A Fast 3D Multigrid Based Space–Charge Routine in the GPT Code

G. Pöplau*, U. van Rienen, Rostock University, Germany
S. B. van der Geer, Pulsar Physics, Soest, The Netherlands
M.J. de Loos, Eindhoven University of Technology, The Netherlands

Abstract

Fast calculation of 3D non–linear space–charge fields is essential for the simulation of high–brightness charged particle beams. We report on our development of a new 3D space–charge routine in the General Particle Tracer (GPT) code. It scales linearly with the number of particles in terms of CPU time, allowing over a million particles to be tracked on a normal PC. The model is based on a non–equidistant multigrid Poisson solver that is used to solve the electrostatic fields in the rest frame of the bunch. Bunch lengthening and emittance growth calculations in a low–energy short electron bunch are chosen as an example of non–linear space–charge effects in a high–brightness photo–injector.

1 INTRODUCTION

Numerical prediction of charged particle dynamics in accelerators is essential for the design and understanding of these machines. Applications such as colliders and SASE–FEL's demand very high quality electron bunches, where any anomaly severely degrades the final performance.

A powerful tool widely used for the study of the behaviour of charged beams is the General Particle Tracer (GPT) [2]. It calculates the trajectories of a large number of sample–particles through the combined external and self–induced fields generated by the charged particles (the so–called space–charge forces). Depending on charge density and energy, a direct point–to–point model can not be used to calculate space–charge forces because of granularity problems and the inherent $\mathcal{O}(N^2)$ scaling between the number of sample particles and CPU time [1].

A method to stabilize the calculations and to rigorously save CPU time, for example as implemented in the SCHEFF routine in PARMELA [4], is to restrict all calculations to 2D, calculate the fields on the edges of a mesh and use interpolation to smooth the fields.

In this paper we introduce a 3D model for the fast calculation of space–charge following the ideas in [7]. The space–charge fields are computed in the rest frame by a non–equidistant multigrid scheme. Hence, the numerical effort scales linearly with the number of particles in terms of CPU time. The new model is well suited for a variety of applications, including the calculation of space–charge fields in a high–brightness photo–injector, see Figure 1. Important questions for the efficiency of the algorithm are the construction of the grid adapted to the particle distribution and the choice of a reliable multigrid scheme.

* supported by a research grant from DESY, Hamburg

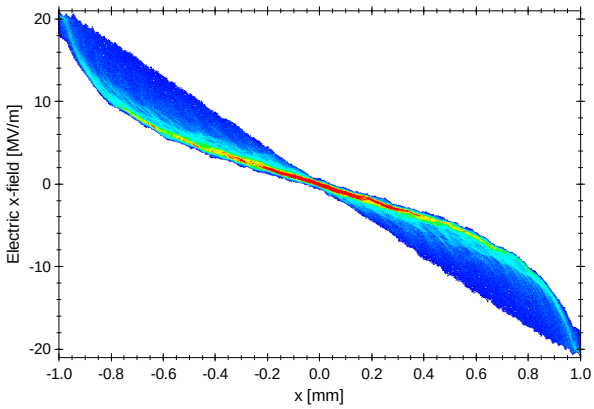

Figure 1: Color–density plot of the projection of the non–linear transverse electric field of a 1 nC hard–edged bunch with a radius of 1 mm and a length of 0.1 mm, as calculated by the new 3D space–charge routine in GPT. The calculation is based on one million particles on a $129 \times 129 \times 129$ mesh.

2 THE 3D SPACE–CHARGE MODEL

The space–charge fields have to be computed in each time step of the numerical integration of the relativistic equation of motion (in GPT a 5th order embedded Runge–Kutta scheme with adaptive step size control is implemented). The space–charge calculation is performed as follows:

1. Transformation of the particles from the laboratory frame to the rest frame by Lorentz transformation.
2. Determination of a non–equidistant 3D Cartesian grid in correspondence to the charge density of the bunch (see subsection 2.1).
3. Approximation of the charge at the grid points.
4. Calculation of the electrostatic potential at the grid points via Poisson's equation applying a multigrid algorithm. The finite difference scheme is used for the discretization of Poisson's equation (see subsection 2.2).
5. Derivation of the electric field in the rest frame and trilinear interpolation of the field values to the particle positions.

6. Transformation of the field to the laboratory frame by Lorentz transformation.

The efficiency and accuracy of the space–charge calculation mainly depends on the determination of the 3D mesh and the applied multigrid scheme to solve Poisson's equation. Both we describe in the next two subsections.

2.1 Mesh Generation

The electromagnetic potential is calculated on a 3D Cartesian mesh where an approximation of the charge in the rest frame is stored at the grid points. The 3D mesh is generated in a cube surrounding the bunch. To reduce the number of mesh lines needed, and thus to reduce CPU time, the density of the mesh lines is increased if the charge–density increases.

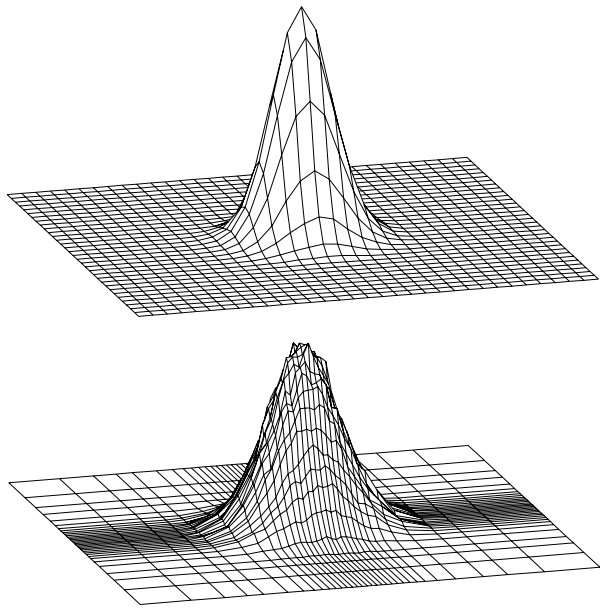

Figure 2: Mesh line positions ((x, y)–plane) for a Gaussian charge density with $fn = 0$ (top) and $fn = 0.5$ (bottom). The vertical axis shows the total charge in each mesh box, where the height of the top has been normalized in both plots.

The actual positioning of the mesh lines is an iterative process. The mesh lines are distributed such that they are spaced according to the distribution of the beam charge density. The parameter fn is introduced to maintain a maximum difference in spacing between neighboring mesh lines, to avoid the creation of a non–optimal mesh line distribution for the Poisson solver. If, e. g. $fn = 0.25$, then the difference in spacing between neighboring mesh lines can not vary by more than 25%. To span the mesh over the bounding box additional mesh lines are added left and right of the bunch. The spacing of these additional mesh lines is increased, restricted to fn, to add as few lines as possible. The addition of mesh lines and the effect of fn is shown in Figure 2. When $fn = 0$, the spacing between all neighboring mesh lines is allowed to vary by 0%, creating an equidistant mesh. Such a mesh is most stable for the multigrid Poisson solver, but it will create many empty mesh boxes. On the other extreme, setting $fn = 0.5$ results in a dense sampling of the electron bunch and sparse sampling of the surrounding area.

2.2 The Multigrid Poisson Solver

After creating the mesh and approximating the charge of the particles on the mesh points, the space–charge forces can be calculated by means of Poisson's equation. First, Poisson's equation is discretized by finite differences using the non–equidistant mesh described in the previous section. The solution of the resulting system of equations (with up to 1 million degrees of freedom) requires a fast and robust solver.

State-of-the-art is the application of a multigrid method as Poisson solver [3]. In model cases the numerical effort scales with the number of mesh points. The multigrid algorithm operates on a certain number of grids starting with the mesh given by the discretization of Poisson's equation. Then a sequence of coarser grids is generated by cutting mesh lines. On an equidistant mesh every second mesh line is removed. Now iteratively, a raw approximation of the solution of the systems of equations is obtained by the application of a few steps of a relaxation scheme (e. g. Gauss–Seidel). This approximation is then improved by a correction vector obtained on the coarser grids (the so–called coarse grid correction).

The coarsening strategy is crucial for the convergence of the multigrid algorithm on non–equidistant grids [5, 6]. Here, the removal of mesh lines follows the rule: Two neighboring steps h_1 and h_2 remain also in the next coarser grid as long as either $h_1 \geq sh_{min}$ or $h_2 \geq sh_{min}$, where h_{min} denotes the overall minimal step size of the corresponding fine level. The factor s is chosen as $s = 1.6$ or $s = 1.7$ with the objective to obtain a decreasing aspect ratio of the mesh spacing.

3 TRACKING EXAMPLE

During a drift, an electron bunch expands both longitudinally and radially due to the space–charge forces. Figure 3 shows a simulation of a hard–edged bunch, starting with an energy corresponding to a Lorentz factor of $\gamma = 5$. The bunch has a total charge of 1 nC, a radius of 1 mm and a length of 0.1 mm ('pancake' bunch). After 100 ps, the hard edges have become smooth, and the density at the head of the bunch is larger than at the tail due to relativistic effects.

The expansion calculated with the new 3D space–charge routine has been compared to the expansion simulated with the well–tested cylindrically symmetric 2D space–charge routine of GPT [2]. The result for the bunch length is shown in Figure 4. It demonstrates the perfect agreement between the 2D and the 3D routine, even at 1000 particles.

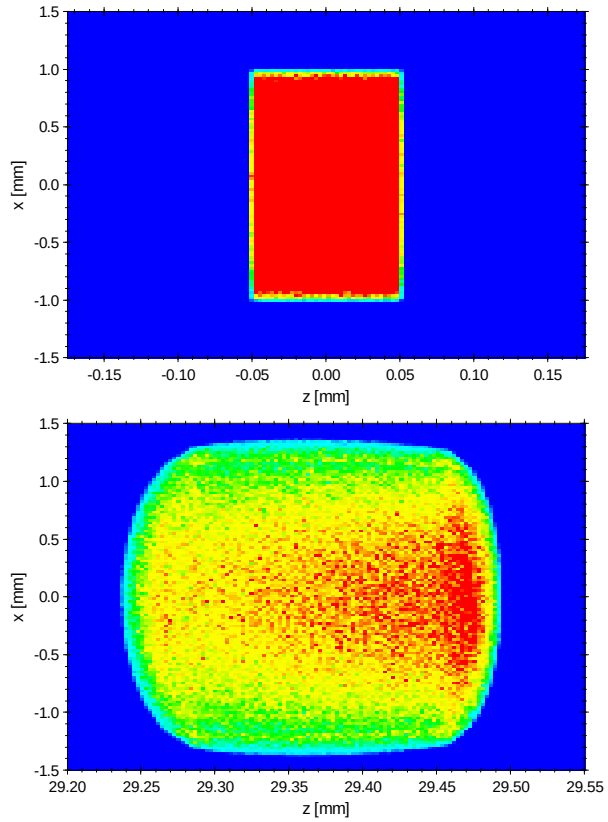

Figure 3: Initial (top) and final (bottom) projections of the charge density of an expanding 'pancake' bunch in the (x, z)–plane. One million particles are used on a $65 \times 65 \times 65$ mesh.

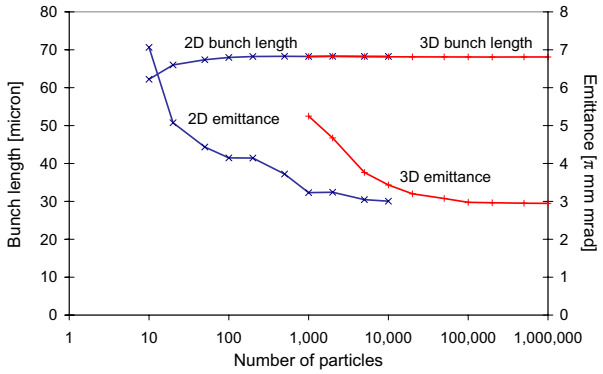

Figure 4: Bunch length (expressed as standard deviation) and final emittance after tracking a 'pancake' bunch with total charge of 1 nC during 100 ps. Tracking a million particles with the 3D model takes only 30 minutes CPU time on a 1.6 GHz Pentium PC.

Although the bunch length converges quite rapidly, the emittance is harder to stabilize. As shown in Figure 4, well over 10,000 particles are needed before the 3D routine converges. The final value is identical to the 2D case, where about a 1000 'rings' are sufficient. The very smooth convergence curve indicates that there is potential for improvement in the algorithm. It should be noted that this example is a quite extreme case where the emittance explodes from 0 to about 3 π mm mrad in just 30 mm.

4 CONCLUSIONS

A new 3D space–charge routine implemented in the GPT code has been described in this paper. The new method allowing 3D simulations with a large number of particles on a common PC is based on a multigrid Poisson solver for the calculation of the electrostatic potential in the rest frame. Numerical results of the 3D routine show perfect agreement with the standard 2D space–charge model of the GPT code.

REFERENCES

[1] S.B. van der Geer, M.J. de Loos, "The General Particle Tracer Code. Design, implementation and application", PhD thesis, Einhoven, 2001.

[2] General Particle Tracer (GPT), release 2.52, Pulsar Physics, De Bongerd 23, Soest, The Netherlands.

[3] W. Hackbusch, "Multi-Grid Methods and Applications", Springer, Berlin, 1985.

[4] B.E.C. Koltenbah, C.G. Parazzoli, "Space Charge Calculations of Elliptical Cross–Section Electron Pulses in PARMELA", Nucl. Instr. and Meth. in Phys. Res. A, Vol. 429, 1999, 281–286.

[5] G. Pöplau, U. van Rienen, J. Staats, T. Weiland, "Fast Algorithms for the Tracking of Electron Beams", Proceedings of EPAC 2000, Vienna, 2000, 1387–1389.

[6] G. Pöplau, U. van Rienen, "Multigrid Solvers for Poisson's Equation in Computational Electromagnetics", Proceedings of the 3rd Conference on Scientific Computating in Electrical Engineering (SCEE–2000), (U. van Rienen, D. Hecht, M. Günther, eds.), LNSCE **18**, Springer, Berlin, 2001, 169–176.

[7] J. Staats, T. Weiland, S. Kostial, A. Richter, "Tracking of Electron Beams with Numerically Determined Space Charge Forces", Proceedings of the 1999 Particle Accelerator Conference PAC'99, New York, 1999, 2740–2742.