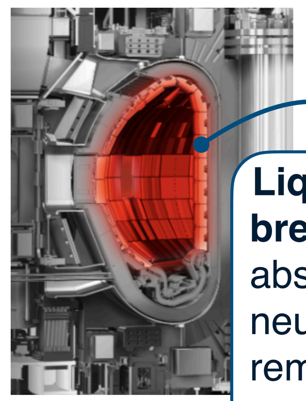
GPU-Accelerated Linear Solvers for High-Order Finite Element Methods

Yichen Guo[1], Noel Chalmers[2], Eric de Sturler[1], Tim Warburton[1]

[1]Virginia Tech, [2]AMD

Overview

1. Motivation: liquid metal breeder blankets [1]

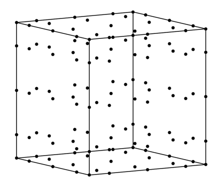


Liquid metal breeder blanket absorbs fusion neutrons, removes heat, and produces tritium fuel for the reactor.

2. Magnetohydrodynamics equations

$\mathbf{u}$: velocity field $\mathbf{B}$: magnetic field

$$
\begin{aligned}
\frac{\partial \mathbf{u}}{\partial t} - \frac{1}{Re}\nabla^2 \mathbf{u} + \nabla p &= \mathbf{F}(\mathbf{u},\mathbf{B}) \\
\frac{\partial \mathbf{B}}{\partial t} - \frac{1}{Rm}\nabla\cdot r_w \nabla \mathbf{B} + \nabla q &= \mathbf{G}(\mathbf{u},\mathbf{B}) \\
\nabla\cdot\mathbf{u} &= 0 \\
\nabla\cdot\mathbf{B} &= 0.
\end{aligned}
$$



High-order hex element

3. High-order Finite Element Discretization

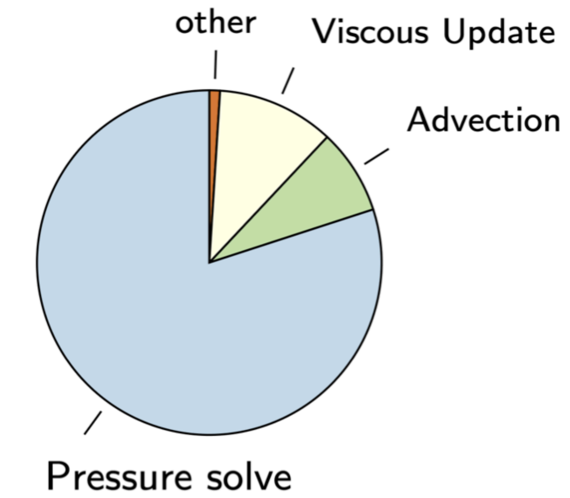
$\mathbf{u}$ and $\mathbf{B}$ are approximated by piecewise polynomials, satisfying the weak form:

$$
\begin{aligned}
\left(\frac{\partial \mathbf{u}_h}{\partial t}, \mathbf{v}\right) + \frac{1}{Re}(\nabla \mathbf{u}_h, \nabla \mathbf{v}) + (\nabla p, \mathbf{v}) &= (\mathbf{F}, \mathbf{v}) \\
\left(\frac{\partial \mathbf{B}_h}{\partial t}, \phi\right) + \frac{1}{Rm}(r_w \nabla \mathbf{B}_h, \nabla \phi) + (\nabla q, \phi) &= (\mathbf{G}, \phi) \\
(\nabla\cdot\mathbf{u}_h, \mathbf{w}) = 0, \quad (\nabla\cdot\mathbf{B}_h, \psi) &= 0.
\end{aligned}
$$

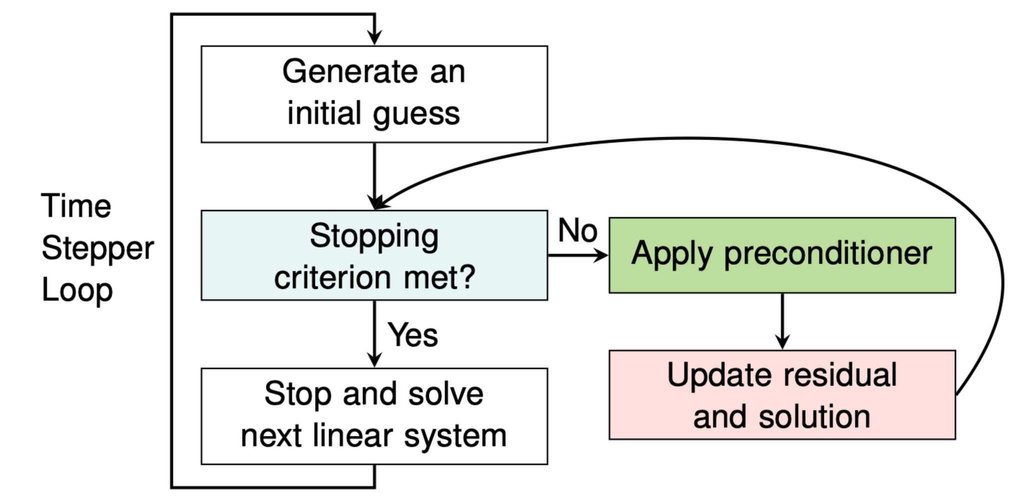
↓

Solve a sequence of linear systems

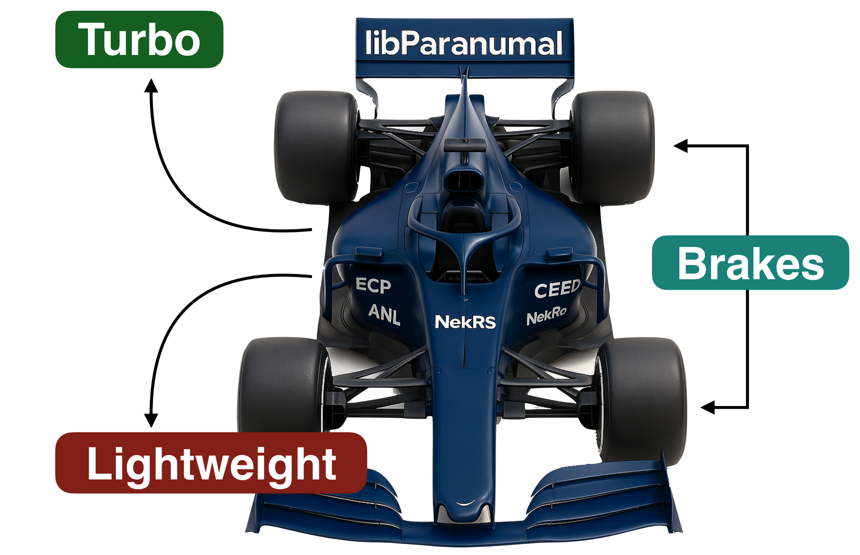
4. Linear solves take 80% of time on Summits's 27k GPUs [2]



5. Iterative Linear Solvers



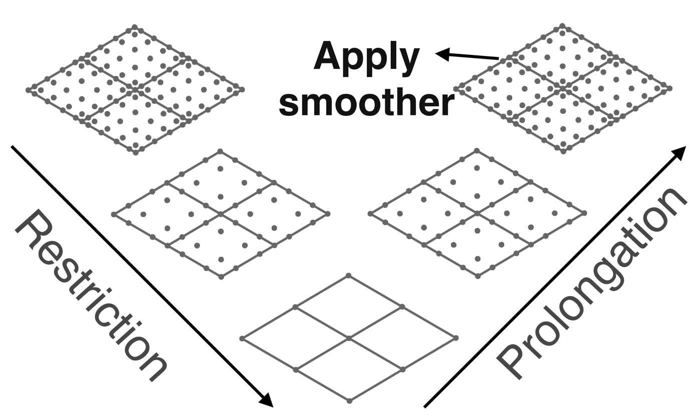
6. Accelerate Linear Solvers



Turbo: Preconditioner

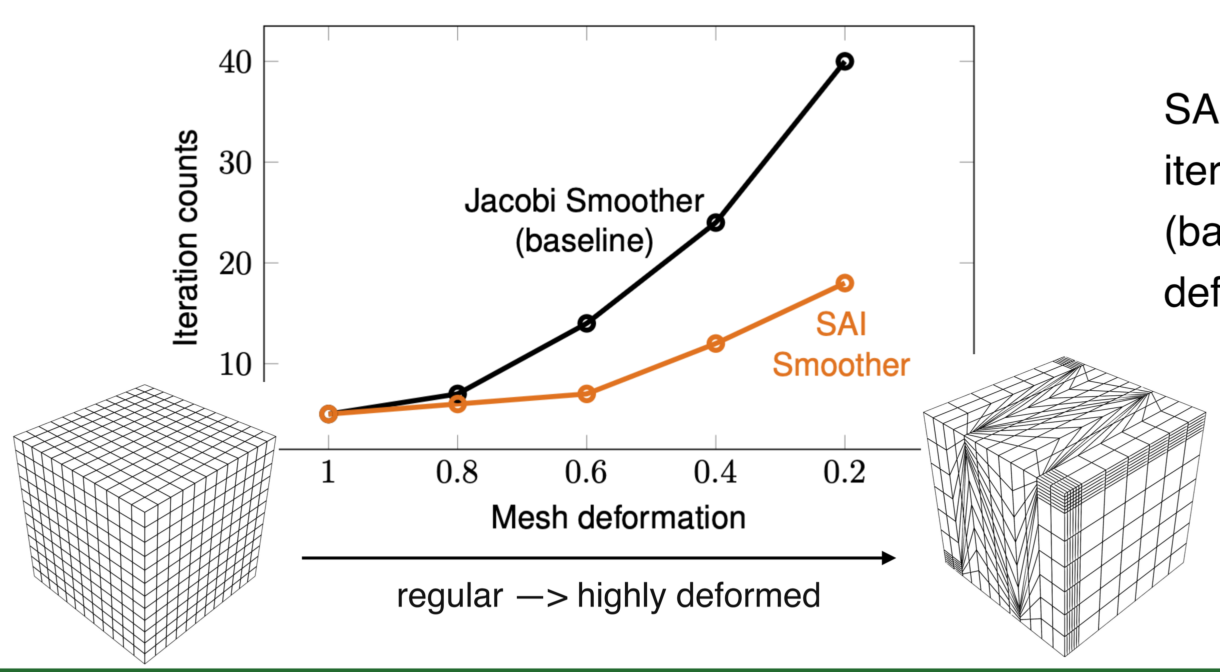
p-multigrid preconditioner [3] can improve rate of convergence

Sparse Approximate Inverse smoother [4] (SAI)



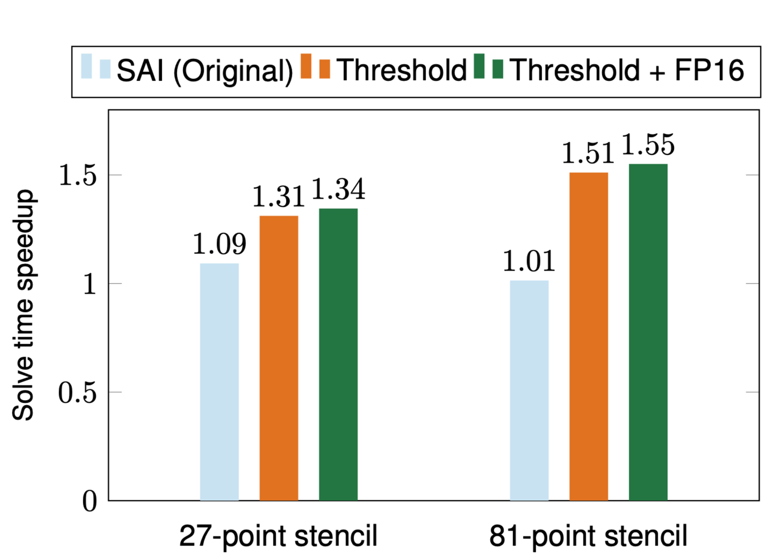
- Improves convergence of iterative solvers
- Matrix-free construction
- Thresholding SAI smoother matrix
- Custom FP16 + INT16 CSR compression

Comparison of Iteration Counts



SAI saves **50%** iterations over Jacobi (baseline) for highly deformed meshes.

Comparison of Solve Times



- Test case: highly deformed mesh
- ⭐ Larger stencil + thresholding + half precision → most efficient
- ⭐ Longer setup, faster solves
- Pays off when solving a sequence of linear systems

Lightweight: Mixed-Precision Solver

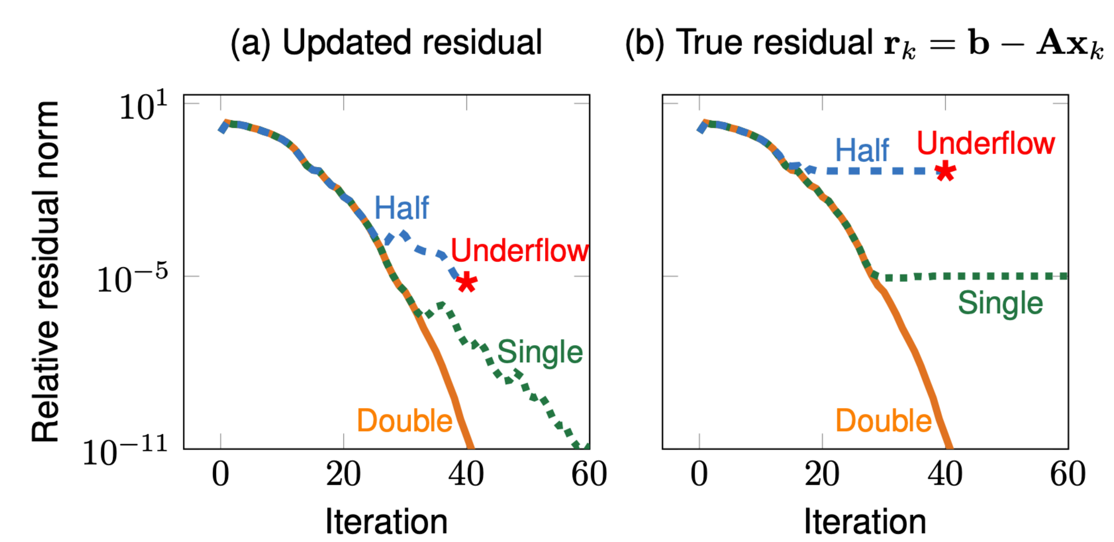
Compute kernels are **memory-bound**:

- Matrix-vector mult.
- Inner product
- Vector operations

↓

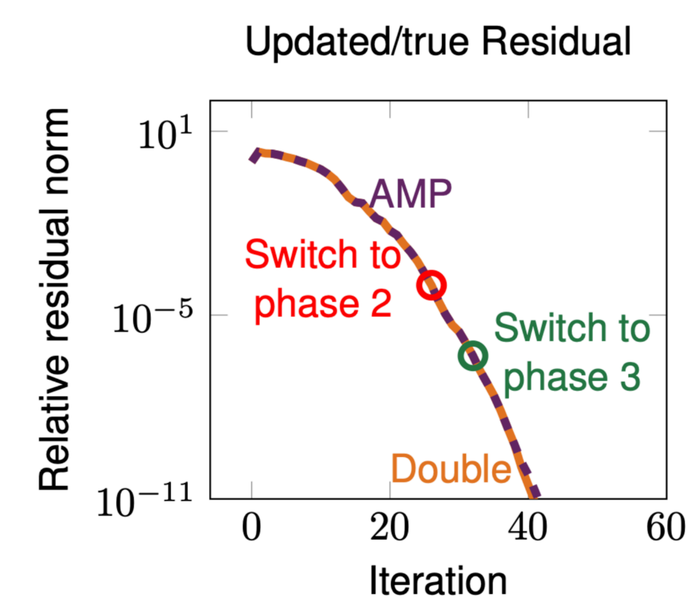
Using low precision can reduce data transfer [5].

Naive low-precision implementation leads to stagnant convergence & data underflow.

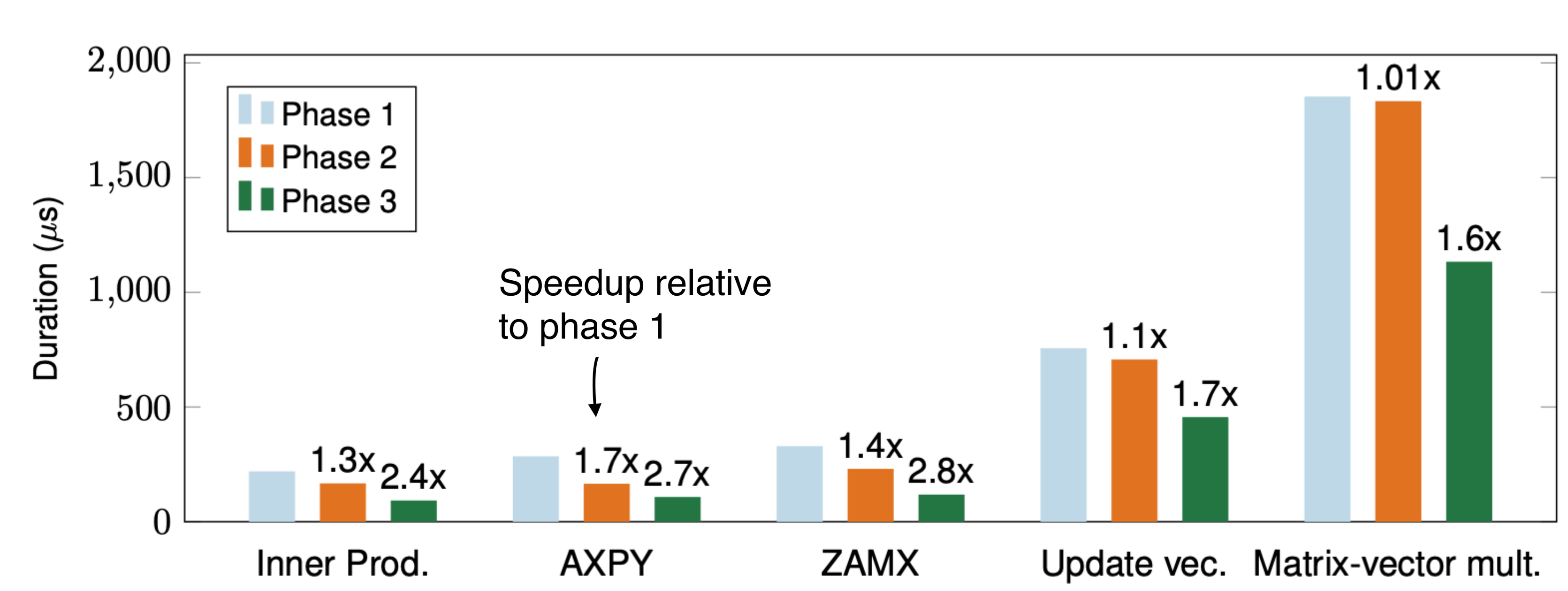


Remedy: Adaptive Mixed-Precision Iterative Methods (AMP)

- Dynamically adjust precision during iterations.
- Vectors start in double, then transition to single and half based on switching criteria.
- ⭐ Maintain convergence comparable to double precision implementation.

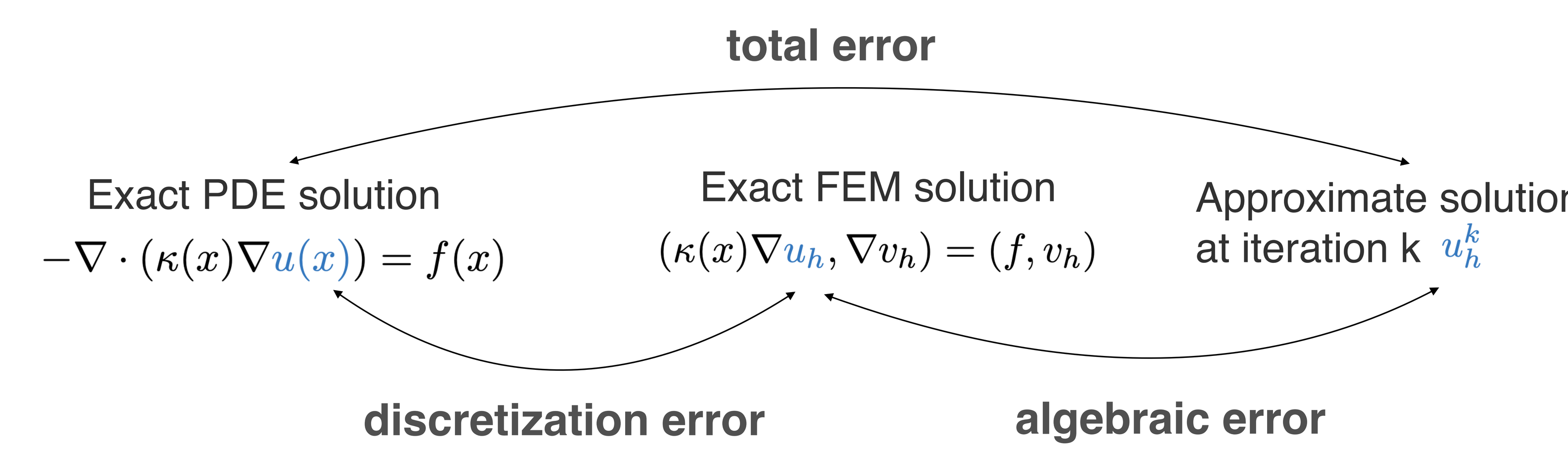


Comparison of main kernel runtimes across Phases 1-3



Brakes: Stopping Criteria

Total error has two sources: discretization and algebraic errors



Stopping criterion that balances *discretization error* and *algebraic error.*

The n-th component of the linear system residual $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ is

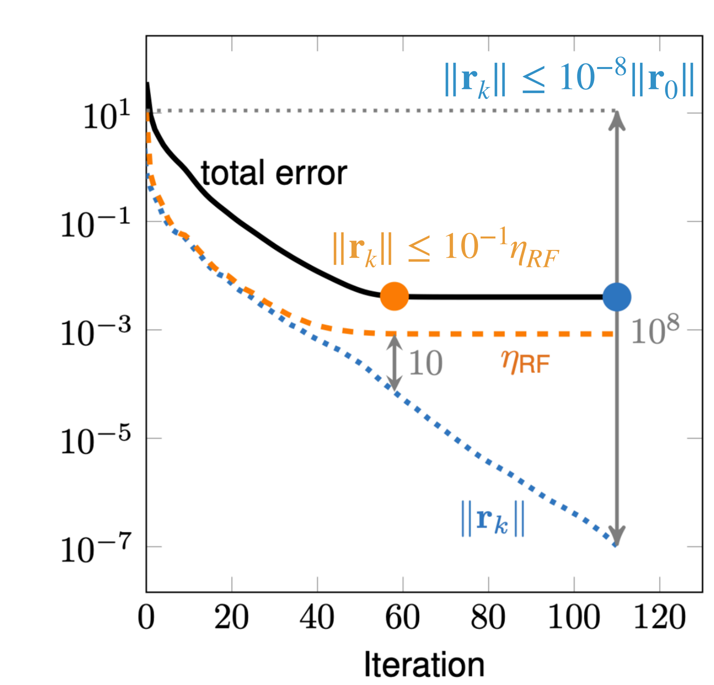
$$
\begin{aligned}
(\mathbf{r}_k)_n &= \mathbf{b}_n - (\mathbf{A}\mathbf{x}_k)_n \\
&= \sum_{K\in\mathcal{T}}\left(\phi_n, f + \nabla\cdot\left(\kappa(x)\nabla u_h^k\right)\right)_K - \sum_{\ell\in\mathcal{E}}\left(\phi_n, \left[\left(\kappa(x)\nabla u_h^k\right)\cdot\mathbf{n}_\ell\right]\right)_\ell \\
&= (\mathbf{R}_k)_n - (\mathbf{R}_k - \mathbf{r}_k)_n.
\end{aligned}
$$

$$\eta_{\mathrm{RF}} := \|\mathbf{R}_k\|_{\ell^2,\kappa} + \|\mathbf{R}_k - \mathbf{r}_k\|_{\ell^2,\kappa}$$

IF $\|\mathbf{r}_k\|_{\ell^2,\kappa} \le \tau\,\eta_{\mathrm{RF}}$ THEN STOP.

algebraic error estimate ⟶ discretization error estimate

Comparison of Performance



Criterion	Iter.	Runtime(s)	Error
$\|\mathbf{r}_k\| \le 10^{-8}\|\mathbf{r}_0\|$	111	3.25	4.0×10^{-3}
$\|\mathbf{r}_k\| \le \tau\,\eta_{\mathrm{RF}}$	58	1.85	4.2×10^{-3}
Ratio	1.91	1.75	0.96

- ⭐ Efficient termination with maintained accuracy
- Calculating η_{RF} introduces ~10% overhead per iteration

[1] iter, https://www.iter.org/machine
[2] M. Min et al., Optimization of full-core reactor simulations on summit, 2022
[3] H. Sundar et al., Comparison of multigrid algorithms for high-order continuous FE discretizations, 2015
[4] W.P. Tang et al., Sparse approximate inverse smoother for multigrid, 2000.
[5] H. Nicholas et al., Mixed precision algorithms in numerical linear algebra, 2022

John K. Costain Graduate Fellowship

Check out our paper: Y. Guo et al., Stopping Criteria of the CG Algorithm for High-order Finite Element Methods, 2025