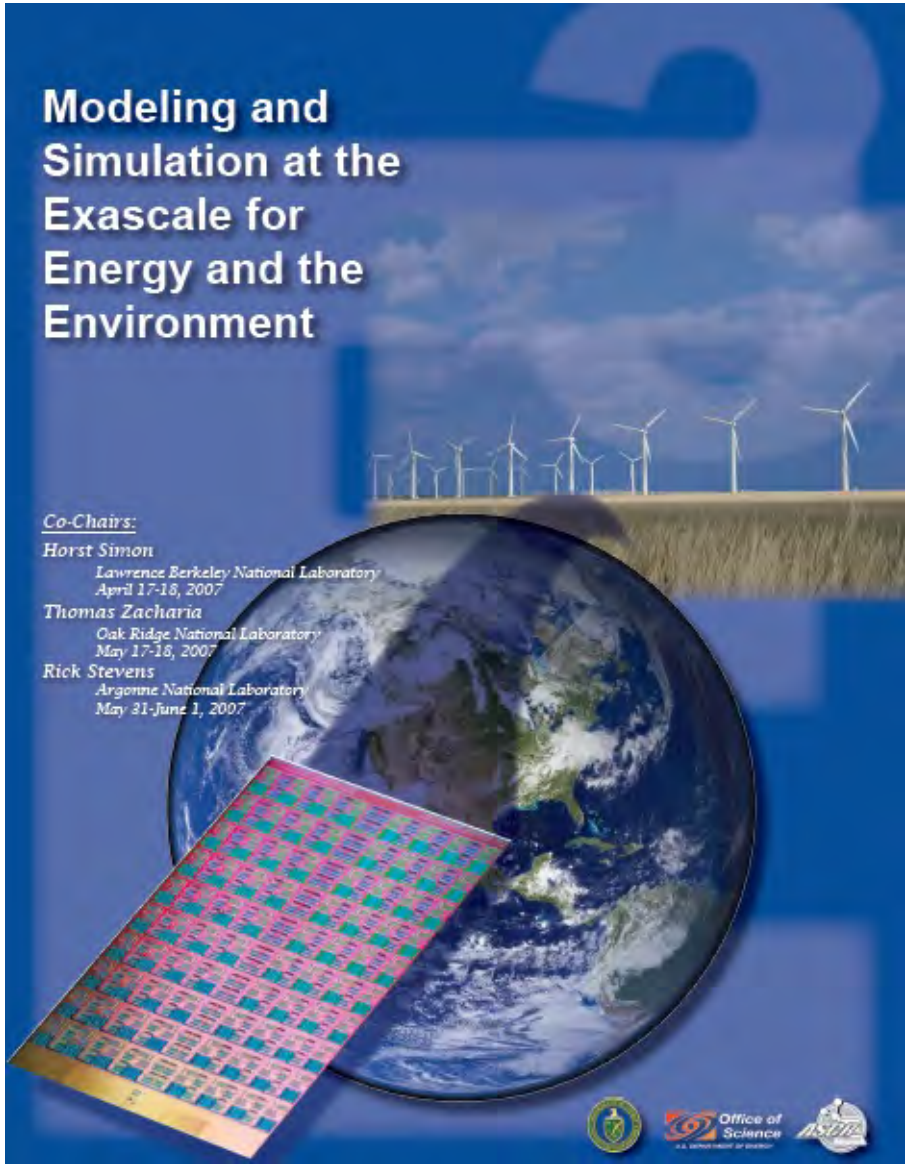
Focus on *Jacobian-free* implicit methods

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



SC07

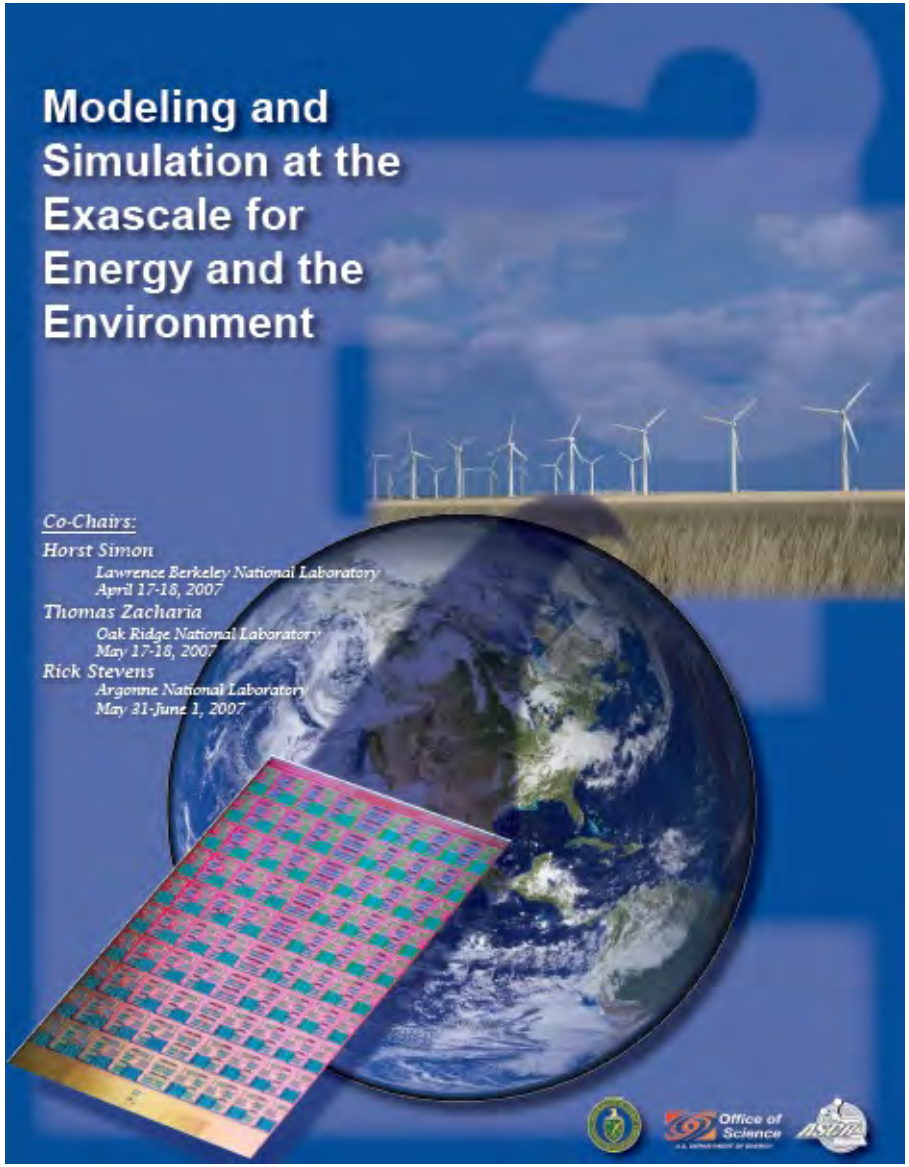
DOE "E^3" report (2007) highlights weaknesses of explicit methods



"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."

SC07

DOE E^3 report highlights opportunities for implicit methods



"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 increasingly include 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."

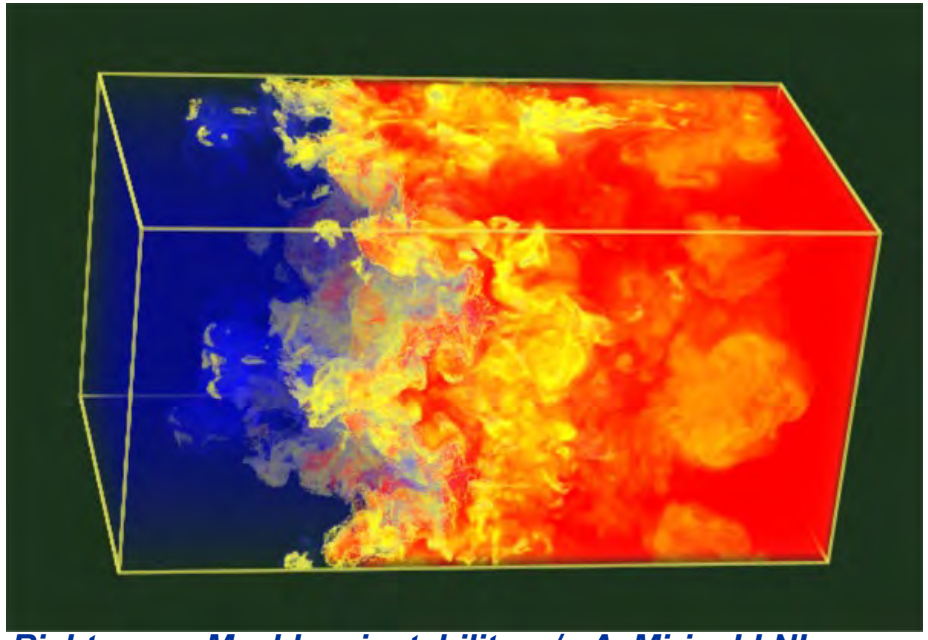
SC07

Some reports predicated on scalable implicit solvers …

among many others from DOE, NSF, NASA, NRC, etc.

Motivation #1
Many simulation opportunities are multi-scale

- **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, $\tau << T$**

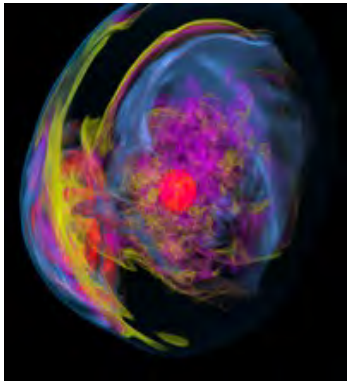


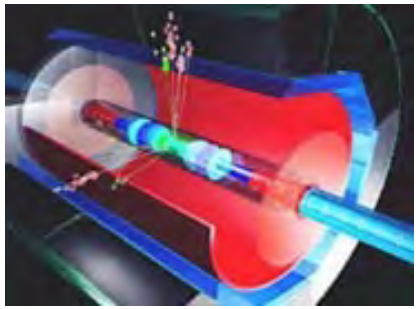
Richtmeyer-Meshkov instability, c/o A. Mirin, LLNL

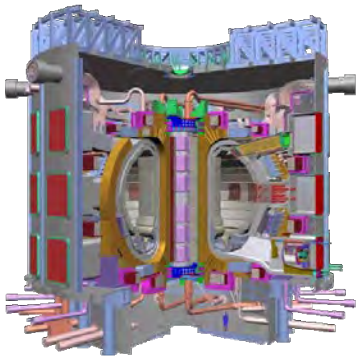
- **Analyst must *isolate dynamics of interest* and *model the rest* in a system that can be discretized over more modest range of scales**
- **Often involves filtering out of high frequency modes, quasi-equilibrium assumptions, etc.**
- **May lead to infinitely "stiff" subsystem requiring implicit treatment**

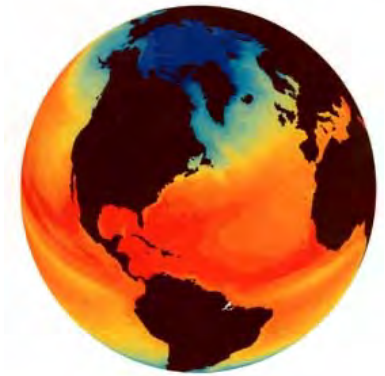
SC07

e.g., DOE's SciDAC* portfolio is multi-scale

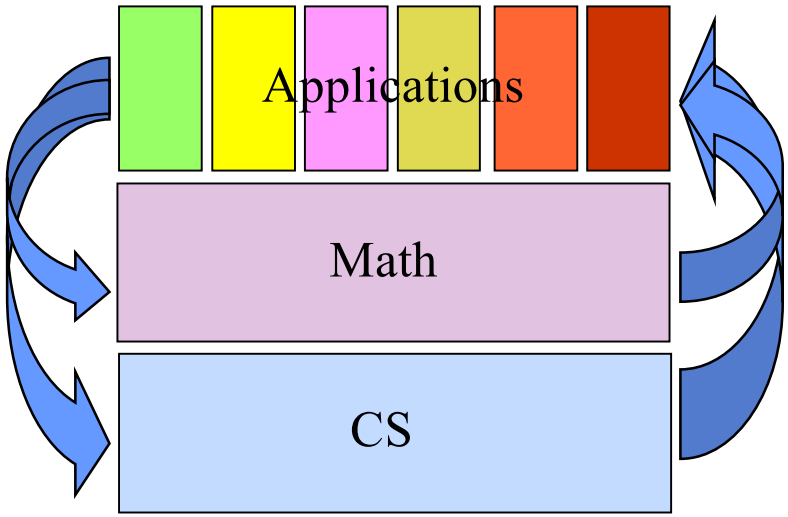








Many applications drive



Common technologies respond

* Scientific Discovery through Advanced Computing

SC07

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 mechanisms
- 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 requirements must still be satisfied; with long time steps, one can use high-order temporal integration schemes.

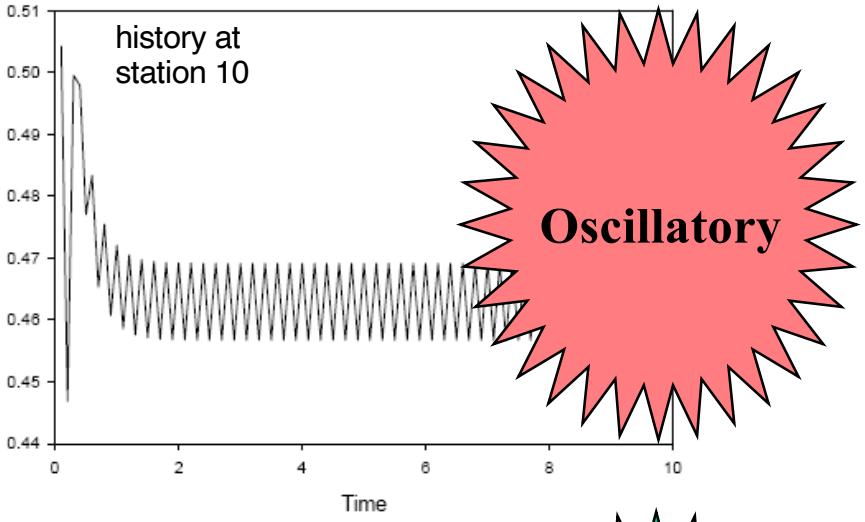
SC07

Methods that are not fully nonlinearly implicit may fail, even if stable

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

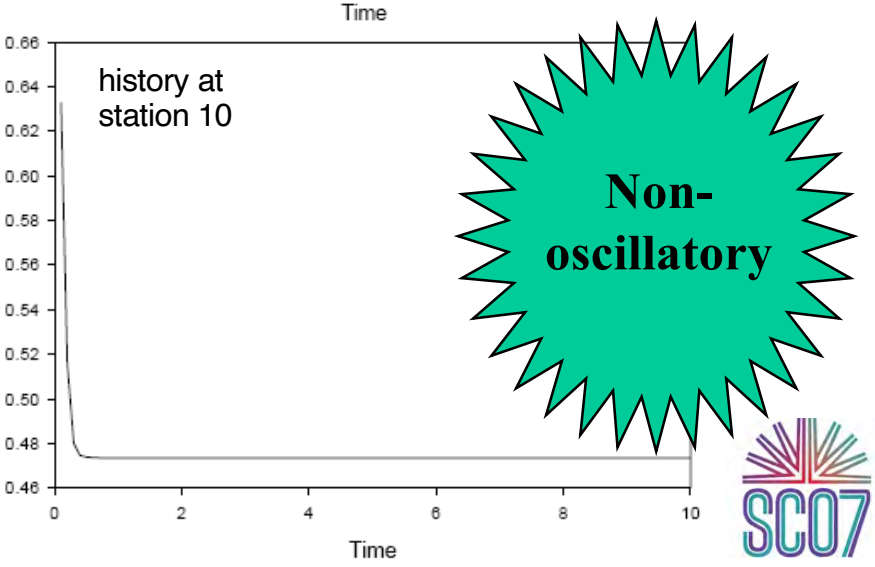
Linearly implicit, nonlinearly explicit:

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

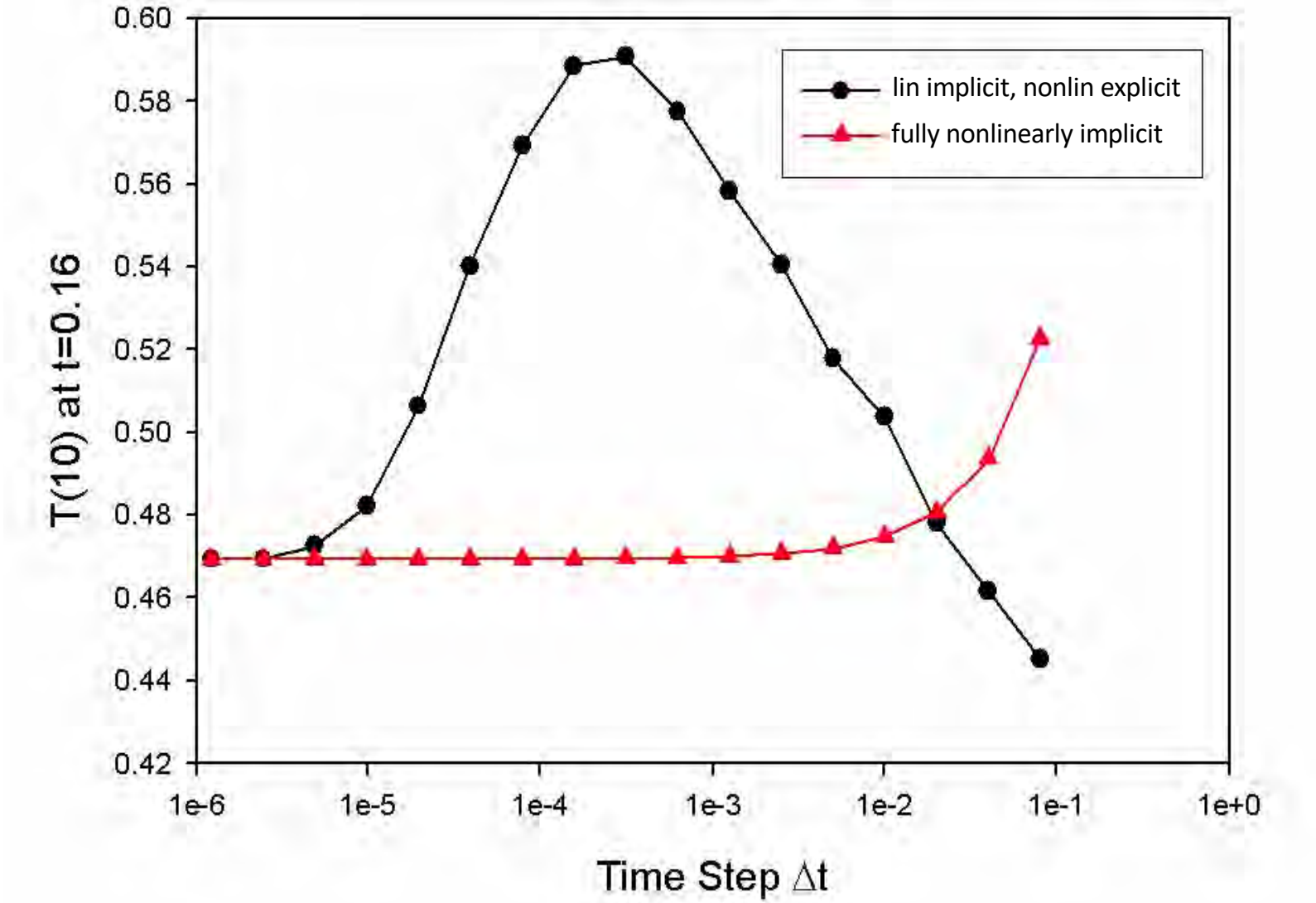


Linearly *and* nonlinearly implicit:

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



Accuracy as a function of timestep – GLF23 with gradient-dependent diffusivity

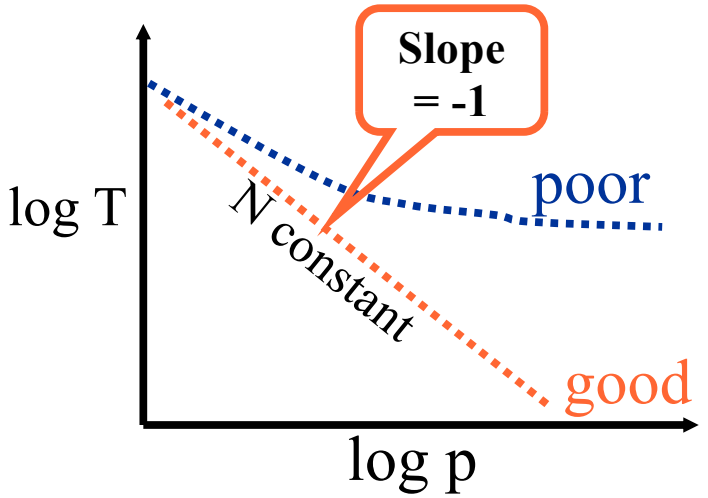


For sufficiently small timestep, the nonlinearly implicit and linearly implicit with lagged diffusivity converge on the same result, but the nonlinear implicit permits timesteps 10^4 times larger with same accuracy (c/o S. Jardin, PPPL).

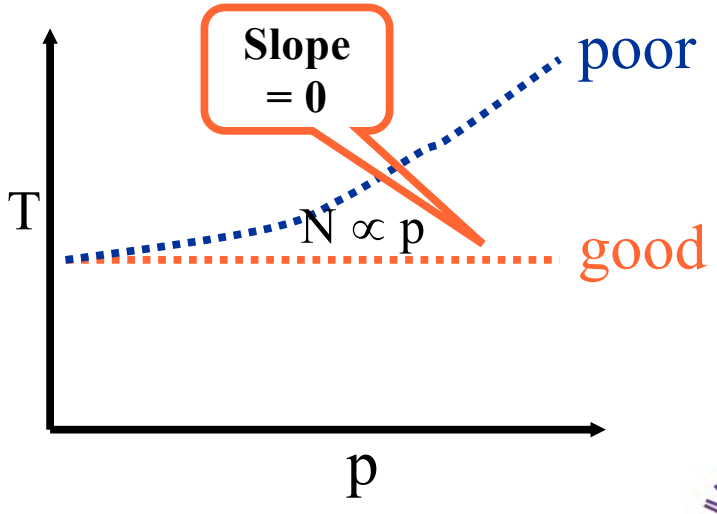
SCO7

Explicit methods do not weak scale; (review weak vs. strong scalability)

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



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



SC07

Runtimes of explicit methods grow under weak scaling due to CFL stability constraints

- **Illustrate for CFL-limited hyperbolic and parabolic eqns**
- **Parallel wall clock time**

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

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

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

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

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

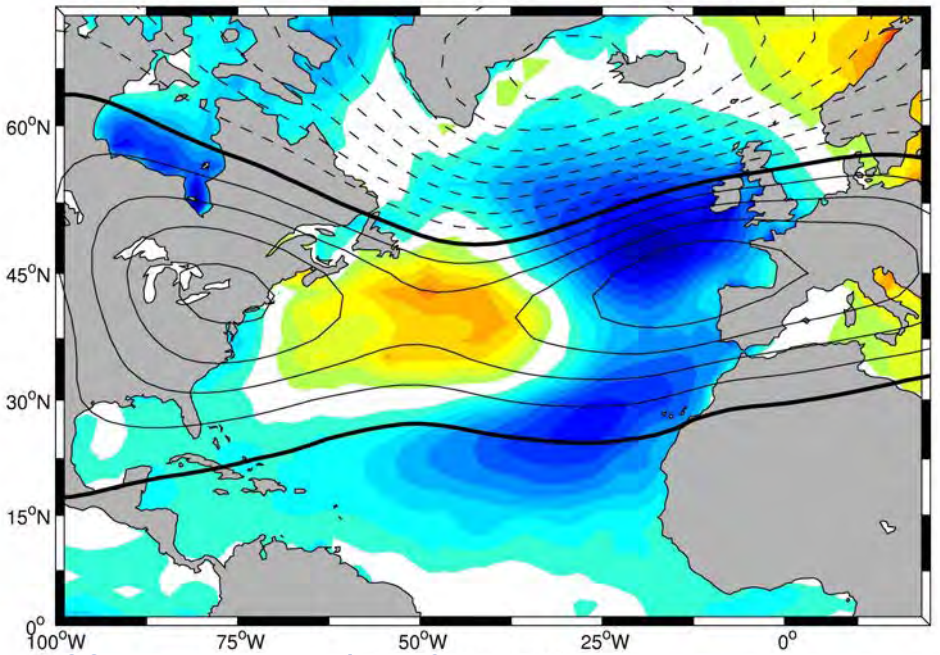
d-dimensional domain, length scale L
$d+1$-dimensional space-time, time scale T
h computational mesh cell size
τ computational time step size
$\tau=\mathcal{O}(h^\alpha)$ stability bound on time step
$n=L/h$ number of mesh cells in each dim
$N=n^d$ number of mesh cells overall
$M=T/\tau$ number of time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

"blackboard"

SC07

Motivation #2
Many simulation requirements are multi-physics

- **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**

SC07

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")** $\dot{u}_D = -\lambda u, \quad u_D(t_k) = u_R(t_{k+1}), \quad t_k < t \le t_{k+1}$
- **Overall advance** $U_{k+1} = u_D(t_{k+1})$

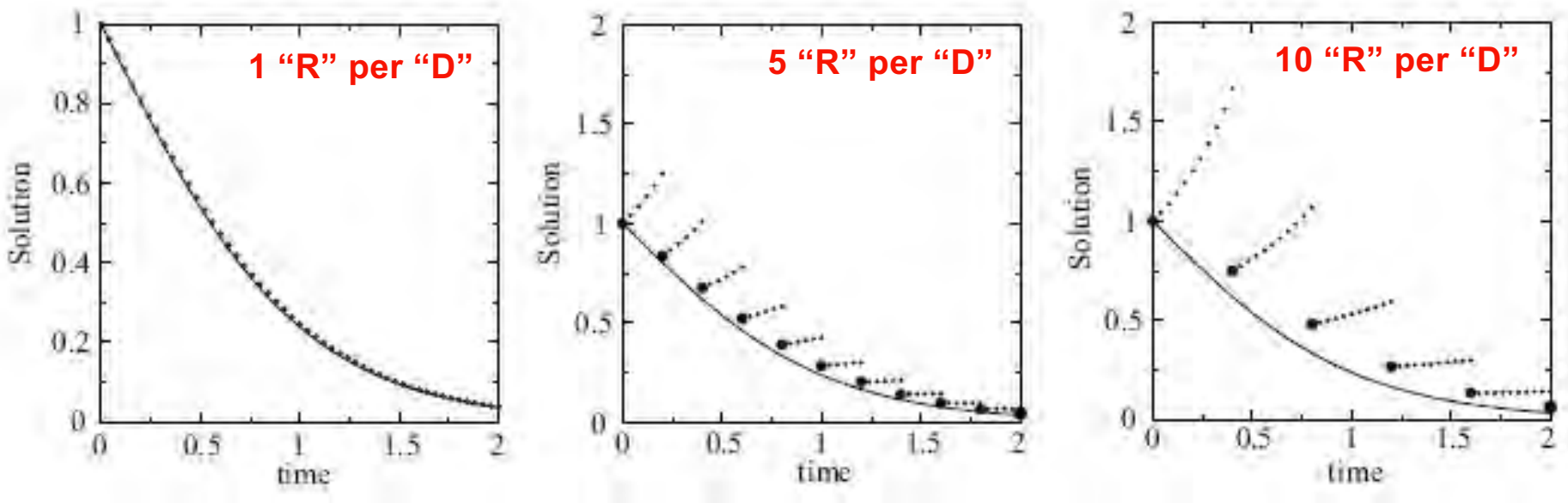
- **Phase 1 solution** $u_R(t) = \dfrac{U_k}{1 - U_k(t - t_k)}$
- **Phase 2 solution** $u_D(t) = u_R(t_{k+1}) \exp(-\lambda(t - t_k))$
- **Overall advance** $U_{k+1} = U_k \dfrac{\exp(-\lambda \Delta t)}{1 - U_k \Delta t}$

Can blow up in finite time!

SCO7

Operator splitting can destabilize multi-physics

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



Continuous

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

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

Discrete

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

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

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

This is a prototype for a Fourier mode of a reaction-diffusion PDE

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

- **Diffusive time-scale depends only on the wavenumber, whereas reactive time-scale changes with amplitude**
- **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 independently**
- **See *Estep et al.* (2007) for additional examples of splitting multiple components in one domain, multiple domains, etc.**

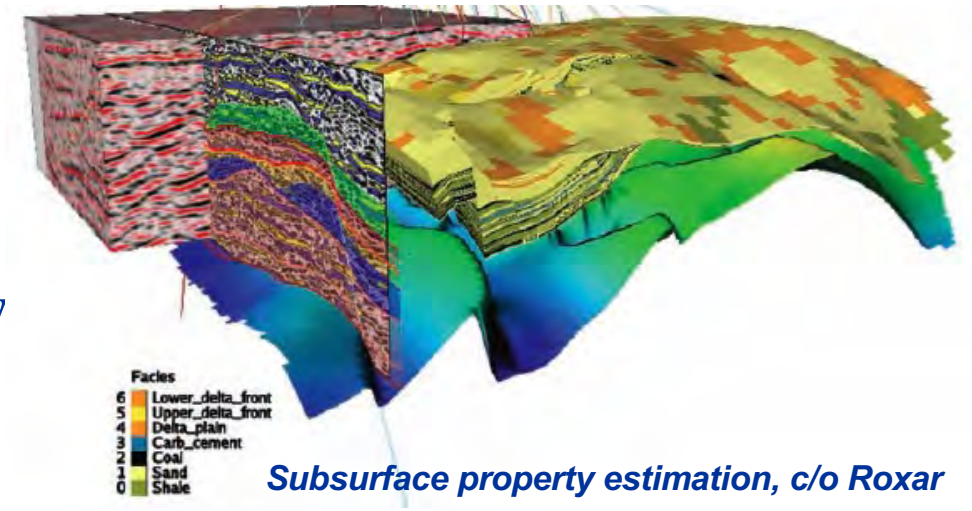
2022 note:

Giovanni Russo noted in Monday's panel that additive ($\underline{F}=\underline{F}_1+\underline{F}_2$) and component ($\underline{u}=(v,w)^T$) splitting of $\underline{u}'=\underline{F}(\underline{u})$ can be placed in correspondence.

SC07

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

- UQ, sensitivity, optimization, parameter estimation, boundary control all require the ability to apply the inverse action of the Jacobian – available in all Newton-like implicit methods – and can also use its adjoint

SC07

Multi-solve efficiently via adjoints

- **Forward operator equation** $Lu = f$
- **Desired functional of solution** $\ell(u) = \langle \ell, u \rangle$
- **Define adjoint operator** $L^* v = g$

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

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

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

$$\langle v, f \rangle$$

… reduces to a mere inner product for each forcing f

SC07

Significance for multi-solve problems

- 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
  - Rather than solving the forward problem for each input
- Part of the price to be paid in coding (ability to solve with linearized adjoint) is often included in the price paid to take the forward problem implicit
  - Caveat: a key shortcut for solving with L when L is a Jacobian, namely the matrix-free application of L (see later), is not directly available for L^*
  - Instead of one function evaluation per matvec, it takes a number of function evaluations corresponding to the number of significant singular values

SC07

Significance for inverse problems

- **Inverse problems are often formulated as PDE-constrained optimization problems**
  - objective function (mismatch of model output and "true" output)
  - equality constraints (PDE)
  - possible inequality constraints
- **Can be 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 (so-called KKT system)
- **Solving with Newton requires the Jacobian of gradient, which is the Hessian of Lagrangian**
  - Makes use of the Jacobian of PDE system and its adjoint

SC07

Comparing nonlinear explicit and implicit methods

	Explicit	Naïvely 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

Components of scalable solvers for PDEs

- **Subspace solvers**
  - elementary smoothers
  - incomplete factorizations
  - full direct factorizations
- **Global linear preconditioners**
  - Schwarz and Schur DD methods
  - multigrid
- **Linear accelerators**
  - Krylov methods
- **Nonlinear rootfinders**
  - Newton-like methods

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

opt. combins. of subspace solvers

mat-vec algs.

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 $Ax = b$

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

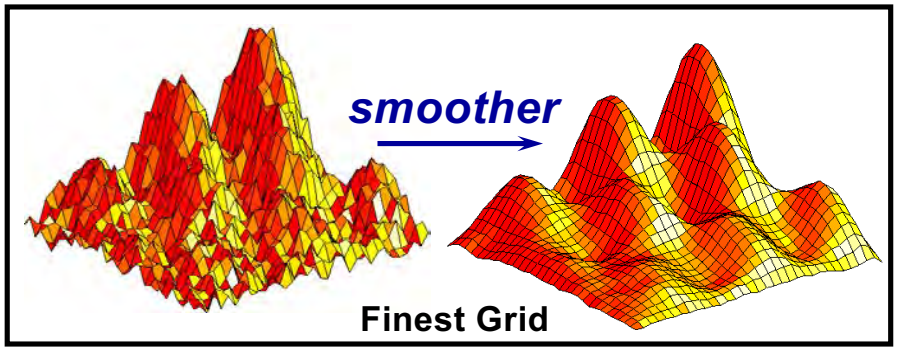
- Evaluate *residual accurately*, but *solve approximately*, where B^{-1} is an approximate inverse to A
- A sequence of complementary solves can be used, e.g., with B_1 first and then B_2 one has

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

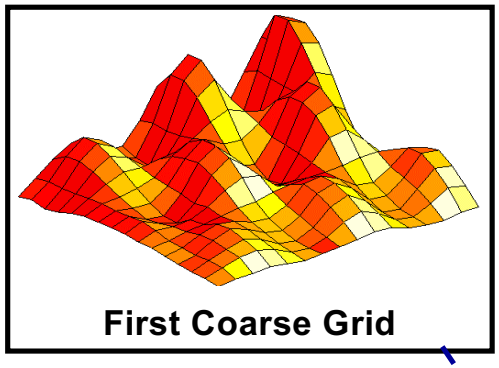
- Optimal polynomials of $(B^{-1}A)$ lead to various *preconditioned Krylov methods*
- Scale recurrence, e.g., with $B_2^{-1} = R^T (RAR^T)^{-1} R$, leads to *multilevel methods*

SC07

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



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

A Multigrid V-cycle

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 by automatic differentiation, so that the actual Jacobian elements are *never explicitly needed*

Carl Jacobi

- One still builds the Krylov space on a true $F'(u)$ (to within floating point limitations)

SC07

Jacobian-free 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:

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

Jacobian-free NK is not necessarily *matrix*-free

- 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

SC07

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
- This map defines a (usually weakly) converging nonlinear method

$$M : F(u^k) \mapsto \delta u$$
$$u^{k+1} \leftarrow u^k + \delta u$$

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

Nonlinear implicitness is usually easy to add to linear implicitness

- **Linear versus nonlinear problems**
  - **solving linear systems often constitutes 90% of the running time of a large PDE simulation**
  - **the nonlinearity is often a fairly straightforward outer loop, in that it introduces *no new types of messages or synchronizations* not already present in Krylov-Schwarz, and has overall many fewer synchronizations than the preconditioned Krylov method or other linear solver inside it**
- **We can wrap Jacobian-free Newton based on the true residual outside**
  - **as a clean up of first-order splitting errors or errors from other convenient approximations**

Example of physics-based preconditioning: 1D shallow water system (Knoll-K, 2004)

continuity

$$\frac{\partial h}{\partial t} + \frac{\partial uh}{\partial x} = 0$$

momentum

$$\frac{\partial uh}{\partial t} + \frac{\partial u^2 h}{\partial x} = -gh\frac{\partial h}{\partial x}$$

Underlined terms cause fast gravity surface wave with speed $(gh)^{1/2}$

Semi-implicit discretization to step over the wave stability limit is

$$\frac{h^{n+1} - h^n}{\Delta t} + \frac{\partial (uh)^{n+1}}{\partial x} = 0 \qquad \frac{(uh)^{n+1} - (uh)^n}{\Delta t} + \frac{\partial (u^2 h)^n}{\partial x} + gh^n \frac{\partial h^{n+1}}{\partial x} = 0$$

Solve discretized momentum equation for time-advanced flux:

$$(uh)^{n+1} = -\Delta t g h^n \frac{\partial h^{n+1}}{\partial x} + S^n \qquad \left(S^n = (uh)^n - \Delta t \frac{\partial (u^2 h)^n}{\partial x} \right)$$

Substitute into discretized continuity equation:

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

Example of physics-based preconditioning: 1D shallow water system, cont.

Solving this linear parabolic equation for h^{n+1} across the domain and substitute into the equation for $(uh)^{n+1}$ pointwise

Voila! **A single scalar parabolic solve is sufficient to update both fields, but leaves a linearization error**

To turn this into a preconditioner for a fully implicit treatment, we need a map from residuals to updates:

continuity residual

$$\frac{\delta h}{\Delta t} + \frac{\partial \delta uh}{\partial x} = -\text{res}_h$$

momentum residual

$$\frac{\delta uh}{\Delta t} + gh^n \frac{\partial \delta h}{\partial x} = -\text{res}_{uh}$$

parabolic solve for δh

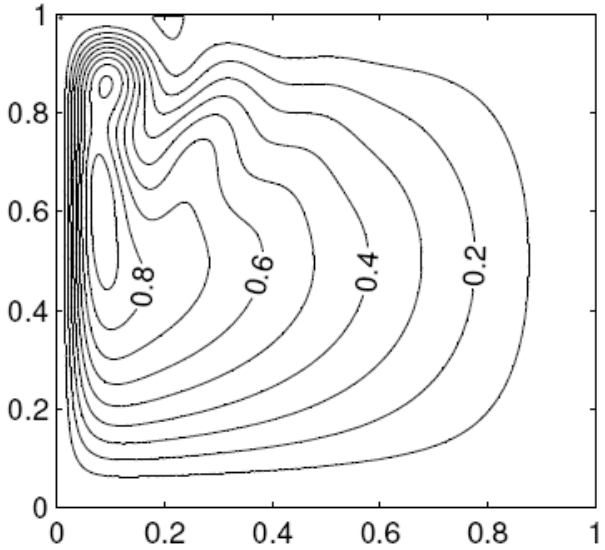
$$\frac{\delta h}{\Delta t} + \frac{\partial}{\partial x}\left(\Delta t g h^n \frac{\partial \delta h}{\partial x}\right) = -\text{res}_h + \frac{\partial}{\partial x}(\Delta t \text{res}_{uh})$$

Pointwise update for δuh

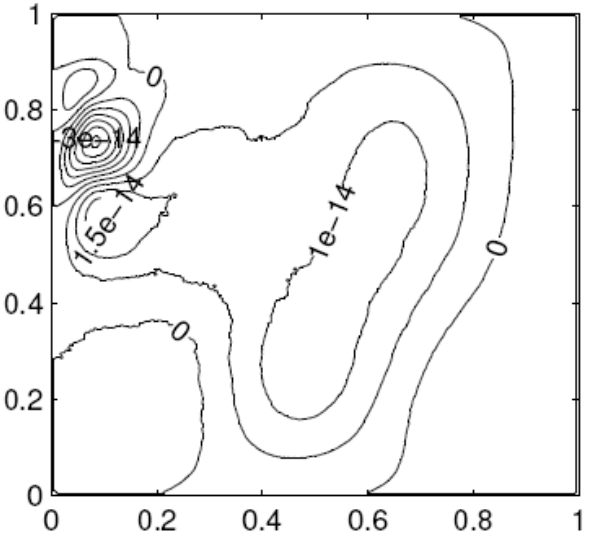
$$\delta uh = -\Delta t g h^n \frac{\partial \delta h}{\partial x} - \text{res}_{uh}$$

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

- 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!

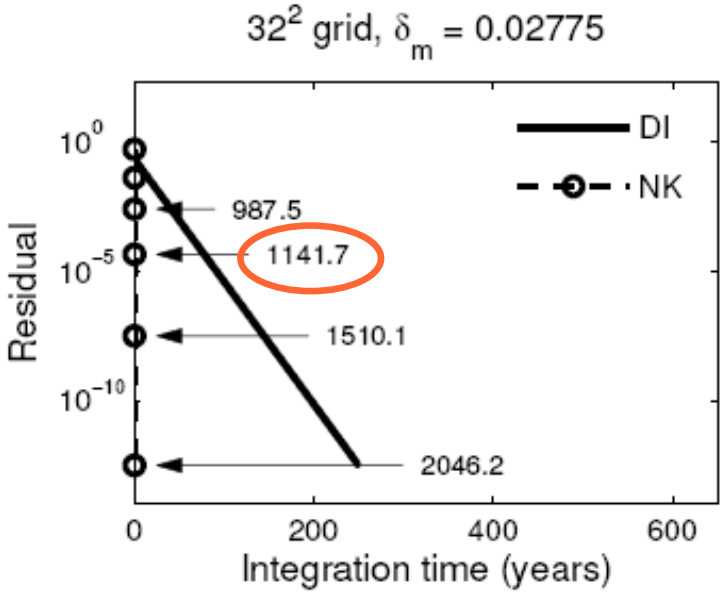


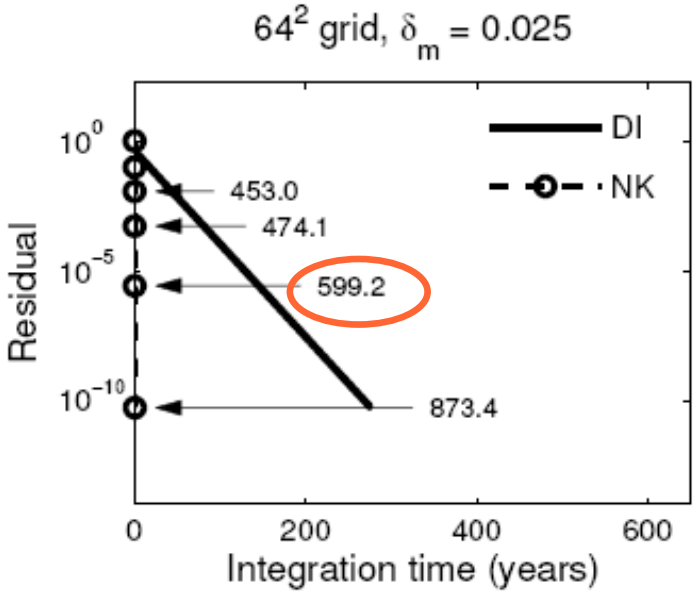
converged streamfunction

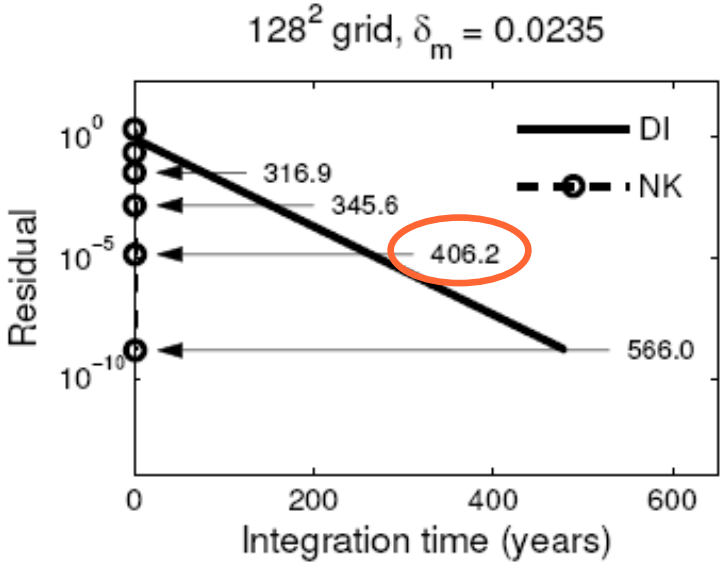


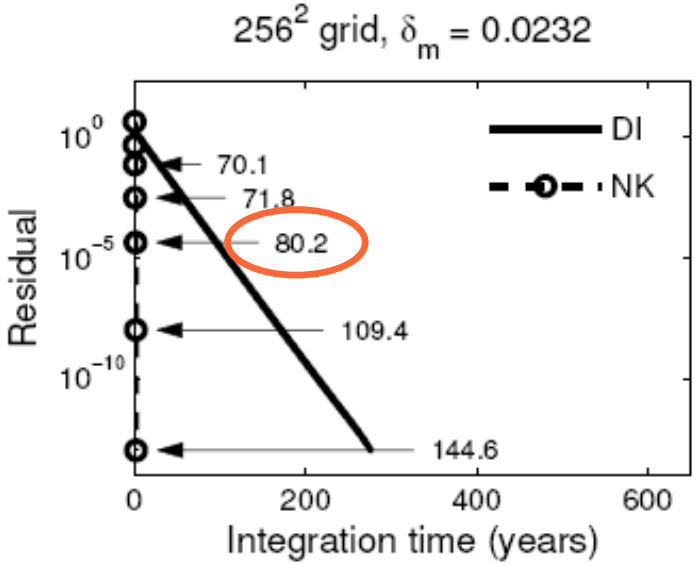
difference between DI and NK (10^{-14})

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 an undergraduate research project

Back to 2022

What has changed in time integration?

- Multi-partitioned RK
- Multirate/infinitesimal RK
- Partitioned Rosenbrock
- Multirate/infinitesimal Rosenbrock
- Splitting schemes
- Acceleration via surrogates
- Symplectic, split Hamiltonians
- …

(from Sandu, 2022)

(from Constantinescu, 2022)

- **Many tasty new apples on the tree and delicious new entrees at the picnic!**
- **A "Cambrian explosion" of operator splitting methods that exploit natural separations on the 2x2 state space of { stiff | nonstiff } and { linear | nonlinear } terms in order to achieve solutions of desired temporal accuracy at lower cost relative to fully implicit solvers**

Is monolithic nonlinear implicitness obsolete?

- **From an idealistic numerical analyst's perspective, perhaps**
  - Other methods with the same or lower asymptotic cost can achieve higher order error by exploiting equation structure
  - In a variety of systems amenable to rigorously quantifying errors, cost-benefit analysis favors more surgical approaches
  - Opportunities from new mathematics should influence future software development starting today
- **From a practical perspective, no**
  - Understanding of equation structure in complex dynamically evolving systems may be incomplete
  - Such systems, if set up to run for days on thousands of cores, can ill afford unanticipated losses of stability
  - Many workhorse codes currently suffering from first-order splitting error need to be retrofitted with monolithic nonlinear implicit solvers, as an "on-ramp" to and backstop for more sophisticated approaches

We don’t have an arms race, but a love fest

- **The new tasty apples still have some dependency on nonlinearly implicit solvers**
  - **Apart from constant coefficient splitting (e.g., to exploit optimal Poisson solvers) there are few reasons to restrict the implicit parts to be linear**
- **Implicit solvers have good customers in sophisticated operator-split, multi-stage integrators**
  - **As the sophisticated integrators are advancing, so are the components of the nonlinearly implicit solvers needed to support them**
  - **Some of these components, e.g., Krylov solvers, linear preconditioners, nonlinear preconditioners are immediately transferable**

To conclude … about nonlinear preconditioning from recent talk at IPAM ☺

http://helper.ipam.ucla.edu/publications/bdcrc2020/bdcrc2020_17523.pdf

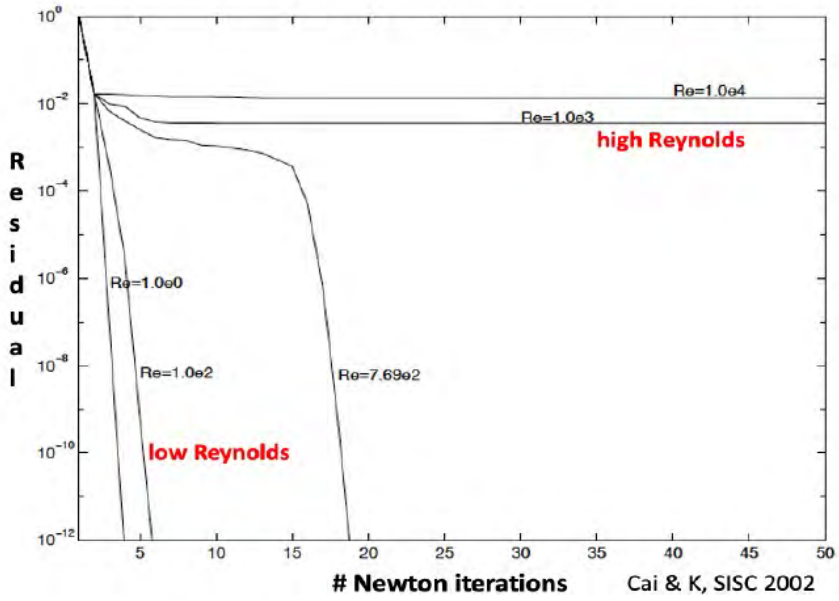
Why we're going there ...

- While the convergence rate of Newton's method is asymptotically independent of the spatial refinement of discretized PDEs, the embedded linear solves are not.
- In a fully implicit reservoir simulator, the linear solves on each time step typically consume 85% or more of the total execution time. Similarly for aerodynamics, combustion, etc.
- The linear solves also make up the most memory-demanding and least scalable phase of the code.
- One means of reducing their consumption is to do fewer *global* Newton solves. Nonlinear preconditioning replaces some global Newton solves by better linearly conditioned *local* Newton solves.
- "Multiphase flow introduces strong nonlinearities, and a naive implementation of Newton's method may fail to converge when large timesteps are taken... Smart strategies to address the intrinsic nonlinearities in these systems remain an active area of research." – Tchelepi et al., CMAME, 2019

Origin of "nonlinear stiffness"

Problem: when high-order terms dominate, the linear model is not a suitable approximation to $F(x)$.

- Strong nonlinearities result in a long plateau period of the residual $\|F(x_k)\|$.
- Only a small number of the n components of the solution may undergo significant updates in Newton corrections that are highly damped by linesearch backtracking or trust region globalization.



Newton methods may thus waste considerable resources solving global linear systems in problems that are "nonlinearly stiff" until they find the convergence domain. Such instances include material fronts, reaction fronts, shocks, recirculation bubbles, etc.

Example: nonlinear stiffness in reservoir problems

Possible sources:

- High contrast heterogeneous permeability or porosity
- Strong nonlinearity of relative permeability functions
- Spatially varying capillary pressure
- Faults, channels, and voids

These may force Newton to damp to death, or force a small timestep to robustify Newton's method.
This erodes the intrinsic advantages of a fully implicit method, namely:

- Choosing the timestep adaptively based on temporal truncation error accuracy requirements alone
- Going higher than first-order in time

Enter nonlinear preconditioning

- A nonlinear "preconditioner" performs nonlinear relaxation within one or more subsets of equations and unknowns, inside the context of an outer Newton method "accelerator."
  - Analogous to linear preconditioners, such as domain decomposition or multigrid, inside a Krylov accelerator
  - Such *linearly* preconditioned Krylov methods are often used inside *both* the nonlinear subproblems and the global problem
- A prime consideration in selecting an *inner* nonlinear preconditioner is whether the resulting *outer* nonlinear problem is amenable to linear preconditioning of the Jacobian.
- Early advocated left nonlinear preconditioners ASPIN and MSPIN complicate outer linear preconditioning by replacing a sparse outer Jacobian with a dense one.
- Right preconditioners like INB-NE retain the original Jacobian.
- Left preconditioner NEPIN (introduced herein) can also employ the original Jacobian.

Nonlinear preconditioning

- **Left preconditioning:** solve "equivalent system" (same root) with better balanced nonlinearities

$$\mathcal{F}(x) = G(F(x)) = 0$$

  - refs: Additive Schwarz Preconditioned Inexact Newton (**ASPIN**), Cai & K (2002); Multiplicative Schwarz Preconditioned Inexact Newton (**MSPIN**), Liu & K (2015); Restricted Additive Schwarz Preconditioned Exact Newton (**RASPEN**), Dolean *et al.* (2016); Nonlinear Elimination Preconditioning Inexact Newton (**NEPIN**), Liu *et al.* (2021)

- **Right preconditioning:** start from a better initial guess by first correcting within a subset of equations:

$$F(y) = 0, y = G(x)$$

  - refs: Nonlinear **FETI-DP** and **BDDC**, Klawonn *et al.* (2014); Nonlinear Elimination (**INB-NE**), Hwang *et al.* (2015) & Luo *et al.* (2020)

Preconditioning: Linear/Nonlinear, Left/Right

Linear preconditioning

$$F(x) = Ax - b = 0$$

Left:

$$G(F(x)) = M^{-1}(Ax - b) = 0$$

- $\kappa\left(M^{-1}A\right)$ smaller than $\kappa(A)$
- $F(x) = 0$ and $G(F(x)) = 0$ have same solution
- Linear system is changed

Right:

$$AM^{-1}y = b,$$
$$y = Mx.$$

- Linear system is unchanged

Nonlinear preconditioning

$$F(x) = 0$$

Left:

$$G(F(x)) = 0$$

- $G(F(x))$ less nonlinearly stiff than $F(x)$
- $F(x) = 0$ and $G(F(x)) = 0$ have same solution
- Nonlinear system is changed

Right:

$$F(y) = 0,$$
$$y = G(x).$$

- Nonlinear system is unchanged

Cautionary quotation

"All linear problems are alike;
each nonlinear problem is nonlinear in its own way."

– Keyes, with apologies to Tolstoy*

* see first line of *Anna Karenina* (1878)

Two-phase flows in porous media

Darcy's law and saturation equations for incompressible two-phase flow:

$$\begin{cases} \mathbf{u}_\alpha = -\lambda_\alpha \mathbf{K}(\nabla p_\alpha - \rho_\alpha g \nabla D), & \alpha = w, n, \\ \phi \dfrac{\partial s_\alpha}{\partial t} + \nabla \cdot \mathbf{u}_\alpha = q_\alpha, & \alpha = w, n, \\ s_w + s_n = 1. \end{cases} \tag{31}$$

Subscripts w, n represent the wetting (water) and non-wetting (oil) phases.

Unknowns are velocities $\mathbf{u}_\alpha$ and saturations s_α for phases $\alpha = w, n$.

p_α, ρ_α, q_α are, respectively, pressure, density, and external source or sink for phases $\alpha = w, n$.

Constitutive assumptions, next slide...

Constitutive assumptions

- $\mathbf{K}$ and ϕ are the absolute permeability tensor and porosity of the porous media (possibly discontinuous and varying by orders of magnitude).
- The mobility function λ_α is given by $\lambda_\alpha(s_w) = k_{r\alpha}(s_w)/\mu_\alpha$, for relative permeability $k_{r\alpha}(s_w)$ and viscosity μ_α.
- The capillary pressure function $p_c(s_w)$ is related to the pressure of both phases by $p_c(s_w) = p_n - p_w$.
- D is the depth at position (x, y, z) and g is the magnitude of the gravitational acceleration.

The relative permeability and capillary pressure functions are given as (Van Duijn 1998, Hoteit 2008):

$$k_{rw}(s_w) = s_e^\beta, \quad k_{rn}(s_w) = (1 - s_e)^\beta,$$

$$p_c(s_w) = -\frac{B_c}{\sqrt{K}}\log(s_e), \quad s_e = \frac{s_w - s_{rw}}{1 - s_{rw} - s_{rn}}.$$

Discretization

The problem is discretized by a fully implicit discontinuous Galerkin finite element method:

- Temporal discretization: backward Euler (first-order for now)
- Spatial discretization: Non-symmetric Interior Penalty Galerkin (NIPG) method (Epshteyn & Riviere, Appl. Numer. Math., 2007)

The fully implicit DG discretization results in a nonlinear algebraic system

$$F(x) = 0 \tag{32}$$

to be solved at each time step, where x is the vector of unknowns, which we may manipulate to be s_w and p_w. $F(x)$ is a highly nonlinear function, where the nonlinearities come from the relative permeability $k_{r\alpha}(s_w)$ and the capillary pressure function $p_c(s_w)$. Extra difficulties in solving (32) are induced by the heterogeneity of ϕ and K.

Software and Solution

LibMesh (**libmesh.github.io**) for the finite element construction

- piecewise quadratic for pressure
- piecewise linear for saturation

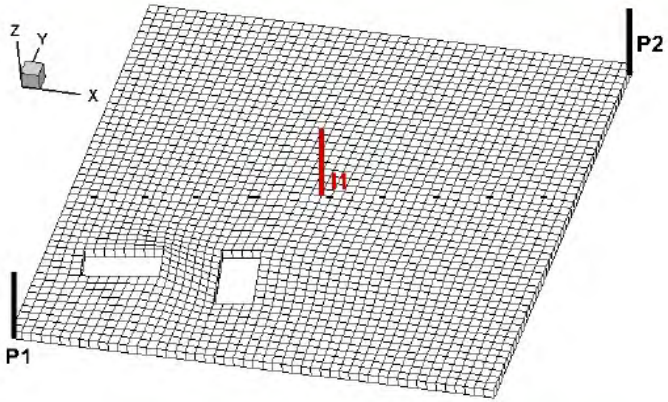
PETSc (**petsc.org**) for the algebraic solution

- relative tolerance for outer iteration global inexact Newton convergence on each timestep: 10^{-5}
- relative tolerance for inner inexact Newton convergence: 10^{-1}
- Jacobian and preconditioner evaluated only once per inner Newton iteration and reused
- GMRES restart dimension: 100
- overlap parameter for restricted additive Schwarz linear preconditioner: 1 cell
- subdomain preconditioner: ILU(2)

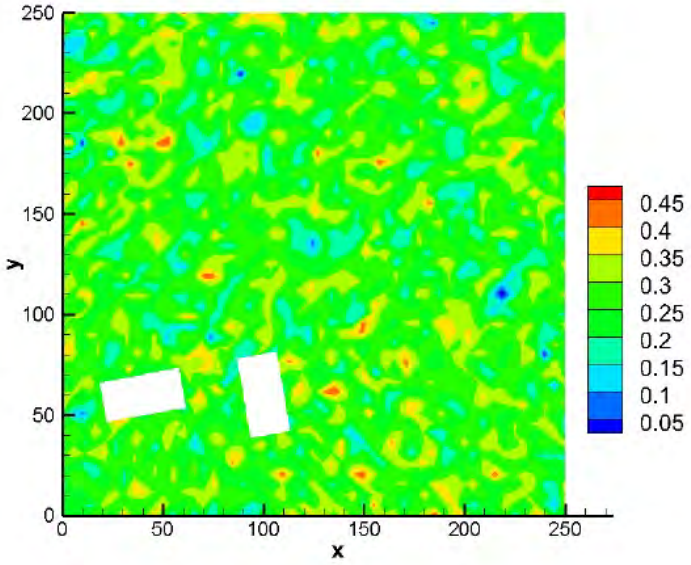
Shaheen-2 (**hpc.kaust.edu.sa**) for the scalable computation

- Cray XC-40 with dual socket Intel Haswell nodes and Aries dragonfly network
- ranked #7 in the Top 500 in 2015 (now #89)

Example: 2D domain with obstacles



Domain dimensions	250 m×250 m×5 m
Rock properties	homogeneous: $\phi = 0.2$, $K = 100$ mD
	heterogeneous: $\phi \in [0.01, 0.48]$, $K \in [0.00177, 718.69]$ mD
Fluid properties	μ_w (cP)/μ_n (cP) = 1/1, 1/2, 1/4
	$\rho_w = 1025$ kg/m^3, $\rho_n = 849$ kg/m^3
Relative permeabilities	$\beta = 2$ in (2.8)
Capillary pressure	$\bar{B}_c = B_c/\sqrt{K}$ in (2.8), $B_c = 0 - 18$ bar·mD$^{1/2}$
Residual saturations	$s_{rw} = 0$, $s_{rn} = 0$
Injection rate	43.2 m^3/day
Production well	$r_w = 0.1$ m, $s_k = 0$, $p_{bh} = 1$ bar [9]



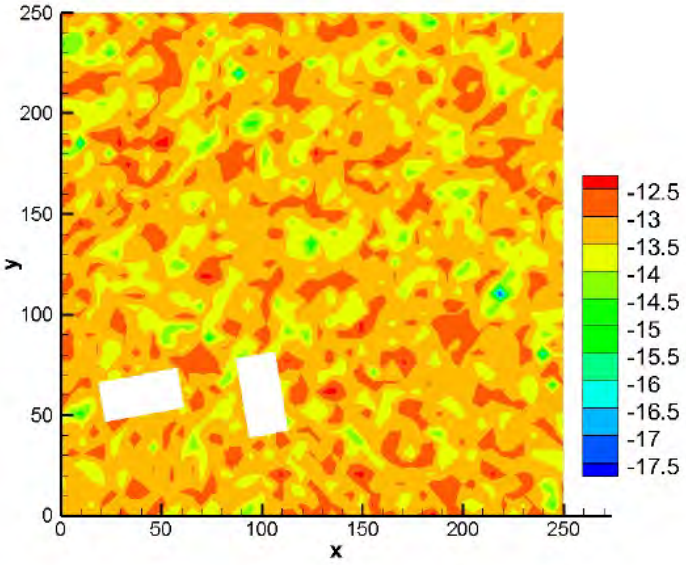
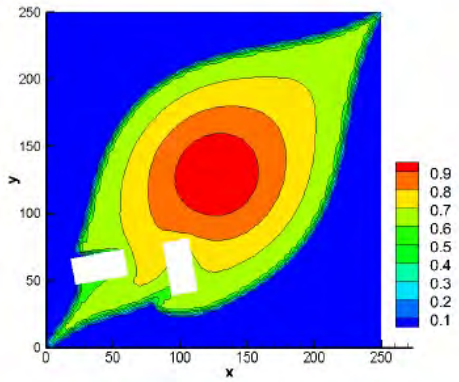
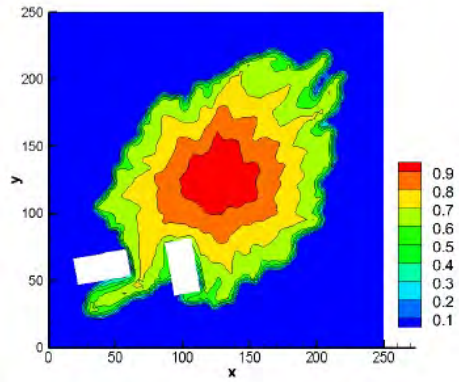


Figure: (top) The mesh, (bottom left) porosity, and (bottom right) permeability ($\log_{10} K$) for a square horizontal domain with obstacles.

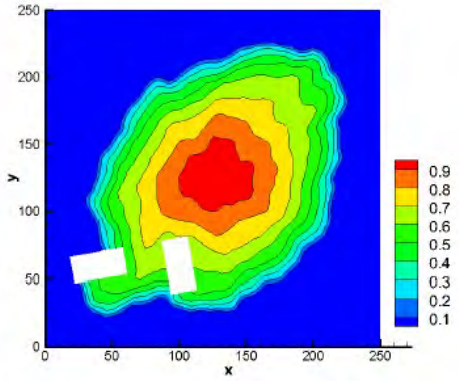
INB-NE solutions with various constitutive relationships



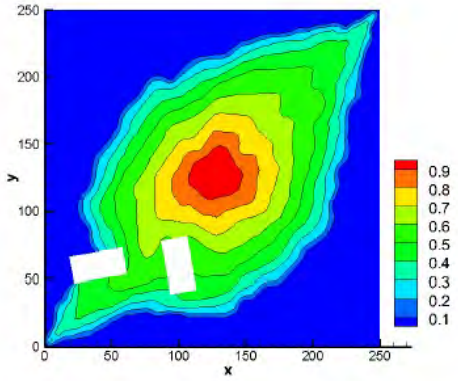
(a) Homo, $\mu_w/\mu_n = 1/2,\ B_c = 0$



(b) Hetero, $\mu_w/\mu_n = 1/2,\ B_c = 0$



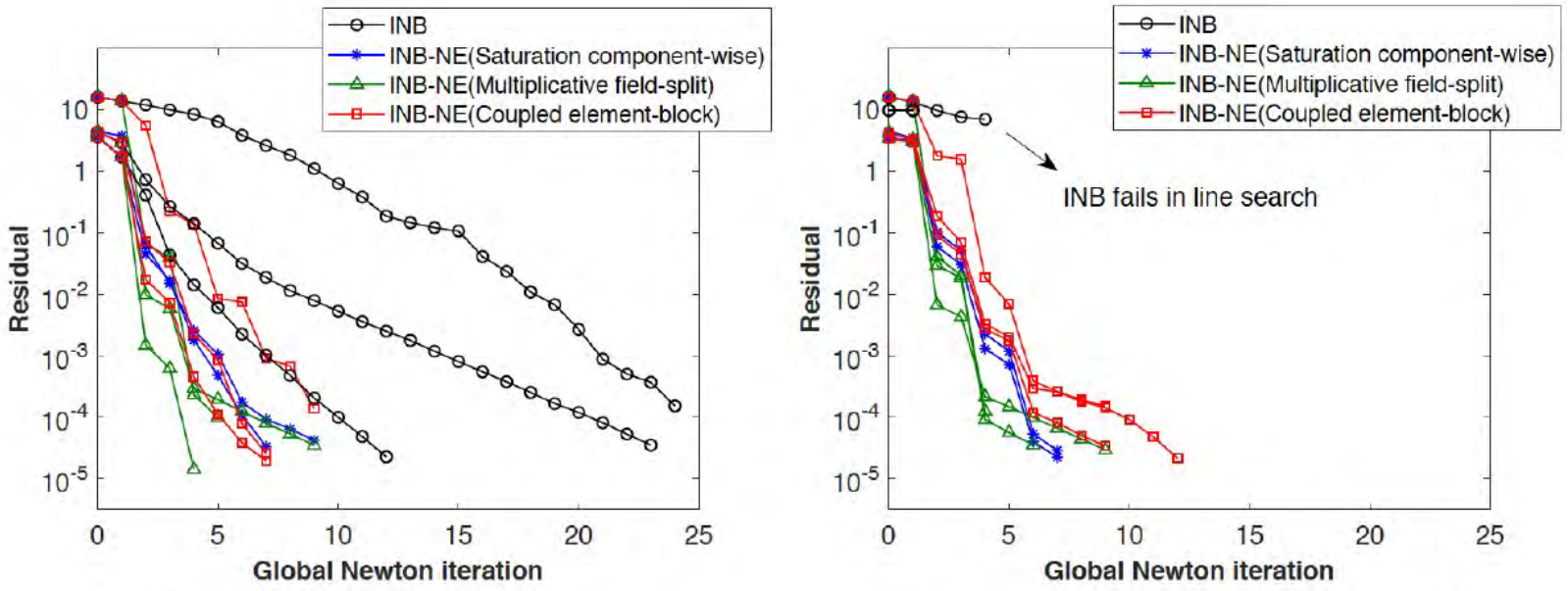
(c) Hetero, $\mu_w/\mu_n = 1/2,\ B_c = 8$



(d) Hetero, $\mu_w/\mu_n = 1/4,\ B_c = 10$

Figure: Wetting saturation at 500 days for oil displacement.

Residual history



(a) Case 2 ($\mu_w/\mu_n = 1/2$, $B_c = 0$)

(b) Case 3 ($\mu_w/\mu_n = 1/2$, $B_c = 8$)

Figure: Nonlinear residual history at the 1^{st}, 5^{th}, 10^{th} time steps, using INB and INB-NE for case $\mu_w/\mu_n = 1/2$. Left: $B_c = 0$. Right: $B_c = 8$.

Step length

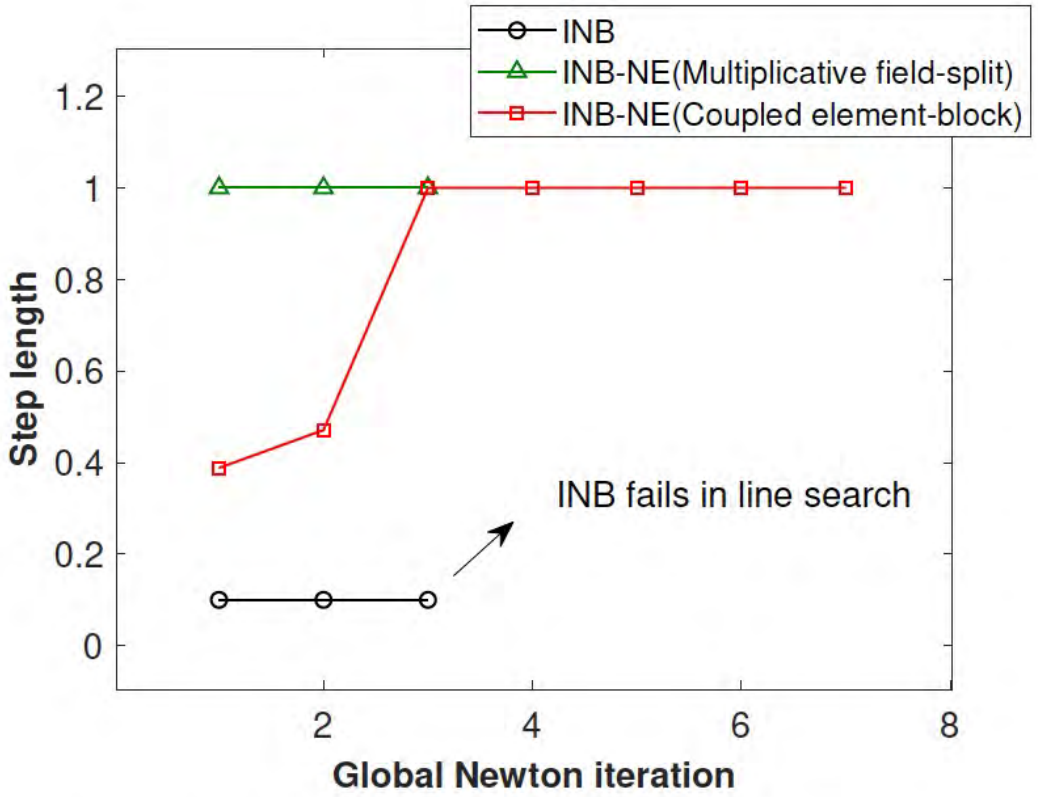
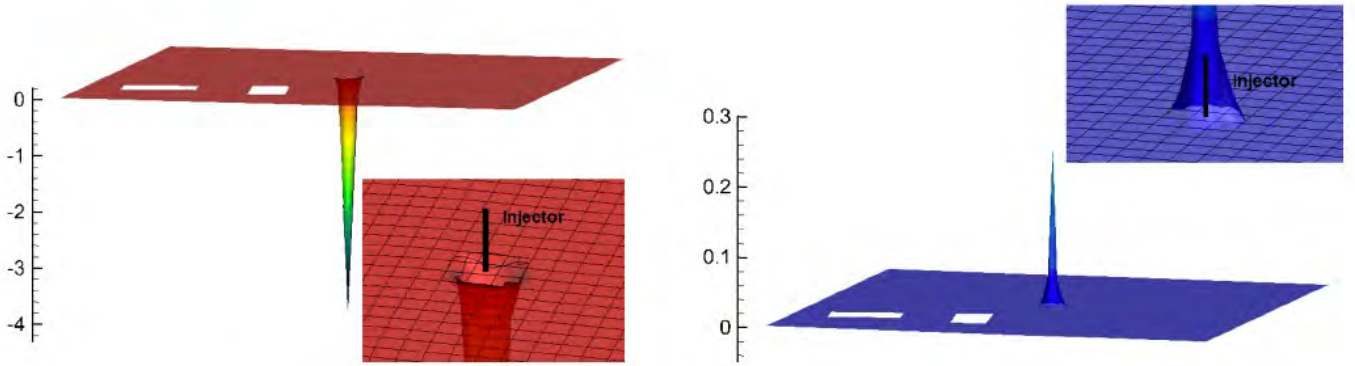
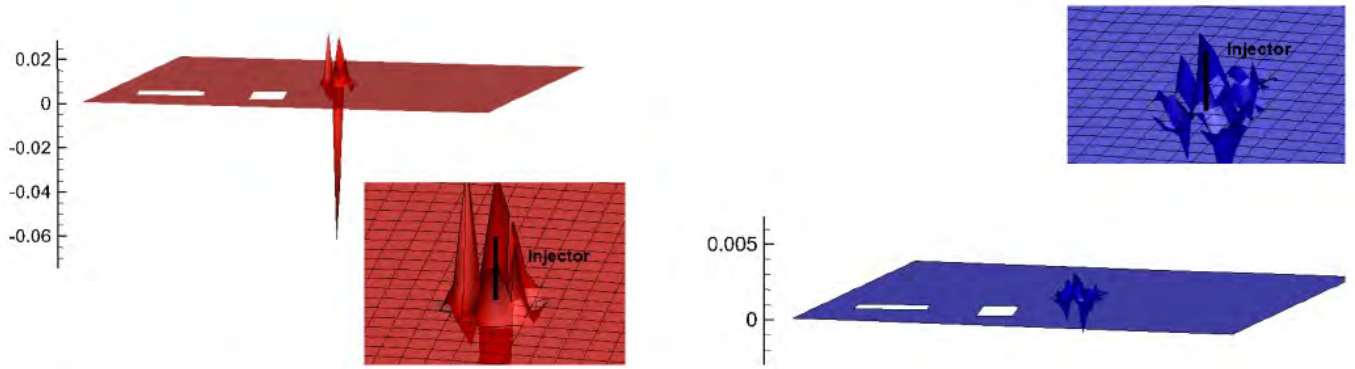


Figure: Step length λ^k at the 1^{st} time step, using INB and INB-NE for case $\mu_w/\mu_n = 1/2$, $B_c = 8$.

Before and after NE / coupled element-block



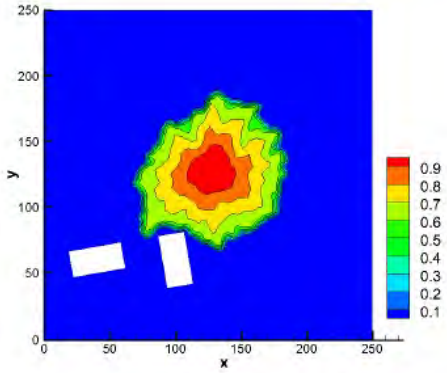
(a) before NE, 2^{nd} Newton, 1^{st} timestep



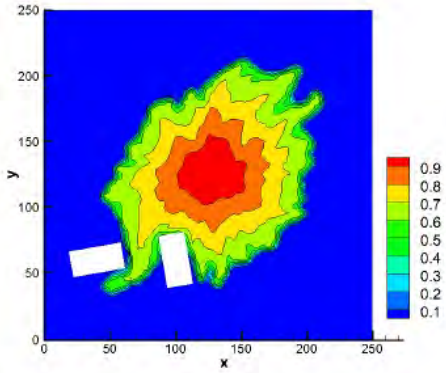
(b) after NE, 2^{nd} Newton, 1^{st} timestep – note scale change!

Figure: Left: saturation residual. Right: pressure residual.

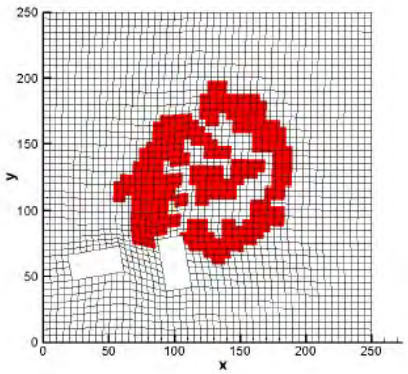
Element-block elimination



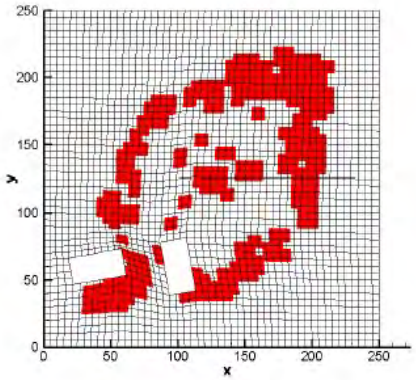
(a) Wetting saturation, 200^{th} step



(b) Wetting saturation, 400^{th} step



(c) Bad subset

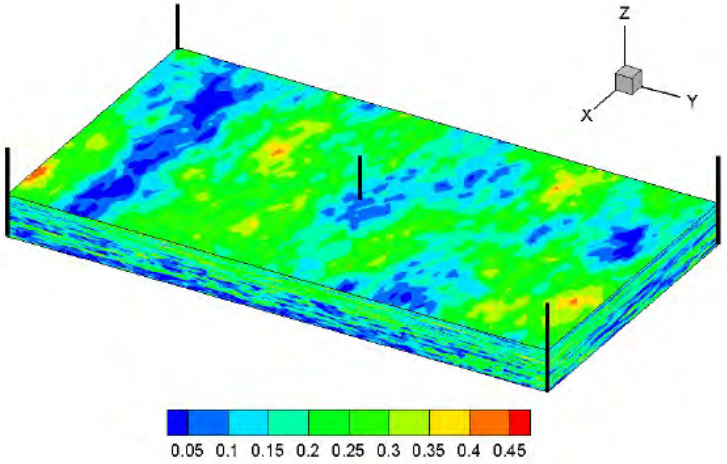


(d) Bad subset

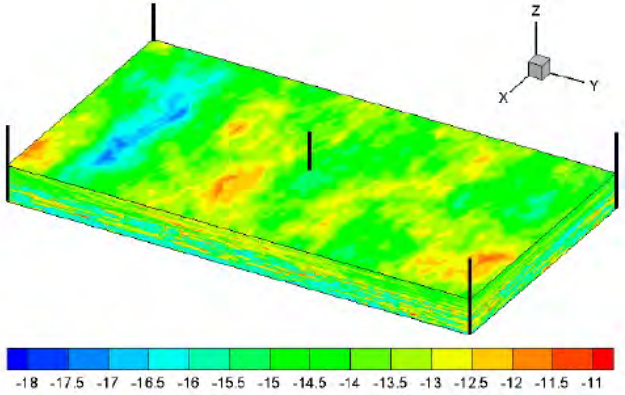
Figure: Results at 2^{nd} global Newton iteration at the 200^{th} time step (left) and 400^{th} time step (right) for case $\mu_w/\mu_n = 1/2$, $B_c = 0$.

Example: SPE10 setup

Domain dimensions	250 m×250 m×5 m
Rock properties	homogeneous: $\phi = 0.2$, $K = 100$ mD
	heterogeneous: $\phi \in [0.01, 0.48]$, $K \in [0.00177, 718.69]$ mD
Fluid properties	μ_w (cP)/μ_n (cP) = 1/1, 1/2, 1/4
	$\rho_w = 1025$ kg/m^3, $\rho_n = 849$ kg/m^3
Relative permeabilities	$\beta = 2$ in (2.8)
Capillary pressure	$\bar{B}_c = B_c/\sqrt{K}$ in (2.8), $B_c = 0 - 18$ bar·mD$^{1/2}$
Residual saturations	$s_{rw} = 0$, $s_{rn} = 0$
Injection rate	43.2 m^3/day
Production well	$r_w = 0.1$ m, $s_k = 0$, $p_{bh} = 1$ bar [9]

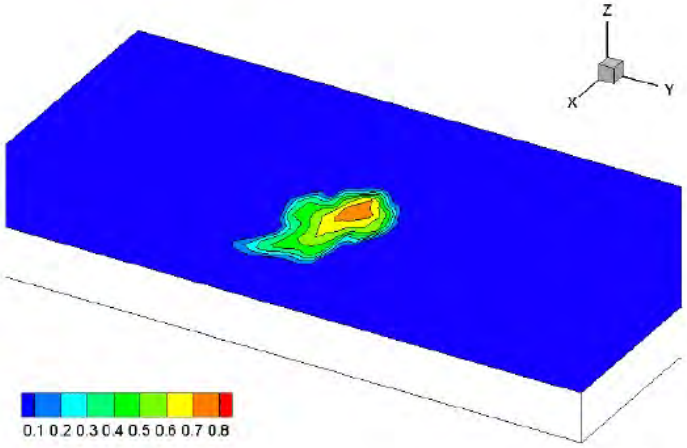


(a) Porosity

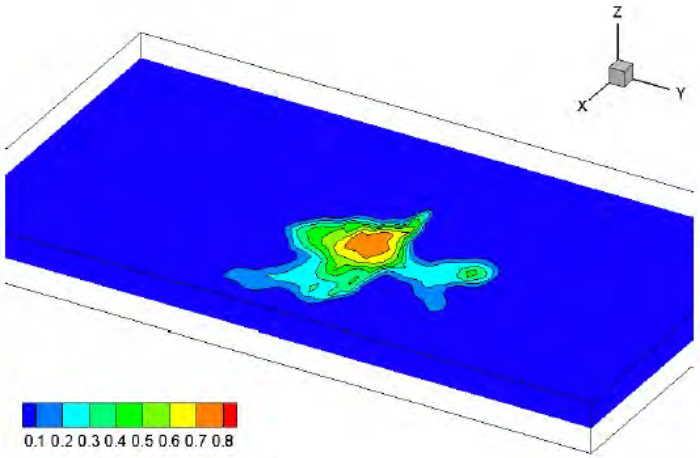


(b) Permeability ($\log_{10} K_{yy}$)

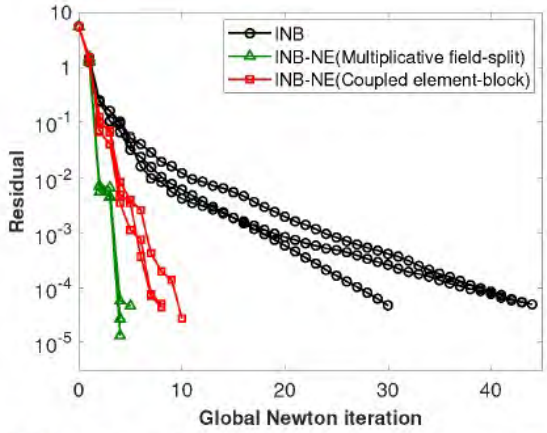
Example: SPE10 results



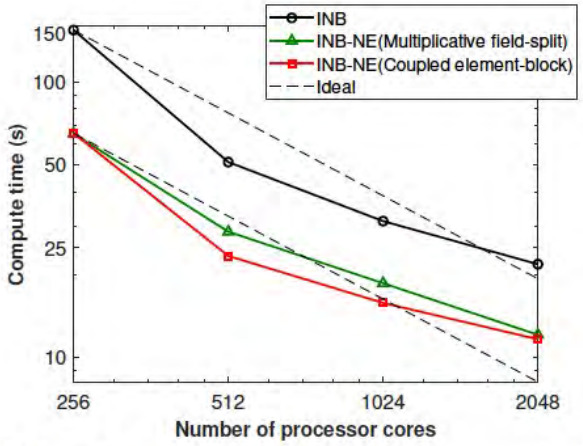
(c) Top layer sat at $t = 850$



(d) Middle layer sat at $t = 850$



(e) Residuals at the 6^{th}, 8^{th}, and 10^{th} time steps



(f) Strong scaling with 689,920 DOFs

Conclusions

- Nonlinear preconditioners can be effective in improving on the convergence of global Inexact Newton with Backtracking (INB) iterations.
  - They robustify Newton's method, often obviating other globalization methods, by going to the core of the difficulty – removing a few components of the correction that require severe damping.
- Nonlinear preconditioning expends extra local computational cost for the solution of nonlinear subproblems to reduce the computation, communication, and synchronization costs of the global outer iterations, by reducing their number.

Additional Benefits

- A simple adaptive framework allows switching nonlinear preconditioning on and off during the outer Newton iterations.
  - Nonlinear preconditioning is not free, and should be employed only when the global Newton iterate lies outside of the domain of quadratic convergence of Newton's method.
- *A posteriori* approximate error bounds on the linear functionals of interest are available using by-products of the nonlinear preconditioning split systems.
  - Many times, one does not need the full solution of a system of PDEs throughout a domain, but a linear functional of the solution, such as a value at a point, a subdomain or full domain average, or a flux integrated over a portion of the boundary.
  - These averaged or integrated quantities of interest may converge faster than the solution of the PDE itself, and a bound on their error can be computed as a by-product of residuals available during the iterations.

Future Work

- Better or more automated identification of "bad" components.
  - "Cascadic" ideas, wherein one starts with a small cluster of "bad" points and enlarges in a multistage nonlinear elimination process, have proved effective (Luo et al, SISC 2020).
  - Machine learning may be effective.
- Use of dynamic runtime systems to better allocate work in parallel implementations, in view of the imbalanced work of the inner nonlinear iterations.
  - Different preconditioning subproblems have different nonlinear difficulty.
  - Some outer iteration tasks (but not a synchronous full iteration!) can overlap with preconditioning subproblems.
- Fully asynchronous nonlinear iterations (within each time step for time-implicit problems or overall for steady-state problems).

We're committed to making full nonlinear implicitness a practical component in timestepping strategies for complex multiscale, multiphysics, multicomponent problems