Intro. Lecture 04: Numerical Methods for Particle and Distribution Methods: Introduction to the PIC Method*

Prof. Steven M. Lund
Physics and Astronomy Department
Facility for Rare Isotope Beams (FRIB)
Michigan State University (MSU)

US Particle Accelerator School (USPAS) Lectures On
"Self-Consistent Simulations of Beam and Plasma Systems"
Steven M. Lund, Jean-Luc Vay, Remi Lehe, and Daniel Winklehner

US Particle Accelerator School Summer Session
Colorado State U, Ft. Collins, CO, 13-17 June, 2016
(Version 20160614)

* Research supported by:
FRIB/MSU, 2014 On via: U.S. Department of Energy Office of Science Cooperative Agreement DE-SC0000661and National Science Foundation Grant No. PHY-1102511
and
LLNL/LBNL, Pre 2014 via: US Dept. of Energy Contract Nos. DE-AC52-07NA27344 and DE-AC02-05CH11231

Detailed Outline
Introductory Lectures on Self-Consistent Simulations

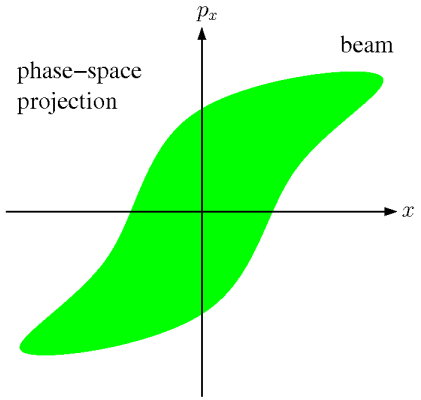
Numerical Methods for Particle and Distribution Methods:
Introduction to the Particle in Cell (PIC) Method

A. Overview
B. Integration of Equations of Motion
- Leapfrog Advance for Electric Forces
- Leapfrog Advance for Electric and Magnetic Forces
- Numerical Errors and Stability of the Leapfrog Method
- Illustrative Examples
C. Field Solution
- Electrostatic Overview
- Green's Function Approach
- Gridded Solution: Poisson Equation and Boundary Conditions
- Methods of Gridded Field Solution
- Spectral Methods and the FFT
D. Weighting: Depositing Particles on the Field Mesh and Interpolating Gridded Fields to Particles
- Overview of Approaches
- Approaches: Nearest Grid Point, Cloud in Cell, Area, Splines
E. Computational Cycle for Particle in Cell Simulations

Numerical Solution of Particle and Distribution Methods
A: Overview

Particle Methods – Generally not used at high space-charge intensity
Distribution Methods – Preferred (especially PIC) for high space-charge.
We will motivate why now.

Why are direct particle methods are not a good choice for typical beams?



N particle coordinates

$\{\mathbf{x}_i, \mathbf{p}_i\}$

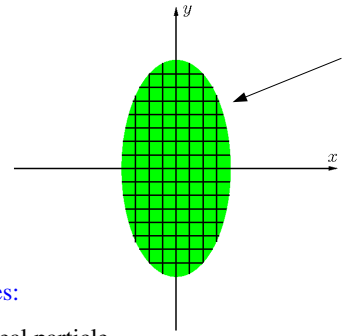
Physical beam (typical)
$N \sim 10^{10} - 10^{14}$ particles

Although larger problems are possible every year with more powerful computers, current processor speeds and memory limit us to $N \lesssim 10^8$ particles

- Fast multipole and other advanced methods may show promise to circumvent issue

Numerical Solution of Particle and Distribution Methods (2)

Represent the beam evolving in the Vlasvo model by Lagrangian "macroparticles" advanced in time



Partition local density into macroparticles

Macroparticle Properties:

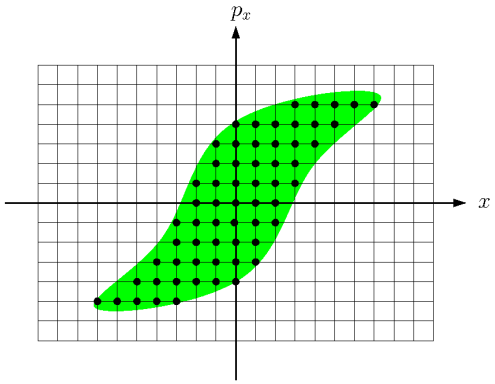
- Same q/m ratio as real particle
  - Gives same single particle dynamics in the applied field
- More collisions due to macroparticles having more close approaches
  - Enhanced collisionality is unphysical
  - Controlled by smoothing the macroparticle interaction with the self-field. More on this later. Must check that results are converged.

Numerical Solution of Particle and Distribution Methods (3)

Direct Vlasov as an example:

Discretize grid points $\{\boldsymbol{x}_i, \boldsymbol{p}_i\}$

Advance distribution $f(\boldsymbol{x},\boldsymbol{p},t)$ at discrete grid points in time



- Continuum distribution advanced on a discrete phase-space mesh
  - Extreme memory for high resolution. Example: for 4D x-p_x, y-p_y with 100 mesh points on each axis -> $100^4 = 10^8$ values to store in fast memory (RAM)
- Discretization errors can lead to aliasing and unphysical behavior (negative probability, etc.)

Numerical Solution of Particle and Distribution Methods (4)

Both particle and distribution methods can be broken up into two basic parts:

0) Moving particles or distribution evaluated at grid points through a finite time (or axial space) step

1) Calculation of beam self-fields consistently with the distribution of particles

In both methods, significant fractions of run time may *be devoted to diagnostics*

- Moment calculations can be computationally intensive and may be "gathered" frequently for evolution "histories"
- Phase space projections ("snapshot" in time)
- Fields (snapshot in time)

Diagnostics are also critical!

- Without appropriate diagnostics runs are useless, even if correct
- Must accumulate and analyze/present large amounts of data in an understandable format
  - Trends often as important as numbers

Significant code development time may also be devoted to creating (loading) the initial distribution of particles to simulate

- Loading will usually only take a small fraction of total run time
- Can simulate particles born off of source too – but sources often have very difficult physics issues also

B: Integration of Equations of Motion

Higher order methods require more storage and numerical work per time step

- Fieldsolves are expensive, especially in 3D, and several fieldsolves per step can be necessary for higher order accuracy

Therefore, low-order methods are typically used for self-consistent space-charge. The "leapfrog" method is most common

- Only need to store prior position and velocity
- One fieldsolve per time step

Illustrate the leapfrog method for non-relativistic particle equations of motion:

- Develop methods for particles but can be applied to moments, distributions,...

$$m\frac{d\mathbf{v}}{dt} = \mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$
$$\frac{d\mathbf{x}}{dt} = \mathbf{v}$$

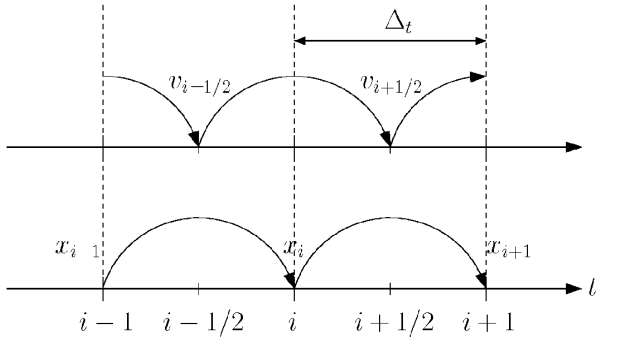
Leapfrog Method for Electric Forces

Leapfrog Method: for *velocity independent* (Electric) forces

Leapfrog Advance (time centered): Advance velocity and position out of phase

$$1) \quad m\frac{\mathbf{v}_{i+1/2} - \mathbf{v}_{i-1/2}}{\Delta_t} = \mathbf{F}_i \qquad \mathbf{F} = \mathbf{F}(\mathbf{x})$$

$$2) \quad \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\Delta_t} = \mathbf{v}_{i+1/2}$$



Leapfrog Method: Order

To analyze the properties of the leapfrog method it is convenient to write the map in an alternative form:

$$\frac{\mathbf{x}_{i+1}-\mathbf{x}_i}{\Delta_t} = \mathbf{v}_{i+1/2}$$

$$i \to i-1 \qquad \frac{\mathbf{x}_i-\mathbf{x}_{i-1}}{\Delta_t} = \mathbf{v}_{i-1/2}$$

Subtract the two equations above and apply the other leapfrog advance formula:

$$\Longrightarrow \quad m\frac{\mathbf{v}_{i+1/2}-\mathbf{v}_{i-1/2}}{\Delta_t} = \boxed{m\frac{\mathbf{x}_{i+1}-2\mathbf{x}_i+\mathbf{x}_{i-1}}{\Delta_t^2} = \mathbf{F}_i}$$

Note correspondence of formula to discretized derivative:

$$\left.\frac{\partial^2 f}{\partial t^2}\right|_i = \frac{f_{i+1}-2f_i+f_{i-1}}{\Delta_t^2} + \mathcal{O}(\Delta_t^2)$$

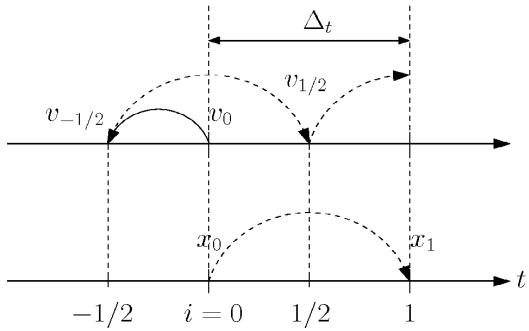
- $\mathbf{x}_{i+1}$ fixed from $\mathbf{x}_i$, $\mathbf{x}_{i-1}$, $\mathbf{F}_i$ to $\mathcal{O}(\Delta_t^4)$
- Leapfrog method is 2nd order accurate

Initial conditions must be desynchronized in leapfrog method

Leapfrog Method: Synchronization

Since **x** and **v** are not evaluated at the same time in the leapfrog method synchronization is necessary both to start the advance cycle and for diagnostics

- Initial conditions: typically, **v** is pushed back half a cycle



- When evaluating diagnostic quantities such as moments the particle coordinates and velocities should first be synchronized analogously to above

Leapfrog Method for Magnetic and Electric Forces -- The Boris Method

Velocity Dependent Forces

Another complication in the evolution ensues when the force has velocity dependence, as occurs with magnetic forces. This complication results because ***x*** and ***v*** are advanced out of phase in the leapfrog method

$$\mathbf{F} = q\mathbf{E} + q\mathbf{v}\times\mathbf{B}$$

velocity term

- Electric field **E** accelerates
- Magnetic field **B** bends particle trajectory without change in speed |**v**|

A commonly implemented time centered scheme for magnetic forces is the following 3-step "Boris" method:

J. Boris, in *Proceedings of the 4th Conference on the Numerical Simulation of Plasmas* (Naval Research Lab, Washington DC 1970)

The Boris Advance

Boris Advance: The coordinate advance is the same as previous leapfrog and the velocity advance is modified as a 3 step procedure:

1) Half-step acceleration in electric field

$$\mathbf{v}_i^{(1)} = \mathbf{v}_{i-1/2} + \frac{q}{m}\mathbf{E}_i\frac{\Delta_t}{2}$$

2) Full step rotation in magnetic field. Here choose coordinates so that $\mathbf{B}_i$ is along the *z*-axis and and resolve $\mathbf{v}_i^{(1)}$ into components parallel/perpendicular to *z*

$$\mathbf{B}_i = B_i\hat{z} \quad \omega_{ci} = \frac{qB_i}{m} \quad B_i = |\mathbf{B}_i|$$

$$\| \mathbf{B}_i \ : \quad v_{z,i}^{(2)} = v_{z,i}^{(1)}$$

$$\perp \mathbf{B}_i \ : \quad \begin{bmatrix} v_{x,i}^{(2)} \\ v_{y,i}^{(2)} \end{bmatrix} = \begin{bmatrix} \cos(\omega_{ci}\Delta_t) & \sin(\omega_{ci}\Delta_t) \\ -\sin(\omega_{ci}\Delta_t) & \cos(\omega_{ci}\Delta_t) \end{bmatrix} \begin{bmatrix} v_{x,i}^{(1)} \\ v_{y,i}^{(1)} \end{bmatrix}$$

3) Half-step acceleration in electric field

$$\mathbf{v}_{i+1/2} = \mathbf{v}_{i+1/2}^{(3)} = \mathbf{v}_i^{(2)} + \frac{q}{m}\mathbf{E}_i\frac{\Delta_t}{2}$$

Boris Advance Continued (2)

Complication: on startup, how does one generate the out-of-phase **x**, **v** advance from the initial conditions?

- Calculate **E**, **B** with initial conditions
- Move **v** backward half a time step
  - Rotate with **B** a half-step
  - Decelerate a half-step in **E**

Similar comments hold for synchronization of **x**, **v** for diagnostic accumulation

Now we will look at the numerical properties of the leapfrog advance cycle

- Only use a simple "electric" force example to illustrate issues

Leapfrog Advance: Errors and Numerical Stability

To better understand the leapfrog method consider the simple harmonic oscillator:

$$\frac{d^2x}{dt^2} = -\omega^2 x, \quad \omega = \text{const} \implies \begin{aligned} x &= C_0 \cos\omega t + C_1 \sin\omega t \\ &= x_0 \cos(\omega t + \psi_0) \end{aligned}$$

Discretized equation of motion

$$\frac{x_{i+1} - 2x_i + x_{i-1}}{\Delta_t^2} = -\omega^2 x_i$$

Exact solution

$x_0 = \text{const}$ (amplitude)

$\psi_0 = \text{const}$ (phase)

Try a solution of the form $x_i = ce^{j\tilde{\omega} i \Delta_t}$, $j \equiv \sqrt{-1}$, $c = \text{const}$ (complex), $\tilde{\omega} = \text{const}$

$$\Rightarrow e^{j\tilde{\omega}\Delta_t} - 2 + e^{-j\tilde{\omega}\Delta_t} = -\omega^2\Delta_t^2$$

$$2 - 2\cos(\tilde{\omega}\Delta_t) = \omega^2\Delta_t^2$$

$$\sin^2\left(\frac{\tilde{\omega}\Delta_t}{2}\right) = \frac{\omega^2\Delta_t^2}{4}$$

$$\Rightarrow \sin\left(\frac{\tilde{\omega}\Delta_t}{2}\right) = \frac{\omega\Delta_t}{2}$$

This has solutions for $\omega\Delta_t < 2$ and it is straightforward to show via expansion that for small $\omega\Delta_t$

$$\omega\Delta_t = \tilde{\omega}\Delta_t + \frac{(\tilde{\omega}\Delta_t)^3}{24} + \mathcal{O}\left[(\tilde{\omega}\Delta_t)^5\right]$$

Leapfrog Errors and Numerical Stability Continued (2)

It follows for the leapfrog method applied to a simple harmonic oscillator:

- For $\omega\Delta_t < 2$ the method is stable
- There is *no* amplitude error in the integration
- For $\omega\Delta_t \ll 1$ the phase error is
  - Actual phase: $\psi \equiv \omega\Delta_t i$
  - Simulated phase: $\tilde{\psi} \equiv \tilde{\omega}\Delta_t i \approx \omega\Delta_t i + \frac{(\omega\Delta_t)^3}{24} i$
  - Error phase: $\Delta\psi \equiv \tilde{\psi} - \psi \approx \frac{(\omega\Delta_t)^3}{24} i$

Note: i to get to a fixed time $\sim \Delta_t^{-1}$ and therefore phase errors decrease as $\mathcal{O}(\Delta_t^2)$

// Example: $\omega = 2\pi/\tau$

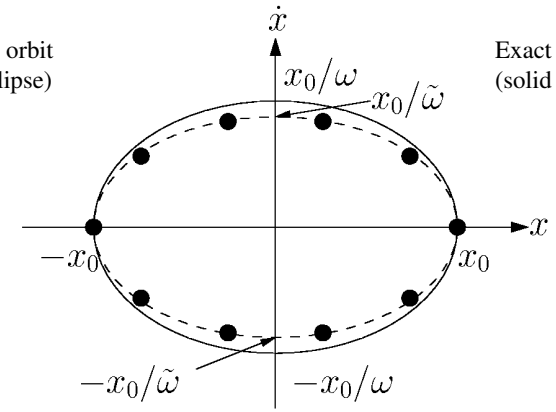
Time step	Steps for a π phase error	
$\Delta_t = 0.1\tau$	$24\pi/(0.1 \cdot 2\pi)^3$	$\approx 3 \times 10^2$
$\Delta_t = 0.01\tau$	$24\pi/(0.01 \cdot 2\pi)^3$	$\approx 3 \times 10^5$ //

Leapfrog Errors and Numerical Stability Continued (3)

Contrast: Numerical and Actual Orbit: Simple Harmonic Oscillator

Numerical orbit (dashed ellipse)

Exact orbit (solid ellipse)



Emittance = (Phase Space Area)/π

Exact: $\varepsilon = x_0^2/\omega$

Numerical: $\tilde{\varepsilon} = x_0^2/\tilde{\omega}$

$$\frac{\tilde{\varepsilon}}{\varepsilon} \approx 1 - \frac{(\omega\Delta_t)^2}{24}$$

Leapfrog Errors and Numerical Stability Continued (4)

The numerical orbit conserves phase space area regardless of the number of steps taken! The slight differences between the numerical and actual orbits can be removed by rescaling the angular frequency to account for the discrete step

- More general analysis of the leapfrog method shows it has "symplectic" structure, meaning it preserves the Hamiltonian nature of the dynamics
- Symplectic methods are important for long tracking problems (typical in accelerators) to obtain the right orbit structure
  - Runge-Kutta methods are not symplectic and can result in artificial numerical damping in long tracking problems

Example: Contrast of Non-Symplectic and Symplectic Advances

Contrast: Numerical and Actual Orbit for a Simple Harmonic Oscillator

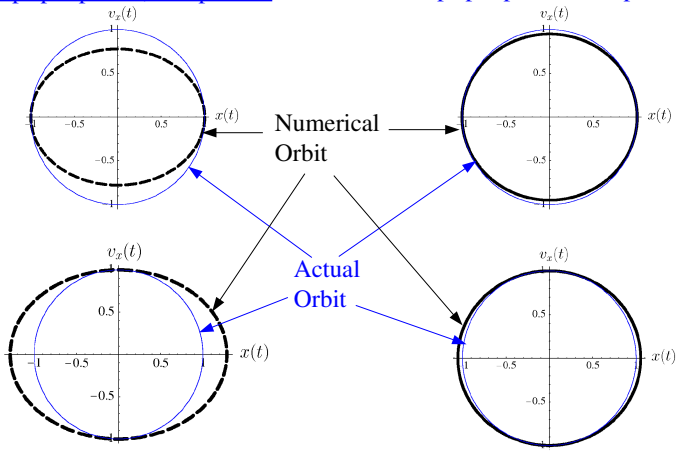
use scaled coordinates (max extents unity for analytical solution)

Symplectic Leapfrog Advance:

5 steps per period, 100 periods | 10 steps per period, 100 periods

Cosine-type initial conditions

Sine-type initial conditions



Example: Contrast of Non-Symplectic and Symplectic Advances (2)

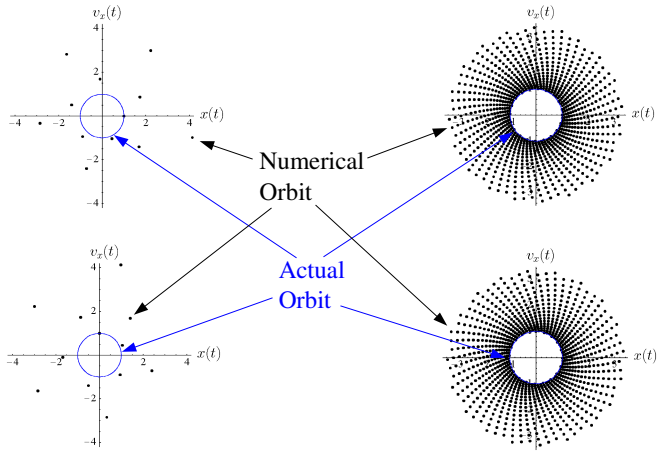
Contrast: Numerical and Actual Orbit for a Simple Harmonic Oscillator

Non-Symplectic 2nd Order Runge-Kutta Advance: (see earlier notes on RK advance)

6 steps per period, 10 periods | 20 steps per period, 50 periods

Cosine-type initial conditions

Sine-type initial conditions



Example: Contrast of Non-Symplectic and Symplectic Advances (3)

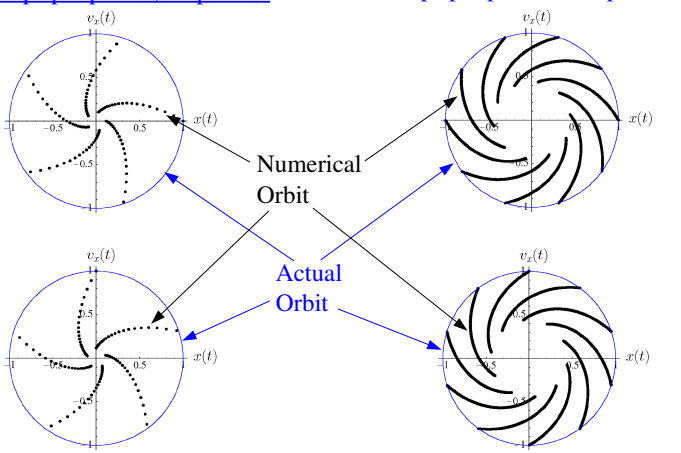
Contrast: Numerical and Actual Orbit for a Simple Harmonic Oscillator

Non-Symplectic 4th Order Runge-Kutta Advance: (analog to notes on 2nd order RK adv)

5 steps per period, 20 periods | 10 steps per period, 200 periods

Cosine-type initial conditions

Sine-type initial conditions



Example: Leapfrog Stability Applied to the Nonlinear Envelope Equation in a Continuous Focusing Lattice

For linear equations of motion, numerical stability requires:

$$k\Delta_s < 2$$

Here, *k* is the wave number of the phase advance of the quantity evolving under the linear force. The continuous focusing envelope equation is nonlinear:

$$\frac{d^2 r_x}{ds^2} + k_{\beta 0}^2 r_x - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_x^2}{r_x^3} = 0$$

$$\frac{d^2 r_y}{ds^2} + k_{\beta 0}^2 r_y - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_y^2}{r_y^3} = 0$$

$k_{\beta 0}$ = focus wave-number = const

Q = dimensionless perveance = const

$\varepsilon_{x,y}$ = rms edge emittances = const

Several wavenumbers *k* expected to be expressed in the envelope evolution:

$k_\beta = \sigma / L_p$ .... Depressed Particle Betatron Motion

$k_{\beta 0} = \sigma_0 / L_p$ .... Undepressed Particle Betatron Motion

$k_Q \equiv \sqrt{k_{\beta 0}^2 + 3k_\beta^2}$ .... Quadrupole Envelope Mode

$k_B \equiv \sqrt{2k_{\beta 0}^2 + 2k_\beta^2}$ .... Breathing Envelope Mode

Example: Leapfrog Stability and the Continuous Foc. Envelope Equation (2)

Expect that $k_{\beta 0}\Delta_s < 2$ for the fastest (largest *k*) component determines stability.

Numerical simulations for an initially matched envelope with: $\sigma/\sigma_0 = 1/2$

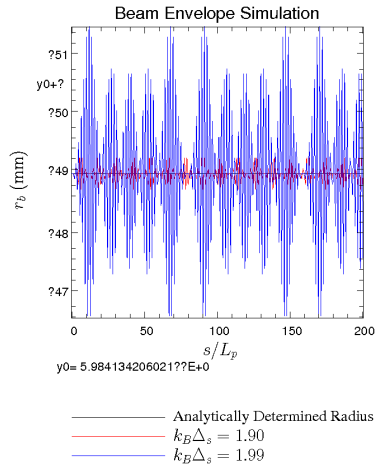
$k_\beta\Delta_s$	$k_{\beta 0}\Delta_s$	$k_Q\Delta_s$	$k_B\Delta_s$	Stable?
0.500	1.00	1.32	1.58	Yes
0.600	1.20	1.59	1.90	Yes
0.630	1.26	1.67	1.99	Yes
0.635	1.27	1.68	2.01	No
0.640	1.28	1.69	2.02	No

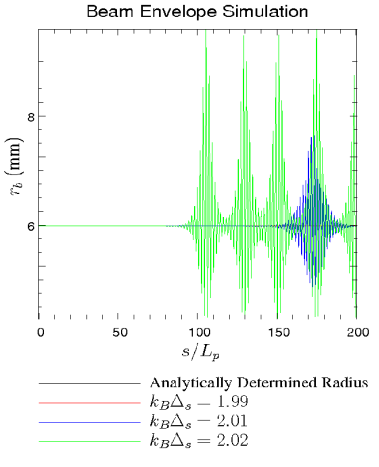
The highest *k*-mode, the breathing mode determines stability, i.e. $k_B\Delta_s < 2$ is the stability criterion. Other values of σ/σ_0 produce results in agreement with this conclusion.

Example: Leapfrog Stability and the Continuous Foc. Envelope Equation (3)

Numerical simulations an initially matched envelope with: $\sigma_0 = 80^\circ,\ \sigma/\sigma_0 = 1/2$

Note that numerical errors seed small amplitude mismatch and that the plot scale to the left is ~ 10^{-13}, corresponding to numerical errors.





Comments of 2D and 3D Axisymmetric Particle Moves

To be added:

Comments on moving ring particles:
- 3D axisymmetry => particles rings, 3D axisymmetry => particles are infinite cylindrical shells.
- Angular momentum will be conserved for such particles (can rotate)
- Easier to do in many cases using x-y movers

C: Field Solution

The self-consistent calculation of beam-produced self-fields is vital to accurately simulate forces acting on particles in intense beams

$$\mathbf{F} = q\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right)$$

Linear structure of the Maxwell equations allow fields to be resolved into externally applied and self (beam generated) components

$$\mathbf{E} = \mathbf{E}_a + \mathbf{E}_s$$
$$\mathbf{B} = \mathbf{B}_a + \mathbf{B}_s$$

$\mathbf{E}_a,\ \mathbf{B}_a$ applied fields generated by magnets and electrodes

- Can be calculated at high resolution in external codes and imported or specified via analytic formulas
- Sometimes calculated from code fieldsolve via applied charges and currents and boundary conditions

$\mathbf{E}_s,\ \mathbf{B}_s$ self fields generated by beam charges and currents

- At high beam intensities can be a large fraction (on average) of applied fields
- Can be important to calculate with realistic boundary conditions

Electrostatic Field Solution

For simplicity, we restrict analysis to electrostatic problems to illustrate methods:

$\mathbf{B} = \mathbf{B}_a$ $\mathbf{B}_a$ specified via another code or theory

$\mathbf{E} = \mathbf{E}_a + \mathbf{E}_s$ $\mathbf{E}_a$ due to biased electrodes and $\mathbf{E}_s$ due to beam space-charge

The Maxwell equations to be solved for $\mathbf{E}$ are

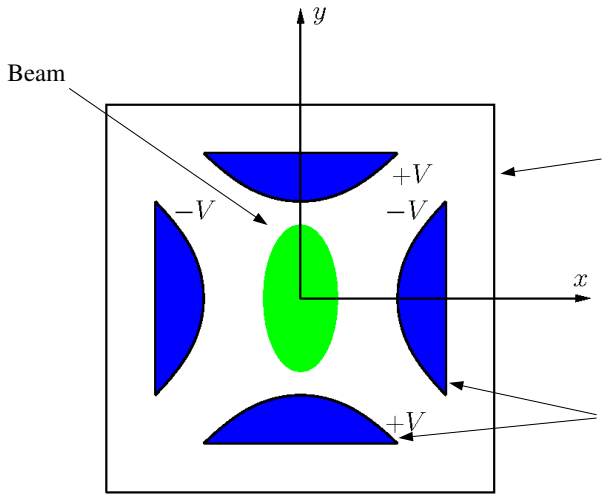
$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \quad + \text{ boundary conditions on } \mathbf{E}$$
$$\nabla \times \mathbf{E} = 0$$

$\nabla \times \mathbf{E} = 0$ implies that we can always take $\mathbf{E} = -\nabla\phi$ and so

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0} \quad + \text{ boundary conditions on } \phi$$
$$\mathbf{E} = -\nabla\phi$$

Electrostatic Field Solution: Typical Problem

As an example, it might be necessary to solve (2D) fields of a beam within an electric quadrupole assembly.



ϕ or $\dfrac{\partial\phi}{\partial n}$ specified on domain boundary or consistently to model assembly in free space

Quadrupole electrodes held at ±V

Electrostatic Field Solution by Green's Function

Formally, the solution to ϕ can be constructed with a Green's function, illustrated here with Dirichlet boundary conditions:

$$\nabla^2 G(\mathbf{x}, \mathbf{x}') = -4\pi\delta(\mathbf{x} - \mathbf{x}')$$
$$G(\mathbf{x}, \mathbf{x}')|_{\mathbf{x}'_b} = 0$$
$$\mathbf{x}'_b \equiv \mathbf{x}' \text{ on boundaries}$$

Definitions:

$$\frac{\partial}{\partial n} \equiv \hat{\mathbf{n}} \cdot \frac{\partial}{\partial \mathbf{x}}$$

$\hat{\mathbf{n}} \equiv$ Unit normal vector to boundary surface

This yields (Jackson, *Classical Electrodynamics*)

$$\phi(\mathbf{x}) = \frac{1}{4\pi\epsilon_0}\int_V d^3x'\ \rho(\mathbf{x}')G(\mathbf{x}, \mathbf{x}') - \frac{1}{4\pi}\oint_S d^2x'\ \phi(\mathbf{x}')\frac{\partial G(\mathbf{x}, \mathbf{x}')}{\partial n'}$$

Self-field component $\equiv \phi_s$ $\quad$ Applied field from electrode potentials $\equiv \phi_a$

$$\mathbf{E}_s = -\frac{\partial\phi_s}{\partial \mathbf{x}} \qquad \mathbf{E}_a = -\frac{\partial\phi_a}{\partial \mathbf{x}}$$

Electrostatic Field Solution by Green's Function (2)

Applied Field ϕ_a:

Can be calculated on mesh in advance and need not be recalculated if transverse geometry does not change

- Can be analytical in simple situations

Self Field ϕ_s :

Let: q_M, $\mathbf{x}_i$ = Macro-particle charge and coordinate

N_p = Macro-particle number

$$\phi_s = \frac{1}{4\pi\epsilon_0}\int d^3x'\,\rho(\mathbf{x}')G(\mathbf{x},\mathbf{x}') = \frac{q_M}{4\pi\epsilon_0}\sum_{i=1}^{N_p}\int d^3x'\,\delta(\mathbf{x}'-\mathbf{x}_i)G(\mathbf{x},\mathbf{x}')$$

$$\phi_s = \frac{q_M}{4\pi\epsilon_0}\sum_{i=1}^{N_p} G(\mathbf{x},\mathbf{x}_i)$$

Comment:
We evaluate at macropararticle coordinate with no shape factor for simplicity.

Then the field at the ith macro-particle is (self-field term removed):

$$\mathbf{E}_{si} = \left.\frac{\partial\phi_s}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_i} = \frac{q_M}{4\pi\epsilon_0}\sum_{\substack{j=1\\ j\neq i}}^{N_p}\left.\frac{\partial G(\mathbf{x},\mathbf{x}_j)}{\partial\mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_i}$$

Electrostatic Field Solution by Green's Function (3)

The Green's Function expression for ϕ_s will, in general, be a numerically intensive expression to evaluate at *each* macroparticle

- $N_p(N_p - 1)$ terms to evaluate and G itself will in general be complicated and may require many costly numerical operations for each term, limiting N_p
- Small N_p for which this procedure is practical will result in a noisy field
  - Enhanced, unphysically high, close approaches (collisions) with poor statistics can change the physics
- Special "fast multipole" methods based on Green's functions can reduce the scaling to ~N_p or ~$N_p ln(N_p)$.
  - Coefficient is large and smoothing is not easily implemented, potentially rendering such methods inferior to gridded methods (to be covered) for Vlasov evolution
  - May prove superior for scattering effects in particle formulations

// Example: Self fields in free space

$$G(\mathbf{x},\mathbf{x}') = \frac{1}{|\mathbf{x}-\mathbf{x}'|}\;;\quad \mathbf{E}_{si} = \frac{q_M}{4\pi\epsilon_0}\sum_{\substack{j=1\\ j\neq i}}^{N_p}\frac{(\mathbf{x}_i-\mathbf{x}_j)}{|\mathbf{x}_i-\mathbf{x}_j|^3}$$

//

Field Solution on a Discrete Grid

An alternative procedure is needed to

0) Calculate fields efficiently by discretization of the Maxwell equations
1) Smooth interactions to compensate for limited particle numbers

Approach: Solve the Maxwell Equations on a discrete spatial grid and then smooth the interactions calculated from the gridded field.

Discretization: 2D uniform grid (1D and 3D analogous)

$$x_i = x_{min} + i\Delta_x \quad \Delta_x = (x_{max}-x_{min})/n_x \quad i = 0,1,2,...,n_x$$
$$y_j = y_{min} + j\Delta_y \quad \Delta_y = (y_{max}-y_{min})/n_y \quad j = 0,1,2,...,n_y$$

$$\left.\begin{aligned}\mathbf{E}_{ij} &= \mathbf{E}(x_i,y_j)\\ \phi_{ij} &= \phi(x_i,y_j)\\ \rho_{ij} &= \rho(x_i,y_j)\end{aligned}\right\}$$ Field components, potential, and charge are gridded

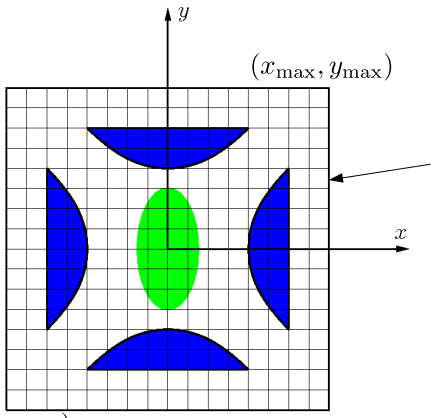
Comments:

- ρ_{ij} must be calculated from macro-particles, not necessarily on mesh points
- Fields will ultimately be needed at marco-particle coordinates, not on mesh points

These issues will be covered later under "particle weighting" in Sec. 4.D

Field Solution on a Discrete Grid: Example Problem, Beam in an Electric Quadrupole

Beam in an electric quadrupole lattice (2D)



specified on grid boundary and possibly on surfaces within the grid

Gridded Field Solution: Discretized Poisson Eqn.

For 2nd order centered differencing (see 4.B Basic Numerical Methods), the Poisson Equation

$$\nabla^2\phi = -\frac{\rho}{\epsilon_0} \qquad \Longrightarrow \qquad \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = -\frac{\rho}{\epsilon_0}$$

becomes

$$\frac{\phi_{i+1,\,j} - 2\phi_{i,\,j} + \phi_{i-1,\,j}}{\Delta_x^2} + \frac{\phi_{i,\,j+1} - 2\phi_{i,\,j} + \phi_{i,\,j-1}}{\Delta_y^2} = -\frac{\rho_{i,\,j}}{\epsilon_0}$$

with the gridded field components calculated as

$$E_x = -\frac{\partial\phi}{\partial x} \qquad E_y = -\frac{\partial\phi}{\partial y} \qquad \Longrightarrow \qquad E_{x_{ij}} = -\frac{\phi_{i+1,\,j} - \phi_{i-1,\,j}}{2\Delta_x} \qquad E_{y_{ij}} = -\frac{\phi_{i,\,j+1} - \phi_{i,\,j-1}}{2\Delta_y}$$

Boundary conditions must also be incorporated as constraint equations

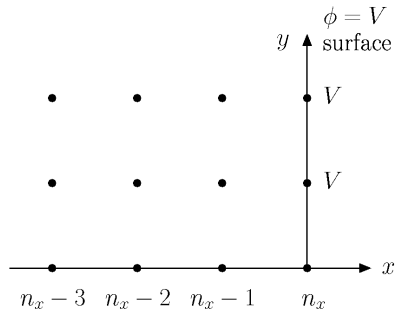
Dirichlet Conditions: ϕ specified on surfaces

Neumann Conditions: $\dfrac{\partial\phi}{\partial n}$ specified on surfaces

Gridded Field Solution: Discretized Dirichlet Boundary Cond

Dirichlet Conditions: ϕ specified on surface

Example: $\phi = V = \text{const}$ at right grid edge



$\phi_{n_x,\,j} = V = \text{const}$

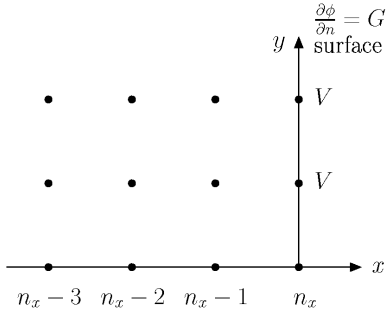
For $i = n_x - 1$ cells

$$\frac{V - 2\phi_{n_x-1,\,j} + \phi_{n_x-2,\,j}}{\Delta_x^2} + \frac{\phi_{n_x-1,\,j+1} - 2\phi_{n_x-1,\,j} + \phi_{n_x-1,\,j-1}}{\Delta_y^2} = -\frac{\rho_{n_x-1,\,j}}{\epsilon_0}$$

Gridded Field Solution: Discretized Neumann Boundary Cond

Neumann Conditions: $\dfrac{\partial\phi}{\partial n}$ specified on surfaces

Example: $\dfrac{\partial\phi}{\partial n} = G = \text{const}$ at right grid edge



Use 1st order forward difference formula at surface

$$\frac{\phi_{n_x-1,\,j} - \phi_{n_x,\,j}}{\Delta_x} = G$$

Solution of Discretized Poisson Eqn -- Direct Matrix Method

The finite-differenced Poisson Equation and the boundary conditions can be expressed in matrix form as:

$$\overline{\mathbf{M}} \cdot \mathbf{\Phi} = \mathbf{S}$$

$\overline{\mathbf{M}}$ = Coefficients matrix from local finite differences. This matrix will be sparse, i.e., most elements will equal zero

$\mathbf{\Phi}$ = Vector of unknown potentials at grid points

$\mathbf{S}$ = "Source" terms resulting from beam charge deposited on the grid (ρ_{ij}) and known potentials from boundary condition constraints

Formal solution found by matrix inversion:

$$\mathbf{\Phi} = \overline{\mathbf{M}}^{-1} \cdot \mathbf{S}$$

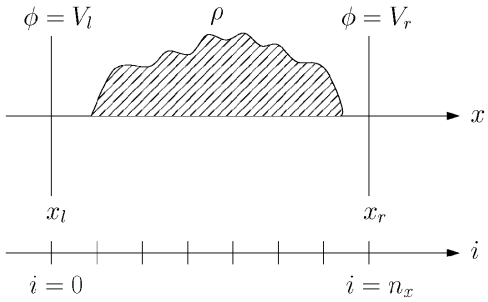
Direct inversion of $\overline{\mathbf{M}}^{-1}$ is not practical due to the large dimension of the problem

- $\overline{\mathbf{M}}$ will in general be sparse due to use of local, low-order finite differencing
- Many fast, numerically efficient inversion methods exist for sparse matrices
  - Specific method best used depends on type of differencing and BC's

Example Discretized Field Solution

To illustrate this procedure, consider a simple 1D example with Dirichlet BC's

$$\frac{d^2\phi}{dx^2} = -\frac{\rho}{\epsilon_0} \qquad \phi(x_l) = V_l \quad \text{left BC} \qquad \phi(x_r) = V_r \quad \text{right BC}$$



Discretize to $\mathcal{O}(\Delta_x^2)$:

$$\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta_x^2} = -\frac{\rho_i}{\epsilon_0}$$

$$\phi_0 = V_l$$
$$\phi_{n_x} = V_r$$

$$x_i = x_l + i\Delta_x, \quad \Delta_x = (x_r - x_l)/n_x; \quad i = 0, 1, 2, ..., n_x$$

Note: ρ_0 , ρ_{n_x} irrelevant

- Correspond to surface terms that fix boundary condition potentials

Example Discretized Field Solution (2)

The 1D discretized Poisson equation and boundary conditions can be expressed in matrix form as:

$$\begin{bmatrix} -2 & 1 & & & & & 0 \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ 0 & & & & & 1 & -2 \end{bmatrix} \cdot \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_{n_x-3} \\ \phi_{n_x-2} \\ \phi_{n_x-1} \end{bmatrix} = -\frac{\Delta_x^2}{\epsilon_0} \begin{bmatrix} \rho_1 + \frac{\epsilon_0}{\Delta_x^2} V_l \\ \rho_2 \\ \rho_3 \\ \vdots \\ \rho_{n_x-3} \\ \rho_{n_x-2} \\ \rho_{n_x-1} + \frac{\epsilon_0}{\Delta_x^2} V_r \end{bmatrix}$$

Matrix has tri-diagonal structure and can be rapidly inverted using optimized numerical methods to efficiently calculate the ϕ_i

- Sparse matrices need not be stored in full (waste of memory)

Field Solution Methods on Grid

Many other methods exist to solve the discretized field equations. These methods fall into three broad classes:

1) Direct Matrix Methods
- Fast inversion of sparse matrix

2) Spectral Methods
- Fast Fourier Transform (FFT)
  - Periodic boundary conditions
  - Sine transform ($\phi = 0$ on grid boundary)
  - FFT + capacity matrix for arbitrary conductors
  - Free space boundary conditions

3) Relaxation Methods
- Successive over-relaxation (SOR)
  - General boundary conditions and structures
- Multigrid (good, fast, and accurate method for complicated boundaries)

Field Solution Methods on Grid Continued (2)

Sometimes methods in these three classes are combined. For example, one might employ spectral methods transversely and invert the tri-diagonal matrix longitudinally.

Other discretization procedures are also widely employed, giving rise to other classes of field solutions such as:
- Finite elements
- Variational
- Monte Carlo

Methods of field solution are central to the efficient numerical solution of intense beam problems. It is not possible to review them all here. But before discussing particle weighting, we will first overview the important spectral methods and FFT's

Spectral Methods and the FFT

The spectral approach combined with numerically efficient Fast Fourier Transforms (FFT's) is commonly used to efficiently solve the Poisson Equation on a discrete spatial grid

- Approach provides spectral information on fields that can be used to smooth the interactions
- Efficiency of method enabled progress in early simulations in the 1960s
  - Computers had very limited memory and speed
- Method remains important and can be augmented in various ways to implement needed boundary conditions
  - Simple to code using numerical libraries for FFT
  - Efficiency still important ... especially in 3D geometries

Spectral Method: Discrete Fourier Transform

Illustrate in 1D for simplicity (multidimensional case analogous)

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

Continuous Fourier Transforms (Reminder)

$$\tilde{\phi}(k) = \int_{-\infty}^{\infty} dx\, e^{ikx}\phi(x) \qquad \tilde{\rho}(k) = \int_{-\infty}^{\infty} dx\, e^{ikx}\rho(x)$$

$$\phi(x) = \int_{-\infty}^{\infty} \frac{dk}{2\pi}\, e^{-ikx}\tilde{\phi}(k) \qquad \rho(x) = \int_{-\infty}^{\infty} \frac{dk}{2\pi}\, e^{-ikx}\tilde{\rho}(k)$$

$$i \equiv \sqrt{-1}$$

Transform Poisson Equation: $k^2\tilde{\phi}(k) = \dfrac{\tilde{\rho}(k)}{\epsilon_0}$

$$\phi(x) = \int_{-\infty}^{\infty} \frac{dk}{2\pi} e^{-ikx} \frac{\tilde{\rho}(k)}{\epsilon_0 k^2}$$

Similar procedures work to calculate the field on a finite, discrete spatial grid

- Develop by analogy to continuous transforms

// Aside: Transform conventions and notation vary

Physics convention:

- Reflects common usage in dynamics and quantum mechanics

$$\tilde{\phi}(k) = \int_{-\infty}^{\infty} dx\, e^{ikx}\phi(x)$$

$$\phi(x) = \int_{-\infty}^{\infty} \frac{dk}{2\pi}\, e^{-ikx}\tilde{\phi}(k)$$

Symmetrical convention:

- Factors of $\sqrt{2\pi}$ used symmetrically can be convenient numerically

$$\tilde{\phi}(k) = \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}}\, e^{ikx}\phi(x)$$

$$\phi(x) = \int_{-\infty}^{\infty} \frac{dk}{\sqrt{2\pi}}\, e^{-ikx}\tilde{\phi}(k)$$

Sometimes $i \to -i$

Subtlety:

- If $\phi(x) \neq 0$ as $|x| \to \infty$ then k must contain a large enough positive imaginary part for transform to exist and contour to carry out inversion contour must be taken consistently //

Discrete Fourier Transform (2)

Discretize the problem as follows:

$$x_j = x_{min} + j\Delta_x; \quad \Delta_x = \frac{x_{max} - x_{min}}{n_x}; \quad j = 0, 1, 2, ..., n_x$$

$n_x + 1$ grid points

$$\phi_j \equiv \phi(x_j)$$

n_x distinct values (periodic)

$$k_n \equiv \frac{2\pi n}{n_x \Delta_x} \qquad n = -\frac{n_x}{2}, ..., 0, ..., \frac{n_x}{2}$$

- In this section we employ j as a grid index to avoid confusion with $i \equiv \sqrt{-1}$

The discrete transform is the defined by analogy to the continuous transform by:

$$\tilde{\phi}(k_n) = \int_{-\infty}^{\infty} dx\, e^{ik_n x}\phi(x) \iff \tilde{\phi}_n \equiv \Delta_x \sum_{j=0}^{n_x} e^{ik_n(x_j - x_{\min})}\phi_j$$

$$= \Delta_x \sum_{j=0}^{n_x} \exp\left(\frac{i2\pi nj}{n_x}\right)\phi_j$$

Discrete Fourier Transform (3)

$$\underline{\text{Grid}} \qquad\qquad \underline{\text{Transform}}$$
$$\phi_j \quad (n_x+1 \text{ values}) \iff \tilde{\phi}_n \quad (n_x+1 \text{ complex values})$$

Note that $\tilde{\phi}_n$ is periodic in *n* with period n_x

$$\tilde{\phi}_{-n} = \tilde{\phi}_{n_x - n}$$

- Let $n = 0, 1, 2, ..., n_x$ so *n* and *j* have the same ranges

Then an inverse transform can be constructed *exactly*:

$$\phi_j = \frac{1}{n_x \Delta_x} \sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x}\right) \tilde{\phi}_n$$

- This exact inversion is proved in the problems by summing a geometric series

Spectral Methods: Aliasing

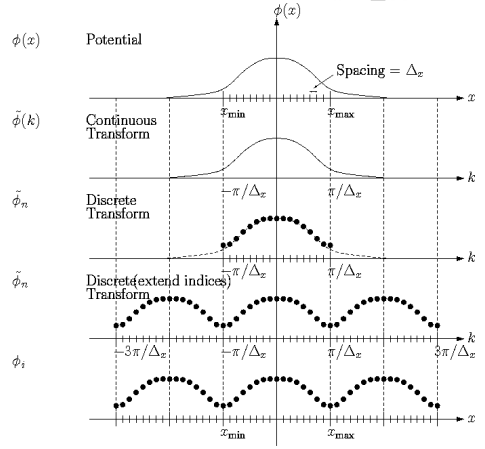
The discrete transform describes a periodic problem if indices are extended

- Discretization errors (aliasing) can occur

Figure to be edited:

$$\lim_{x \to \pm\infty} \phi(x) = 0$$

Plots will be replaced with real transforms based on a Gaussian distribution in future versions of the notes



Discrete Transform Formulas

Application of the Discrete Fourier Transform to solve Poisson's Equation:

$$E_x = -\frac{d\phi}{dx} \iff E_{xj} = -\frac{\phi_{j+1} - \phi_{j-1}}{2\Delta_x}$$

$$\frac{d^2\phi}{dx^2} = -\frac{\rho}{\epsilon_0} \iff \frac{\phi_{j+1} - 2\phi_j + \phi_{j-1}}{\Delta_x^2} = -\frac{\rho_j}{\epsilon_0}$$

Applying the discrete transform yields:

$$\tilde{E}_{xn} = i\kappa_n \tilde{\phi}_n \qquad \kappa_n = k_n \left[\frac{\sin(k_n\Delta_x)}{k_n\Delta_x}\right] \qquad k_n \equiv \frac{2\pi n}{(n_x+1)\Delta_x}$$
$$\equiv k_n (k_n \Delta_x)$$

Poisson's Equation becomes:

$$\tilde{\phi}_n = \frac{\tilde{\rho}_n}{\epsilon_0 K_n^2} \; ; \quad K_n^2 \equiv k_n^{2\,2}(k_n\Delta_x/2) = k_n^2 \left[\frac{\sin(k_n\Delta_x/2)}{k_n\Delta_x/2}\right]^2$$

Note: factors of K_n^2 need only be calculated once per simulation (store values)

Derivation of Discrete Transform Eqns.

/// Example Derivation of a formula for the discrete transformed E-field:

Discretized E-field $E_{xj} = -\frac{\phi_{j+1} - \phi_{j-1}}{2\Delta_x}$

Transforms

$$\phi_j = \frac{1}{(n_x+1)\Delta_x} \sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x+1}\right) \tilde{\phi}_n$$

$$E_{xj} = \frac{1}{(n_x+1)\Delta_x} \sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x+1}\right) \tilde{E}_{xn}$$

Substitute transforms into difference formula:

$$2\Delta_x \sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x+1}\right) \tilde{E}_{xn}$$

$$= -\sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x+1}\right) \tilde{\phi}_n \left\{\exp\left(-\frac{i2\pi n}{n_x+1}\right) - \exp\left(\frac{i2\pi n}{n_x+1}\right)\right\}$$

$$= \Delta_x \sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x+1}\right) \tilde{E}_{xn} = i \sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x+1}\right) \sin\left(\frac{2\pi n}{n_x+1}\right) \tilde{\phi}_n$$

$$\Delta_x \sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x+1}\right)\tilde{E}_{xn} = i\sum_{n=0}^{n_x} \exp\left(-\frac{i2\pi nj}{n_x+1}\right)\sin\left(\frac{2\pi n}{n_x+1}\right)\tilde{\phi}_n$$

This equation must hold true for each term in the sum proportional to $\exp\left(-\frac{i2\pi nj}{n_x+1}\right)$ to be valid for a general j.

$$\Rightarrow \tilde{E}_{xn} = \frac{i}{\Delta_x}\sin\left(\frac{2\pi n}{n_x+1}\right)\tilde{\phi}_n$$

$$k_n = \frac{2\pi n}{(n_x+1)\Delta_x}$$

$$\Rightarrow \boxed{\tilde{E}_{xn} = ik_n\left[\frac{\sin(k_n\Delta_x)}{k_n\Delta_x}\right]\tilde{\phi}_n = ik_n \mathrm{dif}(k_n\Delta_x)\tilde{\phi}_n}$$

///

Comments on Discrete Fourier Transform (DFT):

- Formulated method in 1D for simplicity, but straightforward to generalize to 2D or 3D
  - Apply transform along each coordinate axis
- Can optimize further than the simple sketch given here: Field is real
- Zero potential outer grid boundary conditions simply incorporated by use of sine transform variant
- Symmetries can be exploited to implement free space boundary conditions by using a mesh only 4x larger than the region containing particles
  - Allows use of fine mesh only where needed
  - Implemented in some common accelerator codes like IMPACT
  - See Hockney and Eastwood book
- Method outlined can be augmented by the use of capacity matrices to put conducting structures within mesh
  - Beyond scope of this discussion

Spectral Methods: Discrete Transform Field Solution

Typical discrete Fourier transform field solution (not optimized)

Forward Transform		Multiply		Inverse Transform		Finite Difference	
ρ_j	$\longrightarrow$	$\tilde{\rho}_n$	$\longrightarrow$	$\tilde{\phi}_n$	$\longrightarrow$	ϕ_j	$\longrightarrow$ E_{xj}
	DFT		$\cdot\frac{1}{K_n^2}$		IDFT		

DFT = Discreet Fourier Transform
IDFT = Inverse Discreet Fourier Transform

Comments

- K_n^2 factors can be calculated once and stored to increase numerical efficiency
- E_{xj} is typically found on grid using finite difference of ϕ_j's rather than from $\tilde{E}_{xj}$ and inverse discreet Fourier transform
  - Less optimized numerical work
  - More simply integrated in code with other discretized grid methods

Discussion of Spectral Methods and the FFT

The Fast Fourier Transform (FFT) makes this procedure numerically efficient

- Discrete transform (no optimization), ~$(n_x + 1)^2$ complex operations
- FFT exploits symmetries to reduce needed operations to ~ $(n_x + 1)ln(n_x + 1)$
  - Huge savings for large n_x
- The needed symmetries exist only for certain numbers of grid points. In the simplest manifestations: $n_x + 1 = 2^p$, $p = 1, 2, 3, ...$
  - Reduced freedom in grid choices
  - Other manifestations allow $n_x + 1 = 2^p$ and products of prime numbers for more possibilities

The FFT can be combined with other procedures such as capacity matrices to implement boundary conditions for interior conductors, etc.

- Allows rapid field solutions in complicated conductor geometries when capacity matrix elements can be pre-calculated and stored
- Symmetries can be exploited using 4x domain size to implement free-space boundary conditions (see Hockney and Eastwood)

FFT is the fastest method for simple geometries

- Simple to code using typical numerical libraries for FFT's

D: Weighting: Depositing Particles on the Field Mesh and Interpolating Gridded Fields to Particles

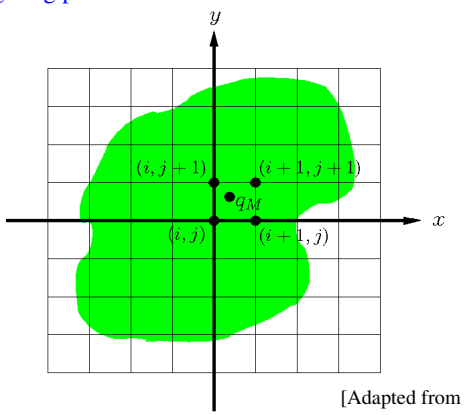
We have outlined methods to solve the electrostatic Maxwell's equations on a discrete spatial grid. To complete the description we must:

- Specify how to deposit macro-particle charges and current onto the grid – Macroparticles not generally at mesh points
- Specify how to interpolate fields on the spatial grid points to the macroparticle coordinates (not generally at mesh points) to apply in the particle advance
- Smooth interactions resulting from the small number of macro-particles to reduce artificial collisions resulting from the use of an unphysically small number of macro-particles needed for rapid simulation

This is called the *particle weighting* problem

Weighting (2)

Particle weighting problem for electrostatic fields



[Adapted from Birdsall and Langdon]

It is found that it is usually better to employ the same weighting schemes to deposit both the macro-particle charges and currents on the mesh and to extrapolate the fields at gridded points to the macro-particles

- Avoids unphysical self-forces where the particle accelerates itself

Weighting Methods

Many methods of particle weighting exist. They can be grouped into 4 categories:

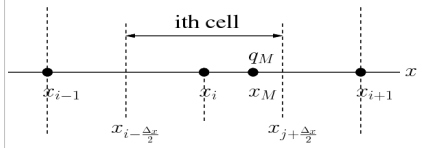
1) Nearest Grid Point
2) Cloud in Cell (CIC)
   - Shaped particles
   - PIC method, linearly shaped particles
3) Multipole
   - Dipole, subtracted dipole, etc.
4) Higher order methods
   - Splines
   - ***k***-space cutoffs in discrete transforms
   - ⋮

Possible hybrid methods also exist. We will illustrate methods 1) and 2) for electrostatic problems. Descriptions of other methods can be found in the literature.

Weighting: Nearest Grid Point

1) Nearest Grid Point: Assign charges to the nearest grid cell

- Fast and simple: Show for 1D; 2D and 3D generalization straightforward
- Noisy

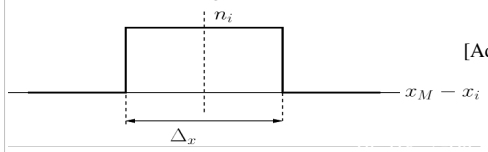


$q_M =$ Charge of macro-particle

$x_M =$ Coordinate of macro-particle

$x_i =$ Closest grid cell

Projected Density, ith Grid



[Adapted from Birdsall and Langdon]

Charge Deposition on ith Grid:

$$q_i = q_M$$

Field "Interpolation" to Particle:

$$E_x|_{x=x_M} = E_{xi}$$

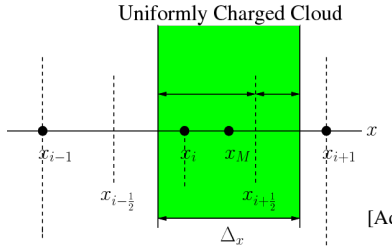
Comments on nearest grid point weighting:

- Very easy to code and fast, but square shaped particles with abrupt transitions as they move through the grid result in enhanced statistical noise
  – Method consequently not commonly used
- Currents can be interpolated to grid similarly for electromagnetic field solves and/or diagnostics. Deposit macro-particle current density contributions
  $q_M \mathbf{v}_M$ $\quad$ $\mathbf{v}_M$ = Macro-Particle velocity
  on nearest mesh point.
- The 1D example is contrived: 1D Poisson equation Green's function simple/fast (sum charge to left – sum charge to right)
  – Use 1D only to show method simply; 2D and 3D relevant

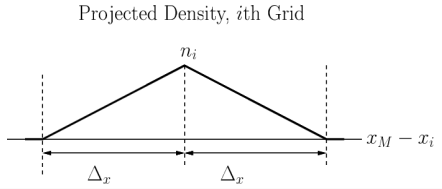
Weighting : Cloud in Cell

2) Cloud in Cell: Shaped macro-particles pass freely through each other

- Smoother than Nearest Grid Point, but more numerical work
- For linear interpolation results in simple, commonly used "Particle in Cell" (PIC) method



[Adapted from Birdsall and Langdon]



Cloud in Cell (2)

q_M , x_M = Charge and coordinate of macro-particle

x_i = Closest grid cell

Charge Deposition on ith Grid:

$$q_i = q_M \left[\frac{\Delta_x - (x_M - x_i)}{\Delta_x} \right] = q_M \frac{x_{i+1} - x_M}{\Delta_x}$$

$$q_{i+1} = q_M \left[\frac{x_M - x_i}{\Delta_x} \right]$$

Field Interpolation to Particle:

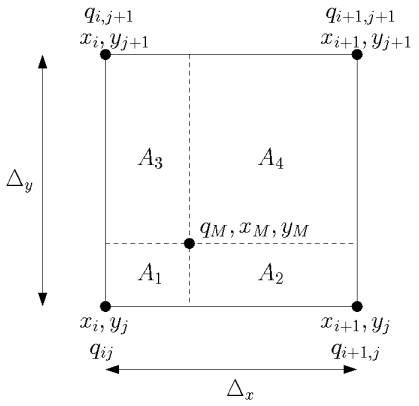
$$E_x|_{x=x_M} = \left[\frac{x_{i+1} - x_M}{\Delta_x} \right] E_i + \left[\frac{x_M - x_i}{\Delta_x} \right] E_{i+1}$$

Comments:

- Linear interpolation results in triangularly shaped particles
- Shape smooths interactions reducing collisionality
  – Vlasov evolution with limited number of shaped particles
- Simple shape is fast to calculate numerically
- Currents can be similarly deposited on grid similarly for electromagnetic solving and/or diagnostics by depositing current density contributions $q_M \mathbf{v}_M$

Weighting: Area Weighting

In a 2D cloud-in-cell system, weighting is accomplished using rectangular "area weighting" to nearest grid points



q_M = Macro-particle charge

(x_M, y_M) = Macro-particle coordinates

q_{ij} = Mesh charges

Area Weighting (2)

Charge Deposition to Four Nearest Grids:

$$q_{ij} = q_M \frac{A_4}{\Delta_x \Delta_y} \qquad A_4 = (x_{i+1} - x_M)(y_{j+1} - y_M)$$

$$q_{i+1,j} = q_M \frac{A_3}{\Delta_x \Delta_y} \qquad A_3 = (x_M - x_i)(y_{j+1} - y_M)$$

$$q_{i,j+1} = q_M \frac{A_2}{\Delta_x \Delta_y} \qquad A_2 = (x_{i+1} - x_M)(y_M - y_j)$$

$$q_{i+1,j+1} = q_M \frac{A_1}{\Delta_x \Delta_y} \qquad A_1 = (x_M - x_i)(y_M - y_j)$$

Field Interpolation From Four Nearest Grids:

$$\mathbf{E} = \frac{A_4}{\Delta_x \Delta_y}\mathbf{E}_{ij} + \frac{A_3}{\Delta_x \Delta_y}\mathbf{E}_{i+1,j} + \frac{A_2}{\Delta_x \Delta_y}\mathbf{E}_{i,j+1} + \frac{A_1}{\Delta_x \Delta_y}\mathbf{E}_{i+1,j+1}$$

Comments:

- Procedure easily generalized to 3D using opposing diagonal volume elements of the eight grid points bounding the grid cell
- Currents can be interpolated to grid similarly for electromagnetic solving and/or diagnostics

// Aside: Efficient numerical numerical operation for area weighting

Give outline on how to efficiently
code for rapid calculation with
minimal number of multiplications.

//

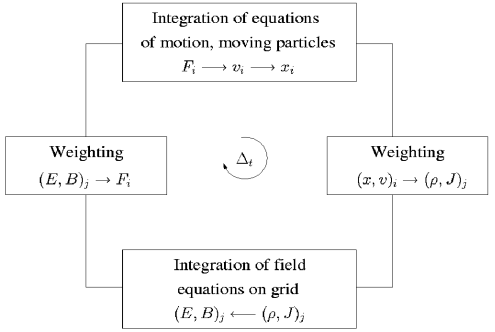
Higher Order Weighting: Splines

To be added: Slide on Splines to
illustrate what is meant by higher order methods
Make Points:
- Requires more numerical work and harder to code
- Some schemes can introduce neg probability problems
- Should evaluate against simpler low order methods using same computer power to see which method wins.

S4E: Computational Cycle for Particle-In-Cell Simulations

We now have (simplified) notions of the parts that make up a Particle-In-Cell (PIC) simulation of Vlasov beam evolution

0) Particle Moving
1) Field Solver on a discrete grid
2) Weighting of particle and fields to and from the grid



[Adapted from Birdsall and Langdon]

Computational Cycle for Particle-In-Cell Simulations Contd.

Comments:

- Diagnostics must also be accumulated for useful runs (see Intro. Lec. 05)
  - Particles (coordinates and velocities) and fields will need to be synchronized (common time) when diagnostics are accumulated
- Initial conditions must be set (particle load, see Intro. Lec. 06)
  - Particle and field variables may need appropriate de-synchronization to initialize advance
- Benchmarking/Testing is critical and also very difficult
  - Must test thoroughly to convince yourself answers are correct
  - Known problems useful for testing: analytic, when possible, allows precise error evaluation
  - Invariants (e.g., system canonical angular momentum in axisymmetric systems) provide strong checks
  - Other benchmarked codes on established problems provide good checks
  - Push algorithms to clear failure so you learn what is dangerous
  - Can be surprising how methods fail when applied outside of original intended context: important to check/verify with problem changes!

Ignorance is not bliss in simulation. It is very dangerous.
Check often and carefully.

Corrections and suggestions for improvements welcome!

These notes will be corrected and expanded for reference and for use in future editions of US Particle Accelerator School (USPAS) and Michigan State University (MSU) courses. Contact:

Prof. Steven M. Lund
Facility for Rare Isotope Beams
Michigan State University
640 South Shaw Lane
East Lansing, MI 48824

lund@frib.msu.edu
(517) 908 – 7291 office
(510) 459 - 4045 mobile

Please provide corrections with respect to the present archived version at:

https://people.nscl.msu.edu/~lund/uspas/scs_2016

Redistributions of class material welcome. Please do not remove author credits.

References:

Particle Methods

C.K. Birdsall and A.B. Langdon, *Plasma Physics via Computer Simulation*, McGraw-Hill Book Company (1985)

R.W. Hockney and J.W. Eastwood, *Computer Simulation using Particles*, Institute of Physics Publishing (1988)