A Parallel Poisson Solver Library for Accelerator Modeling Applications

Ji Qiang\textsuperscript{*1}, He Zhang\textsuperscript{2}, Jean-Luc Vay\textsuperscript{1}, Axel Huebl\textsuperscript{1}, David Grote\textsuperscript{3}, Kiran Sonnad \textsuperscript{4}, and Chad Mitchell\textsuperscript{1}

\textsuperscript{1}Lawrence Berkeley National Laboratory, Berkeley, CA, USA
\textsuperscript{2}Thomas Jefferson National Accelerator Facility, Newport News, VA, USA
\textsuperscript{3}Lawrence Livermore National Laboratory, Livermore, CA, USA
\textsuperscript{4}Tech-X Corporation, Boulder, Co, USA

August 31, 2020

1 Introduction

Particle accelerators play a key role in high energy and nuclear physics studies, and many other applications. Design and operation of those sophisticated and expensive devices require extensively using multi-physics computer simulations. Those multi-physics computer models include space-charge effects, beam-beam effects, electron cloud effects, and other physical effects. In order to simulate the aforementioned effects self-consistently, one has to solve the Poisson equation at each step. A parallel, fast Poisson solver will be critical for the quick return in accelerator modeling applications.

In the accelerator community, a number of fast Poisson solvers were used and developed for accelerator modeling applications [1–24]. Subject to different boundary conditions, those Poisson solvers involve different numerical methods that solve the Poisson’s Equation on a grid. For an open boundary condition, an FFT-based Green’s function method was developed [3–10]. For a closed boundary condition with regular shape, a finite difference spectral method was used [11–15]. For a closed boundary condition with irregular shape, a multigrid finite difference method is often used [16–18, 25].

Besides solving the differential Poisson equation directly, the integral equation method provides a category of alternative solvers based on the integral format. These solvers convert the differential equation into an integral equation.

\textsuperscript{*}jqiang@lbl.gov
on the boundary. They do not need the grid that covers the whole domain and thus avoid any issues caused by the grid. The fast multipole method [19, 20], which has been implemented in a few accelerator simulation tools [21, 22], can be especially efficient with open boundaries. The integral equation method can also be applied to boundary value problems with any geometry. Finally, we note that some algorithms and codes [23, 24, 26] have been developed by computational mathematicians but have not been transplanted into the accelerator community.

2 Current and future challenges

So far, those Poisson solvers were developed on different computer platforms for accelerator modeling applications. Some of the implementations are serial while others take advantage of parallel architectures. They were also developed using different programming languages, e.g. Fortran, C++, Python or MatLab. There is also no uniform interface for accelerator modelers to use those Poisson solvers conveniently or interchangeably. Furthermore, the parallel Poisson solvers have often not been optimized for good efficiency and scalability on massive parallel computers, and very few have been ported to GPUs.

3 Advances needed to meet challenges

In order to meet the above challenges, we propose to develop a fast, portable, parallel Poisson solver library, which would improve the usage of those fast Poisson solvers and benefit the accelerator modeling applications. Modern programming practices and tools can be leveraged to develop the library that will be portable across CPUs and GPUs, and with standardized interfaces that make it easy for accelerator modelers to use in their applications, toolkits and ecosystem [27, 28]. Multiple levels of parallelism (on-node and multi-nodes) will be supported, with special attention to efficiency and scalability. The library will also include a detailed and up-to-date user documentation, as well as automated test suites and well-benchmarked examples.

References


The solver specifically considered in this work is the parallel Algebraic Recursive Multilevel Solver (pARMS), which is a distributed-memory parallel solver based on Krylov subspace methods. First we integrate a randomization technique referred to as Random Butterfly Transformations (RBT) that has been successfully applied to remove the cost of pivoting in the solution of dense linear systems. In this PhD thesis, we focus on preconditioned iterative solvers for large sparse linear systems, and more specifically on solvers based on Krylov subspace methods. The randomized algorithms are now gaining ground in HPC applications and are more and more used to enhance linear algebra solvers since they can outperform deterministic methods while providing accurate results. We propose a parallel dart throwing algorithm for efficient Poisson disk sampling on surfaces. In this section, we describe our algorithm for uniform surface sampling, where the radius r is uniform everywhere on the surface. We start with Euclidean distance, then extend it to geodesic distance using a fast approximation. For simplicity, we only consider triangle mesh surfaces, but other surface types can be easily incorporated. We do not require mesh parametrization. Our algorithm builds upon [Wei 2008], which performs parallel dart throwing in a continuous n-dimensional Euclidean space. Using 3D To use parallel solver in FreeFEM, we need to load the dynamic library corresponding to this solver. For example, to use MUMPS solver as parallel solver in FreeFEM, write in the .edp file: load "MUMPS_FreeFem". If the libraries are not loaded, the default sparse solver will be loaded (default sparse solver is UMFPACK). The Table 2 gives this new value for the different libraries. Table 2 Default sparse solver for real and complex arithmetics when we load a parallel sparse solver library. Libraries. Default sparse solver. Designed a parallel C++ solver with libraries MPI, MKL, and LAPACK based on Kaczmarz scheme to solve a sparse linear system Ax = f of order 1 million. Implemented the RCK scheme to do reordering in sparse matrix and utilized Block Gauss-Seidel and permutations to transform the model into independent least square problems within a parallel Conjugate Gradient framework, resulting in a more robust and scalable method compared with GMRES using ILU preconditioner. We present a parallel implementation of a direct solver for the Poisson' 09/12/2019 â€™ by Jaber J. Hasbestan, et al. â€™ 0 â€™ share. Good scaling properties for a semi-Lagrangian solver of the gyrokinetic model have been demonstrated for the GYSELA code by Grandgirard et al. (2016) and Latu et al. (2016). Another challenge compared to the gyrokinetic model is the reduced complexity of the kinetic model causing a reduced computational complexity that renders the code more memory-bound. The developments were made within the framework of the library SeLaLib222Selalib home, http://selalib.gforge.inria.fr/, accessed on 2018-04-15. The 6d distribution function is discretized on a 6d Fortran array.