Efficient Particle Simulation of a Virtual Cathode Using a Grid-Free Treecode Poisson Solver

Andrew J. Christlieb, Robert Krasny, and John P. Verboncoeur, *Member, IEEE*

***Abstract*—An efficient grid-free numerical method is developed for particle simulations in plasma dynamics. The method relies on a treecode algorithm to solve the Poisson equation for interacting charged particles. Such algorithms have been extensively used in astrophysics, fluid dynamics, and molecular dynamics, and our aim is to assess their capability in plasma dynamics in comparison with traditional mesh-based methods such as particle-in-cell (PIC). It is well-known that PIC simulations have difficulty resolving local particle interactions within a grid cell, and we expect the grid-free treecode algorithm to have an advantage for problems involving small-scale features and highly nonuniform particle distributions. To illustrate this point we apply direct summation, treecode, and PIC methods to simulate a virtual cathode in one dimension. The treecode and PIC methods are found to be much faster than direct summation, but the treecode yields the correct solution while spurious features are present in the PIC results. The present work treats a one-dimensional problem, but the treecode algorithm generalizes directly to higher dimensions.**

***Index Terms*—Coulomb potential, grid-free, multipole expansion, particle-in-cell method, Poisson solver, treecode algorithm, virtual cathode.**

I. Introduction

A COMMON approach in simulating a low-density plasma is to model the system from the Lagrangian perspective as a cloud of charged particles [1]. The evolution is obtained by integrating the particles forward in time by Newton's equation under the action of internal and external forces. The most expensive part of the process is the self-consistent computation of the particle forces. For an electrostatic system, the force F_i acting on particle x_i due to all the other particles is

$$F_i = -q_i \sum_{\substack{j=1 \\ j\neq i}}^{N} \nabla_x G(x_i, x_j) q_j \tag{1}$$

where q_i is the charge on particle i, $G(x, y)$ is the Green's function or Coulomb potential at x due to a particle of unit charge located at y, and N is the total number of particles. Since the force must be computed for each particle in the system, the direct summation approach is an $O(N^2)$ process. The goal of this work is to develop a more efficient procedure for computing the electrostatic forces that arise in charged particle simulations.

The particle-in-cell (PIC) method is an effective and widely used option for reducing the cost of the particle force computation [1], [2]. This method assigns the particle charges to a regular mesh using a weighting function. The force is then obtained at the mesh points using a fast mesh-based Poisson solver, and finally it is interpolated back to the particle locations. This is typically an $O(N \log N)$ process, a vast improvement over direct summation, but for complicated domains or problems involving small-scale features such as boundary layers and shocks, an exceedingly fine mesh may be needed to resolve the solution. Several approaches are currently used to overcome this difficulty such as the particle–particle/particle–mesh ($\mathrm{P^3M}$) method [2] and adaptive mesh refinement [3], but here we pursue a grid-free alternative based on treecode algorithms.

Treecode algorithms were originally developed to compute gravitational forces in astrophysics [4], and they are also used extensively in fluid dynamics [5] and molecular dynamics [6]. These algorithms replace the particle–particle interactions by suitably chosen particle–cluster interactions which are approximated by a multipole expansion. The Fast Multipole method also makes use of cluster–cluster interactions [7], [8]. It is natural to use these algorithms in plasma dynamics [9], [10], and while they are often applied to problems with free-space or periodic boundary conditions, they can also be adapted to handle problems on a bounded domain as in the example given below.

Our aim is to compare treecode algorithms with traditional particle–mesh methods, and in this preliminary work we restrict attention to a collisionless plasma system. In particular, we consider the classical problem involving the formation of a virtual cathode (VC) in the gap between two biased electrodes, one of which is a thermionic emitter [11], [12]. Although this is a relatively simple one-dimensional (1-D) test case, the solution is difficult to resolve on a fixed mesh due to the presence of temporal oscillations and spatially localized features. In the following sections, we first describe the virtual cathode problem, then review the basic ideas underlying our approach to treecode algorithms [13], [14], and finally compare the results of direct summation, treecode, and PIC simulations.

II. Virtual Cathode

A. Physical Description

In the 1-D virtual cathode problem, two infinite parallel flat plates are located at $x = \alpha, \beta$ with applied voltages V_α and V_β, respectively. The gap between the plates is initially a vacuum. One of the plates is heated to the point where electrons are

Manuscript received August 30, 2003; revised December 2, 2003. This work was supported by National Science Foundation under Grant DMS-9977371 and Grant DMS-0107187.

A. J. Christlieb and R. Krasny are with the Mathematics Department, University of Michigan, Ann Arbor, MI 48109-1109 USA (e-mail: christli@umich.edu).

J. P. Verboncoeur is with the Electrical and Computer Engineering Department, University of California, Berkeley, CA 94720-1770 USA.

Digital Object Identifier 10.1109/TPS.2004.826146

emitted and a current flows across the gap. For a sufficiently high emission rate, the maximum current between the plates approaches a limiting value J_{CL}, predicted by the Child–Langmuir law. When this happens, a virtual cathode is formed somewhere between the two plates, and the emitted electrons turn around and strike the emitting plate. As the electrons repel each other, the potential minimum rises, thereby allowing the emitted electrons to again start crossing the gap, and the entire sequence repeats. In this work, we consider the classical formulation of the problem in which the electrodes are held at ground and the emitted electrons are given an initial velocity [11].

B. Mathematical Formulation

The total potential $\Phi(x)$ satisfies the Poisson equation

$$\frac{d^2\Phi}{dx^2} = -\frac{\rho}{\epsilon_0} \tag{2}$$

subject to boundary conditions

$$\Phi(\alpha) = V_\alpha, \quad \Phi(\beta) = V_\beta \tag{3}$$

where $\rho(x)$ is the charge density and $x = \alpha$ and $x = \beta$ are the left and right endpoints of the gap. The charge density is

$$\rho(x) = \sum_{j=1}^{N} \delta(x - x_j) q_j \tag{4}$$

where q_j and x_j are the charge and location of particle j, and $\delta(x)$ is the Dirac delta function. The total potential is written $\Phi(x) = \Phi_P(x) + \Phi_H(x)$, where Φ_P is a particular solution satisfying

$$\frac{d^2\Phi_P}{dx^2} = -\frac{\rho}{\epsilon_0} \tag{5}$$

and Φ_H is a homogeneous solution satisfying

$$\frac{d^2\Phi_H}{dx^2} = 0. \tag{6}$$

The particular solution is chosen to be

$$\Phi_P(x) = \frac{1}{\epsilon_0} \sum_{j=1}^{N} G(x, x_j) q_j \tag{7}$$

where $G(x, y) = (1/2)|x - y|$ is the Green's function for the 1-D Poisson equation. The homogeneous solution has the form

$$\Phi_H(x) = Ax + B \tag{8}$$

and by imposing the boundary conditions in (3) we obtain

$$\begin{aligned} A &= \frac{1}{\beta - \alpha} \left(V_\beta - V_\alpha + \Phi_P(\alpha) - \Phi_P(\beta) \right) \\ B &= \frac{\beta}{\beta - \alpha} \left(V_\alpha - \Phi_P(\alpha) \right) - \frac{\alpha}{\beta - \alpha} \left(V_\alpha - \Phi_P(\beta) \right). \end{aligned} \tag{9}$$

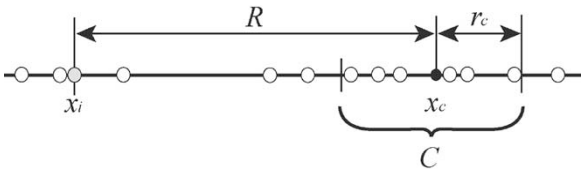

Fig. 1. Schematic diagram of a particle–cluster interaction in one dimension, where x_i: particle, C: cluster, x_c: cluster center, r_c: cluster radius, and $R = |x_i - x_c|$: particle–cluster distance.

The induced force on particle i is then given by

$$F_i = -q_i \left(\frac{d\Phi_P(x_i)}{dx} + \frac{d\Phi_H(x_i)}{dx} \right). \tag{10}$$

Once the force is known, the particles are advected by Newton's equation, $m_i \ddot{x}_i = F_i$. The component of force arising from the particular solution Φ_P is evaluated using the treecode, as described next.

III. Treecode Algorithm

We review the formulation of the treecode in one dimension, bearing in mind that the algorithm generalizes directly to higher dimensions [4], [9], [13], [14]. The key idea is to replace the particle–particle interactions by particle–cluster interactions.

A. Particle–Cluster Interactions

The component of force due to the particular solution is determined from (7)

$$\begin{aligned} F_i &= -q_i \frac{d\Phi_P(x_i)}{dx} = -\frac{q_i}{\epsilon_0} \sum_{\substack{j=1 \\ j \neq i}}^{N} \partial_x G(x_i, x_j) q_j \\ &= -\frac{q_i}{\epsilon_0} \sum_{C} \sum_{j=1}^{N_C} \partial_x G(x_i, y_j) q_j = \sum_{C} F_i^C \end{aligned} \tag{11}$$

where $C = \{y_j, j = 1, \ldots, N_C\}$ is a cluster of particles and

$$F_i^C = -\frac{q_i}{\epsilon_0} \sum_{j=1}^{N_C} \partial_x G(x_i, y_j) q_j \tag{12}$$

is the interaction between particle x_i and cluster C. Fig. 1 depicts a particle–cluster interaction. Note that the expression for F_i should omit the singular term $x_j = x_i$. The procedure for choosing the clusters will be explained, and for now, it is enough to assume that the clusters entering in (11) are nonoverlapping, and their union is the entire particle distribution with x_i deleted.

Next, consider the Taylor expansion of the Green's function about the center of a cluster y_c

$$\begin{aligned} G(x_i, y_j) &= G\left(x_i, (y_j - y_c) + y_c\right) \\ &\simeq \sum_{k=0}^{p} \frac{1}{k!} \partial_y^k G(x_i, y_c)(y_j - y_c)^k \end{aligned} \tag{13}$$

where p is the order of the approximation. Substituting this into (12) and using the relation $\partial_x G = -\partial_y G$, the particle–cluster force can be expressed as

$$\begin{aligned} F_i^C &\simeq \frac{q_i}{\epsilon_0}\sum_{j=1}^{N_C}\sum_{k=0}^{p}\frac{1}{k!}\partial_y^{k+1}G(x_i,y_c)(y_j-y_c)^k q_j \\ &= \frac{q_i}{\epsilon_0}\sum_{k=0}^{p}\frac{1}{k!}\partial_y^{k+1}G(x_i,y_c)\sum_{j=1}^{N_C}(y_j-y_c)^k q_j \\ &= \frac{q_i}{\epsilon_0}\sum_{k=0}^{p}T_k(x_i,y_c)M_k^C \end{aligned} \tag{14}$$

where $T_k(x_i,y_c)$ is the kth Taylor coefficient of the Green's function and M_k^C is the kth moment of the cluster. This procedure separates the calculation of the particle–cluster interaction F_i^C into two operations, computing the cluster moments (which are stored and reused for other particles x_i) and computing the Taylor coefficients of the Green's function (which do not depend on the particle distribution within the cluster). This means that the particle–cluster interaction F_i^C can be evaluated using the Taylor approximation at a cost which is essentially independent of the number of particles in the cluster. For each particle force F_i, there are typically $O(\log N)$ clusters entering into (11); hence, evaluating all the particle forces this way is an $O(N\log N)$ process. Similar considerations apply in two and three dimensions, and in those cases one is dealing with a classical multipole expansion [7], [13], [14]. Next, we explain how the particles are divided into clusters.

B. Tree Construction

The treecode uses a hierarchical tree structure to define the particle clusters. First we define several quantities and operations:

X	$=\{x_1,\ldots,x_N\}$ root cluster of all particles;
$\Omega(C)$	number of particles in cluster C, e.g., $\Omega(X)=N$;
$MP(C)$	midpoint $=(1/2)(\max(C)+\min(C))$;
$L(C)$	left child $=\{x_i \mid x_i\in C, x_i<MP(C)\}$;
$R(C)$	right child $=\{x_i \mid x_i\in C, x_i>MP(C)\}$;
$\nu(C)$	index operator $=\{i,\ldots,j\}$, where $C=\{x_i,\ldots,x_j\}$;
N_0	maximum number of particles in a leaf cluster.

The tree is composed of data structures called *nodes*. Each node is associated with a cluster; it contains the number of particles in the cluster, the particle indexes within a global storage array, and pointers to the left and right child nodes of the cluster. The nodes of the tree are defined using the following recursive procedure.

$\mathbf{buildtree}(C)$

- if $\Omega(C)<N_0$, then
  - $node=\{\Omega(C),\nu(C),(null,null)\}$
  - return $node$
  - stop
- else
  - $L=L(C)$, $R=R(C)$
  - $nodeL=\mathbf{buildtree}(L)$, $nodeR=\mathbf{buildtree}(R)$
  - $node=\{\Omega(C),\nu(C),(nodeL,nodeR)\}$
  - return $node$
- end.

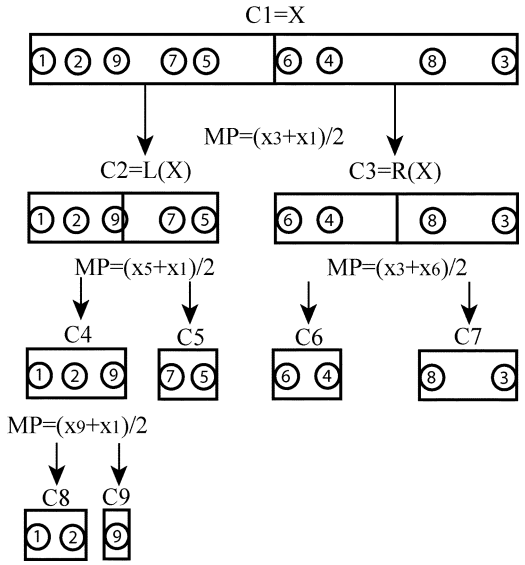

Fig. 2. Schematic example of 1-D tree construction with $N=9$, $N_0=2$. Algorithm produces a tree having nine clusters and four levels.

The procedure is initiated by calling root $=\mathbf{buildtree}(X)$ and on return the set of all nodes defines the tree. An example is shown schematically in Fig. 2. The 1-D case is especially simple, but a similar procedure is used for clustering particles in 2-D and 3-D [4].

C. Force Evaluation

As indicated in (11), the force F_i acting on a given particle x_i is expressed as a sum of particle–cluster forces F_i^C for suitably chosen clusters C. The treecode has two options for evaluating each particle–cluster term F_i^C; it can apply either the Taylor approximation in (14) or the exact direct summation in (12). The present simulations used the Barnes–Hut criterion [4] for deciding when the Taylor approximation is sufficiently accurate and this is explained next.

Referring to Fig. 1, let x_c be the cluster center, r_c the cluster radius, and $R=|x_i-x_c|$ the particle–cluster distance. A user-specified error tolerance parameter $\theta<1$ is also defined, and we say that particle x_i is *well separated* from cluster C if the relation $R>r_c/\theta$ holds. In that case, the Taylor approximation is used to evaluate F_i^C. Otherwise, the code considers interactions between particle x_i and the children of cluster C. The rationale is that the children have smaller radii and hence the error criterion is more likely to be satisfied. If the cluster C has no children, then it is a leaf of the tree, and F_i^C is evaluated by direct summation.

In summary, given a particle x_i, the force induced by nearby particles is computed directly and the force induced by well-separated particles is computed using a particle–cluster Taylor approximation. The number of terms needed in the approximation to guarantee a desired level of accuracy can be determined on the fly for each particle–cluster interaction and in several important cases (including two and three dimensions with free-space

boundary conditions); the necessary Taylor coefficients of the Green's function can be constructed using a three term recurrence relation [13], [14].

The force evaluation procedure is actually much simpler in the present $1D$ situation since the relevant Green's function is piecewise linear and only the first derivative is nonzero. The particle–cluster force in this case is given by

$$F_i^C = -\frac{q_i}{\epsilon_0}\partial_x G(x_i, x_c) Q_C \tag{15}$$

where

$$Q_C = \sum_{j=1}^{N_C} q_j \tag{16}$$

is the zero-order moment or total charge in cluster C. In particular, this means that the first-order Taylor approximation is exact in one dimension. The treecode computes the particle force by setting $F_i = \mathbf{computeF}(x_i, root)$, using the following recursive procedure.

$\mathbf{computeF}(x_i, nodeC)$:

- if $R > r_C/\theta$, then
  - compute Q_C and x_c (unless already computed)
  - compute F_i^C using first-order Taylor approximation
  - return F_i^C
- else
  - if $nodeL = null$, then
    - compute F_i^C by direct summation
  - else
    - $FR = \mathbf{computeF}(x_i, nodeR)$
    - $FL = \mathbf{computeF}(x_i, nodeL)$
    - $F_i^C = FR + FL$
  - end
  - return F_i^C
- end

IV. Results

We investigated the behavior of a 1-D virtual cathode using treecode (TC), direct summation (DS), and PIC simulations. The DS simulation is exact, and in all cases, the TC agreed with the DS (to within roundoff error); as noted, this is because the first-order Taylor approximation is exact in one dimension. The injection current was varied by adjusting the charge/mass ratio of the particles. With a low injection current (0.05), the normalized electron density (n/n_0) converges to a steady state, as shown in Fig. 3. In this case, the PIC and TC/DS simulations agree very well. The PIC simulation used standard area weighting and the meshsize $(\Delta x/L = 1/30)$ was chosen by decreasing the value until the solution converged to a steady state. All three methods used the same procedures for injecting particles at the emitting plate and deleting them at the absorbing plate.

Next, we examine results obtained using a higher value for the injection current (0.1), just above the threshold where the VC forms. In this case, the solution converges to a time-periodic state with spatially localized features. Fig. 4 shows the time trace

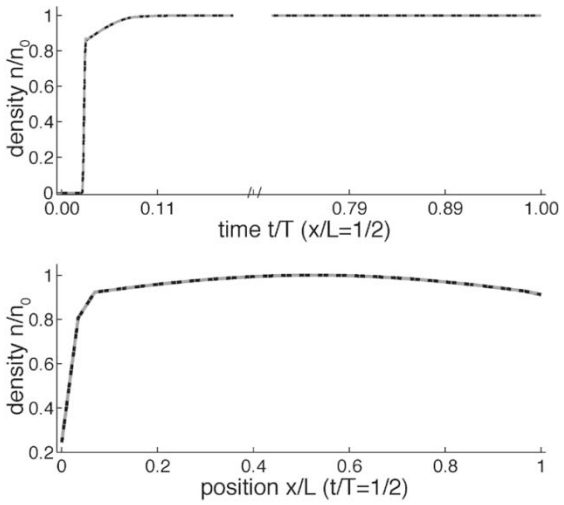

Fig. 3. Virtual cathode problem, low injection current. Both the TC and PIC results are plotted, indicating that a stable steady state is attained. (a) Density versus time at $x/L = 1/2$. (b) Density versus position across the gap at $t/T = 1/2$.

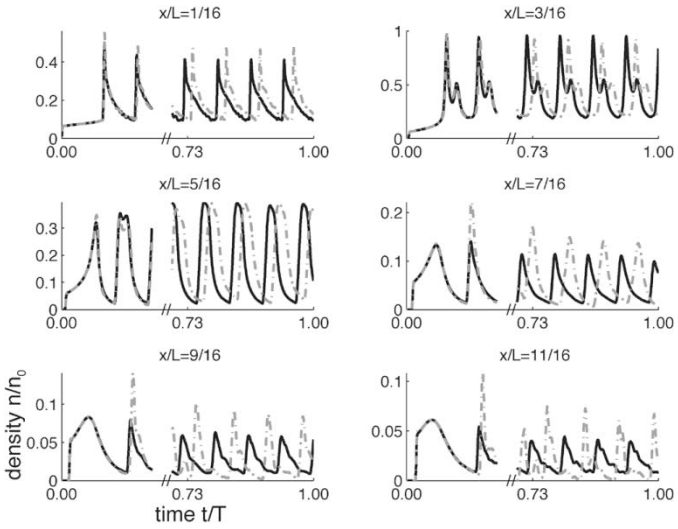

Fig. 4. Virtual cathode problem, high injection current. Particle density versus time is displayed at locations x/L from the injection point at the left end of the gap between the filaments; treecode (solid, black), and particle-in-cell (dashed, gray). Note the difference in the vertical scale at each location.

of the electron density at several locations x/L across the gap between the plates. The solid line (black) is the TC/DS and the dashed line (gray) is the PIC. The PIC simulation used the same meshsize as in the steady-state computation above. In Fig. 4, some significant differences between the TC/DS and PIC results are shown. First, the oscillation period for the PIC result is 5% greater than the TC/DS result. The phase lag in the PIC result is visible at each x/L location displayed in Fig. 4. Second, the TC/DS predicts a more uniform distribution of particles in space, while the PIC result has irregular voids. This is shown clearly in Fig. 5, which presents the particle distribution in phase space at a fixed time. This plot shows that some of the emitted particles have turned around and are striking the emitting plate. The discrepancy between TC/DS and PIC in the spatial distribution of particles is attributed to the under-resolution of local forces near the VC in the PIC simulation.

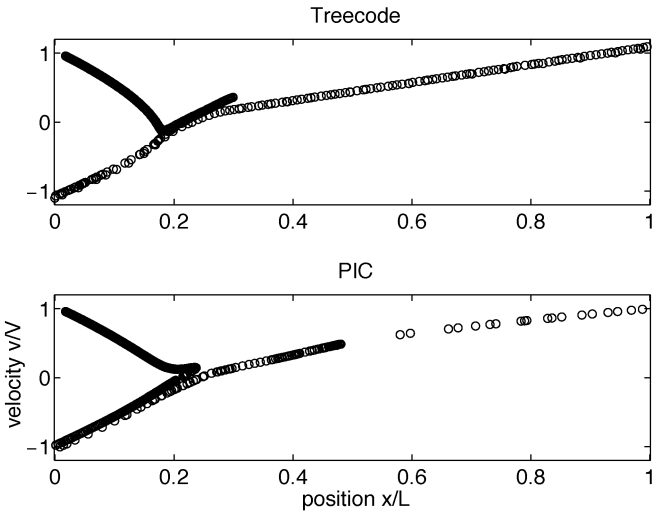

Fig. 5. Virtual cathode problem, high injection current. Particles are plotted in phase space at a fixed time.

These runs were performed on a SunBlade 100 in Matlab. The PIC simulation required 7 h, the TC 12 h, and the DS 7.5 days. Note that the resolution of the PIC results can be improved by reducing the meshsize at the cost of increasing the CPU time. We are currently implementing all three methods in C which will drastically reduce the CPU times, and we reserve final judgment on the relative performance of the methods until the new results are analyzed.

V. Conclusion

We presented a grid-free treecode algorithm for rapid computation of electrostatic forces in charged particle simulations and we compared the results with a mesh-based PIC simulation. The treecode replaces particle–particle interactions by particle–cluster interactions, and in the present approach these are approximated using a Taylor expansion of the electrostatic Green's function with respect to Cartesian coordinates. For a system of N charged particles, the treecode reduces the cost of evaluating the particle forces from $O(N^2)$ to $O(N \log N)$ without introducing spurious mesh-based effects. As an example, we applied the treecode to the virtual cathode problem in one dimension and found that while it required approximately twice the CPU time of PIC, it does a better job than PIC in resolving small-scale features of the solution. These results were obtained using Matlab, and further study using a C code is under way to validate this conclusion.

Although the present work deals with a 1-D problem, treecode algorithms are applicable in two and three dimensions as well. The treecode can be combined with boundary integral techniques to handle a variety of boundary conditions, e.g., Dirichlet, Neumann, or periodic. We are currently developing such a code to study focusing effects in ion optics and the dynamics of a magnetically confined electron column [15]. The particle–cluster treecode algorithm described here is highly parallelizable since the particle force calculations are independent from one another (once the tree is constructed) and this is also a topic for future study.

References

[1] C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation*. Bristol, U.K.: IOP, 1991.

[2] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*. Bristol, U.K.: IOP, 1988.

[3] J. L. Vay, P. Colella, P. McCorquodale, B. Van Straalen, A. Friedman, and D. P. Grote, "Mesh refinement for particle-in-cell plasma simulations: applications to and benefits for heavy ion fusion," *Laser Particle Beams*, vol. 20, pp. 569–575, 2002.

[4] J. Barnes and P. Hut, "A hierarchical $O(N \log N)$ force-calculation algorithm," *Nature*, vol. 324, pp. 446–449, 1986.

[5] G.-H. Cottet and P. D. Koumoutsakos, *Vortex Methods Theory and Practice*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[6] T. Schlick, "Molecular modeling and simulation," in *An Interdisciplinary Guide*. New York: Springer-Verlag, 2002.

[7] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *J. Comput. Phys.*, vol. 73, pp. 325–348, 1987.

[8] H. Cheng, L. Greengard, and V. Rokhlin, "A fast adaptive multipole algorithm in three dimensions," *J. Comput. Phys.*, vol. 155, pp. 468–498, 1999.

[9] S. Pfalzner and P. Gibbon, *Many-Body Tree Methods in Physics*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[10] ——, "Direct calculation of inverse-bremsstrahlung absorption in strongly coupled, nonlinearly driven laser plasmas," *Phys. Rev. E*, vol. 57, pp. 4698–4705, 1998.

[11] C. K. Birdsall and W. B. Bridges, *Electron Dynamics of Diode Regions*. New York: Academic, 1966.

[12] J. P. Verboncoeur and C. K. Birdsall, "Rapid current transition in a crossed-field diode," *Phys. Plasmas*, vol. 3, pp. 712–713, 1996.

[13] C. I. Draghicescu and M. Draghicescu, "A fast algorithm for vortex blob interactions," *J. Comput. Phys.*, vol. 116, pp. 69–78, 1995.

[14] K. Lindsay and R. Krasny, "A particle method and adaptive treecode for vortex sheet motion in three-dimensional flow," *J. Comput. Phys.*, vol. 172, pp. 879–907, 2001.

[15] A. J. Christlieb, R. Krasny, and J. P. Verboncoeur, "A treecode algorithm for simulating electron dynamics in a Penning-Malmberg trap," *Comput. Phys. Commun.*.

Andrew J. Christlieb received the B.S. degrees in mathematics, engineering mathematics, and electrical engineering from the University of Michigan, Dearborn, in 1996. He received the M.S. degree in applied mathematics and the Ph.D. degree from an interdisciplinary NSF-sponsored program, mathematics and computers in engineering, from the University of Wisconsin, Madison, in 1998 and 2001, respectively. His thesis advisor was Professor W. N. G. Hitchon in the Electrical and Computer Engineering Department, University of Wisconsin, Madison.

In the summer of 1999, he worked as a graduate student Intern under the supervision of Dr. J. Foster in the Electric Propulsion Group at the NASA John H. Glenn Research Center. While there, he worked on updating particle-in-cell codes for the simulation of ion optics and developed data acquisition systems for ion thrusters. In 2001 and 2002, he was a Postdoctoral Research Fellow with Prof. I. Boyd in the Aerospace Engineering Department, University of Michigan, Ann Arbor, where he worked on numerical simulations of micro-scale lifting bodies. In 2002, he joined the Mathematics Department, University of Michigan, Ann Arbor, as an Assistant Professor. His research interests include efficient Boltzmann solvers (including particle-based approaches), scientific computing, and modeling of dynamical systems such as plasmas and micro/nano-scale fluidics. He is currently working on developing grid-free particle methods for plasma dynamics. His applications of interest include space propulsion systems, plasma processing, and micro/nano-based technologies.

Robert Krasny received the B.S. and M.A. degrees in mathematics from the State University of New York, Stony Brook, in 1973 and 1975, respectively, and the Ph.D. degree in applied mathematics from the University of California, Berkeley, in 1985.

He is currently a Professor of mathematics at the University of Michigan, Ann Arbor. Prior to this, he held a National Science Foundation Postdoctoral Research Fellowship at the Courant Institute of Mathematical Sciences, New York, from 1984 to 1987. Aside from academic positions, he has also worked as a Scientific Programmer in the Reactor Safety Division at Brookhaven National Laboratory, Upton, NY, from 1977 to 1979. His research includes developing effective numerical algorithms and using them to investigate mathematical and physical properties of complex physical systems. In particular, he has developed Lagrangian particle methods for fluid flow and has applied these methods to study vortex sheet roll-up, formation of vortex rings, and chaotic dynamics in vortex cores. His recent work focuses on improving the capability of treecode algorithms for problems in fluid dynamics, molecular dynamics, and plasma dynamics.

John P. Verboncoeur (M'96) received the B.S. degree (with high honors) in engineering science from the University of Florida, Gainsville, in 1986 and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 1987 and 1992, respectively, holding the DOE-administered Magnetic Fusion Energy Technology Fellowship.

As a Postdoc at Berkeley and then Lawrence Livermore National Laboratory, Livermore, CA, he did pioneering work on bounded plasma models and applying object-oriented technology to computational plasma physics. He was appointed Associate Professor in Residence in 2001. He currently leads the Computational Engineering Science Program at the University of California, in its third year with about 40 students. His research interests include theoretical and computational plasma physics, broadly defined to include electromagnetics, vacuum electronics, beam optics, plasma discharges, fusion energy, and numerical methods. He is the author/coauthor of the Berkeley suite of particle-in-cell Monte Carlo collision (PIC-MCC) codes, including XPDP1 and XOOPIC. He has authored/coauthored over 30 journal articles and has taught ten international workshops and mini courses on plasma simulation. He has also worked on a number of nonacademic projects including the Strategic Air Command Executive Support System, the U.S. Postal Service Mail Forwarding System, and the TRW Credit Data Consumer Report System, in addition to developing and licensing a number of commercial hardware and software tools.