Non-Equidistant Particle-In-Cell for Ion Thruster Plumes

IEPC-2013-067

Presented at the 33rd International Electric Propulsion Conference,
The George Washington University, Washington, D.C., USA
October 6–10, 2013

K.F. Lüskow*, J. Duras†, O. Kalentev‡, K. Matyash§, J. Geiser¶ and R. Schneider‖
Ernst-Moritz-Arndt Universität Greifswald, Felix-Hausdorff-Straße 12, 17489 Greifswald, Germany

D. Tskhakaya **
Institute of Applied Physics, Vienna University of Technology,
Wiedner Hauptstrasse 8-10/E134, A-1040 Vienna, Austria

Particle-in-Cell (PIC) codes are standard tools for plasma simulations. Therefore, they are perfectly suited to simulate ion thrusters. Their restrictions of the cell size due to the resolution of the Debye length strongly limits the computationally possible size of the simulation domain. One way to overcome this shortcoming is to use non-equidistant grids, although such methods have problems with energy conservation. A first step for using non-equidistant PIC is an appropriate Poisson solver. A detailed discussion for this is presented.

Nomenclature

U	= voltage
$\lambda_{D,e}$	= electron Debye length
$X_p(t)$	= position of particle p at time t
$\omega_{p,e}$	= electron plasma frequency
V_p	= velocity of particle p
T	= temperature
n	= density
Δx_i	= distance between grid point x_i and x_{i+1}
ρ_i	= charge density assigned to grid point i
ϵ_0	= vacuum permittivity
E_i	= electric field in grid point i
N_G	= number of grid points
ϕ_i	= potential on grid point i
ω	= relaxation parameter for the SOR method

*Postgraduate, Ernst-Moritz-Arndt University of Greifswald, lueskow@physik.uni-greifswald.de.
†PhD student, Ernst-Moritz-Arndt Universität Greifswald, durasj@uni-greifswald.de.
‡Researcher, Ernst-Moritz-Arndt Universität Greifswald, okalenty@ipp.mpg.de.
§Researcher, Ernst-Moritz-Arndt Universität Greifswald, knm@ipp.mpg.de.
¶Researcher, Ernst-Moritz-Arndt Universität Greifswald, geiserj@uni-greifswald.de.
‖Full Professor, Ernst-Moritz-Arndt Universität Greifswald, schneider@uni-greifswald.de.
**Researcher, Vienna University of Technology, david.tskhakaya@uibk.ac.at

I. Introduction

PARTICLE-in-Cell (PIC) codes are standard tools for plasma simulations[1] and therefore ideal to simulate ion thrusters. Due to the non-Maxwellian characteristics of the plasma in ion thrusters fluid models are not sufficient. Fully kinetic models are needed. A typical 2D cylindrical domain for a High-Efficiency-Multistage-Plasma-Thruster (HEMP-T) simulation with calculated potential is shown in Fgure 1. At $z = 0$ an anode with $U = 500\,\mathrm{V}$ is placed, at $z = 89\,\mathrm{mm}$ a zero flux boundary condition is set. At $r = 24\,\mathrm{mm}$ the boundary condition is $\phi = 0\,\mathrm{V}$ for a grounded metal representing the outermost cylinder of the thruster[2]. The channel is limited by a dielectrics, which ends in a ring of grounded metal marked as shaded area in Figure 1. The thruster exit is located at $z = 49\,\mathrm{mm}$.

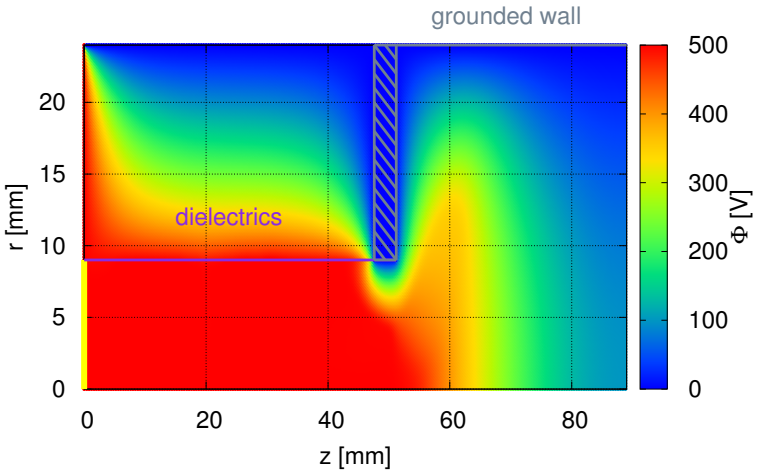

Figure 1. Calculated potential for a typical thruster geometry.

To improve the simulation in the plume region the size of the domain needs to be extended. In smaller domains the boundary conditions have a strong impact on the potential, which can cause numerical artifacts.

In the left part of Figure 2 the potential calculated with a small domain identical to the one from Figure 1 is shown. The size of the domain is 890×240 cells and includes channel and near-field plume region. In the right part of Figure 2 the domain is enlarged. Its size is 1270×480 cells with the same size for each cell as in the small domain. Especially in the plume region there are large differences observable between both solutions.

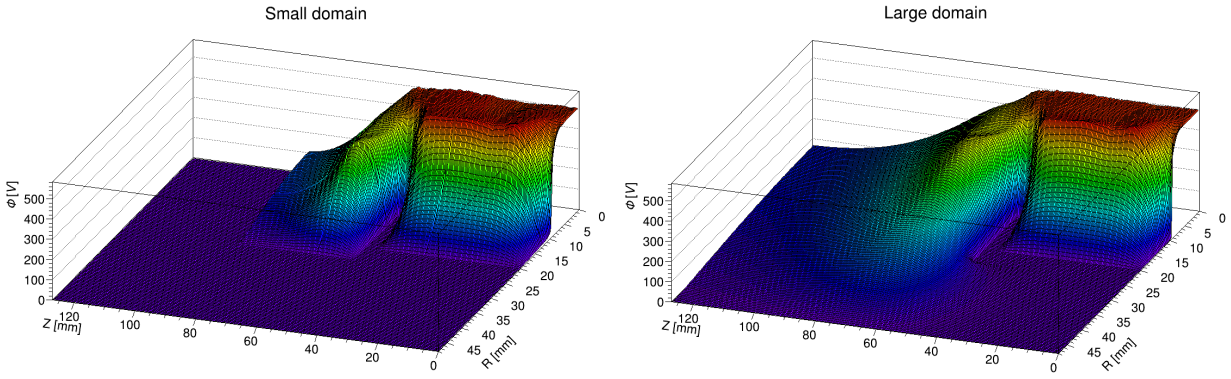

Figure 2. PIC potential solution for different domain sizes.

Whereas in the small domain the decrease in potential in the plume takes place very close to the exit driven by the boundary condition, the larger domain shows a much smoother decrease. At the borders of the smaller domain, where $\phi = 0\,\mathrm{V}$ is fixed, the large domain still shows potential values of several Volts. This difference results in stronger electric fields in the smaller domain compared to the larger one. The potential in the larger domain is less influenced by the boundary condition. An even larger domain is expected to

close the discrepancy between modeling and experiment showing up in the angular ion distribution[2].

A larger domain size leads to much longer computational times. PIC codes require cell sizes that resolve the Debye scale. Since for PIC the solver is the most costly part, a coarser grid can significantly reduce computational time. For plume simulations a non-equidistant grid is desirable, because the Debye scales of the channel and the plume are quite different. For electrons the Debye length in the thruster channel is $\lambda_{D,e} \approx 14.8\,\mu\text{m}$, whereas in the far-plume region the smallest lengths scale is $\lambda_{D,e} \approx 1.3\,\text{mm}$. Therefore, is should be possible to use larger cells in the plume and to extend the domain. In the far-plume, where one wants to use larger cells, there are only few space charges remaining, so that the potential solution is very close to the vacuum solution and a coarsening of the cell size should be acceptable.

The outline of the paper is as follows: After a short introduction into the problem of non-equidistant grids, different algorithms that can be used for a non-equidistant Poisson solver are presented and compared.

II. Problem of Non-equidistant Grids

In order to study effects of non-equidistant grids on the energy conservation a simple numerical test is presented. A single electron with an initial non-zero velocity is studied in a 1d3v PIC code using a non-equidistant grid. In the code, the electron is initialized at $X_e(t=0) = 4\lambda_{D,e}$, in a domain of $L = 10\lambda_{D,e}$ length. Since the Debye length $\lambda_{D,e}$ and the plasma frequency $\omega_{p,e}$ is only defined for many particle systems, in this set up it is just an arbitrary length scale, respectively time scale. The corresponding quantities were chosen as $T_e = 6\,\text{eV}$ and $n_e = 1 \times 10^{12}\,\text{cm}^{-3}$, which results in a length scale of $\lambda_{D,e} = 1.8 \times 10^{-5}\,\text{m}$ and a time scale of $\omega_{p,e} = 5.64 \times 10^{10}\,\text{s}^{-1}$. The initial velocity is set with $V_e^x = \lambda_{D,e}\omega_{p,e}$ and $V_e^y = V_e^z = 0$. The applied non-equidistant grid consists of two parts but with individually different cell sizes, where at $x = 5\lambda_{D,e}$, the cell size changes from $\Delta x_1 = 0.1\lambda_{D,e}$ to $\Delta x_2 = 5\Delta x_1$. The time step is chosen as $\Delta t = 0.002\omega_{\text{p,e}}^{-1}$. For the Poisson solver a three point central difference scheme for non equidistant grid

$$\frac{\phi_{i+1}\Delta x_{i-1} - \phi_i(\Delta x_{i-1} + \Delta x_i) + \phi_{i-1}\Delta x_i}{0.5(\Delta x_{i-1} \cdot \Delta x_i)(\Delta x_{i-1} + \Delta x_i)} = -\frac{1}{\varepsilon_0}\frac{\rho_i}{0.5(\Delta x_{i-1} + \Delta x_i)}$$

is applied. Here Δx_i is the length of the grid cell right to the grid point x_i which can be different from the cell size Δx_{i-1} to the left of x_i. For the boundary conditions $\phi_0 = 0$ and $\phi_{N_G} = (L - 2X_e)E$ is set, where E is the electric field generated by the electron.

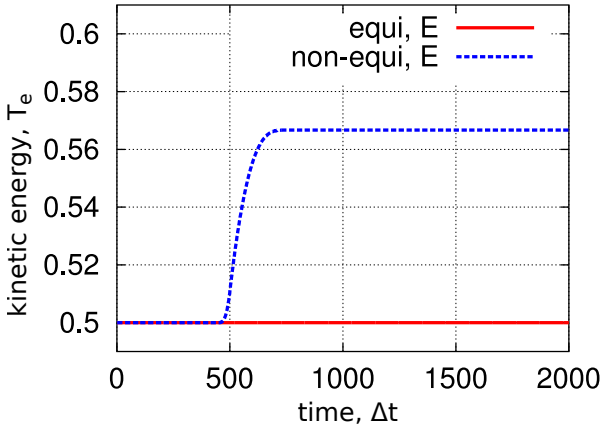

Figure 3. Kinetic energy of one electron in its own field as a function of time, for an equidistant grid (red solid line) and a non-equidistant grid (blue dashed line). The non-uniform grid boundary is reached at $\sim 500\Delta t$.

Figure 3 shows the evolution of the electron kinetic energy as a function of time, for a non-equidistant grid in comparison with an equidistant grid. One can clearly see that during the crossing of the grid separation boundary the inaccurate electric field pushes the electron. A self-force appears as artificial acceleration by crossing a cell size change from small cells to larger ones and therefore the kinetic energy is artificially changed. This happens in the time period, where the particle is assigned to grid points related to different cell sizes.

In the case of an equidistant grid the calculation of the electric field in an electrostatic 1D PIC code is typically done by a two-point central difference scheme. For a non-equidistant grid, it becomes

$$E_i = -\frac{\phi_{i+1} - \phi_{i-1}}{\Delta x_{i-1} + \Delta x_i} \ .$$

To avoid self forces, E_p has to be zero. For neighboring cells of the same size $\Delta x_{i-1} = \Delta x_i$, the potential ϕ_{i+1} and ϕ_{i-1} have the same values and the electric field is

$$E_p = \frac{\phi_{p+1} - \phi_{p-1}}{2\Delta x_{p-1}} = 0 \quad \text{for} \ \ \Delta x_{p-1} = \Delta x_p \ .$$

No self force is generated. But if $\Delta x_{i-1} \neq \Delta x_i$,the potential values are not equal and the resulting electric field differs from zero

$$E_p = \frac{\phi_{p+1} - \phi_{p-1}}{\Delta x_{p-1} + \Delta x_p} \neq 0 \quad \text{for} \ \ \Delta x_{p-1} \neq \Delta x_p \ .$$

This artificial electric field affects the particle as a self-force, although the calculated potential is correct. The antisymmetry in the potential and the resulting error in the electric field calculation can be seen in Figure 4.

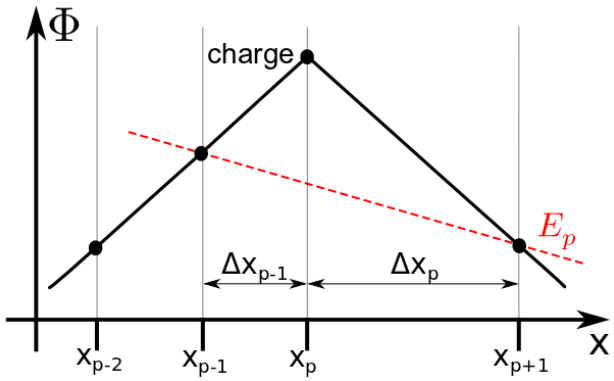

Figure 4. Electric field calculation for a charged particle located at x_p in a non-equidistant grid with a change of cell size $\Delta x_{p-1} \neq \Delta x_p$.

III. Poisson Solver

A. One-Grid Approach

While implementing PIC with a non-equidistant grid every part of the algorithm has to be adapted. In the Poisson solver the finite volume method is used to discretize the Laplacian. Methods to solve the resulting matrix equation and the discretization itself is discussed in the following.

1. LU Decomposition

One possibility is that, the complete grid is discretized and one matrix is created as a result of the finite difference or the finite volume method. Both leads automatically to large $N_G \times N_G$ matrices, with N_G as the number of grid points. Different sizes of grids are used to test the computing time of the algorithm to calculate the vacuum solution without space charge. For each grid size the matrix decomposed using the LU decomposition and the backsolve is done. Only the backsolve is repeated every timestep, so its calculation time is important. A non-equidistant finite volume solver set-up and matrix inversion has to be done only once, because the Laplacian operator depends only on metric coefficients. For the backsolve a practical limit is to stay below 1s.

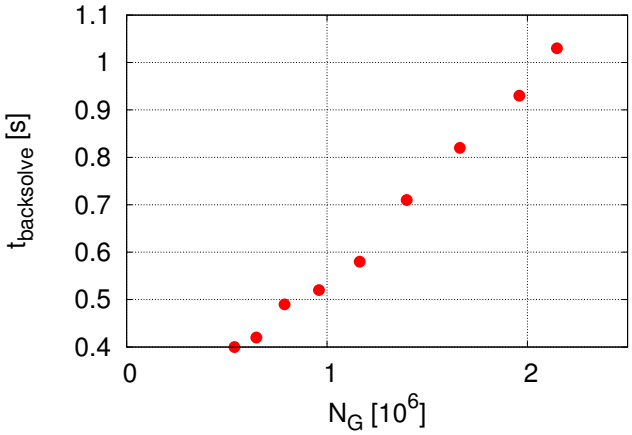

Figure 5. Backsolve time as a function of number of grid points N_G.

To study the physics of an ion thruster it is necessary to enlarge the simulation domain of the runs[2]. To find out how large the matrix and therefore the simulation domain can be before one reaches the runtime limit, the size of the domain is increased and the calculation time of one backsolve is measured. Times for initialization of the grid and the creation of the matrix scale quadratic with the number of grid cells. The result for the backsolve time is shown in Figure 5. The backsolve time scales linearly with grid size. Within a backsolve time below one second, it seems to be uncritical for this two-dimensional code to reach grid sizes near 2×10^6 cells.

2. Successive Over-Relaxation

In the following alternatives for the LU decomposition to solve the system of equations will be shown, since for large 3D systems a LU decomposition including the backsolve will be too costly in terms of numerics[5]. Especially their potential for parallelization is much more promising than the Super-LU algorithm allowing only to overcome the grid size limit. One possibility is the Successive Over-Relaxation (SOR). According to Nicholls[4] the principle of SOR is given by

$$\phi^{n+1} = \omega\phi^* + (1-\omega)\phi^n = \phi^n + \omega(\phi^* - \phi^n) \quad . \tag{1}$$

The super-scripted variables define the timestep the potential value is taken from. ω is the relaxation parameter. In general $\omega > 1$ (over-relaxation) speeds-up the process of converging, whereas $\omega < 1$ (under-relaxation) is used to stabilize the algorithm. ϕ^* is the result of a certain numerical discretization method. Two of them will be presented and tested in the following.

3. Jacobi Method

The Jacobi method is an explicit iterative scheme. In a two-dimensional grid it is defined by

$$\phi^*_{i,j} = \frac{1}{a_{i,j}} \left(a_{i-1,j}\phi_{i-1,j} + a_{i+1,j}\phi_{i+1,j} + a_{i,j-1}\phi_{i,j-1} + a_{i,j+1}\phi_{i,j+1} \right) \quad . \tag{2}$$

So $\phi^*_{i,j}$ is calculated only by values from the timestep before. Due to that this method is perfectly suited for parallelization, every line of the discretization matrix can be distributed to a single processor. The disadvantage is, that many iterations are needed until a solution has a variation of $\Delta\phi \leq 10^{-5}$V. This limit is used because it is many orders of magnitude smaller than the potential values of about some Volts in the plume. The iterative behavior is shown in the left part of Figure 6 and is tested for different ω.

One can see, that as expected a larger ω speeds up the iterative process. Values $\omega > 1$ are not possible, since they lead to a divergent solution. This test was done with a cylindrical domain of 10500 cells without any particles. The scaling with different domain sizes are shown in the lower right part of Figure 6. Independent of ω the number of iterations to reach the convergence limit scales linearly with the number of grid points, whereas the time scales with a higher exponent than 1. For this method $\omega = 1$ is the best choice for this particular test case.

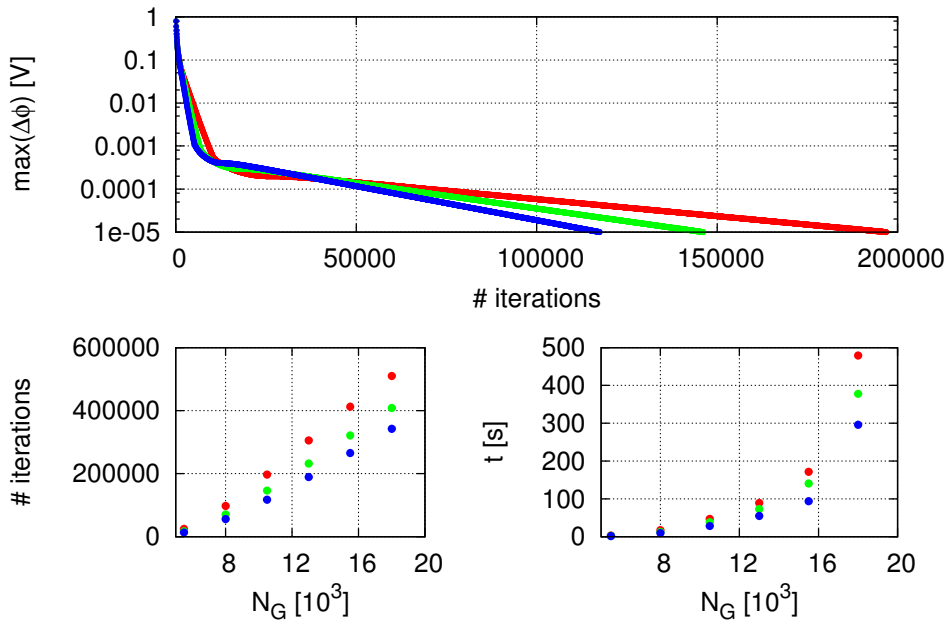

Figure 6. Convergence characteristics of the Jacobi method with different SOR parameter for $N_G = 10500$ (upper plot); iterations (bottom left) and time (bottom right) needed to reach a convergence of $\Delta\phi < 10^{-5}$V over number of grid cells N_G with the Jacobi method and different SOR parameter red $\omega = 0.5$; green $\omega = 0.75$ and blue $\omega = 1$.

4. Gauss-Seidel Method

In contrast to the Jacobi method the Gauss-Seidel method is implicit. The iterative character is kept. The difference to the Jacobi method is that one uses not only values from the previous timestep, but also values that are already calculated for the current timestep. The underlying equation of the Gauss-Seidel algorithm is given in (3).

$$\phi^*_{i,j} = \frac{1}{a_{i,j}} \left(a_{i-1,j}\phi^{n+1}_{i-1,j} + a_{i+1,j}\phi^{n}_{i+1,j} + a_{i,j-1}\phi^{n+1}_{i,j-1} + a_{i,j+1}\phi^{n}_{i,j+1} \right) \quad . \tag{3}$$

With the obtained ϕ^* one is again able to calculate ϕ^{n+1} using the SOR equation (1). One can make the same tests for the Gauss-Seidel method and gets Figure 7.

Even with the same parameter $\omega = 0.5$ one needs about 40000 iterations less to reach the convergence criteria $\max(\Delta\phi) < 10^{-5}$V. For $\omega = 1$ the difference between both methods is even larger. One gets a $\max(\Delta\phi) < 10^{-5}$V about 80000 iterations earlier with the Gauss-Seidel method. With the Gauss-Seidel method it is possible to use relaxation parameters $\omega > 1$ which speeds up the convergence. With $\omega = 1.75$ the solution is converged after less than 20000 iterations whereas with Jacobi's method one needs a minimum of 100000 iterations even at the largest ω.

Both methods scale linearly with number of iterations needed to reach a specific $\Delta\phi$. The quadratic behavior of the required time shown in the lower right plot in each figure is trivial to understand, because the total size of the matrix is proportional to the square of the number of grid cells N_G. Therefore, it is necessary to think about other strategies of minimizing the computational time for large cell numbers.

5. Strategies for Parallelization

Parallelization is one approach to speed up the calculation. Although the Gauss-Seidel method seems very promising in the numerical comparisons until now, it is very difficult to parallelize. Due to the implicit character it is not so trivial as for the Jacobi method.

One possible idea for PIC's Poisson solver will be to divide the computational domain into several sub-domains, calculate each sub-domain with Gauss-Seidel method on a single core and only do the matching at the interface between the sub-domains with Jacobi. To validate this idea, one makes tests with two domains, where different solver are introduced. Close to the channel one uses the accurate and fast Gauss-Seidel

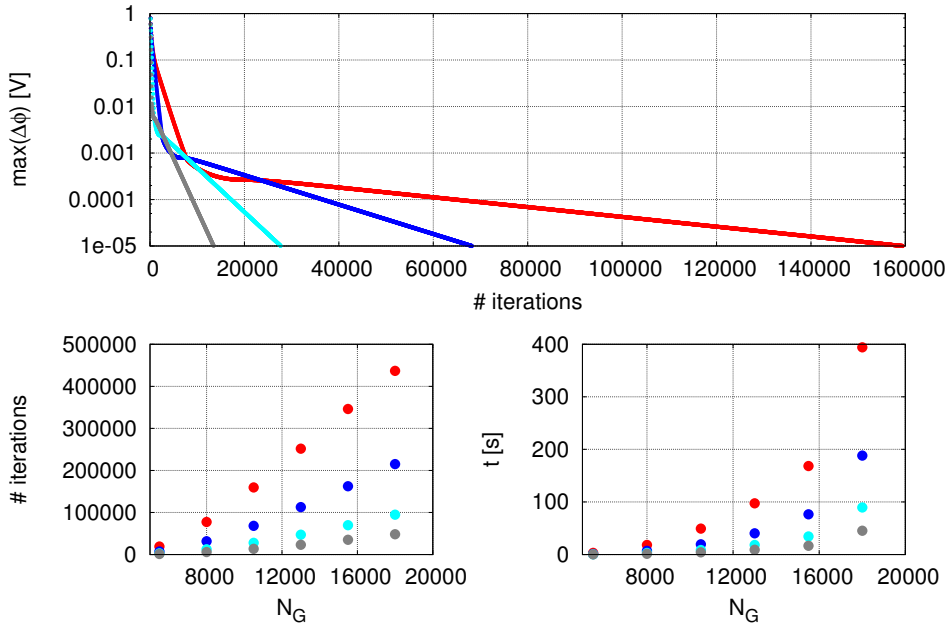

Figure 7. Convergence characteristics of the Gauss-Seidel method with different SOR parameter for $N_G = 10500$ (upper plot); iterations (bottom left) and time (bottom right) needed to reach a convergence of $\Delta\phi < 10^{-}$V as a function of number of grid cells N_G with the Gauss-Seidel method and different SOR parameter red $\omega = 0.5$; blue $\omega = 1$; cyan $\omega = 1.5$ and gray $\omega = 1.75$.

method, in the plume one implements the Jacobi method. The Gauss-Seidel method was modified with SOR using the highest possible SOR factor $\omega = 1.75$. Beginning from a run without the Jacobi method one introduces more and more cells where Jacobi's method is applied to up to a run where one uses the Jacobi method for every cell. Figure 8 shows how many iterations are needed to reach convergence as a function of the number of cells with the Jacobi method. For this test a grid with 33000 cells and one with 50500 cells are used.

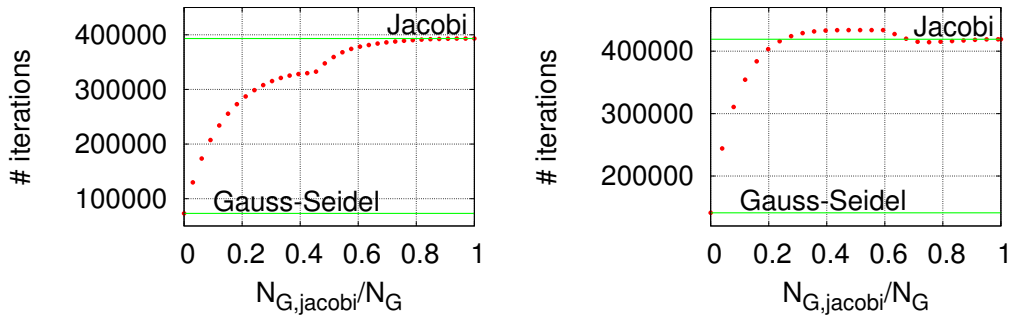

Figure 8. Number of iterations to reach numerical convergence as a function of percentage of Jacobi cells for $N_G = 33000$ (left) and $N_G = 50500$ (right).

Even with a small Jacobi region one needs much more iterations than with pure Gauss-Seidel method. Therefore, one has to carefully check, whether this option is good to use for parallelizing the PIC solver. The convergence seems to be strongly coupled to the slowest method which was used and therefore is orientated on the Jacobi method with its slow convergence. Even with a few cells where the Jacobi method is used the convergence is much slower than with a pure Gauss-Seidel domain. Interesting is the different behavior for the two grids. While with $N_G = 33000$ cells one has a continuous increase of needed iterations, for $N_G = 50500$ cells has a maximum between 40% and 60% and decreases then to the pure Jacobi solution. Both have in common, that even for a few cells using the Jacobi method, the convergence is slowed down.

6. *Reconverging after a small Perturbation*

More important than finding an initial solution is the reaction to a small perturbation. In other words one will check, how fast a solution reconverges in both methods if one takes a converged solution and adds a small oscillation of only a few percent. For these tests one distinguishes between numerical and physical convergence. Numerical convergence means that changes are only in the range of the numerical accuracy, whereas physical convergence means that the changes are so small, that the changes in the potential are so small that they have only negligible impacts on electric fields. Therefore, in the following numerical convergence is defined as $\Delta\phi < 10^{-5}$V and physical convergence as $\Delta\phi < 0.1\,$V. Important is to analyze the reaction to different amplitudes of the added signals. At first the goal should be to reach numerical convergence with both methods. The result is shown in Figure 9.

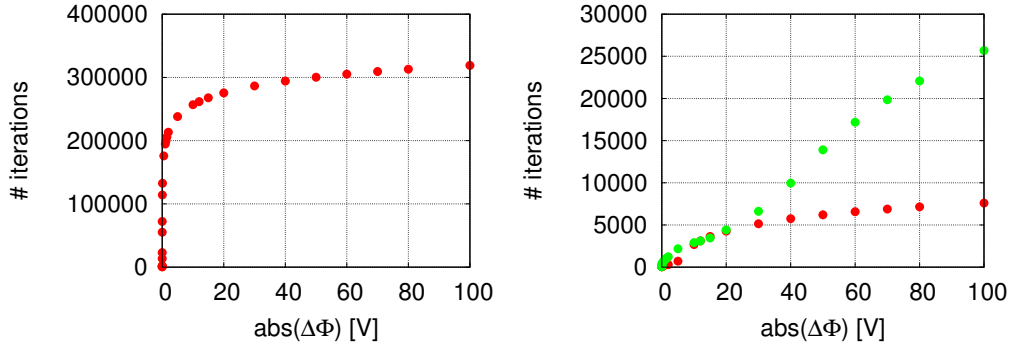

Figure 9. Needed number of iterations to reconverge after adding a small perturbation $\Delta\Phi$ to a already converged solution with the Jacobi method (left), the Gauss-Seidel method (right with green dots) and Gauss-Seidel with a SOR factor of $\omega = 1,75$ (right with red dots).

All these tests are done on a grid of 10500 cells. Obviously, the Gauss-Seidel method is faster than the Jacobi method. The SOR factor $\omega = 1.75$ gives an extra speed up. There is one more difference that is not evident from Figure 9 itself. These tests are done with an initial solution that is already converged. To reach this converged solution from scratch with Jacobi method one needs 117518 iterations. So using this solution and adding an oscillation makes it much harder for the algorithm to reconverge. For an amplitude of $\Delta\Phi = 50\,$V one needs about three times the number of iterations, so probably it would be easier to start from scratch in every step, which is not convenient for the PIC algorithm.

The behavior of the Gauss-Seidel method depends on the different SOR factors. With a SOR factor of 1 one needs 68209 iterations to converge from the scratch to the vacuum solution and with $\omega = 1.75$ still 13491. Compared with the number of iterations Figure 9 these are much lower, so with an already converged solution it is easier to reconverge after a perturbation. This can probably be explained by the implicit characteristics of the Gauss-Seidel method.

The result for physical convergence is shown in Figure 10.

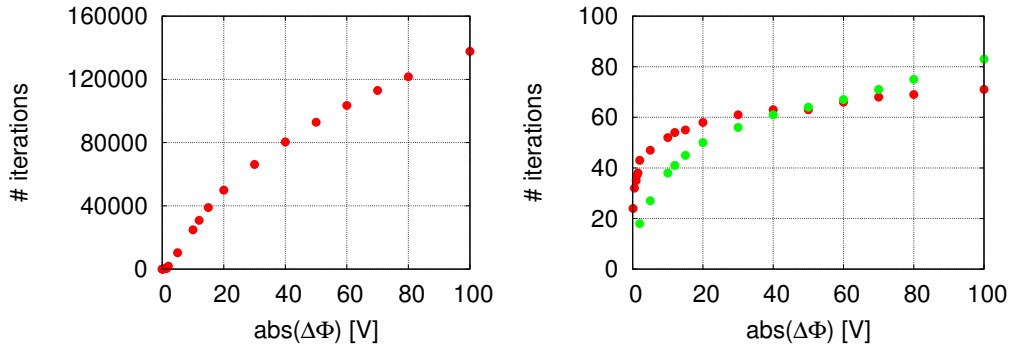

Figure 10. Needed number of iterations to reconverge in a physical sense after adding a small perturbation $\Delta\phi$ to a already converged solution with the Jacobi method (left), the Gauss-Seidel method (right with green dots) and Gauss-Seidel with a SOR factor of $\omega = 1.75$ (right with red dots).

Because with 10500 cells one converges even with the Jacobi method too fast to compare it, for the test

of physical convergence the grid was enlarged to 45500 cells. Even with this larger grid one needs much less iterations to converge. Both methods are similar at that point. Interesting is that with Gauss-Seidel there is a range $|(\Delta\phi)| \in [0; 40]$V where the Gauss-Seidel method without SOR is faster than with a SOR factor of 1.75.

At this point one should mention again, that the Gauss-Seidel method can not be parallelized trivially. Hence it is not well suited as an alternative for the one grid approach for 3D PIC, although of its numerous advantages in terms of numerical characteristics. To avoid this bottleneck one can make domain decomposition and calculate every domain with the Gauss-Seidel method on a single core and only do the matching between the cores with the Jacobi method, so a parallelization is possible, but not as trivial as with the Jacobi method. The question is what will be more promising for the application in PIC. In 3D simulations the backsolve of the Poisson solver with these large matrices can get easily beyond 1s [5], why one has to search for alternative algorithms. One possible approach is discussed in the following.

B. Hierarchical Multi-Grid Approach

Another method to treat non-equidistant meshes uses a hierarchical multi-grid approach to reduce the calculation time. With this ansatz it is not necessary to invert one huge $N_G \times N_G$ matrix. Therefore, it will be possible to save computation time and memory. In addition a decomposition of the domain and parallelization of the code is easier.

At first the potential on a coarse grid in the whole domain is calculated. The values from that solution are used as boundary conditions for the finer mesh, where only a smaller subdomain is solved. Afterwards the fine solution is interpolated back to the coarse grid. This procedure can be redone arbitrarily often. So a specific solution can be iterated until convergence. For a first test a grid of 50×50 cell size $\Delta r_2 = \Delta z_2 = 2\Delta r_1 = 2\Delta z_1$ was implemented over the whole domain as the coarse mesh. The channel $r < 5\,\text{mm}$ and $z < 4\,\text{mm}$ is the region of the finer grid with cells with size Δr_1 and Δz_1. At the left side an anode with $\phi = 400\,\text{V}$ is implemented. At $5\,\text{mm} < r < 10\,\text{mm}$ and $0\,\text{mm} < z < 4\,\text{mm}$ there is a dielectrics with $\epsilon_r = 4$. The other boundary conditions for r and z are set to $\phi = 0\,\text{V}$.

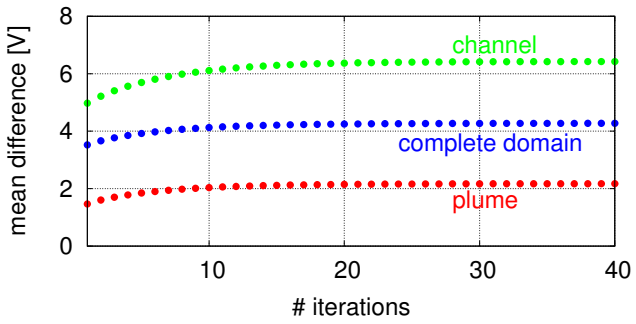

Figure 11. Mean difference $\Delta\phi$ after each iteration: plume (red), channel (green), whole domain (blue).

This method is iterative. Therefore, one has to study the development of the solution in dependence of the iterations. One compares the solution after each iteration with the solution of the fine mesh and calculates the difference $\Delta\phi$ averaged over the whole domain. The result is shown in Figure 11. After about 20 iterations there are no changes anymore. Since in the PIC simulation about 10^7 timesteps are needed to approach a steady-state solution for typical thruster simulations these iterations are automatically done.

The potential converges into a solution that is different to that one, which was calculated with one fine grid on the whole domain. The mean difference on the domain for the converged solution is $\Delta\phi = 4.3\,\text{V}$, which equals about 1% of the largest value at the anode. Additionally the mean difference over the whole domain is mostly dominated by the error directly in the corner, where the anode and grounded metal get together. Since there are no particle in this region due to the dielectrics, this estimate can be seen as a worst case scenario. Compared with physical electric fields, the difference is very small and can therefore be neglected.

The largest error of about 100 V appears in the channel at the upper left border. This is a region inside the dielectrics, where no plasma is existing. The potential is mostly driven there by the two boundary conditions for the anode and the grounded metal. Therefore, this error is not affecting the PIC solution,

because no particle is influenced by this artificial electric field. All other errors are much smaller by a factor of about 100. Overall, the errors are well below 0.1%.

If one considers the Poisson solvers as exact and not time-critical for large non-equidistant grids, the problem in the energy conservation for non-equidistant grids must lie somewhere else. It can be shown, that the problem is the calculation of the electric field using the central difference method. A modified electric field calculation to minimize the error in the pusher is discussed in[3].

IV. Conclusion

The restriction of the cell size in Particle-in-Cell codes due to the resolution of the Debye length strongly limits the computationally possible size of the simulation domain. This prohibits an complete self-consistent kinetic model of the thruster including its whole plume. One way to overcome this shortcoming is to use non-equidistant grids, although such methods have problems, which were shown in a single-particle test for energy conservation.

The Poisson solver as one possible error source was studied in detail. It was found out, that in 2D simulations the Poisson solver is not a problem and one has to look at other parts of PIC like the particle pusher to find and minimize the error. A modified electric field calculation to minimize the error in the pusher is discussed in[3].

Acknowledgments

This work was supported by the German Space Agency DLR through Project 50 RS 1101. This work of D. Tskhakaya was supported by EURATOM and carried out within the framework of the European Fusion Development Agreement. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1]R.W. Hockney and J.W. Eastwood, *Computer simulation using particles*, McGraw-Hill, New York, 1981.

[2] O. Kalentev, K.Matyash, R. Schneider, F. Taccogna, N. Koch and M. Schirra, *Kinetic simulation of the stationary HEMP thruster including the near-field plume region*, Proceedings of the 31st International Electric Propulsion Conference, pages 20-24, 2009.

[3]J.Duras, D. Tskhakaya, O.Kalentev, K. Matyash, K.F. Lüskow and R. Schneider, *Self-force in 1D electrostatic Particle-in-Cell codes for non-equidistant grids*, Contrib. Plasma Phys. (to be published).

[4]Anthony Nicholls and Barry Honig, A rapid finite difference algorithm, utilizing successive over-relaxation to solve he Poisson-Boltzmann equation, Journal of computational chemistry, 12(4):435-445,1991.

[5]J. Duras, *Instabilities in ion thrusters by plasma-wall interactions*, Diploma thesis, Brandenburgisch Technische Universität Cottbus, 2011.