UDC 519.6 Vestnik of St. Petersburg University. Series 10. 2014. Issue 4

H. Zhang, M. Berz

EFFICIENT TREATMENT OF SPACE CHARGE EFFECTS USING DA-BASED FMM METHODS

Michigan State University, East Lansing, Michigan, 48824, USA.

We present a fast multipole method based on differential algebraic methods for the calculation of the self-fields of all charged particles on each other inside a bunch in tracking simulations. It relies on an automatic multigrid-based decomposition of charges in near and far regions and the use of high-order differential algebra methods to obtain decompositions of far fields that lead to an error that scales geometrically with the order. Different from direct summation, the computational expense scales linear with the particle number. Some simulation results are presented to illustrate the practical performance of the method for realistic problems.

Keywords: space charge effects, fast mulitpole method, differential algebra.

The Fast Multipole Method in the Differential Algebra Framework. The Coulomb interaction between the particles inside a bunch of charged particles is one of the most important collective effects in the study of beam dynamics. This effect becomes more significant when the particle beam has lower energy, such as in contemplated time resolved electric microscopes, or has higher density, such as in modern high brilliance particle accelerators or free electron laser devices [1,2]. Since analytical approaches are only applicable to very special cases, numerical simulations are usually necessary to study this effect. Apparently the computational expense scales with N^2 if the Coulomb interaction of an N particle bunch is calculated by pairwise summation of Coulomb forces, which makes it practically difficult to simulate a large number of particles. To increase the efficiency, many algorithms such as the particle particle interaction (PPI) method [3,4,5,6], the particle in cell (PIC) method [7,8,9], the tree code [10,11], and the fast multipole method (FMM)[12,13,14,15], have been put forward and used in many simulation codes. The FMM, the key idea of which was first published in 1987[12], scales linearly with the number of particles N. It is based on a clever Taylor expansion of suitable superpositions of many-body fields. In the following, we will present the formulation of the FMM in the differential algebra (DA) framework and how we use it in tracking simulation. Some simulation results will be provided.

The Algorithm of the FMM. The basic idea of the FMM is to treat the source particles in different ways according to their distances to the observer. The potential or the field of those close to the observer is calculated directly by conventional pairwise Coulomb

interaction, while that of those far away from the observer is represented in terms of expansions involving powers of $1/r$, which we refer to as a far multipole expansion. Furthermore, far multipole expansions corresponding to separate regions can be translated and combined. Finally far multipoles can be locally expanded involving powers of r, which we call local expansions, in the region close to the observer. The local expansions can also be translated and combined.

Assume we are considering a bunch of charged particles, which can be enclosed in a cubic box. The cubic box can be divided into eight smaller cubic boxes of equal volume. The small boxes form a new level of boxes, and we call them the child boxes of the large cubic box. Similarly, the large cubic box is called the parent box of the smaller boxes. Each small box can itself be divided into eight child boxes in the same way for a new level of boxes. This process proceeds until a pre-specified level, resulting in a hierarchical structure of the boxes.

The boxes of the same level have at most three kinds of relations. For a box b, those boxes who touch it are called its neighbors. Furthermore, we call box b itself and all its neighbors the near region of b. Finally, those child boxes of b's parent boxes neighbors that are not b's neighborsare said to be in b's interaction list. The other boxes are in b's far region.

To better illustrate these three relations, an example of a 2D case is shown in fig. 1. The situation in the 3D case is conceptually similar.

The algorithm of the FMM can be described as follows:

1) cut the boxes of interests into the desired levels;

2) calculate the far multipole expansion of each box of the finest level according to the charged particles inside;

3) calculate the far multipole expansion of each box in the coarser levels by shifting the far multipole expansions of the child boxes into the center of their parent box and combining them;

4) for each box of each level, convert the far multipole expansions of the boxes in its interaction list into the local expansions inside itself and combine them;

5) from the coarsest level to the second finest level, translate the local expansion of each box into each of its child boxes and add it to the local expansion of the child box;

6) for each box of the finest level, evaluate the local expansion on each particle inside, which gives the potential or the field due to the particles outside the near region of the box. Then calculate the contribution from the particles in its near region by the pairwise Coulomb formula. Add up both results on each particle to get the potential or the field;

In the following we will present how to perform all the relevant far multipole expansions and the local expansions using the differential algebraic method. The following DA expansion can be calculated automatically by COSY Infinity 9.0 and the composition of two DA maps is also a general operation in COSY[16,17].

The Far Multipole Expansion from the Charged Particles. Assume we have a box centered at (x, y, z) enclosing n charged particles where the position of the i^{th} particle is denoted by (x_i, y_i, z_i). The far multipole expansion for this box can be calculated as follows:

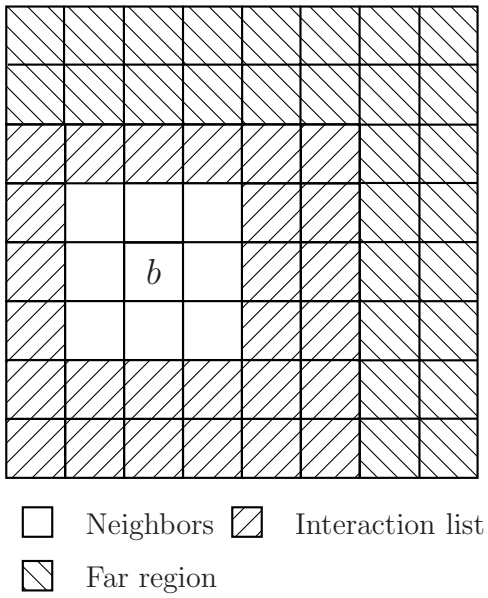

Figure 1: Three different relations of boxes of the same level

$$\begin{aligned}
\phi &= \sum_i^n \frac{1}{\sqrt{(x_i-x)^2+(y_i-y)^2+(z_i-z)^2}} = \\
&= \sum_i^n \frac{1/\sqrt{x^2+y^2+z^2}}{\sqrt{1+\frac{x_i^2+y_i^2+z_i^2}{x^2+y^2+z^2}-\frac{2x_ix}{x^2+y^2+z^2}-\frac{2y_iy}{x^2+y^2+z^2}-\frac{2z_iz}{x^2+y^2+z^2}}} = \\
&= \sum_i^n \frac{d_1}{\sqrt{1+(x_i^2+y_i^2+z_i^2)d_1^2-2x_id_2-2y_id_3-2z_id_4}}, \qquad (1)
\end{aligned}$$

where

$$d_1 = \frac{1}{\sqrt{x^2+y^2+z^2}} = \frac{1}{r}, \qquad d_2 = \frac{x}{x^2+y^2+z^2} = \frac{x}{r^2},$$
$$d_3 = \frac{y}{x^2+y^2+z^2} = \frac{y}{r^2}, \qquad d_4 = \frac{z}{x^2+y^2+z^2} = \frac{z}{r^2}.$$

It is readily apparent that using d_1, d_2, d_3 and d_4 as independent DA variables, the resulting operation can be directly carried out using DA arithmetic.

Translation of a Far Multipole Expansion. If we have a far multipole expansion in a cubic box centered at the origin $(0,0,0)$, we can translate it to (x'_o, y'_o, z'_o), the center of its parent box. The new DA variables are chosen as follows:

$$d'_1 = \frac{1}{\sqrt{(x-x'_o)^2+(y-y'_o)^2+(z-z'_o)^2}} = \frac{1}{\sqrt{x'^2+y'^2+z'^2}} = \frac{1}{r'},$$
$$d'_2 = \frac{x-x'_o}{r'^2} = \frac{x'}{r'^2}, \qquad d'_3 = \frac{y-y'_o}{r'^2} = \frac{y'}{r'^2}, \qquad d'_4 = \frac{z-z'_o}{r'^2} = \frac{z'}{r'^2}. \qquad (2)$$

In equations (2) (x', y', z') is the position of the observer in the new frame of $(x'_o, y'_o z'_o)$, and r' is the distance from the observer to the new origin (x'_o, y'_o, z'_o). The new DA variables and the old DA variables in equation (1) are related via

$$\begin{aligned}
d_1 &= \frac{1}{r} = \frac{1}{\sqrt{(x - x'_o + x'_o)^2 + (y - y'_o + y'_o)^2 + (z - z'_o + z'_o)^2}} = \\
&= \frac{1}{\sqrt{1/d_1'^2 + x_o'^2 + y_o'^2 + z_o'^2 + 2x'_o d'_2/d_1'^2 + 2y'_o d'_3/d_1'^2 + 2z'_o d'_4/d_1'^2}} = \\
&= \frac{d'_1}{\sqrt{1 + (x_o'^2 + y_o'^2 + z_o'^2) d_1'^2 + 2x'_o d'_2 + 2y'_o d'_3 + 2z'_o d'_4}} = d'_1 \cdot \sqrt{R}, \\
d_2 &= x \cdot d_1^2 = (x - x'_o + x'_o) \cdot d_1^2 = (d'_2 + x'_o d_1'^2) \cdot R, \\
d_3 &= y \cdot d_1^2 = (y - y'_o + y'_o) \cdot d_1^2 = (d'_3 + y'_o d_1'^2) \cdot R, \\
d_4 &= z \cdot d_1^2 = (z - z'_o + z'_o) \cdot d_1^2 = (d'_4 + z'_o d_1'^2) \cdot R,
\end{aligned} \tag{3}$$

where

$$R = \frac{1}{1 + (x_o'^2 + y_o'^2 + z_o'^2) d_1'^2 + 2x'_o d'_2 + 2y'_o d'_3 + 2z'_o d'_4}.$$

If we substitute equation (3) into equation (1), we obtain the far multipole expansion in the new frame. If one considers equation (1) as a map $M_{\mathrm{c2m}}(d_1, d_2, d_3, d_4)$ and equation (3) as a map $M_1(d'_1, d'_2, d'_3, d'_4)$, the substitution is actually merely the composition of the two maps

$$M_{\mathrm{m2m}} = M_{\mathrm{c2m}} \circ M_{1,}$$

which is a common operation in DA methods, and which here yields the expression of ϕ in the new frame with the new DA variables.

Conversion of a Far Multipole Expansion into a Local Expansion. Assuming box a centered at $(0, 0, 0)$ is in the interaction list of box b centered at (x'_o, y'_o, z'_o), a far multipole expansion inside a can be converted into a local expansion inside b. The new DA variables can be chosen as the coordinates of the observer (x', y', z') in the new frame of (x'_o, y'_o, z'_o)

$$d'_1 = x', \quad d'_2 = y', \quad d'_3 = z',$$

The old DA variables and the new DA variables are related via

$$\begin{aligned}
d_1 &= \frac{1}{\sqrt{x^2 + y^2 + z^2}} = \frac{1}{\sqrt{(x'_o + d'_1)^2 + (y'_o + d'_2)^2 + (z'_o + d'_3)^2}} = \sqrt{R}, \\
d_2 &= \frac{x}{x^2 + y^2 + z^2} = \frac{x'_o + d'_1}{(x'_o + d'_1)^2 + (y'_o + d'_2)^2 + (z'_o + d'_3)^2} = (x'_o + d'_1) \cdot R, \\
d_3 &= \frac{y}{x^2 + y^2 + z^2} = \frac{y'_o + d'_2}{(x'_o + d'_1)^2 + (y'_o + d'_2)^2 + (z'_o + d'_3)^2} = (y'_o + d'_2) \cdot R, \\
d_4 &= \frac{z}{x^2 + y^2 + z^2} = \frac{z'_o + d'_3}{(x'_o + d'_1)^2 + (y'_o + d'_2)^2 + (z'_o + d'_3)^2} = (z'_o + d'_3) \cdot R,
\end{aligned} \tag{4}$$

where

$$R = \frac{1}{(x'_o + d'_1)^2 + (y'_o + d'_2)^2 + (z'_o + d'_3)^2}.$$

If we call equation (4) the map $M_2(d'_1, d'_2, d'_3)$, the local expansion can again be written as a composition of M_{c2m} and M_2 as

$$M_{\mathrm{m2l}} = M_{\mathrm{c2m}} \circ M_2.$$

Translation of a Local Expansion. A local expansion in a box centered at $(0, 0, 0)$ can be translated into one of its child boxes centered at (x'_o, y'_o, z'_o). If we choose the new DA variables as the coordinates of the observer (x', y', z') in the new frame of (x'_o, y'_o, z'_o), it is easy to see that the relation of the old DA variables and the new DA variables is just a shift:

$$d_1 = x'_o + d'_1, \quad d_2 = y'_o + d'_2, \quad d_3 = z'_o + d'_3. \tag{5}$$

We call equation (5) the map M_3, then the new local expansion can be calculated by composing M_{m2l} with M_3 as

$$M_{\mathrm{l2l}} = M_{\mathrm{m2l}} \circ M_3.$$

With the local expansion of the potential, which is an n^{th} order polynomial of the observer's position (x, y, z), we can calculate the potential of the charged particles outside the near region of a box. Taking the derivative with respect to the position (x, y, z), we get the expansion of the field (E_x, E_y, E_z) up to the order $n - 1$, by which we can calculate the field. The potential and the field of the charged particles in the near region are calculated by the pairwise Coulomb formula. Altogether, the above four operations of far field expansion, far field translation, local expansion, and local translation form the core of the DA-FMM method and are sufficient for the computation of all space charge fields; for a more detailed discussion, we refer to [18].

Use of the FMM in Tracking Simulations. The FMM described above always starts from a cubic box, then cuts it into eight child boxes and keeps all of them. The method works well when the bunch sizes in all three dimensions are close to each other. However, in practice this is not always the case. In fact, usually, the bunch sizes evolve with time, and the bunch may have an oblate shape where one of the dimensions is much smaller than the others, or a prolate shape where one of the dimensions is significantly larger than the other two. In these cases, the FMM above will lose some of its efficiency because a large number of the resulting boxes are empty.

However, what plays the critical role in the FMM algorithm is the hierarchical relation between the boxes, while the details of the cutting can be modified as necessary. We will now discuss a more efficient way of cutting boxes which still maintains the necessary hierarchical relation.

Strategy of Cutting Boxes. Assume we know the length of the bunch in all the three dimensions (l_x, l_y, l_z). Without loss of generality, we assume $l_x \geq l_y \geq l_z$. We also know the total number of the particles N, and the average number of particles inside each box of the finest level n, from which we can get the least number of boxes needed N_b. The new idea of box cutting can be described as follows:

1) Enclose the bunch by a cubic box whose size is slightly greater or equal to the longest side of the bunch l_x. We say this box is of level zero. Without loss of generality we assume this box is centered at $(0, 0, 0)$.

2) Try to cut this cube box from the center into eight equal size cube boxes. Compare the coordinates (c_x, c_y, c_z) of the center of each child box with $(l_x/2, l_y/2, l_z/2)$. If a child

box whose center satisfies $|c_i| < l_i/2$ for all the three directions, it is accepted. Otherwise, if in any direction, $|c_i| < l_i/2$ is not satisfied, c_i is set to zero to get a new center point with the coordinates of the other directions unchanged, and the child box is replaced by a new box that is centered at the newly obtained point. In this way the number of child boxes is decreased. For example, assuming two child boxes with centers (x_1, y_1, z_1) and (x_1, y_1, z_2), if in x and y directions, $|c_i| < l_i/2$ is satisfied, but the absolute values of both z_1 and z_2 are greater than $l_z/2$, they will be replaced by one box whose center is $(x_1, y_1, 0)$. The side length of the child boxes is set to be half of the parent box.

3) Cut each box of the current finest level in the same way as 3) to obtain the boxes of a finer level.

4) Repeat 3) until the box number at the finest level is greater than or equal to N_b.

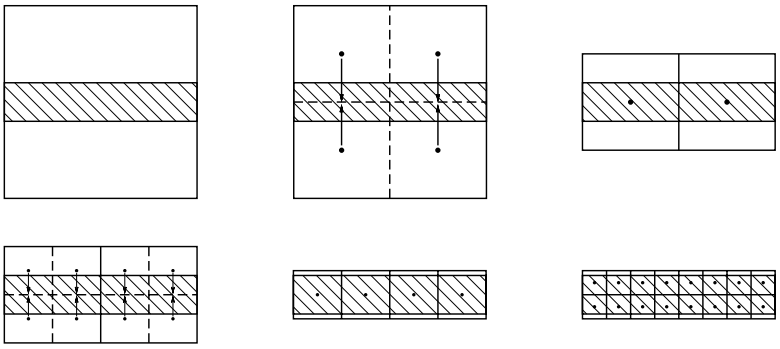

Figure 2: The box cutting strategy

To make the idea clearer, let us consider a 2D example as shown in fig. 2. The shaded part shows the dimensions of the bunch. First, we enclose the bunch by a square box. Then we try to cut the square box into four child boxes, but we notice that the y coordinates of all the centers of the child boxes do not satisfy $|c_y| < l_y/2$. So we set the y coordinates of them zero, and the four center points fall into two new points as fig. 2 shows. We use these two points as the centers of the new child boxes whose side length is half of their parent box, so that we get two child boxes of the first level instead of four. Then we repeat the same cutting process for each box of the first level, we get two child boxes for each of them. So we have four boxes of the second level. When we cut the boxes of the second level, we find all the centers of the child boxes satisfy $|c_i| < l_i/2$, so we accept all of them and get 16 boxes of the third level. As shown in fig. 2, when we use one square box to enclose the bunch in the beginning, there is a lot of blank place in the box. But after we find the proper cutting, the dimension of the boxes fits the bunch well. In this way, we avoid the problem of a large number of empty boxes, which negatively affects the performance of the FMM method.

Although the strategy is described as above, in practice we do not need to check the centers of all child boxes for each cutting. What we really need to know is how many times we need to cut each dimension for each level. For this purpose, we can simply compare the size of each dimension to a scale. For the first level the scale is set to be the size of the largest dimension, and for each finer level the scale is divided by two. Each dimension that is larger than half of the scale will be cut. If a dimension is cut in some level, it will definitely be cut in all the finer levels. We label each box by four numbers (l, n_x, n_y, n_z). l is the level, and n_i is the index of box in the i^{th} direction; $n_i = 0$ if the i^{th} direction is not cut

in the l^{th} level, or $n_i \in [1, 2^{l_i}]$ if the i^{th} direction is cut l_i times in the l^{th} level. Assuming the zero level box is centered at the origin $(0, 0, 0)$, the center position of each box can be simply calculated from its label as

$$c_{ln_i} = bs_l \cdot n_i + 0.5 \cdot bs_l \cdot (1 - 2^{l_i})$$

where c_{ln_i} is the center position in the i^{th} dimension of the $\vec{n}^{\text{th}}$ box of the l^{th} level, and bs_l is the box size of the l^{th} level boxes. The center positions are needed when we translate or convert the far multipole expansions and the local expansions.

Frame Rotation. Another method to avoid empty boxes is to set the frame consistent with the principal axes of the bunch as shown in fig. 3 for the 2D case. We start from a frame whose origin is the center of mass of the bunch. The position of the i^{th} particle is $\mathbf{X}_i = (x_{i1}, x_{i2}, x_{i3})$. If the coordinate axes are denoted by x_j, $j = 1, 2, 3$, then the moment of inertia coefficient matrix element I_{jk} can be written as

$$I_{jk} = \sum_i m_i (r_i^2 \delta_{jk} - x_{ij} x_{ik}),$$

where i is the index of the particle and $r_i^2 = x_{i1}^2 + x_{i2}^2 + x_{i3}^2$. In the principal axes frame, only the diagonal elements of the matrix $\mathbf{I}$ are nonzero. Assuming a matrix $\mathbf{P}$ that diagonalizes $\mathbf{I}$ as $\mathbf{P}^{-1}\mathbf{I}\mathbf{P} = \mathbf{I}' = \text{diag}\{I'_{11}, I'_{22}, I'_{33}\}$, the position of the particle in the principal axes frame is $\mathbf{X}' = \mathbf{P}^{-1}\mathbf{X}$. If the electric field in the principal axes frame is $\mathbf{E}'$, the field in the original frame is $\mathbf{E} = \mathbf{P}\mathbf{E}'$.

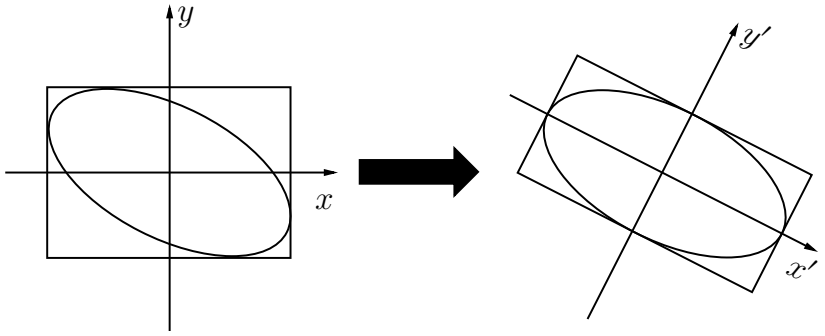

Figure 3: Frame rotation

So when we have a bunch in the lab frame, we first shift the origin of the frame to the center of mass of the bunch, and rotate the frame so that it is consistent with the principal axes of the bunch. We then cut the boxes as stated in the previous subsection ("Strategy of Cutting Boxes") and calculate the potential or/and the field. Finally we translate the field back to the lab frame.

This approach automatically determines an initial enclosing box of smallest possible volume.

Examples of Tracking Simulations. We used the FMM in tracking simulations, and compared the results with those obtained by the much more expensive use of a pairwise Coulomb formula. One example is shown in fig. 4. We have a bunch of 2 082 000

electrons with a three dimensional Gaussian distribution. The initial size of the bunch is $(115\mu\text{m}, 81\mu\text{m}, 1\mu\text{m})$. We use 100 000 macroparticles, each of which represents 20.82 electrons, and a fourth order Runge – Kutta integrator with fixed step size to simulate the free expansion of the bunch without any external field. The step size is 1 ps, and the simulation runs for 100 steps to 100 ps. The results of the FMM with a fifth order DA expansion are presented in dots, and the results of the pairwise Coulomb formula are presented in lines. The evolution of the bunch size with respect to time in z direction is shown on the left, and those in x and y directions are shown on the right. The final bunch sizes are $(155.21\mu\text{m}, 127.08\mu\text{m}, 103.15\mu\text{m})$ and $(155.29\mu\text{m}, 127.06\mu\text{m}, 103.13\mu\text{m})$ respectively. The relative difference is less than 0.0515%.

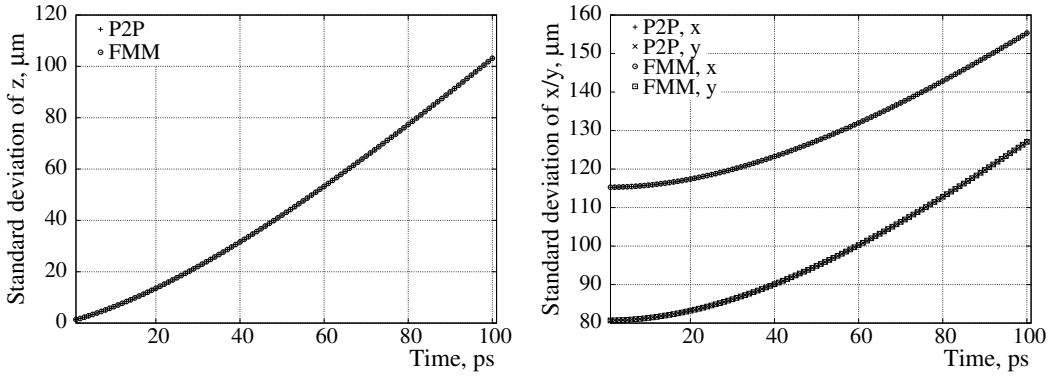

Figure 4: Simulation of the free expansion of an electron bunch using the FMM and the pairwise Coulomb formula

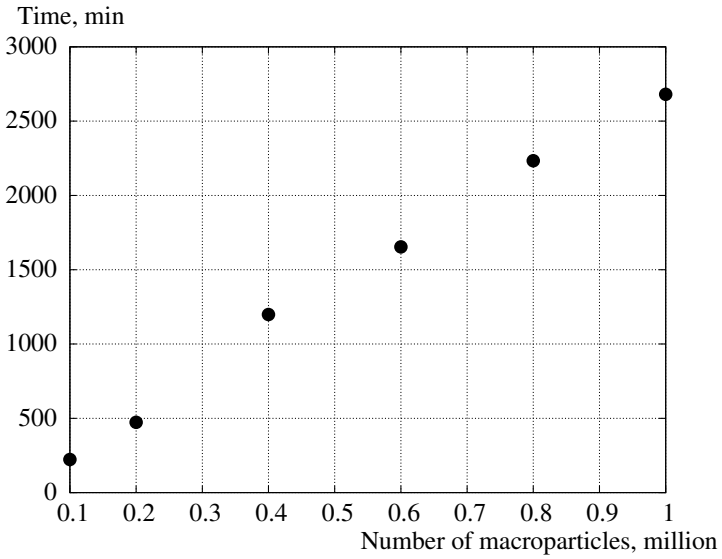

Figure 5: Computational expense for the simulations for different numbers of macroparticles

Fig. 5 shows the computational expense for the simulations of the free expansion of a proton bunch. The bunch has 2 082 000 protons with a uniform distribution in all dimensions, and its initial size is $(66.45\mu\text{m}, 46.76\mu\text{m}, 0.58\mu\text{m})$. We use a fourth order Runge-Kutta integrator with fixed step size 1 ps and simulate to 100 ps on our computer with 4 Core (8 Hyperthreaded) Intel Xeon Processor X5677 running at approximately 3.5GHz. Our simulation program is a single process program, although the FMM can be parallelized in principle. The final bunch size is $(66.52\mu\text{m}, 46.84\mu\text{m}, 29.57\mu\text{m})$. The macroparticle numbers range from 100 000 to 1 000 000, and the computational expenses for the simulations with different numbers of macroparticles are presented in fig. 5. Apparently the computational expense does indeed increase linearly with the number of macroparticles, as expected from the above theoretical arguments.

Future Work. We have shown that the FMM can be used in the tracking simulation, and its computational expense scales with the macroparticle number. However, we also notice that the efficiency of treating a bunch with the Gaussian distribution is worse than that of a bunch with a uniform distribution when they have the same number of macroparticles and similar shapes. Some examples are presented in table. We calculated the electric field for a bunch of 1 000 000 electrons with either the Gaussian distribution or the uniform distribution of varying bunch shapes. (dx, dy, dz) is the r.m.s. size of the bunch, and (lx, ly, lz) is how many times the bunch is divided in (x, y, z) direction. The single process program runs on our computer with 4 Core (8 Hyperthreaded) Intel Xeon Processor X5677 running at approximately 3.5 GHz. We can see the computational expense of a bunch with a uniform distribution is much less than that of a bunch with the Gaussian distribution of similar shape. This is because the FMM has the best efficiency when all of the boxes of the finest level have the same number of particles inside. On the other hand, when a bunch has a Gaussian distribution, its center has a much higher charge density than its edge, which results in the boxes at the edge being either empty or enclosing few particles and the boxes at the center enclosing much more particles than expected. To solve this problem, we are working on the adaptive FMM, by which we cut the boxes into finer levels where the charge density is higher, so that we can make sure all the boxes of the finest levels (different lowest levels at different positions) enclose similar numbers of particles, which guarantees the efficiency [13,19].

Computational expenses for the Gaussian/uniform distribution bunches

Distr.	dx, μm	dy, μm	dz, μm	lx	ly	lz	Time, min
GS	99.91	99.95	100.03	5	5	5	16.10
U	99.91	99.96	100.00	5	5	5	5.93
GS	1.00	99.95	100.03	0	6	6	10.64
U	1.00	99.96	100.00	0	6	6	3.00
GS	1.00	1.00	100.03	3	3	9	10.44
U	1.00	1.00	100.00	3	3	9	6.53

Acknowledgments. For various fruitful discussions, we would like to thank Kyoko Makino, Alex Wittig, and Ravi Jagasia. Financial support was appreciated from Michigan State University and the US Department of Energy.

References

1. Jansen G. H. Coulomb interactions in particle beams. Advances in electronics and electron physics supplement 21. Boston: Academic Press, 1990, 546 p.

2. Kruit P., Jansen G. H. Handbook of charged particle optics. Second Ed. Boca Raton: CRC Press, 2009, 528 p. (pp. 341–391).

3. Martini M., Prome M. Computer studies of beam dynamics in a proton linear accelerator with space charge. *Part. accel.*, 1971, vol. 2, pp. 289–299.

4. Garnett R. W., Wangler T. P. Space-charge calculation for bunched beams with 3-D ellipsoidal symmetry. *Proc. of the 1991 IEEE Particle accelerator conference* (APS Beam Physics), 1991, 330 p.

5. Lapostolle P. M., Lombardi A. M., Nath S., Tanke E., Valero S., Wangler T. P. A new approach to space charge for linac beam dynamics codes. *Proc. of the 18th Intern. linear accelerator conference*, 1996, 375 p.

6. Lapostolle P. M., Lombardi A. M., Tanke E., Valero S., Garnett R. W., Wangler T. P. A modified space charge routine for high intensity bunched beams. *Nuclear instruments and methods A*, 1996, vol. 379 (1), pp. 21–40.

7. Pichoff N., Lagniel J. M., Nath S. Simulation results with an alternate 3D space charge routine, PICNIC. *Proc. of the XIX Intern. Linac conference*, 1998, vol. 141, pp. 23–28.

8. Poplau G., van Rienen U., van der Geer B., de Loos M. Multigrid algorithms for the fast calculation of space-charge effects in accelerator design. *IEEE Transactions on magnetics*, 2004, vol. 40 (2 Part 2), pp. 714–717.

9. Batygin Y. K. Particle-in-cell code BEAMPATH for beam dynamics simulations in linear accelerators and beamlines. *Nuclear instruments and methods A*, 2005, vol. 539, issue 3, pp. 455–489.

10. Barnes J., Hut P. A hierarchical O (N log N) force-calculation algorithm. *Nature*, 1986, vol. 324 (4), pp. 446–449.

11. Barnes J. E. A modified tree code: Don't laugh; it runs. *Journal of computational physics*, 1990, vol. 87 (1), pp. 161–170.

12. Greengard L., Rokhlin V. A fast algorithm for particle simulations. *Journal of computational physics*, 1987, vol. 73 (2), pp. 325–348.

13. Carrier J., Greengard L., Rokhlin V. A fast adaptive multipole algorithm for particle simulations. *SIAM Journal on scientific and statistical computing*, 1988, vol. 9 (4), pp. 669–686.

14. Beatson R. K., Greengard L. A short course on fast multipole methods. Wavelets, multilevel methods and elliptic PDEs. Oxford: Clarendon Press; New York: Oxford University Press, 1997, 302 p. (pp. 1–37).

15. Shanker B., Huang H. Accelerated cartesian expansions – A fast method for computing of potentials of the form $r^{-\nu}$ for all real ν. *Journal of computational physics*, 2007, vol. 226 (1), pp. 732–753.

16. Berz M. Modern Map Methods in Particle Beam Physics. San Diego: Academic Press, 1999, 318 p.

17. Berz M., Makino K. COSY INFINITY Version 9.1 programmer's manual: Technical report MSUHEP-101214. Michigan, USA: Department of Physics and Astronomy, Michigan State University, East Lansing, 2011, 93 p. (see also http://cosyinfinity.org).

18. Zhang H., Berz M. Fast multipole method in the differential algebra framework. *Nuclear instruments and methods A*, 2011, vol. 645, pp. 338–344 (doi:10.1016/j.nima.2011.01.053).

19. Cheng H., Greengard L., Rokhlin V. A fast adaptive multipole algorithm in three dimensions. *Journal of computational physics*, 1999, vol. 155 (2), pp. 468–498.

He Zhang – doctor of philosophy; e-mail: zhanghe@msu.edu
Martin Berz – doctor of philosophy, professor; e-mail: berz@msu.edu