Introduction to PIC Codes and Astra

Chad Mitchell,
Lawrence Berkeley National Laboratory

References

Steve Lund's notes on computation
(and intense beams in general with J. Barnard)
http://hifweb.lbl.gov/USPAS_2011/

The ASTRA code+documentation from Klaus Floettmann:
http://www.desy.de/~mpyflo/

"Computer Simulation Using Particles"
R W Hockney, J W Eastwood

Robert Ryne, "Computational Methods in Accelerator Physics"
http://uspas.fnal.gov/materials/09UNM/ComputationalMethods.pdf

Contents

- Why simulations?
- Commonly-used simulation tools
- Particle-in-cell algorithm overview
- Numerical artifacts and convergence
- Introduction to ASTRA
  1) generating the initial beam
  2) the main ASTRA input file
  3) diagnostic output

Why simulations?

“The purpose of computation is insight, not numbers.”
Richard Hamming

Simulations quantify the expected performance of proposed machines to produce high confidence that systems will work as intended/promised to funding agencies.

Simulations build intuition about machine performance before experiments begin.

- Offer full, non-intrusive diagnostics at nearly any location in the machine
- Effects can be turned on/off to isolate

Simulations allow analysis of more realistic situations than analytically tractable.

- Detailed and complex geometries
- Non-ideal beam distributions
- Nonlinear effects

Simulations are cheaper and faster than running the experiment, and can exploit dramatic ongoing improvements in computational hardware and software.

The highest understanding and confidence is achieved when results from analytical theory, numerical simulation, and experiment all converge.

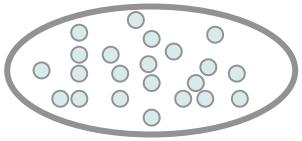
Commonly-Used Simulation Tools

A number of codes have been developed over the years that are heavily used in the design and optimization of high-brightness electron injectors. These codes must be capable of simulating electron beams in the presence of intense nonlinear space charge forces, which play a critical role at low energy.

An incomplete list of 'popular' codes for injector simulation include (alphabetical order):

- **ASTRA**: free downloadable at http://www.desy.de/~mpyflo/
- **GPT**: commercial (http://www.pulsar.nl/gpt/)
- **HOMDYN:** free, contact Massimo Ferrario (Massimo.Ferrario@lnf.infn.it)
- **IMPACT-T**: free, contact Ji Qiang (jqiang@lbl.gov)
- **PARMELA**: free, export limitations apply. http://laacg1.lanl.gov/laacg/services/

All of the above are *particle* codes, with the exception of HOMDYN, which is an *envelope* code. Particle codes represent the dynamics of the physical beam by tracking the positions and momenta of a large number N_p of simulation particles or *macroparticles*.



Particle-In-Cell (PIC) Algorithm Overview

Field Solution

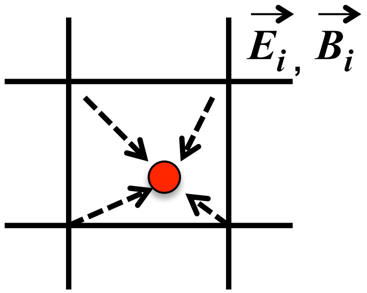
Solve the field equations on the grid.
Maxwell's equations – *electromagnetic PIC*
Poisson equation – *electrostatic PIC*
Poisson equation in the beam rest frame
- *quasistatic PIC*

$$\nabla^2 \phi = -\rho/\epsilon_0$$
$$\vec{E}_b = -\nabla \phi$$

lab
frame

Field Interpolation

Interpolate both external and self-fields to the particle positions.



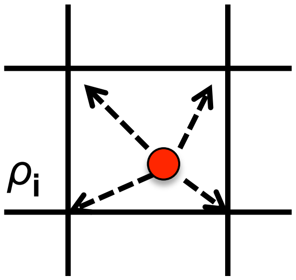
Particle Push

Integrate the equations of motion one time step to update particle positions and momenta.

$$\vec{F} = q\left(\vec{E} + \vec{v} \times \vec{B}\right)$$
$$\frac{d\vec{r}}{dt} = \vec{v}, \qquad \frac{d\vec{p}}{dt} = \vec{F}$$

Charge Deposition

Use simulation particles to deposit charge and/or current onto a spatial grid.



Repeat for each
time-step

Numerical Artifacts and Convergence

Three numerical parameters are critical for any PIC simulation:

1) **time step:** Δt
2) **grid cell size:** $\mathbf{H}$
3) **number of simulation particles:** N_p

Even if the external fields throughout the machine are precisely known, the use of discretized time Δt and space $\mathbf{H}$ will limit the resolution of physics that can be captured accurately by the simulation, as well as introducing unphysical effects.

Typically $N_p << N_e$, where N_e is the physical number of electrons in the beam. This artificially increases the fluctuations of self-fields experienced by the particles. In injector simulations, these effects primarily result in additional growth of the beam emittances and energy spread not present in the physical electron beam.

It is therefore important to check the convergence of the simulation output with respect to the three parameters above in the limit as:

$$\Delta t \longrightarrow 0 \,, \qquad \mathbf{H} \longrightarrow 0 \,, \qquad N_p \longrightarrow N_e \,.$$

Introduction to ASTRA

A Space charge TRacking Algorithm

1) Developed at DESY for photoinjector simulations
2) Serial and parallel implementations (will focus on serial)
3) A collection of different programs/utilities
    1) `generator` – Generates the initial distribution from the distribution input file
    2) `astra` – Propagates the particles according to the astra input file
    3) `fieldplot` – Plots the external and space charge fields
    4) `postpro` – Calculates and plots properties of the final distribution
    5) `lineplot` – Plots the evolution of beam properties along the beamline

Both 2-D (cylindrical coordinates R-Z for axisymmetric systems) and 3-D field solvers are available. We will be using the 2-D solver only.

Introduction to ASTRA

Calculation of external fields

- For RF fields – input the longitudinal on-axis electric field $E_z(z,r=0)$
- For solenoids – input the longitudinal on-axis magnetic field $B_z(z,r=0)$
- Fields off-axis are determined from Maxwell's equations using Taylor series about the axis.

Calculation of space charge fields (2-D)

- Use a cylindrical grid with **Nlong_n** longitudinal and **Nrad** transverse cells.
- Transform to the beam rest frame, assume constant charge density within each cell and integrate to get the SC force.

Particle pusher

- Using the internal and external fields, the individual particles are propagated using a Runge-Kutta 4th order integrator.
- The minimum and maximum time steps for the integrator are user defined, and should be checked for convergence.

Generating the Initial Beam

Use “generator” on a distribution input file with suffix “--.in”

Example input file for generator

```
&INPUT
  FNAME = 'Example.ini'
  Add=FALSE,      N_add=0,
  IPart=500,      Species='electrons'
  Probe=True,     Noise_reduc=T,          Cathode=F
  Q_total=1.0E0

  Ref_zpos=0.0E0, Ref_Ekin=2.0E0

  Dist_z='gauss',       sig_z=1.0E0,      C_sig_z=2.0
  Dist_pz='g',          sig_Ekin=1.5,     cor_Ekin=0.0E0

  Dist_x='gauss',       sig_x=0.75E0,
  Dist_px='g',          Nemit_x=1.0E0,    cor_px=0.0E0
  Dist_y='g',           sig_y=0.75E0,
  Dist_py='g',          Nemit_y=1.0E0,    cor_py=0.0E0
 /
```

Name of particle output file

Bunch charge (nC)

Number of particles

Distribution type

Format of generated particle bunch

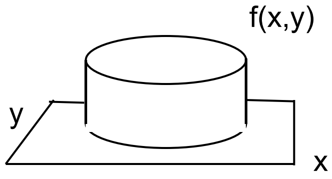
Number	1	2	3	4	5	6	7	8	9	10
Parameter	x	y	z	px	py	pz	clock	macro charge	particle index	status flag
Unit	m	m	m	eV/c	eV/c	eV/c	ns	nC		

Astra assumes a planar cathode and includes image charge forces (when the cylindrical 2-D algorithm is used).

Common Types of Initial Beam Distributions

Radial uniform (transverse)

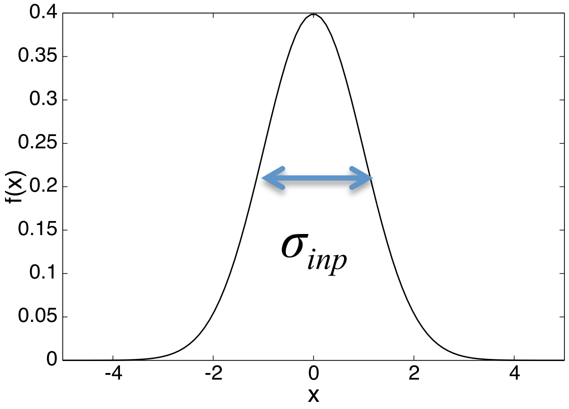
$$f(x,y) = \frac{1}{\pi r^2} \qquad \text{for } x^2 + y^2 \leq r^2$$



Gaussian (transverse or longitudinal)

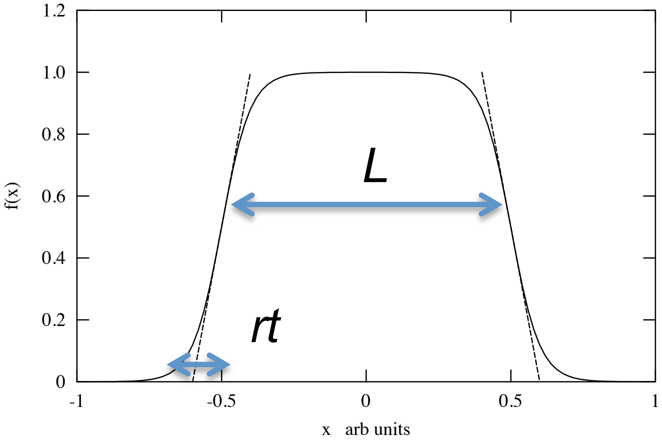
$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_{inp}} \exp\left(-\frac{1}{2}\frac{x^2}{\sigma_{inp}^2}\right)$$

commonly truncated for |x| > Cσ_{inp} for some C>0



Plateau (longitudinal)

$$f(x) = \frac{1}{L} \cdot \frac{1}{1 + \exp\left(\frac{2}{rt}(2|x| - L)\right)} \qquad rt \leq \frac{L}{2}$$



Structure of the Main Input File

This is a file with suffix "--.in"

Initial Beam

```
&NEWRUN
 Head=' Example of ASTRA users manual'
 RUN=1
 Distribution = 'Example.ini',      Xoff=0.0,   Yoff=0.0,
 TRACK_ALL=T,      Auto_phase=T

 H_max=0.001,      H_min=0.00
```

Diagnostics

```
&OUTPUT
 ZSTART=0.0,       ZSTOP=1.5
 Zemit=500,        Zphase=1
 RefS=T
 EmitS=T,      PhaseS=T
/
```

Distance from cathode at which to end simulation (m)

Field Solver

Space charge is off. Set LSPCH=T to include space charge forces!

```
&CHARGE
 LSPCH=F
 Nrad=10, Cell_var=2.0, Nlong_in=10
 min_grid=0.0
 Max_Scale=0.05
/
```

Number of radial and longitudinal grid points

Layout of elements

```
&CAVITY
 LEField=T,
 File_Efield(1)='3_cell_L-Band.dat',  C_pos(1)=0.3
 Nue(1)=1.3,       MaxE(1)=40.0,     Phi(1)=0.0,
/

&SOLENOID
 LBField=T,
 File_Bfield(1)='Solenoid.dat', S_pos(1)=1,2
 MaxB(1)=0.35, S_smooth(1)=10
/
```

This value is added to the z-coordinate of points in the on-axis field input file. If the input file data is symmetric around zero, then this value gives the distance of the element's midpoint from the cathode.

Diagnostic Output

Output particle distribution: Name.zpos.001

Column format is the same as that used for the initial particle distribution.

Number	1	2	3	4	5	6	7	8	9	10
Parameter	x	y	z	px	py	pz	clock	macro charge	particle index	status flag
Unit	m	m	m	eV/c	eV/c	eV/c	ns	nC		

Output rms moments: Name.Xemit.001, Name.Yemit.001, Name.Zemit.001

Contain the evolution of the bunch centroid, rms size, and emittance along the beamline.

$$\langle x \rangle = \frac{1}{N}\sum_{j=1}^{N} x_j \qquad x' = p_x/p_z$$

Note: the emittance calculation uses canonical rather than mechanical momentum.

$$\epsilon_{nx}^2 = \frac{1}{m^2c^2}\left(\langle x^2\rangle\langle p_x^2\rangle - \langle xp_x\rangle^2\right)$$

Columns

Xemit	z m	t ns	x_{avr} mm	x_{rms} mm	x'_{rms} mrad	$\varepsilon_{x,norm}$ π mrad mm	$x\cdot x'_{avr}$ mrad
Yemit	z m	t ns	y_{avr} mm	y_{rms} mm	y'_{rms} mrad	$\varepsilon_{y,norm}$ π mrad mm	$y\cdot y'_{avr}$ mrad
Zemit	z m	t ns	E_{kin} Mev	z_{rms} mm	ΔE_{rms} kev	$\varepsilon_{z,norm}$ π keV mm	$z\cdot E'_{avr}$ keV