

The WARP Code: Modeling High Intensity Ion Beams

David P. Grote, Alex Friedman

LLNL, Livermore, CA, USA

Jean-Luc Vay

LBNL, Berkeley, CA, USA

Irving Haber

University of Maryland, College Park, MD, USA

Abstract. The Warp code, developed for heavy-ion driven inertial fusion energy studies, is used to model high intensity ion (and electron) beams. Significant capability has been incorporated in Warp, allowing nearly all sections of an accelerator to be modeled, beginning with the source. Warp has as its core an explicit, three-dimensional, particle-in-cell model. Alongside this is a rich set of tools for describing the applied fields of the accelerator lattice, and embedded conducting surfaces (which are captured at sub-grid resolution). Also incorporated are models with reduced dimensionality: an axisymmetric model and a transverse “slice” model. The code takes advantage of modern programming techniques, including object orientation, parallelism, and scripting (via Python). It is at the forefront in the use of the computational technique of adaptive mesh refinement, which has been particularly successful in the area of diode and injector modeling, both steady-state and time-dependent. In the presentation, some of the major aspects of Warp will be overviewed, especially those that could be useful in modeling ECR sources. Warp has been benchmarked against both theory and experiment. Recent results will be presented showing good agreement of Warp with experimental results from the STS500 injector test stand. Additional information can be found on the web page http://hif.lbl.gov/theory/WARP_summary.html.

INTRODUCTION

The Warp code was originally developed to model the high current, high brightness beams that are required heavy-ion driven inertial confinement fusion (HIF).[1,2] HIF offers a path to fusion as an energy source. It relies on having ion beams focused down onto the small fusion target, driving it to ignition. In order to provide the required energy, the ion beams must be high current, but have low enough emittance (or temperature) to be focusable. These beams are “space-charge dominated” – the self-field effects are significantly larger than the thermal effects. The beams act as non-neutral plasmas. An ideal method to simulate these beams is the particle-in-cell (PIC)

method from plasma physics. This method fills the phase-space with representative particles and couples them by solving Maxwell’s equation on a grid.

The Warp code begins with the PIC method and extends it by incorporating a description of the applied fields of the accelerator lattice. The PIC method is implemented in axisymmetric mode, transverse slice mode, and in full 3-D mode. Due to the relatively low energy per nucleon of the beams in HIF, only an electrostatic, Poisson, solver has been implemented. The solver allows internal boundary conditions – extensive tools have been developed for their specification. The particle advance is 2nd order leap-frog, and for the coupling to the grid, linear, or cloud-

in-cell interpolation is done. Multi-species can be modeled, such as multiple charge states and multiple ions. For electrons, an advanced integrator is being developed that allows large time-steps compared to the electron cyclotron frequency.[3] The lattice description allows a range of field descriptions, from uniform, pure multipole components, to axially varying, mixed components, to gridded field data.

The natural mode of operation of the PIC method (and thus of Warp) is to be time-dependent, which is well suited for the modeling of space-charge dominated beams. A consequence is that the fields from the lattice are applied directly to the particles, rather than via mapping methods, as is usual in the modeling of emittance dominated beams.

WARP OVERVIEW

Combined in Warp are many different pieces that cover a wide variety of scenarios, covering various dimensionality, levels of problem description, and kinds of physics. All pieces of the code have been adapted to run in parallel-processing environments.

Warp3D

The original package of Warp was the 3-D package, which models the beam in full three-dimensional physical space and three-dimensional velocity space. The self-fields are calculated on a Cartesian mesh laid down in the frame of the beam. The mesh can move with the beam or remain static. In a bend, warped-Cartesian coordinates are used, which are cylindrical coordinates, with the angle theta replacing the axial coordinate z . A single mesh can contain areas with and without bends. In a bend, the coordinate system follows a defined physical centerline of the bend, which does not necessarily coincide with the trajectories of any particles (which depend only on applied and self fields). The coordinates of the particles, however, are stored relative to the warped coordinates – a particle which does follow the bend centerline will have $x = 0$.

A number of field solvers (Poisson solvers) are available. The first is an FFT based solver. Bends can be included by moving the curvature related terms to the right hand side, treating them as sources, and iterating to convergence. Simple internal boundaries can be included using the capacity matrix method. For larger, more complicated conductors however, the matrix becomes very large and is costly to generate.

For this reason, an iterative multigrid solver was developed. This solver can include arbitrary internal boundaries. At the internal boundaries, cut-cell or embedded boundary conditions are used to maintain second order convergence of the solver. Extensive tools have been developed to specify the conductors, allowing combinations of basic geometric objects, such as cylinders and tori, and more complicated objects, such as those describable as surfaces of revolution. In bends, the curvature terms are directly included in the iteration. An adaptive mesh refinement (AMR) capability in three-dimensions is in development. Two and four-fold transverse symmetries can be taken advantage of for efficiency.

The basic time advance for the Warp3D is fully time-dependent. Various approximations can be used to gain efficiencies, however. For example, “quasi time-dependence” can be used – the particle advance is fully time-dependent, but the self-field calculation is done only periodically. A further approximation is an iterative steady-state mode, where a single bunch of particles is tracked through the system, accumulating the charge density. The self-fields are recalculated with the accumulated density and the iteration repeated. This is a standard method in many gun codes. It can sometimes converge to a bi-stable state, however. The quasi time-dependent does not suffer from this problem.

WarpRZ

Beams can be modeled assuming axisymmetry – variation along the azimuth is ignored. Warp actually follows the particles in full 3-D space, but the charge density is mapped to and the self-fields mapped from the r - z plane. The Poisson solver uses the multigrid method, and includes internal boundaries using the same cut-cell methods as in the 3-D solver. The adaptive mesh refinement methods in the RZ solver are more developed.[4]

WarpXY

The third model implemented is a transverse slice model which effectively models a steady flow. A thin transverse slice of the beam is followed through the lattice, ignoring any z -dependence of the self-fields. Each time step, the particles are advanced to the same z position - they all have the same z -step size. The particles can have a variation in their axial velocity, and z -dependent and z -directed applied fields are included. Each particle has its own time-step size, which is adjusted inversely to the axial velocity to

keep the z step size constant. Each step, as the axial velocity changes, the advance is iterated to update the time-step size of each particle. In bends, the slice moves in steps of the angle θ around the bend - particles at large x in the bend move further each step (the time-step size is adjusted accordingly). There are two Poisson solvers implemented, an FFT based solver with optional capacity matrices, and a multigrid/AMR based solver.

The Lattice

The lattice description is used to set the applied fields and geometry of bends. The fields can be specified at several levels of description. Any elements can be overlapped. The lowest level is the axially uniform, hard edge approximation. Any multipole component can be applied, for example solenoid, dipole, quadrupole, sextapole, *etc.* Accelerating fields can be applied as well. When the fields are applied to the particles, “residence corrections” are used, where, upon entering or exiting the element, the applied field is scaled by the fraction of the time-step spent inside the element. With the corrections, 2nd order accuracy is maintained in the advance.

The next level of description is to use axially varying multipole components. The z variation of the coefficients of the components is tabulated. Currently, linear interpolation is done between data points. Any component or combination of components can be applied, including both fundamentals and their axial derivatives. In a bend, the transverse center follows the curvature.

A further detailed description is to use fully tabulated field data. For magnetic fields, the three components of the field are each specified in three-dimensional Cartesian grids. For electric fields, the potential is specified on a single three-dimensional grid, and finite differences are done on a per particle basis. In both cases, tri-linear interpolation is done between grid points. As with the other descriptions, in a bend, the grid follows the curvature.

For electrostatic elements, the fields can be applied by directly including the conductors as boundary conditions in the self-field calculation. For example, with interdigitated electric quadrupoles, the geometry as described via the lattice can be included. In a diode, the voltage drop can be modeled by including the anode and cathode plates.

The bend elements are different than the others since they are only specifying geometry – no fields are

applied. For any elements that overlap a bend, the center of the element follows the curvature. Currently, Warp only supports bends in one plane, the z - x plane. Also, note that two bends can overlap each other. Figure 1 shows an example lattice that includes bends.

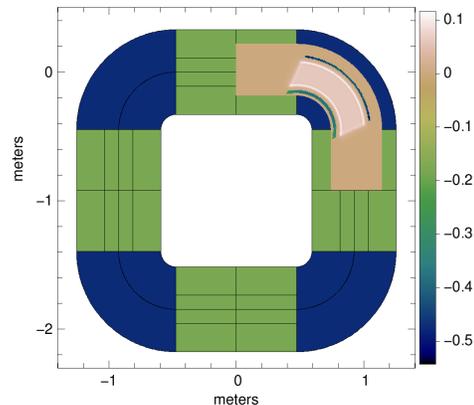


Figure 1: This shows an example lattice – a storage ring experiment at MSU containing only 4 bends. The magnetic field is gridded. The blue shows the extent of the bend. The green shows where the gridded field is (though it is covered over in the bend). The color scale is the B_y field component, in Tesla.

Injection

A significant capability of Warp is its ability to model particle injection. Fixed-current, space-charge limited injection, and secondary emission can be modeled. Injection from a plasma source, using the standard approximation of electrons with the Boltzmann distribution, is in development.[5] The emission of particles can be from curved surfaces. Some examples are shown in Figures 2, 3 and 4.

Unlike many gun codes that launch particles from a virtual surface in front of the true emission surface, Warp launches particles directly from the true emission surface. This offers several advantages: for sources immersed in a magnetic field, particles are advanced correctly in that field from birth; for time-dependent problems, particles can spend a significant time traversing the virtual region in front of the source, and in order to correctly model the head of the beam, the detailed motion in that region must be captured. As part of this, a capability was added to model this region using one-dimensional mesh refinement along lines normal to the emitting surface. The refinement is non-linear, following the Child-Langmuir density scaling. Refinement factors as high as 10,000 are used regularly [4].

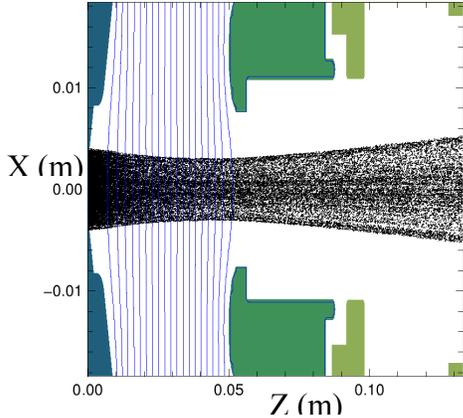


Figure 2: The extraction region of the VENUS ECR source. The nearly vertical lines are evenly spaced contours of constant potential.

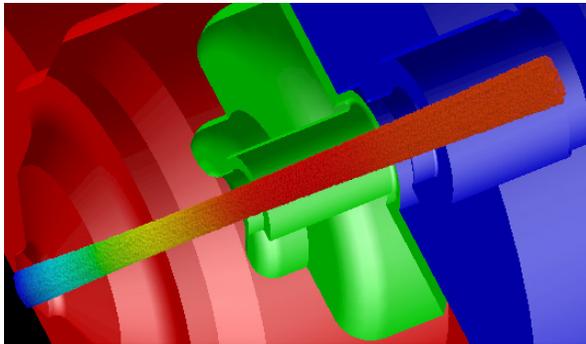


Figure 3: The same region as shown in figure 2, but rendered in 3-d.

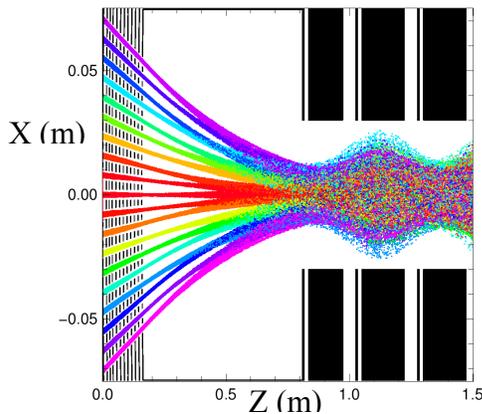


Figure 4: From a multiple beamlet merging injector. This shows the slice at $y=0$. Here, 119 beamlets are independently injected and accelerated and then merged into a single beam that flows into a transport channel.

The modeling of secondary emission of particles is under development. With this capability, a simulation can for example include emission of electrons when an

ion or electron strikes a surface. The motion of these particles are tracked self-consistently.

Python interface

The user interface to Warp is the modern scripting language Python.[6] This is a fully object oriented language that is well developed and is used extensively throughout the world. While the core of the code is written in modern Fortran, Python is the interface for data input, steering, and post-processing. Python gives the user great control over the problem description and how the simulation is carried out. The authors of Warp do not have to foresee all possible modes of operation, diagnostics, post-processing, *etc.* that the users may need. The users input file becomes the “main” routine. Python also gives interactive access, allowing such things as rapid problem setup and debugging, and interactive experimentation to help in aiding the understanding of the problem of interest.

CONCLUSIONS

Warp was originally developed to study the high-current, high-brightness beams required for the HIF approach to fusion energy. It was designed to be flexible, including various degrees of approximation and dimensionality. Warp should work well for ECR ion sources. Complex conductor geometries can be modeled and bends included. Multiple species can be injected and followed. A plasma source model is in development. This covers much of the capability required for ECR source modeling.

REFERENCES

1. A. Friedman, D. P. Grote, I. Haber, “Three Dimensional Particle Simulation of Heavy Ion Fusion Beams,” *Phys. Fluids B*, **4**, 2203 (1992)
2. D. P. Grote, A. Friedman, G. Craig, I. Haber, W. M. Sharp, “Progress Toward Source-to-Target Simulations,” *Nucl. Instrum. Methods Phys. Res. A*, 464, p. 563 (2001).
3. R. Cohen, *et. al.*, “Electron-Cloud Simulation and Theory for High-Current Heavy-Ion Beams,” to be published in *Phys. Rev. ST Accel. and Beams*.
4. J. L. Vay, *et. al.*, “Application of adaptive mesh refinement to particle-in-cell simulations of plasmas and beams,” *Phys. Plasma* **11**, 2928 (2004)
5. http://www.science.doe.gov/sbir/awards_abstracts/sbirstr/cycle22/phase1/024.htm
6. <http://www.python.org>