

SOLUTION OF POISSON'S EQUATION IN ELECTROSTATIC PARTICLE-IN-CELL SIMULATIONS.

Kahnfeld D.[a,*], Schneider R.[a,b], Matyash K.[a,b], Kalentev O.[c], Kemnitz S.[b,d], Duras J.[e,a], Lüskow K.[a], Bandelow G.[a]

[a] *Ernst-Moritz-Arndt University Greifswald, Institute of Physics, Felix-Hausdorff-Str. 6, 17487 Greifswald, Germany*

[b] *Computing Center, Ernst-Moritz-Arndt University of Greifswald, D-17498 Greifswald, Germany*

[c] *Biomedizinische NMR Forschungs GmbH am Max-Planck-Institut für biophysikalische Chemie, D-37077 Göttingen, Germany*

[d] *University Rostock, Institute of Informatics, Albert-Einstein-Str. 22, 18059 Rostock, Germany*

[e] *Department of Applied Mathematics, Physics and Humanities, Nürnberger Institute of Technology, D-90489 Nürnberg, Germany*

[*] kahnfeldd@uni-greifswald.de

Abstract. In electrostatic Particle-in-Cell simulations of the HEMP-DM3a ion thruster the role of different solution strategies for Poisson's equation was investigated. The direct solution method of LU decomposition is compared to a stationary iterative method, the successive over-relaxation solver. Results and runtime of solvers were compared, and an outlook on further improvements and developments is presented.

Keywords: Particle-in-Cell, ion thrusters, Poisson's equation, LU decomposition, successive over-relaxation.

1. Introduction

For spacecrafts the concept of ion thrusters presents a very efficient method of propulsion. Ion thrusters generate a low thrust with much higher efficiency than chemical propulsion systems [1] and are commonly used on satellites in earth orbits.

Thrust is generated by accelerating ions of a plasma discharge and expelling them into space. The plasma within the thruster channel is dominated by electrostatic and magnetic fields, plasma-wall-interaction and non-linear effects. The shape and size of the plume have to be considered in the design of ion thrusters to account for possible damages caused by ion sputtering, but experimental access is difficult [1].

The HEMP-DM3a ion thruster design, as shown in fig. 1, possesses a rotational symmetry. The left boundary of the channel contains the anode with a voltage of 500 V, with the cathode supplied by an electron beam outside the channel which also serves as electron source and neutralizer for the expelled ions. The thruster channel is surrounded by permanent magnet rings of opposite magnetization. This results in a nearly constant magnetic field at the symmetry axis of the thruster with the exception of cusp regions, where two rings with opposite magnetization are located next to each other. The inner boundary of the thruster channel is made up of a dielectric ceramic consisting of Boron Nitrite which has a high threshold energy to reduce sputtering [2]. The thruster's exit is concluded with a grounded metal plate attached outside the dielectric. A more detailed description of the thruster can be found in [3].

Plasma simulations offer the means to understand the plasma physics within an ion thruster and can aid the design of new thruster concepts. A widely applied method is the Particle-in-Cell (PIC) scheme, simulating the trajectories of super-particles consisting of many real particles. Even with modern hardware, state-of-the-art features such as similarity scaling [4] and non-uniform grids [5] have to be used to make simulation of an ion thruster conceivable.

With the access to highly parallel computing clusters the best chance of gaining a speed-up of the simulation is an efficient parallelization. In order to achieve good scalability the communication overhead needs to be kept as small as possible, while load imbalance needs to be avoided by proper work distribution. One bottleneck for an efficient parallelization is the solution of Poisson's equation, which is often obtained by the use of traditional direct methods such as the Gauss algorithm. While very fast, such methods cannot be parallelized, and may lead to memory problems for large domain sizes. Therefore parallel solution strategies need to be investigated, one of which is the successive over-relaxation method.

2. Theory and code description

2.1. Basics of PIC

The Particle-in-Cell (PIC) method is a well-established scheme for simulation of plasmas. In PIC, so-called super-particles are moved within a simu-

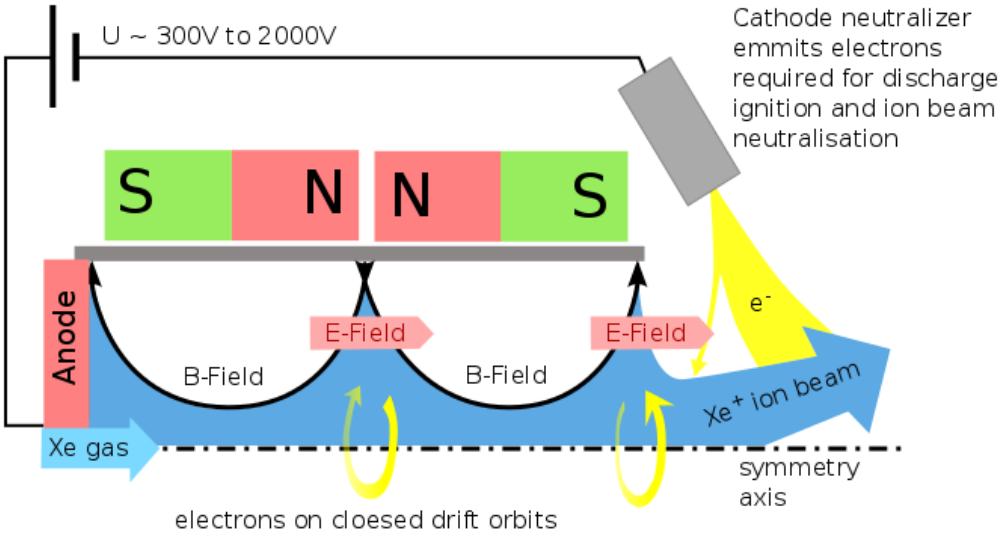

Figure 1. Schematic design of the HEMP-DM3a thruster.

lation domain, each representing a number of real particles. A grid is introduced, dividing the simulation region into cells, with macroscopic quantities such as the charge density n and the electrostatic potential ϕ being calculated only on the grid points. This enables treatment of large systems by calculating the electrostatic potential via Poisson's equation

$$\Delta\Phi(\vec{x},t) = -\frac{n(\vec{x},t)}{\varepsilon\varepsilon_0} \tag{1}$$

only on the grid instead of N^2 direct particle interactions. Collisional effects are only taken into account within each cell separately.

The PIC cycle starts at a given time t_0 by initializing the system and calculating the macroscopic quantities on the grid points using the particle positions and velocities. The forces acting on the particles are calculated on the grid and then reassigned to each particle, resulting in a change of the particle's position and velocity. After calculating further particle interactions, i.e. collisions and surface interactions, the system is advanced by a discrete timestep Δt and returns to the start of the cycle. To assure stability, the timestep has to be chosen small enough to resolve the fastest particle movement. A more detailed description of the PIC method can be found in [6].

2.2. Finite difference scheme and solvers

To calculate the electric field on the grid, Poisson's equation has to be solved. The solutions will be acquired by introducing a finite difference scheme for the spatial second order derivatives. For a two dimensional $M \times N$ grid (x_i, y_j) with constant permittivity ε and charge density $n = n_i - n_e$ eq. 1 takes the form

$$\Delta\Phi = \mathrm{A}\Phi = -\frac{n}{\varepsilon\varepsilon_0}\,, \tag{2}$$

creating a system of linear equations to be solved. The form of the matrix A depends on the discretization stencil that is used. The PIC code discussed here employs a five-point stencil leading to an accuracy of second order. Accuracy may be increased by incorporating more points into the difference scheme, i.e. by using a nine-point stencil, at the cost of increased computation time by additional matrix entries [7].

The resulting $(M\cdot N)\times(M\cdot N)-$dimensional matrix A has a characteristic block structure

$$\mathrm{A} = \begin{pmatrix} \mathrm{D} & \mathrm{B} & 0 & \cdots & 0 \\ \mathrm{B} & \mathrm{D} & \mathrm{B} & \ddots & \vdots \\ 0 & \mathrm{B} & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \mathrm{B} \\ 0 & \cdots & 0 & \mathrm{B} & \mathrm{D} \end{pmatrix}.$$

The matrices D and B then have a dimension of $M \times M$. In the model case of a five-point stencil on a cartesian grid, D is tridiagonal with values of -4 as diagonal entries and values of 1 elsewhere. In that case, B is the unity matrix. In real applications the matrix structure is more complicated, as boundary conditions, non-constant permittivity ε or choice of geometry change the matrix structure. The case of radial coordinates, which is used in the PIC code discussed, is described more closely in [8]. Despite a more complex structure, basic matrix properties such as symmetry are preserved.

2.2.1. LU Decomposition

An often used method to solve systems of linear equations is the LU decomposition, also known as Gauss algorithm. Eq. 2 can be rewritten in the form

$$\mathrm{A}\Phi = b\,. \tag{3}$$

The system can be solved by representing the matrix A as the product of an upper triangular matrix U and a lower triangular matrix L, transforming the

equation to

$$\mathrm{A}\Phi = \mathrm{LU}\Phi = \mathrm{L}\tilde{\Phi} = b\,.$$

If L and U are known, the solution is easy. $\tilde{\Phi}$ is obtained by simply substituting the result of the previous lines into the next one as L has a triangular structure. The next step is to obtain Φ by solving for $\mathrm{U}\Phi = \tilde{\Phi}$ analogously. This step is also known as back-solve. For each back-solve, the complexity is $\sim (M \cdot N)^2 / 2$ [9], making it very efficient.

The problem lies in the computation of the decomposition $\mathrm{A} = \mathrm{LU}$ which shall not be discussed in detail here, but a good review can be found in [9]. It can be shown that a LU decomposition exists for every regular matrix, but pivoting, interchanging rows and columns of the matrix in order to move the matrix elements with the highest absolute value to the diagonal, might be necessary, thus further increasing computation time. The complexity of the decomposition is $\sim (M \cdot N)^3 / 3$.

The Gauss algorithm is a very robust direct method to solve matrix equations. With the exception of rounding errors, which can be minimized by partial or full pivoting, it reliably delivers the right solution. In PIC the LU decomposition offers a reliable and efficient solver for the field solving step, as the matrix structure is well investigated. The decomposition is calculated at the beginning of code execution, as the matrix does not change throughout the execution of the code, and only the back-solve has to be computed every PIC cycle, hence giving a complexity of $\sim (M \cdot N)^2$ per PIC cycle.

However, a parallelization is problematic, as each line within a back-solve step depends on the results of the previous lines, limiting its application to a computational core. Parallel methods are only available for the calculation of the LU decomposition but not for the back-solve [10]. Therefore, in each PIC step it is necessary to reduce the charge densities onto a single core and then distribute the calculated electrostatic potential if the LU decomposition is used. The communication overhead created by this approach cannot be neglected on highly parallel systems.

2.2.2. Successive over-relaxation

On parallel systems, a frequently used method to solve eq. 2 is the use of a stationary iterative procedure. To formally obtain such procedures, eq. 3 is rearranged using a regular matrix B. The $(k+1)-$th iterate is then calculated as

$$\begin{aligned}
\mathrm{A}\Phi &= \mathrm{B}\Phi + (\mathrm{A} - \mathrm{B})\,\Phi = b \\
\mathrm{B}\Phi^{k+1} &+ (\mathrm{A} - \mathrm{B})\,\Phi^k = b \\
\Phi^{k+1} &= \Phi^k - \mathrm{B}^{-1}\left(\mathrm{A}\Phi^k - b\right) = F\left(\Phi^k\right)\,,
\end{aligned}$$

The iterative procedure can be broken down to four steps:

i) Choose a starting point Φ^0 .

ii) Calculate $\mathrm{A}\Phi^k$.

iii) Solve $\mathrm{B}\Delta\Phi^k = b - \mathrm{A}\Phi^k$.

iv) $\Phi^{k+1} = \Phi^k + \Delta\Phi^k$.

B is chosen to have a simple form in order to reduce the necessary number of operations and defines the iterative procedure. Also B is often linked to the matrix A. If it is chosen to be the diagonal of A, the algorithm is known as Jacobi algorithm. If B is chosen to be the sum of the A's diagonal matrix D (with $a_{ii} \neq 0$ for all i) and its lower triangular matrix L (not to be confused with the matrix used in the LU decomposition), the Gauss-Seidel algorithm, with the element index i, is acquired:

$$\begin{aligned}
\mathrm{A} &= \mathrm{D} + \mathrm{L} + \mathrm{R} \\
\mathrm{B} &= \mathrm{D} + \mathrm{L} \\
\Phi^{k+1} &= -\left(\mathrm{D} + \mathrm{L}\right)^{-1}\left(\mathrm{R}\Phi^k - b\right) \\
\Phi_i^{k+1} &= \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}\Phi_j^{k+1} - \sum_{j>i} a_{ij}\Phi_j^k\right)\,. \qquad (4)
\end{aligned}$$

This method is convergent if A is symmetric and positive definite [7]. It can be enhanced by introducing a relaxation parameter ω into the choice of B

$$\mathrm{B}(\omega) = \frac{1}{\omega}\left(\mathrm{D} + \omega\mathrm{L}\right)\,.$$

The algorithm is altered, giving

$$\begin{aligned}
\Phi^{k+1} &= \Phi^k + \omega\left(\tilde{\Phi}^{k+1} - \Phi^k\right) \\
\tilde{\Phi}_i^{k+1} - \Phi_i &= \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}\Phi_j^{k+1}\right. \\
&\qquad \left. \sum_{j>i} a_{ij}\Phi_j^k - a_{ii}\Phi_i^k\right)
\end{aligned}$$

where $\tilde{\Phi}_i^{k+1}$ is calculated via eq. 4. If $\omega < 1$ this is called under-relaxation and can be used to dampen divergent solutions. For $\omega > 1$ the algorithm is known as successive over-relaxation (SOR) which is an often applied method to solve the finite difference scheme for Poisson's equation.

The iteration continues until a termination criterion is met. A possible choice is

$$\frac{\|\Phi^{k+1} - \Phi^k\|}{\|\Phi^{k+1}\|} < \delta$$

in a given vector norm $\|\cdot\|$. Because this criterion is critical for $\Phi^{k+1} \to 0$ the condition

$$\|\Phi^{k+1} - \Phi^k\|_{max} < \varepsilon$$

may be used as well. The maximum norm is chosen to minimize the necessary computational cost.

For the solution to converge, as the Gauss-Seidel algorithm depends on the newly calculated iterates,

the domain should be divided into small subdomains, each solved separately. A chess board pattern, solving first all even and then all uneven grid points, or vice versa, may also be used.

It can be shown [7] that the SOR method is only convergent for $\omega \in (0, 2)$ and that the optimal relaxation parameter can be found in the interval $\omega_{opt} \in (1, 2)$. ω_{opt} can only be analytically calculated for a uniform grid spaced by Δ, as found in [7], but a decent guess is provided by the approximation

$$\omega_{opt} \approx 2 - \Delta\,.$$

The parameter is found in only a narrow range and has a large influence on the convergence rate, thus it needs to be tuned to the grid used. This can be achieved using simple optimization methods such as the hill-climb algorithm.

The complexity of each iteration step is $\sim (M \cdot N)^2$ and the expected number of iteration steps is $\sim (M \cdot N)$, giving the entire SOR method a complexity of $\sim (M \cdot N)^3$ [7]. The complexity is much higher compared to the back-solve of the LU decomposition which scaled quadratically.

The algorithm's structure allows for easy parallelization as the calculation of each point's iterate depends on only the surrounding points, delivering an advantage over LU decomposition. Only the boundary points have to be exchanged during each iteration step. For small subdomains, the communication overhead is kept relatively small.

2.3. Code description

The first simulations of the HEMP-T were performed by K. Matyash et. al. [11] and more recent results can be found in [12]. A 2d3v PIC scheme with radially symmetric 2D domain and a grid spacing of $\Delta z = \Delta r = 0.5\lambda_{D,e}$ on a domain of 1272×480 grid points was used. The particle velocities are treated in 3D. The timestep was chosen to be $\Delta t = 0.2/\omega_{P,e} = 1.2 \cdot 10^{-12}\,\mathrm{s}$ with about 10^6 timesteps necessary to reach a steady state. The simulated plasma consists of neutral Xenon gas, single positively charged Xenon ions and electrons. Particle collisions are simulated using a Monte-Carlo collisions scheme. The collisions include elastic Coulomb, excitation, ionization and elastic neutral-neutral collisions.

To reduce computational costs, similarity scaling as described in [4] is used, reducing the system size but keeping the physical laws intact as the mass-to-charge ration of each species is unchanged. A non-uniform mesh, further discussed in [5], is applied to the simulation region. The ions are moved once per $400\Delta t$ and neutrals are moved once per $2000\,\Delta t$.

A multigrid method incorporating two nested grids, as described in [12], is used for the calculation of the electrostatic potential Φ. A coarse grid covers the entire domain, with a larger grid spacing of $\Delta z_{coarse} = 4\Delta z_{fine}$, while the finer grid only covers the thruster region with a mesh of 888×236 grid points. During the field solve phase, a solution for Φ is first obtained on the coarse grid, with the boundary conditions of the finer grid given by the interpolated values on the coarse grid. Then a solution is obtained for the finer grid. The anode voltage is set to 500 V with a zero potential boundary condition at the upper and a no flux condition at the right boundary. For simplicity, only the solution of Poisson's equation on the fine grids will be discussed, as the behavior on the coarse grid is very similar.

The existing method for solving Poisson's equation is the Gauss algorithm included in the SuperLU library [10], calculating the LU decomposition once, only using the back-solve during each PIC timestep. Within the PIC code the SOR method was implemented as an alternative option to the SuperLU algorithm. The iteration procedure is executed until the termination condition $\|\Phi^{k+1} - \Phi^k\|_{max} < \varepsilon$ is met for two subsequent iterates Φ^{k+1} and Φ^k in dimensionless form. As the domain covers large areas with $\Phi = 0\,\mathrm{V}$, a relative termination condition is not well-suited here. The SOR method requires an initial guess at the start of the iteration, therefore SuperLU is executed once at the start-up of the code, and the solution will be stored as the initial guess of Φ during the first iteration. Alternatively, the SOR algorithm can also be used to obtain the initial guess, but this usually costs more computational time than the SuperLU method, when no parallelization is used. The solution of each following iteration is then stored and used as guess during the next field solve.

For testing of the solvers, a restarted run of the code, with charged particles covering the thruster channel and the exhaust region, is used. This simulated HEMP-T is in a steady state after 14 516 000 timesteps were computed. For the SOR method, the initial guess is a constant potential solution on the domain to ensure comparability. The PIC code and solvers discussed in this work are sequential, but the focus of this work lies on the investigation of an easy to parallelize Poisson solver.

3. Results

In fig. 2 the potential solution is plotted. It can be anticipated that the zero potential boundary conditions on the upper and the no-flux condition on the right boundary differ from the real situation, thus deviating the simulated potential and distorting simulation results. Therefore, a large simulation region for the plume is desirable, but increases computational cost, which can be made up for by introducing an efficient parallelization of the code.

For the SOR method, an ideal relaxation parameter of $\omega_{opt} = 1.981$ was obtained experimentally. A termination condition of $\varepsilon < 10^{-8}$ in dimensionless units was used, which corresponds to a change in potential of roughly $10^{-2}\,\mathrm{V}$, such that the influence of the potential difference on the system can be neglected. The absolute differences in the domain between the

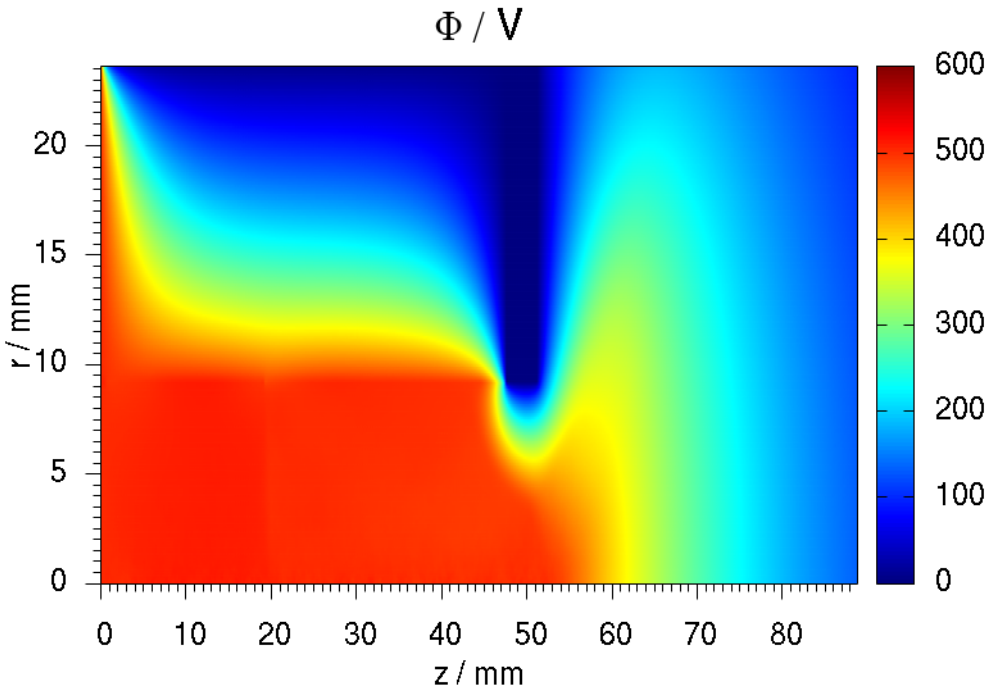

Figure 2. Solution of electrostatic potential in HEMP-DM3a obtained by the use of SuperLU package.

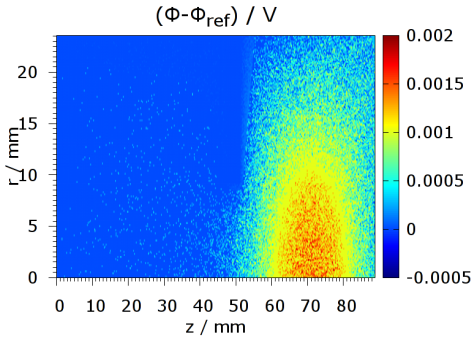

Figure 3. Comparison of solutions of SOR and SuperLU solvers with a termination condition of $\varepsilon < 10^{-8}$.

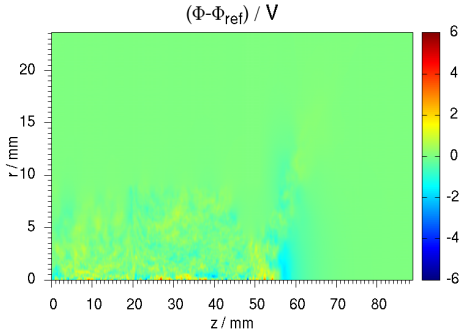

Figure 4. Comparison of solutions of SOR and SuperLU solvers with a termination condition of $\varepsilon < 10^{-8}$ after 9100 PIC steps, averaged over 100 steps

solution of the SOR and SuperLU methods is shown in fig. 3. The deviations are largest in the area of the thruster exit, where the potential gradient is largest. Still the differences are only of the order $\lesssim 10^{-2}$ V. In order to judge the applicability of the SOR method to PIC codes, one also needs to check the potential solution for a larger number of PIC cycles, to ensure that rounding errors will not be adding up in certain regions. Such a long-term comparison can be seen in fig. 4, where the absolute difference in potential after 9100 timesteps, averaged over 100 timesteps is presented. The differences are of the order of several Volts. The plot shows that deviations vary stochastically within the thruster channel and no systematic errors are adding up using the SOR method.

The runtime of the solvers differs drastically. Using the SuperLU back-solve, the execution time of one PIC cycle was just under one second, while one the same machine the time using the SOR solver was measured to be about 23 s per PIC cycle. For the long-term test presented in fig. 4, the overall execution time increased by a factor of 40. This shows the difference in scaling between the back-solve and the SOR method as described above.

4. Conclusions

The SOR method offers an alternative to traditional direct solution methods, i.e. LU decomposition, of Poisson's equation which occurs in finite difference discretizations within electrostatic PIC codes. For

sequential code structures, this method is not recommended as its scaling is one order of magnitude worse than that of LU decomposition. On massively parallel systems however, the situation is different, as the LU back-solve cannot be parallelized, hence creating communication overhead and load imbalance and therefore limiting scalability of parallelizations. One choice of parallel solver would be the SOR method, with a trivial generalization to a multicore environment.

One problem that arises in parallelization of the SOR method is the exchange of domain boundaries within each iteration. For a high number of iterations this creates considerable communication overhead. A possible solution to this problem can be found by increasing computational cost of each iteration, with a reduction of total number of iterations. Multigrid methods [7] make use of the error smoothing property of stationary iterations, such as the Gauss-Seidel iteration, and usually converge within the order of ten iterations, making further investigations of such solvers within parallel PIC codes very attractive.

Acknowledgements

This work was supported by the German Space Agency DLR.

References

[1] Dan A. Goebel and Ira Katz. *Fundamentals of Electric Propulsion: Ion and Hall Thrusters* . JPL SPACE SCIENCE AND TECHNOLOGY SERIES. NASA, 2008.

[2] Norbert Koch, Martin Schirra, Stefan Weis, Alexey Lazurenko, Benjamin van Reijen, Jens Haderspeck, Angelo Genovese, Peter Holtmann, Ralf Schneider, Konstantin Matyash, and Oleksandr Kalentev. The HEMPT Concept - A Survey on Theoretical Considerations and Experimental Evidences . In *Proceedings of the 32nd International Electric Propulsion Conference*, number IEPC-2011-236, September 2011.

[3] Erfahrungsaustausch Oberflächentechnologie mit Plasma- und Ionenstrahlprozessen. *The HEMP Thruster - An Alternative to Conventional Ion Sources?*, volume X., march 2003.

[4] F. Taccogna, S. Longo, M. Capitelli, and R. Schneider. Self-similarity in Hall plasma discharges: Applications to particle models. *Phys. Plasmas*, 12:053502, 2005.

[5] Karl Felix Lüskow, Julia Duras, Oleksander Kalentev, Konstantin Matyash, David Tskhakaya, Jürgen Geiser, and Ralf Schneider. Non-equidistant Particle-In-Cell for Ion Thruster Plumes. In *Proceedings of the 33rd International Electric Propulsion Conference*, October 2013.

[6] D. Tskhakaya, K. Matyash, R. Schneider, and F. Taccogna. The Particle-In-Cell Method. *Contrib. Plasma Phys.*, 47(8-9):563–594, 2007.

[7] J. Stoer and R. Bulirsch. *Numerische Mathematik 2.* Springer Verlag, 5th edition, 2005.

[8] Konstantin Matyash, Ralf Schneider, Andreas Mutzke, Oleksandr Kalentev, Francesco Taccogna, Norbert Koch, and Martin Schirra. Kinetic simulations of SPT and HEMP thrusters including the near-field plume region, December 2009. First presented at the ICNSP'09.

[9] J. Stoer. *Numerische Mathematik 1.* Springer Verlag, 9th edition, 2005.

[10] Xiaoye S. Li. An overview of SuperLU: Algorithms, implementation, and user in- terface . *ACM Trans. Mathematical Software*, 31:302–325, September 2005.

[11] Konstantin Matyash, Ralf Schneider, Andreas Mutzke, Oleksandr Kalentev, Francesco Taccogna, Norbert Koch, and Martin Schirra. Kinetic Simulations of SPT and HEMP Thrusters Including the Near-Field Plume Region. *IEEE Transactions on Plasma Since*, 38(9, Part 1):2274–2280, SEP 2010.

[12] Karl Felix Lüskow. Physics of Ion Thrusters' Plumes . Master's thesis, University of Greifswald, July 2013.