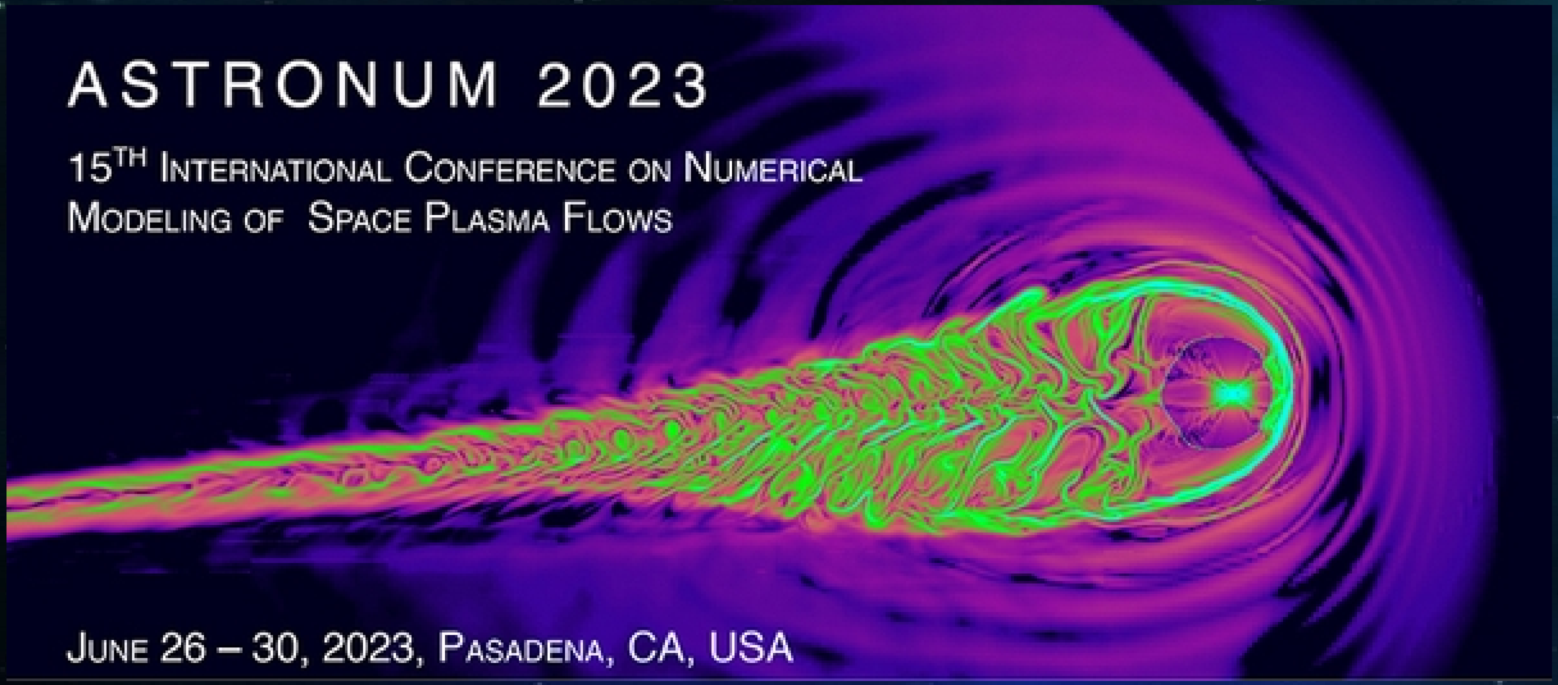
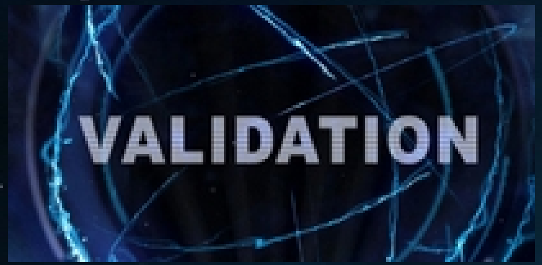
Evaluating a Practical Parabolic Time Step Limit for Unconditionally Stable Schemes in a Thermodynamic MHD Model

Ronald M. Caplan, Craig D. Johnston,
Lars K. S. Daldorff, and Jon A. Linker

Predictive Science Inc.



- Unconditionally stable schemes
- The problem of large time steps
- Practical time-step limit
- MAS thermodynamic MHD model
- Test cases
- Validation
- Performance & scaling
- Summary & future development



Unconditionally Stable Schemes

- Thermodynamic MHD models (like many others) have multiple time scales leading to vastly different explicit time-step stability requirements
- In order to make simulations *tractable*, we need to exceed the most restrictive limits - here, we focus on the parabolic operators
- Unconditionally stable time stepping schemes are guaranteed to be stable for any sized time step.
- Implicit methods (using iterative matrix solvers)
- Explicit methods (e.g. extended stability Runge Kutta)
- When exceeding explicit time-step limits, one must be careful about accuracy

Unconditionally Stable Schemes

Implicit Backward-Euler (BE) + PCG

- Backward Euler $\frac{u^{n+1} - u^n}{\Delta t} = F(u^{n+1})$
- Yields a system of equations to solve
- To avoid requiring nonlinear solvers, we linearize nonlinear terms (e.g. lagged diffusivity)
- We use two non-communicating preconditioners:

PC1

Point-Jacobi
GPU friendly, scalable, cheap, not very effective

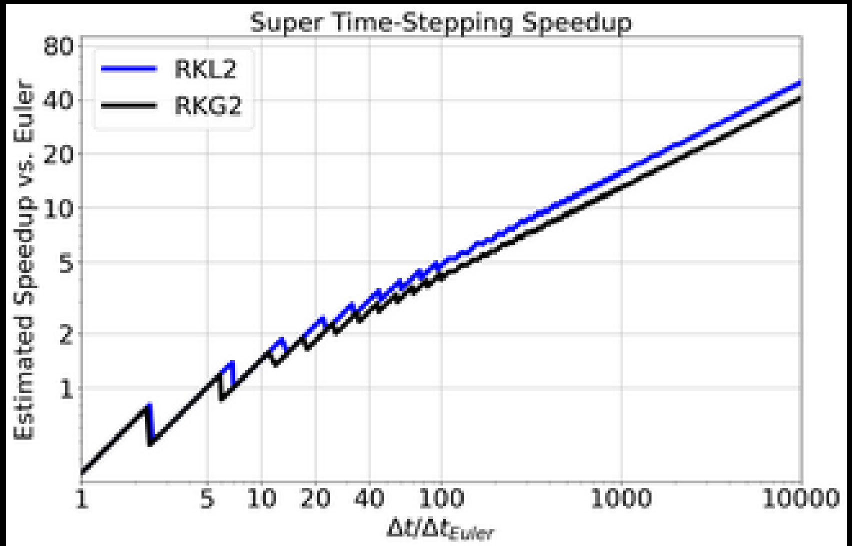
PC2

Non-overlapping domain decomposition zero-fill incomplete LU factorization
Less scalable, expensive, much more effective

Explicit RKG2(3/2) Super Time-stepping

- Extended Stability Runge-Kutta schemes: Explicit RK method with stages added for more stability, rather than for more accuracy
- We use the 2nd-order Gegenbauer method (RKG2) [O'Sullivan (2019)] with an alpha of 3/2

"RKG allows for accurate modeling of solutions at early times and as such if early times are under investigation RKG is the optimal scheme. However, RKC, RKL, RKU, and RKG are all linearly stable and as such will all approach the correct solution asymptotically at long times." [Skaras et. al. (2021)]



TIP: Need to use odd # of stages in RKG2, otherwise amplification factor goes to 1 at highest mode!

Unconditionally Stable Schemes

Implicit BE + PCG

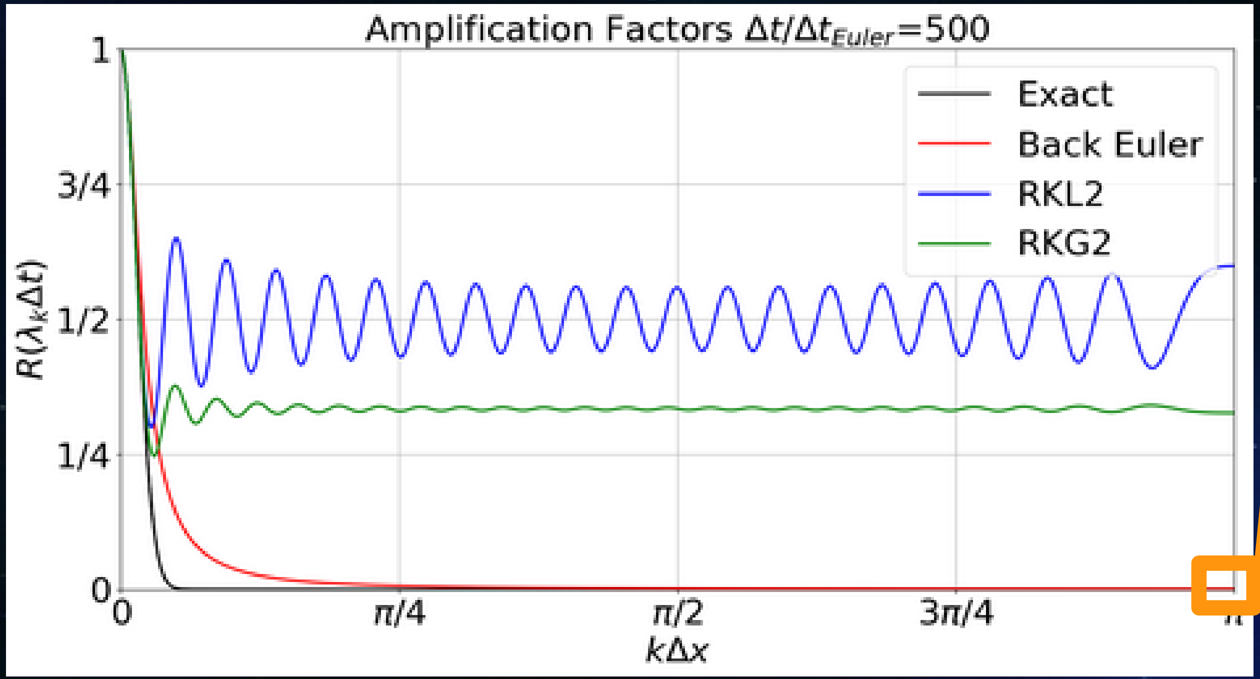
VS.

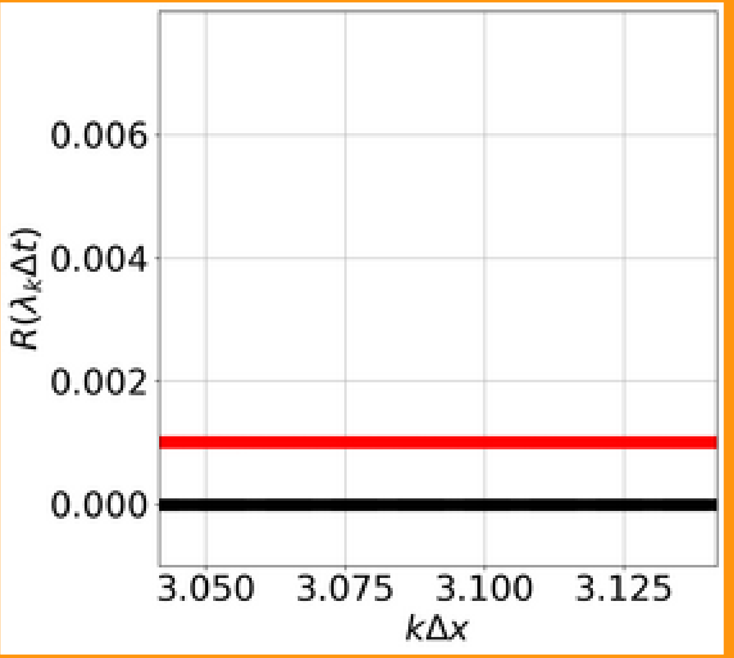
- ✓ Robust, proven method
- ✓ Can be very efficient
- ✓ **L-stable**
- × Can be difficult to implement
- × Requires good preconditioner to be efficient, (can be difficult to formulate and implement efficiently)
- × Requires linear(ized) operator
- × Global communication (dot products) hurts scaling
- × Only 1st-order accurate

Explicit RKG2(3/2)

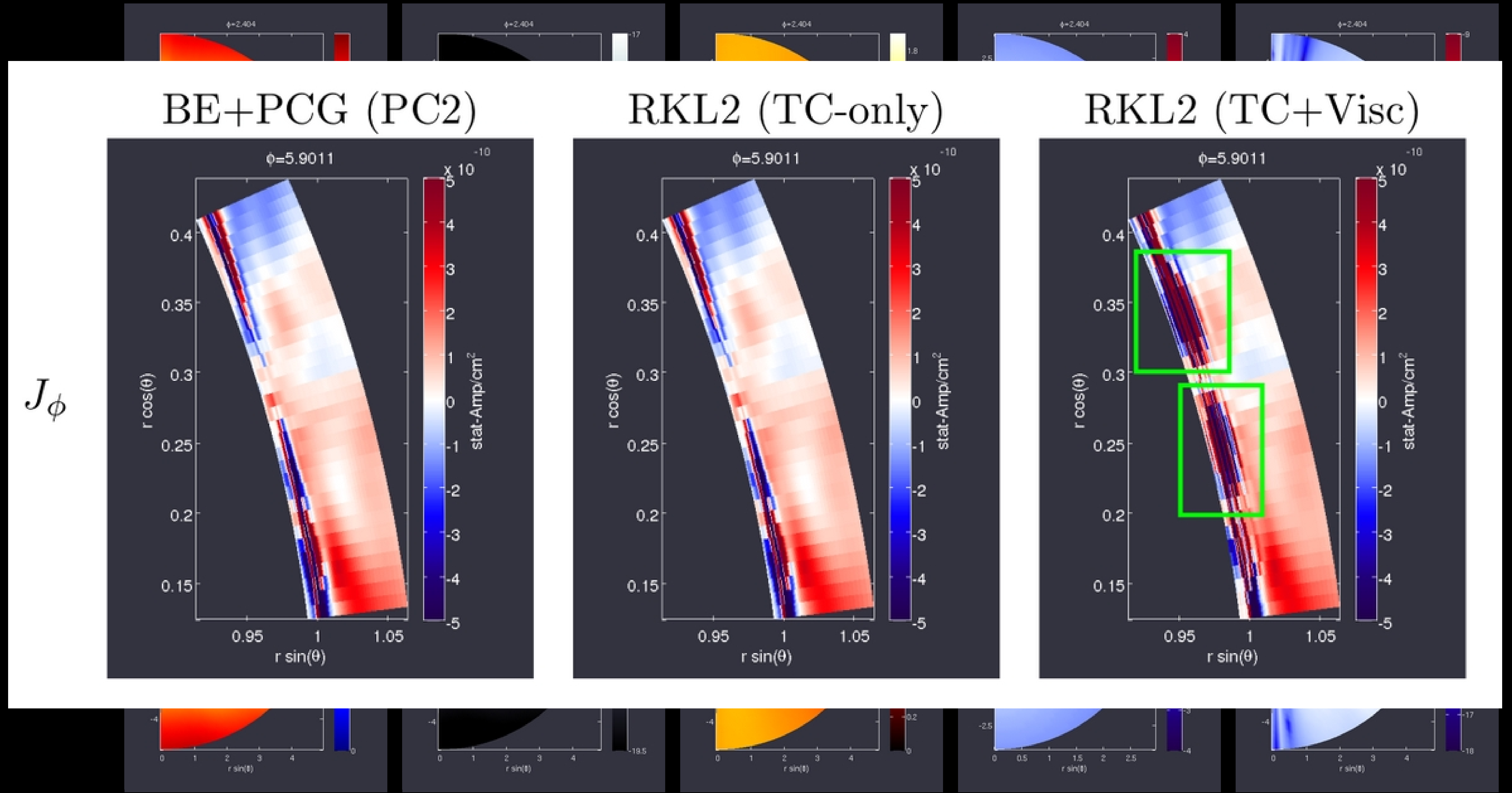
- ✓ Easy to implement
- ✓ Can include nonlinearities
- ✓ Vectorizable (GPU-friendly)
- ✓ No global synchronization points (better scaling)
- ✓ 2nd-order accurate
- × Not as widely adopted & tested
- × Can be slower than implicit methods
- × **Only A-stable**

- Example L-stable vs. A-stable amplification factors
- A-stable method can have problems damping high wave modes over limited time scales





ASTRONUM 16 Result [Caplan et. al. (2017)]



The Problem with Large Time Steps

- Errors with very large time steps are a problem in L-stable methods, but worse in A-stable methods, as they often don't damp high wave modes efficiently
- Extended stability Runge-Kutta schemes fall into this latter category, and this issue can limit their applicability and robustness
- One option is to run the parabolic advance in a series of "outer" cycles (essentially reducing the time step for the operator) which lowers the errors and repeatedly damps high wave modes

"For general applications the universal approach is to try different numbers of steps and study any sensitivities." **[Dawes (2021)]**

- **Is there a simple way to a-priory calculate the minimum number of outer cycles we need/want?**
- **How does adding these cycles affect performance?**

A Practical Time Step Limit

Operator split parabolic advance

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{F}(\mathbf{u})$$

Discrete form (1st expansion term)

$$\frac{u^{n+1} - u^n}{\Delta t} = F(u)$$

$$\left|u^{n+1} - u^n\right| = \Delta t\, |F(u)|$$

Max abs change in u at grid cell k

$$|F_k(u)| \equiv \max\left(|F(u_i)|\right)$$

Bound the relative change in u at the location of maximum absolute change

$$\frac{\left|u_k^{n+1} - u_k^n\right|}{|u_k|} < \alpha < 1$$

Practical time step limit: $\Delta t_p = \alpha \dfrac{|u_k|}{|F(u_k)|}$

$\alpha = 0.95$

Applied adaptively:
After each outer-cycle, recalculate!

Viscosity: $\mathbf{F}_{\mathrm{visc}}(\mathbf{v}) = \dfrac{1}{\rho} \nabla \cdot \left(\nu(\mathbf{x})\, \rho\, \nabla \mathbf{v}\right)$

Thermal Conduction:

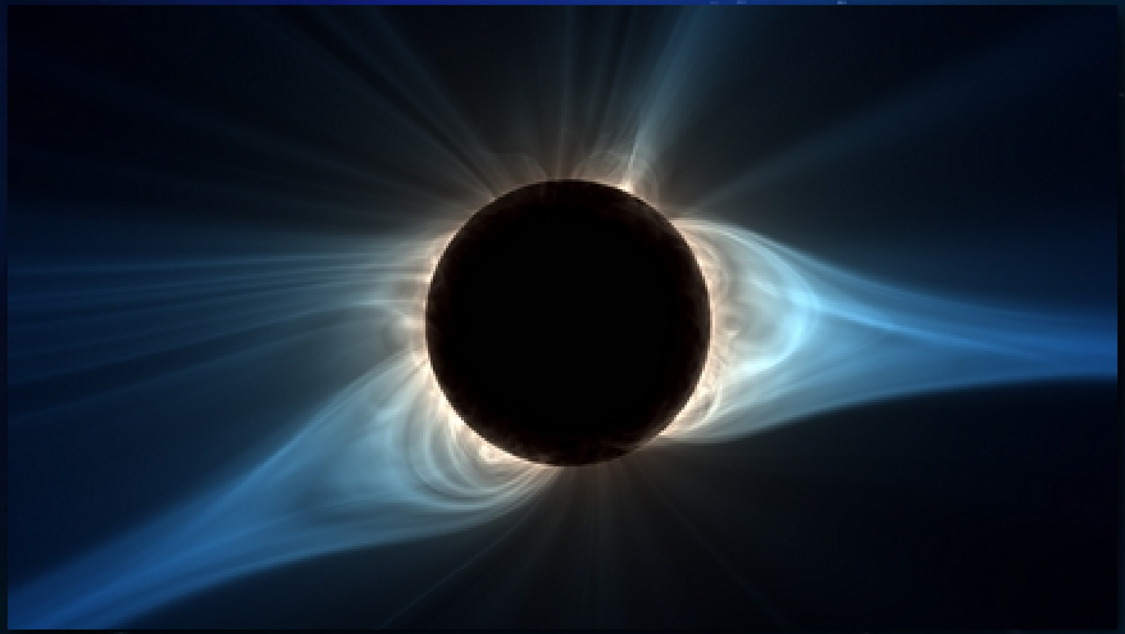
$$\mathbf{F}_{\mathrm{tc}}(T) = \frac{(\gamma - 1)\, m_p}{2\, k} \frac{1}{\rho} \nabla \cdot \left(\kappa(T_0) \cdot \nabla T\right)$$

$$\kappa(T) = f_C(r)\, f_{\mathrm{mod}}(T)\, \kappa_0\, T^{5/2}\, \hat{\mathbf{b}}\hat{\mathbf{b}}$$

$$f_{\mathrm{mod}}(T) = \left(1 + (T/T_{\mathrm{cut}})^{-10}\right)^{1/4}$$

Lagged diffusivity

MAS
MAGNETOHYDRODYNAMIC ALGORITHM OUTSIDE A SPHERE



Purpose:

General-purpose simulations of the corona and heliosphere for use with solar physics and space weather research

Model:

Spherical resistive thermodynamic MHD

Algorithms:

Implicit & explicit time-stepping with finite-difference stencils. Implicit steps use a sparse matrix preconditioned iterative solver

Code:

~70,000 lines of Fortran parallelized with MPI + OpenACC + StdPar

The MAS MHD Model

MAS
MAGNETOHYDRODYNAMIC ALGORITHM OUTSIDE A SPHERE

$$\frac{\partial \mathbf{A}}{\partial t} = \mathbf{v} \times (\nabla \times \mathbf{A}) - \boxed{\frac{c^2\eta}{4\pi}\nabla \times \nabla \times \mathbf{A}}$$

RESISTIVITY

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho\,\mathbf{v})$$

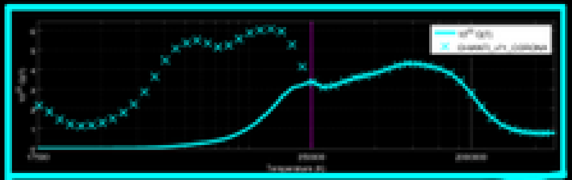
$$\frac{\partial T}{\partial t} = -\nabla \cdot (T\mathbf{v}) - (\gamma - 2)\,(T\,\nabla \cdot \mathbf{v}) + \frac{\gamma - 1}{2k}\frac{m_p}{\rho}\left[\boxed{-\nabla \cdot (\mathbf{q}_1 + \mathbf{q}_2)} - \boxed{\frac{\rho^2}{m_p^2}Q(T)} + \boxed{H}\right]$$

THERMAL CONDUCTION

$$\mathbf{q}_1 = -f(r)\,\beta_{T_{cut}}(T)\,\kappa_0 T^{5/2}\,\hat{\mathbf{b}}\hat{\mathbf{b}} \cdot \nabla T$$

$$\mathbf{q}_2 = (1 - f(r))\,\frac{k}{(\gamma - 1)}\frac{\rho}{m_p}T\,\mathbf{v}\,\hat{\mathbf{b}}\hat{\mathbf{b}}$$

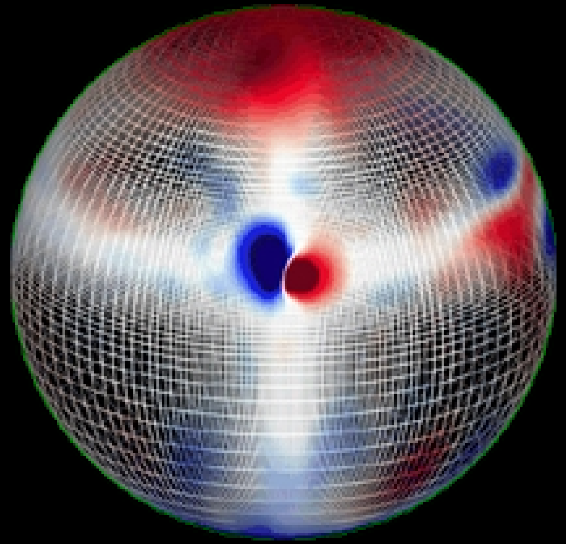
RADIATIVE COOLING



CORONAL HEATING

$$H = H^* + \frac{\rho}{4\lambda_\perp}\left[|z_-|\,z_+^2 + |z_+|\,z_-^2\right]$$

$$\lambda_\perp = \boxed{\lambda_0}\sqrt{\frac{B_w}{|\mathbf{B}|}} \qquad |z_\pm(r = R_\odot)| = \boxed{z_0}$$



(r_i, θ_j, ϕ_k)

ALFVEN WAVES

$$\boxed{\frac{\partial \epsilon_\pm}{\partial t} = -\nabla \cdot (\epsilon_\pm[\mathbf{v} \pm \mathbf{v_A}]) - \frac{\epsilon_\pm}{2}\nabla \cdot \mathbf{v}}$$

$$\frac{\partial \mathbf{v}}{\partial t} = -\mathbf{v} \cdot \nabla\mathbf{v} + \frac{1}{\rho}\left[\frac{1}{c}\mathbf{J} \times \mathbf{B} - \nabla p - \boxed{\nabla\left(\frac{\epsilon_+ + \epsilon_-}{2}\right)} + \rho\,\mathbf{g}\right] + \boxed{\frac{1}{\rho}\nabla \cdot (\nu\rho\nabla\mathbf{v})} + \boxed{\frac{1}{\rho}\nabla \cdot \left(S\,\rho\,\nabla\frac{\partial \mathbf{v}}{\partial t}\right)}$$

VISCOSITY

SEMI-IMPLICIT OPERATOR

WAVE TURBULENCE

$$\frac{\partial z_\pm}{\partial t} = -(\mathbf{v} \pm \mathbf{v_A}) \cdot \nabla z_\pm - \frac{z_\pm |z_\mp|}{2\lambda_\perp} + \frac{z_\pm}{4}(\mathbf{v} \mp \mathbf{v_A}) \cdot \nabla(\ln\rho) + \frac{z_\mp}{2}(\mathbf{v} \mp \mathbf{v_A}) \cdot \nabla(\ln|\mathbf{v_A}|)$$

$\nabla \cdot \mathbf{B} = 0$, $\quad \mathbf{B} = \nabla \times \mathbf{A}$, $\quad \mathbf{J} = \frac{c}{4\pi}\nabla \times \mathbf{B}$

$p = 2kT\rho/m_p$, $\quad \mathbf{g} = -g_0 R_\odot^2\hat{\mathbf{r}}/r^2$, $\quad \gamma = 5/3$

$\hat{\mathbf{b}} = \mathbf{B}/|\mathbf{B}|$, $\quad \mathbf{v_A} = \mathbf{B}/\sqrt{4\pi\rho}$, $\quad B_w = 6.09\,G$, $\quad v_s^2 = \gamma p/\rho$

$\beta_{T_{cut}}(T) = \begin{cases} (T/T_{cut})^{-5/2} & T < T_{cut} \\ 1, & T \geq T_{cut} \end{cases}$, $\quad T_{cut} = 3.5 \times 10^5\,K$

$f(r) = 1 - 0.5\tanh[(r - 10R_\odot)/R_\odot]$

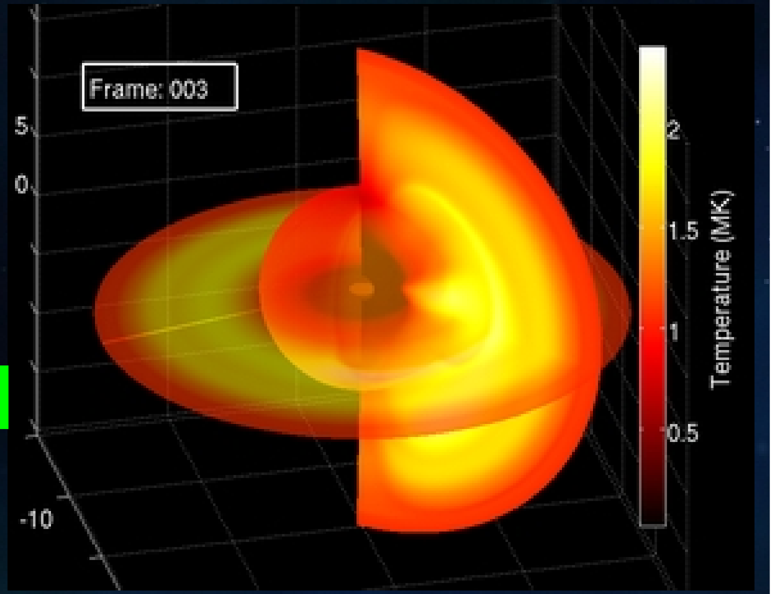
$S = (\Delta t^2 \hat{k}^2)^{-1}\,(C_w^2/(1 - C_f)^2 - 1)$, $\quad C_f = \Delta t\,\hat{k} \cdot \mathbf{v}$, $\quad C_w^2 = 0.25\,\Delta t^2\,\hat{k}^2\,(v_s^2 + |\mathbf{v_A}|^2)$, $\quad \hat{k}^2 = 4\,(\Delta r^{-2} + (r\Delta\theta)^{-2} + (r\Delta\phi\sin\theta)^{-2})$

Different components of the model are used depending on the domain and use-case

Test Cases

Test 1:

Modified low-res test case based on simulations used in **[Reeves et. al. (2019)]**

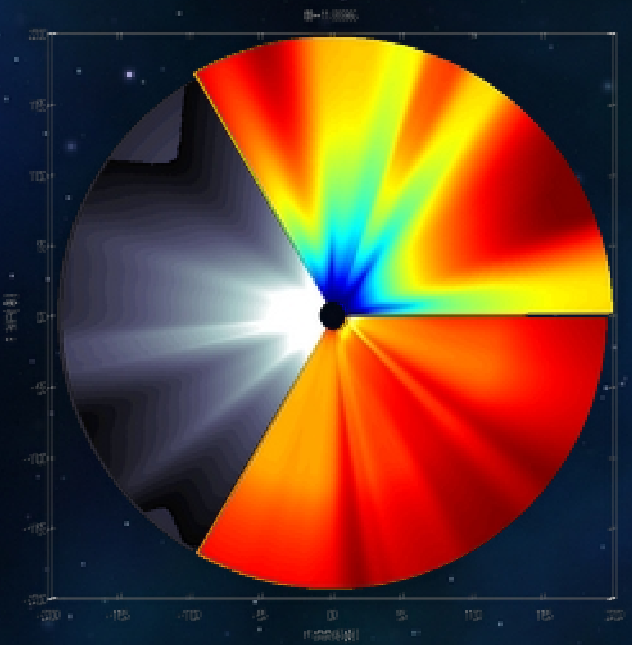


Resolution:
151x151x151
~ 3.4 million points

Thermodynamic MHD relaxation

Test 2:

Modified test case used in Astronum 2016 paper **[Caplan et. al. (2017)]**

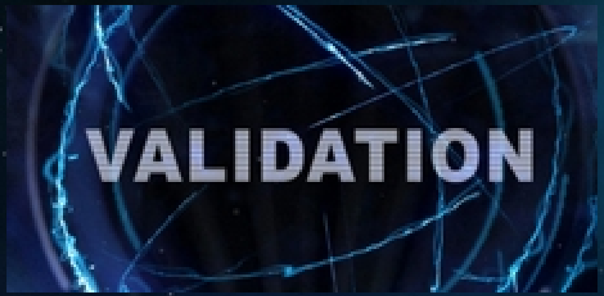


Resolution:
181x251x502
~ 22.8 million points

Thermodynamic MHD relaxation

Validation runs:

Integrate relaxation for ~8 simulation-hours



Scaling runs (Test 2 only):

Integrate for 6 simulation-minutes starting with the ~8 simulation-hour relaxation (subtract restart loading time)

Testing Procedure

Questions to address:

- ❑ **Can we get a solution as good (or better) as BE+PCG with RKG2 if we automatically outer-cycle at the practical time step limit?**
- ❑ **Can we get a "better" solution with BE+PCG if we outer-cycle it?**
- ❑ **How does outer-cycling affect performance?**
- ❑ **Is RKG2 competitive with BE+PCG?**

Tests:

SC1: 1 outer cycle (original)

SCA: Automatic adaptive outer cycles (practical time step)

BE+PCG	RKG2
SC1	SC1
SCA	SCA

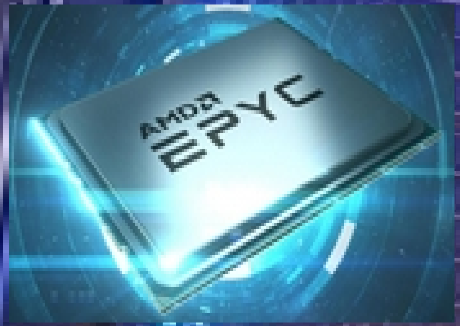
Computational Environment

- Tests performed on both CPUs and GPUs
- On CPUs, the PC2 preconditioner is used for BE+PCG; on GPUs, PC1 is used
- On CPUs, the gfortran compiler is used with OpenMPI 4, on GPUs, the NV compiler is used with OpenMPI 3

In-house workstation with NVIDIA RTX 3090 Ti

CPU	Core i5-13600KF
GPU	RTX 3090 Ti
Peak DP FLOPs	0.625 TFLOP/s
Memory	24 GB
Memory Bandwidth	1008 GB/s

EXPANSE
COMPUTING WITHOUT BOUNDARIES
SDSC
SAN DIEGO SUPERCOMPUTER CENTER

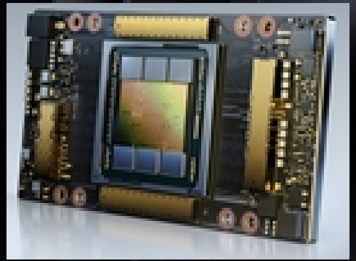


EPYC ROME 7742

SDSC Expanse 2xCPU Node

# CPUs x Model	(2x) EPYC 7742
# Total Cores	128 *(we use 64)*
Peak FLOP/s	7.0 TFLOP/s
Memory	256 GB
Total Memory Bandwidth	381.4 GB/s

DELTA
I NCSA



A100 40GB

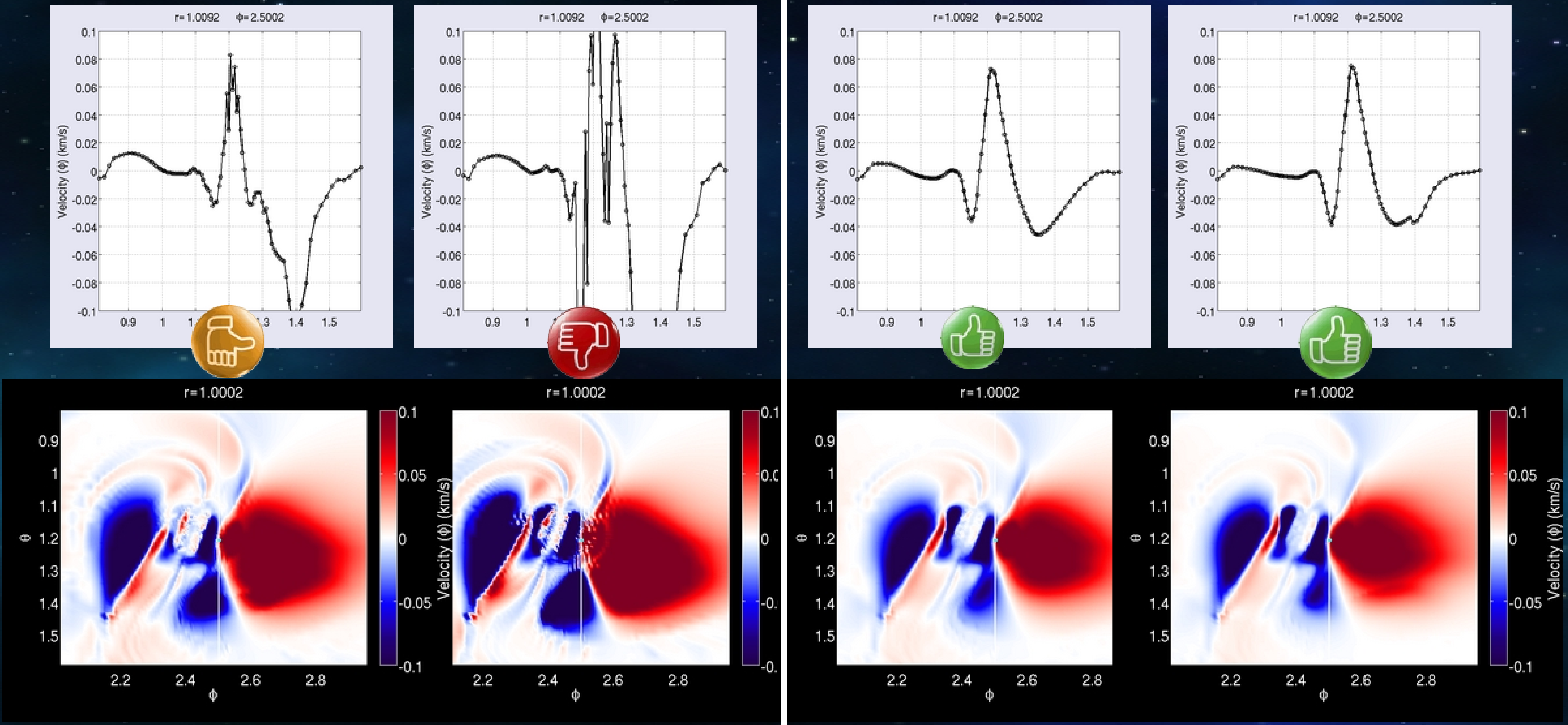
NCSA Delta 8xGPU Node

# CPUs x Model	(2x) EPYC 7742
# GPUs x Model	8x A100-40GB SXM4
Peak DP FLOP/s / GPU	9.8 TFLOP/s
Memory / GPU	40 GB
Memory Bandwidth/GPU	1555 GB/s

Allocations provided by: ACCESS NSF

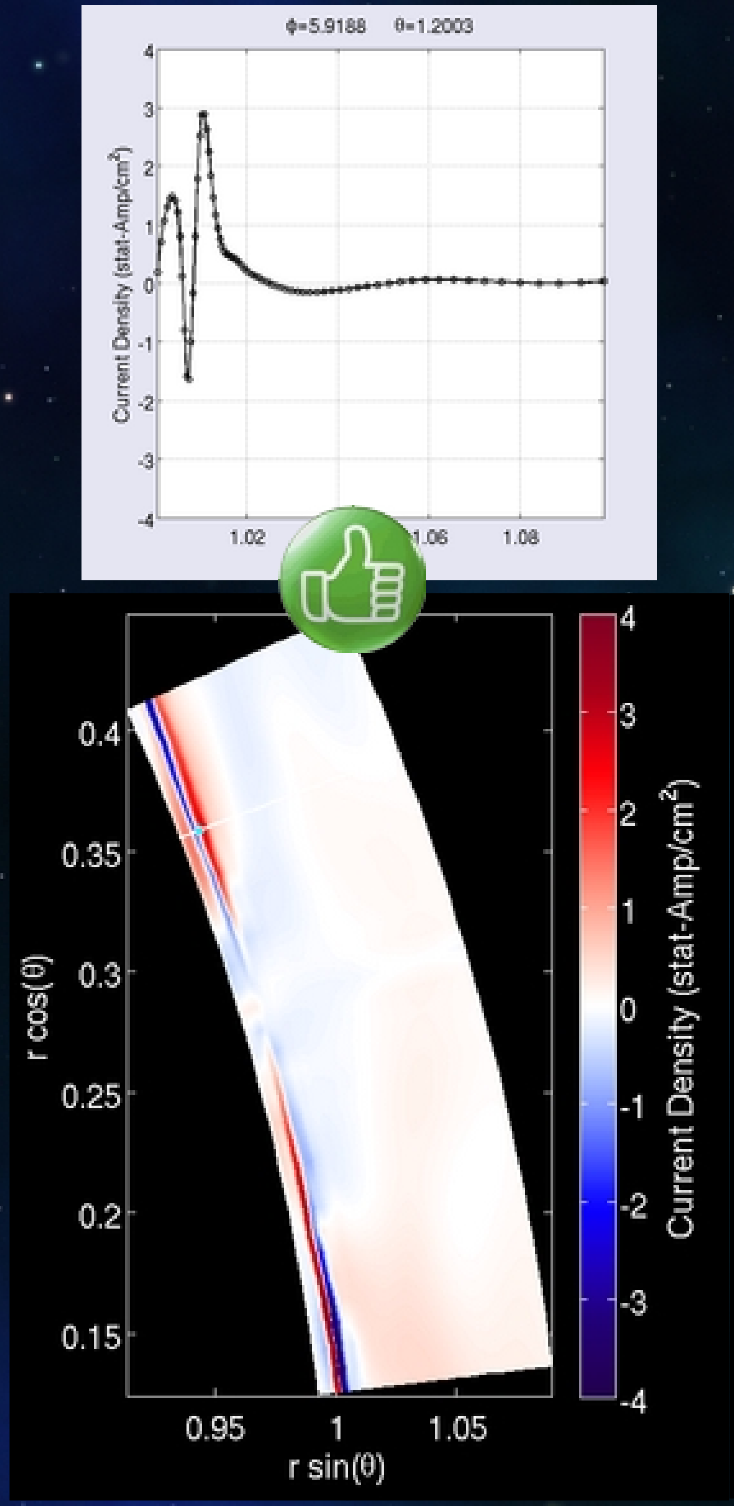
Validation Results Test 1

BE+PCG SC1 **RKG2 SC1** **BE+PCG SCA** **RKG2 SCA**

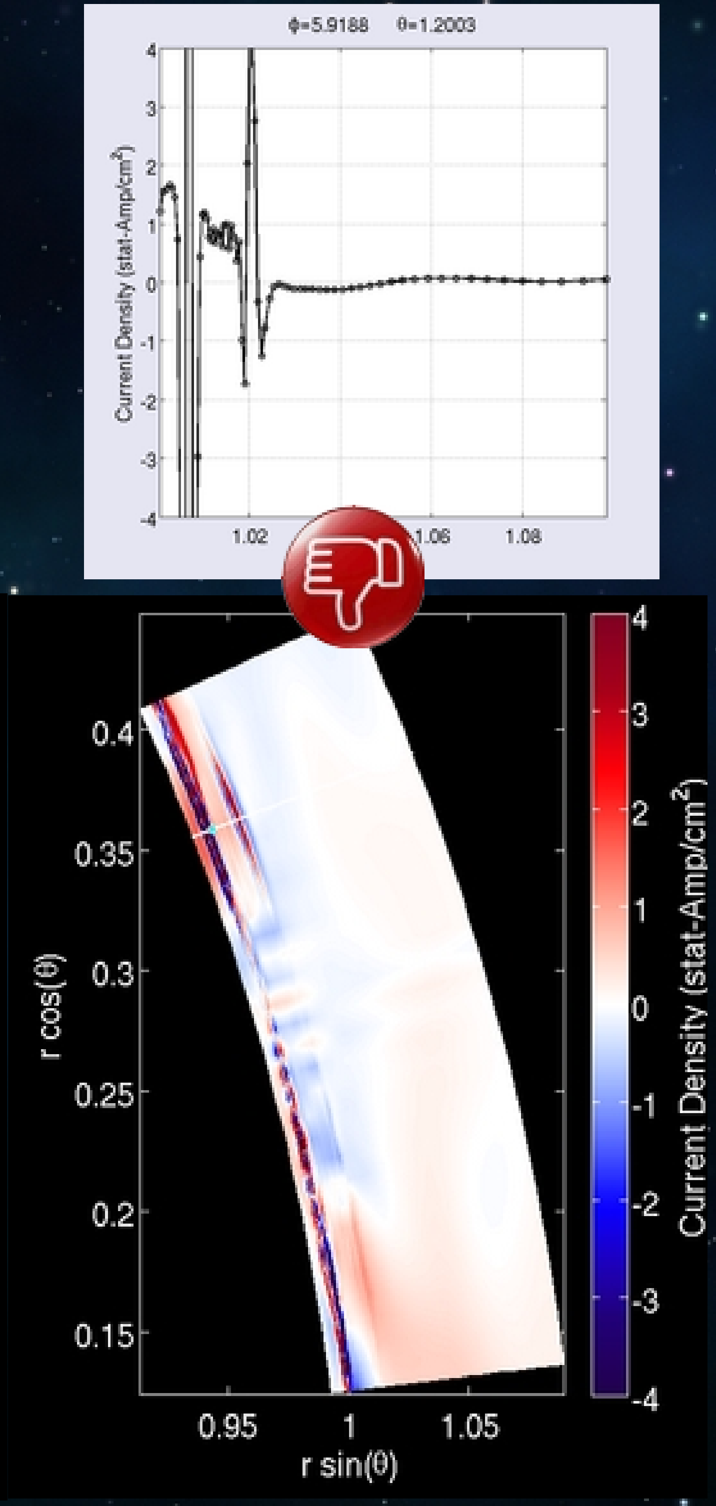


Validation Results Test 2

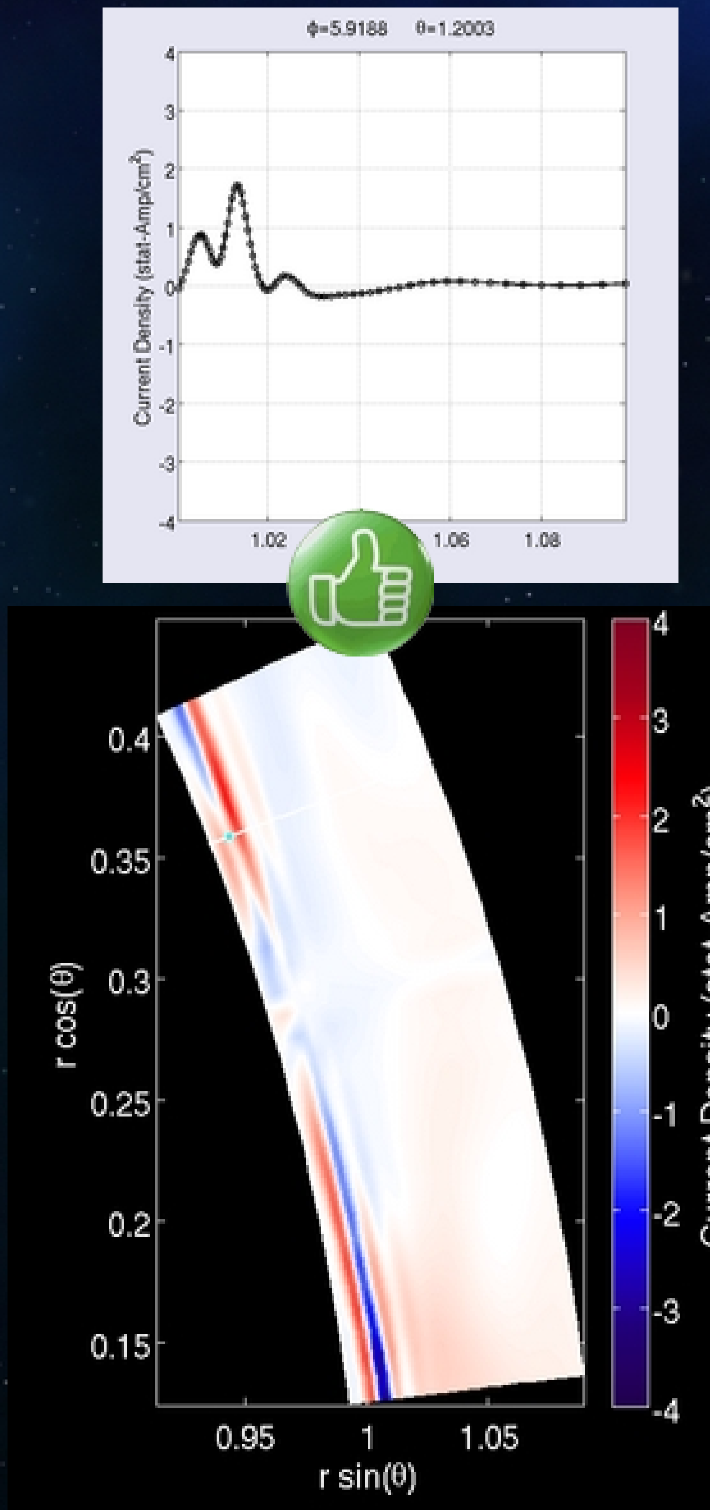
BE+PCG SC1



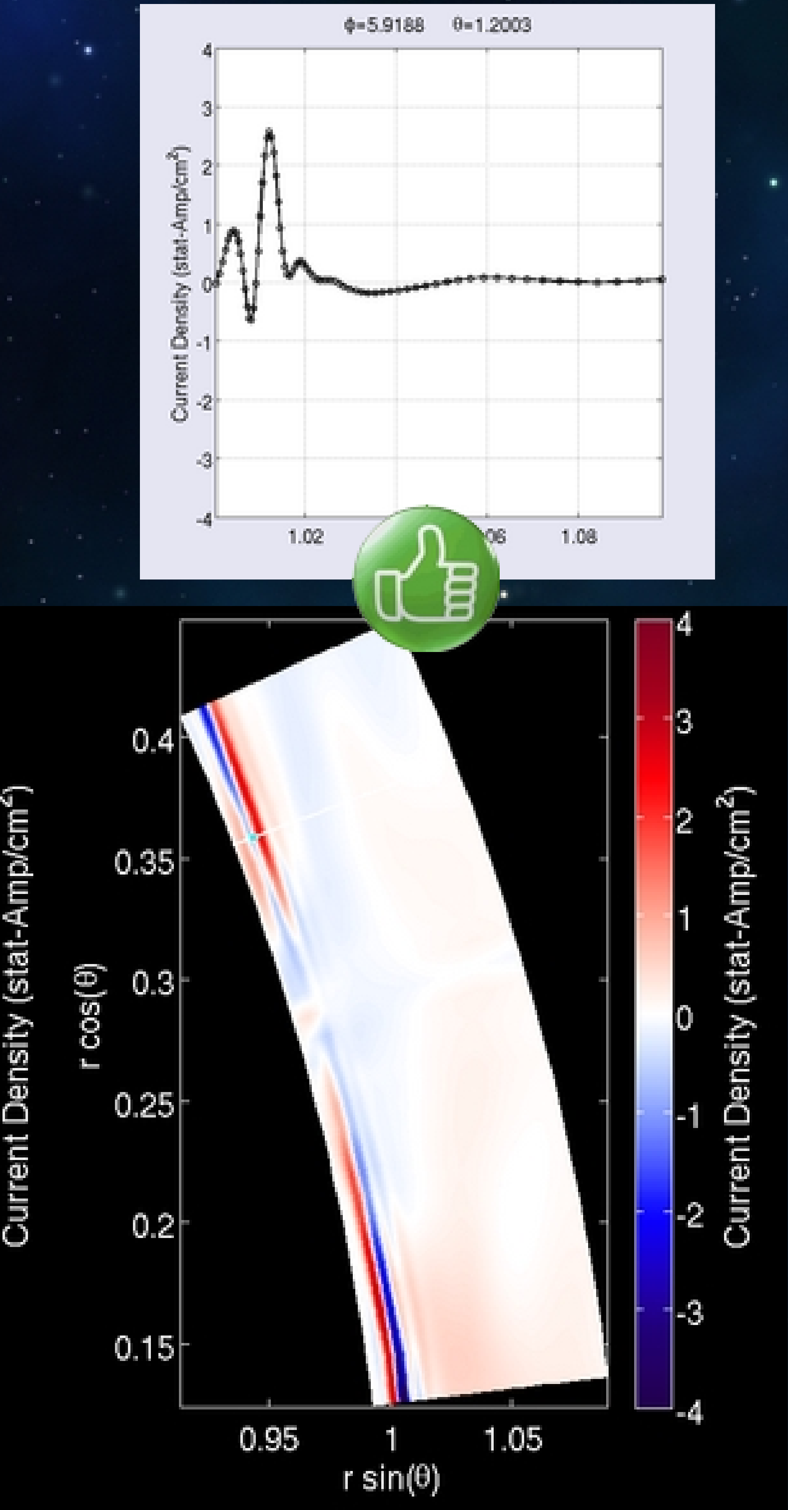
RKG2 SC1



BE+PCG SCA

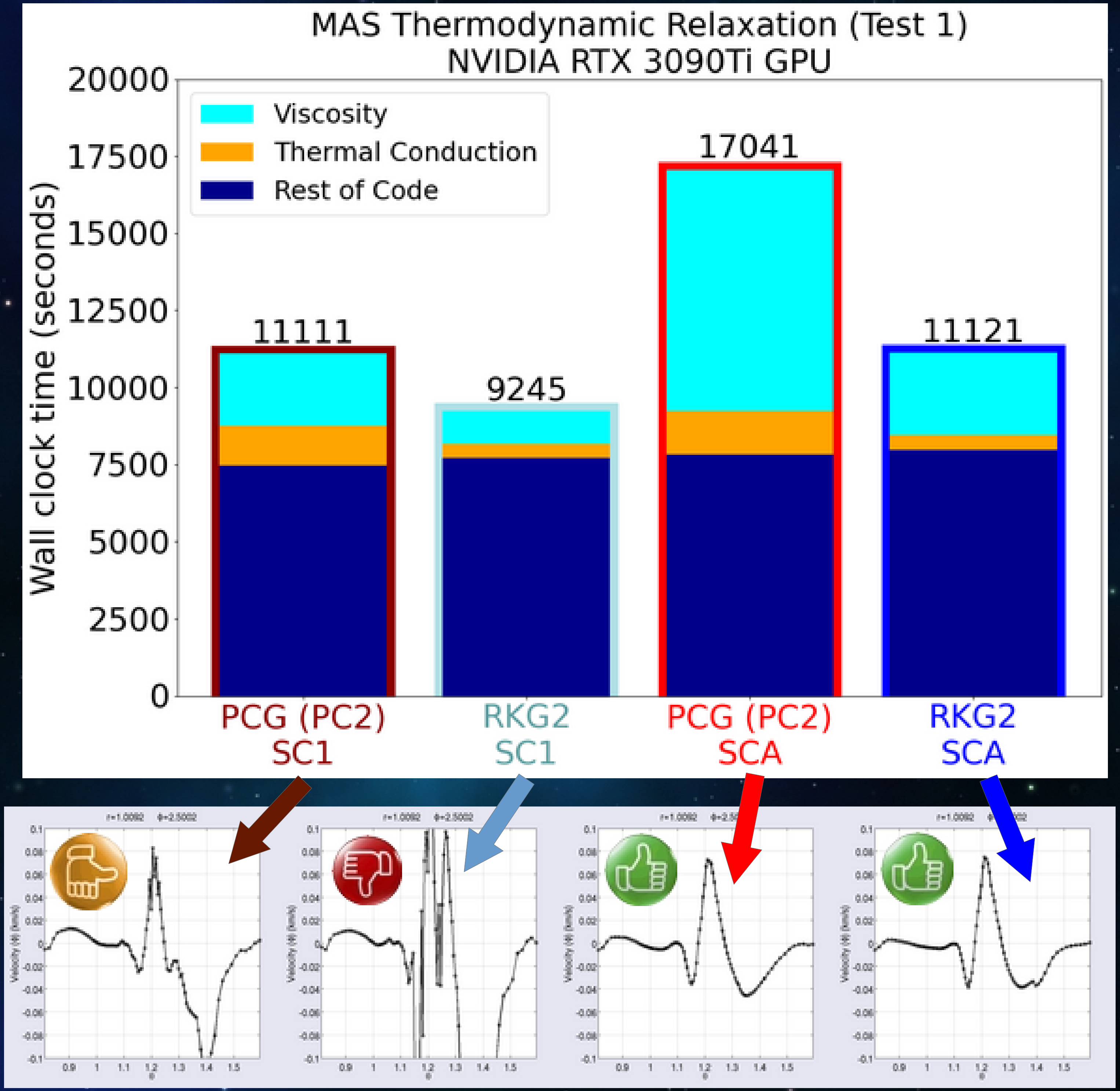


RKG2 SCA

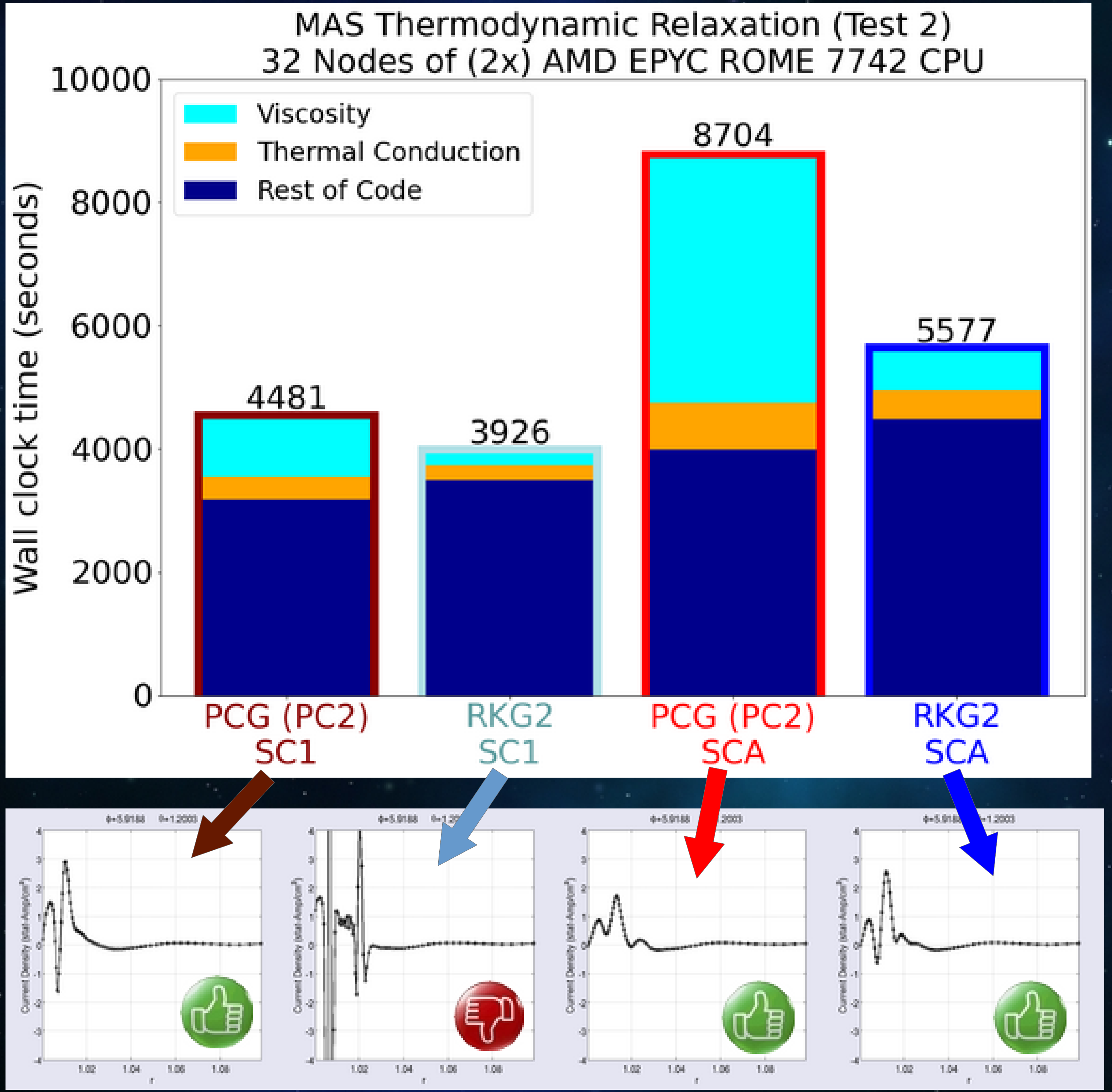


Validation Timing Results

Test 1:

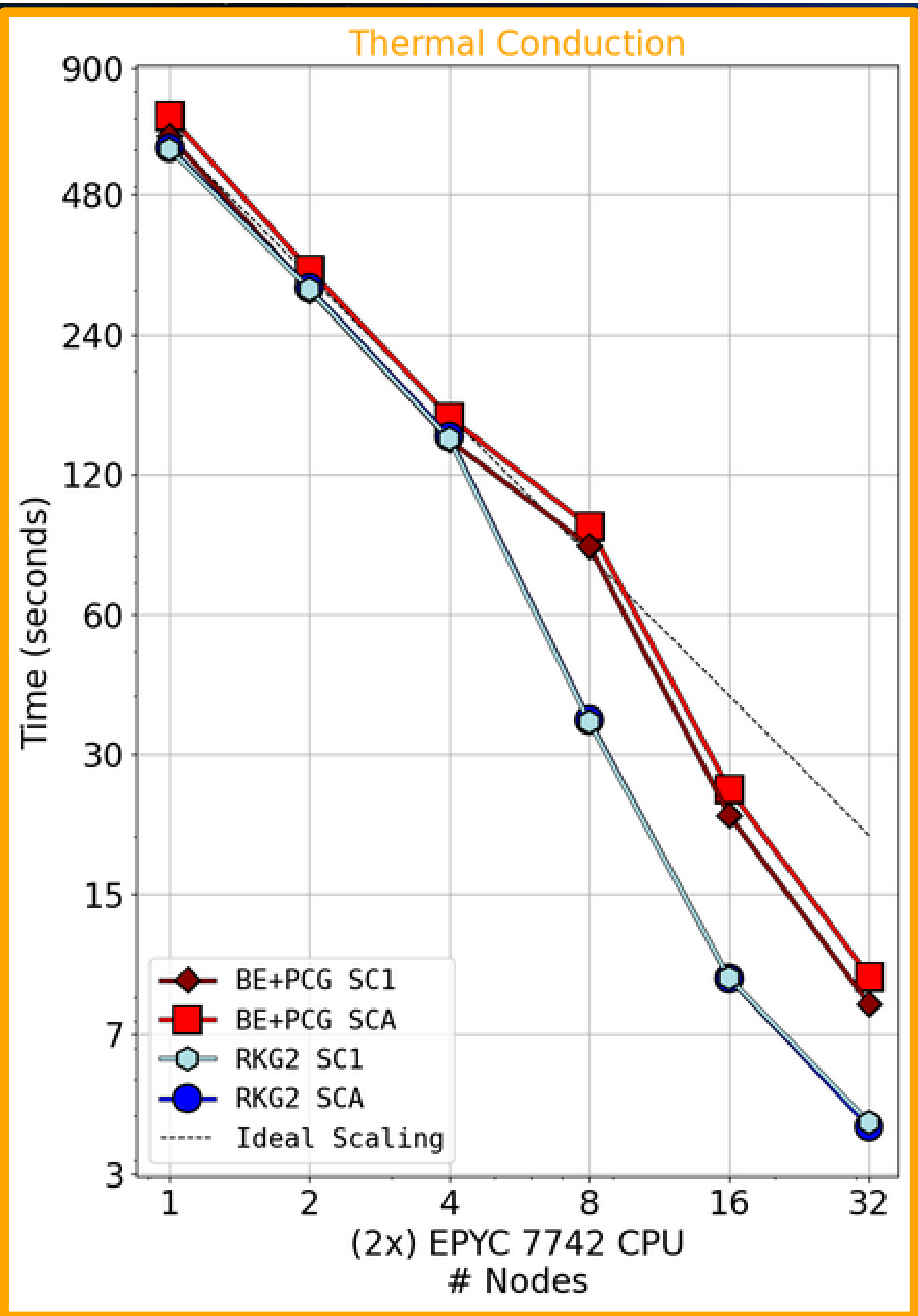


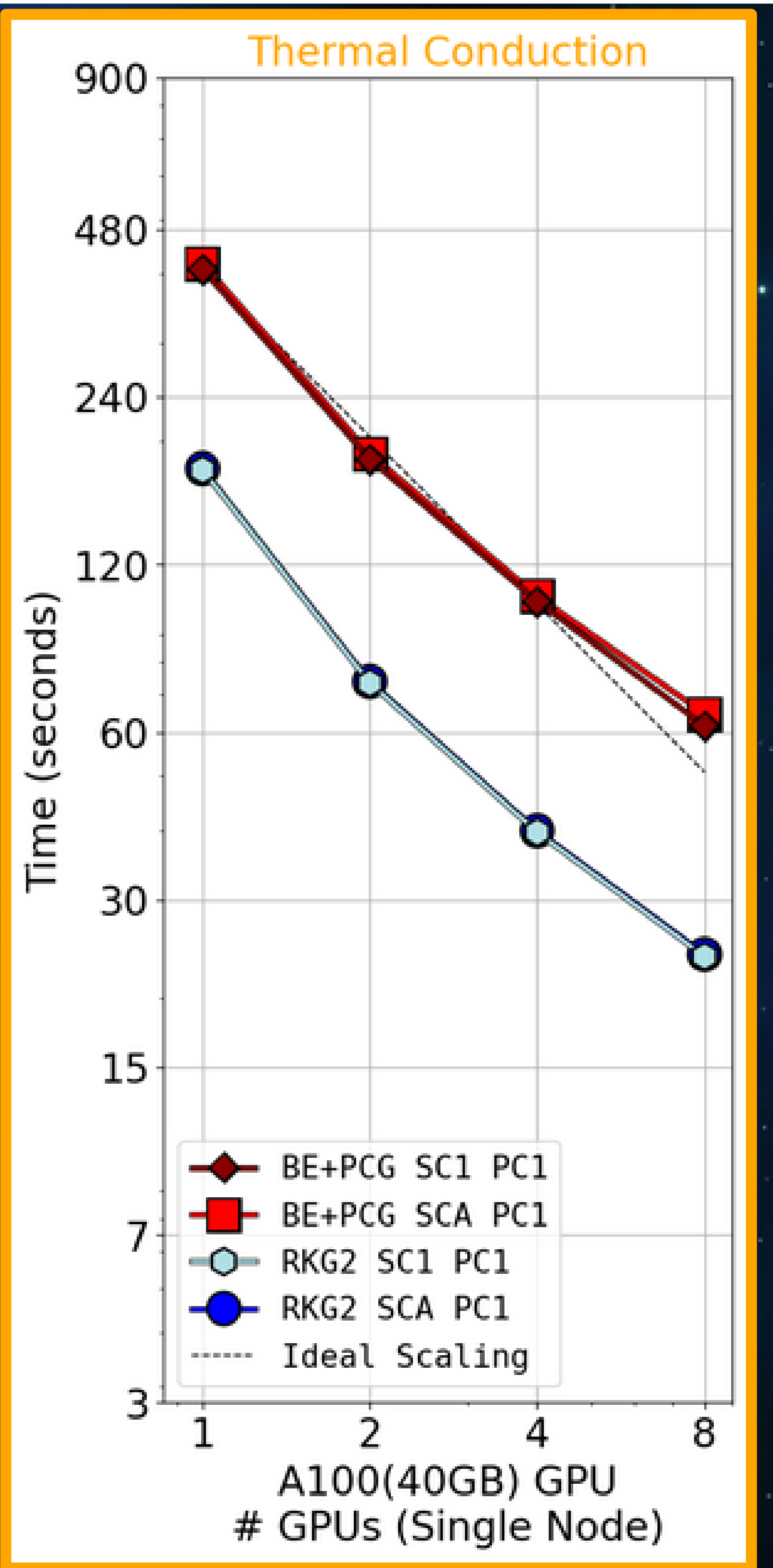
Test 2:



Scaling Results Test 2 Thermal Conduction

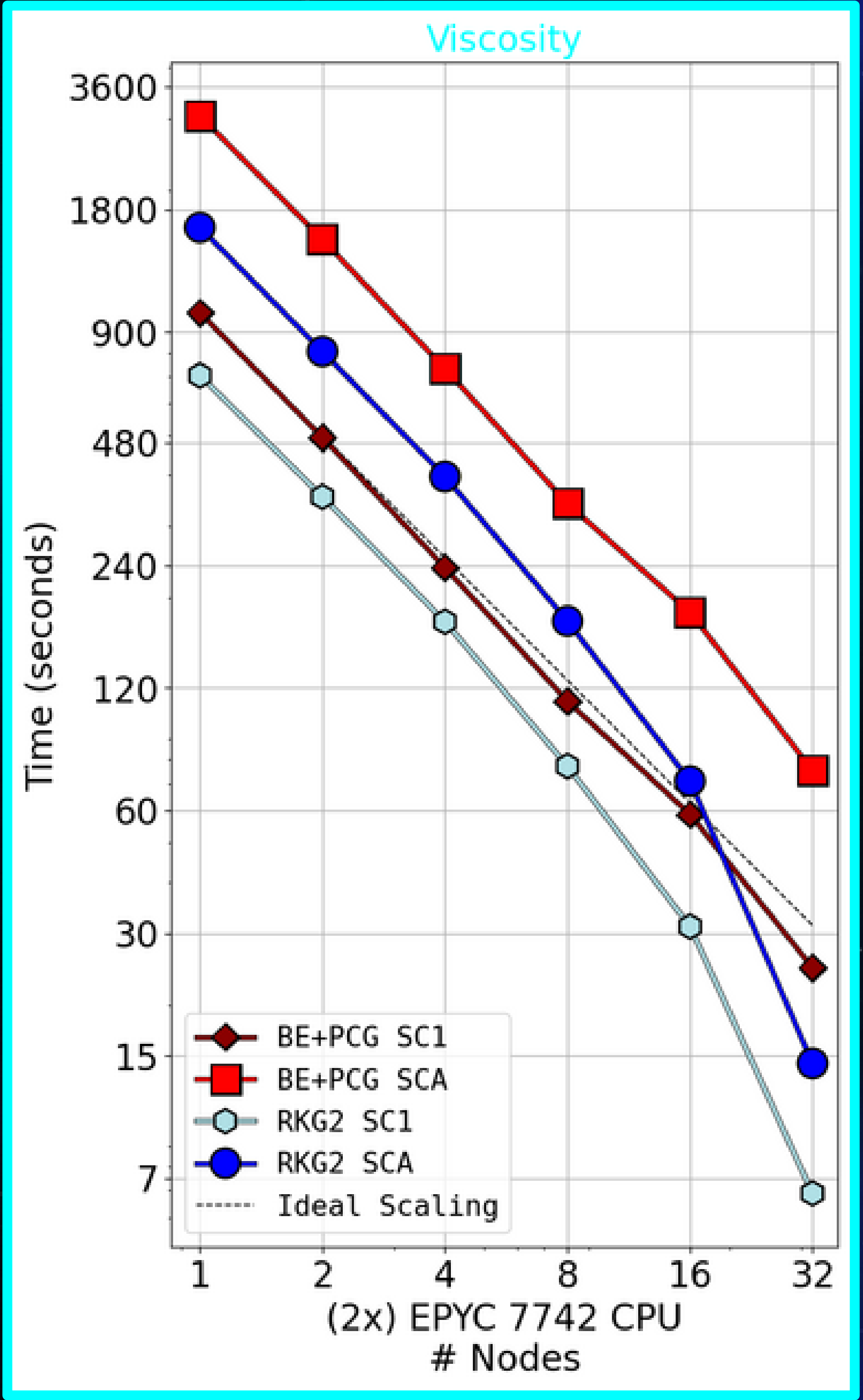
- Very few outer cycles were needed in this test for TC
- Therefore, SCA has similar run times as SC1
- Scaling is better with RKG2 than BE+PCG (PC2) on CPUs
- Run time for RKG2 is much faster than BE+PCG (PC1) on GPUs
- Scaling of both schemes similar on multi-GPU single server runs

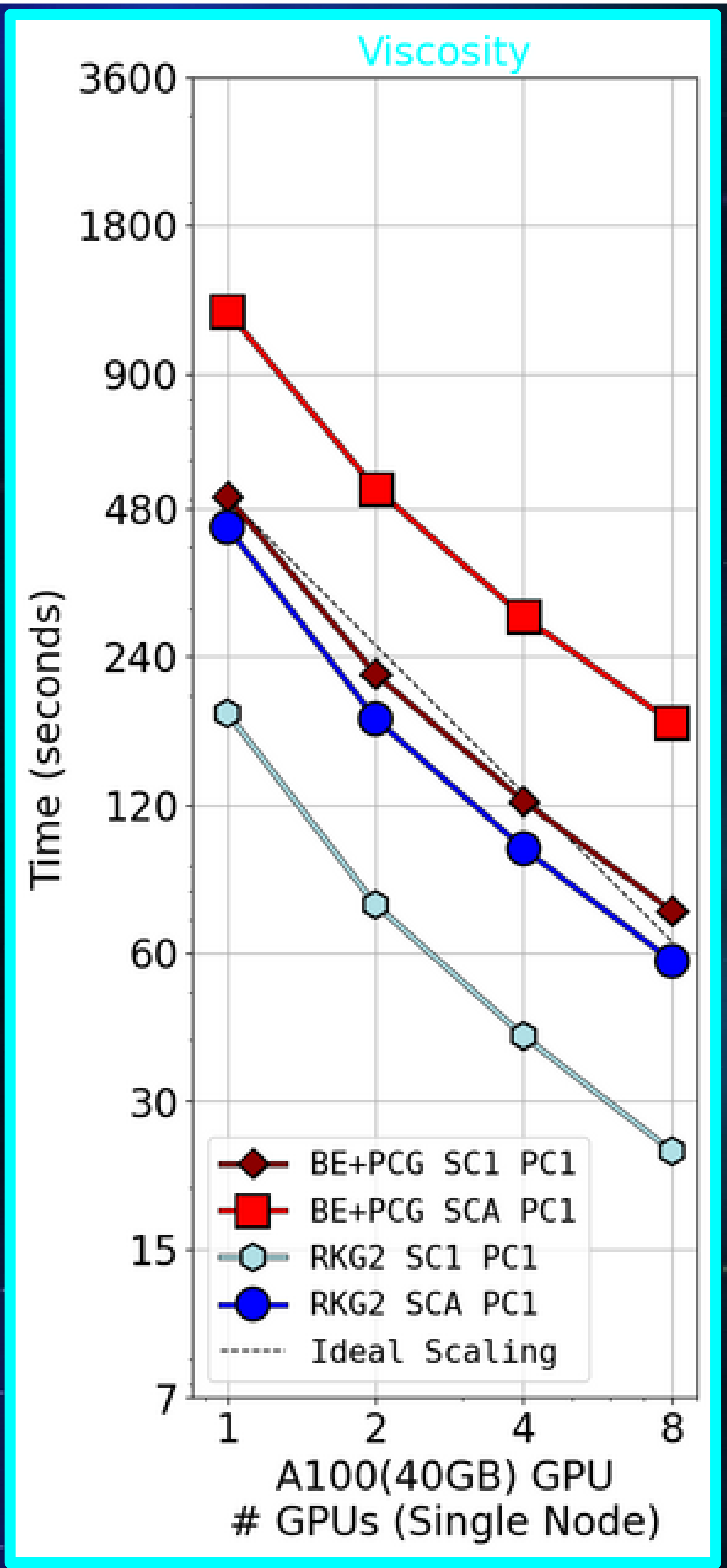




Scaling Results Test 2 Viscosity

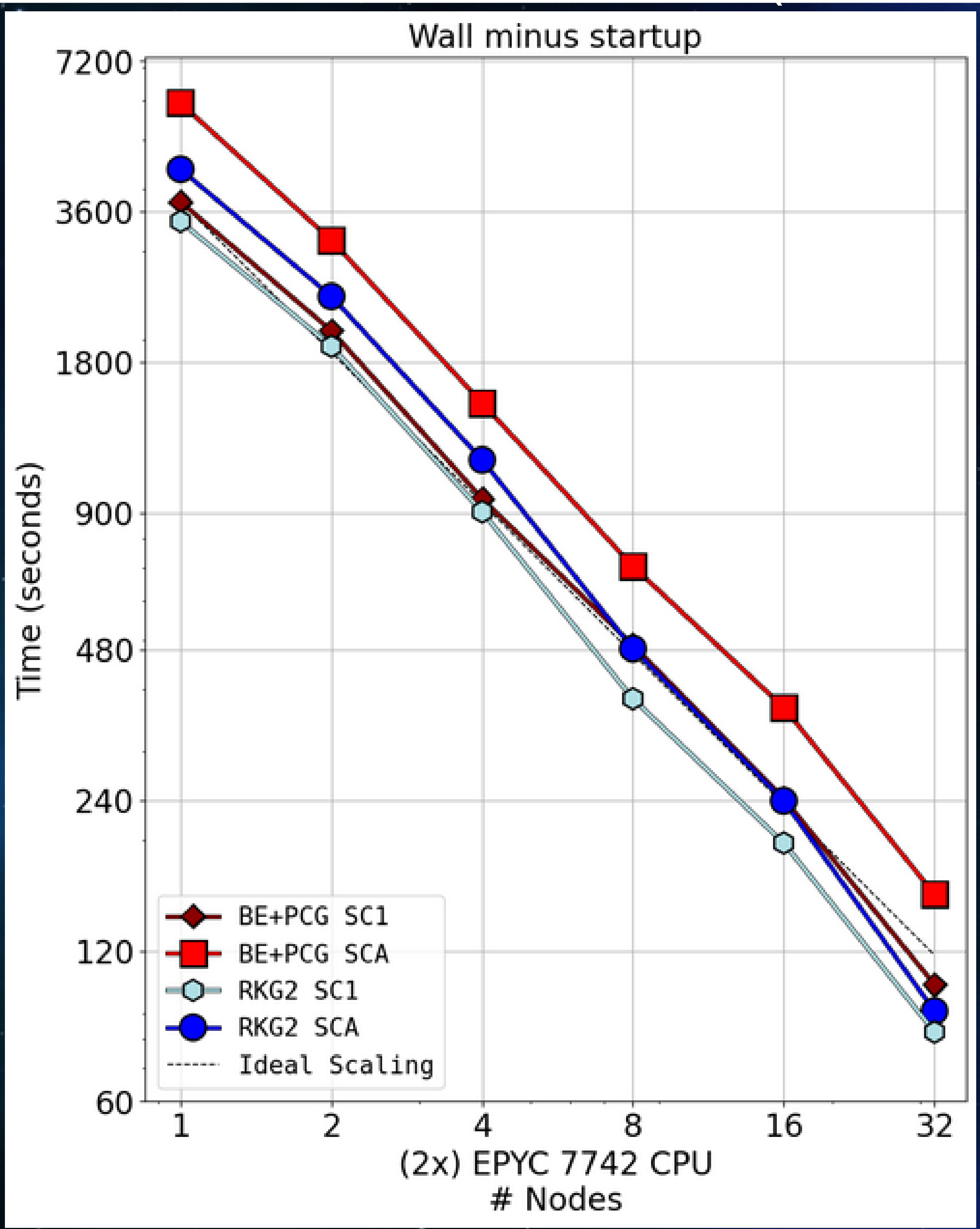
- The test problem required ~10 outer cycles for viscosity
- Therefore, auto-cycle has slower run time than single cycle
- Scaling is better with RKG2 than PCG (PC2) on CPUs
- Run time for RKG2 is faster than PCG (PC1) on GPUs with similar scaling
- RKG2 auto-cycle faster than PCG single cycle:
  - With PCG (PC1), always the case
  - With PCG (PC2), it’s faster with maximum CPUs due to better scaling and PC2 effectiveness decline

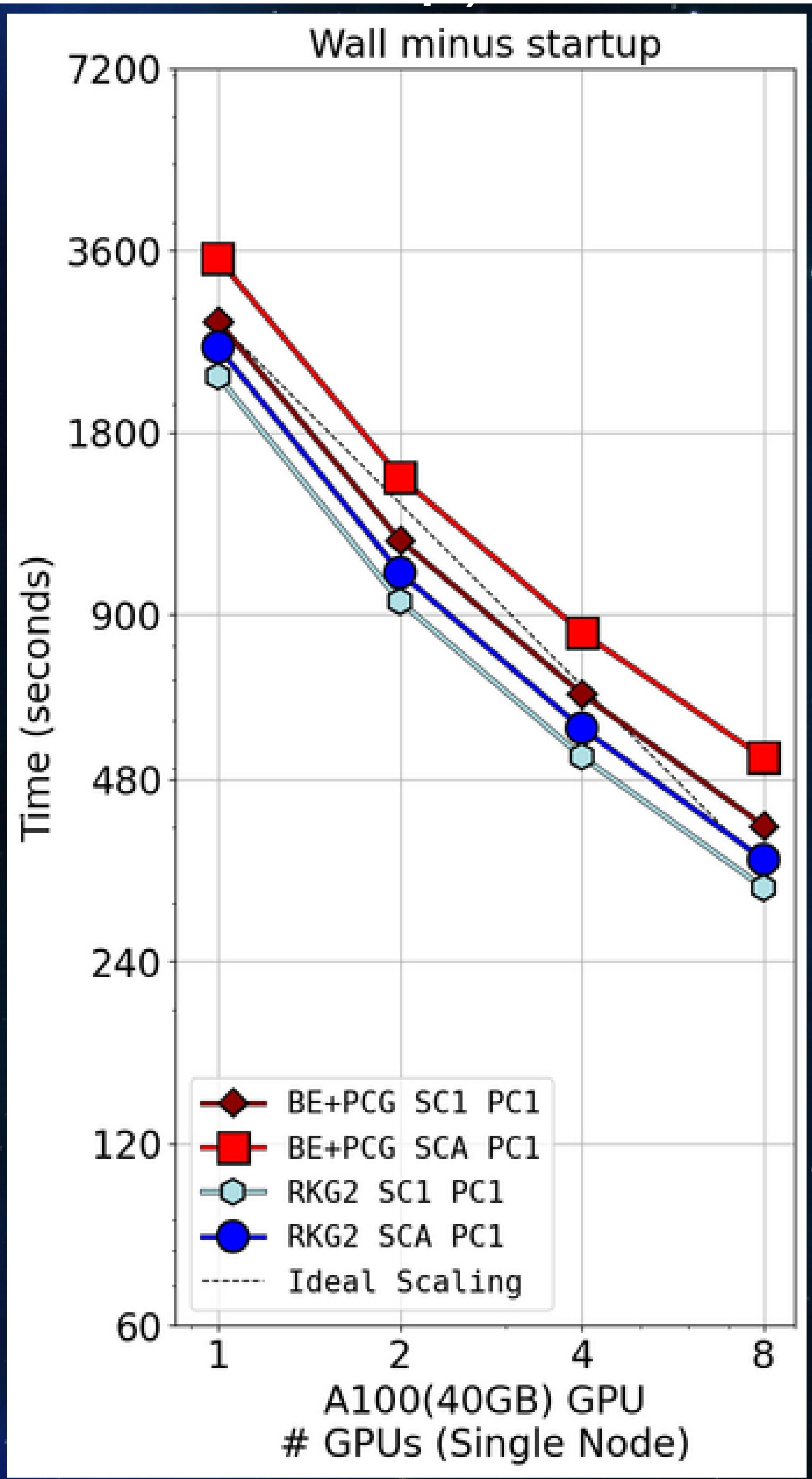




Scaling Results Test 2 Total Wall Clock (minus restart startup)

- RKG2 auto-cycle has comparable or better performance and scaling than BE+PCG single-cycle
- BE+PCG auto-cycle significantly slower than single-cycle

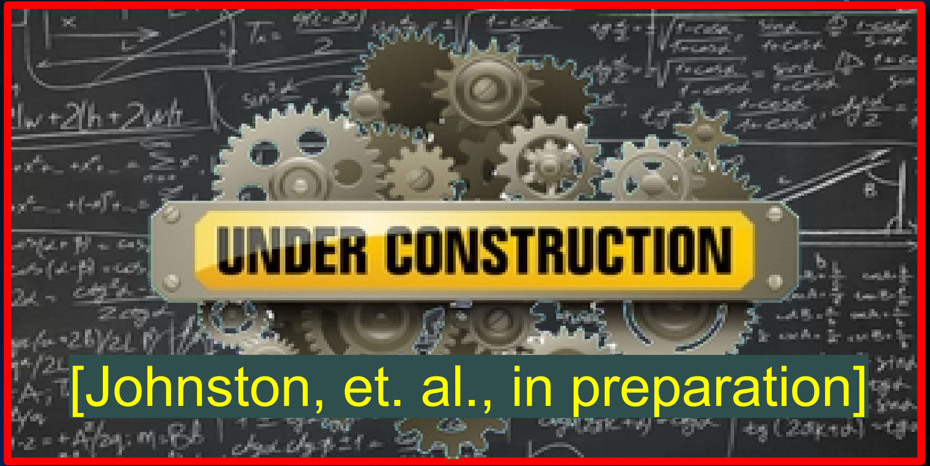




Results

- Can we get a solution as good (or better) as BE+PCG with RKG2 if we automatically outer-cycle at the practical time step limit?
  - **Yes (in our cases)**
- Can we get a "better" solution with BE+PCG if we outer-cycle it?
  - **Yes (in our cases)**
- How does outer-cycling affect performance?
  - **For BE+PCG, significant decrease in performance (in our cases)**
    For RKG2, small decrease in performance (in our cases)
- Is RKG2 competitive with BE+PCG?
  - **Yes (in our cases)**

How general are these results???



[Johnston, et. al., in preparation]

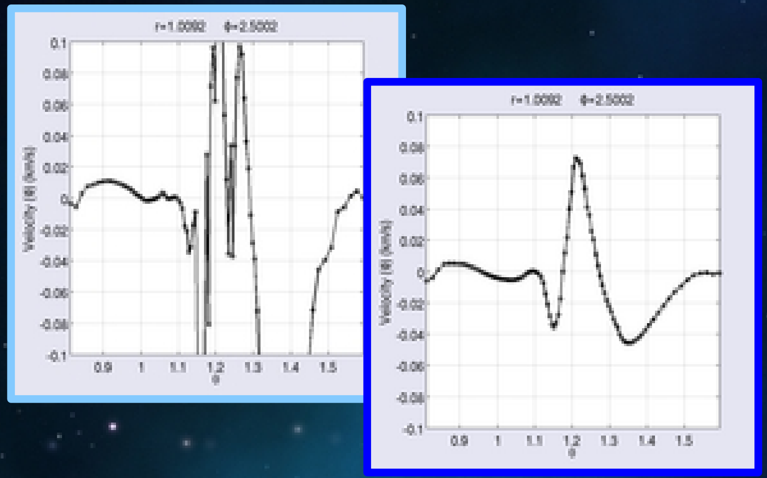
Summary

- Unconditionally stable methods are often necessary when conditionally stable schemes' time-step limits for operators (e.g. parabolic) are too restrictive for practical simulations
- Using "too large" of a time step can create large errors including qualitative changes in the operator's effect and/or inability to damp high wave modes (oscillations)

$$\Delta t_p = \alpha \frac{|u_k|}{|F(u_k)|}$$

$$k : |F(u_k)| = \max |F(u)|$$

- We have tested an easy-to-calculate practical time step limit for such schemes that can be applied as an operator-split outer cycle
- The new limit can help fix known solution issues in extended stability Runge-Kutta (super time stepping) methods that are due to their poor amplification factors at high wave modes



- Testing the method with RKG2 implemented in the MAS MHD code recovered the solution accuracy of the BE+PCG method with negligible effect on performance & scaling
- Work is proceeding on the theoretical formulation/modification of the practical time step, including careful testing, especially in nonlinear cases [Johnston et al, in preparation]