CONSERVATIVE PROJECTION BETWEEN FINITE ELEMENT AND PARTICLE BASES

JOSEPH V. PUSZTAY, MATTHEW G. KNEPLEY, AND MARK F. ADAMS

Abstract. Particle-in-Cell (PIC) methods employ particle representations of unknown fields, but also employ continuum fields for other parts of the problem. Thus projection between particle and continuum bases is required. Moreover, we often need to enforce conservation constraints on this projection. We derive a mechanism for enforcement based on weak equality, and implement it in the PETSc libraries. Scalability is demonstrated to more than 1B particles.

simulation, particle in cell, PIC, PETSc
65F50, 65M60

1. Introduction

The Particle-in-Cell (PIC) method is a hybrid discretization, using a grid of finitely supported basis function to represent some fields, and a distribution of radial basis functions to represent other fields. The first discretization is usually called the *mesh* basis, whereas the second is called the *particle* basis. The particle basis can be useful for representing sharply localized quantities [6, 14], or fields in high dimension, or quantities that are more sensitive to artificial diffusion and dispersion. For example, it is common in plasma physics to represent clusters of electrons as a single "macro particle" eliminating the computational overhead of individual electron interactions [6]. In high dimension, particles are capable of more rapid convergence than mesh discretizations [1]. The self-consistent interaction of particles and fields in PIC methods can be important in resolving localized nonlinear effects [2].

Particles may be represented using a shape function to capture particle width in the same way that finite element shape functions represent field behavior inside a mesh cell. Radial basis function methods are an example of this representation [11]. In our experiments, we will use only the traditional delta function representation of particles, but the algebraic relations derived in Section 2 remain valid for any shape function for which accurate quadrature is available. Each particle is assigned a weight, the coefficients of a basis function in the particle representation.

Joseph V. Pusztay - University at Buffalo, Buffalo, NY. (josephpu@buffalo.edu).
Matthew G. Knepley - University at Buffalo, Buffalo, NY. (kneply@buffalo.edu).
Mark F. Adams - Lawrence Berkeley National Laboratory, Berkeley, CA. (mfadams@lbl.gov).
Submitted/accepted to SISC.

JP was funded by the Department of Energy Exascale Computing Project WDMApp under subcontract S017663 and the Partnership for Edge Physics Simulation, Lawrence Berkeley National Laboratory subcontract 18083698. MGK was partially funded by the Department of Energy Applied Math Research under U.S. DOE Contract DE-AC02-06CH11357 and by the National Science Foundation under grant NSF CSSI: 1931524.

When employing PIC in large scale, long running simulations, particular attention must be paid to the choice of time marching integrator, selection of the mesh/particle bases, and projections between the bases to appropriately conserve mass, momentum, and energy. Conservation of these quantities has motivated a great deal of study in particle methods, particularly in the field of computational plasma physics [25, 10, 22, 8]. In this paper, we focus on a *conservative* projection between basis representations, meaning projection which enforces the conservation of moments of the distribution. For example, conservation of the zeroth moment is equivalent to mass or charge conservation, the first moment to momentum conservation, and the second moment to energy conservation.

To begin, we will briefly define the PIC representation in 2. Performance benchmarks for preconditioning and linear solves are discussed in 3. We remark on the applications for these projection operators in 4 before commenting on future improvements in 5

2. Derivation

Suppose our system is comprised of small, massive bodies interacting with a potential. The standard representation is to assign a shape function to each, such that their weighted sum comprises the particle distribution function f_P. This distribution function can be interpreted as the probability of finding a particle at a given location $\mathbf{x}$ in configuration (or phase) space. Note that since we are working in phase space, x would comprise both position and velocity. Thus, we can find by expected number of particles n by integrating over all phase space

$$\int_\Omega f_P = n. \tag{1}$$

where Ω is the phase space domain. For simplicity, we use delta functions for this discussion, but our method does not depend sensitively on the choice of shape function. The full particle distribution function can then be written

$$f_P = \sum_p w_p \delta(\mathbf{x} - \mathbf{x_p}) \tag{2}$$

where $\mathbf{x}$ represents the configuration space variable, $\mathbf{x_p}$ is the particle location and velocity, and w_p the particle weight. The finite element representation, using function space $\mathcal{V}$, is given by the weighted sum of basis function,

$$f_{FE} = \sum_i f_i \phi_i(\mathbf{x}) \tag{3}$$

where $\phi_i \in \mathcal{V}$ denotes the basis function, and f_i the associated coefficient.

At this point, we must determine what we mean when we say these two expressions represent the same field. For instance, in phase space, the two systems need to contain the same mass, represented by equivalence of the integrals

$$\int_\Omega f_{FE} = \int_\Omega f_P. \tag{4}$$

In PIC codes, it is very common to enforce a tensor produce structure on the phase space. For example, many collision operators, such as the Landau operator, are completely localized in space. This converts the integral over phase space into

an integral over velocity space Ω_V [12, 24]. This means that equality of the first moment in phase space,

$$\int_\Omega \mathbf{x} f_{FE} = \int_\Omega \mathbf{x} f_P. \tag{5}$$

becomes the conservation of momentum

$$\int_{\Omega_V} m\mathbf{v} f_{FE} = \int_{\Omega_V} m\mathbf{v} f_P \tag{6}$$

when multiplied by the mass m. In the same way, conservation of the second moment

$$\int_\Omega |\mathbf{x}|^2 f_{FE} = \int_\Omega |\mathbf{x}|^2 f_P \tag{7}$$

becomes the conservation of kinetic energy

$$\int_{\Omega_V} \frac{1}{2} m|\mathbf{v}|^2 f_{FE} = \int_{\Omega_V} \frac{1}{2} m|\mathbf{v}|^2 f_P. \tag{8}$$

These conditions are all specific instances of what we will call *weak equivalence*, namely the equivalence of two expressions in the subspace spanned by some set of functions,

$$\int_\Omega \phi_i \, f_{FE} = \int_\Omega \phi_i \, f_P \quad \forall \phi_i \in \mathcal{V} \tag{9}$$

The finite dimensional analogue of 9 is perhaps easier to interpret. We begin by expanding each field into its components [15, 16]

$$\int_\Omega \phi_i \sum_j f_j \phi_j = \int_\Omega \phi_i \sum_p w_p \delta(\mathbf{x} - \mathbf{x_p}), \tag{10}$$

$$\sum_j f_j \int_\Omega \phi_i \phi_j = \sum_p w_p \int_\Omega \phi_i \delta(\mathbf{x} - \mathbf{x_p}). \tag{11}$$

We can rewrite this using linear algebraic notation,

$$Mf = Vw \tag{12}$$

where M is the FEM mass matrix

$$M = \int_\Omega \phi_i \phi_j, \tag{13}$$

V the particle mass matrix

$$V = \int_\Omega \phi_i \delta(\mathbf{x} - \mathbf{x_p}), \tag{14}$$

f the vector of finite element coefficients, and w the vector of particle weights. The particle mass matrix entries are the evaluation of FEM basis functions at the particle positions, with the basis function determining the row and the particle determining the column. This simple equation allows us to convert particle weights to finite element coefficients, and vice versa, while preserving all the moments which are contained in the finite element space. Note, that as particle positions or momenta change, V will likewise change.

Computation of M and V is straightforward when delta functions are used to express f_P. If more general shape functions are used, accurate calculation of the elements of V could become expensive since accurate quadrature for the overlap of the FEM basis functions and the particle shape function would be necessary.

Inversion of the finite element mass matrix, M, is straightforward given the favorable conditioning [28], and we employ conjugate gradient method preconditioned with ILU(0) in 3. The particle mass matrix need not be square, and thus a solver for over/underdetermined systems is necessary [26], in most cases with preconditioning for scalability. Typical preconditioning uses an approximation to the normal equations, so we recall that

$$V_{ip} = \int_\Omega \phi_i \delta(x - x_p), \tag{15}$$

so that our normal equations are given by

$$V^T V = V_{qi} V_{ip}, \tag{16}$$

$$= \left(\int_\Omega \phi_i \delta(x' - x_q) \right) \left(\int_\Omega \phi_i \delta(x - x_p) \right), \tag{17}$$

$$= \phi_i(x_q)\phi_i(x_p) \tag{18}$$

In our experiments, preconditioning with the normal equations is quite effective, and reduces the number of iterations by one to two orders of magnitude.

3. Numerical Results

We have implemented the projection algorithm from Section 2 using the PETSc libraries [4, 5, 3]. Our particle basis is embodied by a DMSwarm object [23], and the finite element mesh and function space by a DMPlex object [17, 21]. The PETSc suite of linear solvers is used to solve the algebraic equations for conservative projection, Eq 12.

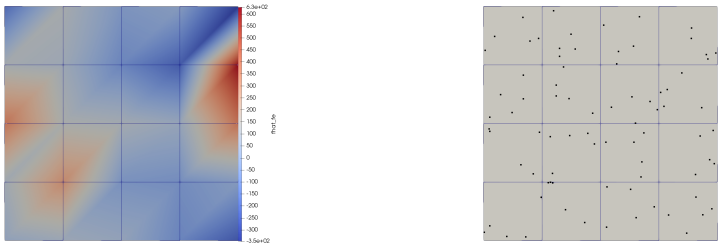
The principal application for this algorithm is currently the projection of a particle field to a finite element space for the purpose of applying a collision operator, in particular the Landau operator [1, 24]. Therefore, we will restrict our phase space to the velocity component. We produce a mesh covering this space that defines our finite element space, and a set of particles which define the particle field. We project the particle field to the finite element space and back, checking for conservation of moments at each step.

Runtime of the projections solves is controlled principally by the choice of preconditioner. A strong preconditioner is also necessary in order to reach high accuracy in the solve. We require errors on the order 10^{-14} in our tests. Inversion of the finite element mass matrix in the particle deposition step is well understood, and our choice of CG/ILU(0) is sufficient. However, the weakly coupled, block nature of the particle mass matrix presents an opportunity to study block preconditioners for the normal equations, using LSQR [26] as the Krylov solver. To this end, a *weak scaling* study is presented for preconditioned LSQR preconditioned with Block Jacobi on the normal equations using sparse LU on each block, as well as Additive Schwarz (ASM) with ICC(0) on each block. All tests were performed using standard PETSc examples, making it straightforward for a reader to reproduce the results, and to alter the solvers, such as replacing our choice of Krylov solver or block subsolver. The numerical results present a study on fixed particle number per cell, with an increasing number of cells such that the number of cells across a fixed domain remain fixed on each node.

All performance studies were conducted on the Geosolver cluster, located at the University at Buffalo Center for Computational Research. The Geosolver consists of

100 Intel Xeon Gold 2.6GHz 12-core 6126 processors, with 8GB of RAM per core, for a total of 1200 cores across 50 nodes with Intel Omnipath interconnects. We present the PETSc log for a representative run consisting of > 1 billion particles in the supplementary materials associated with this article, as well as instructions for solver configurations in PETSc for duplication of the presented results. We present 3 as a smaller version of the system being solved in both a field and particle representation.

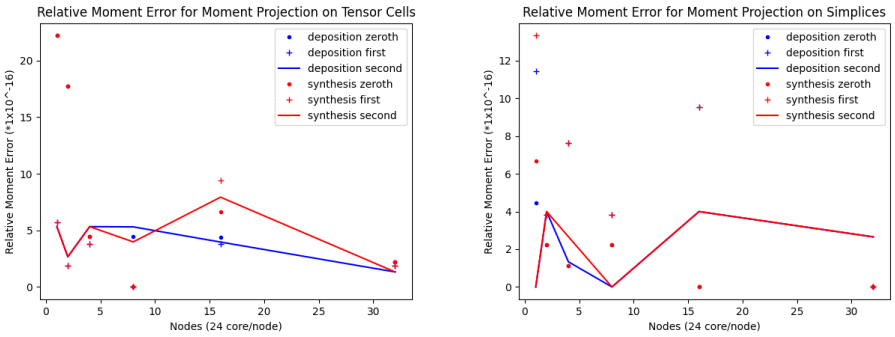
Weak scaling is demonstrated with an initial configuration of $256 \times 256 = 65,536$ quadrilateral, or tensor, cells with 200 particles per cell, and a simplicial mesh of $128 \times 128 = 16,348$ cells with 200 particles per cell. Particle coordinates within each cell are randomized for each run. One node using 24 MPI processes was chosen as a starting point. As the number of nodes were doubled, the domain was further subdivided by increasing the cell count by a factor of two, resulting in each node holding a 256×256 cell mesh in the tensor case, or 128×128 in the simplicial case. All tests use a quadratic element space, with third order quadrature on the cells, however the simulation can change the order and quadrature with command line arguments.



(A) FE Coefficients (B) Particles

FIGURE 1. Visualization of the finite element field (a) arising for a distribution of particles (b) on a 4x4 grid with 5 particles per cell.

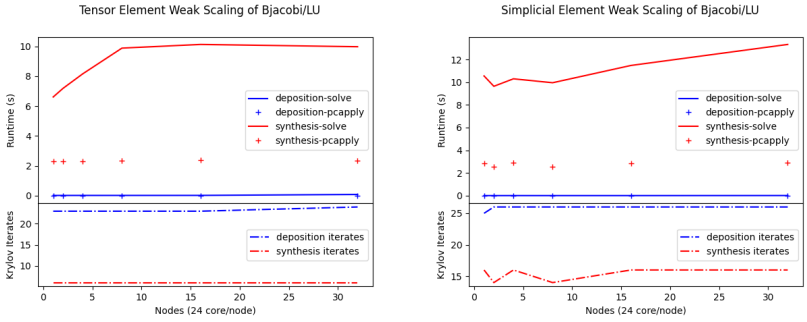
Before we discuss the scaling with respect to a moderate, fixed particle number per cell, it is useful to observe the capabilities of these projections on a smaller scale. To do so, we configure an initial 256x256 tensor mesh on one node, with each cell containing one particle positioned at its centroid. Particle deposition, the projection of the particle field into the FE space (ptof), and particle synthesis, projection from the FE to particle space (ftop), are then performed, and the relative error between the moments of the bases are observed. 2 plots the relative error of each moment as a function of mesh size, and demonstrates the ability to accurately reconstruct the field in the particle synthesis step with a single particle. An additional weak scaling study is presented with a "checkerboard" configuration in the respective subsections for each solver configuration. The particle numbers in the checkerboard configuration alternate between five and six particles for each given cell of the mesh.



(A) Tensor Cells (B) Simplices

FIGURE 2. Moment error for the zeroth (mass), first (momentum), and second (energy) moments after projection from particles to FE (deposition) and FE to particles (synthesis) in a single particle per cell case for simplices and tensor cells. The particle is located at the centroid of each cell, and demonstrates a single particle is sufficient to reconstruct the field with relative moment error within $100 * \epsilon_{machine}$ [30] regardless of overall mesh size.

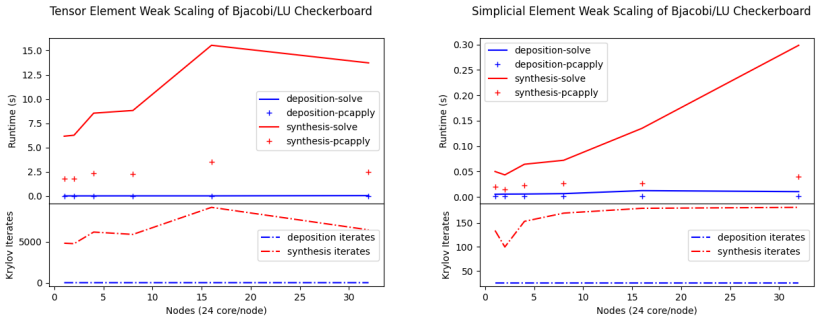
3.1. **Block Jacobi+LU Weak Scaling.** Using LSQR as the outer Krylov solver, with a Block-Jacobi preconditioner built from the normal equations using sparse LU on each block is effective and exhibits good weak scaling. 3a displays this behavior for both particle synthesis and deposition on a tensor mesh, and 3b displays results for the simplicial grid.



(A) 256x256 tensor cells per node (B) 128x128 simplicial cells per node

FIGURE 3. Weak scaling for system solution on tensor and simplicial grids. Both systems have 200 particles per cell and a LSQR/Block Jacobi/LU solver.

Block Jacobi/LU reliably solves the system to within $100 * \epsilon_{machine} \sim 2.2 * 10^{-14}$ regardless of the division of the domain and spacing of the particles when the



(A) 256x256 tensor cells per node

(B) 128x128 simplicial cells per node

FIGURE 4. Weak scaling for system solution on tensor and simplicial grids. Both systems are comprised of a "checkerboard" pattern, alternating five and six particles per cell. The LSQR/Block Jacobi/LU solver is used in both cases.

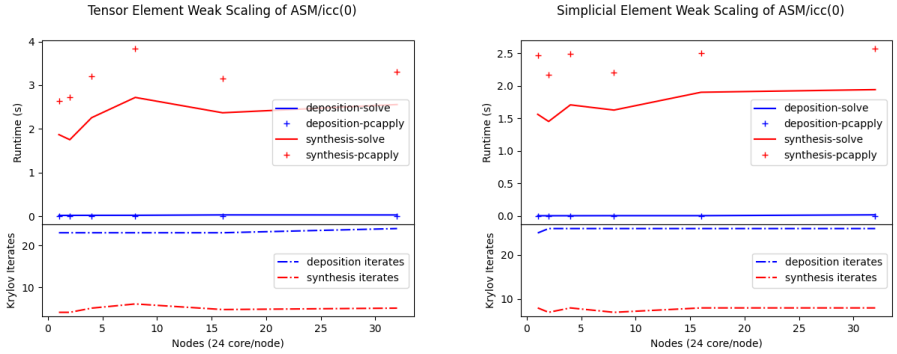
iterative residual tolerance is low enough, at $\sim 10^{-15}$. However, runtimes for moderately large systems can be improved using ASM, as discussed in 3.2. Parallel direct solvers such as MUMPS and SuperLU_dist were considered, but did not decrease the overall time due to increased setup costs, communication overhead, and memory limitations.

In the case of simplicial elements with varying particle number per cell, the number of Krylov iterates scales well, however, at these particle numbers, scatter operations do not scale well and contribute heavily to increases in run time at each increase in the number of nodes and size of the grid. It is also observed in the tensor elements that the increased cost of scatter operations heavily increases run time, with scattering operations on average taking 100x longer in addition to the increase in Krylov iterations necessary to sufficiently solve the system as shown in 4.

3.2. **Additive Schwarz with Incomplete Cholesky.** We can accelerate convergence using ASM [29, 27] since only weak coupling exists between the blocks and their nearest neighbors. Incomplete Cholesky factorization (ICC) [9] is used to precondition the blocks, with the factorization being done in-place. ASM/ICC(0) improves total solve time by a factor of 2 or more in many cases, as seen in 5a, and reduced the memory footprint. However, the time due to communication overhead is noticeable at larger problem sizes, and might be eliminated with a coarse problem. In our largest test case, a run of 1,000,833,800 particles on a 2237×2237 tensor mesh was converged to machine precision, well beyond what could be achieved with the memory constraints in performing a direct solve.

Compred to the Block Jacobi/Sparse LU, 6 demonstrates similar behavior of ASM/ICC(0) in regards to scattering operations with low particle number and varying numbers of particles per cell.

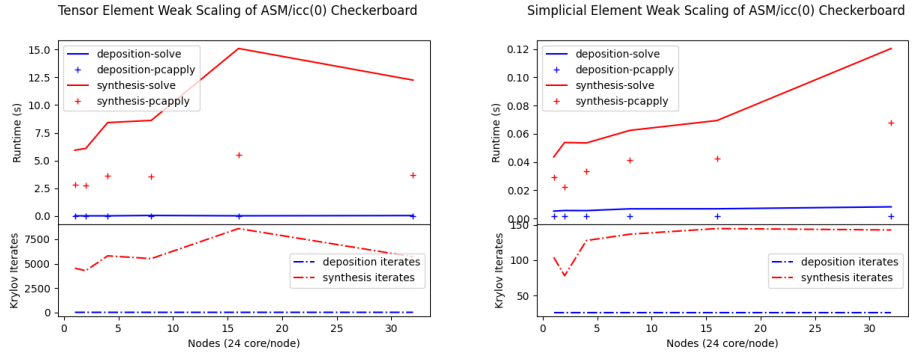
An alternative approach to this problem would be to employ a discontinuous Galerkin (DG) discretization for the FEM basis [18, 13]. This would completely eliminate coupling between blocks, making the solver purely local. However, this would require a more sophisticated solver for the FEM mass matrix, and also for



(A) 256x256 tensor cells per node

(B) 128x128 simplicial cells per node

FIGURE 5. Weak scaling for system solution on tensor and simplicial grids. Both systems have 200 particles per cell and a LSQR/ASM/ICC solver.



(A) 256x256 tensor cells per node

(B) 128x128 simplicial cells per node

FIGURE 6. Weak scaling for system solution on tensor and simplicial grids. Both systems are comprised of a "checkerboard" pattern, alternating five and six particles per cell. The LSQR/ASM/ICC solver is used in both cases.

the associated finite element problem for the continuum physics. For simulations which are already formulated using a DG discretization, this would seem to be a good choice.

4. APPLICATIONS

Conservative projection operations have applications wherever the PIC discretization is used, but we place particular emphasis on applications in plasma physics codes. For instance, the conservative projection operations we present here have been adapted for, and implemented in, the gyrokinetic PIC code XGC [19, 20] in

order to maintain conservation in their particle to velocity-grid mappings. This code uses the same PETSc solvers detailed above to enforce the algebraic condition in Eq. 12. If conservation is not maintained to high precision, it can cause numerical problems for the simulation, such as artificial heating of the plasma in a steep edge pedestal [24]. Furthermore, conservative particle-to-grid transformations offer an opportunity to fully leverage the work of Adams and Hirovijoki in the formulation of conservative discretizations of the Landau Collision integral [12] (implemented in PETSc) when used in other PIC codes where conservation at each step is crucial for overall numerical accuracy.

Another potential application is the direct tracking of different material phases without explicit interface tracking. For example, in a hybrid rocket engine, solid paraffin fuel atomizes and is entrained in the liquid oxidizer [7]. These fuel droplets can be tracked using a particle basis, and interact thermally with the oxidizer using a PIC scheme. Moreover, the transition between mesh based computation of droplet formation and a Lagrangian particle description in the oxidizer flow can be handled using our conservative projection scheme.

5. Conclusions

We have derived a method to conservatively project a field between particle and finite element representations, and studied the weak scaling behavior of these solves with different types of preconditioning on the particle mass matrix. LSQR/BJ/LU offers a reliable solve that scales well, but can be limited by the memory footprint. LSQR/ASM/ICC(0) overcomes the memory limitation while maintaining scalability, but can suffer some loss of accuracy near machine precision. This is likely a result of the factor shift introducing some ill-conditioning. This effect is limited and does not always occur, and in general LSQR/ASM/ICC(0) is a good choice for the particle projection solve.

Acknowledgments

We gratefully acknowledge assistance conducting large scale benchmarks on the Geosolver cluster at the UB Center for Computational Research by Dori Sajdak and Cynthia Cornelius.

References

[1] M. F. Adams, D. P. Brennan, M. G. Knepley, and P. Wang, *Exascale Landau collision operator in the CUDA programming model applied to thermal quench plasmas*, arXiv preprint arXiv:2104.10000, (2021).

[2] O. Allanson, C. Watt, H. Ratcliffe, N. Meredith, H. J. Allison, S. Bentley, T. Bloch, and S. Glauert, *Particle-in-cell experiments examine electron diffusion by whistler-mode waves: 1. benchmarking with a cold plasma*, Journal of Geophysical Research: Space Physics, 124 (2019), pp. 8893–8912.

[3] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc Web page*. `https://www.mcs.anl.gov/petsc`, 2021.

[4] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Karpeyev, D. Kaushik, M. G. Knepley, D. A. May, L. C. McInnes, R. T. Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, *PETSc users manual*, Tech. Rep. ANL-95/11 - Revision 3.11, Argonne National Laboratory, 2019.

[5] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser Press, 1997, pp. 163–202.
[6] C. K. Birdsall and A. B. Langdon, *Plasma physics via computer simulation*, CRC press, 2018.
[7] K. Budzinski, S. S. Aphale, E. K. Ismael, G. S. III, and P. E. DesJardin, *Radiation heat transfer in ablating boundary layer combustion theory used for hybrid rocket motor analysis*, Combustion and Flame, 217 (2020), pp. 248–261.
[8] L. Chacón, G. Chen, and D. Barnes, *A charge- and energy-conserving implicit, electrostatic particle-in-cell algorithm on mapped computational meshes*, Journal of Computational Physics, 233 (2013), pp. 1–9.
[9] T. F. Chan and H. A. V. der Vorst, *Approximate and Incomplete Factorizations*, Springer Netherlands, Dordrecht, 1997.
[10] G. Chen, L. Chacón, and D. Barnes, *An energy- and charge-conserving, implicit, electrostatic particle-in-cell algorithm*, Journal of Computational Physics, 230 (2011), pp. 7018–7036.
[11] B. Fornberg and N. Flyer, *A primer on radial basis functions with applications to the geosciences*, SIAM, 2015.
[12] E. Hirvijoki and M. F. Adams, *Conservative discretization of the landau collision integral*, Physics of Plasmas, 24 (2017), p. 032121.
[13] E. Hirvijoki, M. Kraus, and J. W. Burby, *Metriplectic particle-in-cell integrators for the landau collision operator*, 2018.
[14] R. W. Hockney and J. W. Eastwood, *Computer simulation using particles*, crc Press, 2021.
[15] R. C. Kirby, *Algorithm 839: Fiat, a new paradigm for computing finite element basis functions*, ACM Transactions on Mathematical Software, 30 (2004), pp. 502–516.
[16] M. G. Knepley, J. Brown, K. Rupp, and B. F. Smith, *Achieving high performance with unified residual evaluation*, ArXiv e-prints, (2013).
[17] M. G. Knepley and D. A. Karpeev, *Mesh algorithms for PDE with Sieve I: Mesh distribution*, Scientific Programming, 17 (2009), pp. 215–230. `http://arxiv.org/abs/0908.4427`.
[18] M. Kraus and E. Hirvijoki, *Metriplectic integrators for the Landau collision operator*, Physics of Plasmas, 24 (2017), p. 102311.
[19] S. Ku, C. Chang, and P. Diamond, *Full-f gyrokinetic particle simulation of centrally heated global ITG turbulence from magnetic axis to edge pedestal top in a realistic tokamak geometry*, Nuclear Fusion, 49 (2009), p. 115021.
[20] S. Ku, C. Chang, R. Hager, R. Churchill, G. Tynan, I. Cziegler, M. Greenwald, J. Hughes, S. Parker, M. Adams, et al., *A fast low-to-high confinement mode bifurcation dynamics in the boundary-plasma gyrokinetic code XGC1*, Physics of Plasmas, 25 (2018), p. 056107.
[21] M. Lange, L. Mitchell, M. G. Knepley, and G. J. Gorman, *Efficient mesh management in Firedrake using PETSc-DMPlex*, SIAM Journal on Scientific Computing, 38 (2016), pp. S143–S155.
[22] S. Markidis and G. Lapenta, *The energy conserving particle-in-cell method*, Journal of Computational Physics, 230 (2011), pp. 7037–7052.
[23] D. A. May and M. G. Knepley, *DMSwarm: Particles in PETSc*, in EGU General Assembly Conference Abstracts, EGU General Assembly Conference Abstracts, Apr. 2017, p. 10133.
[24] A. Mollén, M. F. Adams, M. G. Knepley, R. Hager, and C. S. Chang, *Implementation of higher-order velocity mapping between marker particles and grid in the particle-in-cell code xgc*, Journal of Plasma Physics, 87 (2021), p. 905870229.
[25] P. J. Morrison, *Structure and structure-preserving algorithms for plasma physics*, Physics of Plasmas, 24 (2017), p. 055502.
[26] C. C. Paige and M. A. Saunders, *Algorithm 583: LSQR: Sparse linear equations and least squares problems*, ACM Transactions on Mathematical Software (TOMS), 8 (1982), pp. 195–209.
[27] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, United Kingdom, 1996.
[28] A. J. Wathen, *Realistic eigenvalue bounds for the Galerkin mass matrix*, IMA Journal of Numerical Analysis, 7 (1987), pp. 449–457.

[29] O. Widlund and M. Dryja, *An additive variant of the Schwarz alternating method for the case of many subregions*, Technical Report 339, Ultracomputer Note 131, Department of Computer Science, Courant Institute, Dec. 1987.
[30] Wikipedia, *Machine epsilon*. `https://en.wikipedia.org/wiki/Machine_epsilon`, 2021.