Contents

1 Verification and Performance **1**

1.1 Verification 1

1.1.1 Error Quantification 1

1.2 Multigrid Solver 1

1.2.1 Analytical Solutions 2

1.2.2 Random Charge distribution 4

1.2.3 Additional Tests 4

1.2.4 ND vs 3D algorithms 4

1.3 Scaling of the error compared to discretization 4

1.4 Plasma Oscillation 7

1.4.1 Input parameters 7

1.5 Performance 9

1.6 Scaling 9

1.6.1 Perfomance Optimizer 11

1.6.2 Convergence Rate 11

1.6.3 Scaling of the MG Solver 11

1.6.4 Scaling properties PiC 12

Chapter 1

Verification and Performance

1.1 Verification

In this chapter we will go through different methods we used to verify the multigrid solver, as well as scaling measurements. Modular parts of the solver is tested with unittests where feasible. In addition the whole solver is tested with both analytically solvable test cases and randomly generated fields.

1.1.1 Error Quantification

In order to evaluate solutions we will primarily look at the normalized 2-norm of the error, eq. (1.1), and the residual, eq. (1.2). The $\|e\|_2$ is computed from comparing the numerical solution $\widehat{\phi}$ to an analytical solution ϕ and normalized with regards to grid points, N. The residual is found by inserting the numerical solution into the Poisson equation, the remaining part is then the residual, and shows the difference of the current numerical solution and the optimal numerical solution.

$$\|e\|_2 = \sqrt{\frac{\sum \left(\hat{\phi} - \phi\right)^2}{N}} \tag{1.1}$$

$$\bar{r} = \frac{1}{N}\left(\sum_i \nabla^2 \widehat{\phi}_i + \rho_i\right) \tag{1.2}$$

1.2 Multigrid Solver

To test the solver itself we employ a couple different techniques.

For the main test we use a charge distribution with a known analytical solution, and we then we compare the numerical solution to the analytical solution.

Since constructed solutions can often behave to nice, we will also perform a third test on a randomized charge distribution, here we will only look at the residual since we cannot compute the error due to not having an analytical solution. We expect the residual to approach 0.

Lastly we perform a few run on identical charge distributions, domain subdivisions and compare the solutions.

1.2.1 Analytical Solutions

We use a few different constructed charge density fields, which is analytically solvable, to test the performance and correctness of the solver. All the simulations here are ran on a grid of the size $128, 64, 64$ divided into $1, 2, 2$ subdomains, with PINC version 36ad. It uses 5 cycles when presmoothing, solving on the coarsest grid and postsmoothing, the MG solver is instructed to run for 100 MG V-cycles with 2 grid levels.

Sinusoidal function

A sinusoidal source term, ρ can be useful to test the solver since it can be constructed to have very simple derivatives and integrals. Here we use a sinusoidal function that has two positive tops and two negative tops over the total domain. We want the sinus function to go over 1 period over the domain, so we normalize the argument by dividing the grid point value, x_j, y_k, z_l, by the domain length in the direction, L_x, L_y, L_z.

$$\rho(x_j, y_j, z_l) = \sin\left(x_j \frac{2\pi}{L_x}\right) \sin\left(y_k \frac{2\pi}{L_y}\right) \tag{1.3}$$

A potential that fits with this is:

$$\phi(x, y, z) = -\left(\frac{2\pi}{L_x}\right)^2 \left(\frac{2\pi}{L_y}\right)^2 \sin\left(x_j \frac{2\pi}{L_x}\right) \sin\left(y_k \frac{2\pi}{L_y}\right) \tag{1.4}$$

The fig. 1.1 shows the results from running the MG-solver on the test sinusoidal test case described here. As can be expected the potential mirrors the charge distribution, except with an opposite sign and a larger amplitude. A decently large grid was simulated and the mean residual was found to be: $\bar{r} \approx 0.0312$.

Heaviside Function

The solver is also tested with a charge distribution governed by a Heaviside function. This is also suited to testing since the charge distribution is then

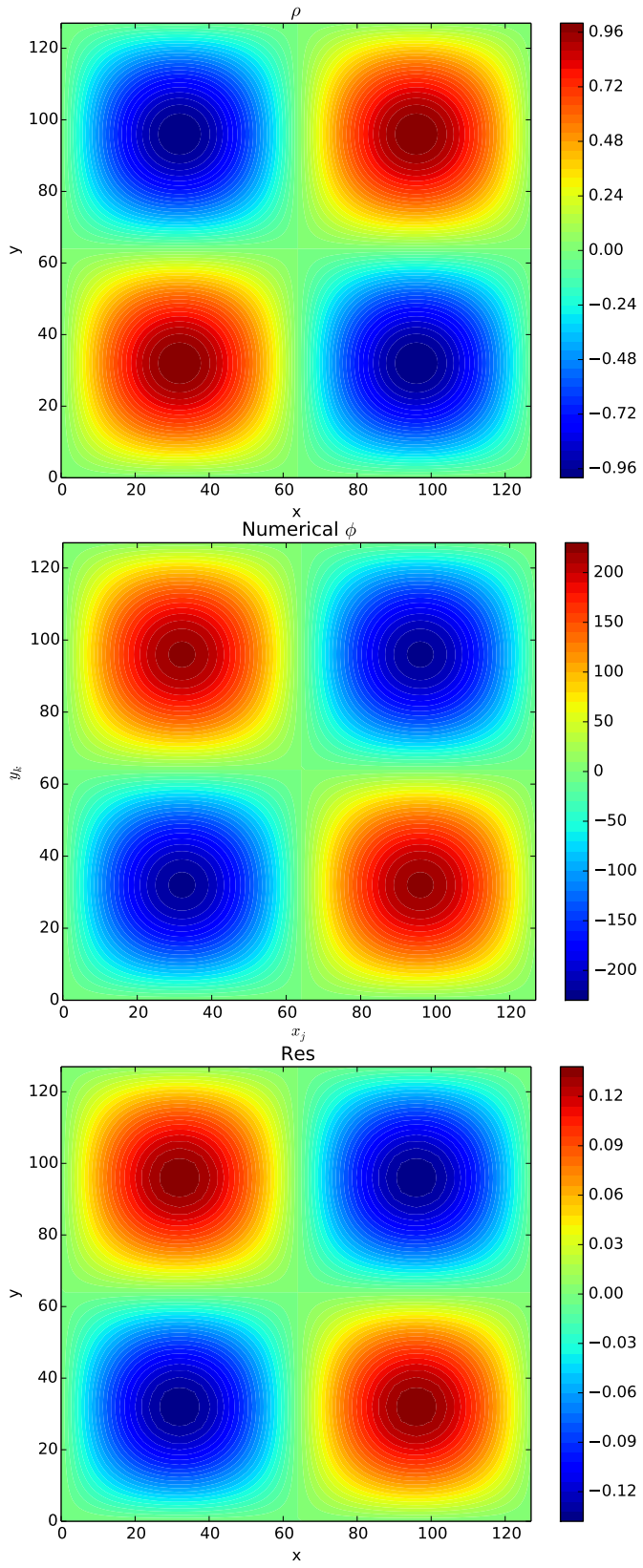

Figure 1.1: This a x, y-plane from the grids cut along $z_l = 32$, from the sinusoidal test case described in section 1.2.1. The top plot shows the charge distribution, the center plot shows the numerical solution of the potential and the bottom plot depicts the residual. All the units are in normalized dimensionless units.

constant planes, and we expect second order polynomial when integrating them. In the test case there are two planes with the value -1 and two planes with 1. In fig. 1.2 the test case, as well as the solution and residual is shown, and we can see the polynomials in the solution. The mean residual $\bar{r}$ was 0.00677.

$$\rho_(x_j, y_k, z_l) = \begin{cases} 1 & y_j \epsilon (0, 32), (64, 96) \\ -1 & y_j \epsilon (33, 65), (97, 127) \end{cases} \tag{1.5}$$

1.2.2 Random Charge distribution

To hopefully avoid some problems, that could appear due to the earlier test cases being to constructed being to orderly, a test with a randomized charge distribution is also included. The fig. 1.3 shows the charge distribution, numerical potential and the residual. The mean residual was found to be $\bar{r} \approx 0.00388$.

1.2.3 Additional Tests

In addition to the tests on shown in this section, we also ran the same tests obtaining similar results on various sizes and directions. Since we wanted to be sure that the program was working independently of the how the domain was divided into subdomains, we also performed tests on different subdomain divisions.

1.2.4 ND vs 3D algorithms

PINC is built to have two sets of algorithms, one N-dimensional and one 3-dimensional. The N-dimensional algorithms is meant to be used in cases where one want to do 1- and 2-dimensional simulations as well as a test for the 3-dimensional algorithm. The 3-dimensional algorithm is generally slightly faster than the N-dimensional due to some extra capabilities in hardcoding certain parts. On a laptop, using two 'Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz' processors, we ran the multigrid solver on a $128, 128, 128$ size problem, with both the 3D and ND algorithms. The ND algorithms used 80.08s to solve it to the given tolerance, while the 3D algorithms used 78.75s achieving only a slight speedup. This is likely due to most of the computational time being spent on the interprocessor communication.

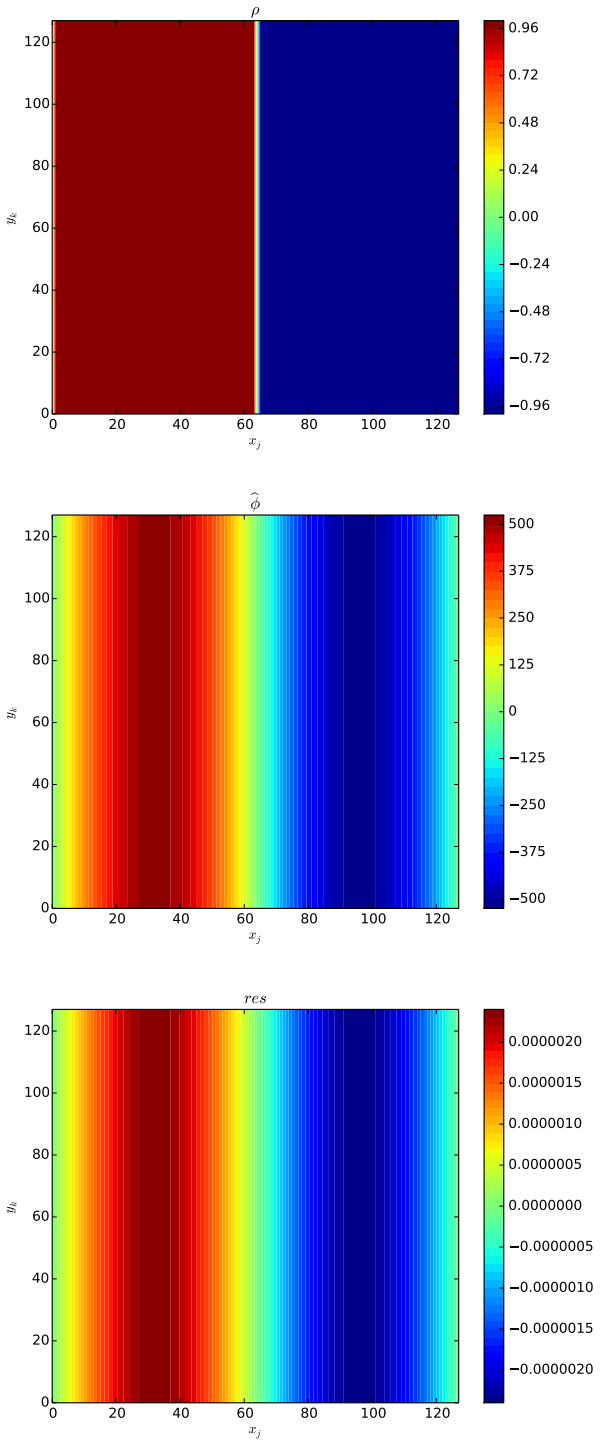

Figure 1.2: As earlier this is a x, y-plane cut along $x_k = 32$, of the grid. The plots show the charge distribution, numerical solution and the solution, from left to right. This is a test case constructed with Heaviside functions. In the solution of the potential the expected second degree polynomial can be seen.

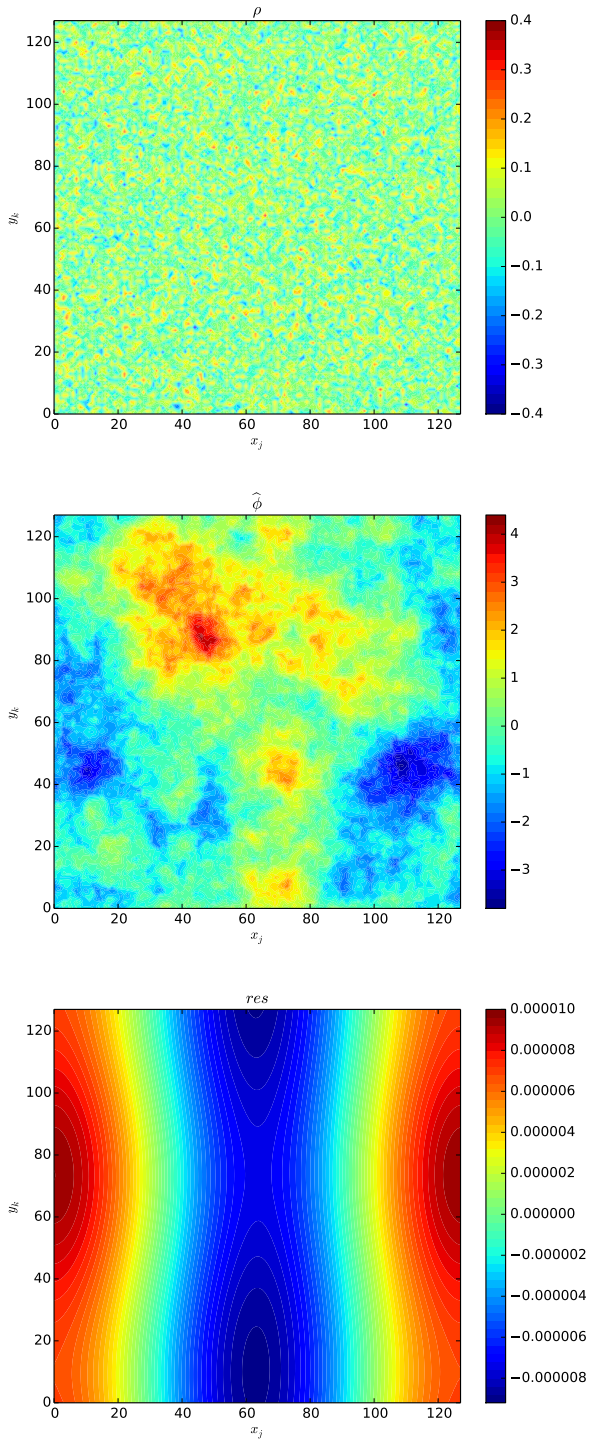

Figure 1.3: As earlier this is a x, y-plane cut along $x_k = 32$, of the grid. The plots show the charge distribution, numerical solution and the solution, from top to bottom. It is visible that the potential shows more structure than the ρ, since the integration smooths the original problem.

1.3 Scaling of the error compared to discretization

The charge density is represented by a discretized grid, due to this a numerical solution will have an inherent error. The error will be of second order, $\mathcal{O}(h^{-2})$, dependent on the stepssize, Δx, due to the first order solver, see **??**.

To investigate that the error the of the solver follows a second order improvement as the stepsize decreases we construct a sinusoidal *rho* as a test case.

$$\rho(x) = \sin\left(\frac{x}{2\pi}\right); \qquad x\epsilon[0, 2\pi] \tag{1.6}$$

This ρ is analytically solvable for the Poisson equation so we can compute the 2-norm of the error, $\|err\|_2$. Then we gradually decrease the stepsize and obtain and compare the norm of the numerical solutions. Since the normalization in PINC is normally done outside the multigrid solver, *rho* had to be suitably scaled to the stepsize. We expect the error to be proportional to the squared stepsize, $err(h) \approx Ch^2$, where C is a constant dependent on the geometry of the problem. By taking the logarithm we obtain

$$\log(err(h)) = 2\log(Ch) \tag{1.7}$$

fig. 1.4 shows the the measured error when solving the sinusoidal charge distribution, eq. (1.6), for both the potential, ϕ, and the electric field, E_x. The problem was solved with different discretizations on a 3-dimensional domain, starting at $[8, 8, 8]$ doubling the grid points each time. The slope on the logarithmic plots was in both cases found to be 2.00, showing a second order error scaling $\mathcal{O}(h^{-2})$. The same test was also performed in 1 and 2 dimensional cases, with varying subdomain configurations and with the sinus shape along the other axes.

1.4 Plasma Oscillation

As a test of the validity of our PiC program, we can use the Langmuir oscillation, described in **??**. This test is inspired by a case set up in Birdsall and Langdon (2004), and modified to fit with our normalization and discretizations. To verify that the oscillation is properly simulated we will look at oscillations in the energy.

1.4.1 Input parameters

Timestep

First we need to ensure that the simulation is does not violate the time- stability criterion, **??**, $\Delta t \leq 2\omega_{pe}$. For computational reasons each particle in the program

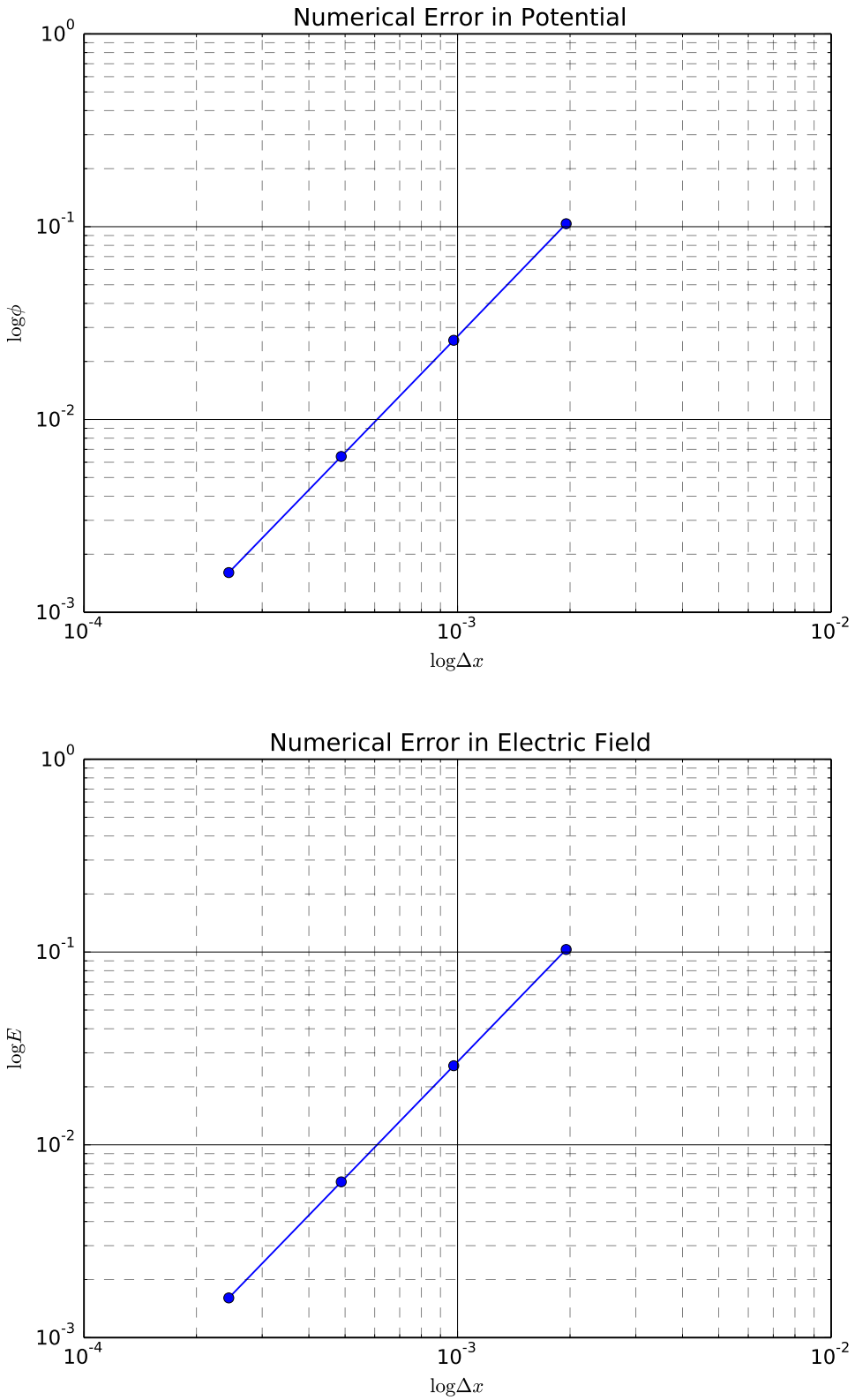

Figure 1.4: Logarithmic plot of the 2-norm of the error of the potential ϕ, top figure, and the x-component of the electric field. The solver was run on a scaled sinus-shaped charge distribution. Both of the plots show a straight line of the error, on the logarithmic plots, with a slope of 2.00. This corresponds to the error scaling with order -2 as a function of the stepsize. All the units are in PINC normalized units.

Size	Stepsize	#Particles per cell
$(64, 64, 64)$	0.2	

represents many real particles and we can adjust the number, of real particles represented, to ensure that the electron plasma frequency is 1.

$$\omega_{pe}^2 = \frac{nq_e^2}{m\epsilon_0} = \left(\frac{N}{V}\right)\left(\frac{q_{e*}}{m_{e*}}\right)q_{e*} \tag{1.8}$$

Here (N/V) is just the number of electron super-particles divided by the volume, q_e and m_e, represents super-particles and needs to be multiplied by the number of particles in a super-particle, κ.

$$\omega_{pe}^2 = \left(\frac{N}{V}\right)\left(\frac{e}{m_e}\right)\kappa e \tag{1.9}$$

Since we want the electron plasma frequency to be 1, κ is set to

$$\kappa = \frac{Vm_e}{Ne^2} \tag{1.10}$$

Now the time is given in units of ω_{pe} and we use $\Delta t = 0.2 < 2$ easily.

Spatial-step

The spatial step need to satisfy the finite grid instability condition, $\Delta x < \varsigma\lambda_{Se}$, **??**. We use a similar procedure as in normalizing the time with regards to ω_{pe} to normalize the length with regards to λ_{Se}. The we end up with the temperature, or its other representation as thermal velocity, as the parameter we can adjust to make sure that it satisfies the condition.

$$\lambda_{Se}^2 = \frac{\epsilon_0 kT_e}{nq_e^2} \tag{1.11}$$

Simulation

This simulation was run with on

1.5 Performance

1.6 Scaling

In this section we investigate the performance of the solver and different scaling measurements. We are interested in both how well the solver performs on a

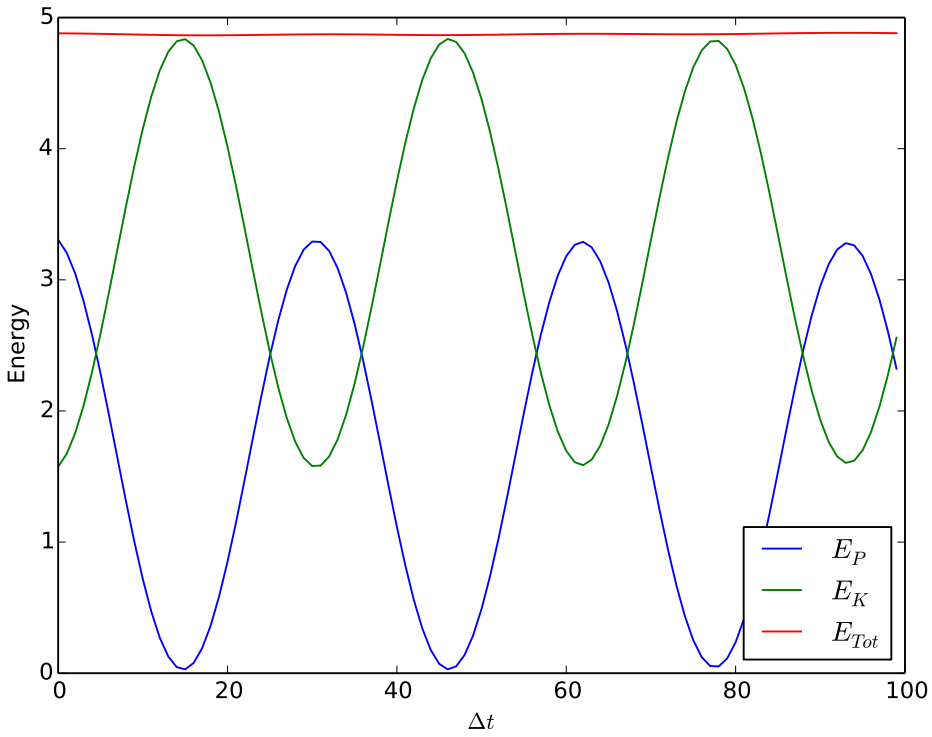

Figure 1.5: This shows the time-evolution of the energy in an perturbed plasma. The energies are in normalized units and $\Delta x = 0.1\omega_{pe}$. The total energy has a maximum variation of 0.22%. In the timespan of $10\omega_{pe}$ the plasma oscillates slightly more than 3 times.

larger number of processors, as well as the perfomance impact of the different parameters in the solver.

We want to obtain a better understanding of how the field resolution can be scaled up without hampering the perfomance of the particle-in-cell simulation to much.

1.6.1 Perfomance Optimizer

A multigrid solver has several parameters that needs to be set correctly for a an optimal performance (Find Source for!!!). These parameters are dependent on the problem size, as well as the computing architecture. Instead of attempting to estimate them beforehand we have included an external script that runs the program with different MG-solver settings on the wanted domainsize and tries to optimize them. The parameters it tries to change is the number of grid levels and the cycles to run for presmoothing, postsmoothing and the coarse solver. It should be worth it to spend some computing power, finding close to optimal settings, prior to running a full scale simulation since the solver needs to run each time step. The performance optimizer naively runs the solver for a predetermined mesh of settings.

1. Smarter testing algorithm 'if better when increase continue increasing, else stop'

OTE TO SELF: This didn't pan out very well

sectionProcessor scaling Now we want to investigate how well the solver is parallelized. In **??** the theoretical considerations for how it should scale is gone through.

1.6.2 Convergence Rate

As an iterative solver a multigrid solver will gradually approach a solution, reducing the residual further each run. In this subsection we will measure the convergence rate, defined in similar way as in Zhukov et al. (2014),

$$p = \left(\frac{r_m}{r_0}\right)^{1/m} \tag{1.12}$$

where r_m, r_0 are the 2-norm of a the residual after m multigrid runs and the inital residual. We presume that each run of the multigrid solver will remove a proportion of the remaining residual. The tests are done on a 128^3 grid on a sinusoidal problem, and the smoothers run for an equal number of runs on each level. (Table of convergence rate numbers) Zhukov et al. found convergence rates of $p \approx 0.15$ and $p \approx 0.17$ using a multigrid solver with a Chebyshev algorithm to smooth.

1.6.3 Scaling of the MG Solver

One of the aims of building a parallell multigrid solver was to be able to enable simulating large plasma problems. To be able to achieve that the solver should be able to scale up very well, i.e. doubling the problem size and the number of available processors should only give a manageable increase in computational time. We don't expect to be able to achieve a perfect parallelization, since there is a certain amount of interprocessor communications necessary that will slow down the algorithm compared to a sequential algorithm. The exact parallel performance is also dependent on the communications channels and the topology between the processor clusters. In **??** the parallel complexeties for the different multigrd algorithms is given and we will look at the parallel proprties for a V, W and FMG algorithm.

To investigate the scaling properties we will run set up a standard problem, and solve it with increasing resolutions. We start with a 64^3 grid on 1^3 computaional core, then we increase the problemsize to 128^3 on 2^3 and so on. These tests were run on (FIND ABEL SPECS). (PUT IN FIGURE OF SCALING (Processors/Time))

1.6.4 Scaling properties PiC

As well as the scaling properties of the multigrid solver a look at the whole PiC model is interesting. This helps us easier pinpoint where further optimizations efforts should be directed. We do not necessarilly except the the particle based algorithms to scale up to larger problems in the same rate as the solver. So above a certain number of processors, or size, there may be a bottleneck.

Bibliography

Birdsall, C. K. and A. B. Langdon (2004). Plasma Physics via Computer Simulation. en. CRC Press. ISBN: 978-1-4822-6306-0.

Zhukov, V. T. et al. (2014). "Parallel multigrid method for solving elliptic equations". en. In: Mathematical Models and Computer Simulations 6.4, pp. 425–434. ISSN: 2070-0482, 2070-0490. DOI: 10.1134/S2070048214040103. URL: http://link.springer.com/article/10.1134/S2070048214040103 (visited on 10/17/2016).