Advancing parabolic operators in thermodynamic MHD models: Explicit super time-stepping vs. implicit methods with Krylov solvers

Ronald M. Caplan, Zoran Mikić
Jon A. Linker, Roberto Lionello

Predictive Science Inc.
www.predsci.com

9990 Mesa Rim Road
Suite 170
San Diego, CA 92121

Funding provided by

SC16 Salt Lake City, Utah | hpc matters.

INTRODUCTION

We explore the performance and scaling of using explicit super time-stepping (STS) algorithms versus implicit schemes with Krylov solvers for integrating parabolic operators in thermodynamic MHD models of the solar corona. Specifically, we compare the second-order Runge-Kutta Legendre (RKL2) STS method with implicit backward Euler computed using the preconditioned conjugate gradient (PCG) solver with both a point-Jacobi and a non-overlapping domain decomposition ILU0 preconditioner.

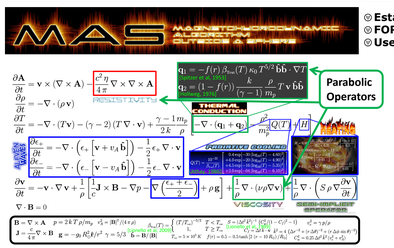
The algorithms are used to integrate a scalar operator (anisotropic Spitzer thermal conduction) and a vector operator (artificial kinematic viscosity) at time-steps much larger than the explicit Euler limit. A key component of the comparison is the use of a real-world simulation on large state-of-the-art HPC systems.

A paper detailing this study has been submitted and is made available [*arXiv:1610.01265*].

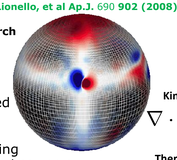
MODEL AND NUMERICAL METHODS

MAS MAGNETOHYDRODYNAMIC ALGORITHM OUTREACH SOLUTIONS

- **Established MHD Solar coronal model [Lionello, et al ApJ. 690 902 (2008)]**
- **FORTRAN 90 + MPI-2**
- **Used extensively in solar physics research**



- Finite difference on logically rectangular, staggered, non-uniform spherical grid
- Advective terms are upwinded
- Wave terms use predictor-corrector
- Semi-implicit term solved using BE+PCG for both predictor and corrector
- Parabolic terms are operator split and use second-order central differences
- Matrix operators stored in modified DIA format, PC2 preconditioner stored in CSR format



Parabolic operators tested with algorithms:

Kinematic Viscosity

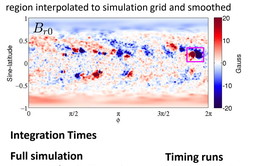
$$\nabla \cdot \left(\nu \, \rho \, \nabla \, \mathbf{v} \right)$$

Thermal Conduction

$$\nabla \cdot \left(\kappa_0 \, T^{5/2} \, \hat{\mathbf{b}} \, \hat{\mathbf{b}} \cdot \nabla \, T \right)$$

REAL-WORLD 3D TEST CASE

Boundary Conditions
Air Force Data Assimilative Photospheric Flux Transport (ADAPT) [Arge et al. 2009] with added backside active region interpolated to simulation grid and smoothed

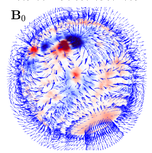


Integration Times

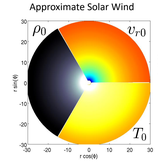
Full simulation
Integrate for ~ 48 sim-hours
Validation runs
Integrate for ~8 sim-hours

Initial Conditions

Magnetic Field:
Potential Field Source Surface



Plasma Fields:
Approximate Solar Wind

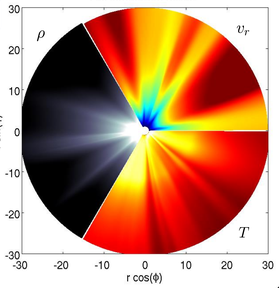


Timing runs
Integrate for 6 sim-min starting with the ~8 sim-hr state

The Grid
181x251x602
~27 million points

$$\min\left[\Delta r\right] \approx 340\,\mathrm{km}$$
$$\max\left[\Delta r\right] \approx 590,000\,\mathrm{km}$$
$$\max\left|1 - \frac{\Delta r_{i+1}}{\Delta r_i}\right| < 6\%$$

Full Simulation Result



Implicit Backward Euler with Preconditioned Conjugate Gradient

Backward Euler (BE): simplest L-stable method

$$\frac{\partial u}{\partial t} = F(u, \mathbf{r}) \quad \Rightarrow \quad \frac{u^{n+1} - u^n}{\Delta t} = \mathrm{M}\, u^{n+1}$$

Applying BE to PDE yields a system of equations to solve

$$(1 - \Delta t\, \mathrm{M})\, u^{n+1} = u^n \quad \Rightarrow \quad \mathbf{A}\, x = y$$

To solve system, use Krylov subspace iterative method (e.g. Conjugate Gradient)

- Robust, proven method
- Can be very efficient
- Pre-made libraries available for many applications
- L-stability allows the use of very large time-steps

- Requires a quality preconditioner to be efficient, which can be very difficult to formulate, implement, vectorize, and scale. The PCs can also suffer "break down". Requires linear operator
- Choosing the convergence tolerance is not trivial.
- Global communication for dot products limits scaling (global sync).
- Dot product can suffer from floating point errors requiring quad precision

Preconditioners (helps solver converge)
We use two *communication-free* PCs:
PC1: Point-Jacobi / Diagonal scaling (DIAG)
PC2: Non-overlapping domain decomposition with zero-fill incomplete LU factorization (NDD+ILU0)

PC1 (DIAG):

```
do j = 1 : N
   P_jj = A_jj
enddo
```

```
do i = 1 : N
   z_i = P_ii r_i
enddo
```

PC2 (NDD+ILU0):

```
LU = A
do i = 2 : N
   do k = 1 : i - 1        (LU_ik ≠ 0)
      LU_ik = LU_ik / LU_kk
      do j = k + 1 : N     (LU_ij ≠ 0)
         LU_ij = LU_ij - LU_ik LU_kj
      enddo
   enddo
enddo
P = LU
```

```
do i = 1 : N
   z*_i = r_i
   do j = 1 : i        (LU_ij ≠ 0)
      z*_i = z*_i - LU_ij z*_j
   enddo
enddo
do i = N : 1
   z_i = z*_i
   do j = i + 1 : N    (LU_ij ≠ 0)
      z_i = z_i - LU_ij z_j
   enddo
   z_i = z_i / LU_ii
enddo
```

Point-2-Point comm+sync

$$x_0 = u^n \qquad z_0 = \mathbf{P}^{-1} r_0$$
$$r_0 = b - \mathbf{A}\, x_0 \qquad p_0 = z_0$$
$$\mathbf{P} \approx \mathbf{A} \qquad r_r = r_0 \cdot z_0$$

Global comm+sync

```
do k = 0 : k_max
   y_k = A p_k
   α_k = r_r / (p_k · y_k)
   x_{k+1} = x_k + α_k p_k
   r_{k+1} = r_k - α_k y_k
   z_{k+1} = P^{-1} r_{k+1}
   r_dd = r_r
   r_r = r_{k+1} · z_{k+1}
   Check r_r for convergence
   β_k = r_r / r_dd
   p_{k+1} = β_k p_k + z_k
enddo
```

$$u^{n+1} = x_{k+1}$$

VS

Explicit Runge-Kutta Legendre 2nd order

- Unconditionally stable, but *explicit*!
- Main idea: Runge-Kutta method with stages added for more stability, rather than for more accuracy
- Two main flavors are RKC (Chebyshev-based) and RKL (Legendre-based) and can be recursive or factored
- RKL2 [Meyers et al, J. Comp. Phys. **257,** 594 (2014)]

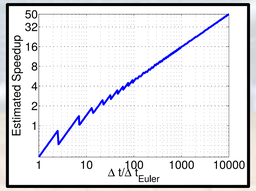
Number of sub-steps

$$s \geq \left\lceil \frac{1}{2} \left(\sqrt{9 + 16 \frac{\Delta t}{\Delta t_{\mathrm{Euler}}}} - 1 \right) \right\rceil$$

$$\mathrm{mod}(s, 2) \neq 0$$

- Very easy to implement (ideal for directive parallelization e.g. OpenMP/OpenACC)
- No global synchronization points
- Can include nonlinearities and/or flux limiting

- New method – low circulation
- Number of sub-cycles can be large depending on the problem and grid
- Amplification factor not monotonically decreasing for increasing wave modes – *can lead to undesirable results when using large time-steps and/or short simulation times*



$$y_0 = u^n$$
$$F_0 = \mathbf{M}\, y_0$$
$$y_1 = y_0 + \tilde{\mu}_1\, \Delta t\, F_0$$

Point-2-Point comm+sync

```
do j = 2 : s
   y_j = μ_j y_{j-1} + ν_j y_{j-2} + (1 - μ_j - ν_j) y_0
         + μ̃_j Δt M y_{j-1} + γ_j Δt F_0
enddo
```

$$u^{n+1} = y_s$$

$$b_0 = b_1 = b_2 = \frac{1}{3} \qquad b_j = \frac{j^2 + j - 2}{2\,j\,(j+1)}$$
$$\tilde{\mu}_1 = \frac{4}{3\,(s^2 + s - 2)} \qquad \tilde{\mu}_j = \frac{4\,(2\,j - 1)}{j\,(s^2 + s - 2)} \frac{b_j}{b_{j-1}}$$
$$\mu_j = \frac{2\,j - 1}{j} \frac{b_j}{b_{j-1}} \qquad \nu_j = -\frac{j - 1}{j} \frac{b_j}{b_{j-2}}$$
$$\gamma_j = (b_j - 1)\, \tilde{\mu}_j$$

HPC ENVIRONMENT

San Diego Supercomputing Center

COMET SDSC

Texas Advanced Computing Center

TEXAS ADVANCED COMPUTING CENTER STAMPEDE
TACC's most powerful supercomputer powers discoveries for science and society

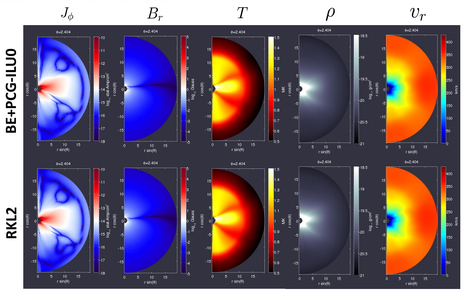
	SDSC Comet	TACC Stampede
Processor Model	Intel Xeon E5-2660v3	Intel Xeon E5-2680v2
Clock Speed	2.5 GHz	2.7 GHz
CPU Cores/node	24	16
DRAM/node (ECC)	128 GB DDR4-2133MHz	32 GB DDR3-1600MHz
Max # of cores/job	1728	4096
Max Flops/node	960 GFlops	346 GFlops
Network	FDR InfiniBand Hybrid Fat-Tree	FDR InfiniBand 2-level Fat-Tree
MPI Library	MVAPICH 2.1	Intel 2015.2.164
Compiler	Intel 2015.2.164	Intel 2015.2.164
OS	CentOS 2.6.32-573	CentOS 2.6.32-431

Allocations provided by:

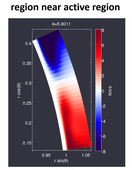
XSEDE Extreme Science and Engineering Discovery Environment

VALIDATION

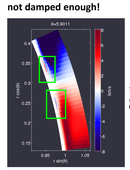
Solution cuts taken after relaxation:



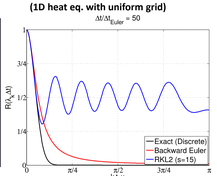
Zoomed view in transition region near active region



Oscillations in localized area not damped enough!



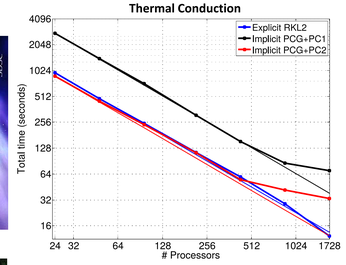
Amplification Factors (1D heat eq. with uniform grid)



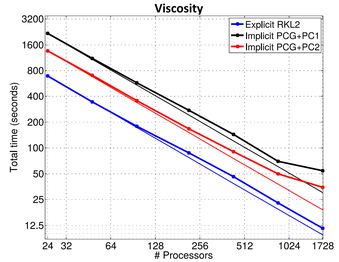
BE+PCG-ILU0 RKL2

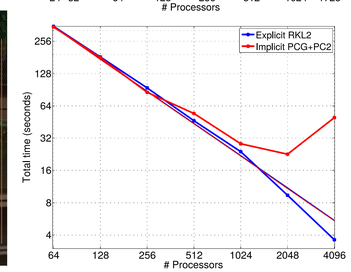
RESULTS

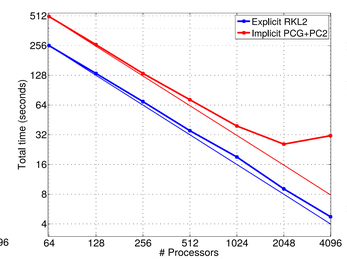
Thermal Conduction



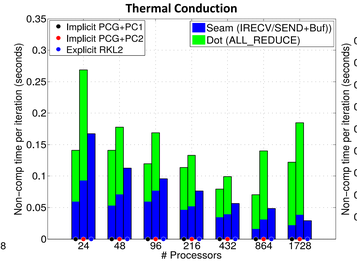
Viscosity



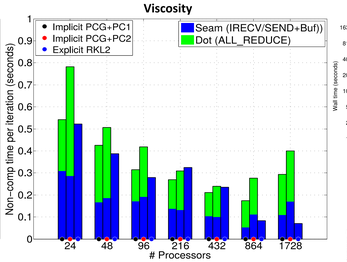


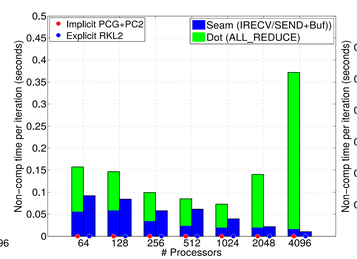


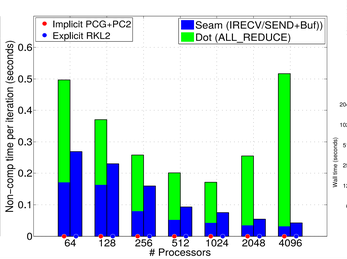
Thermal Conduction



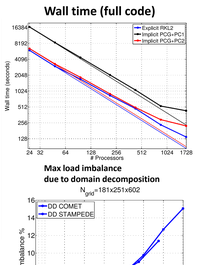
Viscosity



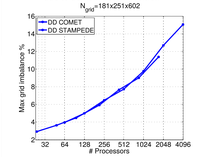




Wall time (full code)



Max load imbalance due to domain decomposition
N_{grid}=181x251x602



Wall time (full code)