

Sp 04 CS267 Final Project: Parallelization of `oopd1`

Jeff Hammel

June 17, 2004

1 Introduction

`oopd1` is a new object-oriented particle in cell (particle-mesh) code for modeling plasma kinetics being written by the Plasma Theory and Simulation Group (PTSG). `oopd1` is an acronym for **Object-Oriented Plasma Device 1-Dimension**, and is a more unified and extensible replacement for the previous PTSG suite of C 1d particle in cell (PIC) codes – `xpdp1`, `xpdc1`, `xpds1` – each written for a separate coordinate system (planar, cylindrical, and spherical, respectively).

1d-3v PIC codes, such as `oopd1`, model systems with one spatial dimension and three velocity components. Since computational work scales linearly with dimension in 1d (instead of quadratically in 2d or cubically in 3d), one dimensional PIC codes are used to simulate systems that can be approximated with such a model, as 1d results are obtained an order of magnitude faster than 2d results for similar systems.. 1d PIC has been used to model vacuum electronics, space plasmas, laser-plasma interactions, plasma etch reactors and numerous other examples.

The hierarchy of classes in `oopd1`, both implemented and planned, is shown in Figure 1.

The purpose of this project is to write a parallel 1d-1v parallel PIC algorithm using C++ and MPI in order to assess the computational issues with including this model in `oopd1`. This simplified code, named `simpic`, will be used to assess the time used in communication, the field solve, and the particle push for several problem sizes in order to diagnose and improve PIC computational performance for serial and parallel versions. With this information, `oopd1` will be later parallelized using MPI including more general parallelization beyond the scope of this CS267 project (initialization, load balancing, arbitrary boundary conditions, multiple field solves, etc.).

2 Particle In Cell Methodology

2.1 Description of Particle In Cell

Particle in cell (PIC) is a particle-mesh method. This means that the phase space distribution of particles, $f(v)$, is represented by collocation of discrete computational particles, with each computational particle representing $np2c$ real particles. Instead of computing the interactions between each particle, the particle density (and higher moments for some PIC methods) is interpolated (weighed) to the mesh. A field equation (Poisson's equation) is solved using this information and the computed force is interpolated back to the particle which is used to update its velocity and position. This alternating field solve and particle push is repeated for each time step. A general flow chart for PIC may be seen in Figure 2. For electrostatic PIC, Poisson's equation is solved for potential and then numerically differentiated to determine the electric field.

The Poisson solver and the particle push are briefly described.

2.2 Poisson Solver

Poisson's equation for electrostatics is applicable when the magnetic field is not present or not time varying (magnetostatic). In this case, the electric field may be written as the gradient of a scalar potential field, Φ :

$$\mathbf{E} = -\nabla\Phi \tag{1}$$

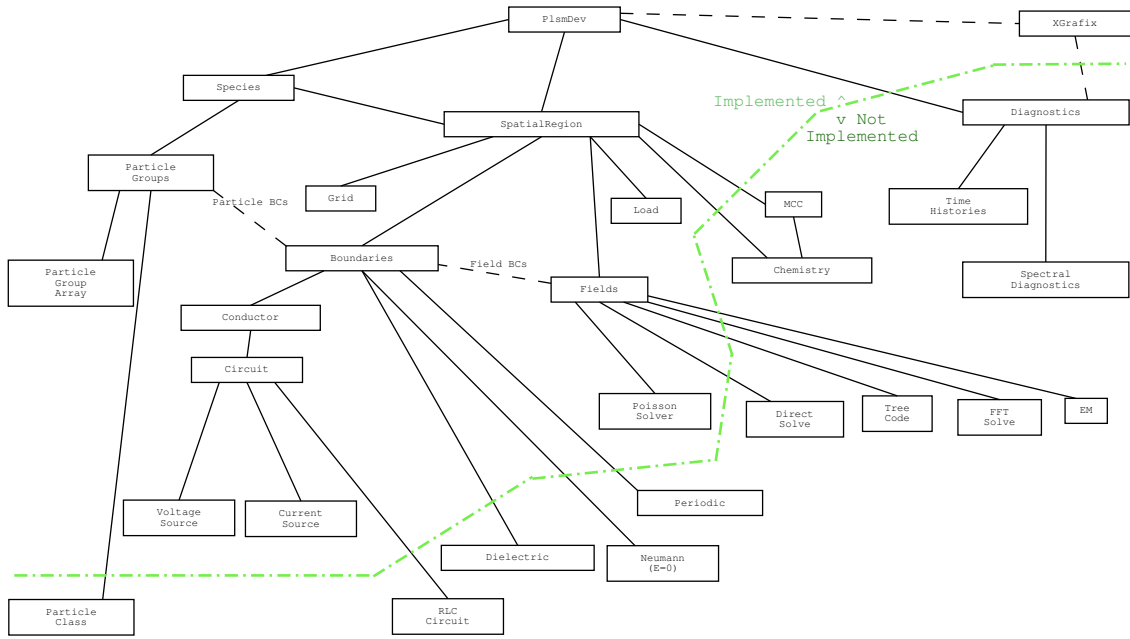


Figure 1: Existing and planned class (object) hierarchy for oopd1

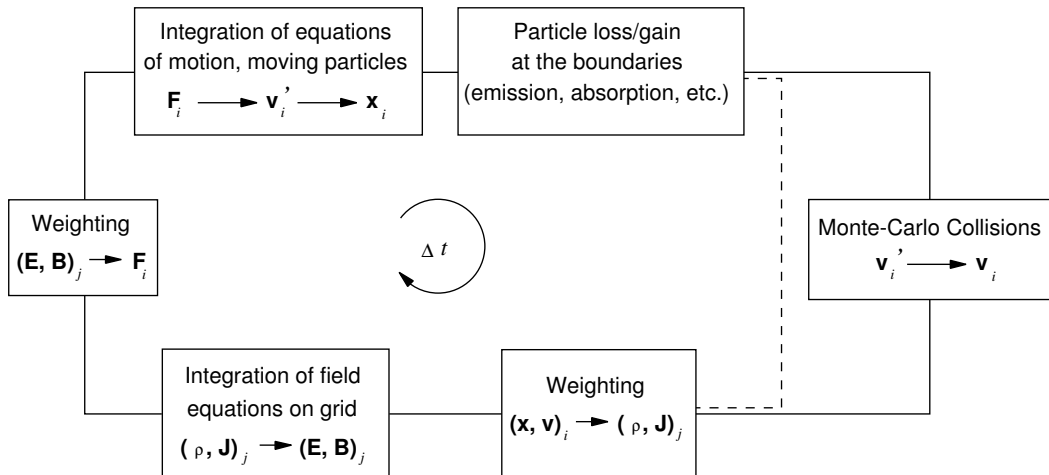


Figure 2: Flow chart for particle in cell. Monte Carlo collisions were not used in this project.

Gauss's law then becomes:

$$\nabla \cdot (\epsilon \mathbf{E}) = -\nabla \cdot (\epsilon \nabla \Phi) = \rho \quad (2)$$

In this equation ρ is charge density, obtained from particle data (for a single species of particles, $\rho = qn$, where q is the charge per particle and n is the number density of the species).

For meshes with a uniform value of ϵ , the Laplacian operator may be used:

Planar:

$$\nabla^2 \Phi = \frac{d^2 \Phi}{dx^2} \quad (3)$$

Cylindrical:

$$\nabla^2 \Phi = \frac{1}{r} \frac{d}{dr} \left(r \frac{d\Phi}{dr} \right) = \frac{d^2 \Phi}{dr^2} + \frac{1}{r} \frac{d\Phi}{dr} \quad (4)$$

Spherical:

$$\nabla^2 \Phi = \frac{1}{r^2} \frac{d}{dr} \left(r^2 \frac{d\Phi}{dr} \right) = \frac{1}{r} \frac{d^2}{dr^2} (r\Phi) = \frac{d^2 \Phi}{dr^2} + \frac{2}{r} \frac{d\Phi}{dr} \quad (5)$$

In finite difference form for a uniform mesh:

Planar:

$$\nabla^2 \Phi_i = \frac{\Phi_{i-1} - 2\Phi_i + \Phi_{i+1}}{\Delta x^2} + O(\Delta x^2) \quad (6)$$

Cylindrical:

$$\nabla^2 \Phi_i = \frac{\Phi_{i-1} - 2\Phi_i + \Phi_{i+1}}{\Delta r^2} + \frac{1}{r_i} \frac{\Phi_{i+1} - \Phi_{i-1}}{2\Delta r} + O(\Delta r^2) \quad (7)$$

Spherical:

$$\nabla^2 \Phi_i = \frac{\Phi_{i-1} - 2\Phi_i + \Phi_{i+1}}{\Delta r^2} + \frac{2}{r_i} \frac{\Phi_{i+1} - \Phi_{i-1}}{2\Delta r} + O(\Delta r^2) \quad (8)$$

For a mesh with variable ϵ , the left hand side of Poisson's equation is more complex:

$$\nabla \cdot (\epsilon \nabla \Phi) = -\rho \quad (9)$$

In one (radial) dimension, the gradient operator is of the same form regardless of coordinate system (Cartesian, cylindrical, spherical):

$$-(E_x) = \nabla \Phi \Rightarrow \frac{d\Phi}{dx} = \frac{\Phi_{i+1} - \Phi_{i-1}}{2\Delta x} + O(\Delta x^2) \quad (10)$$

For one-dimensional planar meshes with uniform permivity ϵ_0 , as studied here, the solution to Poisson's equation yields a tridiagonal matrix where the potential at node i may be solved for in terms of nodes $i + 1$ and $i - 1$.

$$\Phi_{i-1} - 2\Phi_i + \Phi_{i+1} = -\frac{\rho_i (\Delta x)^2}{\epsilon_0} \quad (11)$$

ρ_i is the charge density at node i and is obtained from particle data. Δx is the mesh spacing. Boundary conditions are needed to close the system at the mesh points. In this study, ideal conductors were used (Dirichlet boundary conditions). `oopd1` deals with all three 1d coordinate systems as described above with constant and variable permivity as well several different types of boundary conditions.

2.3 Particle Push

The coordinates, x , and velocities, v , of the particles are initially prescribed. If only the force due to the electric field is considered, the non-relativistic equations of motion for particle j are

$$\frac{dx_j}{dt} = v_j \quad (12)$$

$$\frac{dv_j}{dt} = \frac{qE(x_j)}{m} \quad (13)$$

x_j and v_j are the x -coordinate and component of velocity, respectively. q and m are the charge and mass of particle j (respectively), and $E(x_j)$ is the x -component of electric field at particle j .

Linear interpolation is used to find the electric field at particle j from the nodal values:

$$E(x_j) \equiv E_{i+1} \cdot \frac{x_j - x_i}{x_{i+1} - x_i} + E_i \cdot \left(1 - \frac{x_j - x_i}{x_{i+1} - x_i}\right) \quad (14)$$

The particle is assumed to be located between node i and $i + 1$ in this equation.

A discrete scheme used in PIC to advance the equations of motion is as follows:

$$v_{new} = v_{old} + \frac{qE}{m} \cdot (\Delta t) \quad (15)$$

$$x_{new} = x_{old} + v_{new} \cdot (\Delta t) \quad (16)$$

Δt is the time step used. If the position and velocity components are known at the same time, this is (forward) Euler's method (1st order accurate). If the velocity is staggered half a time step with respect to position, this is the "leap-frog" (midpoint rule) method (2nd order accurate). The latter is accomplished by initially advancing the velocity of all particles half a time step backwards using the initial field.

Once the new position of particle j is known, its charge is weighed to the mesh using the inverse of the formula used to gather the electric field to the particle (Equation 14). The resulting charge density is then used in the Poisson solve.

2.4 Accuracy Requirements for PIC

The basic discretization parameters for (explicit) PIC are mesh spacing, time step, and the number of particles per cell. For electrostatic PIC, the mesh spacing, Δx , and time step, Δt , must be determined from Debye length, λ_D , and the plasma frequency, ω_p , respectively, for accuracy and stability:

$$\Delta x < \lambda_D = \left(\frac{\epsilon_0 T}{en}\right)^{1/2} \quad (17)$$

$$(\Delta t)^{-1} \gg \omega_p = \left(\frac{e^2 n}{\epsilon_0 m}\right)^{1/2} \quad (18)$$

In these equations, T is the plasma temperature in volts, n is the plasma density, ϵ_0 is the permittivity of free space, e is the elementary charge, and m is the mass of the lightest species.

Violation of these conditions will result in inaccurate and possibly unstable solutions. Typically, $\Delta x \approx 0.5\lambda_D$ and $\Delta t \approx 0.1\omega_p^{-1}$.

3 Method of Parallelization

3.1 Overview of Parallelization of `oopd1` and `simpic`

For parallelizing `oopd1` and `simpic` using MPI, the two chief components of PIC computation time, the field solve and the particle push (and weighting), must be implemented in a manner appropriate to distributed computing systems. The particles may be distributed across processors with or without regard to spatial location. The field solve may be split up using and MPI reduce call for summing the charge density

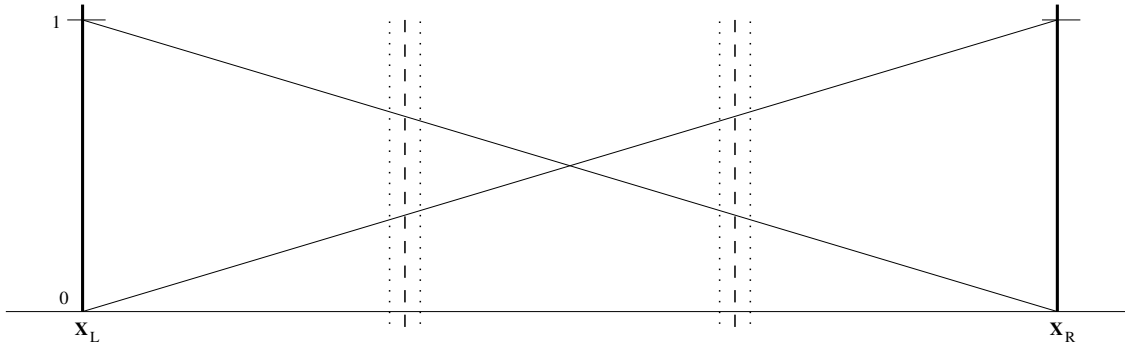


Figure 3: Solution of Laplace’s equation for a planar system with the conductors at x_L and x_R each biased to unity potential with the other held at zero. Three *SpatialRegions* are shown with boundaries between them denoted by dashed lines. Dotted lines denote the “ghost” nodes used in the solution of Poisson’s equation.

across processors or the solution to Poisson’s equation may be split using linear superposition with an MPI reduce call. Alternatively, the tridiagonal matrix may be solved in parallel.

In `simpic` (and, in the future, `oopd1`), the system – both particles and fields – is parallelized spatially. In `oopd1`, these are *SpatialRegion* classes. Particles are associated with a *SpatialRegion* and the electrostatic fields are computed using the principle of linear superposition.

Having particles distributed independently of spatial location is an advantage in terms of load balancing. While some load balancing must be done as particles may be added (due to injection or ionization events) or removed (due to absorption at boundaries), it is minimal and simple to implement. However if the particles are distributed spatially using constant spatial regions, some regions may exist with low plasma density resulting in low particle number and poor load balancing. This may be offset with dynamic load balancing, which is planned for `oopd1` by shifting cells and particles between processors but is beyond the scope of `simpic` and this CS267 project. In addition to the considerations for parallelization of the field solve and particle push, it is advantageous to distribute the simulation spatially in terms of convenience of computing Monte Carlo collisions, chemistry, diagnostic and other quantities of interest in PIC.

Using superposition instead of a global tridiagonal solve results in less communication for reasonable mesh size. In the straightforward implementation of the superposition method, two (scalar) values per processor are communicated to all other processors. Using a global tridiagonal solve, the number of scalars communicated without spatial division of particles is the number of total cells, and the number of scalars communicated if the particles are spatially divided is the number of cells per processor. So less MPI communication is done using the reduce call for the method of linear superposition as long as there are more than two cells per processor, which should be the case for problem sizes large enough to be run in parallel. The importance of this reduction in communication time may not be considerable if there are many particles per cell. Using a parallelized tridiagonal solve may result in similar or better parallel performance than using superposition, which will be investigated and implemented in future study.

3.2 Parallelization of Poisson’s equation using Superposition

If the permittivity is constant between two conductors, the electrostatic equation reduces to

$$\nabla^2 \Phi = \frac{\rho}{\epsilon} \quad (19)$$

In this case, the total potential may be written as the superposition of the driving potential (the potential due to the charge on the bounding electrodes) as well as the sum of the potentials due to various groups of charge sources with the electrodes grounded ($\Phi = 0$).

$$\Phi(x) = \sum_j \Phi_{j(drive)}(x) + \sum_i \Phi_{i(charge)}(x) \quad (20)$$

In Equation 20, the j summation is over electrodes (usually two) and the i summation is over the different groups of charges. In `oopd1`, particles are divided based on *SpatialRegions* (an `oopd1` class). The sum is

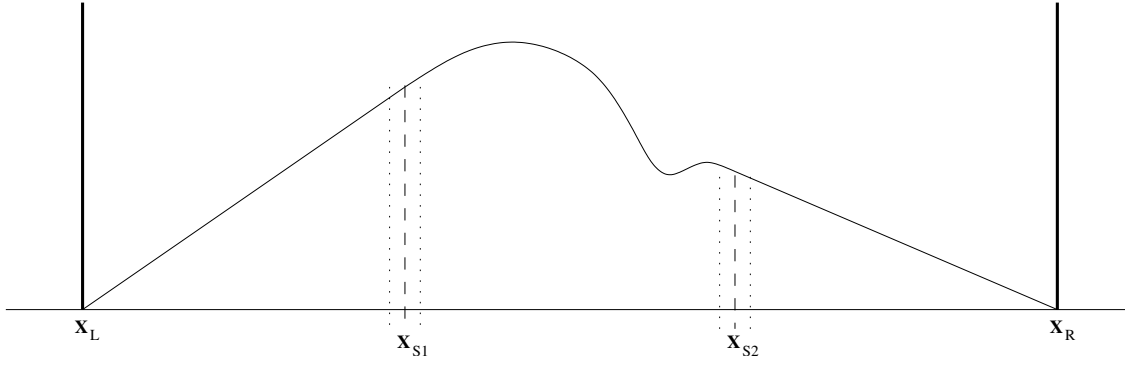


Figure 4: Solution of Poisson's equation for a planar system using a non-zero charge density in the *SpatialRegion* between x_{S1} and x_{S2} with the grounded conductors at x_L and x_R .

then over the different *SpatialRegions*. The potential fields due to conductor boundaries is shown in Figure 3. The potential field due to a charge distribution within a *SpatialRegion* is shown in Figure 4.

Since the Laplace (vacuum) solution is known from geometry, the value of the driving potentials is sufficient to find their contribution to the electrostatic potential within a *SpatialRegion*.

The behavior of the field outside of a *SpatialRegion* for solution of Poisson's equation is known from the potential non the *SpatialRegion* boundary and the geometry dependent behavior of the fields in vacuum.

Planar systems:

$$\frac{d^2 \Phi}{dx^2} = 0 \quad (21)$$

The potential is then linear outside the *SpatialRegion* of interest. For node index i , ghost nodes $i - 1$ and $i + 1$ are introduced for i on a left or right side *SpatialRegion* boundary, respectively.

For node i on the left hand side (see Figure 4):

$$\Phi_{i-1} = \Phi_i - \frac{h_i \Phi_i}{x_{s1} - x_L} = \left[1 - \frac{h_i}{x_{s1} - x_L} \right] \Phi_i \quad (22)$$

In this equation, h_i is the mesh spacing between x_i and x_{i+1} .

For the right hand side:

$$\Phi_{i+1} = \left[1 - \frac{h_i}{x_R - x_{s2}} \right] \Phi_i \quad (23)$$

Computational Procedure:

1. Move particles using existing field
2. Communicate particles moved across *SpatialRegion* boundaries to neighboring nodes
3. Weight particles to grid
4. Solve Poisson's equation using appropriate boundary conditions
5. Update circuit
6. Broadcast values of calculated Φ from each processor to each processor (within two electrodes)
7. Update local Φ values using superposition
8. Communicate local Φ 's to neighboring processors to calculate E -field
9. Calculate E -field

3.3 Parallelization of PIC Particle Push

Particles used to represent plasma phase space in Particle-In-Cell (PIC) codes are associated with the *SpatialRegion* which physically contains it. When a particle strikes the boundary of the *SpatialRegion*, the needed particle data is stored in a buffer of particle data partitioned by species. The particle data is then deleted from the local processor. After moving all particles `MPI_Isend` is used to asynchronously communicate particle data between neighboring processors. While waiting for the MPI communication to complete, the local particles are weighted to the mesh for use in the Poisson solve. The use of asynchronous communication and overlapping work should give a speedup factor near the number of processors for the particle work associated with the PIC methodology.

4 Implementation of Parallel PIC

Instead of parallelizing `oopd1`, a complicated code, a test code `simpic` was written and parallelized. `simpic` is a planar 1d-1v electrostatic PIC code that distributes work via spatial regions. Particles are communicated to other processors when they leave the spatial region, or are absorbed by electrodes at $x = 0$ and $x = L$. Fields are computed using Laplace splitting.

In this simplified implementation, the performance of the MPI particle in cell may be described in terms of three free parameters:

Number of processors, np : variations in relative performance with this parameter illustrate scalability of the problem

Number of particles per cell, ppc : higher values of this parameter indicate a particle-heavy simulation (a high ratio of particle computation time to field computation time). In practice, this number is based on the requirements to adequately resolve the kinetic distribution function of the particle species and ranges from less than ten for cold laminar simulations to greater than one thousand for detailed resolution of the high-velocity “tail” of the distribution function. The values of ppc reported, unless clear from context or otherwise noted, is the value at the start of the simulation.

Mesh spacing, Δx : the mesh spacing is inversely proportional to the number of cells, nc : $\Delta x \propto 1/nc$.

For ease of implementation in `simpic`, the number of cells per processor ($ncpp$) was used in lieu of Δx . The relationship between them is

$$\Delta x = \frac{L}{np \cdot ncpp} \quad (24)$$

The following parameters were considered constant:

- System length, $L = 1$ m
- Initial particle density, $n = 10^{13}$ m⁻³
- Cross sectional area, $A = 1$ m²

The total computational work per timestep, $W_{\Delta t}$, may be expressed in terms of the free parameters:

$$W_{\Delta t} = K_p \cdot ppc \cdot nc + K_c \cdot nc \quad (25)$$

K_p is the number of flops per (computational) particle and K_c is the number of flops per cell in the field solve.

5 Results

5.1 Description of Test Problem

The test problem used for profiling `simpic` simulates the decay of non-neutral positron space charge to bounding electrodes at $x = 0$ and $x = L$. Particles were loaded uniformly with zero initial velocity (cold plasma). Field splitting via the principle of superposition was tested and confirmed to give the analytic

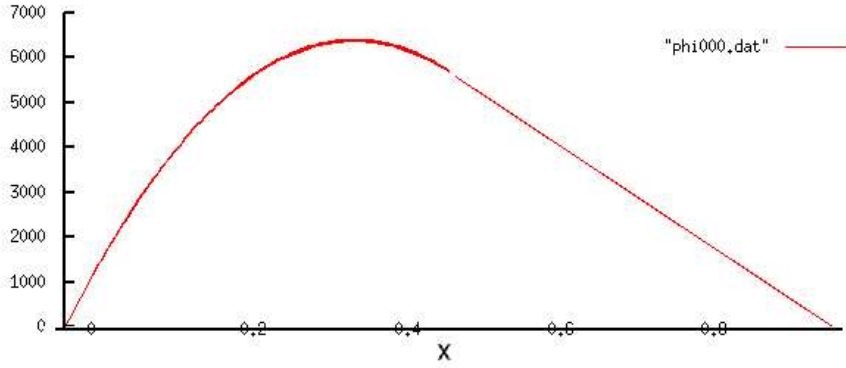


Figure 5: Calculated Poisson solution with grounded conductors and a uniform distribution of charged particles on processor 0 of 2 (left hand side). The boundary condition used is that in Equation 23. The right side is the mirrored symmetry.

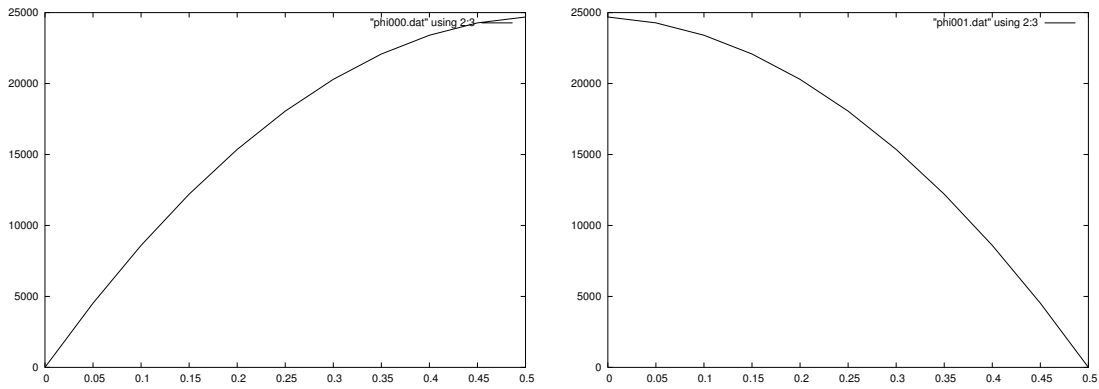


Figure 6: Potential fields on a two processor system after superposition of fields from the other processor.

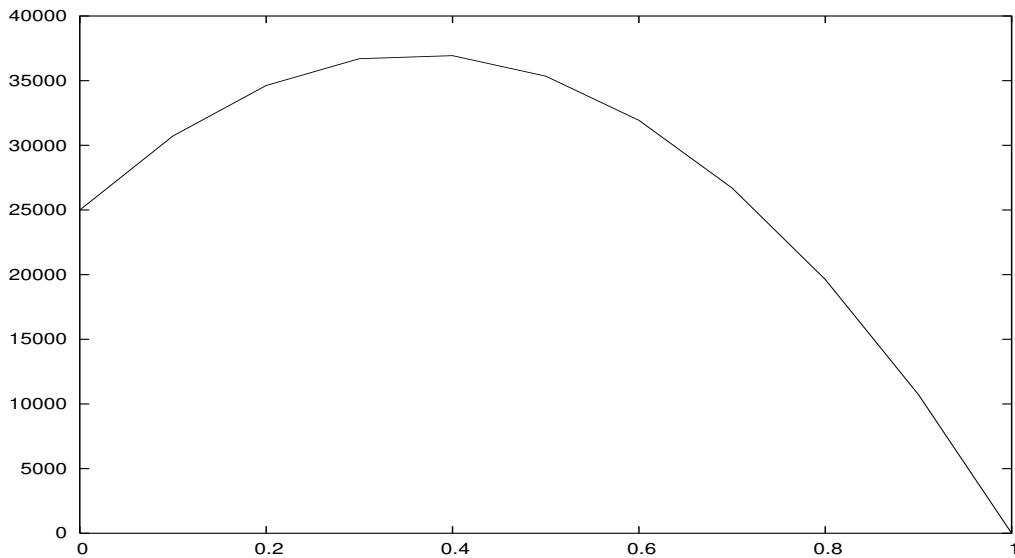


Figure 7: Potential after superposition including a 25000 V drive at the left hand side conductor.

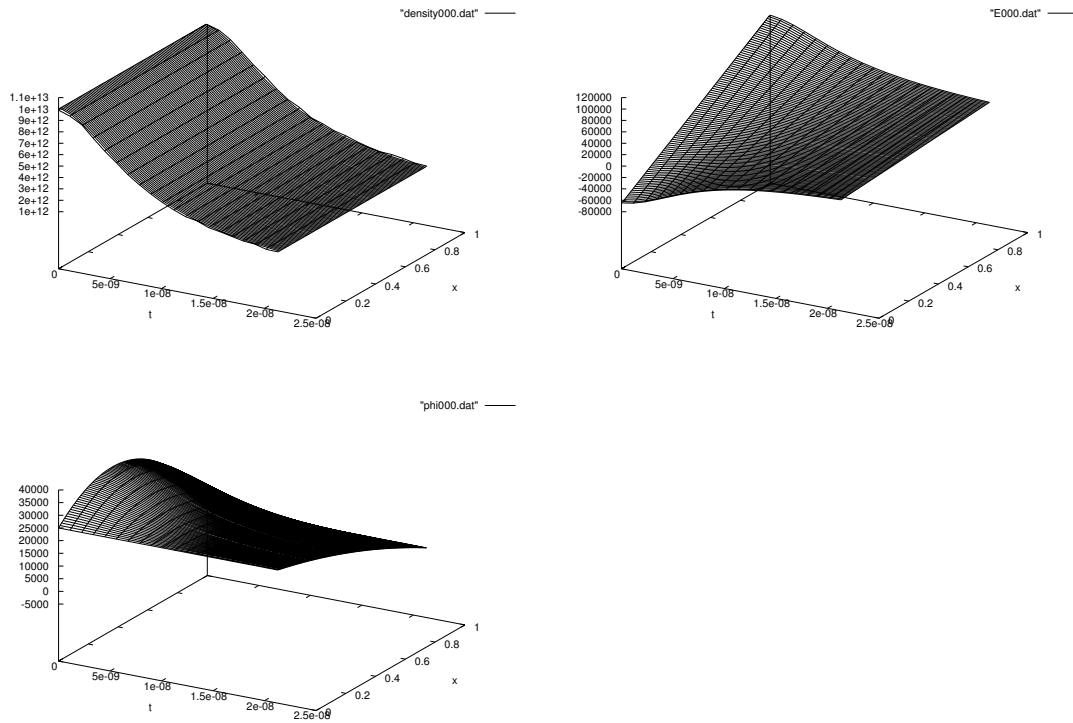


Figure 8: Diagnostics of the test problem as produced by `simpic` run on a single processor: density (upper left), electric field (upper right), and potential (lower left). The quantities are shown as functions of position and time.

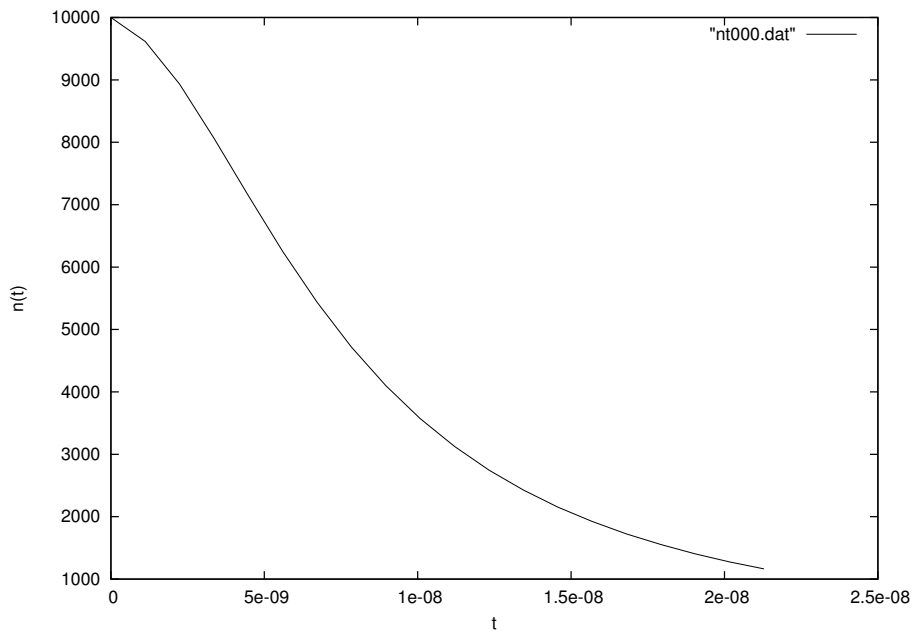


Figure 9: Computational particle number as a function of time for the test case.

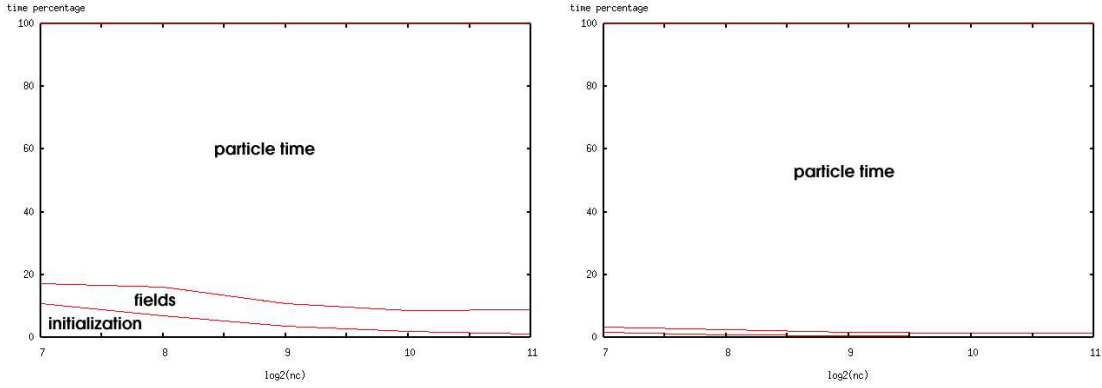


Figure 10: Single processor distribution of time for $ppc = 10$ (left) and $ppc = 100$ (right) for $\Delta t = 0.001w_p^{-1}$. In both cases, particle time dominates.

answer, as shown in Figures 5, 6, and 7. The left hand side driving voltage, 25000 V, was chosen to be comparable to the potential due to the space charge. This was done to introduce an asymmetry to the problem to obtain timings more indicative of problems of practical interest.

Diagnostics of interest showing the physics of the test run were produced on a single processor using 100 particles per cell in 100 cells run for 20 timesteps ($\Delta t = 0.2w_p^{-1}$). Results can be seen in Figures 8 and 9.

5.2 Timings

The fast (1.3GHz) nodes were used on the millennium cluster for the time trials. For each trial, the timings were taken from the overall fastest of five runs. This was done to help reduce the effect of contention and other biases. `MPI_Wtime` is used to obtain the timings and derived quantities.

Example command line for this test:

```
mpirun -np 2 a.out -ppc 100 -ncpp 128 -nt 200 -dtfactor 0.001 -lhsv 25000
```

Only one and two processors were used due to time constraints. The total number of cells ranged from 128 (10^7) to 2048 (10^{11}) in multiples of two. Two values of the initial number of particles per cell were used – a particle-light run, with 10 particles per cell; and a particle-heavy run, with 100 particles per cell. The simulation was run for 200 timesteps to attempt to amortize initialization time.

Two different sized time-steps were used: $\Delta t = 0.1/w_p$ and $\Delta t = 0.001/w_p$. Using the smaller time-step minimizes MPI particle communication, as fewer particles will pass the spatially divided processor boundaries and should keep the system more load balanced as well. In the course of this simulation, less than ten percent of the particles leave the system. The larger time-step puts a greater emphasis on communication and a greater load imbalance occurs in the course of the simulation. Using the larger time-step results in the complete decay of the position space charge to the walls in the course of the simulation.

The distribution of time for the different processes – program initialization, field solve, particle push and weighting, and MPI time – is graph in partition for one and two processors for the various tests. The computational efficiency of the two processor case is shown in Figure 14.

6 Discussion and Conclusions

From the preliminary work on two processors, decent speed-up is observed even for moderate numbers of cells per processor, especially if the simulation is particle heavy. It is likely that the actual speedup scaling observed will fall between the best case ($ppc = 100$, $\Delta t = 0.001/w_p$) and worst case ($ppc = 10$, $\Delta t = 0.1/w_p$) observed in this study for PIC problems of practical interest, and may follow the scaling for the two intermediary test problems. It is noted that for 512-2048 cells, nearly identical efficiencies are observed for the $ppc = 10$, $\Delta t = 0.001/w_p$ and $ppc = 100$, $\Delta t = 0.1/w_p$.

It is noted that imbalance bias introduced in the $\Delta t = 0.1/w_p$ cases is considerable, because the non-neutral space charge has no opposite charge to stabilize the rapid decay.

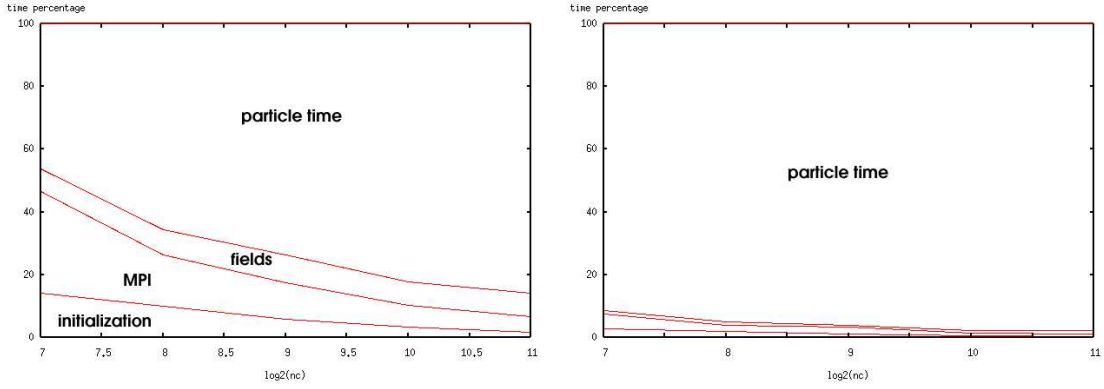


Figure 11: Two processor distribution of time for $ppc = 10$ (left) and $ppc = 100$ (right) for $\Delta t = 0.001w_p^{-1}$.

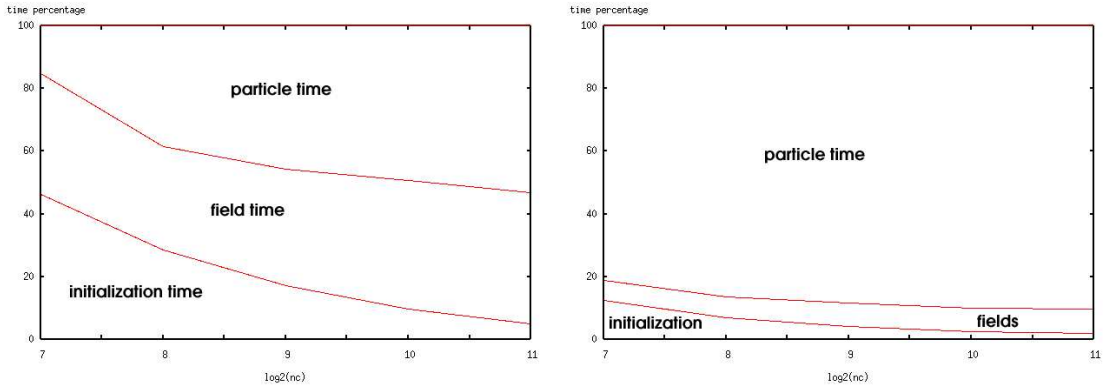


Figure 12: Single processor distribution of time for $ppc = 10$ (left) and $ppc = 100$ (right) for $\Delta t = 0.1w_p^{-1}$. In both cases, particle time dominates.

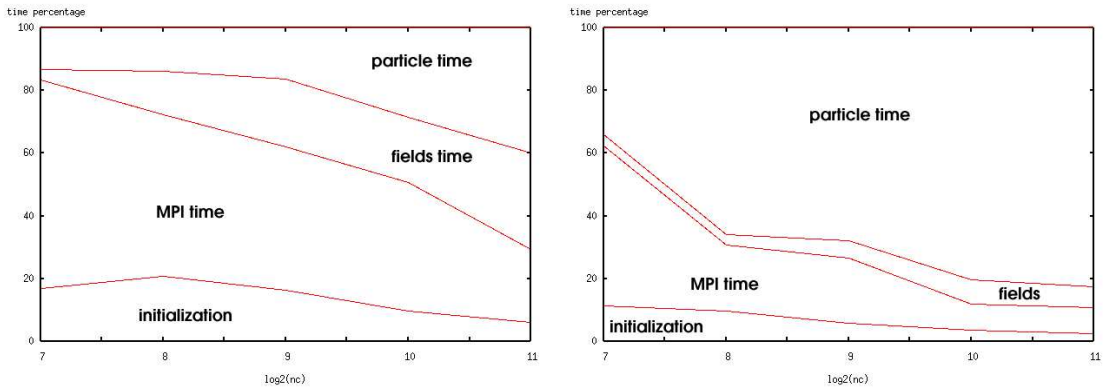


Figure 13: Two processor distribution of time for $ppc = 10$ (left) and $ppc = 100$ (right) for $\Delta t = 0.1w_p^{-1}$.

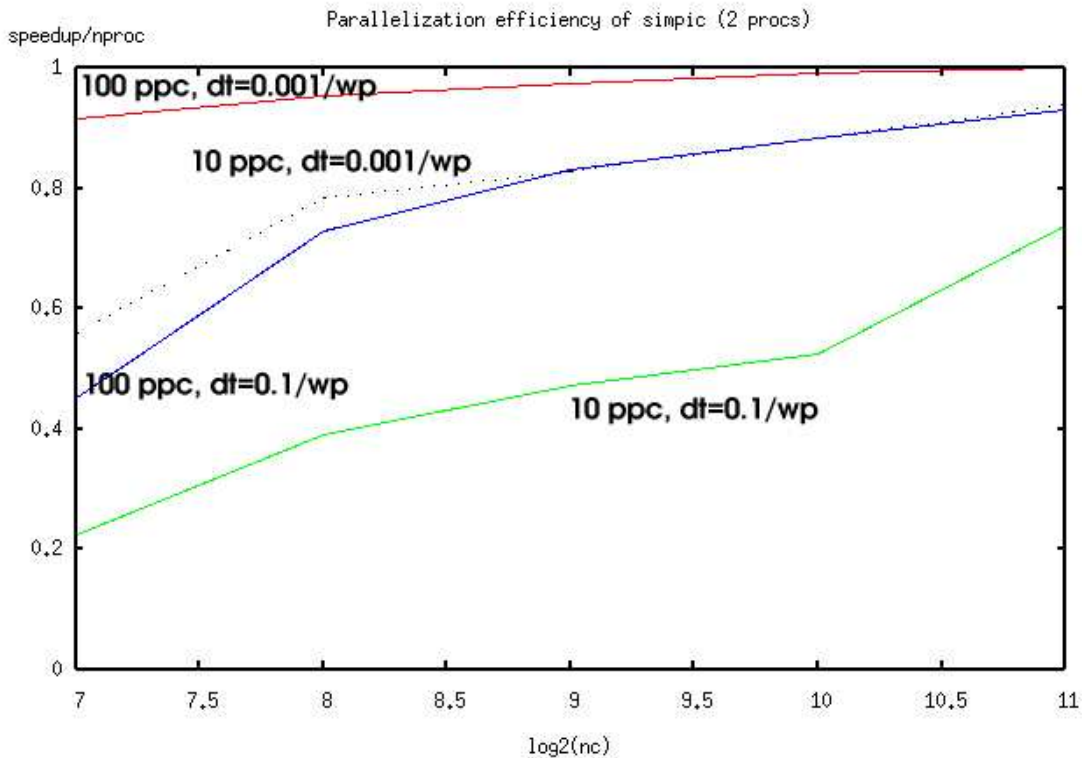


Figure 14: Computational efficiency for the four different test cases versus the problem size scale, $\log_2(nc)$. For two processors, efficiency greater than 0.5 indicates a net speedup.

For future work, the efficiency of parallelization should be investigated on higher numbers of processors. Following this, the methodology implemented in `simpic` may be migrated to `oopdl` with respect to lessons learned during parallelization and testing.

7 References

- Birdsall, C. K. and Langdon, A. B., Plasma Physics via Computer Simulation, Hilger, 1991.
- John David Jackson, Classical Electrodynamics, John Wiley, NY, 1999.
- Lieberman, M. A. and Lichtenberg, A. J., Principles of Plasma Discharges and Materials Processing, John Wiley, NY, 1994.
- Vahedi, V. and Surendra, M., "A Monte Carlo collision model for the particle in cell method: applications to argon and oxygen discharges", *Computer Physics Communications*, 87, pp179-198, 1995.
- Vahedi, V. and DiPeso, G., "Simultaneous Potential and Circuit Solution for Two-Dimensional Bounded Plasma Simulation Codes", *Journal of Computational Physics*, 131, 1997, June, pp149-163.
- John P. Verboncoeur and M. V. Alves and V. Vahedi and C. K. Birdsall, "Simultaneous Potential and Circuit Solution for 1d bounded Plasma Particle Simulation Codes", *Journal of Computational Physics*, 104, 1993, February, pp321-328.
- John P. Verboncoeur and Langdon, A. B. and Gladd, N. T., "An object-oriented electromagnetic PIC code", *Computer Physics Communications*, 1995, 87, pp199-211.
- Kevin J. Bowers, "High Frequency Electron Resonances and Surface Waves in Unmagnetized Bounded Plasmas", UC Berkeley Ph. D. Thesis, 2001.
- David J. Cooperberg, "Modeling and Simulation of High Frequency Surface Waves in Bounded Plasmas", UC Berkeley Ph. D. Thesis, 1998.
- Vahedi, V., "Modeling and Simulation of Rf Discharges Used for Plasma Processing", UC Berkeley Ph. D. Thesis, 1993.