Simulation Techniques for Intense Beams*

Steven M. Lund
Lawrence Livermore National Laboratory (LLNL)

Steven M. Lund and John J. Barnard

USPAS: "Beam Physics with Intense Space-Charge"
UCB: "Interaction of Intense Charged Particle Beams with Electric and Magnetic Fields"

US Particle Accelerator School (USPAS)
University of California at Berkeley (UCB)

Nuclear Engineering Department NE 290H
Spring Semester, 2009
(Version 20090115)

* Research supported by the US Dept. of Energy at LLNL and LBNL under contract Nos. DE-AC52-07NA27344 and DE-AC02-05CH11231.

Simulation Techniques for Intense Beams: Outline

Why Numerical Simulation

Classes of Intense Beam Simulations

Overview of Basic Numerical Methods

Numerical Methods for Particle and Distribution Methods

Diagnostics

Initial Distributions and Particle Loading

Numerical Convergence

Practical Considerations

Overview of the WARP Code

Example Simulations

References

Simulation Techniques for Intense Beams: Detailed Outline

1) Why Numerical Simulation?

2) Classes of Intense Beam Simulations
   - A. Overview
   - B. Particle Methods
   - C. Distribution Methods
   - D. Moment Methods
   - E. Hybrid Methods

3) Overview of Basic Numerical Methods
   - A. Discretization
   - B. Discrete Numerical Operations
     - Derivatives
     - Quadrature
     - Irregular Grids and Axisymmetric Systems
   - C. Time Advance
     - Overview
     - Euler and Runge-Kutta Advances
     - Solution of Moment Methods

Detailed Outline - 2

4) Numerical Methods for Particle and Distribution Methods

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

Detailed Outline - 3

5) Diagnostics

6) Initial Distributions and Particle Loading

7) Numerical Convergence

8) Practical Considerations
   - A. Overview
   - B. Fast Memory
   - C. Run Time
   - D. Machine Architectures

9) Overview of the WARP Code

10) Example Simulations
   - A. ESQ Injector
   - B. ....

Contact Information

Acknowledgments

References

S1: Why Numerical Simulation?

Builds intuition of intense beam physics

- "The purpose of computation is insight, not numbers"
  Richard Hamming, chief mathematician of the Manhattan Project and Turning Award recipient
- Advantages over laboratory experiments:
  - Full nonintrusive beam diagnostics are possible
  - Effects can be turned on and off

Allows analysis of more realistic situations than analytically tractable

- Realistic geometries
- Non-ideal distributions
- Combined effects
- Large amplitude (nonlinear) effects

Insight obtained can motivate analytical theories

- Suggest and test approximations and models to most simply express relevant effects

Why Numerical Simulation? (2)

Can quantify expected performance of specific machines

- Machines and facilities expensive – important to have high confidence that systems will work as intended

Computers and numerical methods/libraries are becoming more powerful

Enables both analysis of more realistic problems and/or better numerical convergence

- Bigger and faster hardware
  - Processor speed increasing
  - Parallel machine architectures
  - Greater memory
- More developed software
  - Improved numerical methods
  - Libraries of debugged code modules
  - Graphics and visualization tools

Why Numerical Simulation? (3)

Simulations are increasingly powerful and valuable in the analysis of intense beams, but should not be used to exclusion

- Parametric scaling is *very* important in machine design
  - Often it is hardest to understand what specific choices should be made in physical aperture sizes, etc.
  - Although scaling can be explored with simulation, analytical theory often best illustrates the trade-offs, sensitivities, and relevant combinations of parameters
- Concepts often fail due to limits of technology (e.g., fabrication tolerances, material failures, and unanticipated properties) and hence full laboratory testing is vital
  - Many understood classes of errors can be probed with simulation.
  - Unanticipated error sources are most dangerous
- Economic realities often severely limit what can be constructed
  - Simulating something unattainable may serve little purpose

Why Numerical Simulation? (4)

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

- Motivates model simplifications and identification of relevant sensitivities

Numerical simulation skills are highly sought in many areas of accelerator and beam physics

- Specialists readily employable
- Skills transfer easily to many fields of physics and engineering

Numerous programming languages are employed in numerical simulations of intense beams

- Most common today: Fortran, Fortran 90, C, C++, Java, ...
- Strengths and weaknesses depend on application, preferences, and history (legacy code)

Results are analyzed with a variety of graphics packages:

- Commonly used: NCAR, Gist, Gnuplot, IDL, Narcisse...
- Plot frames combine into movies
- Use can greatly simplify construction of beam visualization diagnostics

Why Numerical Simulation? (5)

A modern and flexible way to construct simulation packages is to link routines in fast, compiled code with an interactive interpreter such as:

- Examples: Python, Basis, Yorick, ...

Advantages of using interactive interpreters:

- Allows routines to be coded in mixed languages
  - Renders choice of programming languages less important
- Flexible reconfiguration of code modules possible to adapt for specific, unanticipated needs
  - Reduces need for recompilation and cumbersome structures for special uses
  - Aids cross-checking problems and debugging
- "Steering" of code during runs to address unanticipated side effects
- In the case of Python, facilitates modern, object-oriented structure

Why Numerical Simulation? (6)

Discussing particular programming languages and graphics packages is beyond the scope of this class. Here our goal is to survey numerical simulation methods employed without presenting details of specific implementations.

However, we will show examples based on the "WARP" particle-in-cell code developed for intense beam simulation at LLNL and LBNL

- WARP is so named since it works on a "warped" Cartesian mesh with bends
- WARP is a family of particle-in-cell code tools built around a common Python interpreter for flexible operation
- Optimized for the simulation of intense beams with self-consistent electrostatic space-charge forces
- Actively maintained and extended:
  - Diagnostics
  - E-cloud
  - Electromagnetic effects and dense plasmas

More on WARP later after discussion of methods, etc.

⋮

S2: Classes of Intense Beam Simulations
S2A: Overview

There are three distinct classes of modeling of intense ion beams applicable to numerical simulation

1) Particle methods (see: S2B)
2) Distribution methods (see: S2C)
3) Moment methods (see: S2D)

All of these draw heavily on methods developed for the simulation of neutral plasmas. The main differences are:

- Lack of overall charge neutrality
  - Single species typical, though electron + ion simulations are common too
- Directed motion of the beam along accelerator axis
- Applied field descriptions of the lattice
  - Optical focusing elements
  - Accelerating structures

We will review and contrast these methods before discussing specific numerical implementations

S2B: Particle Methods: Equations of Motion

Classical point particles are advanced with self-consistent interactions given by the Maxwell Equations

- Most general: If actual number of particles are used, this is approximately the physical beam under a classical (non-quantum) theory
- Often intractable using real number of beam particles due to numerical work and problem size
- Method also commonly called *Molecular Dynamics* simulations

Equations of motion (time domain, 3D, for generality)
ith particle:

$$\frac{d\mathbf{p}_i}{dt} = \mathbf{F}_i = q_i\left(\mathbf{E} + \frac{d\mathbf{x}_i}{dt} \times \mathbf{B}\right)$$

$$m_i\gamma_i\frac{d\mathbf{x}_i}{dt} = \mathbf{p}_i \quad ; \quad \gamma_i = \left[1 + \frac{\mathbf{p}_i^2}{(m_i c)^2}\right]^{1/2}$$

Initial conditions

$$\mathbf{x}_i(t=0)$$

$$\mathbf{p}_i(t=0)$$

Particle orbits $\mathbf{x}_i(t)$, $\mathbf{p}_i(t)$ solved as a function of time

S2B: Particle Methods: Fields

Fields (electromagnetic in most general form)

Charge Density

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}$$

$$\rho(\mathbf{x},t) = \rho_{\text{ext}}(\mathbf{x},t) + \sum_i q_i \delta[\mathbf{x} - \mathbf{x}_i(t)]$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

external (applied)

particle beam

Current Density

$$\nabla \cdot \mathbf{B} = 0$$

$$\mathbf{J}(\mathbf{x},t) = \mathbf{J}_{\text{ext}}(\mathbf{x},t) + \sum_i q_i \frac{d\mathbf{x}}{dt} \delta[\mathbf{x} - \mathbf{x}_i(t)]$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t}$$

+ boundary conditions on $\mathbf{E}$, $\mathbf{B}$

S2C: Distribution Methods: Equations of Motion

Distribution Methods

- Based on reduced (statistical) continuum models of the beam
- Two classes: (microscopic) kinetic models and (macroscopic) fluid models
- Here, distribution means a function of continuum variables
- Use a 3D collision-less Vlasov model to illustrate concept
  - Obtained from statistical averages of particle formulation

Example Kinetic Model: Vlasov Equation of Motion

$q_i = q$; $m_i = m$; easy to generalize for multiple species (see later slide)

$$\left\{ \frac{\partial}{\partial t} + \mathbf{v} \cdot \frac{\partial}{\partial x} + q\left(\mathbf{E} + \mathbf{v} \times \mathbf{B}\right) \right\} f(\mathbf{x}, \mathbf{p}, t) = 0$$

$$\mathbf{v} = \frac{\mathbf{p}}{\gamma m} = \frac{\mathbf{p}/m}{[1 + \mathbf{p}^2/(mc)^2]^{1/2}}$$

Initial condition

$f(\mathbf{x}, \mathbf{p}, t = 0)$

$f(\mathbf{x}, \mathbf{p}, t)$ evolved from t = 0

$\mathbf{x}, \ \mathbf{p}, \ t$ independent variables

S2C: Distribution Methods: Fields

Fields: Same as in particle methods but with ρ, $\mathbf{J}$ expressed in proper form for coupling to the distribution

Charge Density

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}$$

$$\rho(\mathbf{x},t) = \rho_{\text{ext}}(\mathbf{x},t) + q\int d^3p f(\mathbf{x},\mathbf{p},t)$$

external (applied)

beam

Current Density

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0\epsilon_0 \frac{\partial \mathbf{E}}{\partial t}$$

$$\mathbf{J}(\mathbf{x},t) = \mathbf{J}_{\text{ext}}(\mathbf{x},t) + q\int d^3p\ \mathbf{v} f(\mathbf{x},\mathbf{p},t)$$

+ boundary conditions on $\mathbf{E}$, $\mathbf{B}$

S2C: Distribution Methods: Vlasov Equation

The Vlasov Equation is essentially a continuity equation for an incompressible "fluid" in 6D phase-space. To see this, note that

$$\frac{\partial}{\partial \mathbf{p}} \cdot \mathbf{v} \times \mathbf{B} = 0$$

The Vlasov Equation can be expressed as

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{x}} \cdot (\mathbf{v} f) + \frac{\partial}{\partial \mathbf{p}} \cdot (q[\mathbf{E} + \mathbf{v} \times \mathbf{B}] f) = 0$$

$$\Rightarrow \frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{x}} \cdot \left(\frac{d\mathbf{x}}{dt}\bigg|_{\substack{f \\ \text{orbit}}} \right) + \frac{\partial}{\partial \mathbf{p}} \cdot \left(\frac{d\mathbf{p}}{dt}\bigg|_{\substack{f \\ \text{orbit}}} \right) = 0$$

which is manifestly the form of a continuity equation in 6D phase-space, i.e., probability is not created or destroyed

S2C: Distribution Methods: Collision Corrections

The effect of collisions can be included by adding a collision operator:

$$\left\{ \frac{\partial}{\partial t} + \mathbf{v} \cdot \frac{\partial}{\partial \mathbf{x}} + \frac{\partial}{\partial \mathbf{p}} \cdot (q[\mathbf{E} + \mathbf{v} \times \mathbf{B}]) \right\} f = \left. \frac{\partial f}{\partial t} \right|_{\text{coll}}$$

For most applications in beam physics, $\left. \frac{\partial f}{\partial t} \right|_{\text{coll}}$ can be neglected.

- See: estimates in J.J. Barnard, Intro Lectures

For exceptional cases, specific forms of collisions terms can be found in Nicholson, *Intro to Plasma Theory*, Wiley 1983, and similar plasma physics texts

S2C: Distribution Methods: Comment on the PIC Method

The common Particle-in-Cell (PIC) method is *not* really a particle method, but rather a distribution method that uses a collection of smoothed "macro" particles to simulate Vlasov's Equation. This can understood roughly by noting that Vlasov's Equation can be interpreted as

$$\frac{d}{dt} f(\mathbf{x}, \mathbf{p}, t) = 0$$

Total derivative along a test particle's path

$\Longrightarrow$ Advance particles in a continuous field "fluid" to eliminate particle collisions

Important Point:

PIC is a method to solve Vlasov's Equation, *not* a discrete particle method

This will become clear after these lectures

S2C: Distribution Methods: Multispecies Generalizations

Subscript species with *j*. Then in the Vlasov equation replace:

$$f \longrightarrow f_j$$

$$m \longrightarrow m_j$$

$$q \longrightarrow q_j$$

and there is a separate Vlasov equation for each of the *j* species.

Replace the charge and current density couplings in the Maxwell Equations with and appropriate form to include charge and current contributions from all species:

$$\rho(\mathbf{x},t) = \rho_{\text{ext}}(\mathbf{x},t) + \sum_j q_j \int d^3p\, f_j(\mathbf{x},\mathbf{p},t)$$

$$\mathbf{J}(\mathbf{x},t) = \mathbf{J}_{\text{ext}}(\mathbf{x},t) + \sum_j q_j \int d^3p\, \mathbf{v} f_j(\mathbf{x},\mathbf{p},t)$$

Also, if collisions are included the collision operator should be generalized to include collisions between species as well as collisions of a species with itself

S2C: Fluid Models

Fluid Models

- Obtained from further averages of kinetic model
- Described in terms of "macroscopic" variables (density, flow velocity, pressure...) that vary in ***x*** and *t*
- Models must be closed (truncated) at some order via physically motivated assumptions (cold, negligible heat flow, ...)

Moments:

Density n :
$$n(\mathbf{x},t) = \int d^3p\ f(\mathbf{x},\mathbf{p},t)$$

Flow velocity $\mathbf{V}$:
$$n\mathbf{V}(\mathbf{x},t) = \int d^3p\ \mathbf{v}f(\mathbf{x},\mathbf{p},t)$$

Flow momentum $\mathbf{P}$:
$$n\mathbf{P}(\mathbf{x},t) = \int d^3p\ \mathbf{p}f(\mathbf{x},\mathbf{p},t)$$

Pressure tensor $\mathcal{P}_{ij}$:
$$n\mathcal{P}_{ij}(\mathbf{x},t) = \int d^3p\ [p_i - P_i(\mathbf{x},t)] \times [v_j - V_j(\mathbf{x},t)]f(\mathbf{x},\mathbf{p},t)$$

Higher rank objects $\vdots$ $\vdots$ $\vdots$

S2C: Fluid Models: Equations of Motion

Equations of Motion (Eulerian approach)

Continuity:

$$\frac{\partial n}{\partial t} + \frac{\partial}{\partial \mathbf{x}} \cdot [n\mathbf{V}] = 0$$

Force: ith component

$$n\left(\frac{\partial}{\partial t} + \mathbf{V} \cdot \frac{\partial}{\partial \mathbf{x}}\right) P_i + \sum_j \frac{\partial}{\partial x_j} \mathcal{P}_{ij} = qn[\mathbf{E} + \mathbf{V} \times \mathbf{B}]_i$$

Field:

Maxwell Equations with charge and current density coupling to fluid variables given by:

$$\rho(\mathbf{x}, t) = \rho_{\text{ext}}(\mathbf{x}, t) + qn(\mathbf{x}, t)$$
$$\mathbf{J}(\mathbf{x}, t) = \mathbf{J}_{\text{ext}}(\mathbf{x}, t) + qn(\mathbf{x}, t)\mathbf{V}(\mathbf{x}, t)$$

S2C: Fluid Model: Multispecies Generalization

Subscript species with *j*. Then in the continuity, force, pressure, ... equations replace

Particle Properties	Moments
$m \longrightarrow m_j$	$n \longrightarrow n_j$
$q \longrightarrow q_j$	$\mathbf{V} \longrightarrow \mathbf{V}_j$
	$\vdots$

Replace the charge and current density couplings in the Maxwell Equations with

$$\rho(\mathbf{x},t) = \rho_{\text{ext}}(\mathbf{x},t) + \sum_j q_j n_j(\mathbf{x},t)$$

$$\mathbf{J}(\mathbf{x},t) = \mathbf{J}_{\text{ext}}(\mathbf{x},t) + \sum_j q_j n_j(\mathbf{x},t)\mathbf{V}_j(\mathbf{x},t)$$

S2C: Lagrangian Formulation of Distribution Methods

In kinetic and especially fluid models it can be convenient to adopt *Lagrangian* methods. For fluid models these can be distinguished as follows:

Eulerian Fluid Model:

Flow quantities are functions of space (***x***) and and evolve in time (*t*)

- Example: density $n(\mathbf{x}, t)$ and flow velocity $\mathbf{V}(\mathbf{x}, t)$

Lagrangian Fluid Model:

Identify parts of evolution (flow) with objects (material elements) and follow the flow in time (*t*)

- Shape and position of elements must generally evolve to represent flow
- Example: envelope model edge radii $r_x(s),\ r_y(s)$

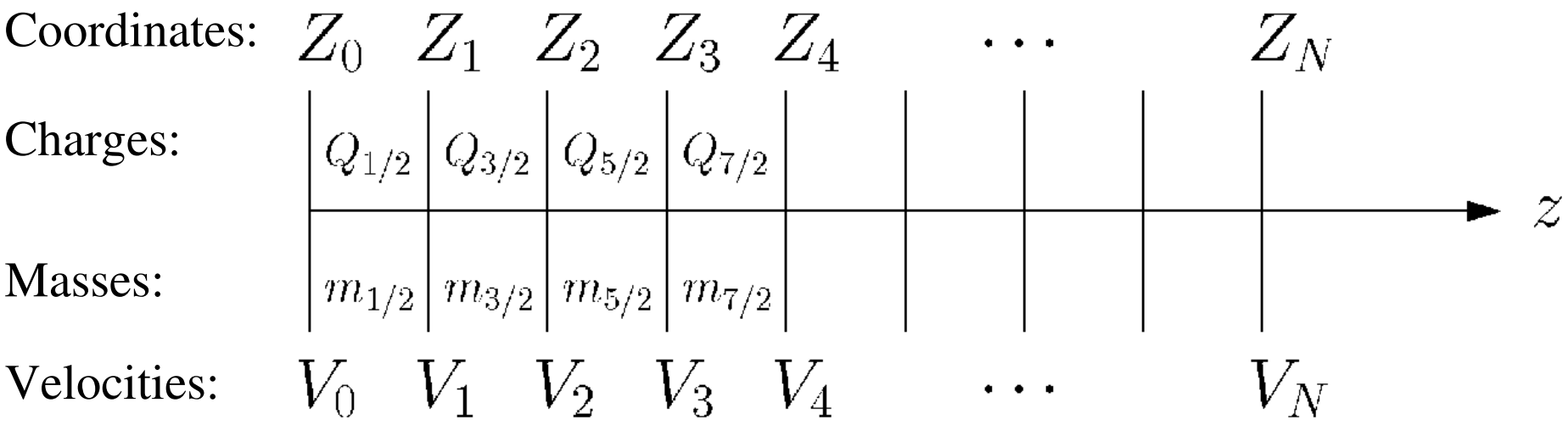
Many distribution methods for Vlasov's Equation are hybrid Lagrangian methods

- Macro particle "shapes" in PIC (Particle in Cell) method to be covered can be thought of as Lagrangian elements representing a Vlasov flow

S2C: Example Lagrangian Fluid Model

1D Lagrangian model of the longitudinal evolution of a cold beam

- Discretize fluid into longitudinal elements with boundaries
- Derive equations of motion for elements



$z = Z_i$ slice boundaries

$\frac{dZ_i}{dt} = V_i$ velocities of slice boundaries

$Q_{i+1/2}$ fixed

$m_{i+1/2}$ fixed

$\frac{q}{m} = \text{const}$

for single species

(set initial coordinates)

Example Lagrangian Fluid Model, Continued (2)

Solve the equations of motion

$$\frac{dZ_i(t)}{dt} = V_i(t)$$

$$\frac{dV_i(t)}{dt} = \frac{q}{m} E_z(Z_i, t)$$

for all the slice boundaries. Several methods might be used to calculate E_z:

1) Take "slices" to have some radial extent modeled by a perpendicular envelope etc. and deposit the $Q_{i+1/2}$ onto a grid and solve:

$$\nabla^2 \phi = -\frac{\rho}{\epsilon_0} \qquad E_z = -\frac{\partial \phi}{\partial z}$$

subject to $E_z \to 0$ as $|z| \to \infty$

2) Employ a "g-factor" model

$$E_z = -\frac{g}{4\pi\epsilon_0}\frac{\partial \lambda}{\partial z}$$

λ calculated from $Q_{i+1/2}$ and radial extent of the elements etc.

3) Pure 1D model using Gauss' Law

S2D: Moment Methods

Moment Methods

- Most reduced description of an intense beam
  - Often employed in lattice designs
- Beam represented by a finite (closed and truncated) set of moments that are advanced from initial values
  - Here by moments, we mean functions of a single variable *s* or *t*
- Such models are not generally self-consistent
  - Some special cases such as a stable transverse KV equilibrium distribution (see: S.M. Lund lectures on Transverse Equilibrium Distributions) are consistent with truncated moment description (rms envelope equation)
  - Typically derived from assumed distributions with self-similar evolution
- See: S.M. Lund lectures on Transverse Equilibrium Distributions for more details on moment methods

S2D: Moment Methods: 1st Order Moments

Many moment models exist. Illustrate with examples for transverse beam evolution

Moment definition:

$$\langle ... \rangle_\perp \equiv \frac{\int d^2x_\perp \int d^2x'_\perp ... f}{\int d^2x_\perp \int d^2x'_\perp f}$$

1st order moments:

$$\mathbf{X} = \langle \mathbf{x} \rangle_\perp \qquad \text{Centroid coordinate}$$

$$\mathbf{X}' = \langle \mathbf{x}' \rangle_\perp \qquad \text{Centroid angle}$$

$$\Delta = \left\langle \frac{\delta p_s}{p_s} \right\rangle_\perp \equiv \langle \delta \rangle_\perp \qquad \text{Off momentum}$$

$$\vdots \qquad\qquad \vdots$$

S2D: Moment Methods: 2nd and Higher Order Moments

2nd order moments:

x moments y moments x-y cross moments dispersive moments

$$\begin{array}{lll} \langle x^2 \rangle_\perp & \langle y^2 \rangle_\perp & \langle xy \rangle_\perp \\ \langle xx' \rangle_\perp & \langle yy' \rangle_\perp & \langle x'y \rangle_\perp , \ \langle xy' \rangle_\perp \\ \langle x'^2 \rangle_\perp & \langle y'^2 \rangle_\perp & \langle x'y' \rangle_\perp \end{array} \qquad \begin{array}{l} \langle x\delta \rangle_\perp , \ \langle y\delta \rangle_\perp \\ \langle x'\delta \rangle_\perp , \ \langle y'\delta \rangle_\perp \\ \langle \delta^2 \rangle_\perp \end{array}$$

It is typically convenient to subtract centroid from higher-order moments

$$\begin{array}{ll} \tilde{x} \equiv x - X & \tilde{x}' \equiv x' - X' \\ \tilde{y} \equiv y - Y & \tilde{y}' \equiv y' - Y' \end{array}$$

$$\langle \tilde{x}^2 \rangle_\perp = \langle (x - X)^2 \rangle_\perp = \langle x^2 \rangle_\perp - X^2, \ \text{etc.}$$

3rd order moments: Analogous to 2nd order case, but more for each order

$$\langle x^3 \rangle_\perp , \ \langle x^2 y \rangle_\perp , \ \ldots$$

S2D: Moment Methods: Common 2nd Order Moments

Many quantities of physical interest are expressed in terms of moments

Statistical beam size: (rms edge measure)

$$r_x = 2\left\langle \tilde{x}^2 \right\rangle_\perp^{1/2}$$
$$r_y = 2\left\langle \tilde{y}^2 \right\rangle_\perp^{1/2}$$

Statistical emittances: (rms edge measure)

$$\varepsilon_x = 4\left[\left\langle \tilde{x}^2 \right\rangle_\perp \left\langle \tilde{x}'^2 \right\rangle_\perp - \left\langle \tilde{x}\tilde{x}' \right\rangle_\perp^2\right]^{1/2}$$
$$\varepsilon_y = 4\left[\left\langle \tilde{y}^2 \right\rangle_\perp \left\langle \tilde{y}'^2 \right\rangle_\perp - \left\langle \tilde{y}\tilde{y}' \right\rangle_\perp^2\right]^{1/2}$$

S2D: Moment Methods: Equations of Motion

Equations of Motion

- Can be expressed in terms of moments of combinations of moments that are of physical interest
- Moments are advanced from specified initial conditions

Form equations:

$$\frac{d}{ds}\mathbf{M} = \mathbf{F}(\mathbf{M})$$

M = vector of moments, generally infinite

F = vector function of **M**, generally nonlinear

Moment methods generally form an infinite chain of equations that do *not* truncate. To be useful the system must be truncated. Truncations are usually carried out by assuming a specific form of the distribution that can be described by a finite set of moments

- Self-similar evolution: form of distribution assumed not to change
  - Analytical solutions often employed
- Neglect of terms

A simple example will be employed to illustrate these points

S2D: Moment Methods: Example: Transverse Envelope Eqns.

Truncation assumption: unbunched uniform density elliptical beam in free space

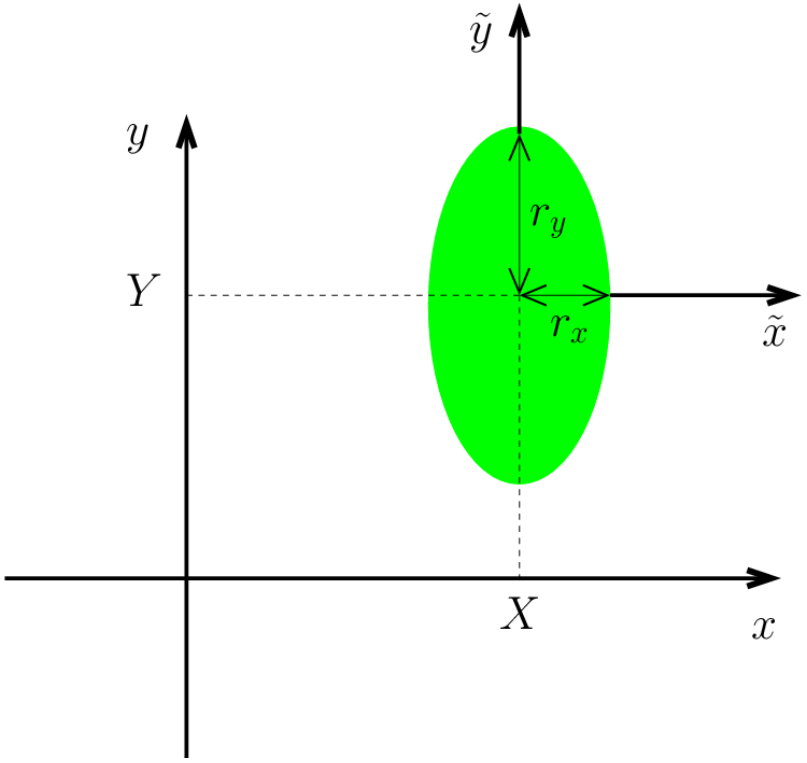
- $\delta = 0$, no axial velocity spread
- All cross moments zero, i.e. $\langle \tilde{x}\tilde{y} \rangle_\perp = 0$

Centroid:

$$X = \langle x \rangle_\perp$$
$$Y = \langle y \rangle_\perp$$

Envelope:

$$r_x = 2 \langle \tilde{x}^2 \rangle_\perp^{1/2}$$
$$r_y = 2 \langle \tilde{y}^2 \rangle_\perp^{1/2}$$



For: $\dfrac{\tilde{x}^2}{r_x^2} + \dfrac{\tilde{y}^2}{r_y^2} < 1$

$$E_{\tilde{x}} = \frac{\lambda}{\pi \epsilon_0} \frac{\tilde{x}}{(r_x + r_y) r_x}$$

$$E_{\tilde{y}} = \frac{\lambda}{\pi \epsilon_0} \frac{\tilde{y}}{(r_x + r_y) r_y}$$

$\lambda =$ line charge density

These results are employed to derive the moment equations of motion
(See S.M. Lund lectures on Transverse Centroid and Envelope Models)

Example Continued (2) - Equations of Motion in Matrix Form

$$\frac{d}{ds}\begin{bmatrix} X \\ X' \\ Y \\ Y' \end{bmatrix} = \begin{bmatrix} X' \\ -\kappa_x(s)X \\ Y' \\ -\kappa_y(s)Y \end{bmatrix}$$

$$\frac{d}{ds}\begin{bmatrix} \langle \tilde{x}^2 \rangle_\perp \\ \langle \tilde{x}\tilde{x}' \rangle_\perp \\ \langle \tilde{x}'^2 \rangle_\perp \\ \langle \tilde{y}^2 \rangle_\perp \\ \langle \tilde{y}\tilde{y}' \rangle_\perp \\ \langle \tilde{y}'^2 \rangle_\perp \end{bmatrix} = \begin{bmatrix} 2\langle \tilde{x}\tilde{x}' \rangle_\perp \\ \langle \tilde{x}'^2 \rangle_\perp - \kappa_x(s)\langle \tilde{x}^2 \rangle_\perp + \frac{Q\langle \tilde{x}'^2 \rangle_\perp}{[4\langle \tilde{x}^2 \rangle_\perp^{1/2}(\langle \tilde{x}^2 \rangle_\perp^{1/2} + \langle \tilde{y}^2 \rangle_\perp^{1/2})]} \\ -2\kappa_x(s)\langle \tilde{x}\tilde{x}' \rangle_\perp + \frac{2Q\langle \tilde{x}\tilde{x}' \rangle_\perp}{[4\langle \tilde{x}^2 \rangle_\perp^{1/2}(\langle \tilde{x}^2 \rangle_\perp^{1/2} + \langle \tilde{y}^2 \rangle_\perp^{1/2})]} \\ 2\langle \tilde{y}\tilde{y}' \rangle_\perp \\ \langle \tilde{y}'^2 \rangle_\perp - \kappa_y(s)\langle \tilde{y}^2 \rangle_\perp + \frac{Q\langle \tilde{y}'^2 \rangle_\perp}{[4\langle \tilde{y}^2 \rangle_\perp^{1/2}(\langle \tilde{x}^2 \rangle_\perp^{1/2} + \langle \tilde{y}^2 \rangle_\perp^{1/2})]} \\ -2\kappa_y(s)\langle \tilde{y}\tilde{y}' \rangle_\perp + \frac{2Q\langle \tilde{y}\tilde{y}' \rangle_\perp}{[4\langle \tilde{y}^2 \rangle_\perp^{1/2}(\langle \tilde{x}^2 \rangle_\perp^{1/2} + \langle \tilde{y}^2 \rangle_\perp^{1/2})]} \end{bmatrix}$$

- Form truncates due to assumed distribution form
- Self-consistent with the KV distribution. See: S.M. Lund lectures on Transverse Equilibrium Distributions

Example Continued (3) - Reduced Form Equations of Motion

Using 2nd order moment equations we can show that

$$\frac{d}{ds}\varepsilon_x^2 = 0 = \frac{d}{ds}\varepsilon_y^2$$

$$\Longrightarrow \quad \boxed{\begin{aligned} \varepsilon_x^2 &= 16\left[\langle x^2\rangle_\perp \langle x'^2\rangle_\perp - \langle xx'\rangle_\perp^2\right] = \text{const} \\ \varepsilon_y^2 &= 16\left[\langle y^2\rangle_\perp \langle y'^2\rangle_\perp - \langle yy'\rangle_\perp^2\right] = \text{const} \end{aligned}}$$

The 2nd order moment equations can be equivalently expressed as

$$\boxed{\begin{aligned} \frac{dr_x}{ds} &= r_x' \;; \quad \frac{d}{ds}r_x' + \kappa_x r_x - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_x^2}{r_x^3} = 0 \\ \frac{dr_y}{ds} &= r_y' \;; \quad \frac{d}{ds}r_y' + \kappa_y r_y - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_y^2}{r_y^3} = 0 \end{aligned}}$$

Example Continued (4) : Contrast Form of Matrix and Reduced Form Moment Equations

Relative advantages of the use of coupled matrix form versus reduced equations can depend on the problem/situation

Coupled Matrix Equations

$$\frac{d}{ds}\mathbf{M} = \mathbf{F}$$

M = Moment Vector

F = Force Vector

- Easy to formulate
  - Straightforward to incorporate additional effects
- Natural fit to numerical routine
  - Easy to code

Reduced Equations

$$X'' + \kappa_x X = 0$$

$$r_x'' + \kappa_x r_x - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_x^2}{r_x^3} = 0$$

etc.

Reduction based on identifying invariants such as

$$\varepsilon_x^2 = 16\left[\left\langle \tilde{x}^2 \right\rangle_\perp \left\langle \tilde{x}'^2 \right\rangle_\perp - \left\langle \tilde{x}\tilde{x}' \right\rangle_\perp^2\right]$$

helps understand solutions

- Compact expressions

S2E: Hybrid Methods

Beyond the three levels of modeling outlined earlier:

0) Particle methods
1) Distribution methods
2) Moment methods

there exist numerous "hybrid" methods that combine features of several methods.

Examples:

- Particle-in-Cell (PIC) models with shaped particles
- Gyro-kinetic models
  - Average over fast gyro motion in magnetic fields: common in plasma physics
- Delta-f models
  - Evolve perturbed distribution with marker particles

⋮

Hybrid Methods Continued (2)

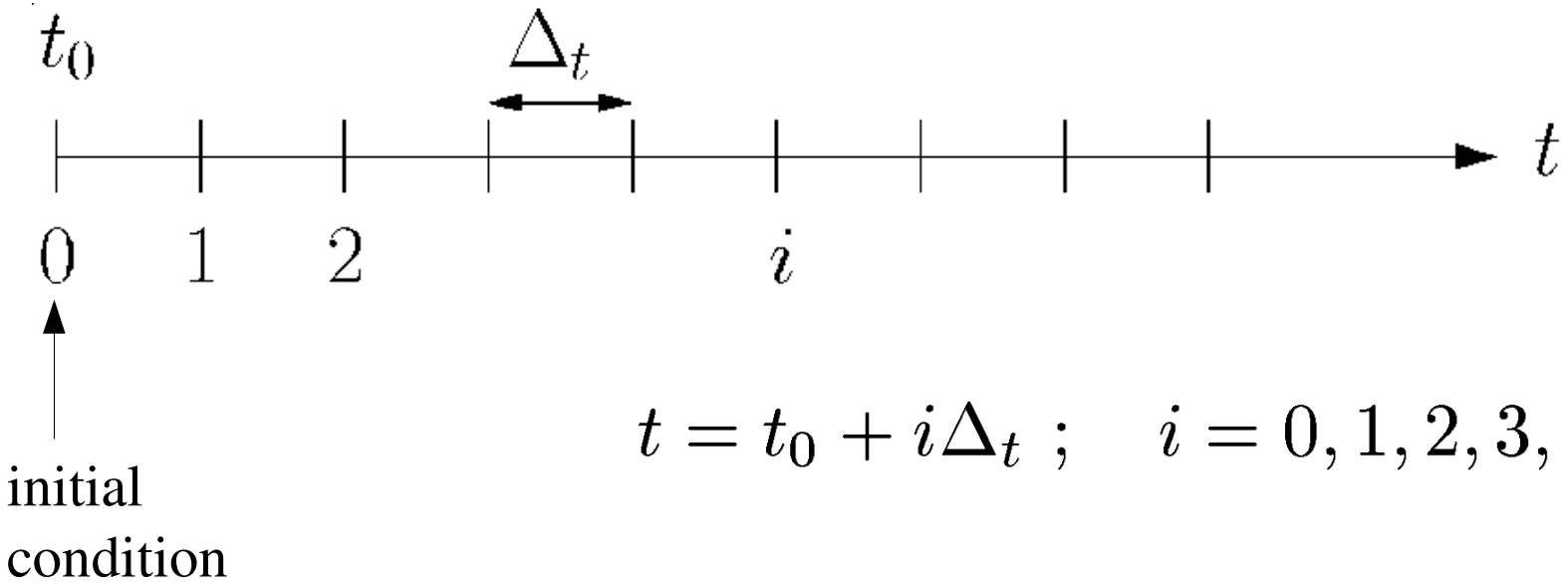
General comments:

- Particle and distribution methods are appropriate for higher levels of detail
- Moment methods are used for rapid iteration of machine design
  - Moments also typically calculated as diagnostics in particle and distribution methods
- Even within one (e.g. particle) there are many levels of description:
  - Electromagnetic and electrostatic, with many field solution methods
  - 1D, 2D, 3D
  - :
- Employing a hierarchy of models with full diagnostics allows cross-checking (both in numerics and physics) and aids understanding
  - No single method is best in all cases

S3: Overview of Basic Numerical Methods
S3A: Discretizations

General approach is to discretize independent variables in each of the methods and solve for dependent variables which in some cases may be discretized as well

time (or s)



$$t = t_0 + i\Delta_t \; ; \quad i = 0, 1, 2, 3, ...$$

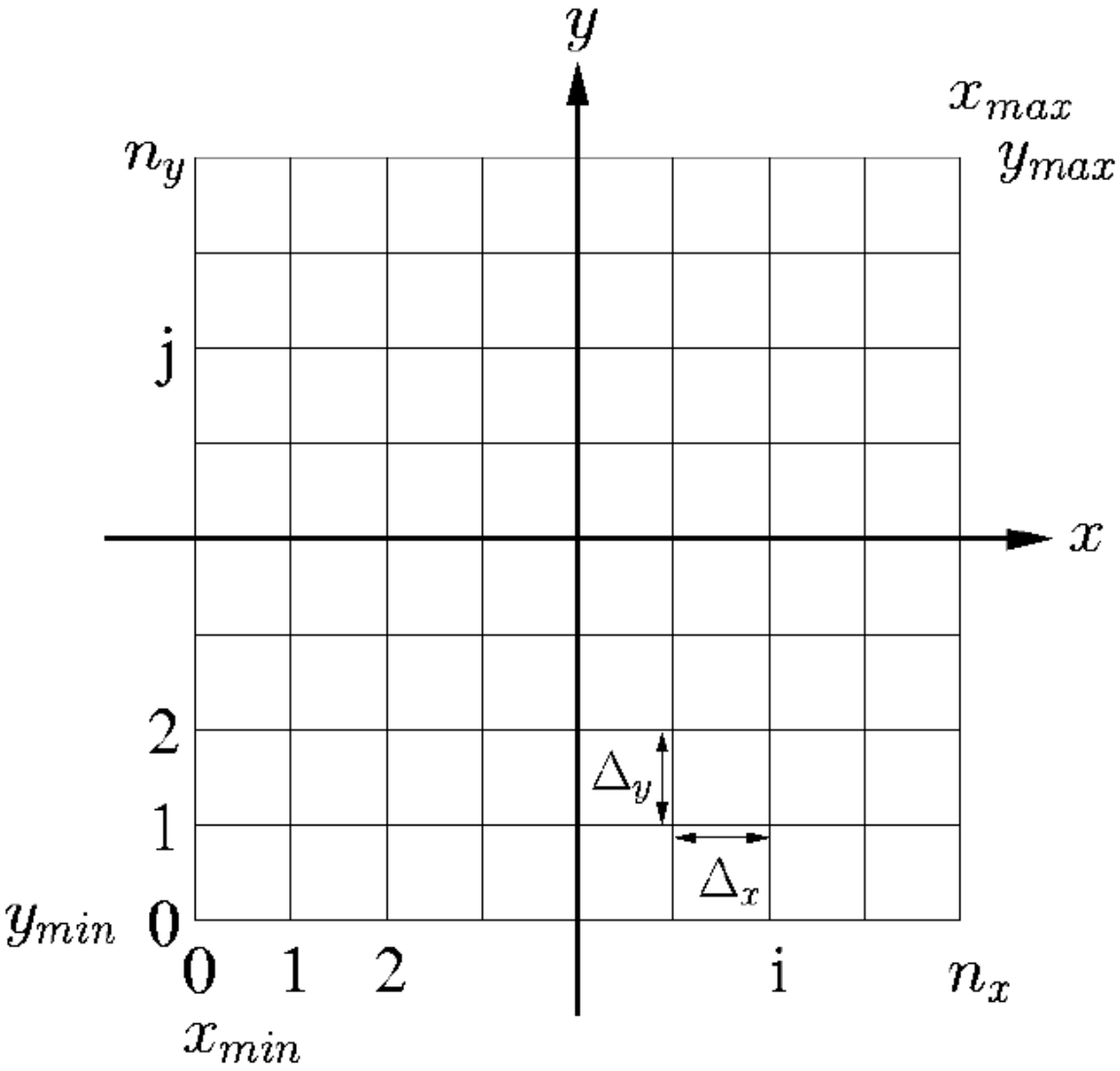
- Nonuniform meshes also possible
  - Can add resolution where needed
  - Increases complexity

In typical applications may apply these descriptions in a variety of ways

- Move a transverse thin slice of a beam...

Transverse Coordinate Discretization

Spatial Coordinates (transverse)



$$x_i = x_{min} + i\Delta_x$$

$$y_j = y_{min} + j\Delta_y$$

$$\Delta_x = (x_{max} - x_{min})/n_x$$

$$\Delta_y = (y_{max} - y_{min})/n_y$$

$$i = 0, 1, 2, ..., n_x$$

$$j = 0, 1, 2, ..., n_y$$

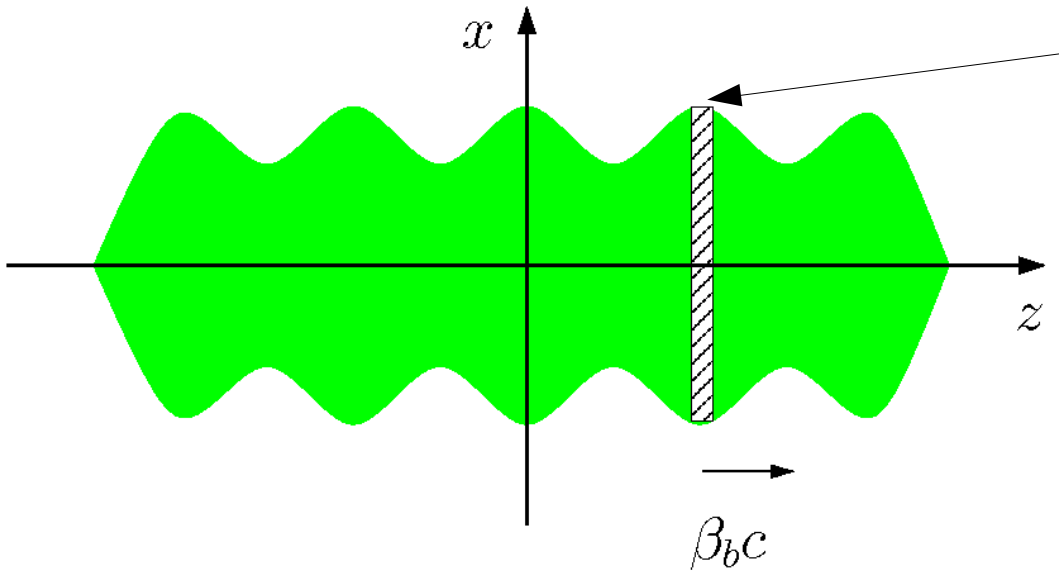
Analogous for 3D, momentum coordinates, etc.

Transverse Coordinate Discretization – Applications

In typical applications may apply these discretizations in a variety of ways:

Transverse Slice Simulation:

- Move a transverse thin "slice" of beam along the axial coordinate *s* of a reference particle



Thin slice of a long pulse is advanced and the transverse grid moves with the slice

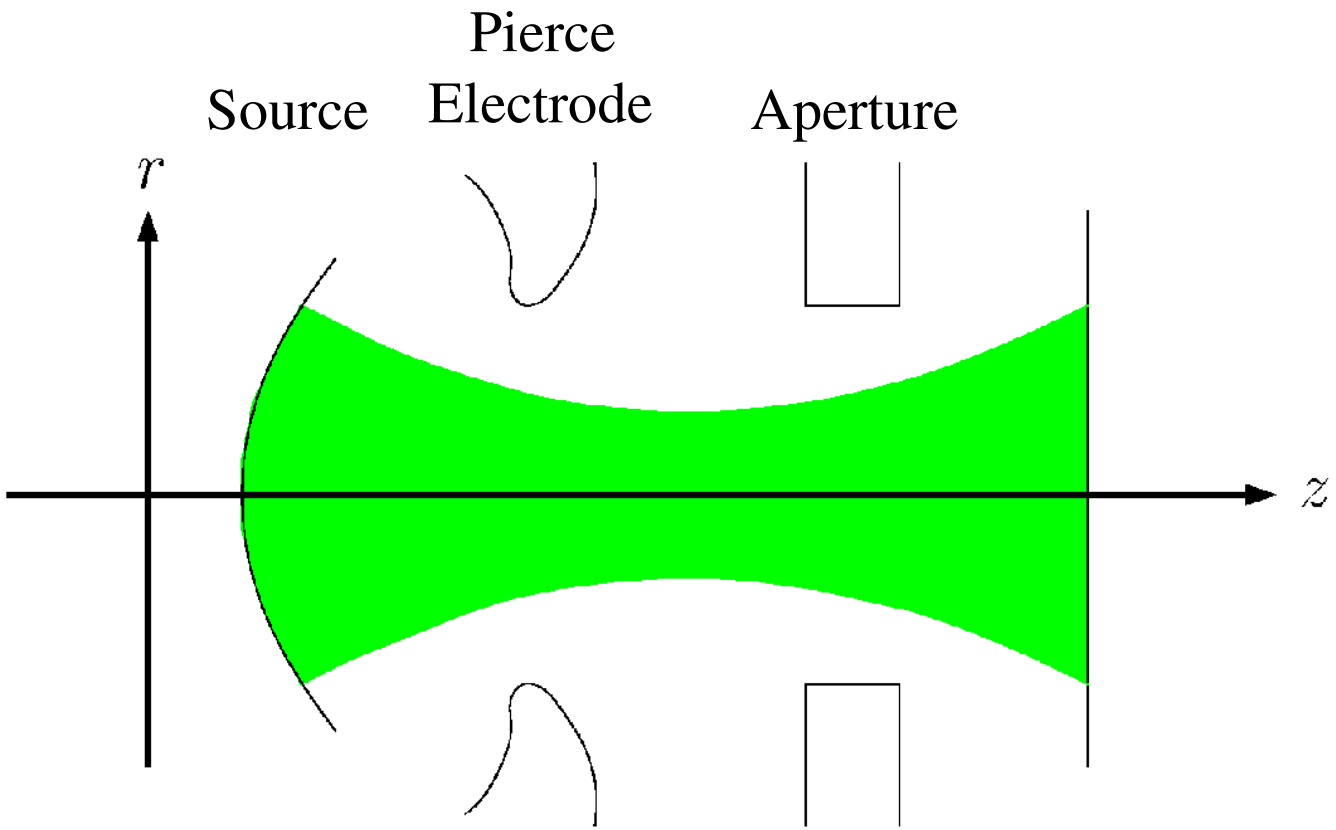
- Limitations:
  - This "unbunched" approximation is not always possible
  - 3D effect can matter, e.g. in short pulses and/or beams ends

Transverse Coordinate Discretization – Applications (2)

Steady State Simulation:

- Simulate the middle of a long pulse where a time stationary beam fills the grid

Example: Mid-Pulse Diode

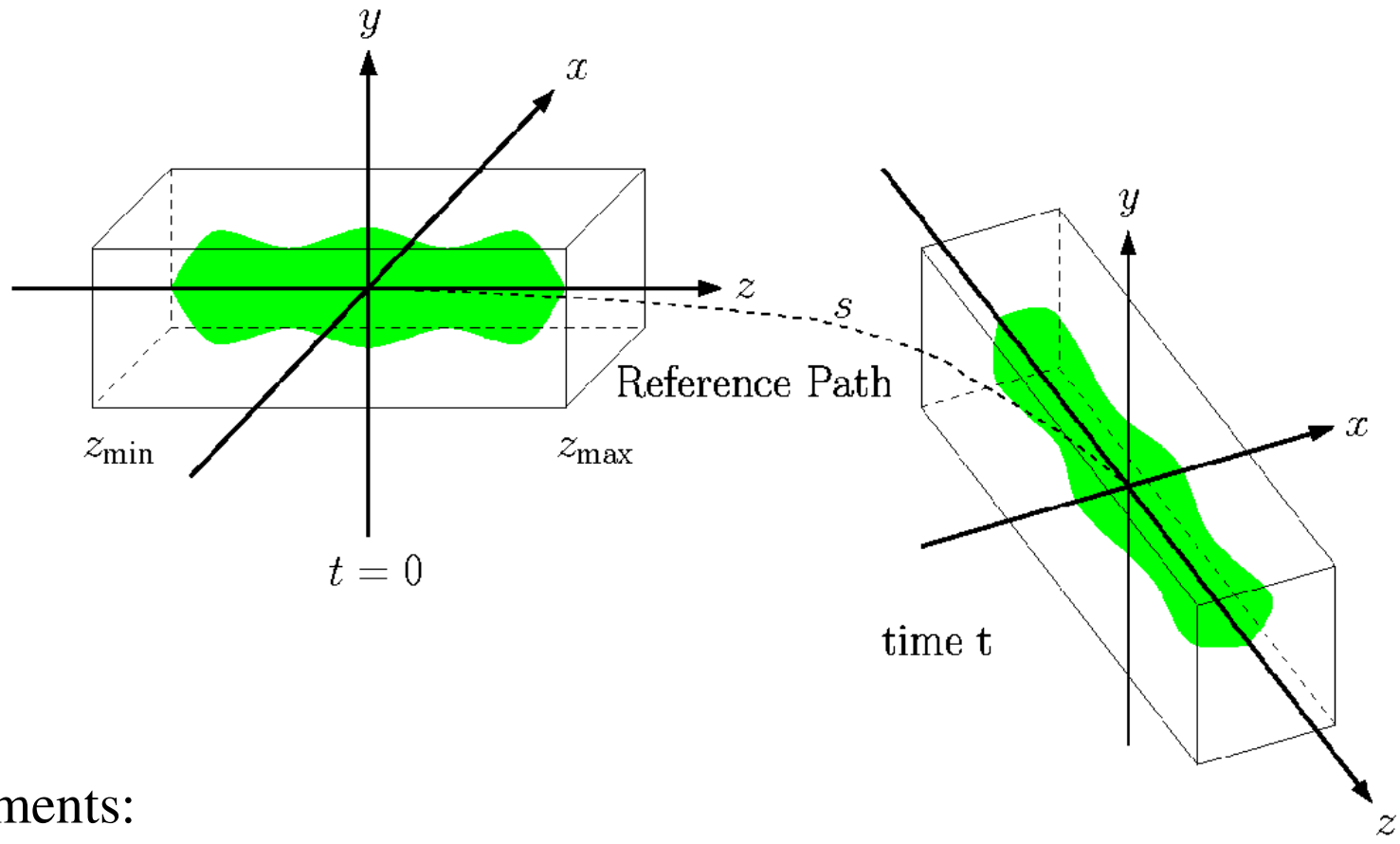


- Mesh is stationary, leading to limitations
  - Beam pulse always has ends: see J.J. Barnard lectures on Longitudinal Physics
  - Assumes that the mid-pulse in nearly time-independent in structure

Transverse Coordinate Discretization – Applications (3)

Full 3D Simulation

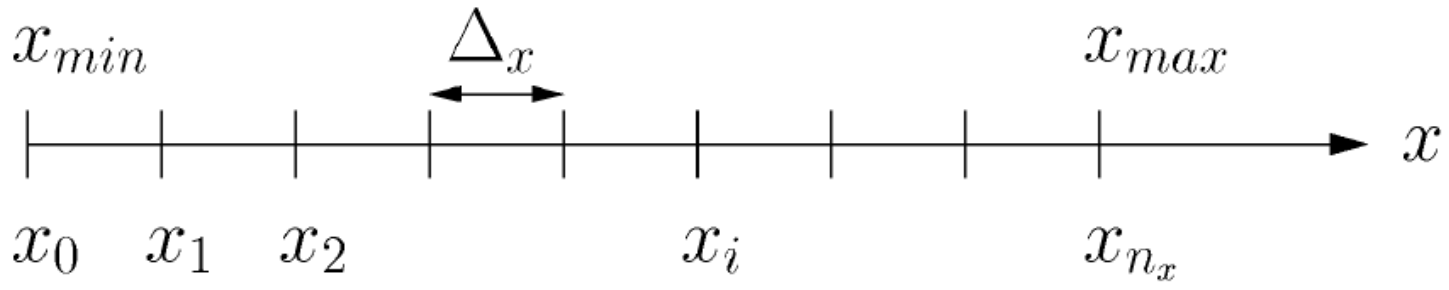
- Simulate a 3D beam with a moving mesh that follows a reference particle (possibly beam centroid).



- Comments:
  - Most realistic level of modeling, but also most numerically intensive
  - Grid can be moved in discretized jumps so that applied fields maintain alignment with the grid

S3B: Discrete Numerical Operations

Let x represent a spatial coordinate and $f(x)$ some continuous function of x



$$x_i = x_{min} + i\Delta_x; \quad \Delta_x = (x_{max} - x_{min})/n_x$$

$$i = 0, 1, 2, ..., n_x$$

Denote $f_i \equiv f(x_i)$, etc. and Taylor expand one grid point forward and backward about $x = x_i$

$$f_{i+1} = f_i + \left.\frac{\partial f}{\partial x}\right|_i \Delta_x + \frac{1}{2!}\left.\frac{\partial^2 f}{\partial x^2}\right|_i \Delta_x^2 + \frac{1}{3!}\left.\frac{\partial^3 f}{\partial x^3}\right|_i \Delta_x^3 + \ldots$$

$$f_{i-1} = f_i - \left.\frac{\partial f}{\partial x}\right|_i \Delta_x + \frac{1}{2!}\left.\frac{\partial^2 f}{\partial x^2}\right|_i \Delta_x^2 - \frac{1}{3!}\left.\frac{\partial^3 f}{\partial x^3}\right|_i \Delta_x^3 + \ldots$$

The same methodology can be applied to other spatial (x, y, etc.), axial (s), and temporal (t) coordinates

Discrete Numerical Operations: Derivatives

Simple, but inaccurate expressions for 1st order derivatives follow immediately from the forward and backward expansions

2 point: (non-centered)

Forward:
$$\left.\frac{\partial f}{\partial x}\right|_i = \frac{f_{i+1} - f_i}{\Delta_x} + \mathcal{O}(\Delta_x)$$

Backward:
$$\left.\frac{\partial f}{\partial x}\right|_i = \frac{f_i - f_{i-1}}{\Delta_x} + \mathcal{O}(\Delta_x)$$

A more accurate, centered discretization for a 1st order derivative is obtained by subtracting the two expansions.

3 point: (centered)
$$\left.\frac{\partial f}{\partial x}\right|_i = \frac{f_{i+1} - f_{i-1}}{2\Delta_x} + \mathcal{O}(\Delta_x^2)$$

- More accuracy generally will require the use of more function points

Discrete Numerical Operations: Derivatives (2)

The expansions can be relabeled (i -> i+1, etc.) and the resulting set of equations can be manipulated to obtain 5-point and other higher-order forms with higher accuracy:

5 point: (centered)

$$\left.\frac{\partial f}{\partial x}\right|_i = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12\Delta_x} + \mathcal{O}(\Delta_x^4)$$

Still higher order, and more accurate, forms are possible but rapidly become cumbersome and require more points.

Similar methods can be employed to obtain discretizations of higher order derivatives. For example,

3 point: (centered)

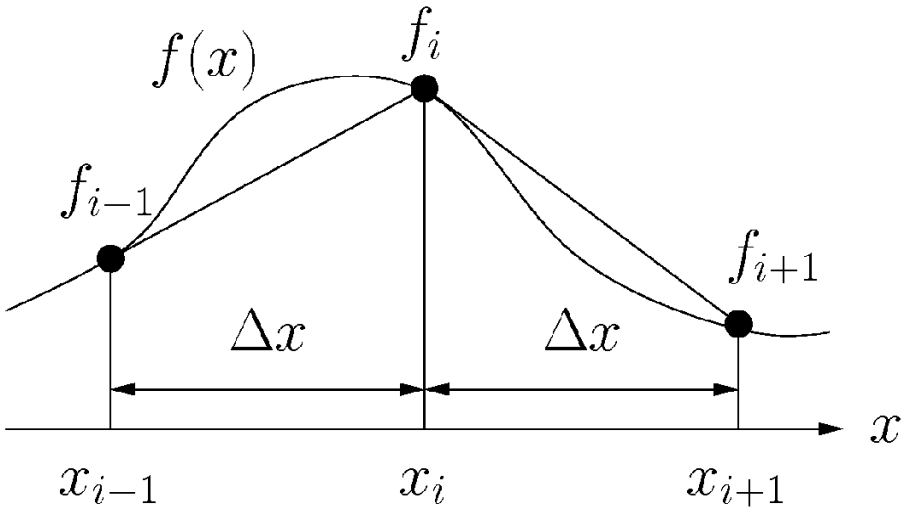
$$\left.\frac{\partial^2 f}{\partial x^2}\right|_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta_x^2} + \mathcal{O}(\Delta_x^2)$$

Discrete Numerical Operations: Integrals/Quadrature

Take n_x even, then $\int_{x_{min}}^{x_{max}} d\tilde{x}\, f(\tilde{x})$ can be composed as sub-integrals of the form

$$\int_{x_{i-1}}^{x_{i+1}} d\tilde{x}\, f(\tilde{x})$$

Using a linear approximation (Trapezoidal Rule):



$$\int_{x_{i-1}}^{x_{i+1}} dx\, f(x) = \frac{f_{i-1} + 2f_i + f_{i+1}}{2}\Delta_x + \mathcal{O}(\Delta_x^3)$$

Discrete Numerical Operations: Integrals/Quadrature (2)

Better approximations can be found (e.g., Simpson's Rule) using Taylor series expansions and the previous discrete derivatives:

$$f(x) = f_i + \frac{f_{i+1} - f_{i-1}}{2\Delta_x} x + \frac{f_{i+1} - f_i + f_{i-1}}{\Delta_x^2} x^2 + \ldots$$

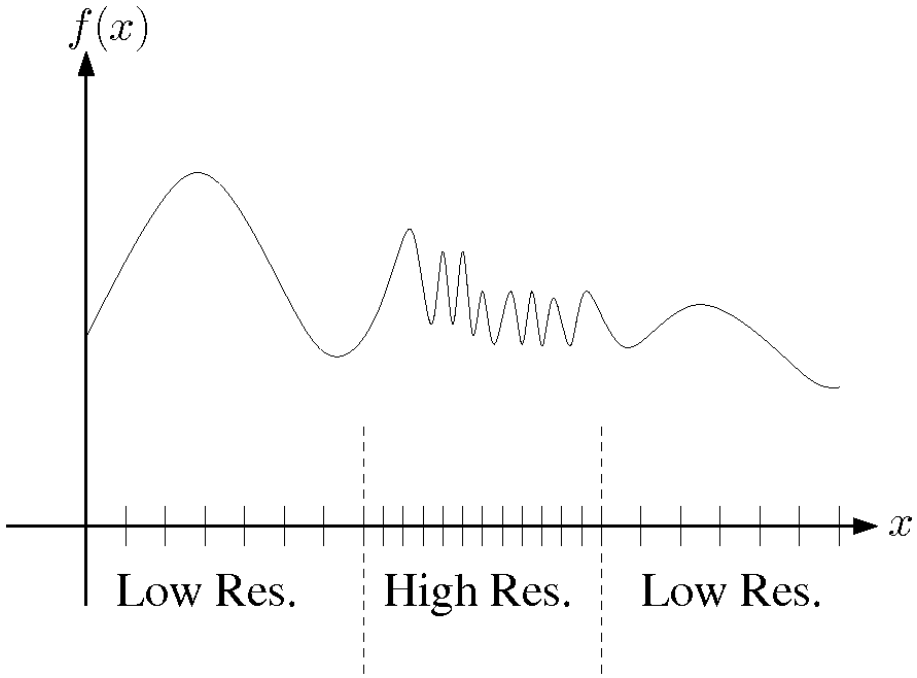
giving:

$$\int_{x_{i-1}}^{x_{i+1}} dx\, f(x) = \frac{f_{i-1} + 4f_i + f_{i+1}}{3} \Delta_x + \mathcal{O}(\Delta_x^5)$$

In the examples given, uniform grids have been employed and the formulas presented for derivatives and integrals are readily generalized to multiple dimensions.

Discrete Numerical Operations: Irregular Grids

Nonuniform grids can be used to effectively concentrate resolution where it is needed



- Can be used most effectively when high resolution is needed only in limited regions and simulation domains are large
- Nonuniform grids make discretized formulas more complicated, particularly with respect to ordering errors
  - A simple example of nonuniform derivative calculation is included in the homework to illustrate methods

Discrete Numerical Operations: Axisymmetric Systems

To be added: Slide to discuss how to solve cylindrically symmetric problems pointing out origin problems. Suggest that it is often better to simply do in 2D *x*-*y* geometry and use conserved angular momentum.

S3C: Numerical Solution of Moment Methods – Time Advance

We now have the tools to numerically solve moment methods. The moment equations may always be written as an N-dimensional set of coupled 1st order ODEs (see: S2C and S.M. Lund lectures on Transverse Envelope Equations):

$$\mathbf{M} = \left(\langle x \rangle_\perp, \cdots, \langle x^2 \rangle_\perp, \cdots \right) \qquad \text{N-dim vector of moments}$$

$$\frac{d\mathbf{M}}{ds} = \mathbf{F}(\mathbf{M}, s) \qquad \text{vector equation of motion}$$

- Methods developed to advance moments can also be used for advances in particle and distribution methods

/// Example: Axisymmetric envelope equation for a continuously focused beam

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

$$R = 2\sqrt{\langle x^2 \rangle_\perp}$$

$$\frac{d}{ds}\begin{bmatrix} R \\ R' \end{bmatrix} = \begin{bmatrix} R' \\ -k_{\beta 0}^2 R + \frac{Q}{R} + \frac{\varepsilon_x^2}{R^3} \end{bmatrix}$$

$k_{\beta 0}^2,\ Q,\ \varepsilon_x^2,$ constants

///

S3C: Numerical Solution of Moment Methods – Euler Advance

Euler's Method:

Apply the forward difference formula

$$\left.\frac{d\mathbf{M}}{ds}\right|_i = \frac{\mathbf{M}_{i+1} - \mathbf{M}_i}{\Delta_s} + \mathcal{O}(\Delta_s) = \mathbf{F}(\mathbf{M}_i, s_i)$$

Rearrange to obtain 1st order Euler advance:

$$\mathbf{M}_{i+1} = \mathbf{M}_i + \mathbf{F}(\mathbf{M}_i, s_i)\Delta_s + \mathcal{O}(\Delta_s^2)$$

- Moments advanced in discrete steps in s from initial values

Note that N_s steps will lead to a total error

$$\text{error} \sim N_s \cdot \mathcal{O}(\Delta_s^2) \sim \frac{s_{max} - s_{min}}{\Delta_s}\mathcal{O}(\Delta_s^2) \sim \mathcal{O}(\Delta_s)$$

- Error decreases only linearly with step size
- Numerical work for each step is only one evaluation of **F**

S3C: Numerical Solution of Moment Methods – Order Advance

Definition:

A discrete advance with error $\mathcal{O}(\Delta_s^n)$ is an (n-1)th order method

- Euler's method is a 1st order method
- Higher order methods are generally used for ODE's in moment methods
  - Cheap to evaluate **F**
- Low order methods are generally used for particle and distribution methods
  - Expensive to evaluate **F**

S3C: Numerical Solution of Moment Methods – Runge-Kutta Advance

Runge-Kutta Method:

Integrate from s_i to s_{i+1}:

$$\frac{d\mathbf{M}}{ds} = \mathbf{F}(\mathbf{M}, s)$$

$$\mathbf{M}_{i+1} = \mathbf{M}_i + \int_{s_i}^{s_{i+1}} ds\, \mathbf{F}(\mathbf{M}, s)$$

Approximate $\mathbf{F}$ with a Taylor expansion through the midpoint of the step, $s_{i=1/2}$

$$\mathbf{F}(\mathbf{M}, s) = \mathbf{F}(\mathbf{M}_{i+1/2}, s_{i+1/2}) + \left.\frac{\partial \mathbf{F}}{\partial s}\right|_{s_{i+1/2}} (s - s_{i-1/2}) \quad + \ldots$$

The linear term integrates to zero, leaving

$$\Rightarrow \mathbf{M}_{i+1} = \mathbf{M}_i + \mathbf{F}(\mathbf{M}_{i+1/2}, s_{i+1/2}) \cdot \Delta_s + \mathcal{O}(\Delta_s^3)$$

Runge-Kutta Advance (2)

Note: only need $\mathbf{M}_{i+1/2}$ to $\mathcal{O}(\Delta_s^2)$ for $\mathcal{O}(\Delta_s^3)$ accuracy
Apply Euler's method for the two-step procedure:

2nd Order Runge-Kutta Method:

Step 1: $$\mathbf{K} = \mathbf{F}(\mathbf{M}_i, s_i)\Delta_s$$

Step 2: $$\mathbf{M}_{i+1} = \mathbf{M}_i + \mathbf{F}\left(\mathbf{M}_i + \frac{\mathbf{K}}{2}, s_i + \frac{\Delta_s}{2}\right)\Delta_s + \mathcal{O}(\Delta_s^3)$$

- Requires two evaluations of **F** per advance
- 2nd order accurate in Δ_s

Higher order Runge-Kutta schemes are derived analogously from various quadrature formulas. Such formulas are found in standard numerical methods texts

- Typically, methods with error $\mathcal{O}(\Delta_s^{N+1})$ will require N evaluations of **F**

S3C: Numerical Solutions of Moment Methods

Many methods are employed to advance moments and particle orbits.

A general survey of these methods is beyond the scope of this lecture. But some general comments can be made:

- Many higher-order methods with adaptive step sizes exist that refine accuracy to specified tolerances and are optimized for specific classes of equations
- Choice of numerical method often relates to numerical work and stability considerations
- Certain methods can be formulated to exactly preserve relevant single-particle invariants
  - "Symplectic" methods preserve Hamiltonian structure of dynamics
- Accelerator problems can be demanding due to multiple frequency scales and long tracking times/distances
  - Hamiltonian dynamics; phase space volume does not decay

S3C: Numerical Solutions of Moment Methods – Numerical Stability

"Numerical Reversibility" test of stability:

In this method, the final value of an advance is used as an initial condition. Then the problem is run backwards to the original starting point and deviations from the initial conditions taken in the original advance are analyzed.

- Often a simple, but stringent test of accuracy
- Will ultimately fail due to roundoff errors and cases where there is a sensitive dependence on initial conditions
- Orbits can be wrong but qualitatively right. We will quantify this notion better later. So lack of full convergence does not necessarily mean that useless results will be obtained.

We will now briefly overview an application of moment equations, namely the KV envelope equations, to a practical high current transport lattice that was designed for Heavy Ion Fusion applications at Lawrence Berkeley National Laboratory.

S3C: Moment Equation Application: Perp. KV Envelope Eqns

Neglect image charges and nonlinear self-fields (emittance constant) to obtain moment equations for the evolution of the beam envelope radii

$$\frac{d^2 r_x}{ds^2} + \kappa_q r_x - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_x^2}{r_x^3} = 0 \qquad r_x = 2\sqrt{\langle x^2 \rangle_\perp}$$

$$\frac{d^2 r_y}{ds^2} - \kappa_q r_y - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_y^2}{r_y^3} = 0 \qquad r_y = 2\sqrt{\langle y^2 \rangle_\perp}$$

$$Q = \frac{qI}{2\pi\epsilon_0 m c^3 \gamma_b^3 \beta_b^3}$$

Dimensionless Perveance
measures space-charge strength

$$\varepsilon_x = 4\left[\langle x^2 \rangle_\perp \langle x'^2 \rangle_\perp - \langle x x' \rangle_\perp^2\right]^{1/2}$$

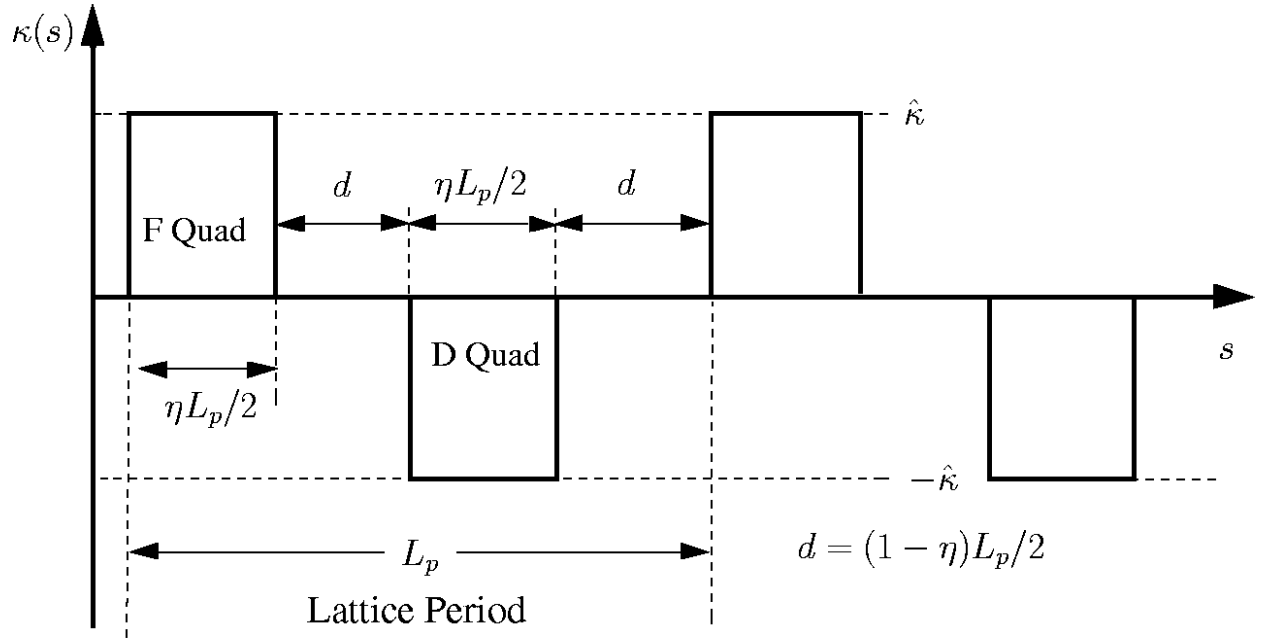
$(\varepsilon_{xn} = \gamma_b \beta_b \varepsilon_x$ normalized)

RMS Edge Emittance
measures x-x' phase-space area
~(beam size)sqrt(thermal temp.)

The matched beam solution together with parametric constraints from engineering, higher-order theory, and simulations are used to design the lattice.

Application Example Continued (2) – Focusing Lattice

Take an alternating gradient FODO doublet lattice



$$d = (1-\eta)\frac{L_p}{2} \qquad \eta = \text{Quadrupole Occupancy } (0 < \eta \leq 1)$$

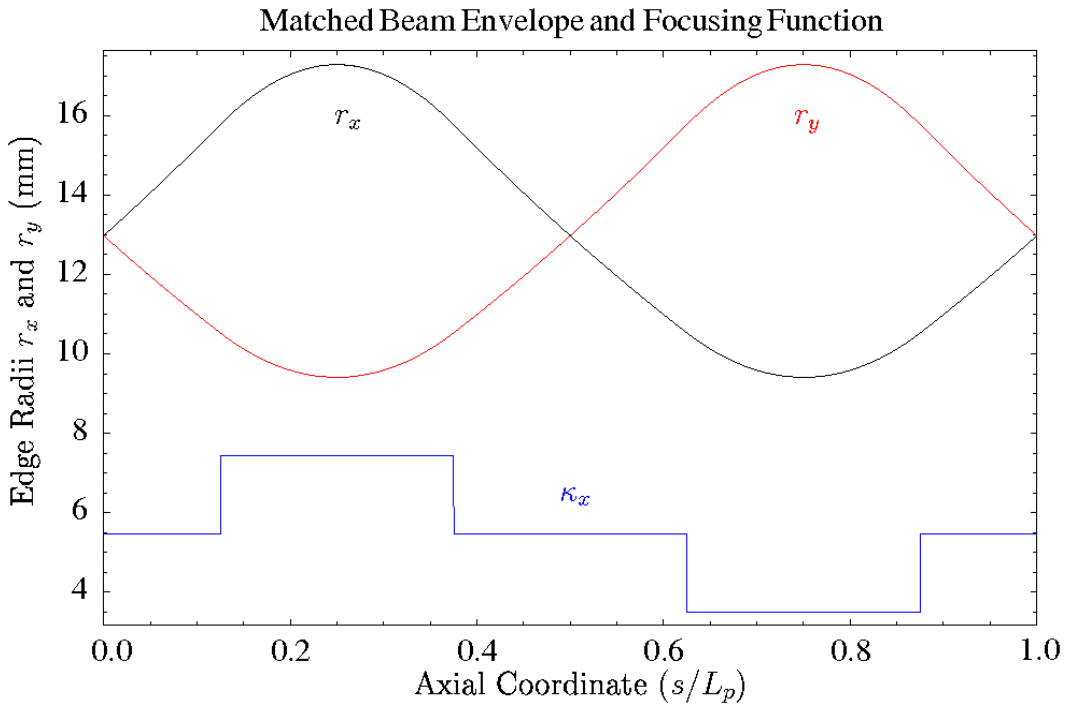
Focusing Strength

$$\hat{\kappa} = \begin{cases} \dfrac{1}{[B\rho]}\left|\dfrac{dB_x}{dy}\right|, & \text{Magnetic Quadrupole} \\ \dfrac{1}{[B\rho]\beta_b c}\left|\dfrac{dE_x}{dy}\right|, & \text{Electric Quadrupole} \end{cases}$$

Rigidity

$$[B\rho] = m\gamma_b\beta_b c/q$$

Application Example Contd. (3) – Matched Envelope Properties



Ion

K^+, E = 2 MeV

Current

I = 800 mA

Lattice

$L_p = 0.5\text{m}$

$\eta = 1/2$

$\sigma_0 = 80^\circ/\text{Lattice Period}$

Envelope Properties:

1) Low Emittance Case: $\varepsilon_x = \varepsilon_y = 50$ mm-mrad; $\sigma = 9.42^\circ/\text{Lattice Period}$

$\text{Max}[r_x] = \text{Max}[r_y] = 17.3$ mm $\quad$ $\text{Max}[r_x'] = -\text{Min}[r_y'] = 47.5$ mrad

$\text{Min}[r_x] = \text{Min}[r_y] = 9.41$ mm $\quad$ $\text{Max}[r_y'] = -\text{Min}[r_x'] = 47.5$ mrad

2) High Emittance Case: $\varepsilon_x = \varepsilon_y = 200$ mm-mrad; $\sigma = 32.13^\circ/\text{Lattice Period}$

$\text{Max}[r_x] = \text{Max}[r_y] = 18.9$ mm $\quad$ $\text{Max}[r_x'] = -\text{Min}[r_y'] = 52.4$ mrad

$\text{Min}[r_x] = \text{Min}[r_y] = 10.1$ mm $\quad$ $\text{Max}[r_y'] = -\text{Min}[r_x'] = 52.4$ mrad

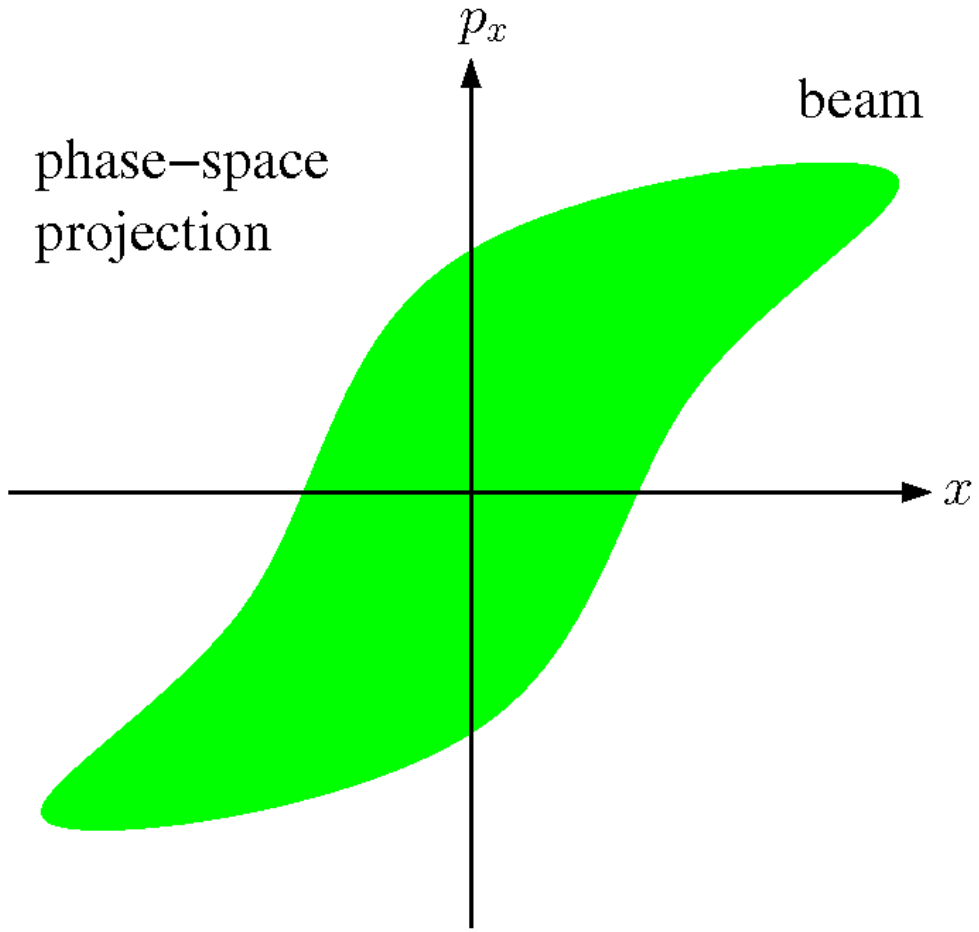
S4: Numerical Solution of Particle and Distribution Methods

S4A: 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

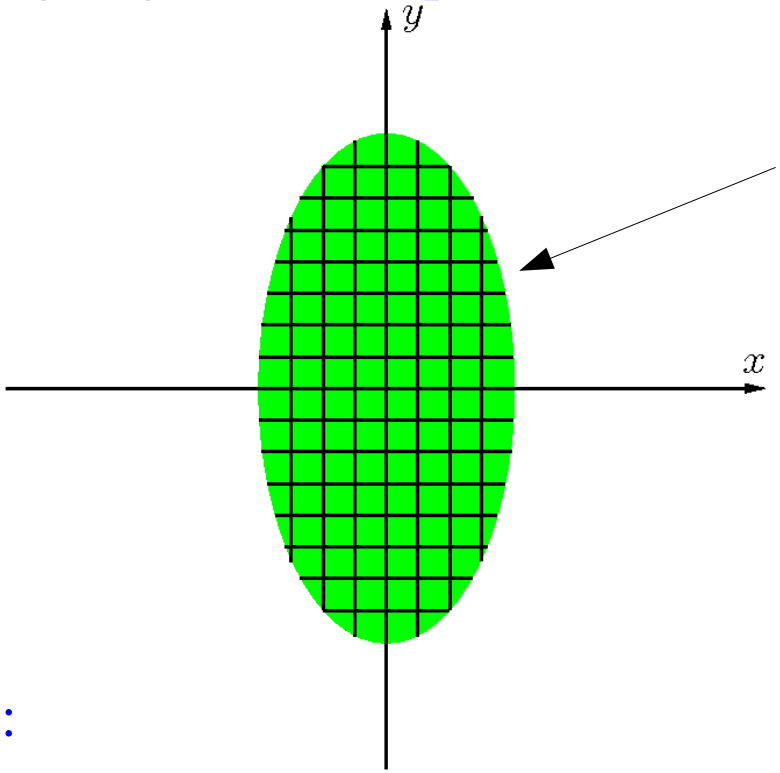
$\{\mathbf{x}_i, \mathbf{p}_i\}$

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

Numerical Solution of Particle and Distribution Methods (2)

Represent the beam by Lagrangian "macroparticles" advanced in time



Macroparticle Properties:

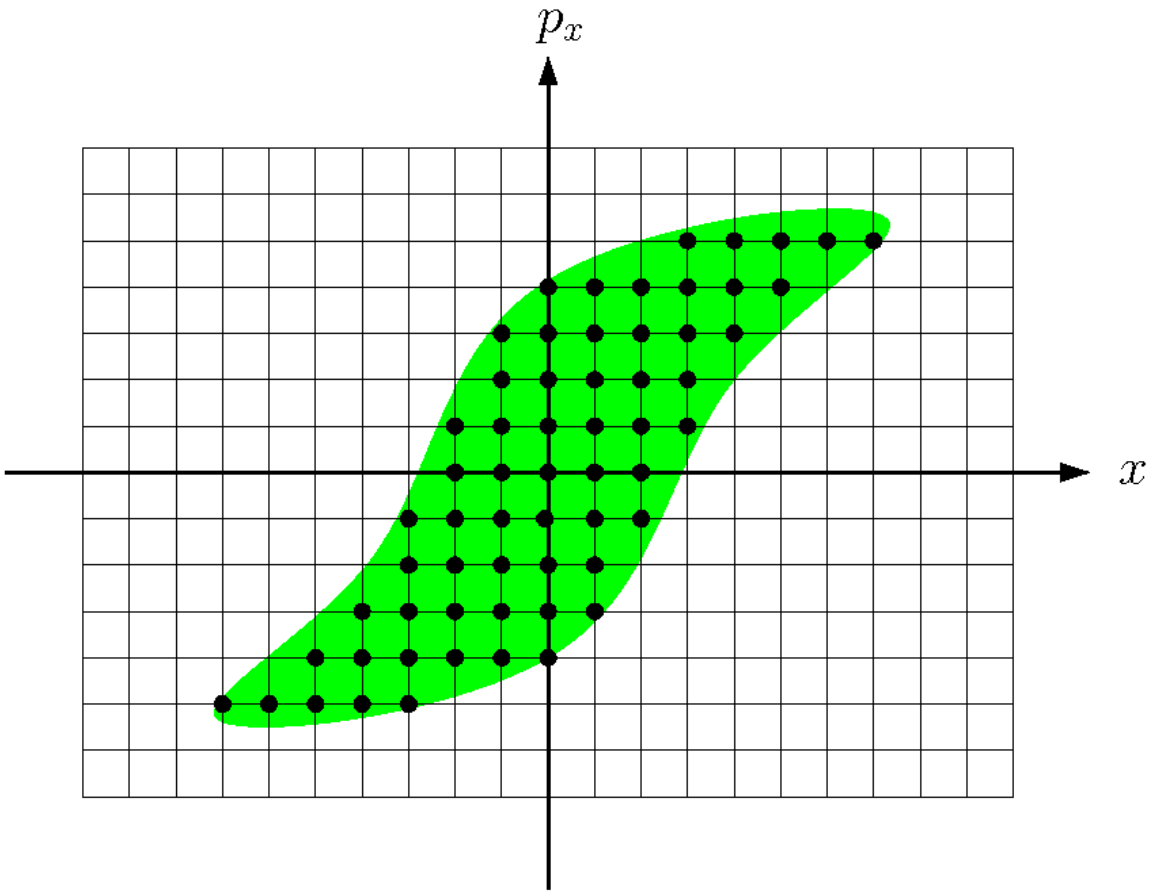
- Same *q/m* ratio as real particle
  - Gives same single particle dynamics in the applied field
- More collisions due to macroparticles having more close approaches
  - Enhanced collisionality is unphysical
  - Controlled by smoothing the macroparticle interaction with the self-field. More on this later.

Numerical Solution of Particle and Distribution Methods (3)

Direct Vlasov as an example:

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

Advance distribution $f(\boldsymbol{x},\boldsymbol{p},\boldsymbol{t})$ 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

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

S4B: 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

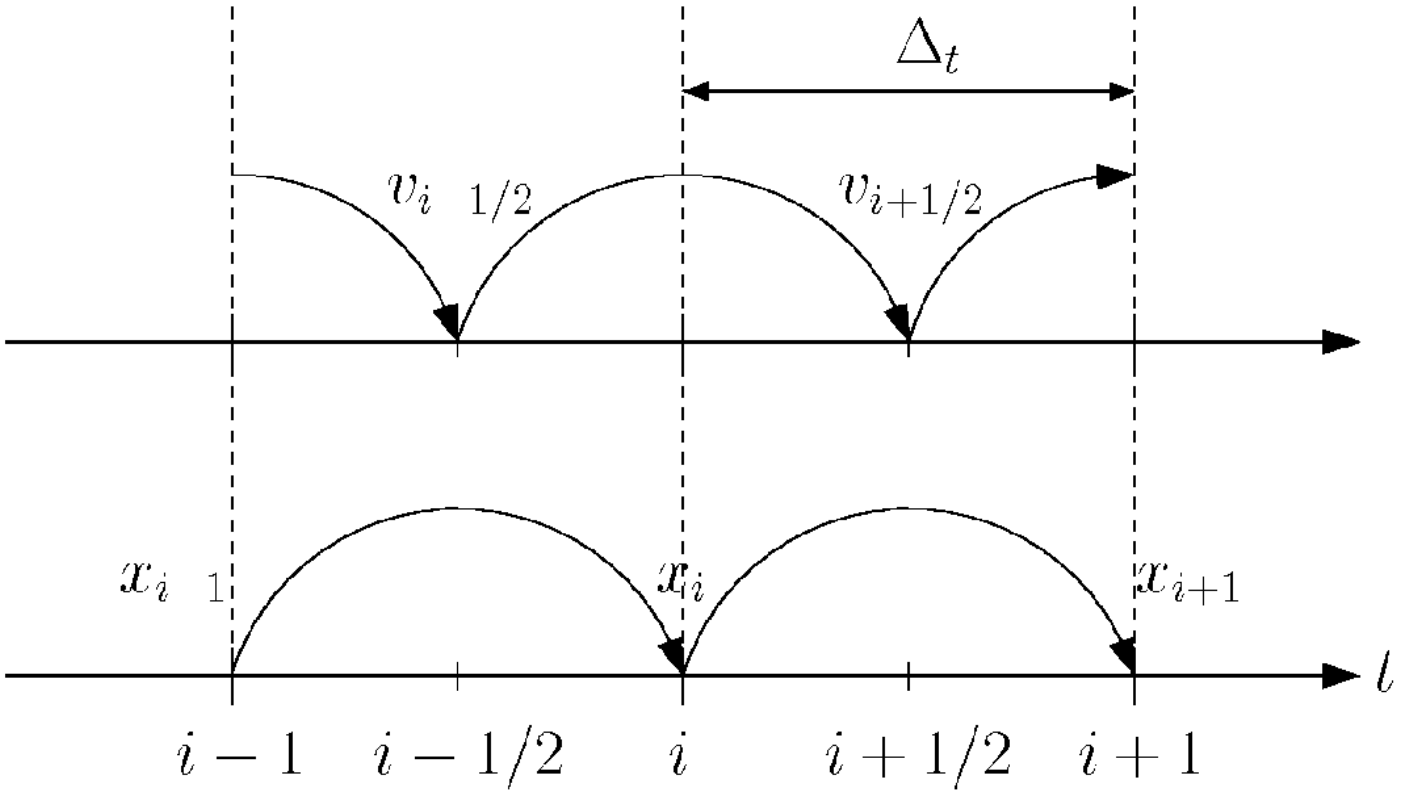
Leapfrog Advance (time centered): Advance velocity and position out of phase

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

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

Velocity:

Position:



Leapfrog Method: Order

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

$$i \to i+1 \;:\; \begin{cases} \dfrac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\Delta_t} = \mathbf{v}_{i+1/2} \\ \dfrac{\mathbf{x}_i - \mathbf{x}_{i-1}}{\Delta_t} = \mathbf{v}_{i-1/2} \end{cases}$$

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:

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

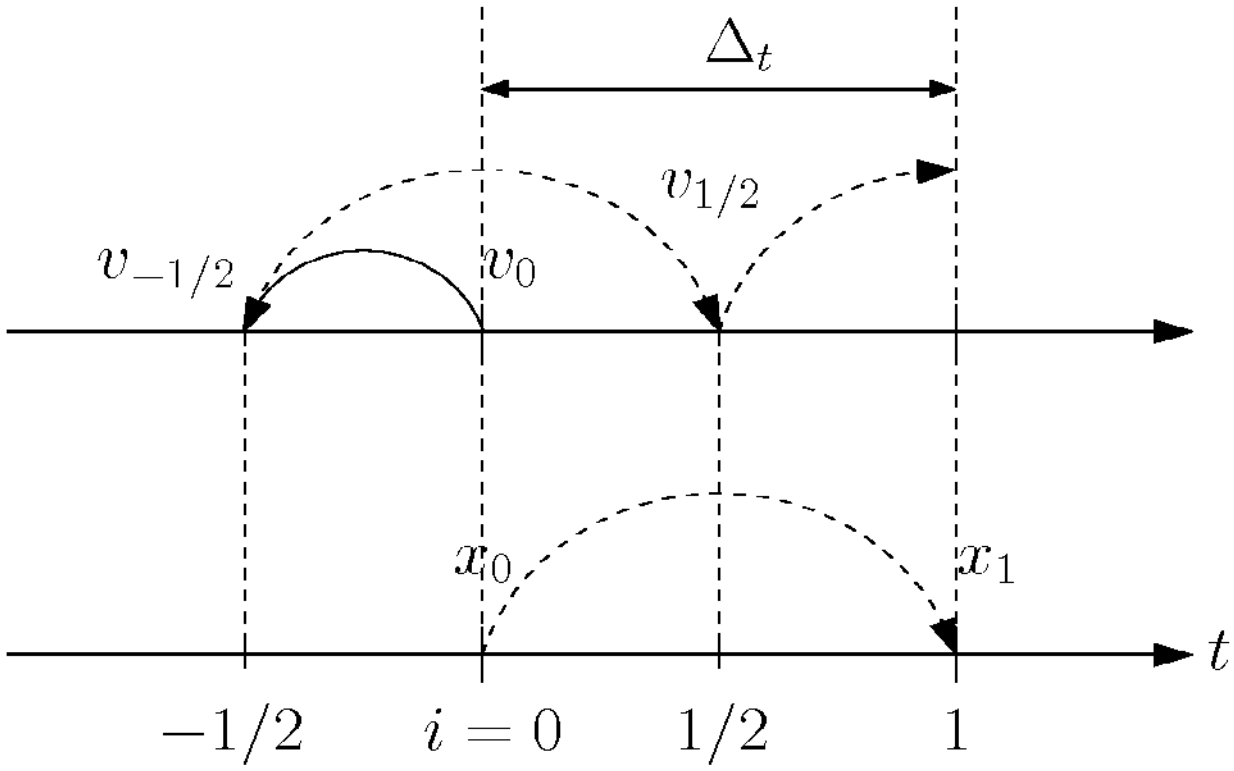
- $\mathbf{x}_{i+1}$ fixed from $\mathbf{x}_i,\ \mathbf{x}_{i-1},\ \mathbf{F}_i$ to $\mathcal{O}(\Delta_t^4)$
- Leapfrog method is 2nd order accurate

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: 3-step, time-centered

1) Half-step acceleration in electric field

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

2) Rotation in magnetic field. Here choose coordinates so that

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

$$\| \mathbf{B}_i \ : \quad v^{(2)}_{z_{i+1/2}} = v^{(1)}_{z_{i+1/2}}$$

$$\perp \mathbf{B}_i \ : \quad \begin{bmatrix} v^{(2)}_{z_{i+1/2}} \\ v^{(1)}_{z_{i+1/2}} \end{bmatrix} = \begin{bmatrix} \cos(\omega_{ci}\Delta_t) & \sin(\omega_{ci}\Delta_t) \\ -\sin(\omega_{ci}\Delta_t) & \cos(\omega_{ci}\Delta_t) \end{bmatrix} \begin{bmatrix} v^{(2)}_{z_{i+1/2}} \\ v^{(1)}_{z_{i+1/2}} \end{bmatrix}$$

3) Half-step acceleration in electric field

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

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:

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

Exact solution

Discretized equation of motion

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

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

$$\begin{aligned} \Rightarrow e^{j\tilde{\omega}\Delta_t} - 2 + e^{-j\tilde{\omega}\Delta_t} &= -\omega^2\Delta_t^2 \\ 2 - 2\cos(\tilde{\omega}\Delta_t) &= \omega^2\Delta_t^2 \\ \sin^2\left(\frac{\tilde{\omega}\Delta_t}{2}\right) &= \frac{\omega^2\Delta_t^2}{4} \\ \Rightarrow \sin\left(\frac{\tilde{\omega}\Delta_t}{2}\right) &= \frac{\omega\Delta_t}{2} \end{aligned}$$

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
$\Delta_t = 0.1\tau$	$24\pi/(0.1 \cdot 2\pi)^3 \approx 3 \times 10^2$
$\Delta_t = 0.01\tau$	$24\pi/(0.01 \cdot 2\pi)^3 \approx 3 \times 10^5$

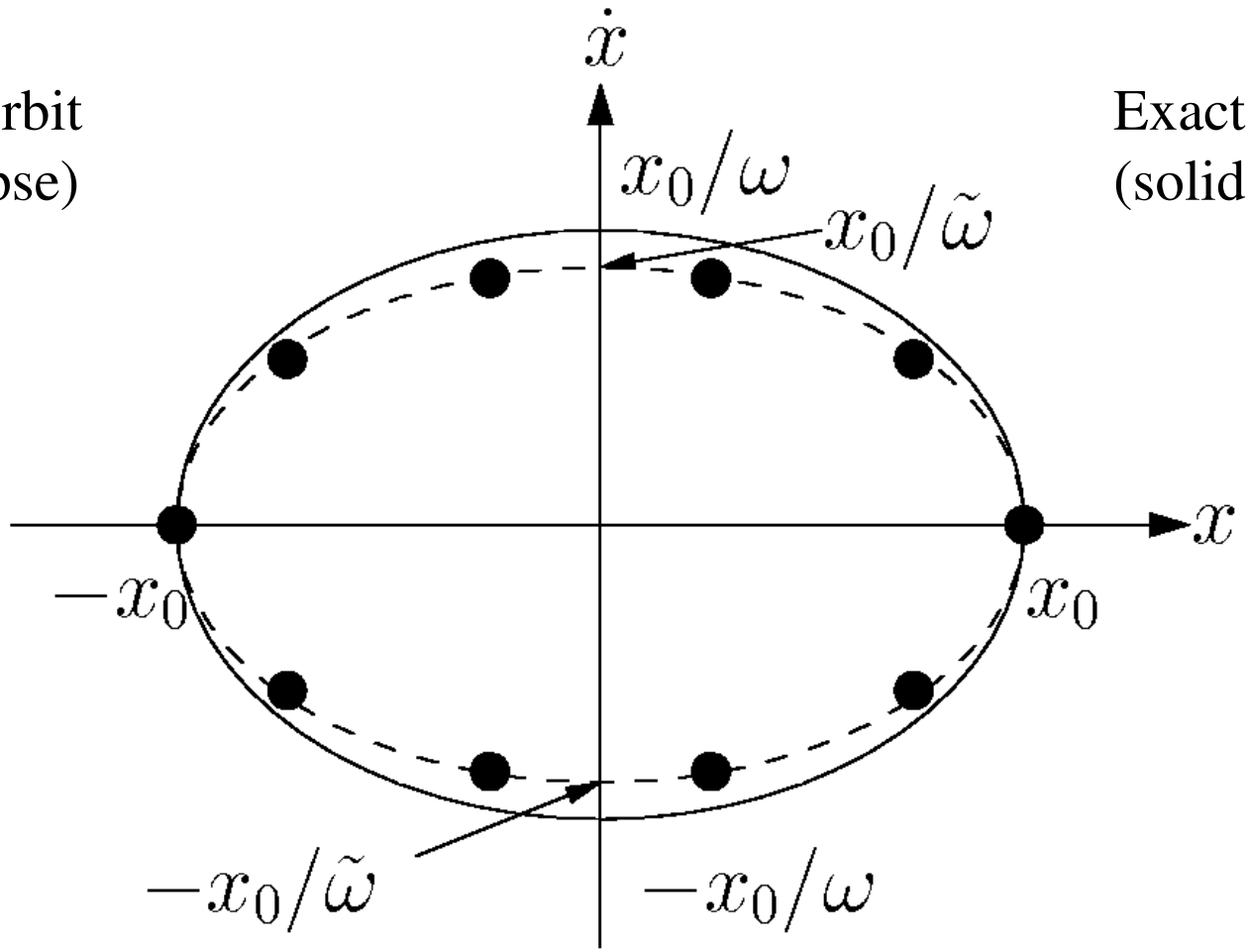
//

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

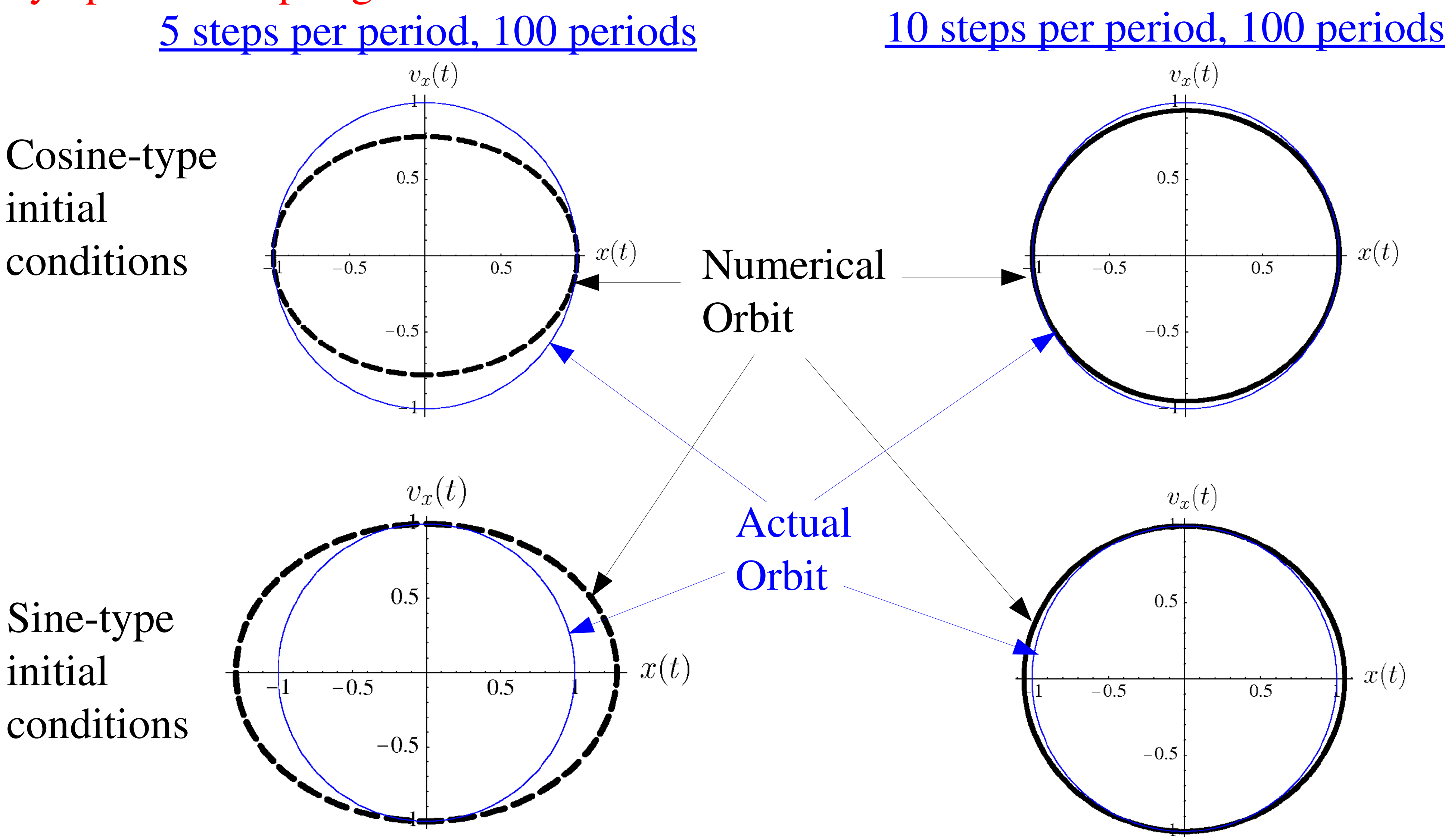
- More general analysis of the leapfrog method shows it has "symplectic" structure, meaning it preserves the Hamiltonian nature of the dynamics
- Symplectic methods are important for long tracking problems (typical in accelerators) to obtain the right orbit structure
  - Runge-Kutta methods are not symplectic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



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

Contrast: Numerical and Actual Orbit for a Simple Harmonic Oscillator

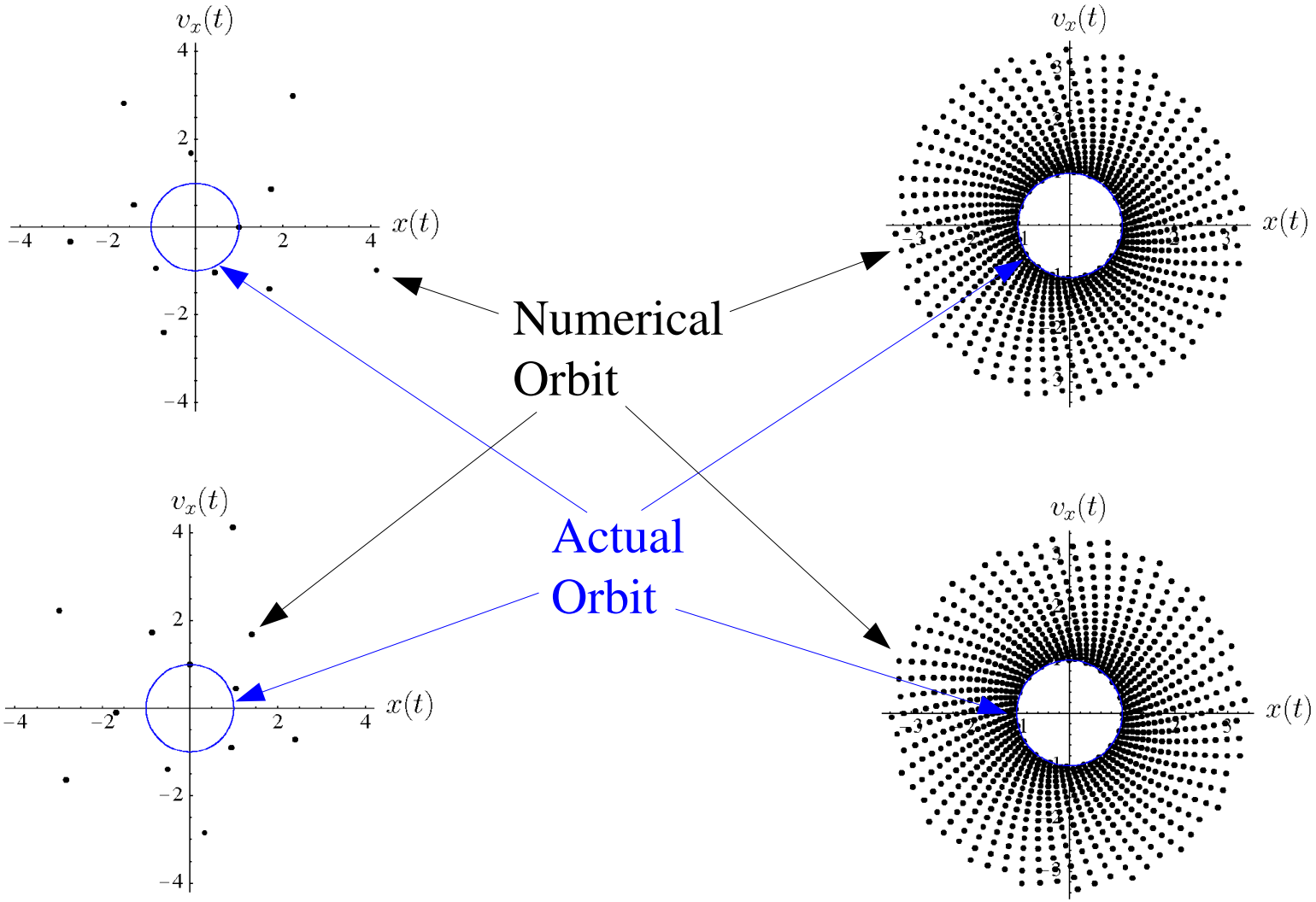
Non-Symplectic 2nd Order Runge-Kutta Advance: (see earlier notes on RK advance)

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

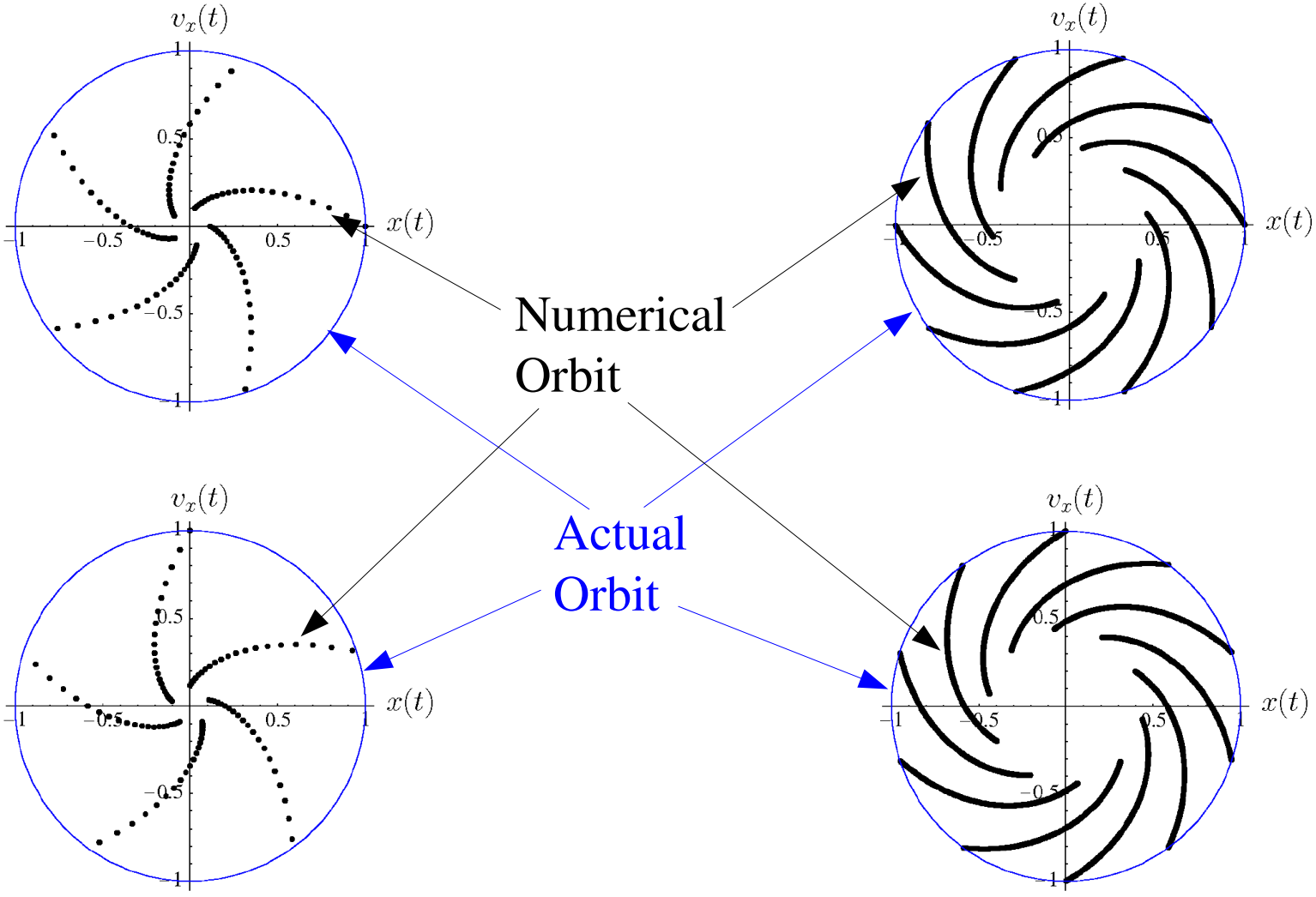
Non-Symplectic 4th Order Runge-Kutta Advance: (analog to notes on 2nd order RK adv)

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:

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

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

Se ...sed 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.

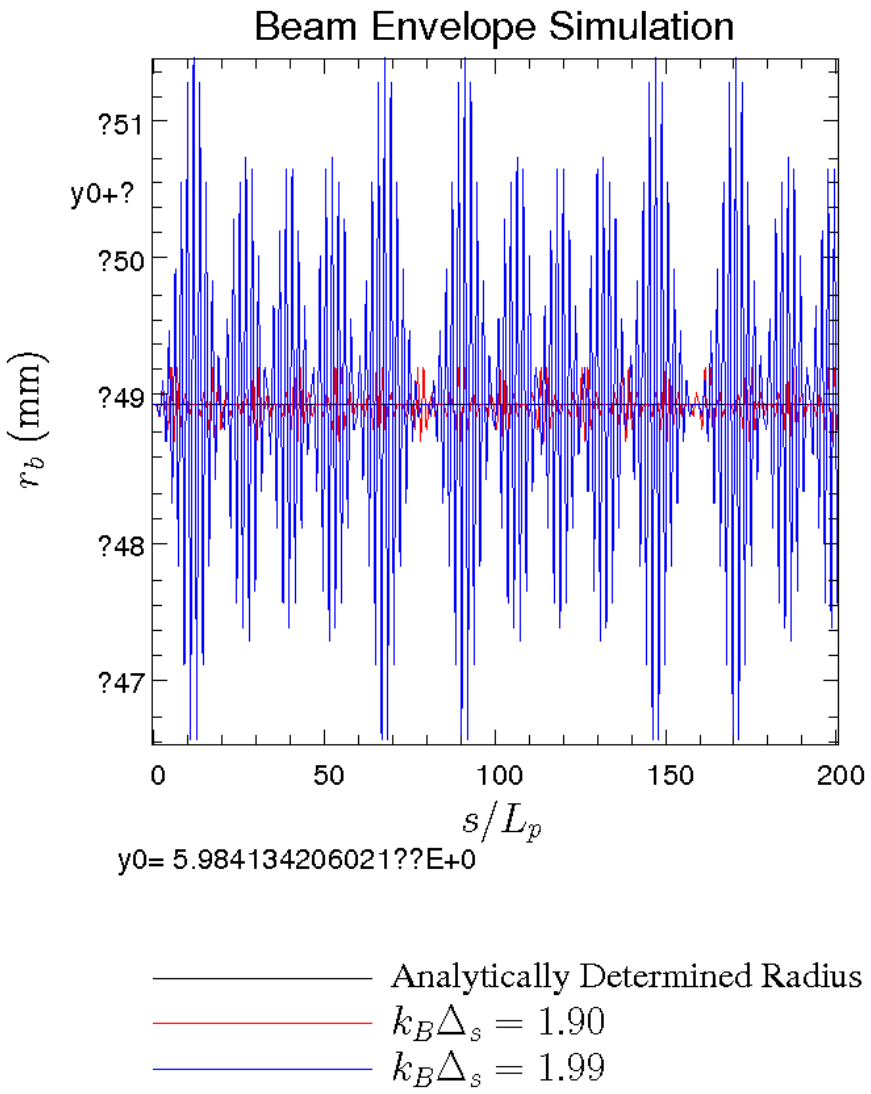
Numerical simulations for an initially matched envelope with: $\sigma/\sigma_0 = 1/2$

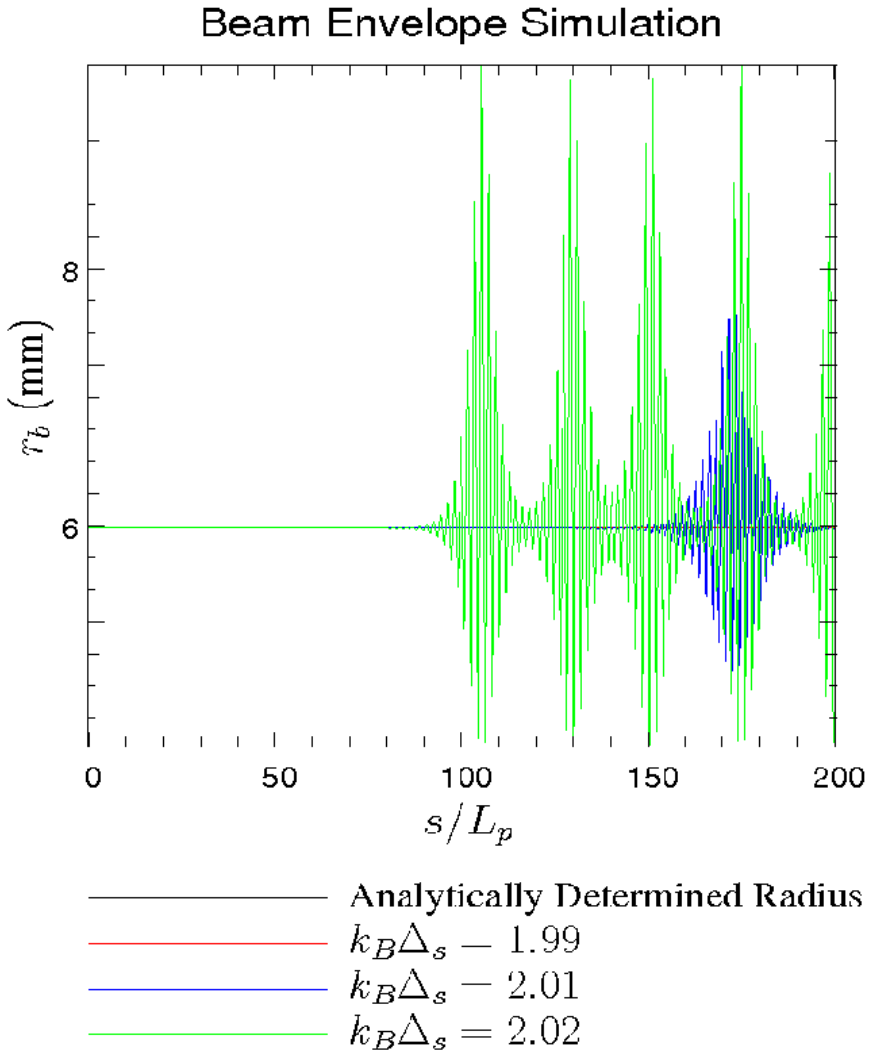
$k_\beta \Delta_s$	$k_{\beta 0}\Delta_s$	$k_Q \Delta_s$	$k_B \Delta_s$	Stable?
0.500	1.00	1.32	1.58	Yes
0.600	1.20	1.59	1.90	Yes
0.630	1.26	1.67	1.99	Yes
0.635	1.27	1.68	2.01	No
0.640	1.28	1.69	2.02	No

The highest k-mode, the breathing mode, appears to determine 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

S4C: Field Solution

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

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

- Techniques outlined here are also applicable to distribution methods

Fields can 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

- Sometimes 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
- Important to calculate with realistic boundary conditions

Electrostatic Field Solution

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

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

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

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

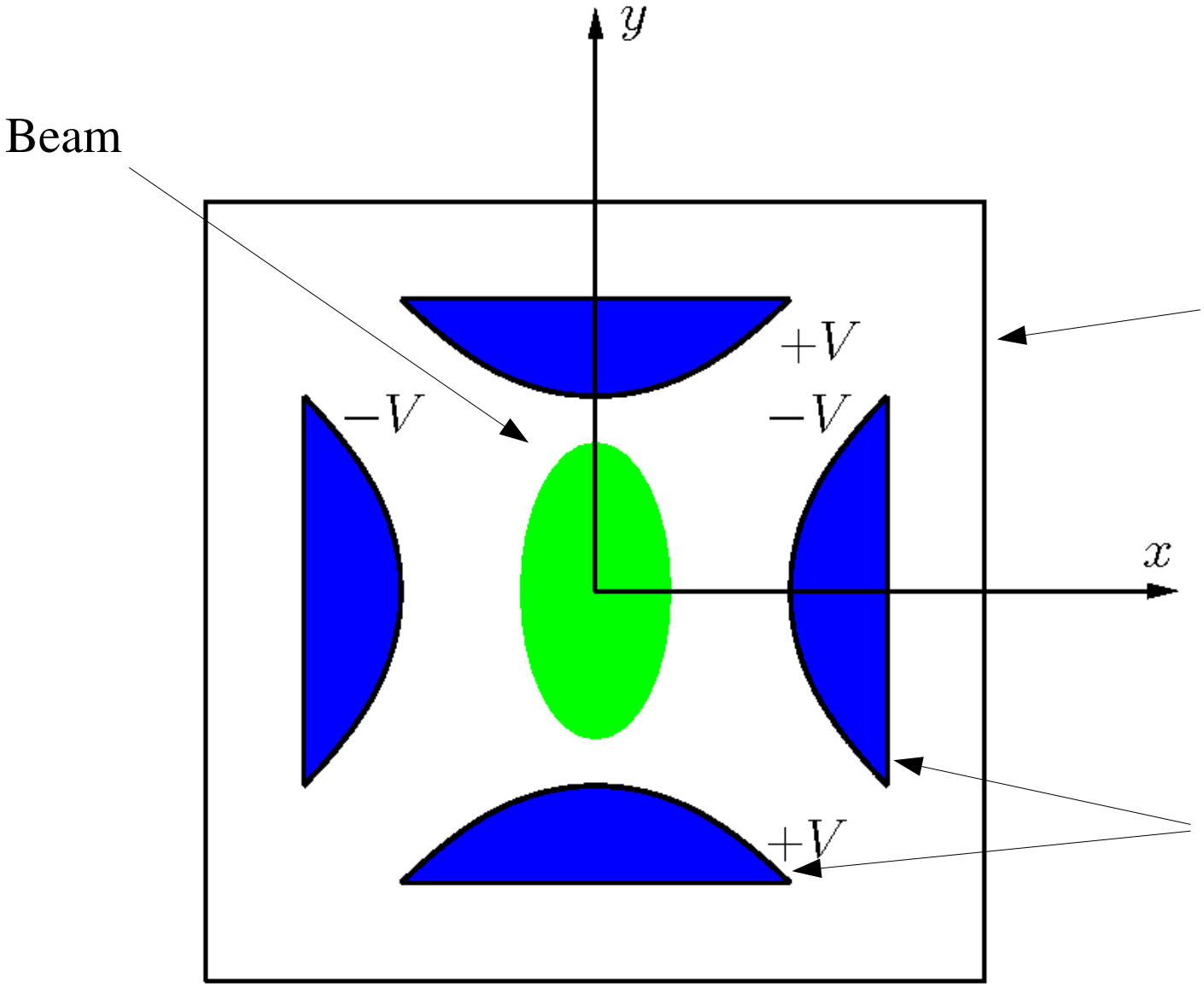
$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \quad + \text{ boundary conditions on } \mathbf{E}$$
$$\nabla \times \mathbf{E} = 0$$

$\nabla \times \mathbf{E} = 0$ 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 $\frac{\partial \phi}{\partial n}$

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

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

Self-field component

$$\equiv \phi_s$$

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

Applied field from electrode potentials

$$\equiv \phi_a$$

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

Electrostatic Field Solution by Green's Function (2)

Applied Field:

ϕ_a can be calculated in advance and need not be recalculated if transverse geometry does not change

- Can be analytical in simple situations

Self Field:

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

N_p = Macro-particle number

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

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

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

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

Electrostatic Field Solution by Green's Function (3)

The Green's Function expression for ϕ_a 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, often rendering such methods inferior to gridded methods to be covered shortly

// 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 disretization 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

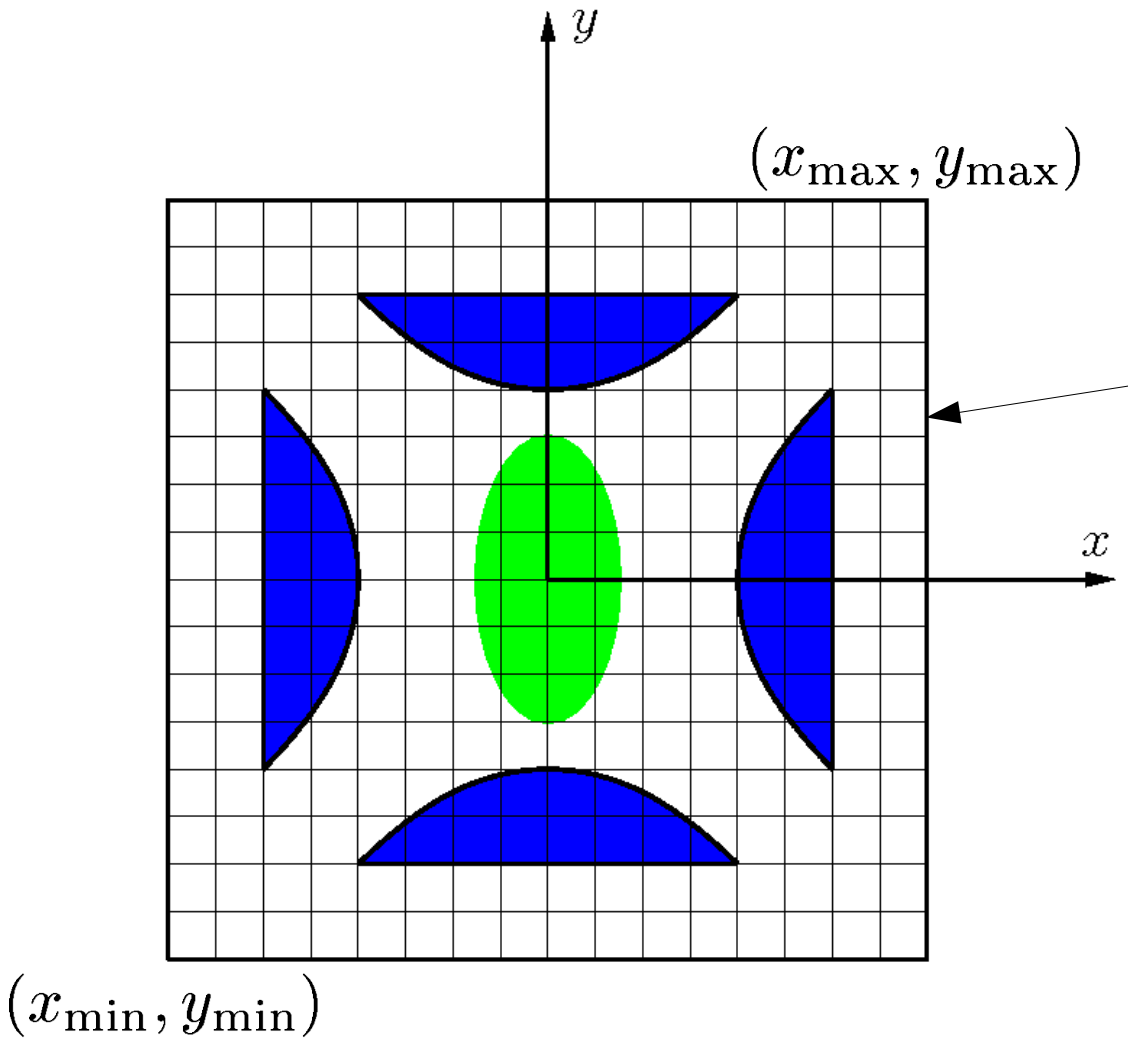
Comments:

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

These issues will be covered later under "particle weighting"

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

Beam in an electric quadrupole lattice (2D)



ϕ or $\dfrac{\partial \phi}{\partial n}$

specified on grid boundary and possibly on surfaces within the grid

Gridded Field Solution: Discretized Poisson Eqn.

For low order differencing, the Poisson Equation 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

$$E_{x\,i\,j} = \frac{\phi_{i+1,\,j} - \phi_{i-1,\,j}}{2\Delta_x}$$

$$E_{y\,i\,j} = \frac{\phi_{i,\,j+1} - \phi_{i,\,j-1}}{2\Delta_y}$$

Boundary conditions must also be incorporated as constraint equations

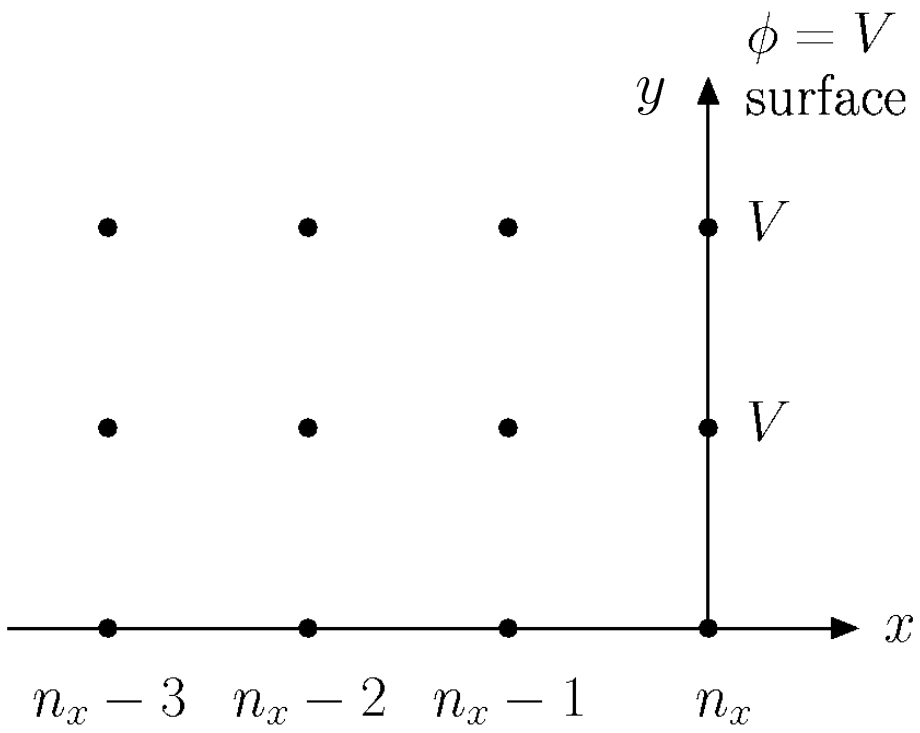
Dirichlet Conditions: ϕ specified on surfaces

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



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

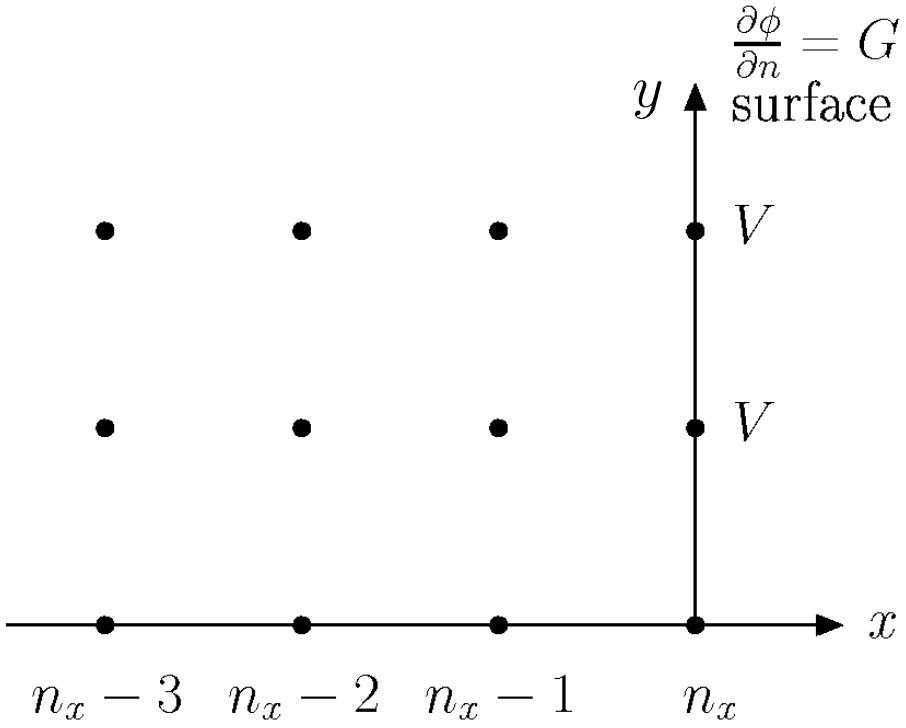
For $i = n_x - 1$ cells

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

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 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:

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

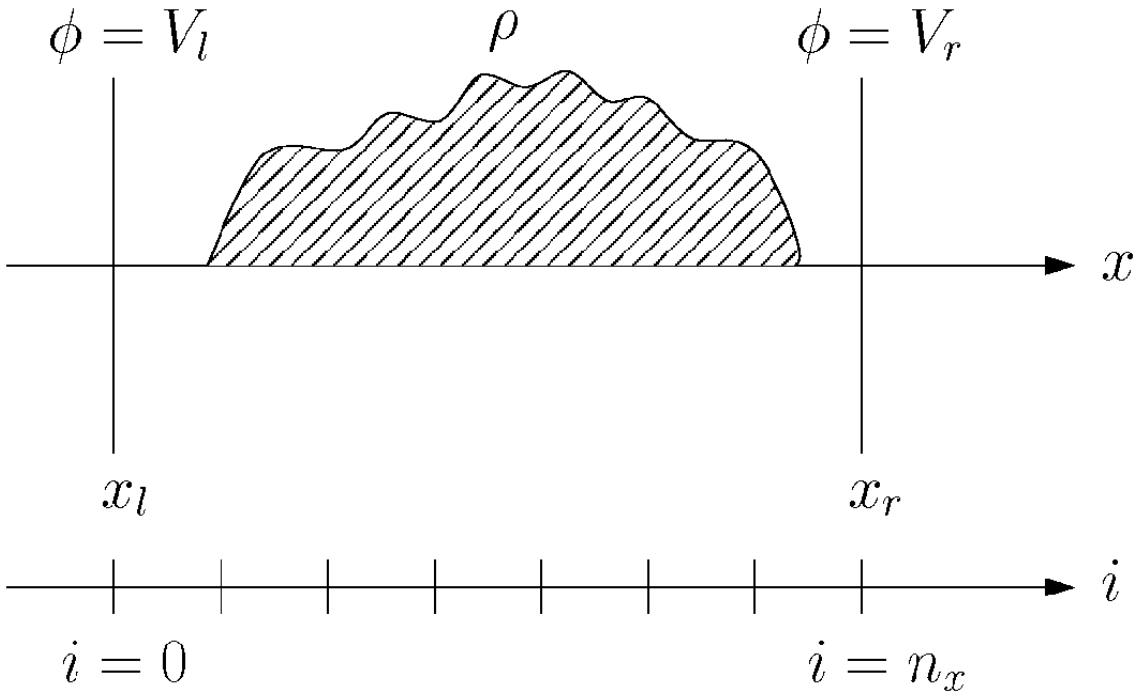
Direct inversion of $\overline{\mathbf{M}}^{-1}$ is not practical due to the large dimension of the problem

- $\overline{\mathbf{M}}$ 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:

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

$$\phi_0 = V_l$$

$$\phi_{n_x} = V_r$$

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

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 to find the ϕ_i

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

S4: Particle Methods – 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
  - 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 with numerical libraries
  - 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)

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

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

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

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)

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

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

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

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

Discrete Fourier Transform (3)

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

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

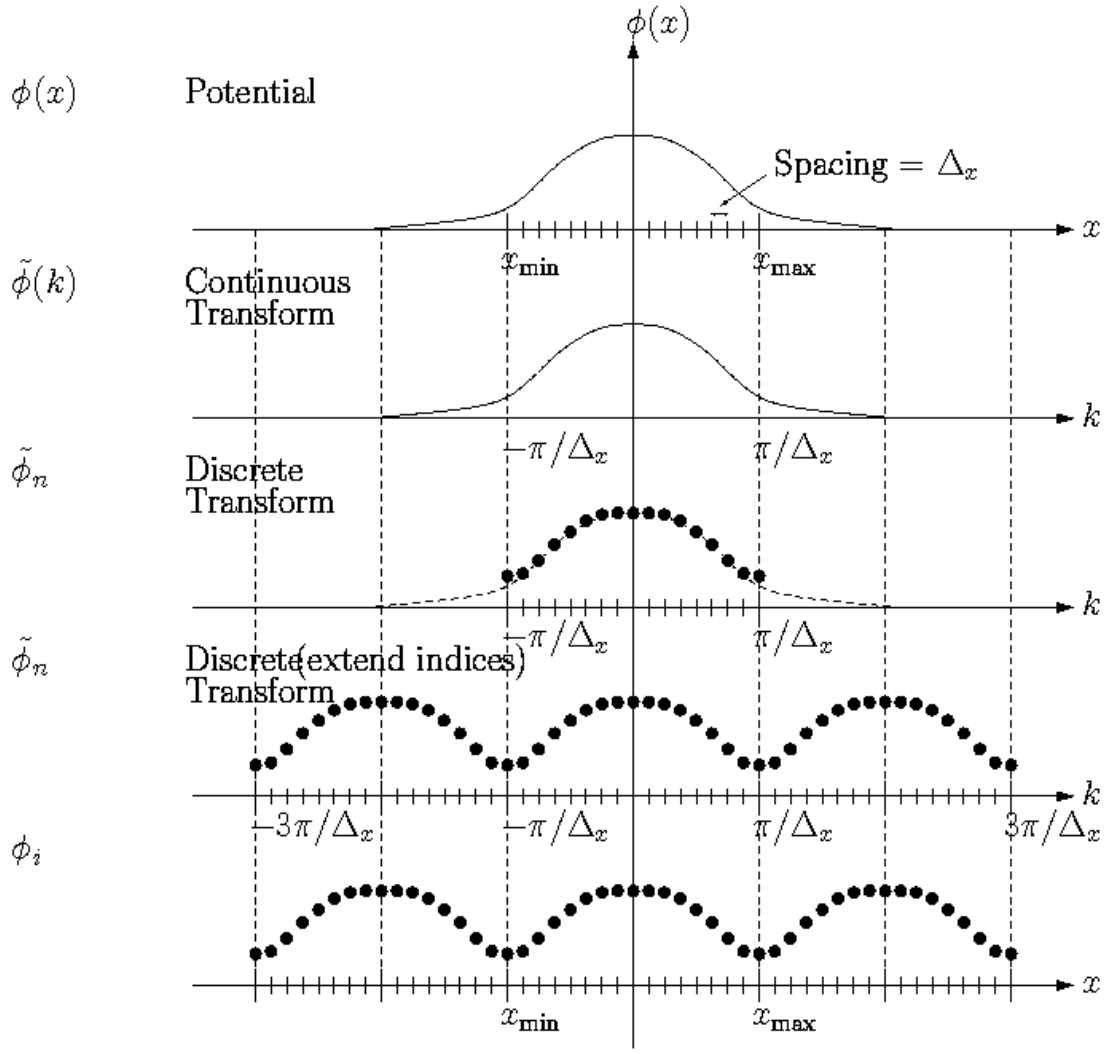
The discrete transform describes a periodic problem if indices are extended

- 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:

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

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

Applying the discrete transform yields:

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

Poisson's Equation becomes:

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

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.

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

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

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

$$= ik_n \mathrm{dif}(k_n\Delta_x)\tilde{\phi}_n$$

///

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		

- K_n^2 factors can be calculated once and stored to increase numerical efficiency

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 griding freedom
  - 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.

- This allows rapid field solutions in complicated geometries when capacity matrix elements can be pre-calculated and stored

FFT is the fastest method for simple geometry

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

S4D: Weighting: Depositing Particles on the Field Mesh and Interpolating Gridded Fields to Particles

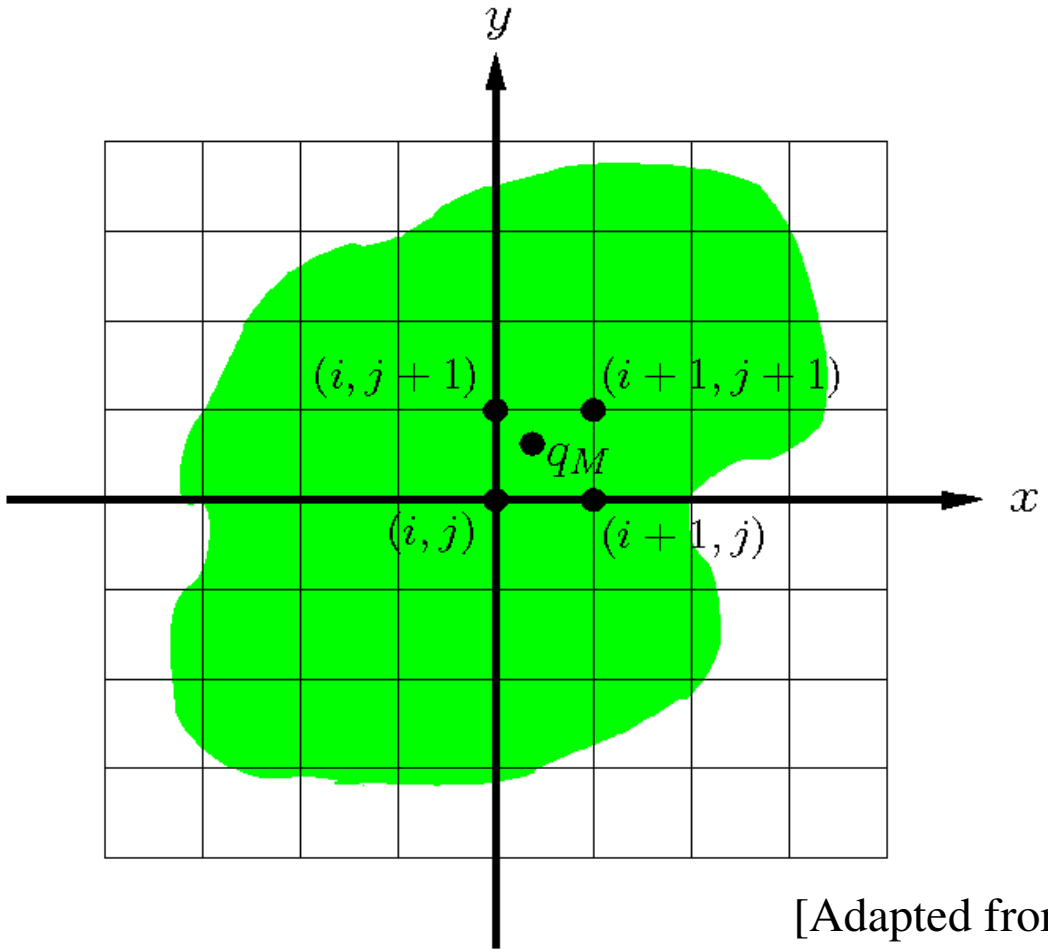
We have outlined methods to solve the electrostatic Maxwell's equations on a discrete spatial grid. To complete the description we must:

- Specify how to deposit macro-particle charges and current onto the grid
- Specify how to interpolate fields on the spatial grid points to the macroparticle coordinates (not generally on the grid) 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:

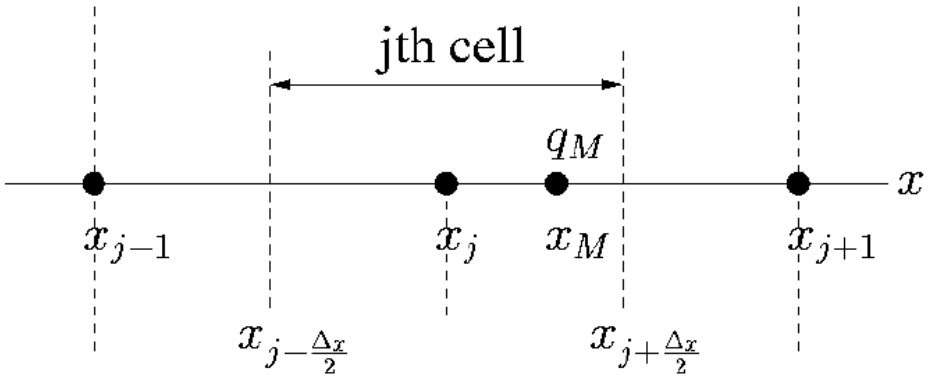
1) Nearest Grid Point
2) Cloud in Cell (CIC)
   - Shaped particles
   - PIC method, linearly shaped particles
3) Multipole
   - Dipole, subtracted dipole, etc.
4) Higher order methods
   - Splines
   - ***k***-space cutoffs in discrete transforms
   - ⋮

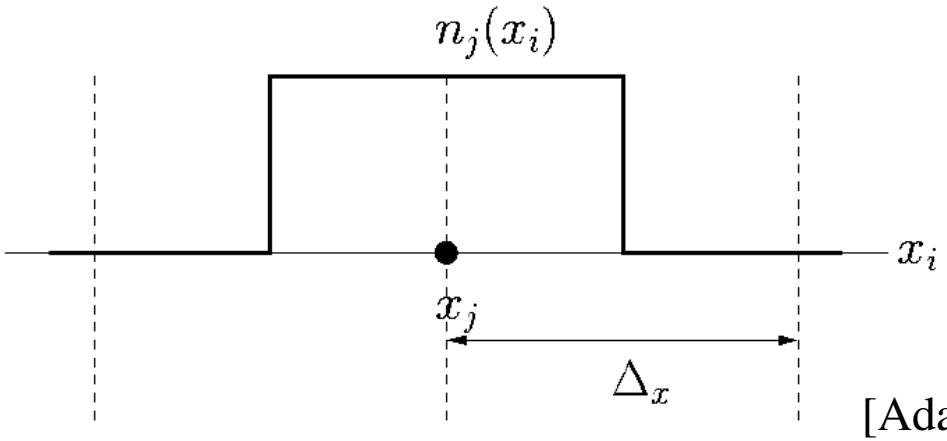
Possible hybrid methods also exist. We will illustrate methods 1) and 2) for electrostatic problems. Descriptions of other methods can be found in the literature.

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

$x_j =$ Closest grid cell

Charge Deposition:

$$q_j = q_M$$

Field "Interpolation":

$$E_x\big|_{x=x_M} = E_{xj}$$

[Adapted from Birdsall and Langdon]

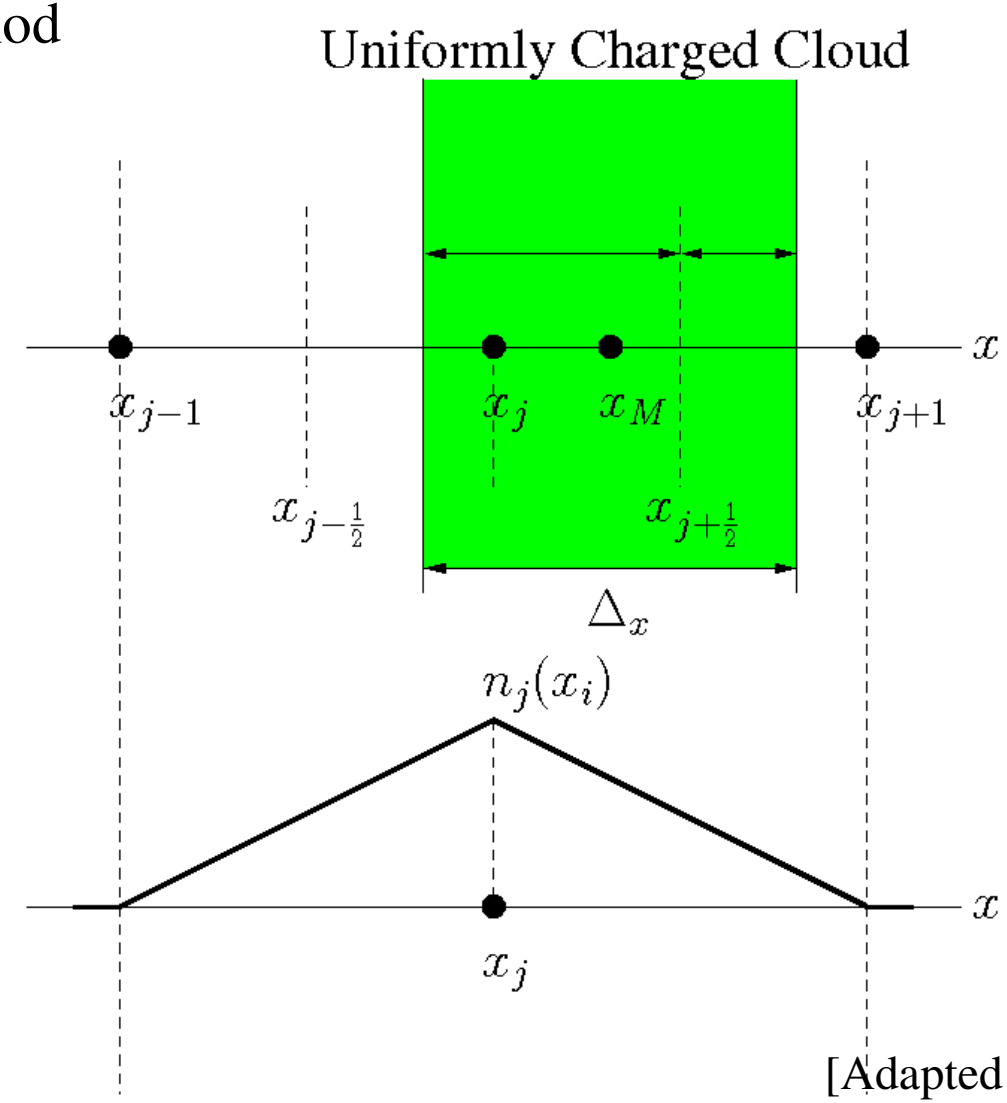
Comments:

- Currents can be interpolated to grid similarly for electromagnetic solving and/or diagnostics

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_j = Closest grid cell

Charge Deposition:

$$q_j = q_M \left[\frac{\Delta_x - (x_M - x_j)}{\Delta_x} \right] = q_M \frac{x_{j+1} - x_M}{\Delta_x}$$

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

Field Interpolation:

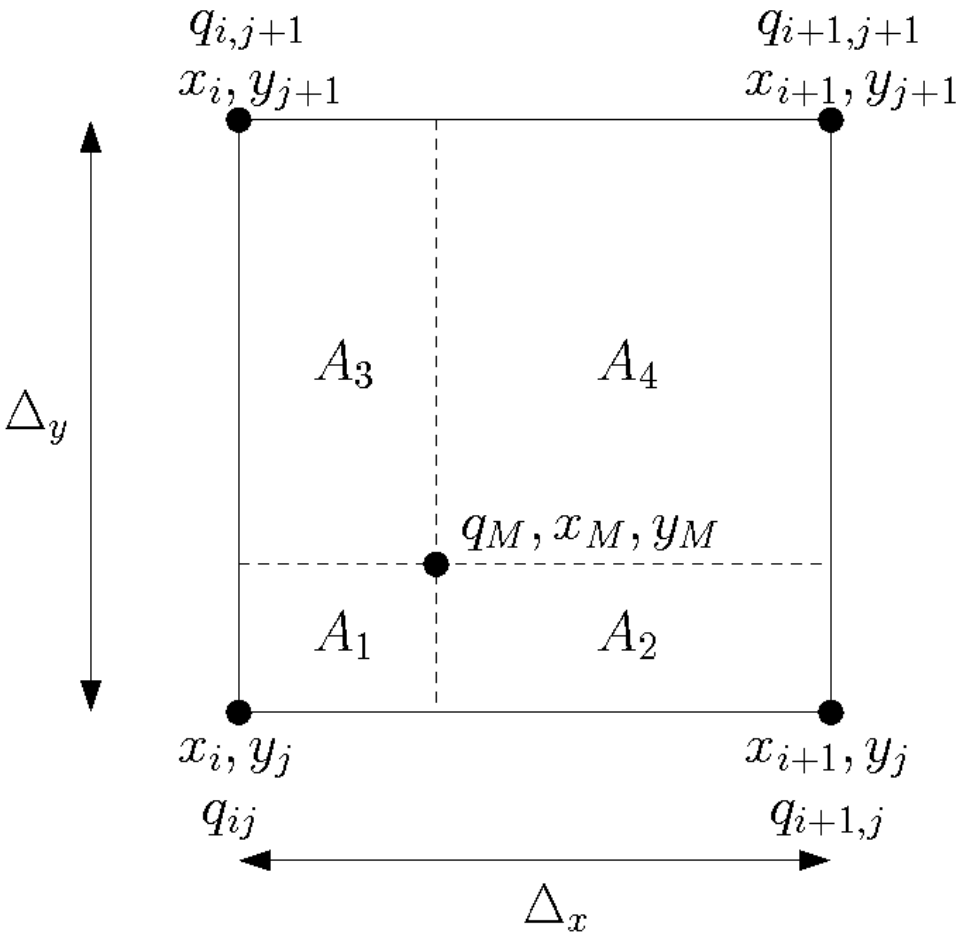
$$E_x\big|_{x=x_M} = \left[\frac{x_{j+1} - x_M}{\Delta_x} \right] E_j + \left[\frac{x_M - x_j}{\Delta_x} \right] E_{j+1}$$

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 interpolated to grid similarly for electromagnetic solving and/or diagnostics

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:

$$q_{ij} = \left(1 - \frac{A_1}{\Delta_x \Delta_y}\right) q_M \qquad A_1 = (x_M - x_i)(y_M - y_j)$$

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

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

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

Field Interpolation:

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

Comments:

- Easily generalized to 3D using volumes
- Currents can be interpolated to grid similarly for electromagnetic solving and/or diagnostics

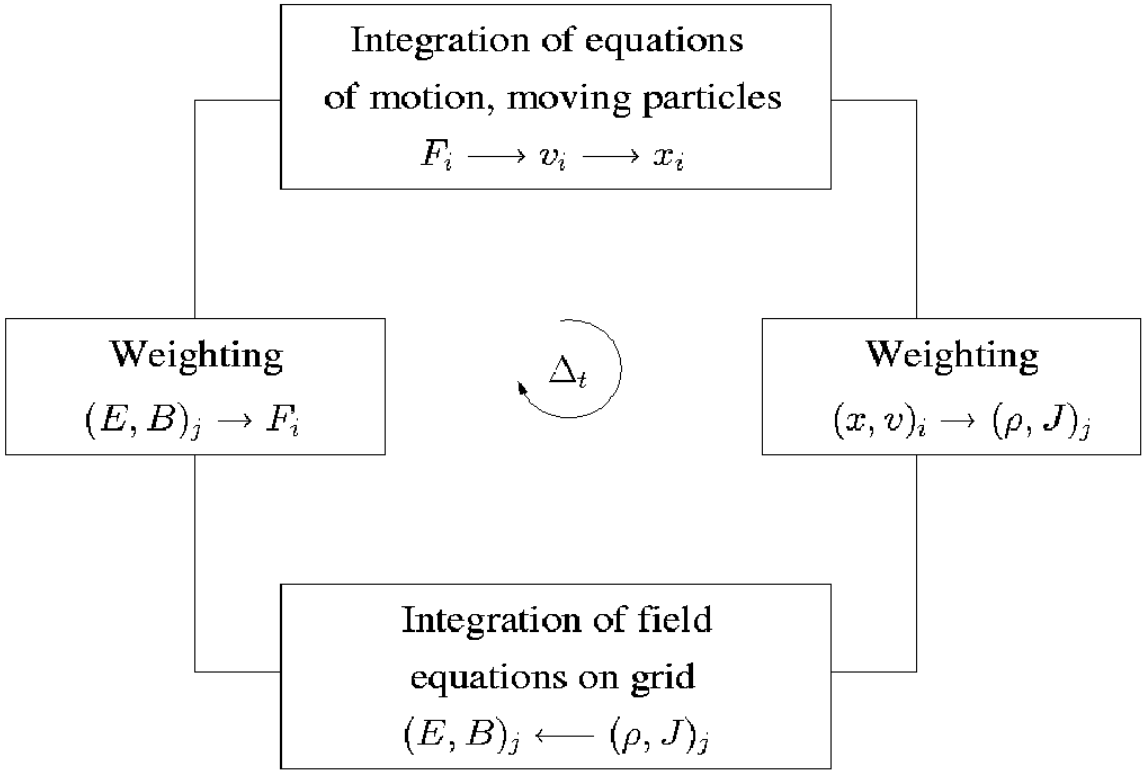
Higher Order Weighting: Splines

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 S5)
  - 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 S6)
  - Particle and field variables may need appropriate de-synchronization to initialize advance

S5: Diagnostics

Diagnostics are *extremely* important. Without effective diagnostics even a correct and well converged simulation is useless. Diagnostics must be well formulated to display relevant quantities in a manner that increases physical understanding by highlighting important processes. This can be difficult since there can be a variety of issues and multiple effects taking place simultaneously.

Diagnostics can be grouped into two broad categories:

1) Snapshot Diagnostics
   - Examples: Particle distribution projections at a particular values of s or t
   - Data can be saved to generate plots after the run or just the needed plots can be generated during the run using linked graphics packages etc.

2) History Diagnostics
   - Examples: moments for the statistical beam centroid, envelope, and emittances
   - Data for history plots must be accumulated and saved over several simulation advance steps

See handwritten notes from USPAS 06 for remaining diagnostics slides

- Will be updated in future versions of the notes

S6: Initial Distributions and Particle Loading

To start the large particle or distribution simulations, the initial distribution function of the beam must be specified.

- For direct Vlasov simulations the distribution need simply be deposited on the phase-space grid

For PIC simulations, an appropriate distribution of macro-particle phase-space coordinates must be generated or "loaded" to represent the Vlasov distribution

Discussion:

In realistic accelerators, focusing elements are *s*-varying. In such situations there are no known smooth equilibrium distributions.

- The KV distribution is an exact equilibrium for linear focusing fields, but has unphysical (singular) structure in 4-dimensional transverse phase-space

Moreover, it is unclear in most cases if the beam is even best thought of as an equilibrium distribution as is typical in plasma physics. In accelerators, the beam in generally injected from a source and may only reside in the machine (especially for a linac) for a small number of characteristic oscillation periods and may not fully relax to an equilibrium like state within the machine.

Initial Distributions: Source-to-Target Simulations

The lack of known, physically reasonable equilibria and the fact that the beams are injected from a source motivates so-called "source-to-target" simulations where particles are simulated off the source and tracked to the target. Such first principles simulations are most realistic if carried out with the actual focusing fields, accelerating waveforms, alignment errors, etc. Source-to-target simulations are highly valuable to measure expected machine performance. However, ideal source-to-target simulations can rarely be carried out due to:

- Source is often incompletely described
  - Example: important alignment and material errors may not be known
- Source may contain physics not adequately in imperfectly modeled
  - Example: plasma injectors with complicated material physics, etc.
- Computer limitations:
  - Memory required and simulation time
  - Convergence and accuracies
  - Limits of numerical methods applied

    Ex: singular description needed for Child-Langmuir model of space-charge limited injection

Initial Distributions: Types of Specified Loads

Due to the practical difficulty of always carrying out simulations off the source, two alternative methods are commonly applied:

1) Load an idealized initial distribution
   - Specify at some specific time
   - Based on physically reasonable theory assumptions
2) Load experimentally measured distribution
   - Construct/synthesize a distribution based on experimental measurements

Discussion:

The 2nd option of generating a distribution from experimental measurements, unfortunately, often has practical difficulties:

- Real diagnostics often are far from ideal 6D snapshots of beam phase-space
  - Distribution must be reconstructed from partial data
  - Typically many assumptions must be made in the synthesis process
- Process of measuring the beam can itself change the beam
- It can sometimes be helpful to understand processes and limitations starting from cleaner, more idealized initial beam states

Discussion Continued:

Because of the practical difficulties of loading a distribution based exclusively on experimental measurements, idealized distributions are often loaded:

- Employ distributions based on reasonable, physical ansatzes
- Use limited experimental measures to initialize:
  - Energy, current, rms equivalent beam sizes and emittances
- Simpler initial state can often aid insight:
  - Fewer simultaneous processes can allow one to more clearly understand how limits arise
  - Seed perturbations of relevance when analyzing resonance effects, instabilities, halo, etc.

A significant complication is that there are no known exact smooth equilibrium distribution functions valid for periodic focusing channels:

- Approximate theories valid for low phase advances may exist
  Davidson, Struckmeier, and others

Formulate a simple approximate procedure to load an initial distribution that reflects features one would expect of an equilibrium

Initial Distributions Based on Continuous Focusing Equilibria

Simple psudo-equilibrium initial distribution:

- Use rms equivalent measures to specify the beam
  - Natural set of parameters for accelerator applications
- Map rms equivalent beam to a smooth, continuous focused matched beam
  - Use smooth core models that are stable in continuous focusing:
  - Waterbag Equilibrium
  - Parabolic Equilibrium
  - Thermal Equilibrium
  - ⋮

} See Notes on: *Transverse Equilibrium Distributions*

- Transform continuous focused beam for rms equivalency with original beam specification
  - Use KV transforms to preserve uniform beam Courant-Snyder invariants

Procedure will apply to any s-varying focusing channel

- Focusing channel need not be periodic
- Beam can be initially rms equivalent matched or mismatched if launched in a periodic transport channel
- Can apply to both 2D transverse and 3D beams

Procedure for Initial Distribution Specification

Assume focusing lattice is given:

$\kappa_x(s), \quad \kappa_y(s)$ specified

Strength usually set by specifying undperessed phase advances

$\sigma_{0x}, \quad \sigma_{0y}$

Step 1:

For each particle (3D) or slice (2D) specify 2nd order rms properties at axial coordinate s

Envelope coordinates/angles:

$$r_x(s) = 2\langle x^2\rangle_\perp^{1/2} \qquad r'_x(s) = 2\langle xx'\rangle_\perp / \langle x^2\rangle_\perp^{1/2}$$
$$r_y(s) = 2\langle y^2\rangle_\perp^{1/2} \qquad r'_y(s) = 2\langle yy'\rangle_\perp / \langle y^2\rangle_\perp^{1/2}$$

Emittance:

$$\varepsilon_x(s) = 4[\langle x^2\rangle_\perp \langle x'^2\rangle_\perp - \langle xx'\rangle_\perp^2]^{1/2}$$
$$\varepsilon_y(s) = 4[\langle y^2\rangle_\perp \langle y'^2\rangle_\perp - \langle yy'\rangle_\perp^2]^{1/2}$$

Perveance:

$$Q = \frac{q\lambda(s)}{2\pi\epsilon_0 m\gamma_b^3(s)\beta_b^2(s)c^2}$$

Procedure for Initial Distribution Specification (2)

If the beam is rms matched, we take:

$$r_x'' + \kappa_x r_x - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_x^2}{r_x^3} = 0$$

$$r_y'' + \kappa_y r_y - \frac{2Q}{r_x + r_y} - \frac{\varepsilon_y^2}{r_y^3} = 0$$

$$\kappa_x(s + L_p) = \kappa_x(s)$$
$$\kappa_y(s + L_p) = \kappa_y(s)$$

$$r_x(s + L_p) = r_x(s)$$
$$r_y(s + L_p) = r_y(s)$$

- Not necessary even for periodic lattices
  - Procedure applies to mismatched beams

Procedure for Initial Distribution Specification (3)

Step 2:

Define an rms matched, continuously focused beam in each transverse s-slice:

Continuous s-Varying

$$r_b(s) = \sqrt{r_x(s) r_y(s)} \qquad \text{Envelope Radius}$$

$$\varepsilon_b(s) = \sqrt{\varepsilon_x(s) \varepsilon_y(s)} \qquad \text{Emittance}$$

$$Q(s) = Q(s) \qquad \text{Perveance}$$

Define a (local) matched beam focusing strength in continuous focusing:

$$r_b'' + k_{\beta 0}^2 r_b - \frac{Q}{r_b} - \frac{\varepsilon_b^2}{r_b^2} = 0 \qquad (Q \to 0)$$

$$\rightarrow \quad k_{\beta 0}^2(s) = \frac{Q(s)}{r_b^2(s)} + \frac{\varepsilon_b^2(s)}{r_b^4(s)}$$

Procedure for Initial Distribution Specification (4)

Step 3:

Specify an rms matched continuously focused equilibrium consistent with step 2:

Specify an equilibrium function:

$$f_\perp(x, y, x', y') = f_\perp(H_\perp) \qquad H_\perp = \frac{1}{2}\mathbf{x}_\perp'^2 + \frac{1}{2}k_{\beta 0}^2\mathbf{x}_\perp^2 + \frac{q\phi}{m\gamma_b^3\beta_b^2c^2}$$

and constrain parameters used to define the equilibrium function with:

$$\lambda = q\int d^2x \int d^2x' \; f_\perp(H_\perp) \qquad \text{Line Charge <--> Perveance}$$

$$r_b^2 = \frac{4\int d^2x \int d^2x' \; x^2 f_\perp(H_\perp)}{\int d^2x \int d^2x' \; f_\perp(H_\perp)} \qquad \text{rms edge radius}$$

$$\frac{\varepsilon_b^2}{r_b^2} = \frac{4\int d^2x \int d^2x' \; x'^2 f_\perp(H_\perp)}{\int d^2x \int d^2x' \; f_\perp(H_\perp)} \qquad \text{rms edge emittance}$$

- Constraint equations are generally highly nonlinear and must be solved numerically
  - Allows specification of beam with natural accelerations variables

Procedure for Initial Distribution Specification (5)

Load N particles in x,y,x',y' phase space consistent with continuous focusing equilibrium distribution $f_\perp(H_\perp)$

Step A (set particle coordinates):

Calculate beam radial number density $n(r)$ by (generally numerically) solving the Poisson/stream equation and load particle x,y coordinates:

$$x = r\cos\theta$$

$$y = r\sin\theta$$

- Radial coordinates r: Set by transforming uniform deviates consistent with $n(r)$
- Azimuthal angles θ: Distribute randomly or space for low noise

Step B (set particle angles):

Evaluate $f_\perp(U,r)$ with $U = \sqrt{x'^2 + y'^2}$ at the particle x, y coordinates loaded in step A to calculate the angle probability distribution function and load x', y' coordinates:

$$x' = U\cos\xi$$

$$y' = U\sin\xi$$

- Radial coordinate U: Set by transforming uniform deviates consistent with $f_\perp(U,r)$
- Azimuthal coordinate ξ: Distribute randomly or space for low noise

Procedure for Initial Distribution Specification (6)

Step 4:

Transform continuous focused beam coordinates to rms equivalency in the system with s-varying focusing:

$$x = \frac{r_x}{r_b} x_i \qquad y = \frac{r_y}{r_b} y_i$$

$$x' = \frac{\varepsilon_x}{\varepsilon_b}\frac{r_b}{r_x} x_i' + \frac{r_x'}{r_b} x_i \qquad y' = \frac{\varepsilon_y}{\varepsilon_b}\frac{r_b}{r_y} y_i' + \frac{r_y'}{r_b} y_i$$

Here, $\{x_i\}$, $\{y_i\}$, $\{x_i'\}$, $\{y_i'\}$ are coordinates of the continuous equilibrium loaded

- Transform reflects structure of Courant-Snyder invariants

Comments on Procedure for Initial Distribution Specification

- Applies to both 2D transverse and 3D beams
- Easy to generalize procedure for beams with centroid offsets
- Generates a charge distribution with elliptical symmetry
  - Sacherer's results on rms equivalency apply
  - Distribution will reflect self-consistent Debye screening
- Equilibria are only pseudo-equilibria since transforms are not exact
  - Nonuniform space-charge results in errors
  - Transform consistent with preserved Courant-Snyder invariants for uniform density beams
  - Errors largest near the beam edge - expect only small errors for very strong space charge where Debye screening leads to a flat density profile with rapid fall-off at beam edge
- Many researchers have presented or employed aspects of the improved loading prescription presented here, including:

  I. Hofmann, GSI M. Reiser, U. Maryland M. Ikigami, KEK

  E. Startsev, PPPL Y. Batygin, SLAC

PIC simulations with the WARP code (see S9) were carried out to verify that the loading procedure results in less fluctuations and waves in self-consistent Vlasov evolutions from the load

Show evolutions from a matched load in a periodic FODO quadrupole transport lattice:

pseudo-thermal
semi-Gaussian (for contrast)

Find:

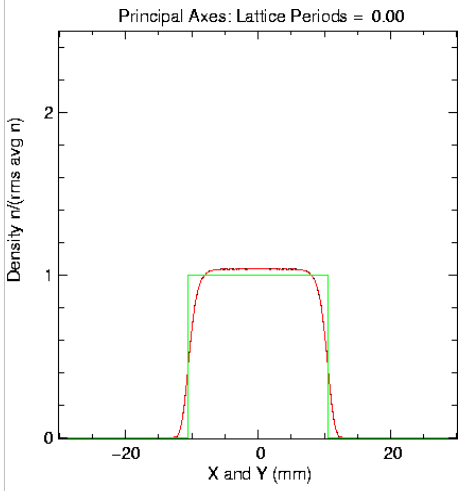
- Works well for $\sigma_0 \lesssim 85^\circ$
  - Should not work where beam is unstable and all distributions are expected to become unstable for $\sigma_0 > \sim 85^\circ$ see:

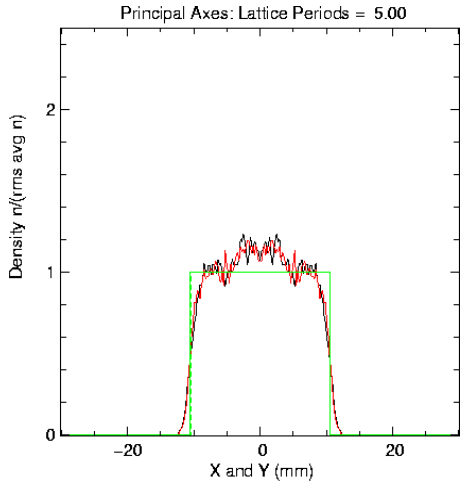
    Experiment: Tiefenback, Ph.D. Thesis, U.C. Berkeley (1986)
    Theory: Lund and Chawla, Proc. 2005 Part. Accel. Conf.
- Works better when matched envelope has less "flutter"
  - Solenoids: larger lattice occupancy η
  - Quadrupoles: smaller σ_0
  - Not surprising since less flutter" corresponds to being closer to continuous focusing

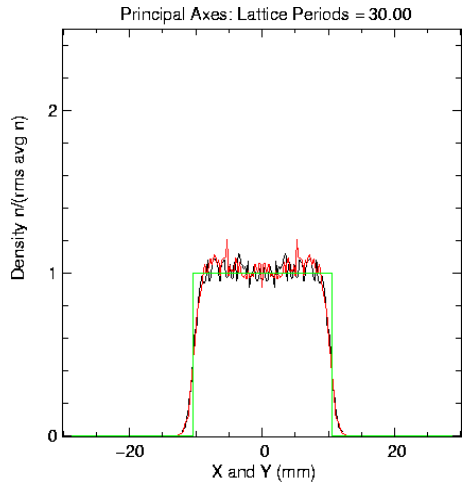
WARP PIC Simulation (see S9) Results – Pseudo Thermal Equilibrium

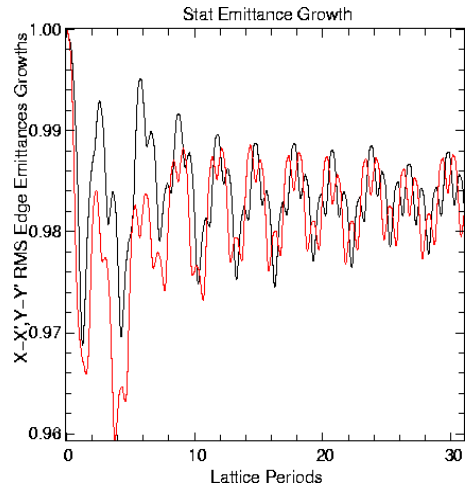
$$\sigma_0 = 70^\circ, \quad L_p = 0.5 \text{ m}, \quad \varepsilon_x = \varepsilon_y = 50 \text{ mm-mrad}$$

$$\sigma/\sigma_0 = 0.2$$

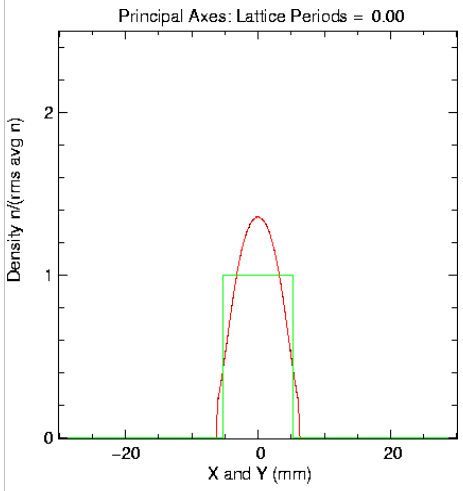


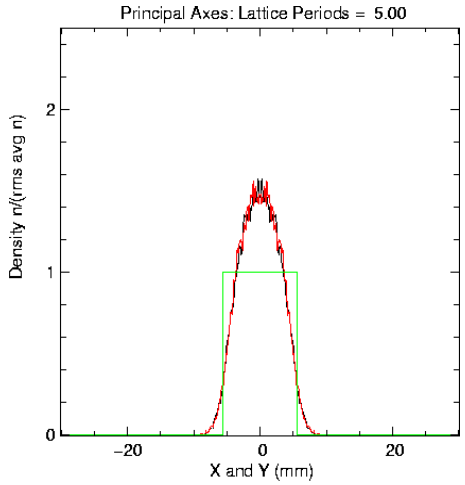


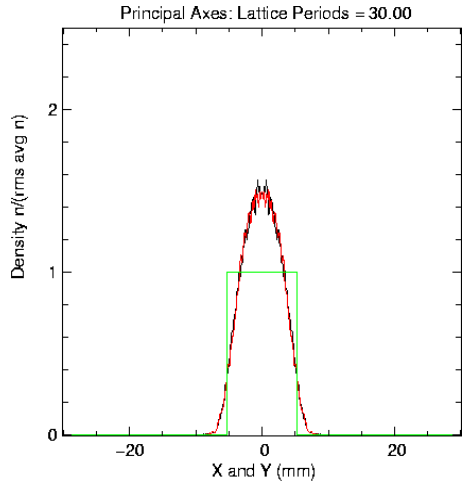


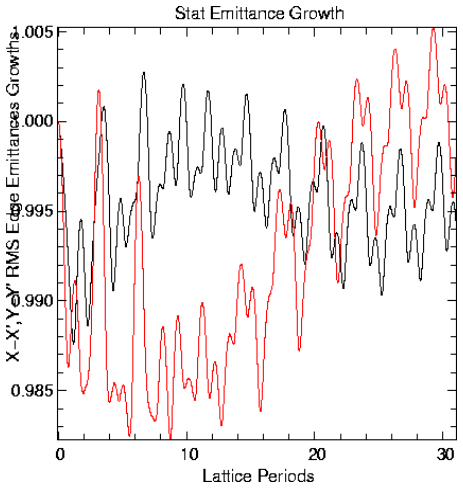


$$\sigma/\sigma_0 = 0.7$$





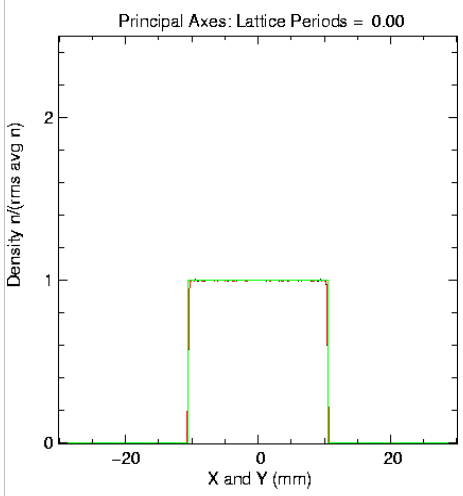


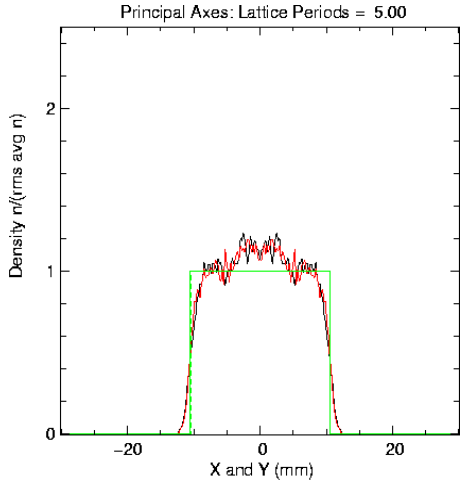


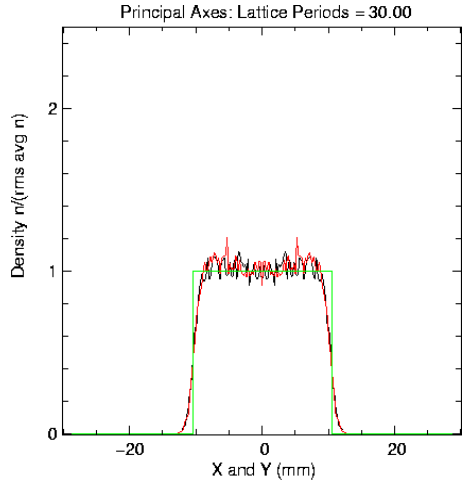
WARP PIC Simulation (see S9) Results – Semi-Gaussian (for contrast)

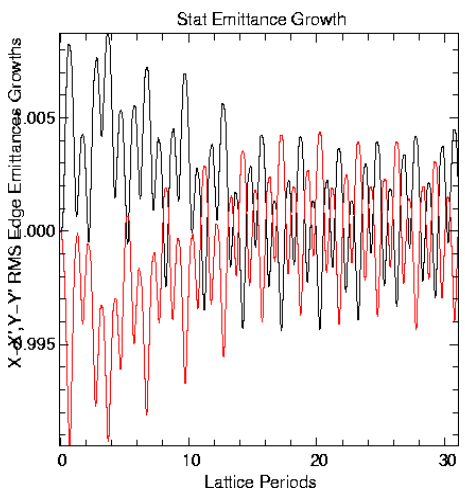
$$\sigma_0 = 70^\circ, \quad L_p = 0.5 \text{ m}, \quad \varepsilon_x = \varepsilon_y = 50 \text{ mm-mrad}$$

$$\sigma/\sigma_0 = 0.2$$

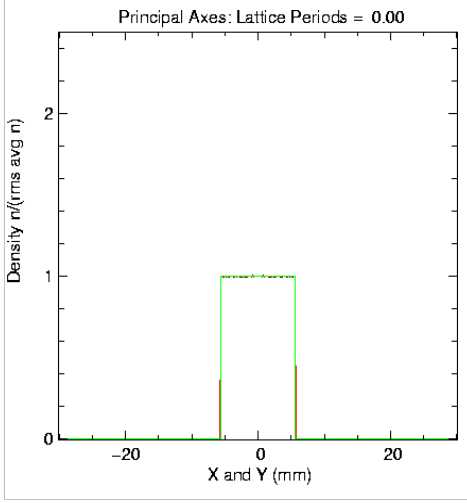


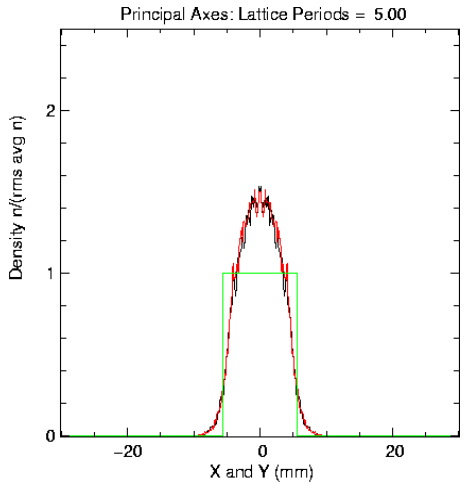


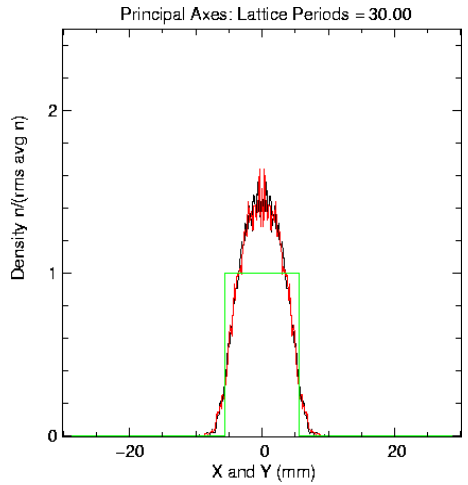


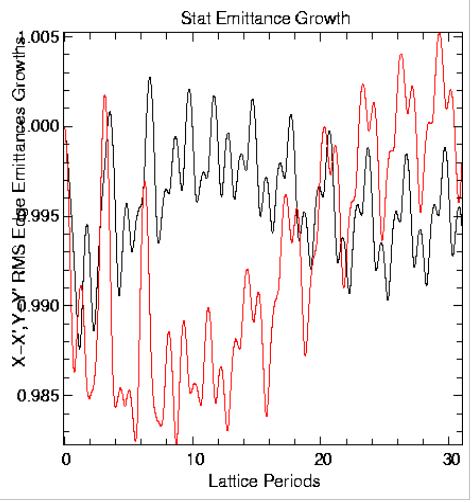


$$\sigma/\sigma_0 = 0.7$$









See handwritten notes from USPAS 06 for remaining distribution loading slides

- Will be updated in future versions of the notes

Initial Loads: The Semi-Gaussian Distribution

See handwritten notes from USPAS 06

- Will be updated in future versions of the notes

S7: Numerical Convergence

Numerical simulations must be checked for proper resolution and statistics to be confident that answers obtained are correct and physical:

Resolution of discretized quantities

- Time *t* or axial s step of advance
- Spatial grid of fieldsolve
- For direct Vlasov: the phase-space grid

Statistics for PIC

- Number of macroparticles used to represent Vlasov flow to control noise

Increased resolution and statistics generally require more computer resources (time and memory) to carry out the required simulation. It is usually desirable to carry out simulations with the minimum resources required to achieve correct, converged results that are being analyzed. Unfortunately, there are no set rules on adequate resolution and statistics. What is required generally depends on:

- What quantity is of interest
- How long an advance is required
- What numerical methods are being employed .....

General Guidance on Numerical Convergence Issues

Although it is not possible to give detailed rules on numerical convergence issues, useful general guidance can be given:

- Find results from similar problems using similar methods when possible
- Analyze quantities that are easy to interpret and provide good measures of convergence for the use of the simulation
  - Some moments like rms emittances:

$$\epsilon_x = 4\left[\langle\tilde{x}^2\rangle_\perp\langle\tilde{x}'^2\rangle_\perp - \langle\tilde{x}\tilde{x}'\rangle_\perp^2\right]^{1/2}$$

$$\tilde{x} \equiv x - \langle x\rangle_\perp$$

$$\tilde{x}' \equiv x' - \langle x'\rangle_\perp$$

  can provide relatively sensitive and easy to interpret measures of relative phase-space variations induced by numerical effects when plotted as overlaid time (or s) evolution "histories"
- Benchmark code against problems with known analytical solutions and properties
  - Apply a variety of numerical methods to judge which applies best
- Benchmark code against established, well verified simulation tools
  - Use different numerical methods expected to be more or less accurate

- Recheck convergence whenever runs differ significantly or when different quantities are analyzed
  - What is adequate for one problem/measure may not be for another
  - Ex: rms envelope evolution easier to converge than collective modes
- Although it is common to increase resolution and statistics till quantities do not vary, it is *also* useful to purposefully analyze poor convergence so characteristics of unphysical errors can be recognized
  - Learn characteristic signature of failures to resolve effects so subtle onset issues can be recognized more easily
- Expect to make *many* setup, debugging, and convergence test runs for each useful series of simulations carried out

See handwritten notes from USPAS 06 for remaining slides

- Will be updated in future versions of the notes

S8: Practical Considerations:
A: Overview

Intense beam simulation problems can be highly demanding on computer resources – particularly for realistic higher dimensional models. The problem size that can be simulated is dictated by computer resources available in fast memory and the run time required to complete the simulation

- Fast Memory (RAM)
- Wall Clock Run Time (Computer Speed)

Both of these can depend strongly on the architecture of computer system that the problem is run on:

- Serial Machine
- Parallel Machine

can strongly influence the size of the problem that can be simulated. We will present rough estimates of the computer memory required for simulations and provide some guidance on how the total simulation time can scale on various computer systems. The discussion is limited to PIC and direct Vlasov simulations.

S8B: Practical Considerations: Fast Memory

Fast computer memory (RAM) dictates how large a problem can be simulated

- If a problem will not fit into fast memory (RAM), computer performance will be severely compromised
- Writes to hard disks are slow

There are 3 main contributions to the problem size for typical PIC or direct Vlasov simulations:

1) Particle Phase Space Coordinates (PIC)
   or Discretized Distribution Function (Direct Vlasov)
2) Gridded Field
3) General Code Overhead

These three contributions to memory required are discussed in turn

Particle and field quantities are typically stored in double precision:

Representation	Digits (Floating Point)	Bytes Memory
Single Precision	8	4
Double Precision	16	8

Most problems

Estimates of Required Fast Memory

1) Particle Phase Space Coordinates (PIC):

B = bytes of floating point number (typically 8 for double precision)
N_p = number macro particles (0 for direct Vlasov)
D = dimension of variables characterizing particles

Memory = $B * N_p * D$ Bytes

The dimension D depends on the specific type of PIC simulation and methods employed

/// Common Examples of *D*:

3D PIC: $D = 7$

$x,\ y,\ z$

$p_x,\ p_y,\ p_z,\ \gamma^{-1}$

2D Transverse Slice PIC: $D = 5$

$x,\ y$

$p_x,\ p_y, \gamma^{-1}$ + p_z (*D*=6) some models

γ^{-1} is often included often to optimize the mover

///

Estimates of Required Fast Memory

1) Discretized Distribution Function (Direct Vlasov):

B = bytes of floating point number (typically 8 for double precision)

N_{pm} = number mesh points of grid describing the discretized particle phase space

Memory = $B * N_{pm}$ Bytes

The value of N_{pm} depends critically on the dimensionality of the phase space

// Examples of N_{pm} scaling for a uniform phase-space meshes:

Problem	Phase Space	N_{pm}	Scaling ($n_x = n_{p_x} \equiv n$ etc)
1D $\parallel$	$z - p_z$	$n_z n_{p_z}$	n^2
2D $\perp$ Slice	$x - p_x,\ y - p_y$	$n_x n_y n_{p_x} n_{p_y}$	n^4
2D Slice	$x - p_x,\ y - p_y,\ p_z$	$n_x n_y n_{p_x} n_{p_y} n_{p_z}$	n^5
3D	$x - p_x,\ y - p_y\ z - p_z$	$n_x n_y n_{p_x} n_{p_y}$	n^6

n_x = number mesh points in x etc.

Rapid growth of N_{pm} with dimensionality severely limits tractability of problems

Memory required for a double precision ($B = 8$) uniform phase-space grid with 100 zone discretization per degree of freedom:

$$n_x = n_{p_x} \equiv n = 100 \text{ etc.}$$

$$\text{D} = \text{dimension of phase-space}$$

$$\text{Memory} = 8 * n^D \text{ Bytes}$$

Problem	D	Memory (Bytes) (n = 100)
1D $\parallel$	2	$80 \times 10^3 \sim 80$ KB
2D $\perp$ Slice	4	$80 \times 10^6 \sim 80$ MB
2D Slice	5	$80 \times 10^9 \sim 80$ GB
3D	6	$8 \times 10^{12} \sim 8000$ GB

//

Rapidly increasing problem size with phase-space dimension D practically limits what can be simulated on direct Vlasov models with reasonable resolution *even on large parallel computers*:

- Irregular phase-space grids that place resolution where it is needed can partially alleviate scaling problem
- Optimal methods must also only grid minimal space exterior to the oscillating beam core in alternating gradient lattices

2) Gridded Field:

Required memory for a gridded field solve depends on the class of field solve (electrostatic, electromagnetic), mesh size, and numerical method employed. For a concrete illustration, consider *electrostatic* problems using a simple FFT field solve:

- Discrete Fourier Transform complex, but transform is of real functions. Proper optimization allows use of transforms using only real ϕ and ρ arrays
- Electric field is typically not stored and is calculated for each particle only where it is needed. Spatial grid location need not be stored.
  - Some methods store gridded E to optimize specific problems

N_{fm} = number mesh points of field spatial grid

$$\text{Memory} = 2 * B * N_{fm} \quad \text{Bytes}$$

Factor of 2 for: ρ, ϕ

Number of mesh points N_{fm} depends *strongly* on the dimensionality of the field solve and the structure of the mesh

- Generally more critical to optimize storage and efficiency (see next section) of fieldsolvers in higher dimensions

Examples for uniform meshes:

$$N_{fm} = n_z \qquad \text{1D (Longitudinal)}$$
$$= n_x n_y \qquad \text{2D (Transverse Slice)}$$
$$= n_x n_y n_z \qquad \text{3D}$$

n_x = number mesh points in x etc.

3) General Code Overhead:

System memory is also used for:

- Scratch arrays for various numerical methods (fieldsolvers, movers, etc.)
- History accumulations of diagnostic moments
- Diagnostic routines
- Graphics packages, external libraries, etc.
  - Graphics packages can be large!

Memory = $M_{overhead}$ Bytes

Characteristic of packages used, size of code, and methods employed. But typical numbers can range 1 MB – 20 MBytes

Summary: Total Memory Required:

For illustrative example, add contributions for electrostatic PIC

PIC: Tot Memory = $B * (N_p * D + 2 * N_{fm}) + M_{overhead}$ Bytes

Direct Valsov: Tot Memory = $2 * B * (N_{pm} + N_{fm}) + M_{overhead}$ Bytes

Reminder: Machine fast memory (RAM) capacity *should not be exceeded*

- Storing data on disk and cycling to RAM generally too slow!

S8C: Practical Considerations: Run Time

Run time can depend on many factors including:

- Type of problem
- Dimensionality of problem and number of particles and/or mesh points
- Numerical methods employed (particle moving, fieldsolve, ...)
- Moments and diagnostics accumulated
- Architecture/speed of computer system

It is not possible to give fully general guidance on estimating run times. However, to better characterize the time required, it can be useful to benchmark the code on the computer to be employed in terms of:

t_{step} = Time for an "ordinary" run step

Generally, parts of the code that more time is spent in should be more carefully optimized to minimize total run time. Care should generally be applied with:

- Particle mover
- Field solver
- Frequent diagnostics such as moments

Diagnostics, loaders, problem setup routines, etc. can often be coded with less care for optimization since they are only executed infrequently. However:

- Diagnostics often take a large amount of development time
  - Often better to code as simply as possible!

Software profiling tools can be useful to best understand where “bottlenecks” occur so effort on optimization can be appropriately directed for significant returns.

Dimenstionality plays a strong role in required run time

Some general guidance for electrostatic PIC Simulations:

1D: (Longitudinal typical)
Fieldsolve generally fast: small fraction of time compared to moving particles

- Green's function methods can be used (Gauss Law)

2D: (Transverse slice typical)
Fieldsolve typically a small fraction of time relative to moving particles if fast gridded methods are applied (like FFT based methods)

- Special boundary conditions can increase the fraction

Method	Numerical Work
FFT with Periodic BC	Small fraction of particle moving
FFT with Capacity Matrix	.
SOR	.
.	.
Green's Function	Dominates particle moving

3D:

Fieldsolve typically comparable in time or dominates time for particle moving even if fast, gridded methods are applied

- Fieldsolve efficiency of *critical* importance in 3D to optimize run time
- Whole classes can be taught just on methods of 3D electrostatic field solves for Poission's equation

Some general guidance for Direct Vlasov Simulations:

The rapid growth of the problem size with the phase space-dimension and available fast computer memory can severely limit problem sizes that can be simulated:

- Numerical work can be significant to advance the discretized distribution over characteristics
- Size of gridded field arrays can be very large leading to slow advances
  - Uniform mesh: D

The type of computer system employed can also strongly influence run time

- Processor Speed
- Memory Speed
  - RAM
  - Fast, optimized cache memory
- System Architecture (see next section)
  - Serial
  - Parallel
- Library Optimization
  - Especially for parallel machines

S8D: Practical Considerations: Machine Architectures

Problems may be simulated on:

1) Serial Machines
   - Single processor or an independently run processor on a multi-processor machine (example: most present multi-"core" processors)

2) Parallel Machine
   - Multi-processors coordinated to work as a large single processor
   - Usually employ independent memory for each processor making up the machine but sometimes uses shared memory among processors

Serial machines represent traditional computers (PCs workstations, etc), whereas parallel machines are generally less familiar.

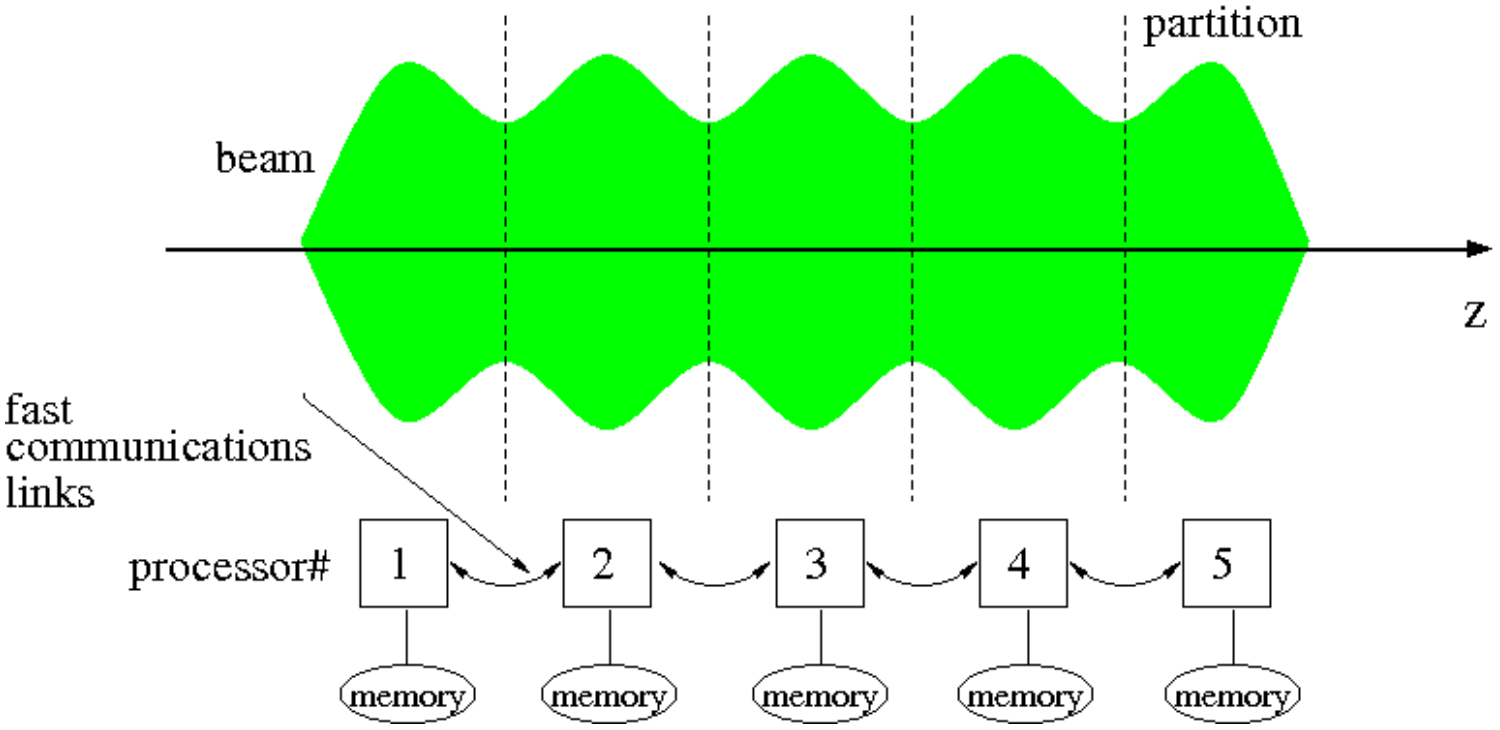
Overview of parallel simulations:

In recent years parallel machines have significantly improved with libraries that allow more "natural" problem formulation with less effort and they are enabling significantly larger simulations to be carried out

- Several 100 million particles typically practical to simulate on large machines

Typical Parallel Machine Architecture

Beam problems may often be conveniently partitioned among processors in terms of axial slices. Schematic example (5 processors):



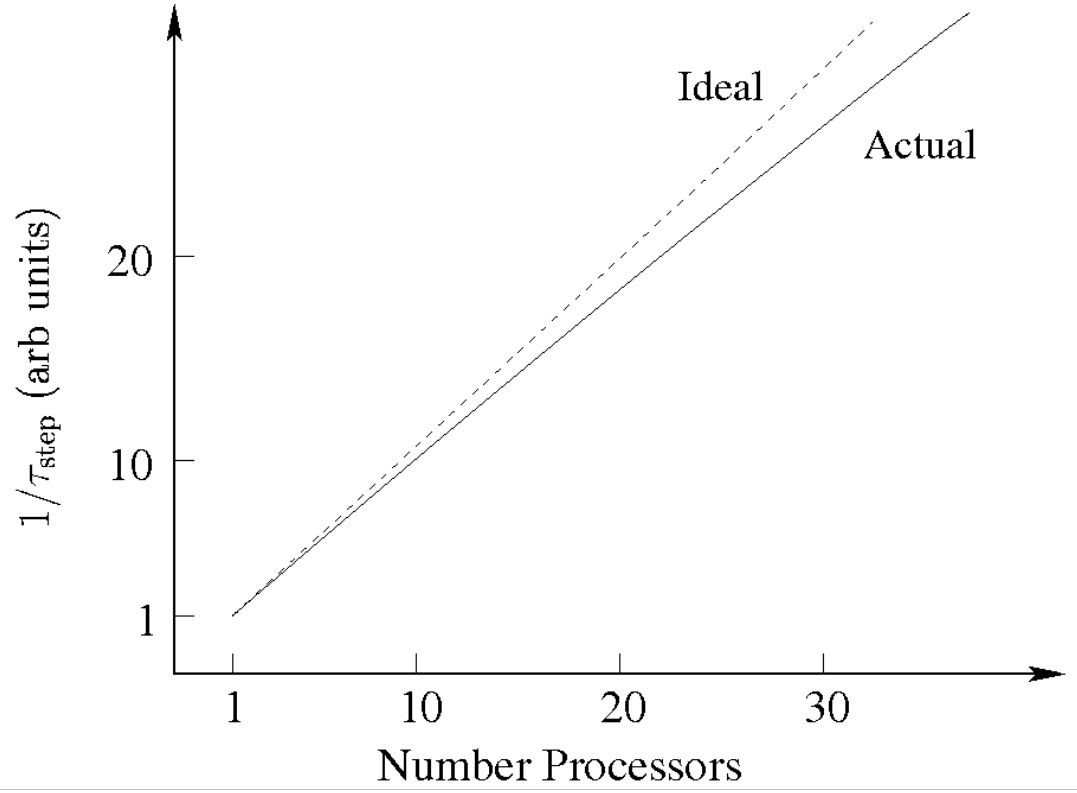
- Sharing of data at boundaries is necessary for fieldsolve
- Problems with axial velocity spread will generally require sorting of particles to maintain the load balance between processors
  - Processors should ideally all perform an equal amount of work since the slowest will dictate the total time of the advance step

Ideal parallelization will result in a linear speedup with processor number

- Actual speedup less due to:
  - Overhead in data transfers
  - Lack of ideal load balance causing processors to wait on the slowest one that the problem is partitioned among

$$\tau_{\text{step}} = \text{Time "ordinary" step in compuational cycle}$$

$$t_{\text{sim}} = \frac{\text{Simulation Time}}{\tau_{\text{step}}}$$



Even with the significant advances in problem size and speed promised by parallel computers, the solution of realistic 3D beam problems with direct (not gridded) fields remains far too large a problem to simulate with present coputer systems. Thus, for detailed simulations, we often push computer resources to the maximum extent possible.

- Better numerical algorithms
- Parallelization
- ....

S9: WARP Code Overview

See handwritten notes from USPAS 06 for remaining slides

- Will be updated in future versions of the notes

S10: Example Simulations

Examples to this point have mostly been simply formulated to illustrate concepts. Here, we present results from more complex simulations carried out in support of experiments, theory, and for machine design. Simulations highlighted include:

- Electrostatic Quadrupole Injector
- Multi-beamlet Injector
- Collective Mode Effects
- Detailed Transport Lattice Design
- Transport Limits in Periodic Quadrupole Focusing Channels
- Electron Cloud Effects for Ion Beam Transport

All these simulations, as well as many of the preceding illustrations in the lecture notes, were produced with the WARP code described in S9. Only select issues from the problems are highlighted.

Example: Electrostatic Quadrupole Injector

See handwritten notes from USPAS 06 for remaining slides

- Will be updated in future versions of the notes

These notes will be corrected and expanded for reference and future editions of US Particle Accelerator School and University of California at Berkeley courses:

"Beam Physics with Intense Space Charge"

"Interaction of Intense Charged Particle Beams with Electric and Magnetic Fields"

by J.J. Barnard and S.M. Lund

Corrections and suggestions for improvements are welcome. Contact:

Steven M. Lund
Lawrence Berkeley National Laboratory
BLDG 47 R 0112
1 Cyclotron Road
Berkeley, CA 94720-8201

SMLund@lbl.gov
(510) 486 – 6936

Acknowledgments

Sven Chilton (UCB, LLNL) assisted in the development of part of these lecture notes and in generating some of the numerical examples and figures

Special thanks are deserved for Alex Friedman, Dave Grote, and Jean-Luc Vay of the Lawrence Livermore and Lawrence Berkeley National Laboratories for help with these notes and extensively educating the authors in simulation methods.

Rami Kishek (UMD) assisted teaching a version of this class and contributed to the notes. Irving Haber (UMD), Christine Celata (LBL), and Bill Fawley (LBL) helped educate the authors on various simulation methods.

Michiel de Hoon helped with an early version of the lectures and with example Lagrangian methods.

References: For more information see:

Numerical Methods

Forman S. Acton, *Numerical Methods that Work*, Harper and Row Publishers, New York (1970)

Steven E. Koonin, *Computational Physics*, Addison-Wesley Publishing Company (1986)

W. Press, B. Flannery, S. Teukolsky, W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press (1992).

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).