Poisson Equation Solver Parallelisation for Particle-in-Cell Model

A. Podolník,[1,2] M. Komm,[1] R. Dejarnac,[1] J. P. Gunn[3]

[1] Institute of Plasma Physics ASCR, Prague, Czech Republic.
[2] Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic.
[3] CEA, IRFM, F-13108 Saint Paul Lez Durance, France.

Abstract. Numerical simulations based on PIC technique like the SPICE2 model developed at IPP ASCR are often used in tokamak plasma physics to investigate the interaction of edge plasma with plasma-facing components. The SPICE2 model has been parallelised with the exception of the Poisson equation solver which considerably slows down the simulations. It is now being upgraded to a parallelised version to be efficient enough to perform more demanding tasks like the ITER tokamak baseline scenario edge plasma whose conditions like high density (up to 10^{20} m^{-}3) and low temperature (1–2 eV) result in simulations taking several months to compute. Performance and scaling are compared for different cases in order to choose the optimal candidate for aforementioned applications.

Introduction

The basis of the particle-in-cell (PIC) technique is to divide the working field into a cell grid. The charge density ρ is then calculated and afterwards its spatial distribution is used in solving the Poisson equation

$$\Delta\phi = -\frac{\rho}{\epsilon_0}\,. \tag{1}$$

The resulting potential of the electric field ϕ, is then used for calculation of forces that affect moving particles. In the case of square grid with cell dimension h, the discretized equation (1) can be written as

$$\frac{\phi_{x-1,y} + \phi_{x+1,y} + \phi_{x,y-1} + \phi_{x,y+1} - 4\phi_{x,y}}{h^2} = -\frac{\rho_{x,y}}{\epsilon_0}\,, \tag{2}$$

with $\phi_{i,j}$ or $\rho_{i,j}$ describing the potential or charge density in the grid cell at position (i, j). This is a system of linear equations for $\phi_{i,j}$. It can be solved iteratively or using a direct solver. The system matrix is sparse, therefore suitable linear algebra software can be used.

One of such models, SPICE2, is being developed at IPP ASCR as a continuation of collaboration with CEA Cadarache. It is a self-consistent PIC model with cloud-in-cell weighting and leapfrog particle motion. During its initial development, the code contained an iterative solver. Implementation of the package UMFPACK [*Davis*, 2004] has improved the performance significantly [*Komm et al.*, 2010]. SPICE2 is partially parallelized using domain decomposition and MPI (distributed memory parallelisation scheme), only the Poisson solver is serial.

The code has been successfully used to simulate various plasmas and geometries [e.g., *Komm et al.*, 2010, *Dejarnac et al.*, 2010]. However, more demanding simulations such as ITER baseline scenario, which is characterized with small Debye length $\lambda_\mathrm{D} \approx 10^{-7}$ m in scrape-off layer, are almost impossible to simulate so far. Because the grid cell size scales with λ_D, the overall grid dimensions rise up to thousands of cells in each direction when simulating an area with centimeter scale. This leads to proportionally large matrices of system (2) which take a long time to compute. They can also consume large amounts of RAM per solver thread.

The task of this paper is to investigate possibilities in parallelisation of the Poisson solver and compare it to the current method.

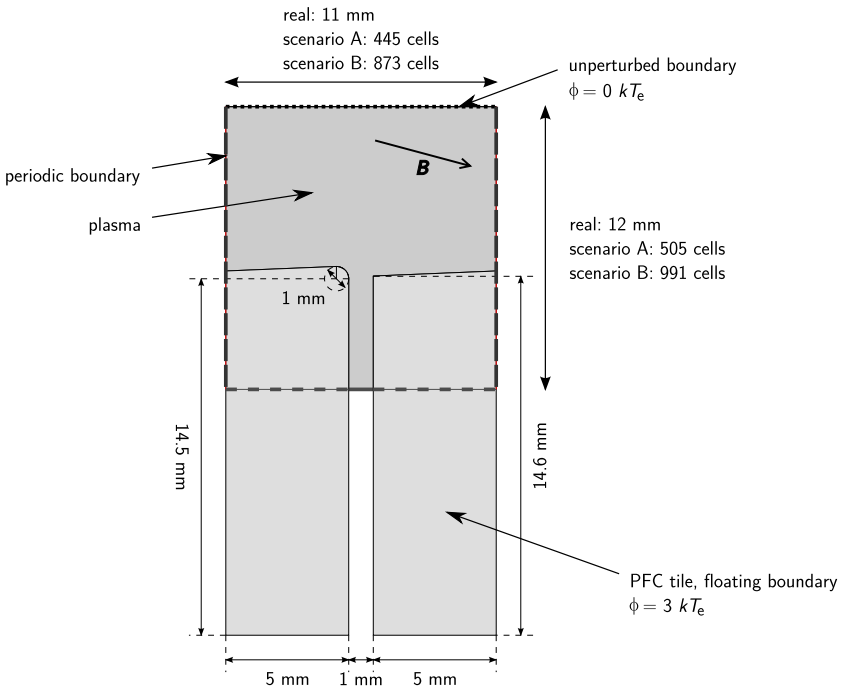

Figure 1. Testing geometry. The tile (light blue) is submerged into plasma (light red). Dimensions are specified for both testing scenarios. Boundary conditions at sides are periodic, bottom side is at floating potential (3 kT_e), top side represents unperturbed plasma (0 kT_e). The working area of the model is marked with wide lines.

Solvers and methods

A few studies have thoroughly compared the performance of sparse solvers. The report by *Gould et al.*, [2005] investigates symmetric system solvers. Fortunately, a few of them has capabilities of solving unsymmetric systems as well:

- **HSL MP48** [*HSL*, 2013]: If the system matrix is supplied in singly-bordered block-diagonal form, the solver performs LU decomposition on each submatrix in parallel. Message passing interface (MPI) parallelisation (the solver can run on 2 or more cores).
- **MUMPS** [*Amestoy et al.*, 2001]: Direct solver employiong multifrontal approach to LU decomposition. This package supports centralized and distributed matrix input. Both variants were investigated and they are marked as MUMPS-C (centralized input) and MUMPS-D (distributed input) in the following text. MPI parallelisation.
- **PARDISO** [*Schenk et al.*, 2004]: Direct solver, supernodal LU decomposition. OpenMP (OMP) parallelisation (MPI in the latest version).
- **UMFPACK**: Direct solver, LU decomposition. Reference case for comparison.

A simplified wrapper containing the matrix preparation, factorisation, and backwards substitution (solution) phase was developed to test the scaling under MPI environment. It mimicks the model workflow by running the factorisation followed by multiple iterations of solution phase. Different solvers can be easily switched using this approach. Memory consumption of the solver was monitored by Linux system tools. The wrapper was run 10 times to provide averaged results.

The computations were carried out using IPP cluster Abacus (2× 8-core Intel Xeon E5-2690, 2.90 GHz; 192 GB RAM) and fusion supercomputer HELIOS (Bullx B510, 8-core Intel Xeon E5-2680, 2.7 GHz; total 70,560 cores, 282,240 GB RAM). Abacus cluster provides scaling up to 16 processes, HELIOS was used to enhance the range up to 128 processes. Further scaling was not studied because the SPICE2 model does not show any performance increase with greater degree of parallelism than that.

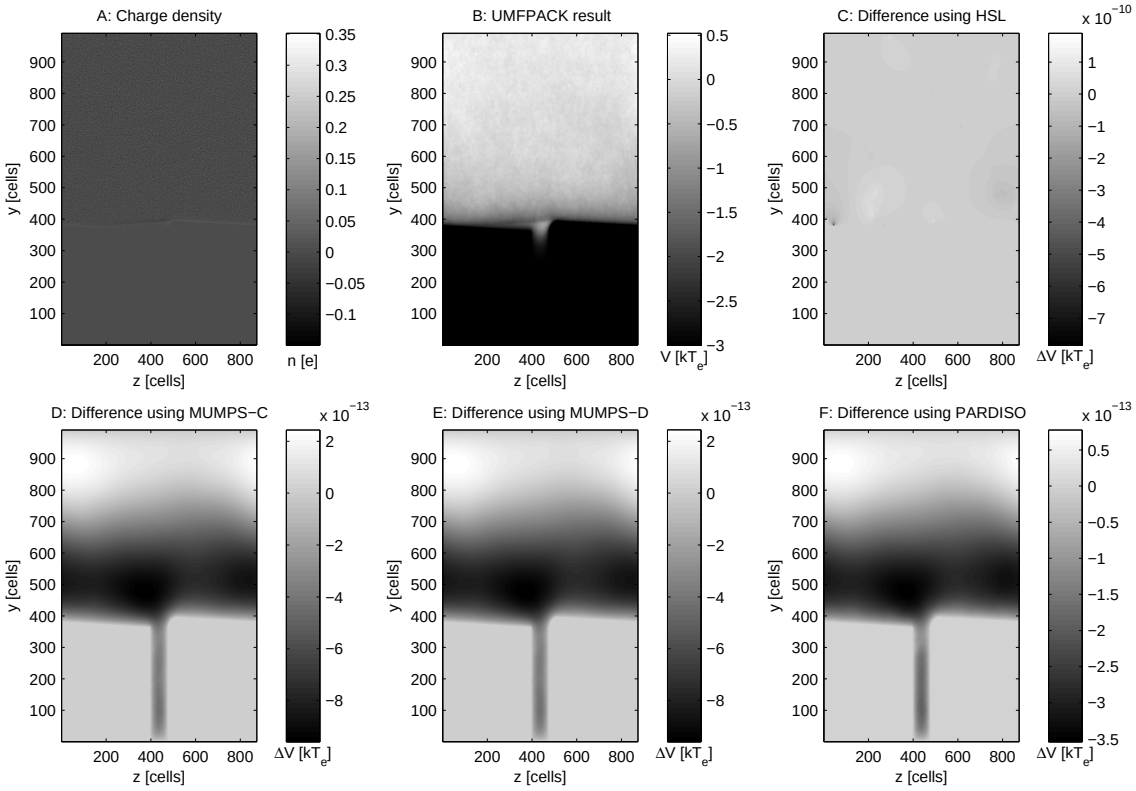

Figure 2. (A) Charge density used to provide the right hand side for equation (2); (B) UMFPACK reference result potential; (D–F): Differences from the result potential; All figures represent the result of the scenario B.

Testing case

The testing case was modelled after a typical scenario simulated with the SPICE2 code — the geometry and plasma parameters were taken from the setup used for the study the effect of shaped plasma-facing components that was carried out for the TEXTOR group. The results provided comparison for the experiment performed at DIII-D tokamak [*Litnovsky et al.*, 2014]. The geometry and dimensions are shown in Figure 1. Plasma parameters were set to $T_\mathrm{e} = 15$ eV, $n_\mathrm{e} = 1.3 \times 10^{18}$ m^{-3}, $T_\mathrm{i}/T_\mathrm{e} = 2$. Magnetic field of magnitude $B = 1.9078$ T[1] and inclination $\alpha = 3\,^\circ$ was superimposed over the region. This has resulted in base grid dimensions of 445×505 cells (scenario A) and subdivided grid dimensions of 873×991 cells (scenario B). The value of charge density ρ (2) was taken from the final output of DIII-D simulation and is shown in Figure 2 (panel A).

Scaling results

Following the implementation of all solvers, the accuracy run was done to prove their usability. It has been found that the results are in acceptable agreement with the reference case (Figure 2). The magnitude of the difference caused probably by rounding error is below $10^{-9}\ kT_\mathrm{e}$.

The time consumption scan was performed for up to 128 processes. Two scenarios were tested on Abacus cluster to provide a comparison for scaling with the problem size. The results are proportional to matrix dimensions and they show similar tendencies in the dependency on the degree of parallelisation. HELIOS cluster was used to enhance the range of the scan up to

[1]Precise value provided by the TEXTOR group.

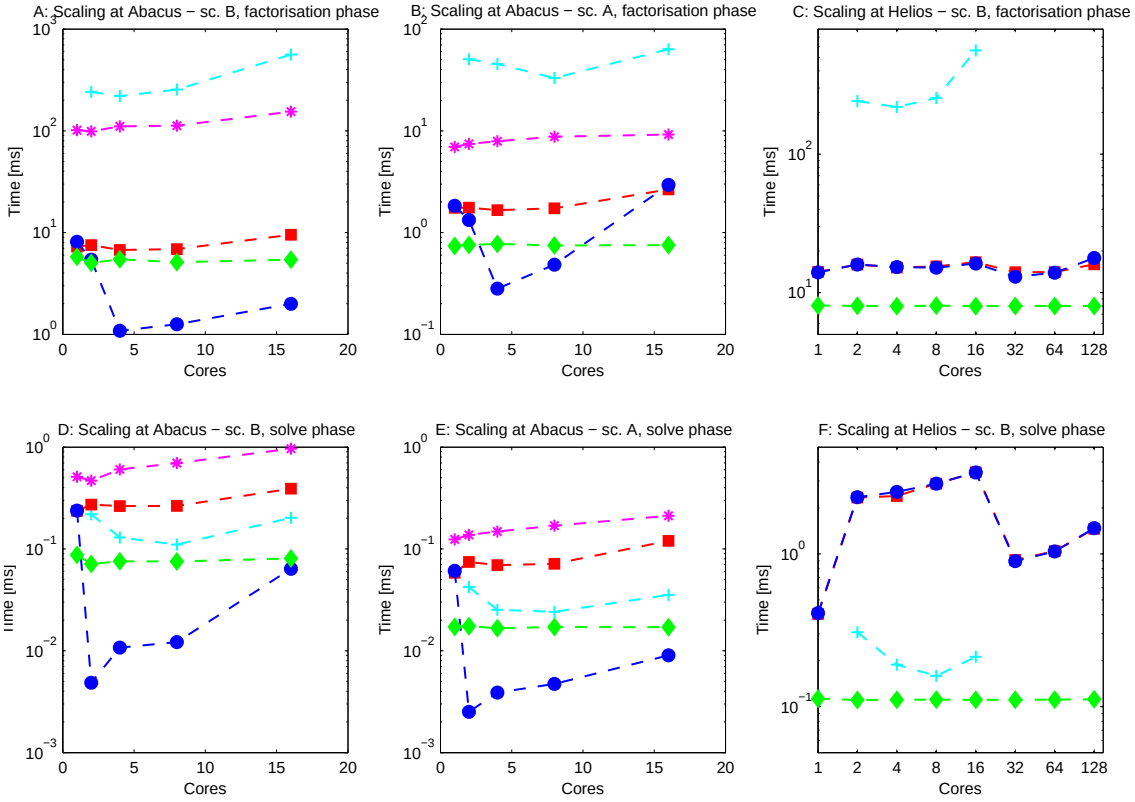

Figure 3. (A–C) Scaling of the time needed to factorize the matrix of the system for different scenarios and machines; (D–F) Scaling of the backwards substitution (solution) phase; Color coding: HSL MP48 (cyan/plus), PARDISO (magenta/asterisk), MUMPS-C (red/square), MUMPS-D (blue/circle), UMFPACK (green/diamond).

128 processes. Scaling results follow (see also Figure 3). Memory usage was determined during the run employing 4 processes.

- HSL MP48: The worst result in the factorisation phase, measurable solve phase scaling up to 8 processes. Peak memory usage: 400 MB in solution phase on host process, approx. 150 MB on other processes
- MUMPS-C: Acceptable result in the factorisation phase, poor solve phase scaling. Peak memory usage: 160 MB in factorisation phase (distributed among all processes)
- MUMPS-D: Acceptable result in the factorisation phase, good solve phase scaling (but seen only on Abacus). Peak memory usage: 140 MB in factorisation phase (distributed among all processes)
- PARDISO: Bad result in the factorisation phase, almost nonexistent solve phase scaling. Peak memory usage: 330 MB (distributed)
- UMFPACK: Good result in factorisation phase, no scaling. Peak memory usage: 220 MB on host process.

MUMPS package provides significant performance leap when the distributed matrix input is used, but this improvement was limited to only one computer. This behavior is probably caused by different versions of some of the libraries used by the solver, but none were found so far.

Conclusions

Four new possibilities of enhancement of current SPICE2 code were investigated. PARDISO and HSL packages provided unconclusive results as they are slower and more memory demanding

than the reference. MUMPS package is the suitable candidate for the SPICE2 upgrade mainly because its lower memory consumption and promising scaling at Abacus. Dependencies which cause the performance increase are yet to find. Other advantage of MUMPS is the capability of employing both OMP and MPI scheme of parallelisation. However, the overall performance (mainly the duration of the solution phase) of UMFPACK solver is so far the best.

SPICE2 solver requires distributed memory parallelisation (MPI) which can limit the scaling capabilities of sparse solvers, because some of them rely on shared memory approach (OMP). The next step is to combine those approaches using suitable package (MUMPS) to provide further possibilities in performance increase and evenly distributed memory usage.

Acknowledgments. A part of this work was carried out using the HELIOS supercomputer system at Computational Simulation Centre of International Fusion Energy Research Centre (IFERC-CSC), Aomori, Japan, under the Broader Approach collaboration between Euratom and Japan, implemented by Fusion for Energy and JAEA.

References

Amestoy, P. R., et al., A fully asynchronous multifrontal solver using distributed dynamic scheduling, *SIAM Journal of Matrix Analysis and Applications*, 23 (1), 15–41, 2001

Davis, T. A., Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method, *ACM Transactions on Mathematical Software (TOMS)*, 30 (2), 196–199, 2004.

Dejarnac, R., et al., Detailed Particle and Power Fluxes Into ITER Castellated Divertor Gaps During ELMs, *IEEE Trans. Plasma Sci.*, 38 (4), 1042–1046, 2010.

Gould, N. I. M, et al., A numerical evaluation of sparse solvers for symmetric systems, *CCLRC Technical Report*, RAL-TR-2005-005, 2005

HSL (2013), A collection of Fortran codes for large scale scientific computation, *http://www.hsl.rl.ac.uk*

Komm, M., et al., Particle-In-Cell Simulations of the Ball-Pen Probe, *Contrib. Plasma Phys.*, 50 (9), 814–818, 2010.

Litnovsky, D., et al., Optimization of tungsten castellated structures for the ITER divertor, *21st International Conference on Plasma Surface Interactions*, poster P3-016, 2014

Schenk, O., et al., Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO, *Journal of Future Generation Computer Systems*, 20 (3), 475–487, 2004