New approaches to space-time splitting with higher accuracy

Hans Johansen (hjohansen@lbl.gov)
Applied Math and Computational Research Division
Lawrence Berkeley National Laboratory

Go20 Conference, 20th May 2025

Outline for my talk

TLDR:

Low-order splitting schemes can be combined with Integral Deferred Corrections, to create higher-order methods with more parallelism and faster solvers.

1. Background and motivation: AMR for (non-)linear hyperbolic problems using HPC
2. Operator splitting: good, bad, ugly and some lessons learned
3. Progress: Integral Deferred Corrections (IDC) on GPU's

Collaborative effort with:
Rochi Chowdhury, Nate Overton-Katz (LBNL)
Adrian Sandu, Alex Novotny (Virginia Tech)
Ben Ong (Michigan Tech)

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Base Math and SciDAC / FASTMath programs

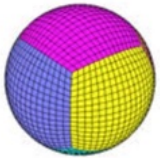
Part 1: Dynamic Adaptive Mesh Refinement (AMR)

Example “toy” AMR application:

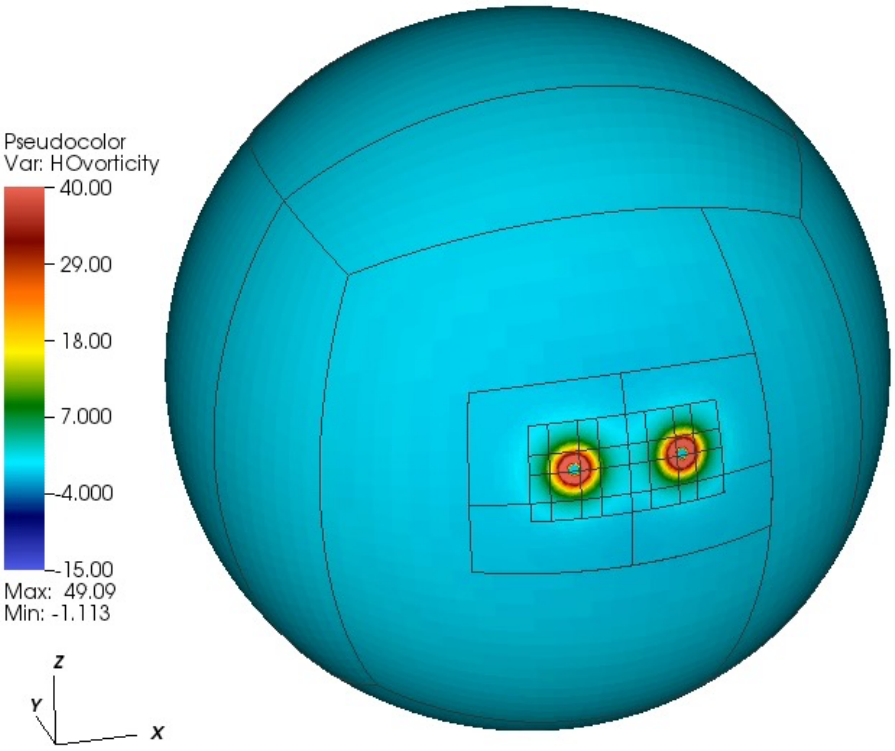
- Vortex dynamics in shallow water equations:

$$\frac{\partial h\mathbf{v}}{\partial t} + \nabla\cdot(h\mathbf{v}\otimes\mathbf{v}) + f\hat{\mathbf{k}}\times(h\mathbf{v}) + gh\nabla H = 0,$$

$$\frac{\partial h}{\partial t} + \nabla\cdot(h\mathbf{v}) = -\beta P,$$



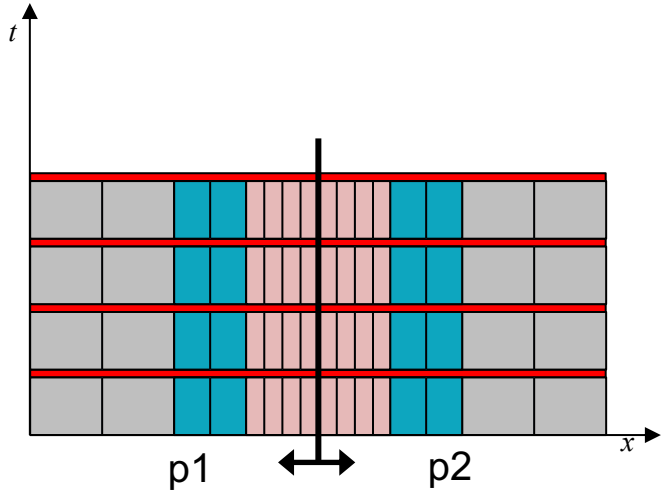
- FV on the “cubed” sphere
- Block structured AMR, with 3 levels, 4x refinement
- Explicit RK4 w/ “local time stepping”
- Refinement based on dynamic quantities:
  → magnitude, gradients, error estimates, etc.



AMR time stepping: 1D space + time example

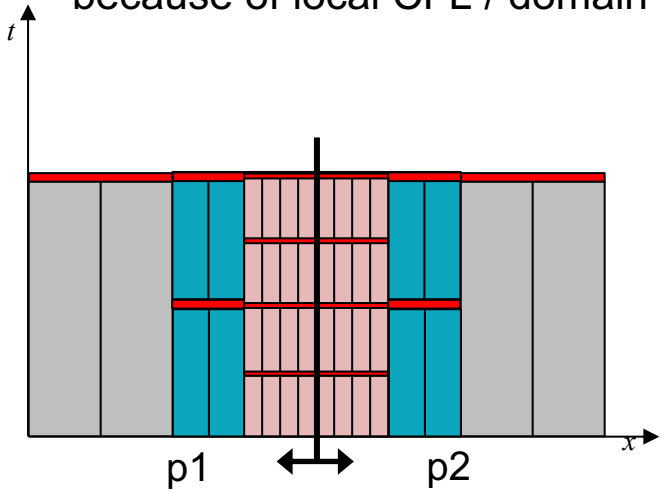
AMR *global* time step

- (Good) load balancing easy
- (Bad) coarse vs. fine CFL
- (?) Better for implicit/parabolic because of global coupling



AMR *local* time step

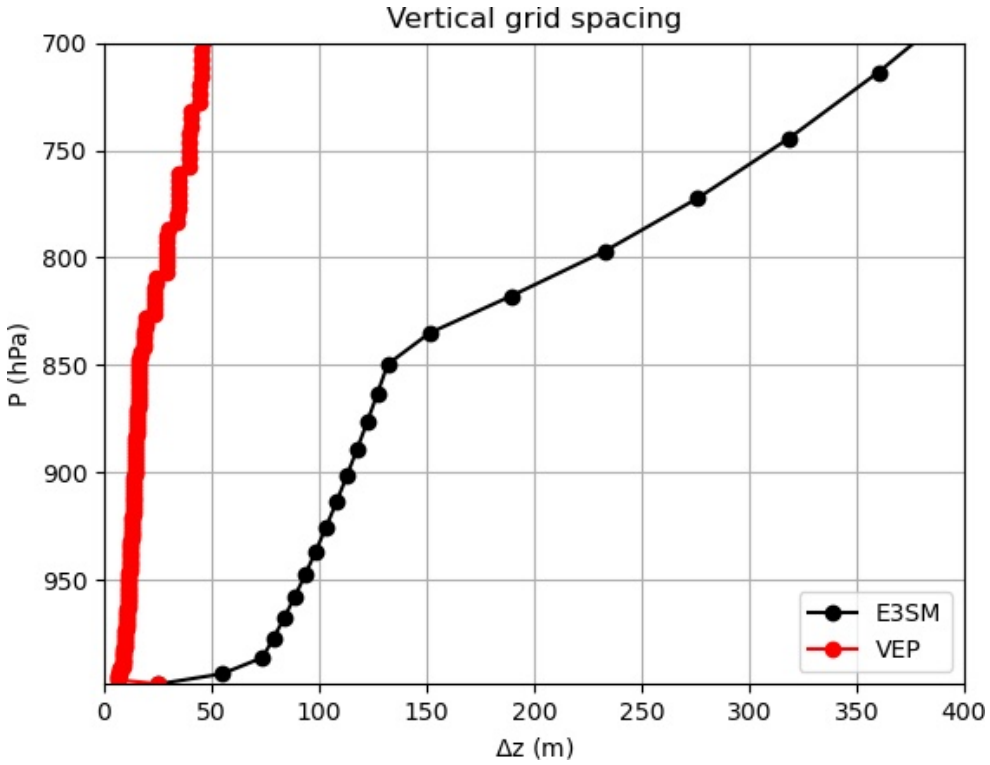
- (Good) local CFL, less work
- (Bad) coarse-fine synchronization, “coarse grid bottleneck” like MG
- (?) Better for explicit/hyperbolic because of local CFL / domain



What does 3D AMR w/ ImEx look like?

In 3D, horizontal explicit / vertical implicit "HEVI" but higher-order in space + time:

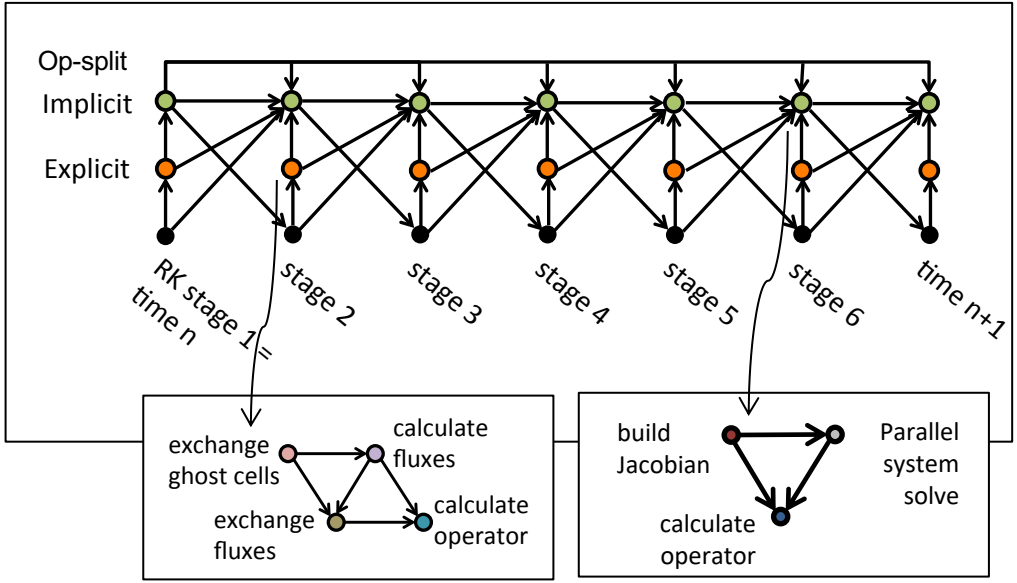
- Higher order (ARK4) requires many stages L-, A-stable additive problems
- Explicit nonlinear finite volume:
  - Transport w/ op-split source terms
- Implicit nonlinear "column" dynamics:
  - CFL ~ 4-100, ***stiff + hyperbolic***
- Also applies to "global time step" AMR



What does 3D AMR w/ ImEx look like?

In 3D, horizontal explicit / vertical implicit "HEVI" but higher-order in space + time:

- Higher order (ARK4) requires many stages L-, A-stable additive problems
- Explicit nonlinear finite volume:
  - Transport w/ op-split source terms
- Implicit nonlinear "column" dynamics:
  - CFL ~ 4-100, ***stiff + hyperbolic***
- Also applies to "global time step" AMR



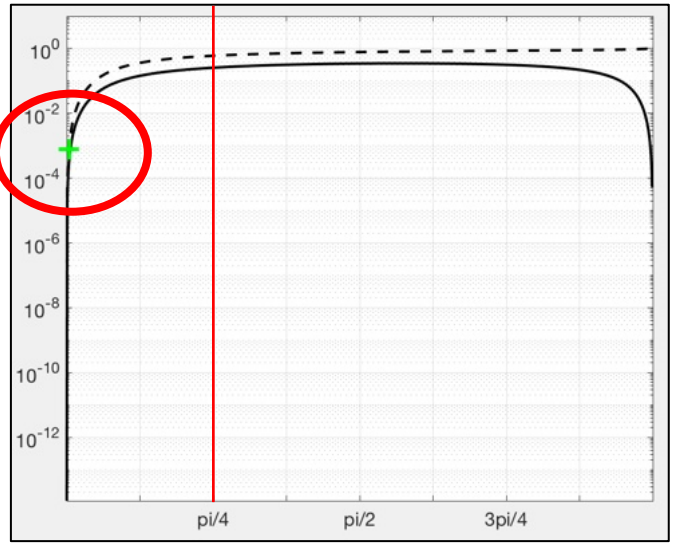
→ Lots of synchronization / communication for spatial parallelism!

Time Integrators: Implicit low-order bad for hyperbolic

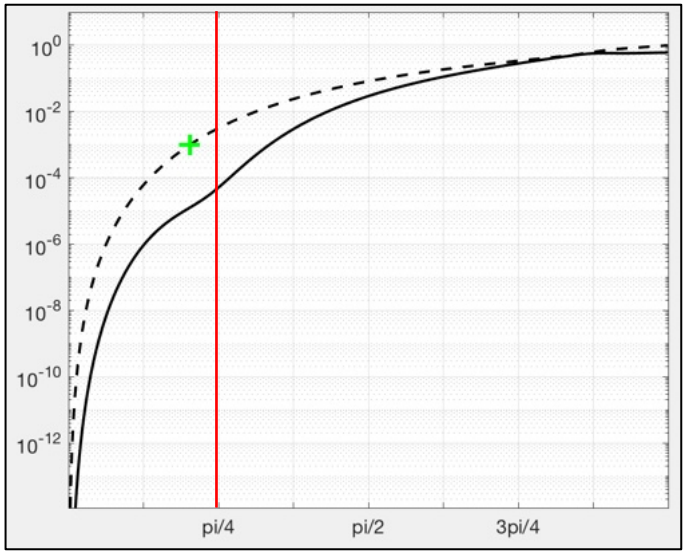
Advection at CFL=4, want 10^{-3} relative error for $8\Delta x$ modes or bigger?

4th-order centered + Implicit Euler

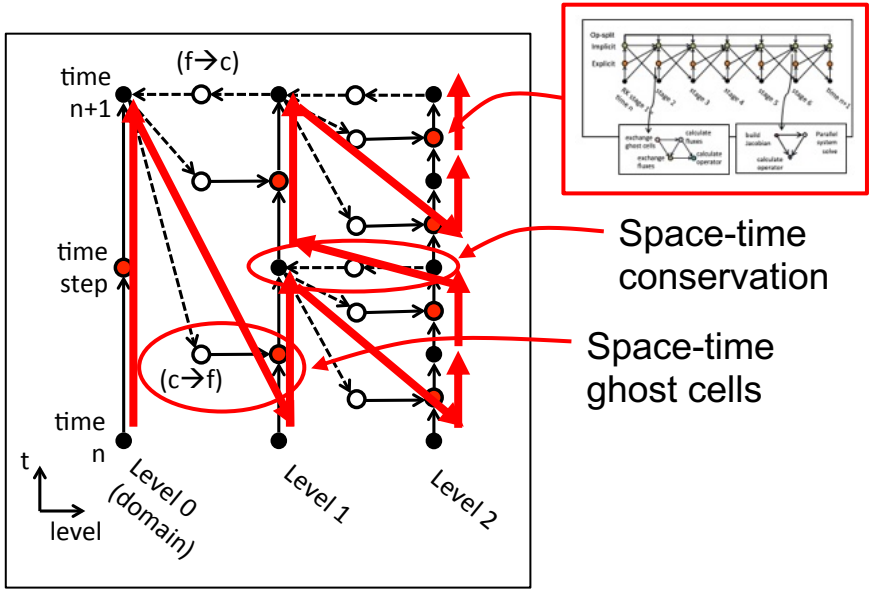
10^{-3} relative error



7th-order upwind + Implicit Hermite integrator with 1 extrapolation (8th-order from dt, dt/2)



AMR with Local time stepping



Space-time conservation

Space-time ghost cells

→ *Serial dependencies bottleneck around communication, strong scaling, and parallel coarse grid solvers!*

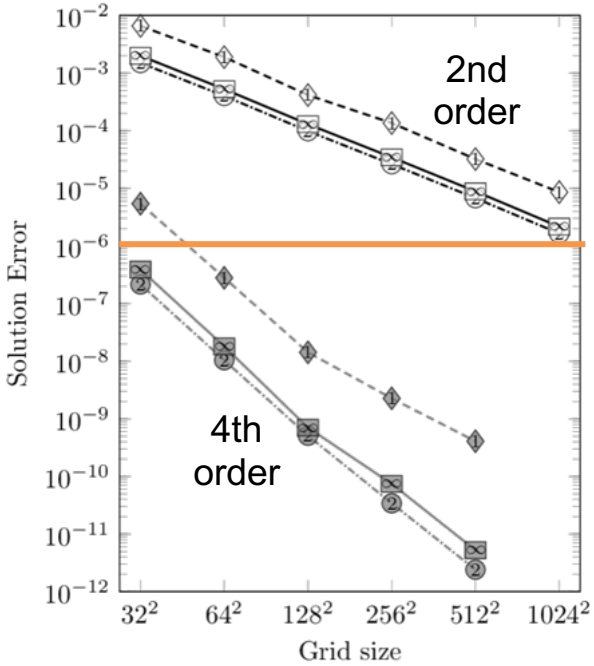
AMR with local time stepping is complicated:

- 3 levels, 2x refinement each with:
  - Time step like single level
  - "Coarse-fine" boundary conditions

  → Space-time interpolation!
- "Subcycling" – recursive time stepping
  - Finer level waits for coarse to finish
  - More communication
- Conservation?
  - Fluxes on edges must be sync'ed
  - Fine level average overwrites coarse
- Global solvers composed from levels
  - Very expensive, requires "sync" across levels for conservation

HPC and High-Order Methods

- Higher-order is ***efficient***
  → Many fewer cells for the same error across a wide variety of problems
- HPC is ***big***, LBNL NERSC Perlmutter:
  ~8000 CPUs (128 core)
  ~7000 GPUs (NVIDIA A100's)
- Communication much more expensive (and every message has latency cost)



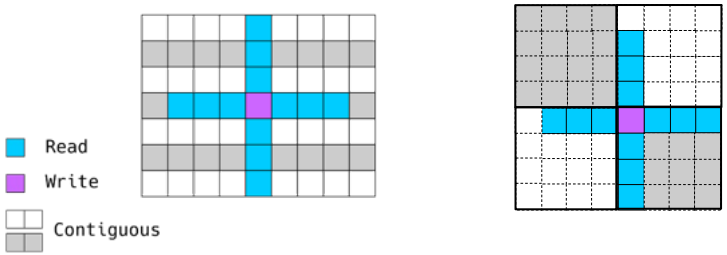
When you access memory, do everything you can with it!

Memory movement / latency dominates at scale

Higher-order stencils create 2 kinds of memory movement issues …

***On node**:*

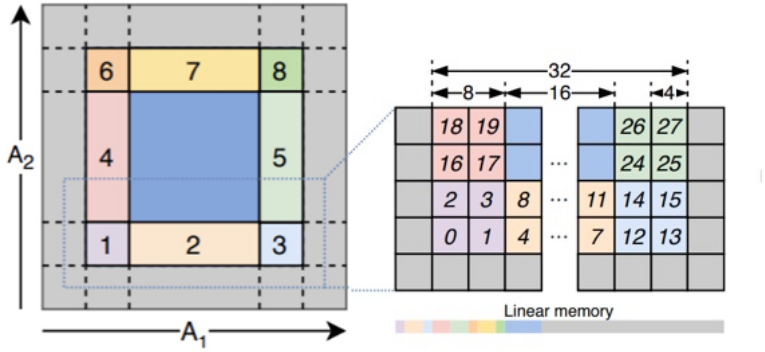
- Typical stencil loops use (i,j,k) array layout
- For larger stencils, more strided memory access, limits thread parallelism
- Tiling doesn't help much – but Bricks do!



***Off node**:*

- Bigger ghost zones, more communication
- Strided packing into MPI buffers → ***latency***
- Surface-to-volume worse with strong scaling

Use a "Brick" layout!

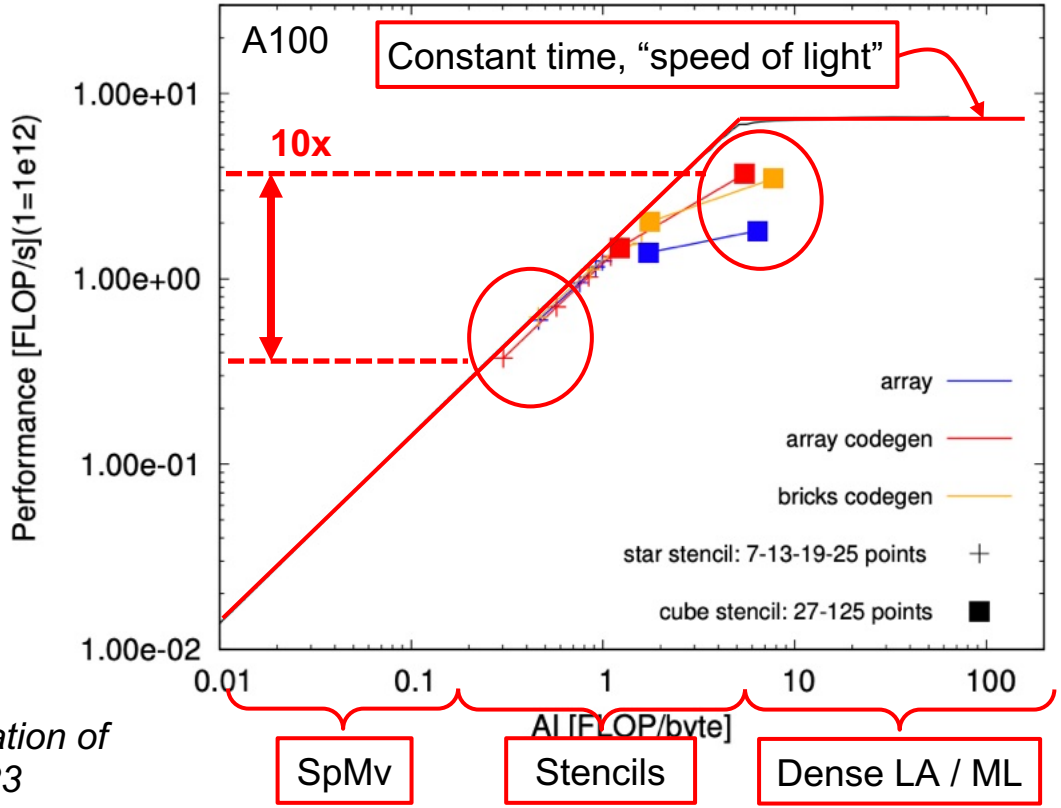


Higher-order stencils on “The Roofline”

Stencils are “low AI”:

- Bandwidth-bound for all but the largest “cube” stencils
- Memory access has big strides for non-unit direction
- You can reduce cache misses by using “brick” layouts
- Code generation needed for optimal roofline performance

(Communication roofline is worse!)

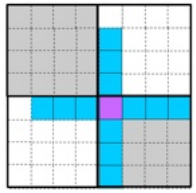


Antepara, et al, “Performance Portability Evaluation of Blocked Stencil Computations on GPUs,” SC ‘23

Part 1 summary: what is the “best” AMR?

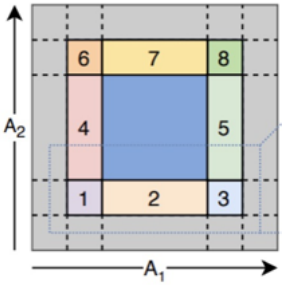
1. ***Use higher-order methods for efficiency***
   - Memory movement is the GPU mind-killer!
   - Optimize: cache reuse (with Bricks)
   - For a given accuracy, use less data (and even less mesh/solver metadata)

Higher-order stencils w/ Bricks



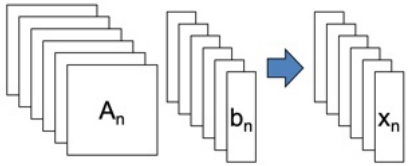
2. ***Avoid serial bottlenecks to improve scaling***
   - Optimize: maximize bandwidth, avoid latency
   - Expose more parallelism in space and time

Comm-avoiding, Parallel in Time



3. ***Use efficient parallel solvers***
   - Avoid forming and solving global matrices
   - Use cheap operator splitting where you can
   - Batch and parallelize wherever possible

Op Splitting, batch solvers



Parts 2 & 3: Fast, space-time-parallel, high-order options?

Space / time parallelism

- 4D: space-time DG, etc.
- Splitting methods: operators, directions, etc.
- Extrapolation methods
- Multi-rate and peer RK's
- Many more …

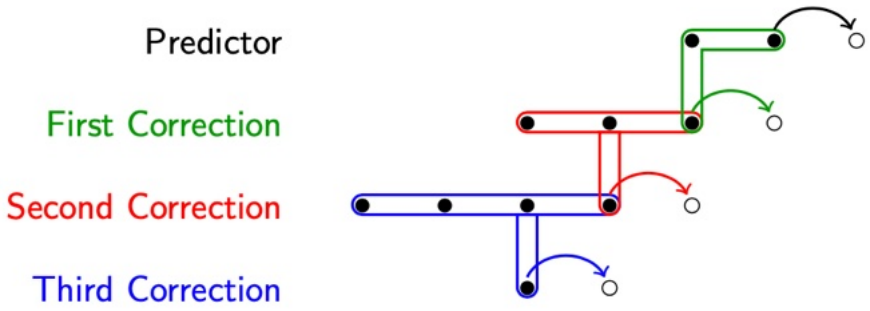
Consider the potential of this particular combination:

- Splitting:

$$\partial_t \phi = A(\phi) + B(\phi)$$

$$\phi^{n+1} = e^{\Delta t\, A} e^{\Delta t\, B} \phi^n + O(\Delta t^2)$$

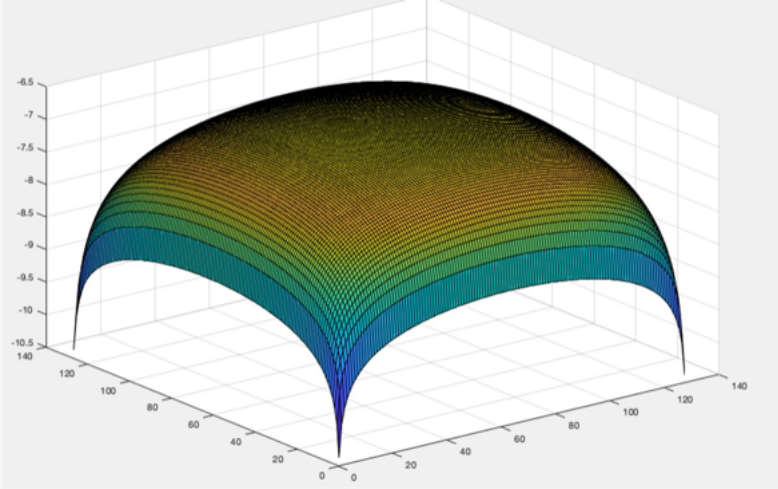
- Parallel Deferred Corrections:



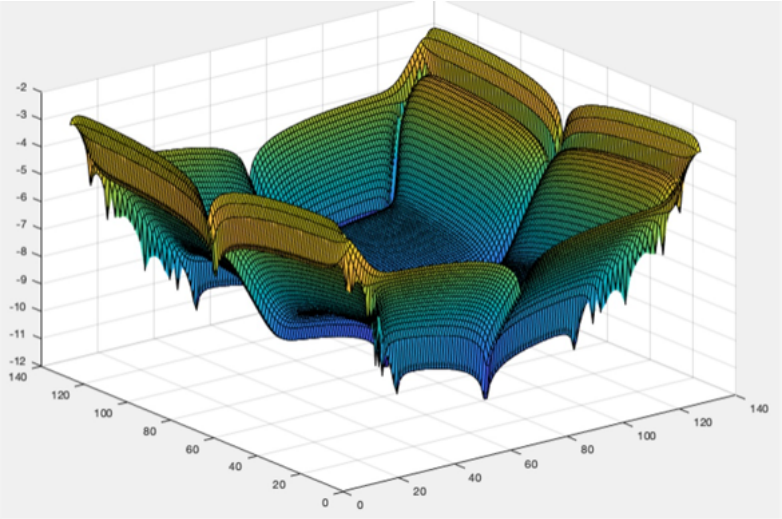
"Parallel High-order Integrators," Christlieb, et al, SISC, 2010, DOI: 10.1137/09075740X

Problems in paradise: 2D and time-dependent bc's

Strang splitting:
homogeneous (0) Dirichlet bc's:



Strang splitting:
inhomogeneous g(t) Dirichlet bc's:



Part 2: Problems in paradise with time-dependent bc's!

1D linear model equation:

- Time-dependent BC's
- Move bc's into "source term"
- Look at exact linear solution for some operator with large norm
- Look at the Taylor series, ask why the convergence is poor?
- Terms need to cancel across 10^8 orders of magnitude! (dx = 1/128, k = 1)

$$\partial_t \phi = L\phi + g(t) \quad \text{on } x \in [0,1]$$

$$g(t) = L_{bc}[\phi(0,t)\ \phi(1,t)] \quad \text{for bc's}$$

$$\phi(x,t) = \exp(-\lambda t)\cos(\pi k x)$$

$$g(t) = L_{bc}\exp(-\lambda t)[1\ (-1)^k]$$

$$\phi^{n+1} = \phi^n + \Delta t \partial_t \phi^n + \frac{1}{2}\Delta t^2 \partial_{tt}\phi^n + O(\Delta t^3)$$

$$\partial_t \phi = L\phi + g(t)$$

$$\partial_{tt}\phi = L\partial_t\phi + \partial_t g(t)$$

$$= L^2\phi + Lg(t) + \partial_t g(t)$$

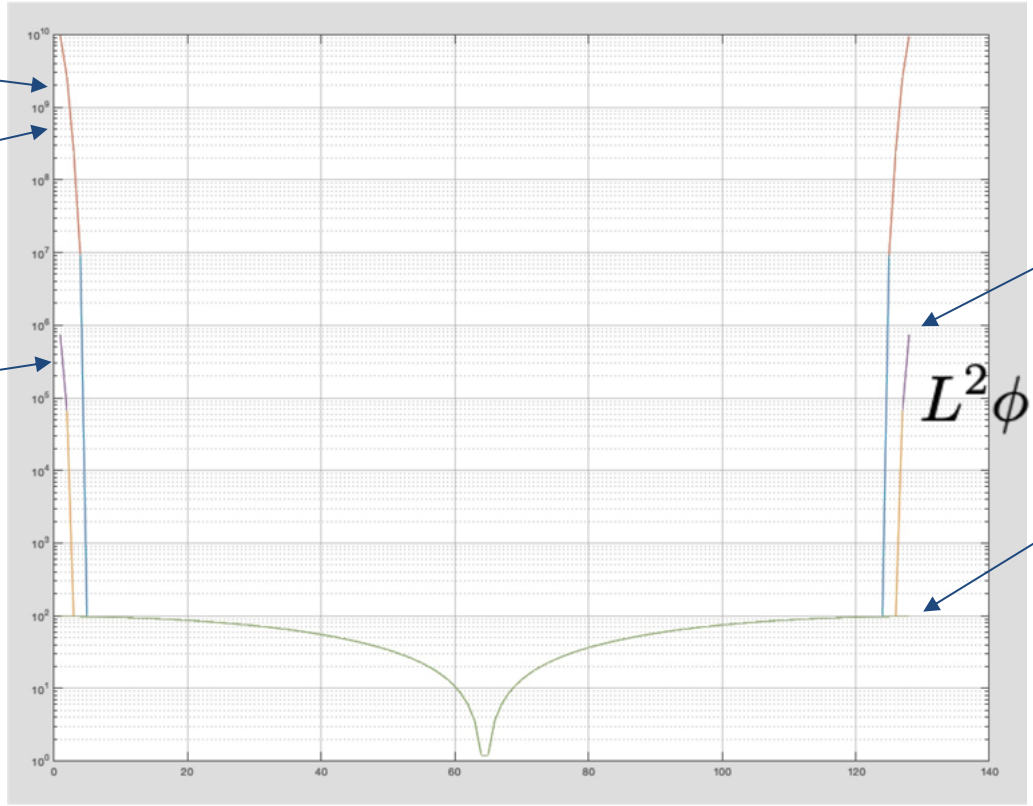
10^10 10^10 10^6 = 10^2

Example: 1D decaying cosine w/ BE, 2nd-deriv error terms

$L^2\phi$

$Lg(t)$

$\partial_t g(t)$



$L^2\phi + Lg(t)$

$L^2\phi + Lg(t) + \partial_t g(t)$

How does this compare to the exact solution?

For BE, terms cancel but missing 1/2 factor:

$$\begin{aligned}\phi^{n+1} &= (I - \Delta t L)^{-1}(\phi^n + \Delta t g^{n+1}) \\ &= (I + \Delta t L + \Delta t^2 L^2)(\phi^n + \Delta t g^{n+1}) + O(\Delta t^3) \\ &= \phi^n + \Delta t(L\phi^n + g^{n+1}) + \Delta t^2(L^2\phi^n + L g^{n+1}) \\ &= \phi^n + \Delta t(L\phi^n + g^n) + 1\Delta t^2(L^2\phi^n + L g^n + \partial_t g^n)\end{aligned}$$

For CN, centering of g makes a difference too (½ time doesn't work)

$$\phi^{n+1} = (I - \frac{1}{2}\Delta t L)^{-1}(I + \frac{1}{2}\Delta t L)(\phi^n + \Delta t g^{n+\frac{1}{2}})$$

vs.

$$\phi^{n+1} = (I - \frac{1}{2}\Delta t L)^{-1}(I + \frac{1}{2}\Delta t L)(\phi^n + \frac{1}{2}\Delta t(g^n + g^{n+1}))$$

How does this compare to the exact solution?

We know the exact solution, it just has to be computed using a quadrature, which determines the truncation error
→ do terms cancel?

Lots of options that are all consistent to 2nd-order but may have very different error constants

$$\partial_t \phi = L\phi + g(t)$$

$$\Rightarrow \phi_e(t) = \exp(Lt)\phi(0) + \int_0^t \exp(L(t-\tau))g(\tau)d\tau$$

$$\phi_e^{n+1} = (I + \Delta t L + \frac{1}{2}\Delta t^2 L^2 + \dots)\phi^n + q(g, L) + O(\Delta t^p)$$

Part 3: Quick introduction to PDC

$$
\begin{aligned}
e &= y - u \\
\partial_t e &= \partial_t y - \partial_t u \\
\partial_t e &= f(t, y) - (r + f(t, u)) \\
\partial_t e + r &= f(t, u + e) - f(t, u)) \\
q \equiv e + \int r(\tau)\, d\tau &= e + v
\end{aligned}
$$

$$
\begin{aligned}
\partial_t q &= f(t, u + q - v) - f(t, u)) \\
&\equiv g(t, q) \text{ with } q(0) = 0.
\end{aligned}
$$

Use predictor for integral of residual:

$$v(t) \equiv u^p(t) - u^p(t_0) + \int_{t_0}^{t} f(\tau, u^p(\tau))\, d\tau$$

In corrector, solve for (e.g. with C-N):

$$q_{n+1} = q_n + \frac{\Delta t}{2} \left(g(t_{n+1}, q_{n+1}) + g(t_n, q_n) \right)$$

Update corrected solution:

$$u^c_{n+1} = u^p_{n+1} + q_{n+1} - v^p_{n+1}$$

Bonus! Homogeneous bc's for error eqn!

WIP: Splitting + RIDC

Good, bad, & ugly of splitting:

Good:

- Simple 1st order, L- and A-stable
- Split solves: cheap, batched

Bad:

- Negative, complex time steps
- Strang: N operators, 2N+1 solves
- Multiplicative splitting → serial
- Loss of L- and A-stability

Ugly:

- Hard to do nonlinear, source terms

Good, bad, & ugly of PDC:

Good:

- Inherits L- & A-stability of base method
- Corrections in parallel wavefront
- Corrector improves order with same accuracy of the base method

Bad:

- Accumulates P values for P^{th} order

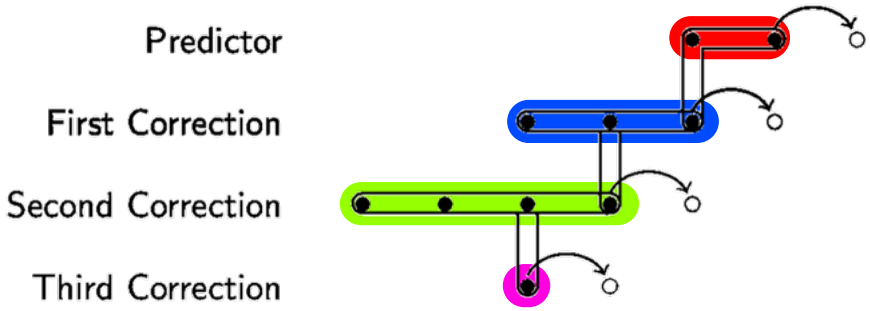
Ugly:

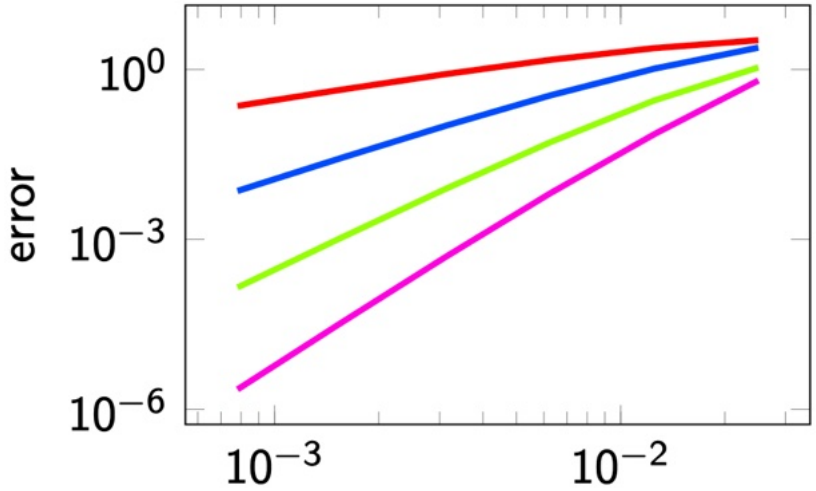
- Is an approximate (split?) operator ok?

Lie-Trotter splitting + PDC

Using Lie-Trotter (first-order) for Advection-Diffusion splitting:

- Requires P copies for P^{th} corrector
- Achieves desired order



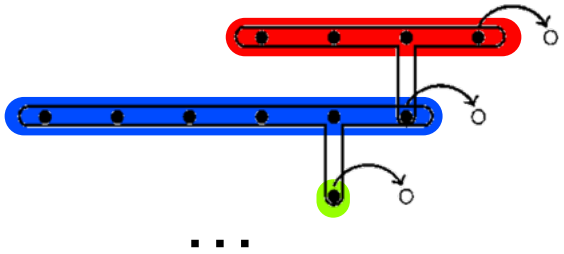


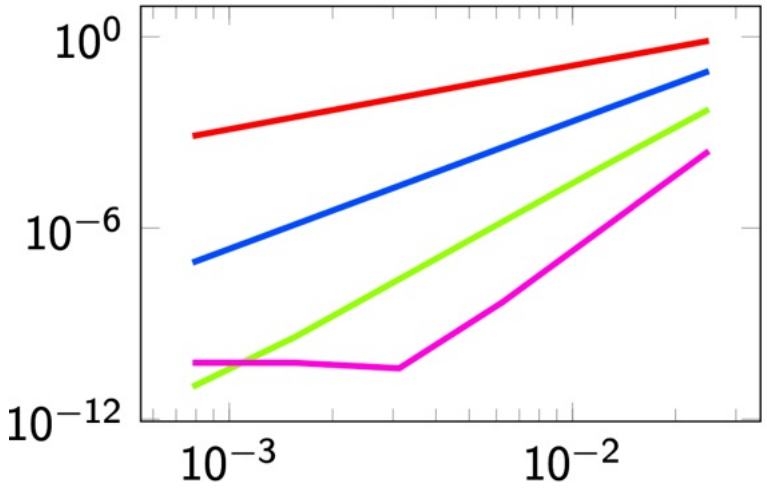
Credit: Ben Ong (Michigan Tech)

Strang splitting + PDC

Using 2nd order Strang splitting:

- Requires 2P+1 copies for Pth corrector
- Achieves desired order
- But stalls early due to conditioning (this has been improved on)





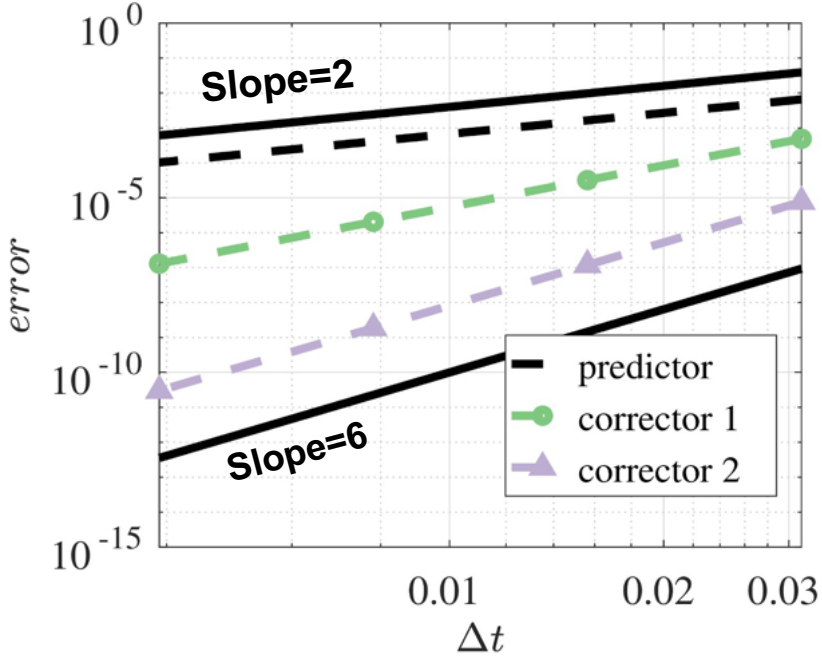
Credit: Ben Ong (Michigan Tech)

Crank Nicholson Correctors improve order by 2

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}$$

1D Advection Diffusion equation:

- 8th order spatial discretization
- periodic BCs
- predictor - Crank Nicholson
- corrector - Crank Nicholson



Credit: Rochi Chowdhury (LBNL)

Why does this work?

Error analysis for splitting:

- Assume linear operators, exact error for one step:
- Error grows only from residual (ideas similar to EXP+ROS but with approximate exponential)
- Like a Rosenbrock method, splitting approximations just need to cancel to higher order!

$$\delta_{n+1}^{[m]} = e^{Lh}\,\delta_n^{[m]} + \int_{t^n}^{t^{n+1}} e^{L(t^{n+1}-\tau)}\, g(\tau)\, d\tau$$

Controls growth of error from residual

Quadrature error to order of corrector

Must be order of corrector

Continuous residual from interpolant

Rosenbrock Methods ideal for Stiff Nonlinear problems

What if the equation is nonlinear?

- Linearly Implicit Runge Kutta methods
- Ideal for Stiff differential equations
- Ideal for Nonlinear System
- Linearizes problem around current time
- Leverages the Jacobian matrix to avoid solving Newton Iterations
- L- & A-stable low-order options

$$y'(t) = f(t, y), t \in [0, T]$$

$$(I - \theta \Delta t J_n) k_i = \Delta t f(t_n + \alpha_i \Delta t, y_n + \Sigma_{j=1}^{i-1} \alpha_{ij} k_j) + \gamma_i \Delta t^2 \frac{\partial f}{\partial t}(t_n, y_n) + \Delta t J_n \Sigma_{j=1}^{i} \gamma_{ij} k_j$$

$$J_n = \frac{\partial f(y_n)}{\partial y}$$

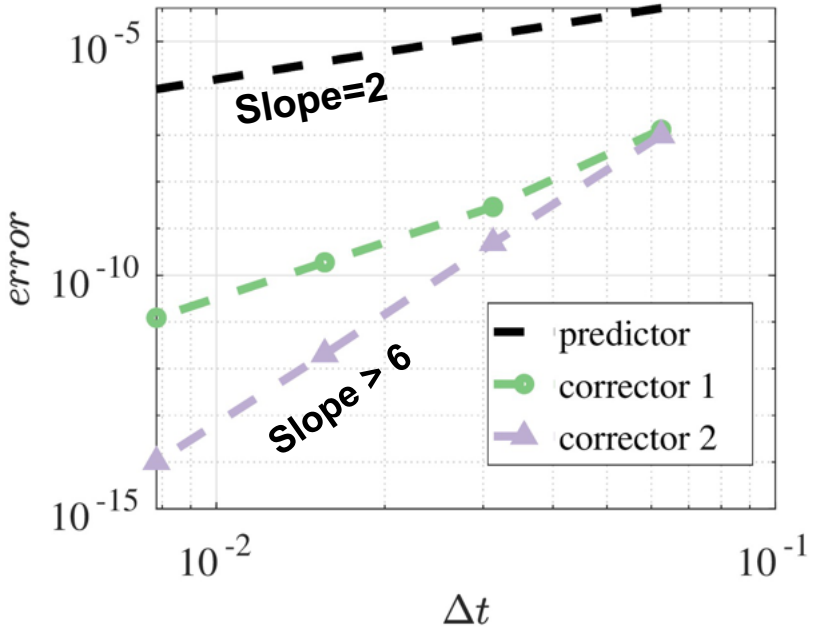
$$y_{n+1} = y_n + \Sigma_{i=1}^{s} b_i k_i$$

Viscous Burgers using ROS-2 and Deferred Correction

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

1D Viscous Burger's equation:

- Periodic BC's
- 8th-order spatial discretization
- ROS-2 predictor and corrector
- Time errors << spatial errors



Credit: Rochi Chowdhury (LBNL)

In combination, very flexible!

Compact Stencil (Padé) least squares system

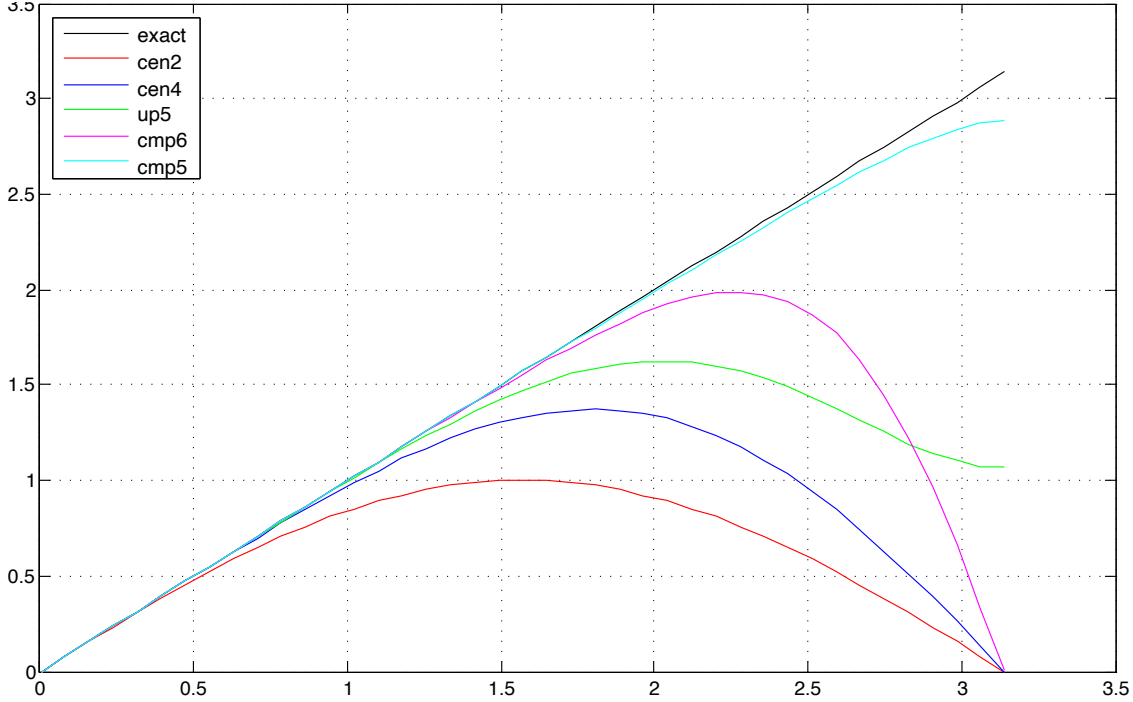
(like Mehrstellen):

$$s_l^T \langle \partial_x \phi \rangle = s_r^T \langle \phi \rangle$$
$$s_l^T D c = s_r^T A c$$
$$D^T s_l = A^T s_r$$

s_l "implicit" stencil on face fluxes

D "implicit" fluxes on coefficients

Null space is the compact stencil!
Only tri- or penta-diagonal solves



In combination, very flexible!

Compact Stencil (Padé) least squares system

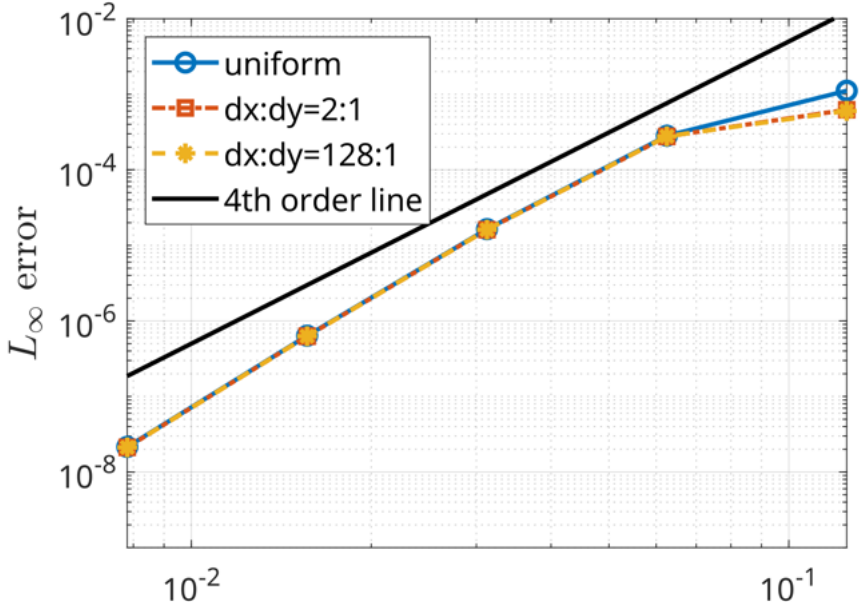
(like Mehrstellen):

$$s_l^T \langle \partial_x \phi \rangle = s_r^T \langle \phi \rangle$$
$$s_l^T D c = s_r^T A c$$
$$D^T s_l = A^T s_r$$

s_l "implicit" stencil on face fluxes

D "implicit" fluxes on coefficients

Null space is the compact stencil!
Only tri- or penta-diagonal solves



Anisotropic ADI for heat equation

Recap and Potential Collaborations?

We've been working hard to make AMR higher-order and more parallel!

- Accuracy through error correction, across AMR levels
- Stability through implicit, cheap solvers
- Software – implementing with a distributed parallel hash on GPUs

Next steps / futures / collaborations:

- Proving stability conditions?
- Parallel, higher-order in space-time w/ AMR?
- More applications: cut cells and mappings for boundary layer, planetary sciences, etc.
- Of course, applications / interesting physics

Thanks!

Please reach out! hjohansen@lbl.gov

Thanks to colleagues, collaborators, and funders!

Applied Mathematics & Computational Research

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Base Math and SciDAC / FASTMath programs

This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility located at Lawrence Berkeley National Laboratory, operated under Contract No. DE-AC02-05CH11231