Multi-Particle Simulation Techniques I

Ji Qiang

Accelerator Modeling Program
Accelerator Technology & Applied Physics Division
Lawrence Berkeley National Laboratory

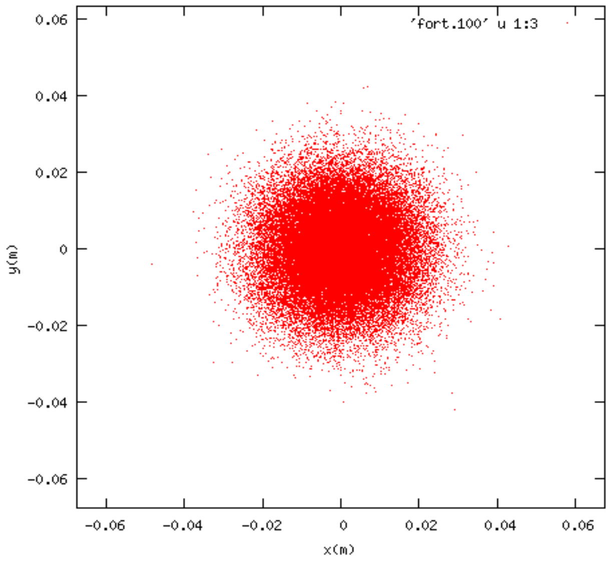
CERN Accelerator School, Thessaloniki, Greece
Nov. 15, 2018

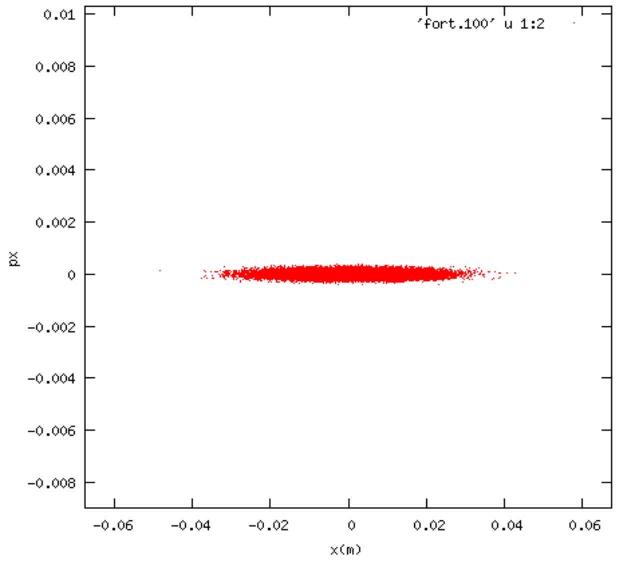
Outline

- Introduction of the particle-in-cell method for multi-particle simulation
- Deposition/interpolation
- Self-consistent field calculations
  - FFT based Green function method for open boundary condition
  - Multigrid method for irregular shape boundary condition
- Particle advance

Introduction

- Particle-in-cell method:
  "the method amounts to following the trajectories of charged particles in self-consistent electromagnetic (or electrostatic) fields computed on a fixed mesh"
- Particle-in-cell codes are widely used in accelerator physics community:
  - PARMELA, ASTRA, GPT, IMPACT-T, IMPACT-Z, GENESIS, GINGER….
- An example of 2D particle-in-cell simulation an mismatched beam transport through a FODO





Courtesy of R. D. Ryne

Governing Equations in Space-Charge Simulation

$$\frac{\partial f(r,p,t)}{\partial t} + \dot{r}\frac{\partial f(r,p,t)}{\partial r} + \dot{p}\frac{\partial f(r,p,t)}{\partial p} = 0$$

$$\nabla^2 \phi = -\rho / \varepsilon$$

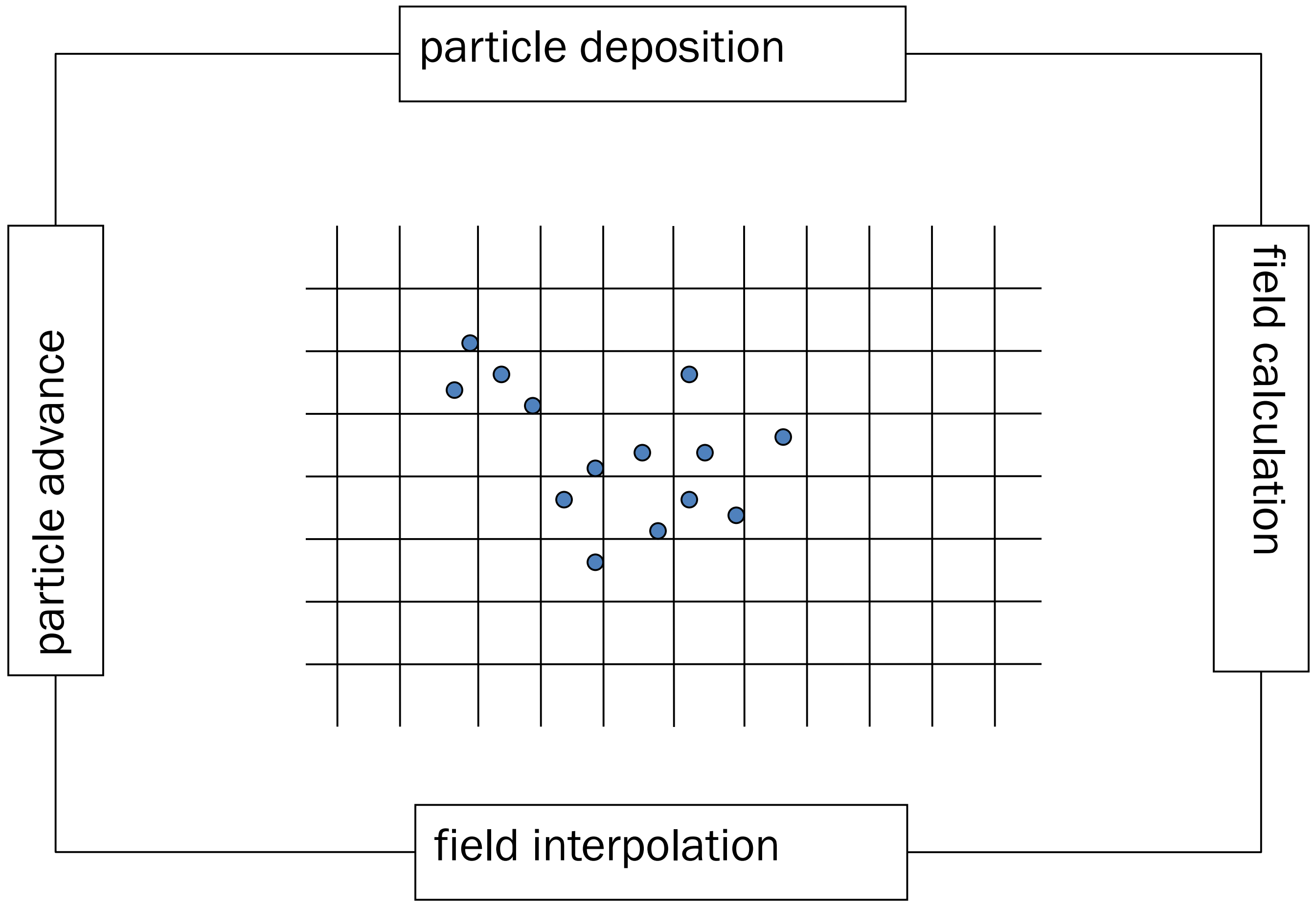
$$\rho = \iiint f(r,p,t) d^3 p$$

$$\rho = \sum w_i \delta(r - r_i)(p - p_i)$$

Particle equations:

$$\boxed{\begin{aligned} \frac{d\boldsymbol{x}_p}{dt} &= \boldsymbol{v}_p, \\ \frac{dm\boldsymbol{v}_p}{dt} &= q\left(\boldsymbol{E} + \boldsymbol{v}_p \times \boldsymbol{B}\right), \end{aligned}}$$

One Step Particle-In-Cell Method

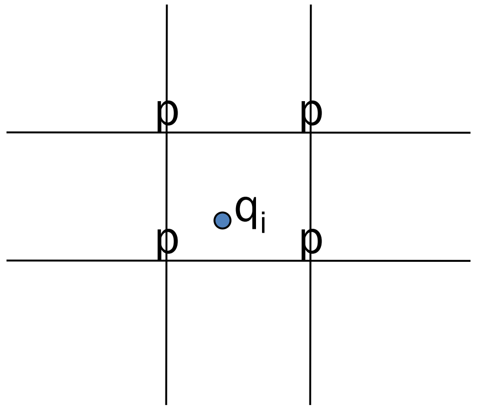


Particle Deposition/Field Interpolation

Particle deposition:

$$\rho_p = \sum_i q_i w(x_i - x_p)$$

$$\mathbf{J}_p = \sum_i q_i w(x_i - x_p)\mathbf{v}_i$$



Field interpolation:

$$\mathbf{E}_i = \sum_p \mathbf{E}_p w(x_i - x_p)$$

$$\mathbf{B}_i = \sum_p \mathbf{B}_p w(x_i - x_p)$$

- *Grid reduces the computational cost compared with direct N-body point-to-point interaction*
- *Grid also provides smoothness to the shot noise and close collision*

Weight Function for Deposition/Interpolation

- Spatial localization of errors

  - At particle separations large compared with the mesh spacing, the field error should be small

- Smoothness

  - The charge assigned to the mesh from a particle and the force interpolated to a particle a particle from the mesh should smoothly vary as the particle moves across the mesh

- Momentum conservation

  - No self force

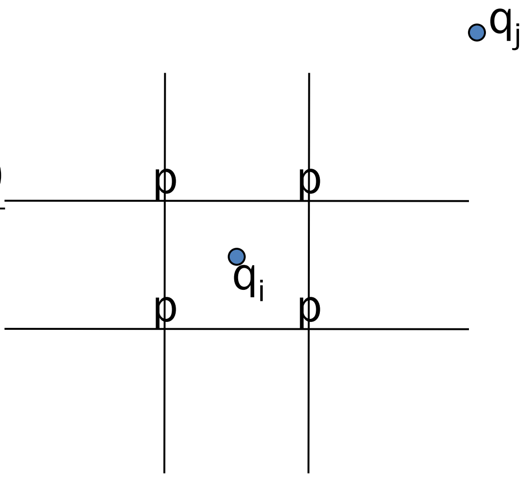
Weight Function for Deposition/Interpolation

- *Spatial localization of errors*

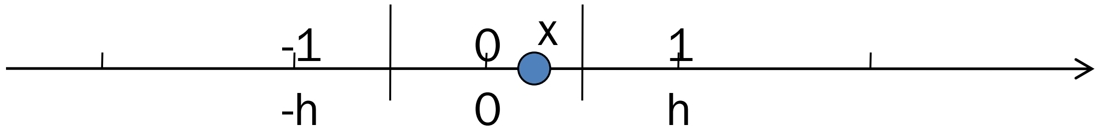
$$\phi(x_j) = \sum_{p=1}^{m} w_p(x_i) G(x_j - x_p) = \sum_{p=1}^{m} w_p(x_i) G(x_j - x_i + x_i - x_p)$$

$$\phi(x_j) = \sum_{p=1}^{m} w_p(x_i) G(x_j - x_i) + \sum_{p=1}^{m} w_p(x_i) \sum_{n=1}^{\infty} \frac{(x_i - x_p)^n}{n!} \frac{d^n G(x_j - x_i)}{dx^n}$$

$$\sum_{p=1}^{m} w_p(x_i) = 1 \qquad \sum_{p=1}^{m} w_p(x_i)(x_i - x_p)^n = const$$



- *Smoothness:*
  - *Continuity of weight function value*
  - *Continuity of derivative*



- *Momentum conservation*

$$F_{self}(x_i) = \sum_{p=1}^{m} \sum_{p'=1}^{m} d(x_p; x_{p'}) w_{dep}(x_i - x_{p'}) w_{\text{int}}(x_i - x_p)$$

$$d(x_p; x_{p'}) = -d(x_{p'}; x_p)$$

Weight Functions for Deposition/Interpolation

NGP:

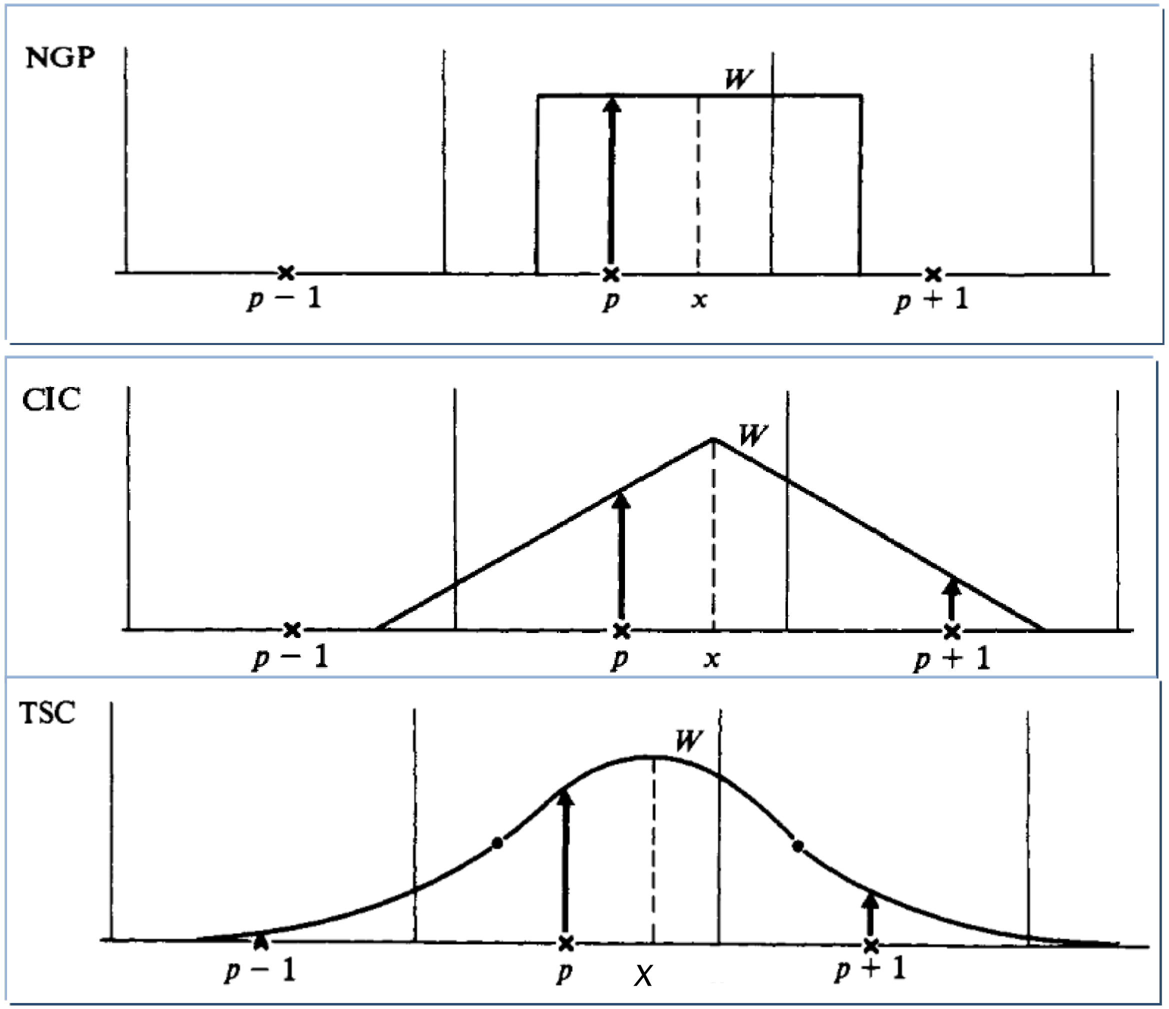
$$w_p(x) = \Pi(\frac{x - x_p}{h})$$

CIC:

$$w_p(x) = 1 - |\frac{x - x_p}{h}|$$

TSC:

$$w_p(x) = \begin{cases} \frac{3}{4} - (\frac{x - x_p}{h})^2 \\ \frac{1}{2}(\frac{3}{2} - |\frac{x - x_p}{h}|)^2 \end{cases}$$



R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*

Self-Consistent Field Calculations

$$\nabla^2 \phi = -\rho / \varepsilon$$

$$\rho = \iiint f(r, p) d^3 p$$

Different Boundary/Beam Conditions Need Different Efficient Numerical Algorithms O(Nlog(N)) or O(N)

FFT based Green function method:

- Standard Green function: low aspect ratio beam
- Shifted Green function: separated particle and field domain
- Integrated Green function: large aspect ratio beam
- Non-uniform grid Green function: 2D radial non-uniform beam

Fully open boundary conditions

Spectral-finite difference method:

Transverse regular pipe with longitudinal open/periodic

Multigrid spectral-finite difference method:

Transverse irregular pipe

Field Calculation with Open Boundary Conditions

Green Function Solution of Poisson's Equation

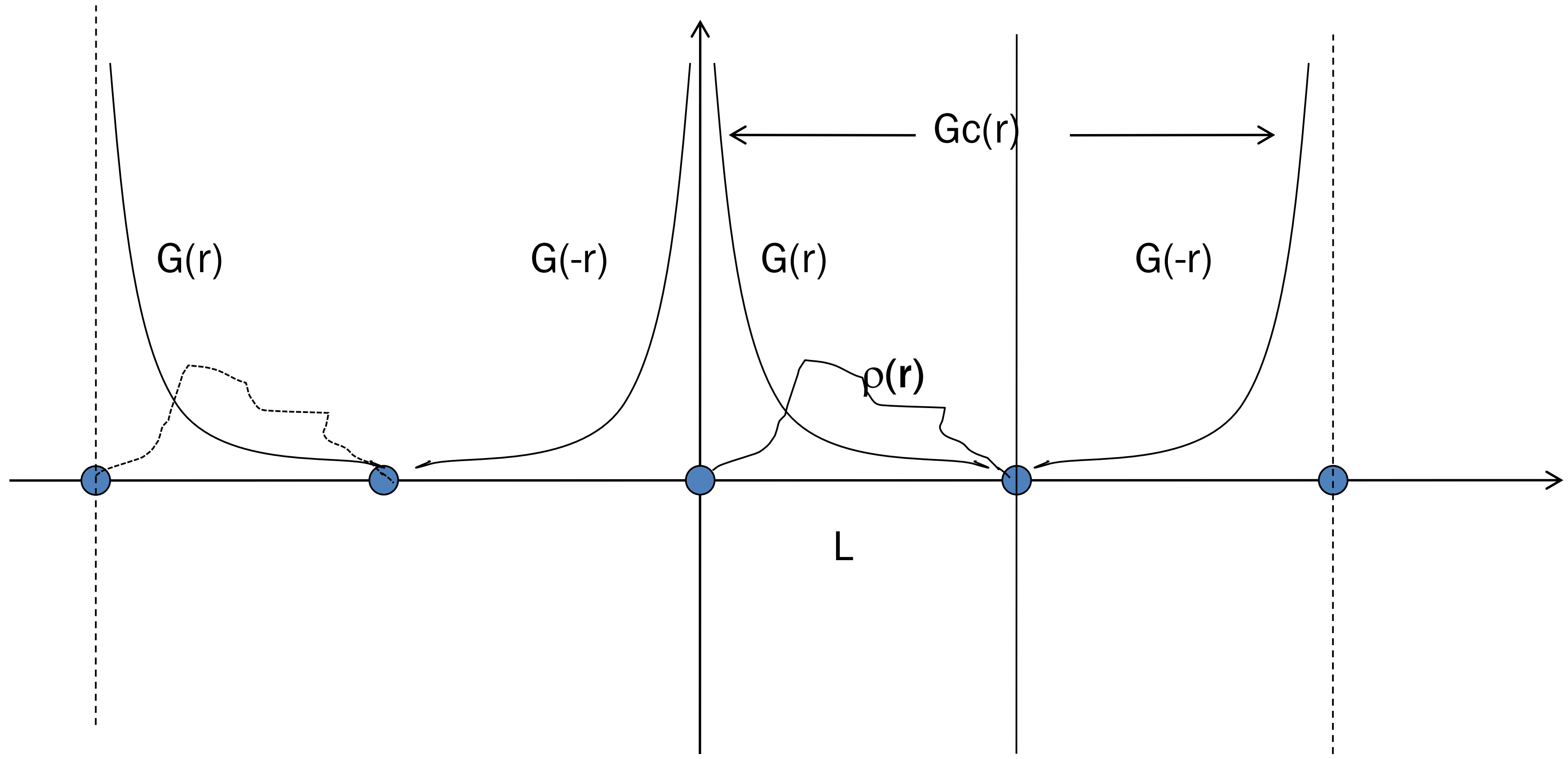
$$\phi(r) = \int G(r, r')\rho(r')dr' \quad ; \quad r = (x, y, z)$$

$$\phi(r_i) = h \sum_{i'=1}^{N} G(r_i - r_{i'})\rho(r_{i'})$$

$$\boxed{G(x, y, z) = 1/\sqrt{(x^2 + y^2 + z^2)}}$$

Direct summation of the convolution scales as N^2 !!!!
N – total number of grid points

Hockney's Algorithm or Zero Padding



- This is different from a real periodic system
- The real calculation is done in discrete coordinate instead of continuous coordinate

R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*

Hockney's Algorithm or Zero Padding

$$\bar{\phi}_c(x_i, y_j) = \frac{h_x h_y}{2\pi\epsilon_0} \sum_{i=1}^{2N_x} \sum_{j=1}^{2N_y} G_c(x_i - x_{i'}, y_j - y_{j'}) \bar{\rho}_c(x_{i'}, y_{j'}),$$

where $i = 1, \ldots, 2N_x$, $j = 1, \ldots, 2N_y$, and

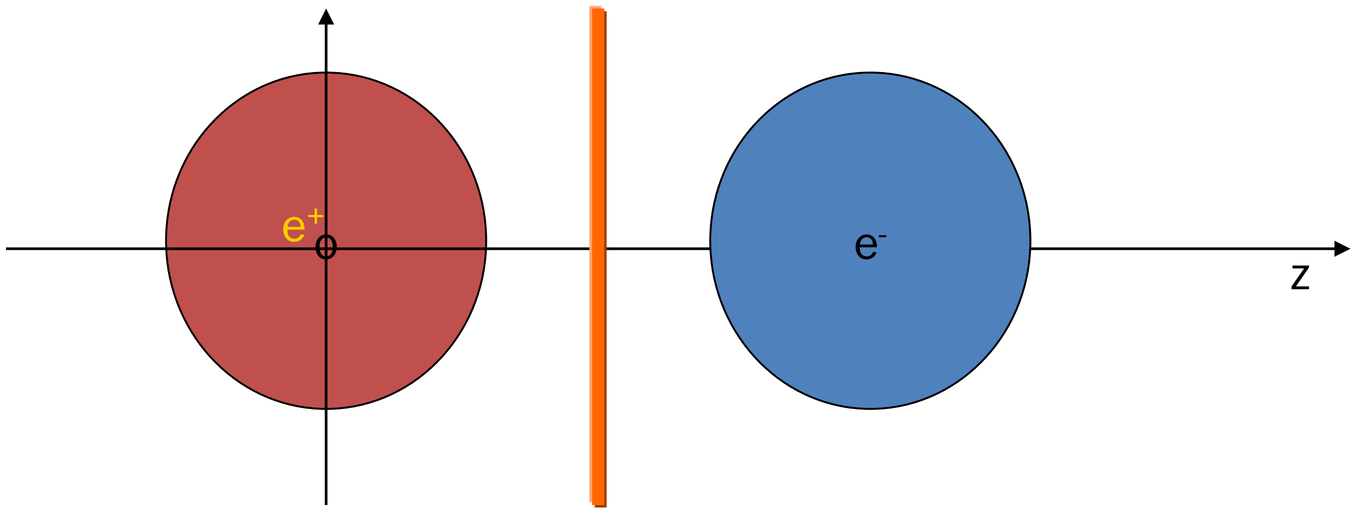
$$\bar{\rho}_c(x_i, y_j) = \begin{cases} \bar{\rho}(x_i, y_j) & : 1 \leqslant i \leqslant N_x; 1 \leqslant j \leqslant N_y, \\ 0 & : N_x < i \leqslant 2N_x \text{ or } N_y < j \leqslant 2N_y, \end{cases}$$

$$G_c(x_i, y_j) = \begin{cases} G(x_i, y_j) & : 1 \leqslant i \leqslant N_x + 1; \quad 1 \leqslant j \leqslant N_y + 1, \\ G(x_{2N_x - i + 2}, y_j) & : N_x + 1 < i \leqslant 2N_x; \quad 1 \leqslant j \leqslant N_y + 1, \\ G(x_i, y_{2N_y - j + 2}) & : 1 \leqslant i \leqslant N_x + 1; \quad N_y + 1 < j \leqslant 2N_y, \\ G(x_{2N_x - i + 2}, y_{2N_y - j + 2}) & : N_x + 1 < i \leqslant 2N_x; \quad N_y + 1 < j \leqslant 2N_y, \end{cases}$$

$$\bar{\rho}_c(x_i, y_j) = \bar{\rho}_c(x_i + 2(L_x + h_x), \quad y_j + 2(L_y + h_y)),$$

$$G_c(x_i, y_j) = G_c(x_i + 2(L_x + h_x), \quad y_j + 2(L_y + h_y)).$$

A Schematic Plot of an e^- Beam and Its Image Charge



cathode

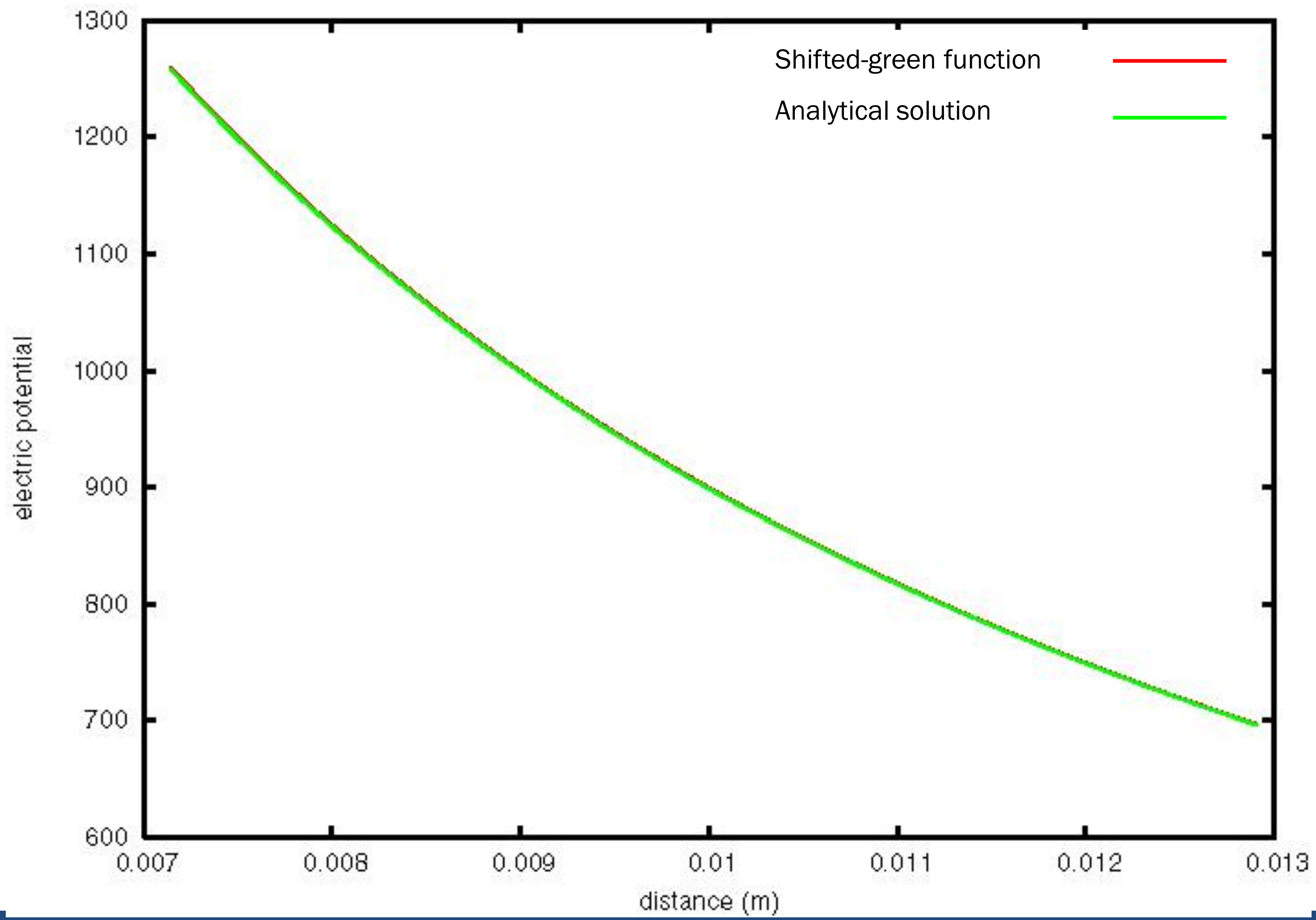
Shifted Green function Algorithm:

$$\phi_F(r) = \int G_s(r, r')\rho(r')dr'$$

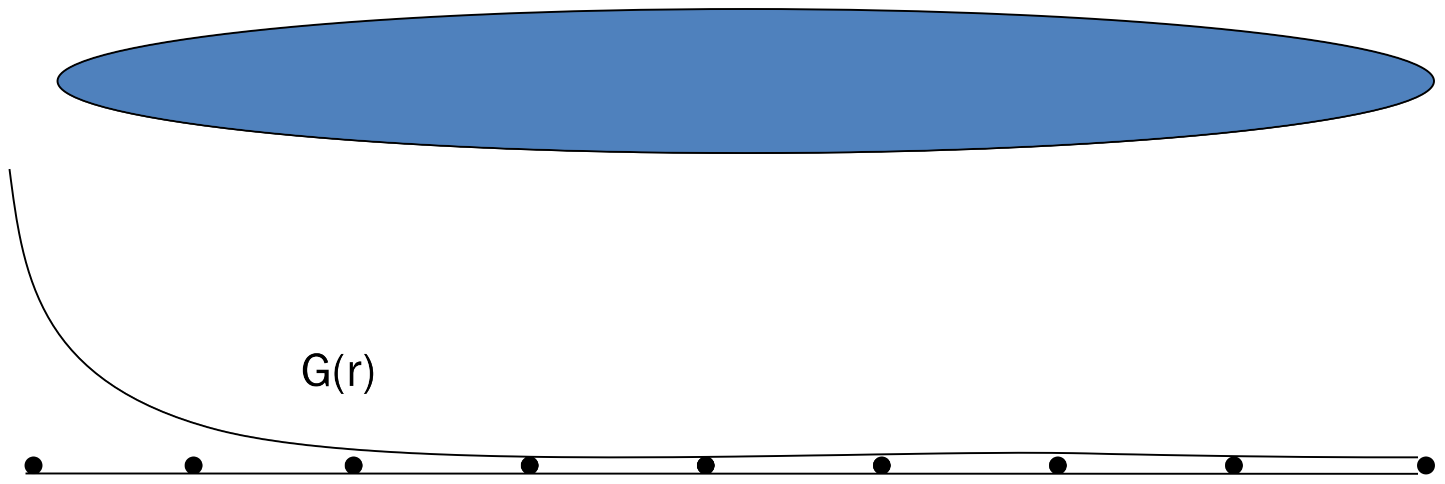
$$G_s(r, r') = G(r + r_s, r')$$

J. Qiang, M. Furman, and R. Ryne, Phys. Rev. ST Accel. Beams, vol 5, 104402 (October 2002).

Test of Image Space-Charge Calculation Numerical Solution vs. Analytical Solution



Integrated Green Function for Large Aspect Ratio Beam



- Lack of resolution along longer side if same number of grids are used for both sides
- Brute force: use more grid points along longer side
- Better way: break the original convolution integral into a sum of small cell integral and use integrated Green's function within each cell

Integrated Green Function

$$\phi_c(r_i) = \sum_{i'=1}^{2N} G_i(r_i - r_{i'})\rho_c(r_{i'})$$

$$\boxed{G_i(r,r') = \oint G_s(r,r')\,dr'}$$

$$\bar{G}_s(x_i - x_{i'}, y_j - y_{j'}) = \int_{x_{i'}-h_x/2}^{x_{i'}+h_x/2} dx' \int_{y_{j'}-h_y/2}^{y_{j'}+h_y/2} dy'\, G_s(x_i - x', y_j - y').$$

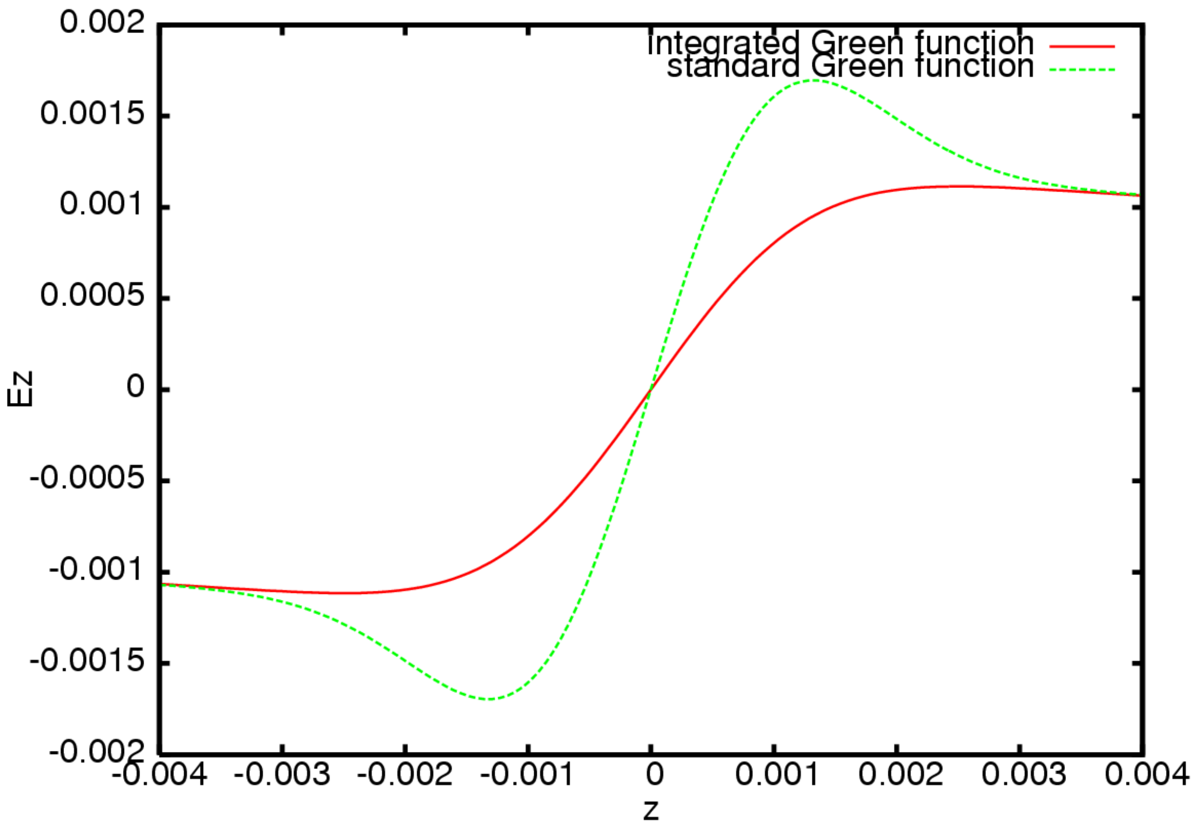
This integration can be done analytically using the indefinite integral:

$$\int\int \ln(x^2+y^2)\,dx\,dy = -3xy + x^2\arctan(y/x) + y^2\arctan(x/y) + xy\ln(x^2+y^2).$$

$$\iiint \frac{1}{\sqrt{x^2+y^2+z^2}}dxdydz \doteq -\frac{z^2}{2}\arctan\left(\frac{xy}{z\sqrt{x^2+y^2+z^2}}\right) - \frac{y^2}{2}\arctan\left(\frac{xz}{y\sqrt{x^2+y^2+z^2}}\right) - \frac{x^2}{2}\arctan\left(\frac{yz}{x\sqrt{x^2+y^2+z^2}}\right)$$
$$+ yz\ln(x+\sqrt{x^2+y^2+z^2}) + xz\ln(y+\sqrt{x^2+y^2+z^2}) + xy\ln(z+\sqrt{x^2+y^2+z^2}). \quad (2)$$

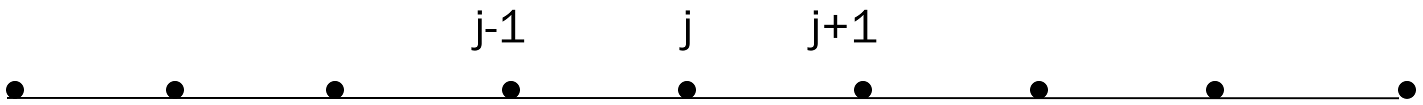
J. Qiang, S. Lidia, R. D. Ryne, and C. Limborg-Deprey, Phys. Rev. ST Accel. Beams, vol 9, 044204 (2006).

A Comparison Example: Aspect Ratio = 30



Poisson Solver with Finite Boundary Conditions

$$\frac{\partial^2 \phi}{\partial x^2} = -\rho$$



$$\frac{\partial^2 \phi}{\partial x^2} = \frac{1}{h_x^2}(\phi_{j+1} - 2\phi_j + \phi_{j-1}) + O(h_x^2)$$

$$\phi_{j+1} - 2\phi_j + \phi_{j-1} = -h_x^2 \rho_j$$

Where j = 1, ...,N

Finite Difference Poisson Solver: Iterative Methods

$$\boxed{A\ X = B}$$

A: is a sparse matrix

- Direct Gaussian elimination: $O(N^3)$
- Iterative method: $O(mN)$

$$\boxed{X = X^i + E}$$

$$\boxed{A\ E = (B - AX^i)}$$

$$\boxed{A = D - L - U}$$

$$\boxed{X^{i+1} = X^i + S(B - AX^i)}$$

Where S is an approximation of A^{-1}, $i = 1,\ldots,m$

Finite Difference Poisson Solver: Iterative Methods

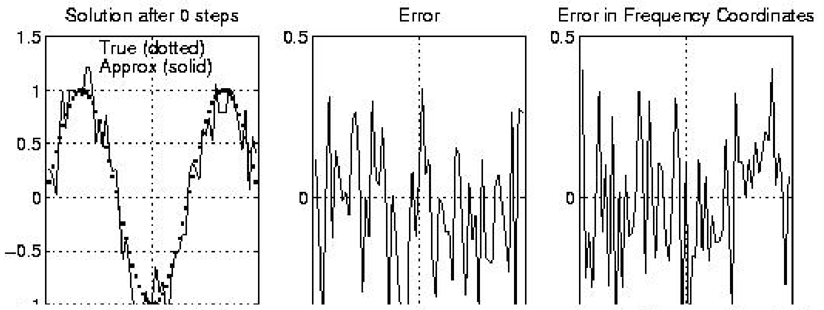
- Classical iterative methods:
  - Jacobi
  - $$\boxed{S = D^{-1}}$$
  - $$\boxed{S = \omega D^{-1}}$$
  - Successive Over Relaxation

$$\boxed{S = (D - L)^{-1}}$$

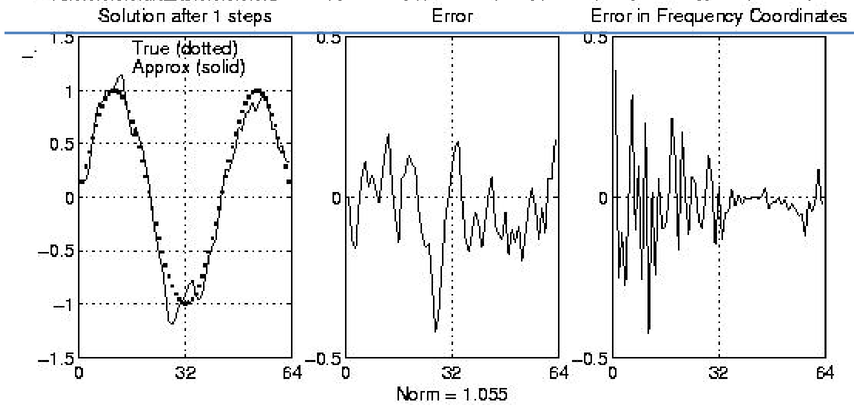
$$\boxed{S = \omega(D - \omega L)^{-1}}$$

- Problems of classical iterative methods:
  - slow convergence

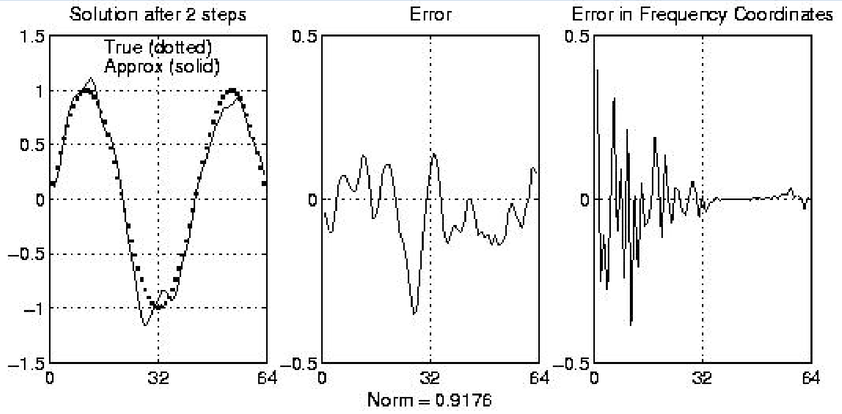
Weighted Jacobi Chosen to Damp High Frequency Error



Initial error
"Rough"
Lots of high frequency components
Norm = 1.65



Error after 1 Jacobi step
"Smoother"
Less high frequency component
Norm = 1.055



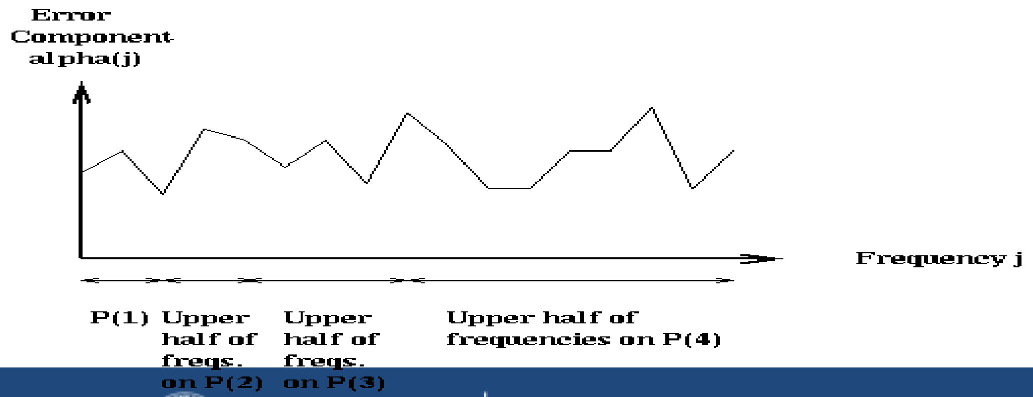
Error after 2 Jacobi steps
"Smooth"
Little high frequency component
Norm = .9176,
won't decrease much more

Ref: J. Demmel's class note

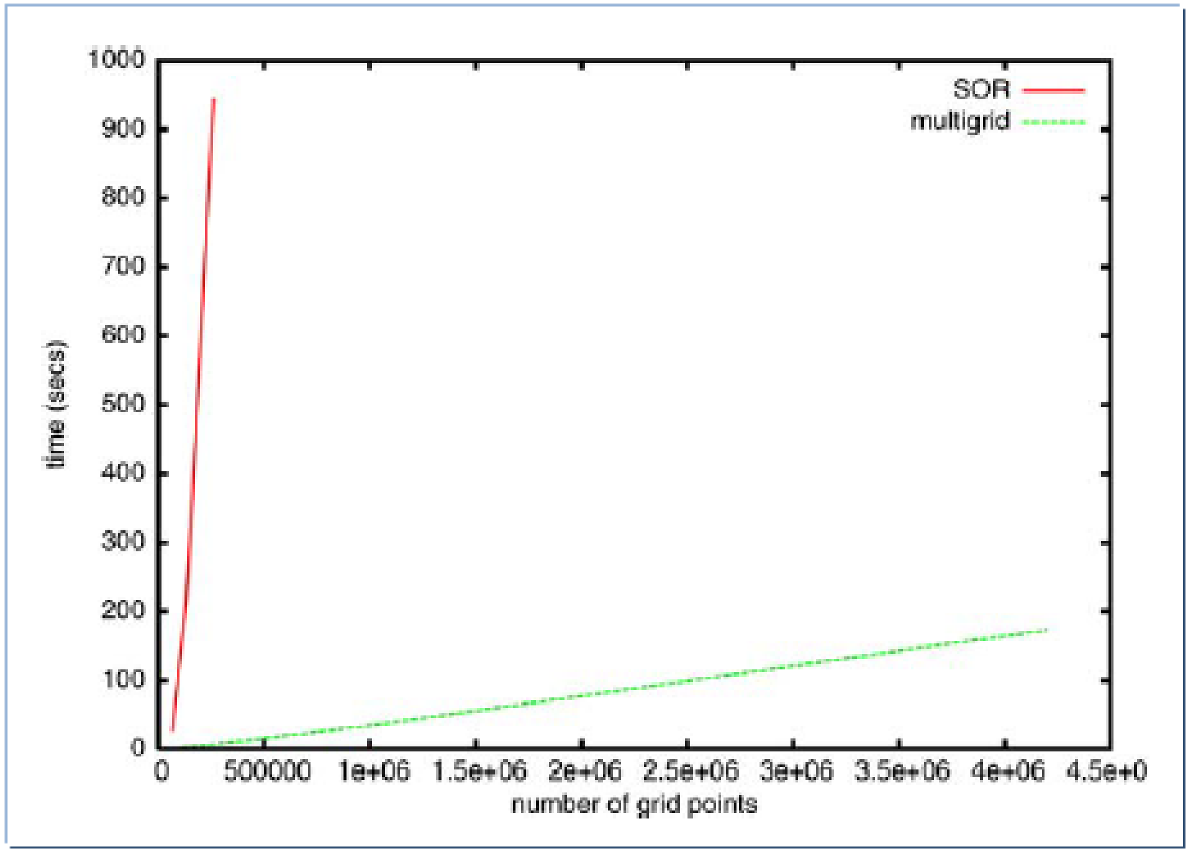
Multigrid Motivation

- Classical sparse-matrix-vector-multiply-based algorithms:
  - low frequency error decreases slowly after a few iterations
  - move information one grid at a time
  - take $N^{1/d}$ steps to get information across grid
  - matrix-vector multiplications are done on the full fine grid
- Multigrid algorithm:
  - smooths out the numerical errors of different frequencies on different scales using multiple grids
  - moves the information across grid by $\Omega(\log n)$ steps
  - most multipications are done on coarse grids

Schematic Description of Multigrid



Comparison of Convergent Time for an Example: SOR vs. Multi-Grid

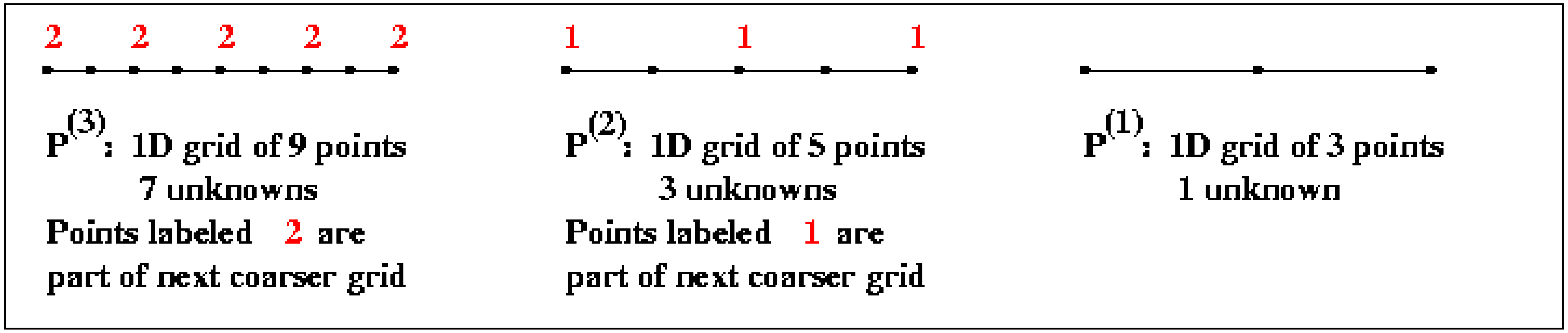


Multi-Grid Iteration Method

- Basic Algorithm:
  - Replace correction problem on fine grid by an approximation on a coarser grid
  - Solve the coarse grid problem approximately, and use the solution as a correction to the fine grid problem and build a new starting guess for the fine-grid problem, which is then iteratively updated
  - Solve the coarse grid problem recursively, i.e. by using a still coarser grid approximation, etc.
- Success depends on coarse grid solution being a good approximation to the fine grid

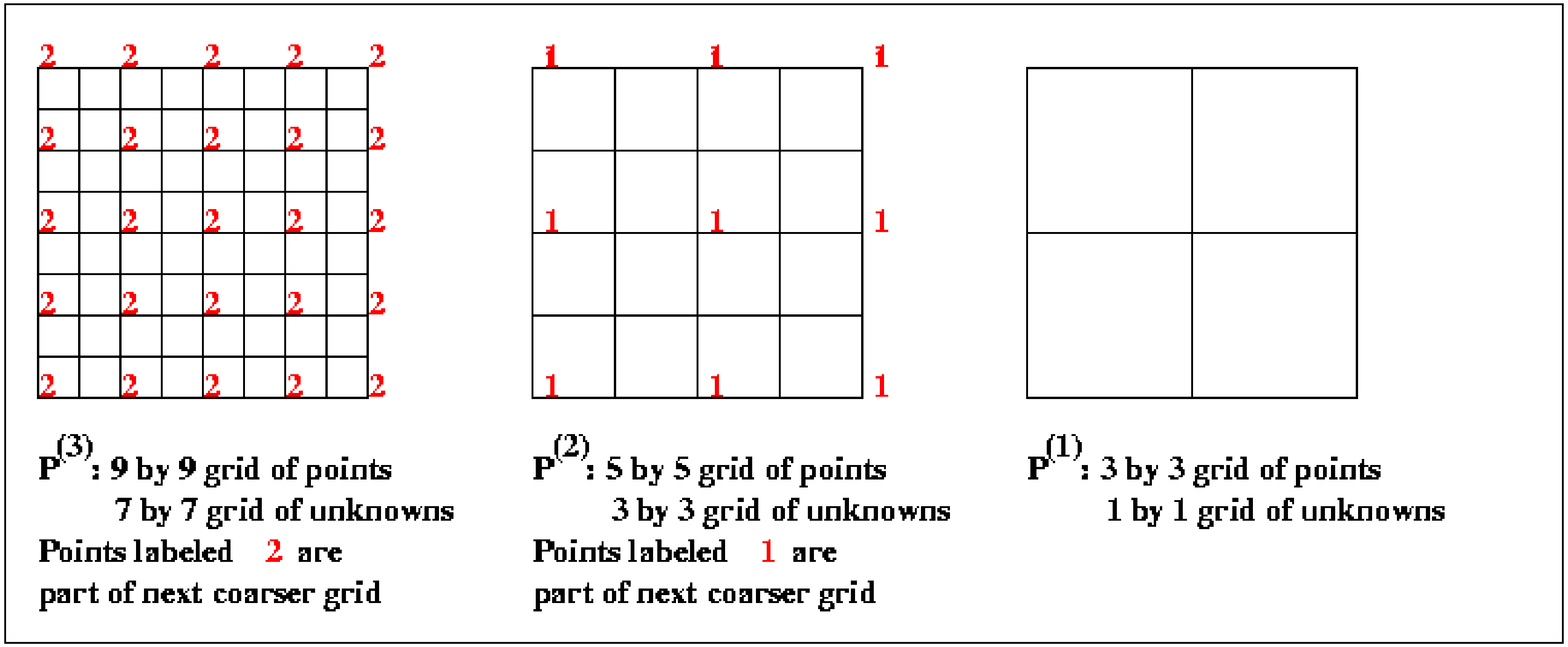
Multigrid Sketch on a Regular 1D Mesh

- Consider a 2^m+1 grid in 1D for simplicity
- Let $P^{(i)}$ be the problem of solving the discrete Poisson equation on a 2^i+1 grid in 1D
- Write linear system as $A(i) * x(i) = b(i)$
- $P^{(m)}$, $P^{(m-1)}$, … , $P^{(1)}$ is a sequence of problems from finest to coarsest


2 2 2 2 2
1 1 1
$P^{(3)}$: 1D grid of 9 points
7 unknowns
Points labeled 2 are
part of next coarser grid
$P^{(2)}$: 1D grid of 5 points
3 unknowns
Points labeled 1 are
part of next coarser grid
$P^{(1)}$: 1D grid of 3 points
1 unknown

Multigrid Sketch on a Regular 2D Mesh

- Consider a 2^m+1 by 2^m+1 grid
- Let $P^{(i)}$ be the problem of solving the discrete Poisson equation on a 2^i+1 by 2^i+1 grid in 2D
    - Write linear system as A(i) * x(i) = b(i)
- $P^{(m)}$, $P^{(m-1)}$, … , $P^{(1)}$ is a sequence of problems from finest to coarsest



$P^{(3)}$: 9 by 9 grid of points
7 by 7 grid of unknowns
Points labeled 2 are part of next coarser grid
$P^{(2)}$: 5 by 5 grid of points
3 by 3 grid of unknowns
Points labeled 1 are part of next coarser grid
$P^{(1)}$: 3 by 3 grid of points
1 by 1 grid of unknowns

Basic Operators of Multigrid Iteration

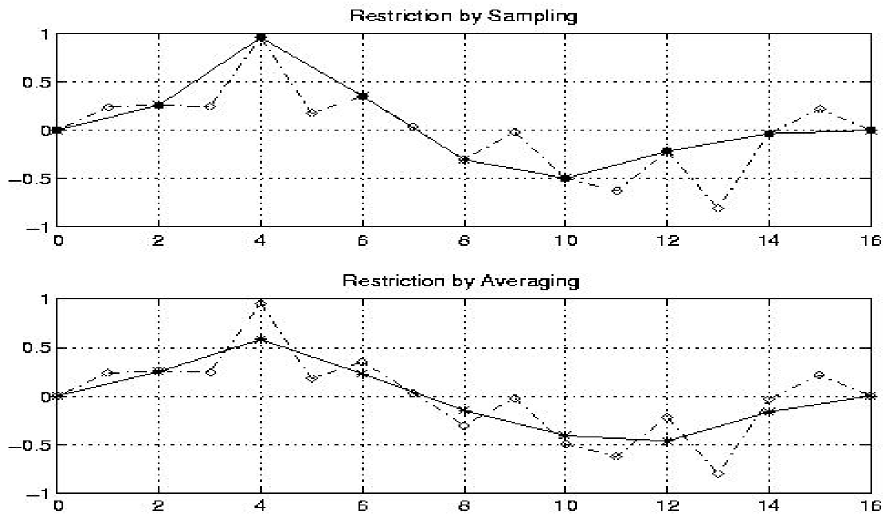
- For problem $P^{(i)}$:
  - b(i) is the RHS and
  - x(i) is the current estimated solution
  - (A(i) is implicit in the operators below.)

$\}$ both live on grids of size 2^i-1

- All the following operators just average values on neighboring grid points
  - Neighboring grid points on coarse problems are far away in fine problems, so information moves quickly on coarse problems
- The restriction operator *R*(i) maps $P^{(i)}$ to $P^{(i-1)}$
  - Restricts problem on fine grid $P^{(i)}$ to coarse grid $P^{(i-1)}$ by sampling or averaging
  - b(i-1)= *R*(i) (b(i))
  - Graphic representation: $\backslash$
- The prolongation operator *P*(i-1) maps an approximate solution x(i-1) to an x(i)
  - Interpolates solution on coarse grid $P^{(i-1)}$ to fine grid $P^{(i)}$
  - x(i) = *P*(i-1)(x(i-1))
  - Graphic representation: /
- The smooth operator *S*(i) takes $P^{(i)}$ and computes an improved solution x(i) on same grid
  - Uses "weighted" Jacobi or Gauss-Seidel
  - x_{improved} (i) = *S*(i) (b(i), x(i))
  - Graph representation: ●

Restriction Operator R(i) - Details

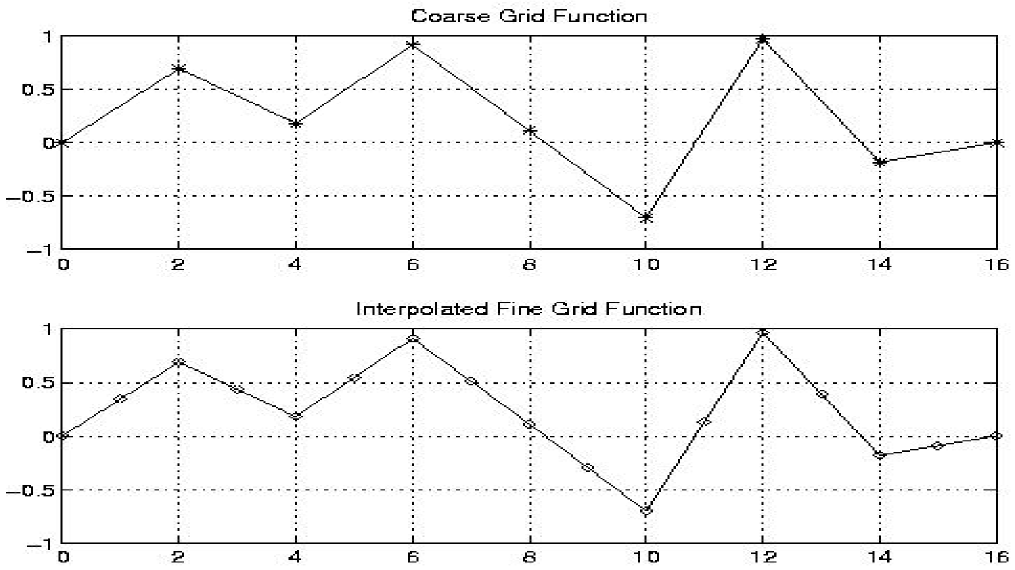
- The restriction operator, R(i), takes
  - a problem $P^{(i)}$ with RHS b(i) and
  - maps it to a coarser problem $P^{(i-1)}$ with RHS b(i-1)
- Simplest way: sampling
- Averaging values of neighbors is better; in 1D this is
  - $x_{coarse}(i) = 1/4 * x_{fine}(i-1) \;+\; 1/2 * x_{fine}(i) \;+\; 1/4 * x_{fine}(i+1)$



- In 2D, average with all 8 neighbors (N,S,E,W,NE,NW,SE,SW)

Prolongation/Interpolation Operator

- The prolongation/interpolation operator *P*(i-1), takes a function on a coarse grid $P^{(i-1)}$, and produces a function on a fine grid $P^{(i)}$
- In 1D, linearly interpolate nearest coarse neighbors
  - ○ $x_{fine}(i) = x_{coarse}(i)$ if the fine grid point i is also a coarse one, else
  - ○ $x_{fine}(i) = 1/2 * x_{coarse}(\text{left of i}) + 1/2 * x_{coarse}(\text{right of i})$



- In 2D, interpolation requires averaging with 2 or 4 nearest neighbors (NW,SW,NE,SE)

Two-Grid Iteration

- Pre-smoothing: compute approximated solution by applying ν_1 steps of a relaxation method on fine grid:

  $$\bar{X}^{i+1}(2) = \bar{X}^{i}(2) + S(B - A\bar{X}^{i}(2));\ i = 1, \cdots, \nu_1$$

- Construct residual vectors:

  $$r(2) = B - A\bar{X}^{\nu_1+1}(2)$$

- Restrict the residual to coarser grid:

  $$r(1) = R(r(2))$$

- Solve exactly on the coarse grid for the error vector:

  $$E(1) = A^{-1}R(r(1))$$

- Prolongate/Interpolate the error vector to fine grid:

  $$E(2) = P(E(1))$$

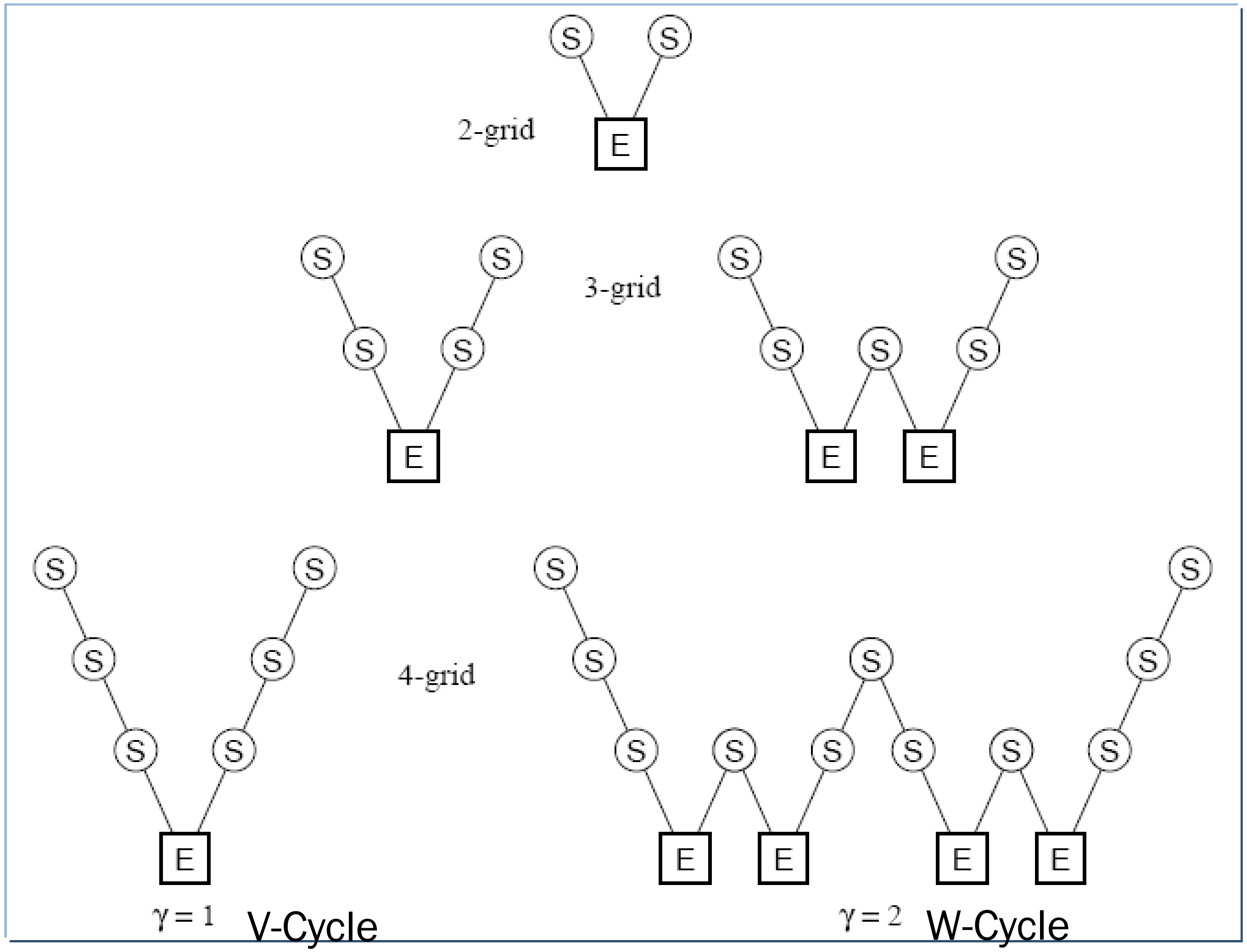
- Compute the improved approximation x on fine grid:

  $$\breve{X}^{1}(2) = \bar{X}^{\nu_1+1}(2) + E(2)$$

- Post-smoothing: compute approximated solution by applying ν_2 steps of a relaxation method on fine grid:

  $$\breve{X}^{i+1}(2) = \breve{X}^{i}(2) + S(B - A\breve{X}^{i}(2));\ i = 1, \cdots, \nu_2$$

Structure of Multigrid Cycles



γ is the number of two-grid iterations at each intermediate stage

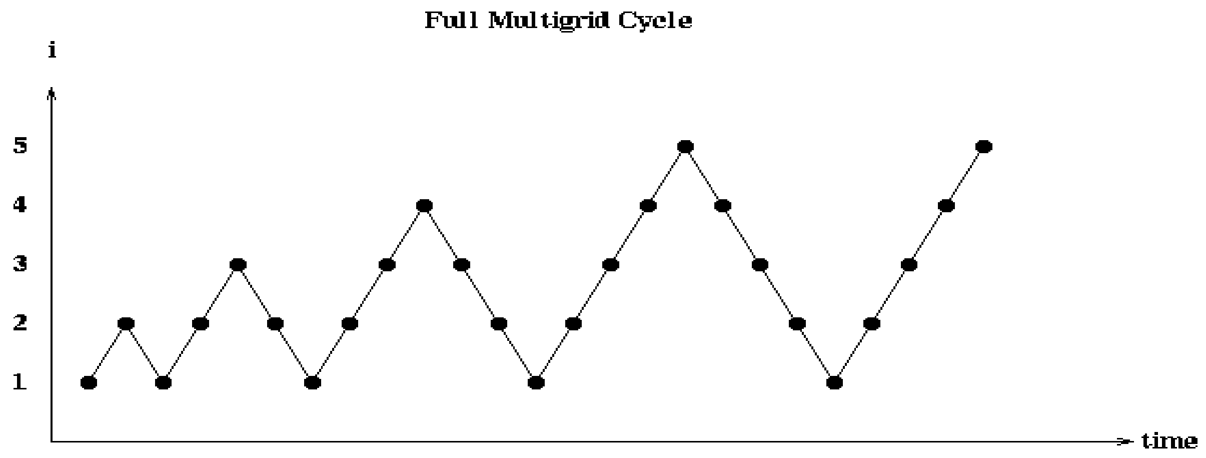
Multigrid V-Cycle Algorithm

```
Function MGV ( b(i), x(i) )
  … Solve A(i)*x(i) = b(i) given b(i) and an initial guess for x(i)
  … return an improved x(i)
  if (i = 1)
     compute exact solution x(1) of $P^{(1)}$          only 1 unknown
     return x(1)
  else
     x(i) = S(i) (b(i), x(i))                 → improve solution by
                                                 damping high frequency error
     r(i)  = A(i)*x(i) - b(i)                 → compute residual
     r(i-1) = R(i)(r(i))                      → restrict from fine to coarser grid
     MGV( r(i-1), e(i-1) )                    → solve A(i)*e(i) = r(i) recursively
     e(i) = P(i-1)(e(i-1))                    → prolongate from coarser grid to fine grid
     x(i) = x(i) - e(i)                       → correct fine grid solution
     x(i) = S(i) ( b(i), x(i) )               → improve solution again
     return x(i)
```

Complexity of a V-Cycle on a 2D Grid

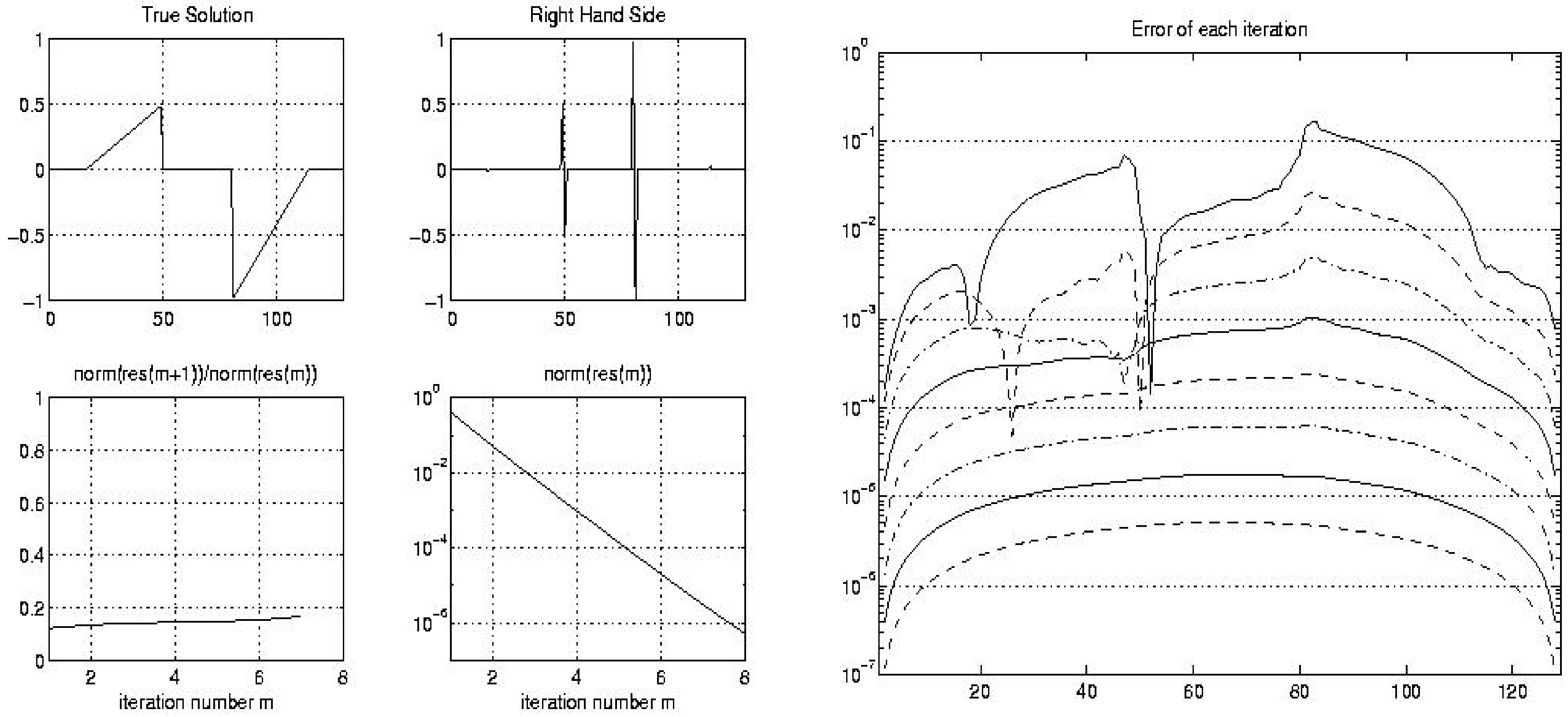
- At level i, the number of unknown is $(2^i-1)\times(2^i-1)$
- On a serial machine
  - Work at each point in a V-cycle is O(the number of unknowns)
  - Cost of Level i is $O((2^i-1)^2) = O(4^i)$
  - If finest grid level is m, total time is:
  - $\sum_{i=1}^{m} O(4^i) = O(4^m) = O(\#\text{ unknowns})$
- On a parallel machine (PRAM)
  - with one processor per grid point and free communication, each step in the V-cycle takes constant time
  - Total V-cycle time is O(m) = O(log #unknowns)

Full/Nested Multigrid (FMG)



- Overview:
    - Solve the problem with 1 unknown on coarsest grid
    - Given a solution to the coarser problem, $P^{(i-1)}$, map it to starting guess for $P^{(i)}$
    - Solve the finer problem using the Multigrid V-cycle
- Advantages:
    - no need for initial guess of solution
    - avoid expensive fine-grid (high frequency) cycles
    - obtain solutions at multiple grid level (can be used for error estimate or extrapolation)

Convergence Picture of Multigrid in 1D



• Error decreases by a factor >5 on each iteration

Particle Advance: Numerical Integration

- Consistency
- Accuracy
- Stability
- Efficiency
- Examples of numerical integrators:
  - Runge-Kutta
  - Leap frog
  - Boris
  - Integrators beyond Boris
  - Symplectic integrators

Numerical Integration: Consistency

1D Example

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = F(x)$$

Euler algorithm:

$$x^{n+1} = x^n + v^n dt$$

$$v^{n+1} = v^n + F(x^n) dt$$

Consistency:

under the limit of dt->0, the discrete model -> continuous model

Numerical Integration: Accuracy

Accuracy: local truncation errors in the numerical discrete algebraic equations compared with the original differential equations

$$\frac{x^{n+1}-x^n}{dt}=\frac{dx}{dt}+\frac{1}{2}\frac{d^2x}{dt^2}dt+O(dt^2)$$

$$\frac{v^{n+1}-v^n}{dt}=\frac{dv}{dt}+\frac{1}{2}\frac{d^2v}{dt^2}dt+O(dt^2)$$

$$\frac{dx}{dt}=v-\frac{1}{2}\frac{d^2x}{dt^2}dt+O(dt^2)$$

$$\frac{dv}{dt}=F(x)-\frac{1}{2}\frac{d^2v}{dt^2}dt+O(dt^2)$$

The above Euler method is the 1st order accuracy.
Higher order accuracy can be obtained using more sub steps.

Numerical Integration: Accuracy

- The order of accuracy can also be expressed as the local truncation error of variables in the numerical integration method.
- The “m^th order method” denotes a numerical integration method that is locally correct through order h^m and makes local errors of order h^{m+1}.

The error for one step is

$$\tilde{\mathbf{e}}(\mathbf{h}) = \tilde{\mathbf{x}}(\mathbf{h}) - \mathbf{x}(\mathbf{0}) = O(h^{m+1})$$

for n step is

$$\tilde{\mathbf{e}}(n\mathbf{h}) = O(h^{m})$$

$$\tilde{x}(h) = \tilde{x}(0) + \int_0^h F(\tilde{x}(t), t)dt$$

$$\tilde{x}(t) = \tilde{x}(0) + \int_0^t F(\tilde{x}(t), t)dt$$

$$F(\tilde{x}(t), t) = F(\tilde{x}(0), 0) + (x - x(0)) F_x + tF_t + \ldots$$

Numerical Integration: Stability

Stability: propagation of errors (e.g. roundoff error) in the discrete algebraic equations . A stable numerical integration algorithm is the one that a small error at any stage does not keep on increasing as number of steps increases

Euler algorithm on computer:

$$X^{n+1} = X^n + V^n dt$$

$$V^{n+1} = V^n + F(X^n) dt$$

$$x = X + e_x$$

$$v = V + e_y$$

Linearized equations of errors:

$$e_x^{n+1} = e_x^n + e_v^n dt$$

$$e_v^{n+1} = e_v^n + F'(x_n) e_x^n dt$$

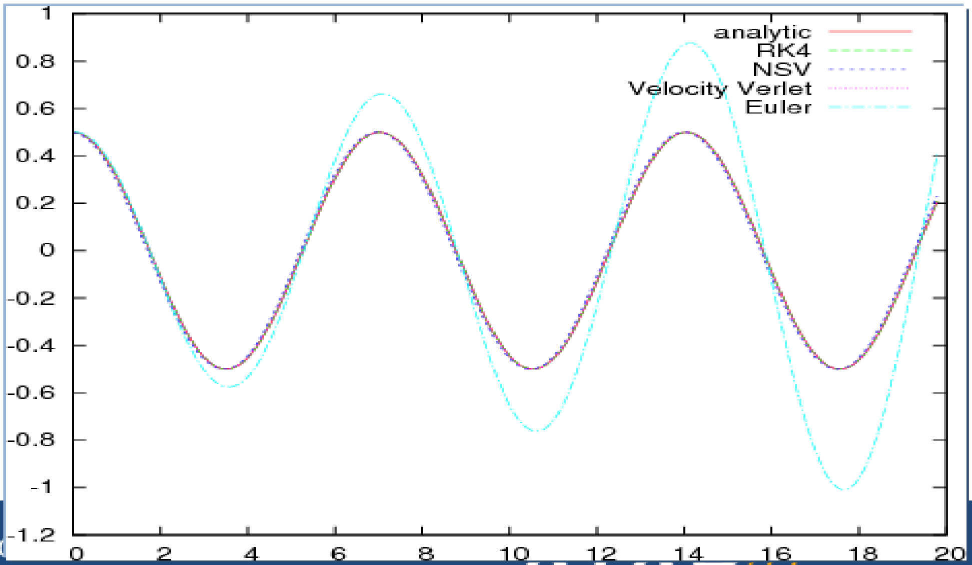
Numerical Integration: Stability

$$\begin{pmatrix} e_x \\ e_v \end{pmatrix}^{n+1} = \begin{pmatrix} 1 & dt \\ F'(x_n)dt & 1 \end{pmatrix} \begin{pmatrix} e_x \\ e_v \end{pmatrix}^{n}$$

For an integration scheme to be numerically stable, the eigenvalues of the error transfer matrix must lie in or on the unit circle.

The explicit Euler method will be unstable

$$\lambda = 1 \pm \sqrt{F'(x_n)}dt$$



Numerical Integrator: Runge-Kutta

$$\boxed{\frac{dx_i}{dt} = f_i(t, x_1, \ldots, x_N)}$$

$$\mathbf{k}_1 = h\mathbf{f}(t_n, \mathbf{x}_\mathbf{n})$$

$$\mathbf{k}_2 = h\mathbf{f}(t_n + \frac{h}{2}, \mathbf{x}_\mathbf{n} + \frac{\mathbf{k}_1}{2})$$

$$\mathbf{k}_3 = h\mathbf{f}(t_n + \frac{h}{2}, \mathbf{x}_\mathbf{n} + \frac{\mathbf{k}_2}{2})$$

$$\mathbf{k}_4 = h\mathbf{f}(t_n + h, \mathbf{x}_\mathbf{n} + \mathbf{k}_3)$$

$$\mathbf{x}_\mathbf{n+1} = \mathbf{x}_\mathbf{n} + \frac{\mathbf{k}_1}{6} + \frac{\mathbf{k}_2}{3} + \frac{\mathbf{k}_3}{3} + \frac{\mathbf{k}_4}{6} + O(h^5)$$

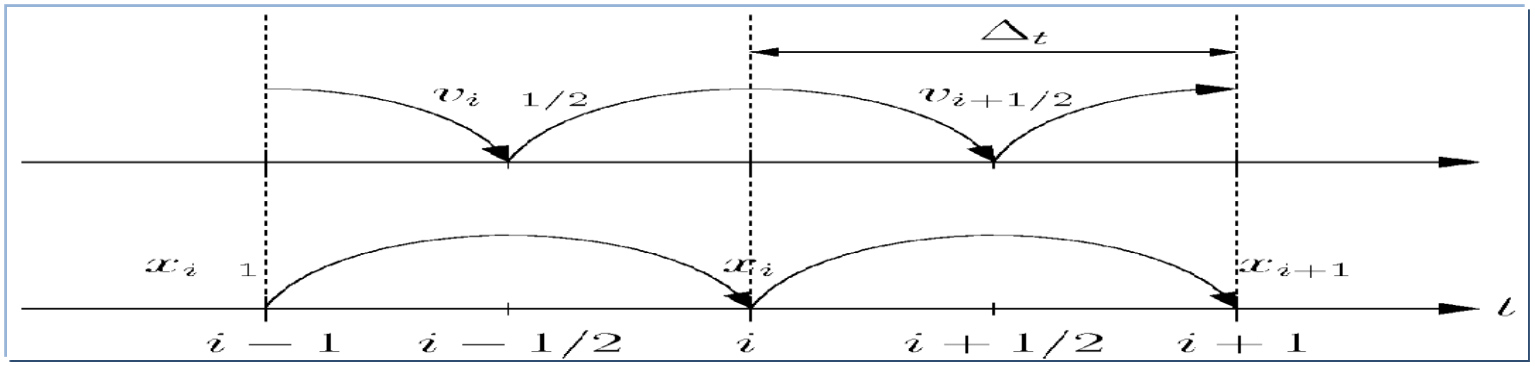
- “one of the most popular schemes for integrating ODEs”
- applicable to arbitrary ODEs
- 4th order accuracy
- variable time step size
- auxiliary storage
- 4 field calculations per step

Numerical Integrator: Leap-Frog

$$\boxed{\begin{aligned}\frac{d\mathbf{x}}{dt} &= \mathbf{v}\\ \frac{d\mathbf{v}}{dt} &= \mathbf{f}(\mathbf{x})\end{aligned}}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^{n} + h\mathbf{v}^{n+\frac{1}{2}}$$

$$\mathbf{v}^{n+\frac{1}{2}} = \mathbf{v}^{n-\frac{1}{2}} + h\mathbf{f}(\mathbf{x}^{n})$$



- easy to implement, low memory storage
- 2nd order accuracy
- time reversible
- stable for $h < \frac{2}{\Omega};\ \Omega = \left(|\frac{df}{dx}|_{\max}\right)^{1/2}$
- single field calculation per step

Numerical Integrator: Boris Algorithm

Lorentz equations of motion

$$\boxed{\begin{aligned}\frac{d\mathbf{x}}{dt} &= \mathbf{v}\\ m\frac{d\mathbf{v}}{dt} &= q(\mathbf{E}+\mathbf{v}\times\mathbf{B})\end{aligned}}$$

Boris Algorithm

$$\mathbf{v}^{n-\frac{1}{2}} = \mathbf{v}^- - \frac{q\mathbf{E}}{m}\frac{h}{2}$$

$$\mathbf{v}^{n+\frac{1}{2}} = \mathbf{v}^+ + \frac{q\mathbf{E}}{m}\frac{h}{2}$$

$$\mathbf{v}^+ - \mathbf{v}^- = \frac{qh}{2m}(\mathbf{v}^+ + \mathbf{v}^-)\times\mathbf{B}$$

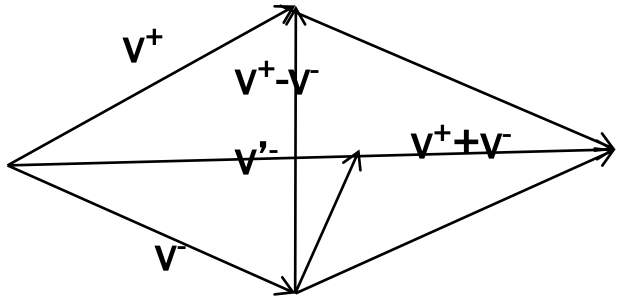
$$\tan(\frac{\theta}{2}) = \frac{qB}{2m}h$$

$$\mathbf{t} = \frac{q\mathbf{B}}{m}\frac{h}{2}$$

$$\mathbf{v}' = \mathbf{v}^- + \mathbf{v}^- \times \mathbf{t}$$

$$\mathbf{s} = \frac{2\mathbf{t}}{1+t^2}$$

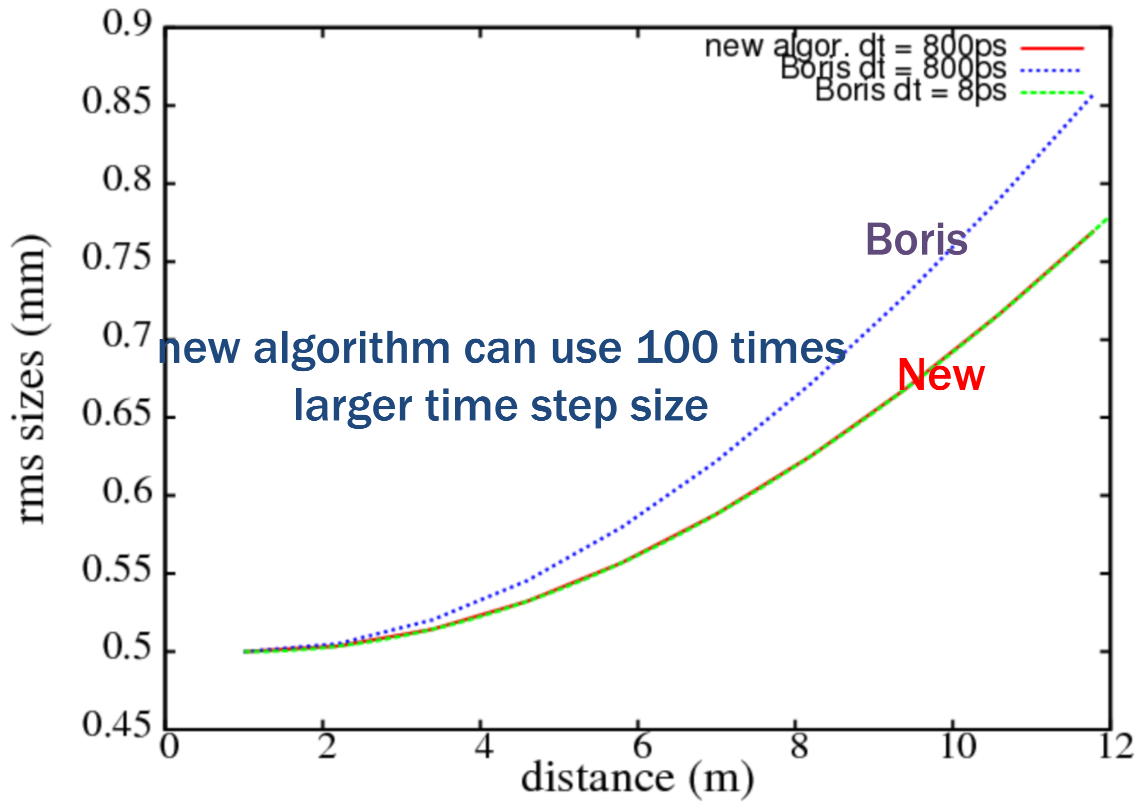
$$\mathbf{v}^+ = \mathbf{v}^- + \mathbf{v}'\times\mathbf{s}$$



- widely used in plasma/accelerator community
- 2nd order accuracy
- time reversible

New Numerical Integrator Needed to Include Space-Charge Fields in Relativistic Electron Beam

- 50 MeV electron beam transports in free space



Fast Numerical Integrator for Relativistic Charged Particle Tracking

Lorentz Force Equations:

$$\frac{d\mathbf{r}}{dt} = \frac{\mathbf{p}c}{\gamma}$$
$$\frac{d\mathbf{p}}{dt} = q\left(\frac{\mathbf{E}}{mc} + \frac{1}{m\gamma}\mathbf{p} \times \mathbf{B}\right)$$

$$\mathbf{r} = (x, y, z)$$
$$\mathbf{p} = (p_x/mc, p_y/mc, p_z/mc)$$

Widely Used Boris Integrator

$$\begin{aligned}
\mathbf{p}_- &= \mathbf{p}(0) + \frac{q\mathbf{E}\tau}{2mc} \\
\gamma_- &= \sqrt{1 + \mathbf{p}_- \cdot \mathbf{p}_-} \\
\mathbf{p}_+ - \mathbf{p}_- &= (\mathbf{p}_+ + \mathbf{p}_-) \times \frac{q\mathbf{B}\tau}{2m\gamma_-} \\
\mathbf{p}(\tau) &= \mathbf{p}_+ + \frac{q\mathbf{E}\tau}{2mc}
\end{aligned}$$

J. Boris, in Proceedings of the Fourth Conference on the Numerical Simulation of Plasmas (Naval Research Laboratory, Washington, DC, 1970), pp. 367.

New Fast Relativistic Integrator

$$\begin{aligned}
\mathbf{p}_- &= \mathbf{p}(0) + \frac{q}{mc}(\mathbf{E} + \mathbf{v}(0) \times \mathbf{B})\tau \\
\mathbf{v}_+ &= \frac{\mathbf{v}(0) + \mathbf{v}_-}{2} \\
\mathbf{p}(\tau) &= \mathbf{p}(0) + \frac{q}{mc}(\mathbf{E} + \mathbf{v}_+ \times \mathbf{B})\tau
\end{aligned}$$

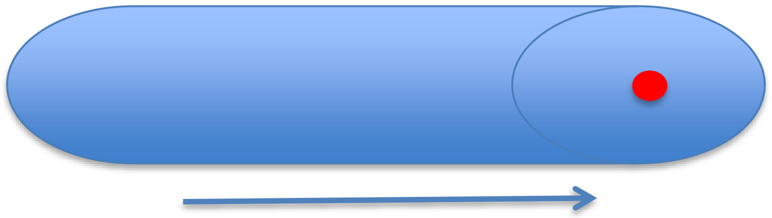
J. Qiang, Nucl. Intrum. Meth. Phys. Res. A, 2017.

Another Relativistic Integrator (Vay)

$$
\begin{aligned}
\gamma_0 &= \sqrt{1+\mathbf{p}\cdot\mathbf{p}} \\
\mathbf{p}_- &= \mathbf{p}(0) + \frac{q\tau}{2mc}(\mathbf{E} + c\mathbf{p}/\gamma_0 \times \mathbf{B}) \\
\mathbf{p}_+ &= \mathbf{p}_- + \frac{q\mathbf{E}\tau}{2mc} \\
\gamma_1 &= \sqrt{1+\mathbf{p}_+\cdot\mathbf{p}_+} \\
\mathbf{t} &= \frac{q\mathbf{B}\tau}{2m} \\
\lambda &= \mathbf{p}_+\cdot\mathbf{t} \\
\sigma &= \gamma_1^2 - \mathbf{t}\cdot\mathbf{t} \\
\gamma_2 &= \sqrt{\frac{\sigma+\sqrt{\sigma^2+4(\mathbf{t}\cdot\mathbf{t}+\lambda^2)}}{2}} \\
\mathbf{t}^* &= \mathbf{t}/\gamma_2 \\
s &= 1/(1+\mathbf{t}^*\cdot\mathbf{t}^*) \\
\mathbf{p}(\tau) &= s[\mathbf{p}_+ + (\mathbf{p}_+\cdot\mathbf{t}^*)\mathbf{t}^* + \mathbf{p}_+\times\mathbf{t}^*]
\end{aligned}
$$

J. V. Vay, Phys. Plasmas 15, 056701 (2008).

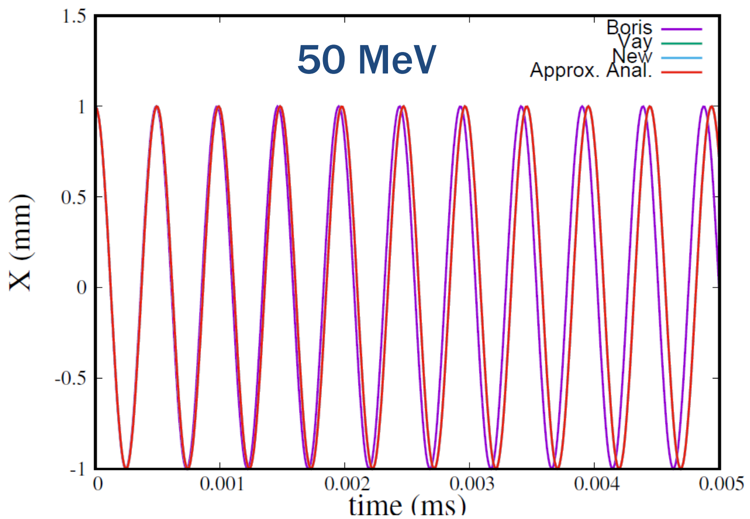
Numerical Test of an Electron Co-Moving with a Positron Coasting Beam with Uniform Transverse Density

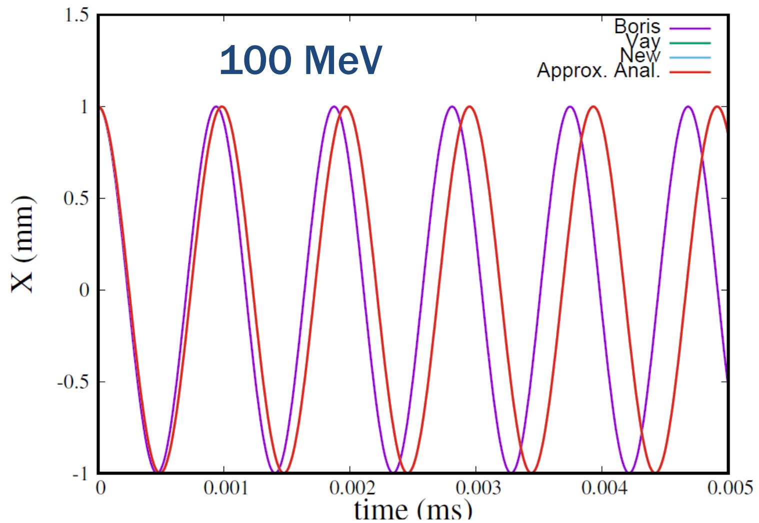


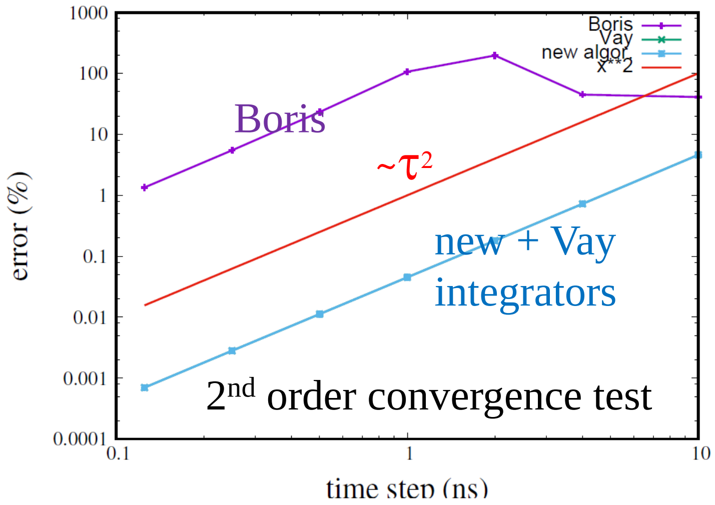
space-charge electric and magnetic fields from positron beam:

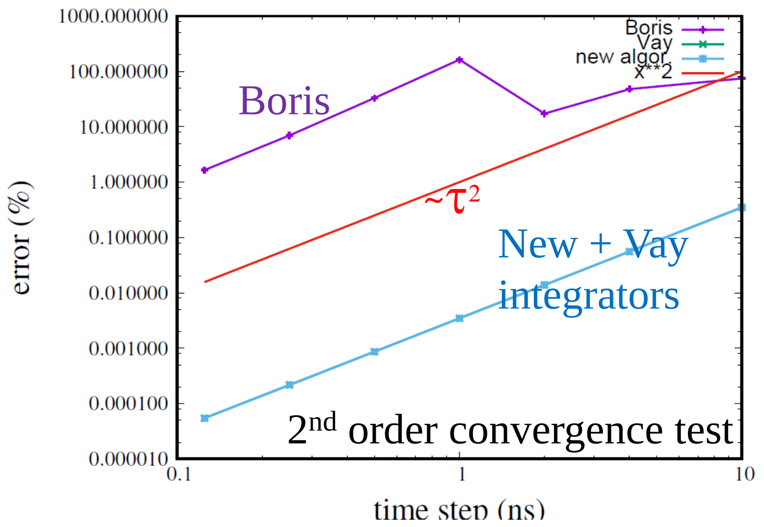
$$
\begin{aligned}
E_x &= E_0 x \gamma_0 \\
E_y &= E_0 y \gamma_0 \\
E_z &= 0 \\
B_x &= -E_0 y \gamma_0 \beta_0 / c \\
B_y &= E_0 x \gamma_0 \beta_0 / c \\
B_z &= 0
\end{aligned}
$$

Numerical Examples Show 2nd Order Accuracy of the New Fast Algorithms in Space-Charge Fields









- All three algorithms show 2nd order accuracy
- Boris algorithm shows much larger error than the other two algorithms

Numerical Integrator: Symplectic Integrator

Hamiltonian equations of motion:

$$\boxed{\begin{aligned}\frac{d\mathbf{x}}{dt} &= \frac{\partial \mathrm{H}}{\partial \mathbf{p}}\\ \frac{d\mathbf{p}}{dt} &= -\frac{\partial \mathrm{H}}{\partial \mathbf{x}}\end{aligned}}$$

Symplectic Integrator preserve:

- the symplectic nature of Hamlitonian equations
- the phase space structure

Numerical integrator:

$$\boldsymbol{\xi}^{n+1} = f(\boldsymbol{\xi}^{n})$$

Define its Jacobian matrix M:

$$M_{ij} = \frac{\partial \xi_i^{n+1}}{\partial \xi_j^{n}}$$

Symplectic condition:

$$M^t J M = J$$

This corresponds to more constraints

$$J = \begin{pmatrix} J_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & J_1 \end{pmatrix} \qquad J_1 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Summary

- Deposition/interpolation in particle-in-cell method
- Self-consistent space-charge field calculations through solving the Poisson equation at each time step using the updated charge density distribution
  - FFT based Green function methods for open boundary condition
  - Spectral (and finite) methods for regular shape boundary condition
- Numerical integrators for particle advance
  - Euler method
  - Runge Kutta method
  - Leap-frog method
  - Boris method
  - Relativistic particle integrator
  - Symplectic integrator