xlim([0.92 1])

GPU acceleration of geometric multigrid solver in PARALLAX

A. Stegmeir*,1, C. Lalescu[2], M. Lin[2] and T. Dannert[2]
[1]Max-Planck-Institut für Plasmaphysik
[2]Max Planck Computing and Data Facility

MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK

ABSTRACT

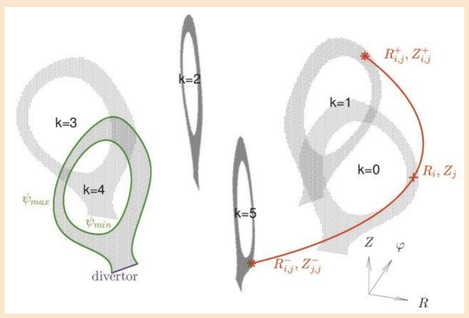
Elliptic equations play a crucial role in turbulence models for magnetic confinement fusion. Regardless of the chosen modeling approach – whether gyrokinetic, gyrofluid, or drift-fluid – the Poisson equation and Ampere's law lead to elliptic problems that must be solved on 2D planes perpendicular to the magnetic field. In this work, we present an efficient solver for such elliptic problems, discretized using finite difference methods. The solver is based on a flexible generalized minimal residual method (fGMRES) with a geometric multigrid preconditioner. We present implementations with OpenMP parallelization and GPU acceleration, with backends in CUDA and HIP. On the node level, significant speed-ups are achieved with the GPU implementation, outperforming standard library solutions such as PETSc. In accordance with theoretical scaling laws for multigrid methods, we observe near-linear scaling of the solver with problem size, O(N). This solver is integrated into the PARALLAX/PAccX libraries and serves as a central component of the boundary codes GRILLIX and GENE-X, where the latter aims for simulations at exascale.

INTRODUCTION / MOTIVATION

- **Field equations are ubiquitous in plasma physics (Poisson, Ampere's law).**
- **In full-f models for edge/SOL codes, the generalised version needs to be solved.**

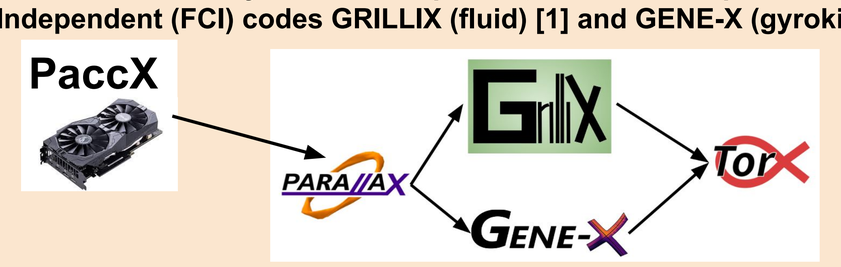
$$\lambda\phi - \nabla\cdot(c\nabla_\perp\phi) = \rho \qquad \begin{aligned}\phi &= \phi_D \quad \text{on} \quad \partial\Omega_D \\ \mathbf{n}\cdot\nabla\phi &= \Gamma \quad \text{on} \quad \partial\Omega_N\end{aligned}$$

- **Can be reduced to set of independent 2D problem within poloidal planes.**

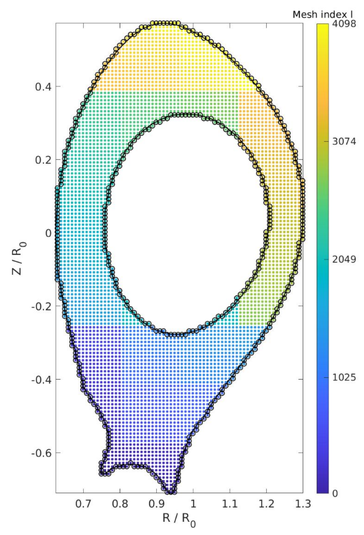


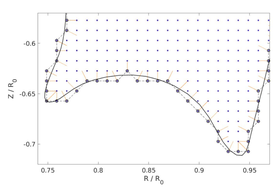
$$\lambda\phi - \xi\partial_R(c\partial_R\phi) - \xi\partial_Z(c\partial_Z\phi) = \rho$$

- PARALLAX library: Provides equilibra, mesh and elliptic solver for Flux-Coordinate Independent (FCI) codes GRILLIX (fluid) [1] and GENE-X (gyrokinetic) [2].



Discretisation





Mesh:

- Logically unstructured.
- Locally Cartesian.
- Guarded by boundary points meandering around continuous boundary.
- Mesh reordering possible to optimize cache performance.

Discretisation:

- Second order finite differences on inner grid points:

$$\begin{aligned}\lambda_{i,j}\phi_{i,j} - \frac{\xi_{i,j}}{2}[&\left(c_{i+1,j}+c_{i,j}\right)\phi_{i+1,j} + \left(c_{i,j}+c_{i-1,j}\right)\phi_{i-1,j} \\ &+\left(c_{i,j+1}+c_{i,j}\right)\phi_{i,j+1} + \left(c_{i,j-1}+c_{i,j}\right)\phi_{i,j-1} \\ &-\left(c_{i+1,j}+c_{i-1,j}+c_{i,j+1}+c_{i,j-1}+4c_{i,j}\right)\phi_{i,j}] = \rho_{i,j}\end{aligned}$$

- Boundary discretisation:

$$\phi_l = \phi_{D,l}, \quad \text{if} \quad (R_l, Z_l) \in \partial\Omega_D$$

$$\frac{\phi_{i,j}^N - \phi_{i,j}}{h\sqrt{\left(n_{i,j}^R\right)^2 + \left(n_{i,j}^Z\right)^2}} = \Gamma_{i,j}, \quad \text{if} \quad (R_l, Z_l) \in \partial\Omega_N$$

Properties of linear equation system:

$$\mathbf{Ax} = \mathbf{b}$$

- Sparse matrix with up to 5 entries per row.
- Matrix is time dependent due to dependency on coefficients.
- Non-zero structure remains fixed in time.
- Otherwise unstructured, non-zero structure can be optimised via mesh reordering.

Abstract solver design pattern:

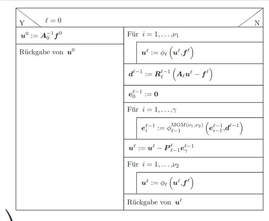
- **`create:`** Factory routine returning solver object. Sets up internal data structures and allocates memory.
- **`update:`** Updates internal data with new coefficients and boundary conditions.
- **`solve:`** Solves system for given right hand side and initial guess.

Only **`update`** and **`solve`** are called within time loop and are performance critical.

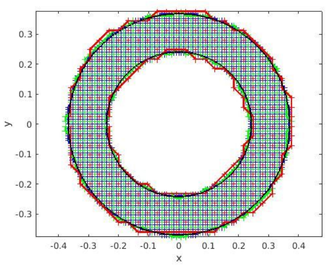
Geometric Multigrid

Solver:

- FGMRES with geometric multigrid preconditioner [3]
- Full multigrid V and W cycle available
- Direct solver (LU) on coarsest level

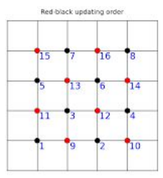


Restriction and prolongation:



- Weighted restriction: $R = \begin{pmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{pmatrix}$
- Prolongation: $P = \begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix}$

Smoother:



- Gauss-Seidel red black (boundaries)
- Shared memory parallelisation:
  - Update even points,
  - Update odd points
  - Update boundary points

The PARALLAX/PAccX solvers

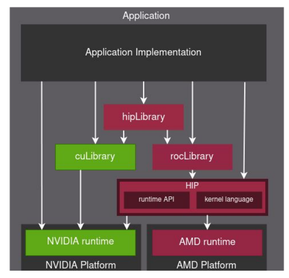
PARALLAX:

- Abstract solver interface, called by GRILLIX and GENE-X
- Fortran implementation of FGMRES solver with geometric multigrid precondidtioner
- OpenMP parallelised (CPU)

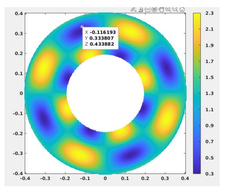
PaccX: PARALLAX Accelerator library

- Called from PARALLAX via C-bind interfaces
- Several solvers available
  - C++: CPU (yet serial)
  - CUDA: FGMRES + multigrid
  - HIP: FGMRES + multigrid
  - Rocalution: Black box library from AMD

NVIDIA CUDA

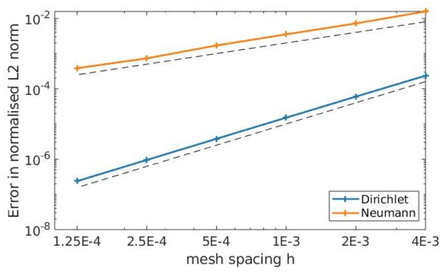


Verification



Procedure:

- Assess numerical solution for analytically prescribed problem.
- Circular flux surfaces, with local Cartesian mesh.
- Verify all backends with Dirichlet and Neumann (only at core) boundary conditions



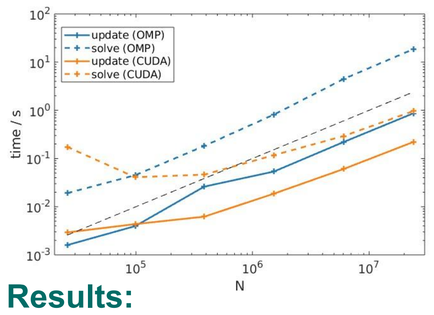
Results:

- Second order convergence for Dirichlet
- First order convergence for Neumann due to poor discretisation of boundary gradients
- All backends yield same results

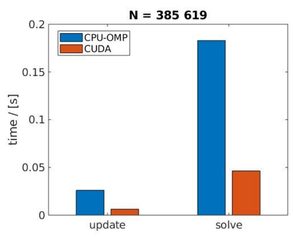
Performance and scaling

Setup of benchmark:

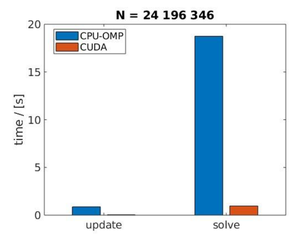
- Raven@MPCDF GPU node: 72 Intel Xeon IceLake-SP with 4x Nvidia A100-SXM4
- Compare Fortran OMP solver with 18 CPU threads vs. PaccX-Cuda solver with 1 GPU



small case: 5x speedup



large case: 20x speedup



Results:

- Textbook scaling O(N) for CPU-OMP solver.
- CUDA solver only approaches O(N) scaling towards large problem sizes.
- CUDA solver overall much faster, speedup to factor 20.
- Consistent results also with HIP on LUMI and VIPER.
- Integration into GRILLIX with MPI framework successful.

CONCLUSIONS / OUTLOOK

- **Solver for field equations is performance critical to GRILLIX and GENE-X.**
- **Geometric multigrid solver was ported to GPU (CUDA and HIP).**
- **Very good speedups up to factor 20 is obtained.**
- **Outperforms standard library solutions (e.g. PETSc).**
- **Towards Exascale: MPI parallelisation needed → Rocalution/GINKGO ?**

REFERENCES

[1] A. Stegmeir et al., Phys. Plasmas 26 (2019) 052517. doi:10.1063/1.5089864.
[2] D. Michels et al., Comput. Phys. Commun. 264 (2021) 107986. doi:10.1016/j.cpc.2021.107986.
[3] Andreas Meister, *Numerik linearer Gleichungssysteme*, Springer Spektrum Wiesbaden, ed. 5, (2015).

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

EUROfusion

Poster No.
TOK Retreat, Hirschberg, May 7-9, 2025

*Corresponding author:
Andreas.Stegmeir@ipp.mpg.de