A 3D PARTICLE SIMULATION CODE FOR HEAVY ION FUSION ACCELERATOR STUDIES*

Alex Friedman, Roger O. Bangerter, Debra A. Callahan, David P. Grote, and A. Bruce Langdon
Lawrence Livermore National Laboratory
P.O. Box 808, Livermore California 94550, USA

Irving Haber
U.S. Naval Research Laboratory
Washington DC 20375, USA

Abstract

We describe WARP, a new particle-in-cell code being developed and optimized for ion beam studies in true geometry. We seek to model transport around bends, axial compression with strong focusing, multiple beamlet interaction, and other inherently 3d processes that affect emittance growth. Constraints imposed by memory and running time are severe. Thus, we employ only two 3d field arrays (ρ and ϕ), and difference ϕ directly on each particle to get **E**, rather than interpolating **E** from three meshes; use of a *single* 3d array is feasible. A new method for PIC simulation of bent beams follows the beam particles in a family of rotated laboratory frames, thus "straightening" the bends. We are also incorporating an envelope calculation, an (r, z) model, and a 1d (axial) model within WARP. The BASIS development and run-time system is used, providing a powerful interactive environment in which the user has access to all variables in the code database.

Introduction

In this paper we briefly describe the elements of the WARP code, concentrating on the 3d particle-in-cell model WARP6; the name derives from the 6d phase space, and from our goal of modeling a bent ("warped") beam. A more detailed description [1] is available. Recent code developments, in particular the ability to load matched beams with tapered ends, are described here. Our initial runs have included tests of infinite (periodic) beams in an alternating-gradient lattice; in preliminary tests, we have noted a rapid conversion of transverse thermal energy into longitudinal thermal energy, when the beam is initially colder in z than in x and y. We have also made runs which model the drift-compression of a tapered-end beam in the presence of misaligned quadrupole focusing elements. A run of each type is briefly presented.

Overall Code Architecture

The code is written in FORTRAN using facilities of the BASIS system [2]. The latter provides a run-time shell which affords a flexible interpreter-driven interface. The user can view, perform transformations on, and/or plot any quantity in the code. Graphics, I/O, etc. are handled by BASIS, and any FORTRAN subroutine can be called from the interpreter directly. BASIS also provides a development system which facilitates truly modular "physics packages" and enables us to think of WARP as either a "code" or a "family of codes."

Extensive phase space plot, contour plot, and particle moment diagnostics are available. Many of these are based on a system of user-specified "windows," which are ranges in z, x, y, and r. Time-history data is collected and plotted at the end of the run. Graphical output is written to a metafile, and (optionally) to the user's terminal. A frame index is generated as an aid to selecting plots for viewing or printing.

3d Package WARP6

The simulation takes place in the laboratory frame; the mesh is a moving "window." The particle advance is (optionally) relativistic; for each particle we store spatial coordinates x, y, z, normalized momenta γv_x, γv_y, γv_z, and (for efficiency) γ^{-1}. The self-field is assumed electrostatic in the beam frame; at present we use this field directly. For faster beams we will obtain the lab-frame self-**E** and **B** via a Lorentz transformation. Electric and magnetic forces are applied using a conventional algorithm [3, Chapter 15]. The code gains efficiency via a number of means; these are described in the subsections below.

Residence Corrections

In leapfrog motion, if a particle were to land within a sharp-edged element on four steps while its neighbor did so on only three, they would receive very different impulses. Thus, the advance is modified to incorporate "residence corrections" (averages over a step) for element forces; these are multiplicative factors equal to the fraction of the velocity advance step actually spent within the element. We estimate z at the beginning and end of the velocity advance step by assuming v_z remains constant. The element force is averaged over a half-step at startup. The procedure allows much bigger steps than otherwise would be possible.

Because the residence correction is (weakly) z-velocity dependent, the symplectic nature of the leapfrog advance is broken. For the systems in which we are interested, we believe that any negative consequences of this are outweighed by the vastly improved accuracy. A similar force-averaging over a step should also be applicable to problems involving fringe fields, etc.

Isochronous Advance

Timestep number n is conceptually defined as the advance of all quantities from time level $n-1$ to time level n. The simplest way to effect this is by means of an "isochronous leapfrog" particle advance [4]. In this scheme, the usual leapfrog velocity advance step is split so that both **x** and **v** move from one integer time level to the next at each step. The numerical properties are identical to that of leapfrog. The scheme simplifies particle loading and diagnostics, and allows us to change Δt while preserving second-order accuracy. However, for speed we use the isochronous advance only on "special" steps on which diagnostics (etc.) are done, and revert to (almost) pure leapfrog for all others. Thus we need make only a single pass through the particles on most steps.

At the end of "special" steps, a second pass is made through the particle list; in this pass, **v** is advanced to the current integer time level n, so that phase-space plots, restart dumps, etc. will use synchronized **x** and **v**. At startup, and at the beginning of any step following a special step, **v** is only advanced a half-step. The code must give the same answers whether or not one stops to synchronize. Thus, to advance **v** to level n we use a full advance to $n+1/2$, followed by an easily invertible half-advance back. The "residence correction" for the latter uses the mean beam v_z.

Fieldsolver

The 3d fieldsolver is performed "in-place" using fast Fourier transforms (FFT's). It performs sine transforms (for metal wall boundary conditions) in x and y, and real periodic transforms in z (so far) [3, Appendix A]. Along all axes vectorization takes place over the "second" dimension, leaving the "third" for possible multi-tasking. The solver uses essentially no scratch space. At present the grid dimensions N_x, N_y, and N_z are constrained to be powers of two. A typical current application uses a $64 \times 64 \times 256$ mesh (just over a million zones). Such a system is solved in about two seconds on a Cray X/MP.

Local Differencing of ϕ

No arrays for **E** are used; instead, ϕ is gathered from cells in the neighborhood of each particle, and then differenced on a particle-by-particle basis. This saves the space of three 3d arrays, and for the trilinear interpolation applied to **E** is quite efficient. In 3d, one needs to pick up 32 ϕ's instead of 24 E's. It is a simple task to collect the ϕ's, since one is collecting like

objects using an index list. The procedure takes advantage of the hardware vector gather capability of our Cray X/MP. In the future, we may consider use of a *single* 3d array (for ρ and ϕ); for efficiency and flexibility we have been using two. It would be necessary to make two passes through the particles (even on a leapfrog step); one can't deposit ρ into the array currently holding E.

Other Aspects of WARP6

The particle advance is vectorized. Deposition of ρ is vectorized with length 8, over cells touched by each particle. As a future refinement, we plan to deposit ρ from eight well-separated particles at once, to get vector lengths of 64. This requires a partial sorting of particles; those depositing simultaneously must not access the same cells. The scheme generalizes those of [5].

"Quiet-start" particle loading [3, and ref. therein] is employed to reduce fluctuations and minimize the number of particles needed. One option generates a transverse "semi-Gaussian" distribution which has a nearly uniform spatial density within the envelope and pseudo-random cut-off Gaussian distributions of z and x, y velocities. Another option generates a "Kapchinskij-Vladimirskij" (K-V) distribution [6] in transverse phase space. WARP's envelope solver is run before particles are loaded, so that the necessary parameters at each z are available.

We have recently added tapered ends to our beams, with a parabolic dependence of line-charge density upon z [7]. For convenience, we assume that ϵ_x and ϵ_y vary as a^2 (and thus as I). This leads to cigar-shaped beams with tune depression independent of z, so that a single envelope solution (suitably scaled) describes the whole beam. We are currently introducing an axial confining force into our 3d and (r, z) simulations, in an effort to simulate "quiescent" finite beams. For "cigar" beams, a linearly ramped electric field does not cancel the self-field very well, and waves are launched. We thus plan to take into account the increase in the "g-factor" toward the ends of the beam, which leads to a faster-than-linear ramp-up.

Other Models

A "cylindrical" (r, z) model, such as the one now in WARP, captures the interaction between axial and some transverse motions. The model is significantly faster than the 3d one, since it takes fewer particles to represent a 5d phase space; furthermore, the assumption of "continuous focusing" (also useful in 3d) eliminates the cost of tracking through quadrupoles. A much finer mesh is also possible. This package will be used for longitudinal stability studies.

We are incorporating a longitudinal 1d model. It will not employ the usual long-wavelength $\partial\lambda/\partial z$ force law, but will instead assume incompressibility to infer the beam radius $r(z)$ from $\lambda(z)$, use the existing (r, z) field solver to obtain $E_z(r, z)$, then average over the beam cross-section to obtain $E_z(z)$. The model will thus be directly comparable with more complete (r, z) simulations, as well as with other longitudinal codes.

Technique for Modeling a Bent Particle Beam

Here, we outline the method we are developing; a somewhat more detailed description appears in [8], along with an outline of a related algorithm for a 2d (transverse) model of a bent beam. We define the coordinate s to be the usual axial coordinate z in a straight section, but a distance along the "centerline" $x = 0$ in a bend (we assume that $y = 0$ is nominally a symmetry plane). We also define r_* to be the local radius of curvature of the centerline; thus s is an angle ψ scaled by r_*. In the bends, x is a radial coordinate: $x = r - r_*$. The axial velocity is $v_z = r\dot{\psi} = -r\dot{\theta}$. Note that $ds/dt = r_*\dot{\psi} \neq v_z$.

Particles are advanced as is usually done in a straight-geometry code; then a transformation is made into a rotated inertial Cartesian coordinate system (different for each particle) in which the x coordinate is aligned along the line between the vessel's nominal center of curvature and the particle position. The frame of reference is never accelerating, so no pseudo forces (centrifugal or Coriolis) need be applied, and the properties of the underlying advance are retained. The algorithmic steps are:

(1) Enter with x_0, $z_0 \equiv s_0$, $\dot{x}_0 \equiv v_{x0}$, $\dot{z}_0 \equiv v_{z0}$. s_0 is the accumulated $z_{\text{straights}} + r_*\psi_{\text{bends}}$. Advance these to x_1, z_1, $\dot{x}_1$, $\dot{z}_1$ via leapfrog. In a straight, stop: $s_1 = z_1$, $v_{x1} = \dot{x}_1$, $v_{z1} = \dot{z}_1$. In a bend, continue.

(2) $r_1 = \sqrt{(r_* + x_1)^2 + (z_1 - z_0)^2}$.

(3) $\cos\psi = \dfrac{r_* + x_1}{r_1}$, $\sin\psi = \dfrac{z_1 - z_0}{r_1}$, $\psi = \arctan\dfrac{z_1 - z_0}{r_* + x_1}$.

(4) Obtain the new position: $x_2 = r_1 - r_*$, $s_2 = s_0 + r_*\psi$.

(5) Rotate $\mathbf{v}$ through the same angle: $v_{x2} = \cos\psi\dot{x}_1 + \sin\psi\dot{z}_1$, $v_{z2} = -\sin\psi\dot{x}_1 + \cos\psi\dot{z}_1$. This completes the step.

Special care is needed to preserve the exactness of the transformations. Residence corrections are needed when a step overlaps the entrance to or exit from a bend.

"Equilibration" Process

This run examines the exchange of thermal energy between transverse and longitudinal motions in an infinite periodic beam. In contrast with some earlier work [9], our beam was initially cold in z. In the run shown, the phase advance per cell, $\sigma_0 = 60°$, was depressed by space charge to $\sigma = 20°$. The simulation used a $64 \times 64 \times 128$ mesh, with walls at ± 5 cm. 108320 particles were loaded into a quiet-start K-V distribution with initial semi-axes 1.88 and 3.05 cm. The lattice period and mesh length were 1.2 m, and Δt corresponded to 2 cm/step. The problem was run for 9000 steps (180 m, or 150 periods). It used 2.64 Mwords of memory and took somewhat less than 6 X/MP hours.

In this run, the z thermal energy rises rapidly at first while the transverse energy falls. When $v_{th,z}$ has risen to about half $v_{th,x}$, slow heating takes over in both z and x, y. The transverse emittance falls during the rapid heating phase, then grows very slowly. The total energy (less that of the nominal beam speed) is conserved to one part in 150. The various energy "components" are shown as functions of time in Figure 1.

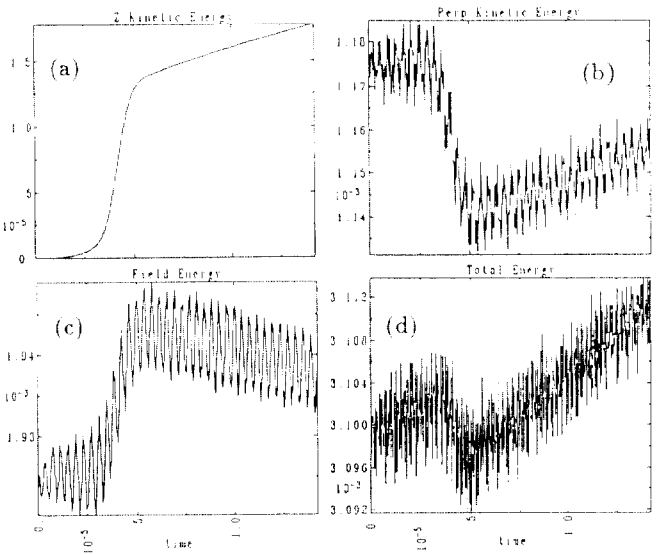

Figure 1. Energies versus time for infinite periodic run: (a) longitudinal kinetic (omits nominal); (b) transverse kinetic (includes AG energy); (c) field; (d) total.

We emphasize the preliminary nature of these results; nonetheless, factor-of-two variations in cell size, particle number, and timestep size did not qualitatively alter the behavior, which also occurs under continuous focusing. We have not observed a correspondingly rapid exchange when the beam was initially warmer in z than in x, y. We conjecture that the initial rapid heating in z may be the result of an anisotropy-driven instability reminiscent of a Harris mode [10], but with betatron motion instead of cyclotron motion. The final state, with $v_{th,x} \sim 2v_{th,z}$, may arise from numerical heating primarily of transverse motions, with collisional transfer into longitudinal motions.

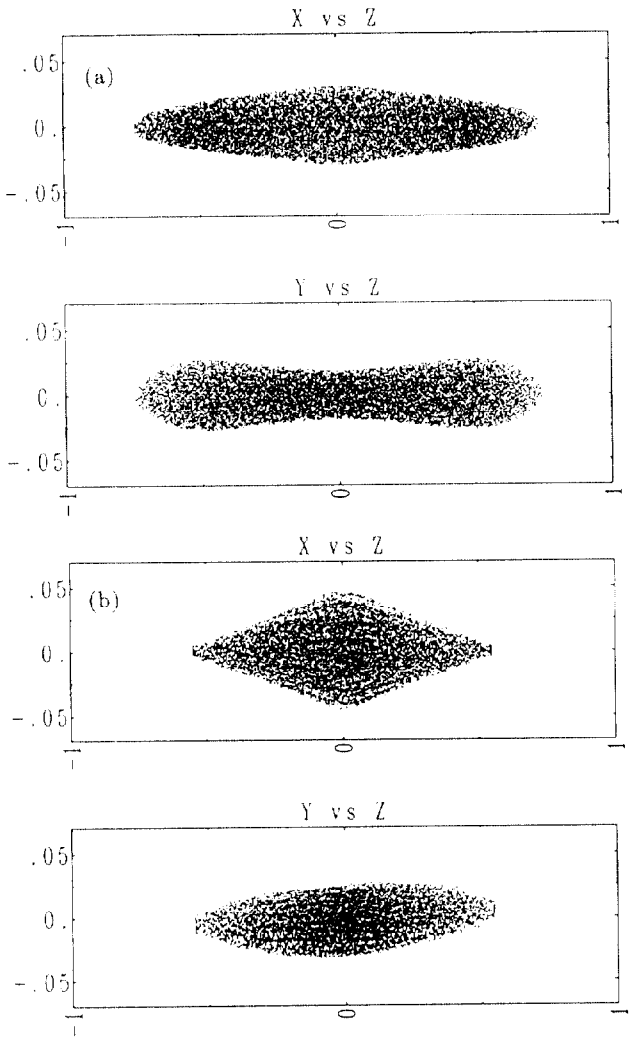

Figure 2. Drift compression run: (a) initial; (b) at step 600.

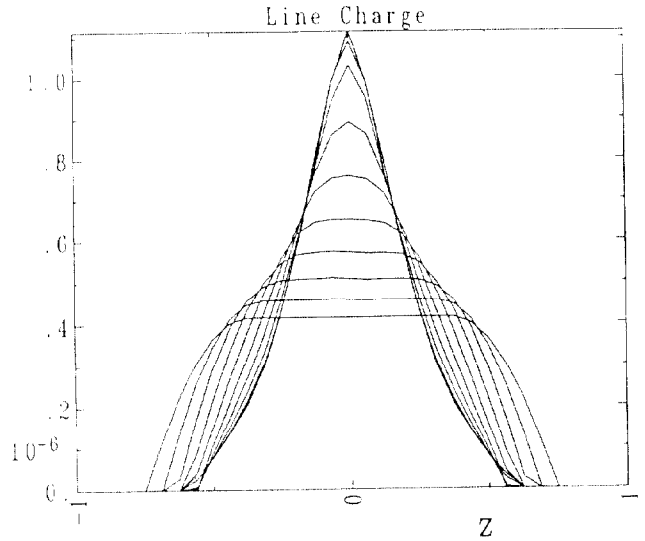

Figure 3. Drift compression run: line charge density versus z at ten times during run.

Drift Compression through Misaligned Quadrupoles

In the final stages of beam transport it is useful to concentrate the current by introducing a "velocity tilt." We have simulated a beam wherein the initial $(v_{\text{tail}} - v_{\text{head}})/v_{\text{center}} = 7.5\%$. The run used 54160 particles in a K-V distribution, and a $64 \times 64 \times 128$ mesh 2 m long with walls at ± 7 cm. Again, $\sigma_0 = 60°$, $\sigma = 20°$, and the lattice period was 1.2 m. Magnetic quadrupoles were misaligned in x and y with RMS deviations 0.5 mm. The beam was run for 900 steps, taking 37 minutes of X/MP time. The particle mover used 7.2 μs/particle/step, the field solver $\sim$1 s/step.

Figure 2 shows "top" and "side" views of the beam at $t = 0$, and after 600 steps. The effect of the drift compression is evident. As a chance result of the particular pseudo-random numbers used, the beam suffered worse cumulative deflections in y than in x. In Figure 3 the line charge density at a number of times during the run is shown overlaid (only a small number of points in z were plotted). Total energy was conserved to one part in 500. The variation in total energy was about 1% of the fall in kinetic energy over the course of the run. The beam-center emittance remained approximately constant in a similar run without misalignments; in a run with 1 mm RMS errors, it grew by $\sim$25%, as a result of the beam's closer approach to the square pipe walls. This growth disappears when the walls are moved out to 9 cm.

Acknowledgments

The authors have benefited from discussions with S. Brandon, P. Dubois, G. Joyce, J. Krall, J. Mark, and D. Nielsen.

*This work was performed under the auspices of the U.S. D.O.E. by Lawrence Livermore National Laboratory under contract W-7405-ENG-48, and by the Naval Research Laboratory under Lawrence Berkeley Laboratory contract DE-AC03-76SF0098.

References

1. A. Friedman, D. A. Callahan, D. P. Grote, A. B. Langdon, and I. Haber, "WARP: A 3D (+) PIC Code for HIF Simulations," to appear in *Proc. of the Conf. on Computer Codes and the Linear Accelerator Community*, Los Alamos NM, January 21–25, 1990, R. K. Cooper and K. C. D. Chan, eds. (to be published); LLNL Report UCRL-102907 (1990).
2. P. F. Dubois, *et.al.*, "The Basis System," Lawrence Livermore National Laboratory Document M-225 (1988).
3. C. K. Birdsall and A. B. Langdon, "Plasma Physics via Computer Simulation," McGraw-Hill, New York, 1985.
4. A. Friedman and S. P. Auerbach, "Numerically Induced Stochasticity;" S. P. Auerbach and A. Friedman, "Long-time Behavior of Numerically Computed Orbits: Small and Intermediate Timestep Analysis," to appear in *J. Comput. Phys.*
5. A. Héron and J. C. Adam, "Particle Code Optimization on Vector Computers," *J. Comput. Phys.* **85**, 284 (1989); E. Horowitz, "Vectorizing the Interpolation Routines of Particle-in-Cell Codes," *J. Comput. Phys.* **68**, 56 (1987).
6. I. M. Kapchinskij and V. V. Vladimirskij, Proc. 2nd Intl. Conf. on High Energy Accelerators, p. 274, CERN (1959).
7. D. Neuffer, "Longitudinal Motion in High Current Ion Beams–A Self-consistent Phase Space Distribution with an Envelope Equation," *IEEE Trans. Nuclear Science* **NS-26**, 3031 (1979).
8. A. Friedman, "Methods for PIC Simulation of Bent Particle Beams in 3d and 2d", Proc. 13th Conf. on Numerical Simulation of Plasma, R. J. Mason, ed., paper PMB-10, Santa Fe, 1989 (unpublished).
9. I. Hofmann and I. Boszik, "Computer Simulation of Longitudinal-Transverse Space-Charge Effects in Bunched Beams," Proc. 1981 Linear Acclerator Conf., Los Alamos National Laboratory Report LA-9234-C, 116 (1982).
10. E. G. Harris, "Unstable Plasma Oscillations in a Magnetic Field," *Phys. Rev. Let.* **2**, 34 (1959).