Computational Methods for Kinetic Processes in Plasma Physics

Ken Nishikawa

Department of Physics/UAH

Basics of PIC Simulation methods

- Collisionless plasmas
- Finite-size particles
- Electrostatic codes
- Charge assignment and force interpolation (already in 3-D system)
- Filtering action of shape function
- Summary

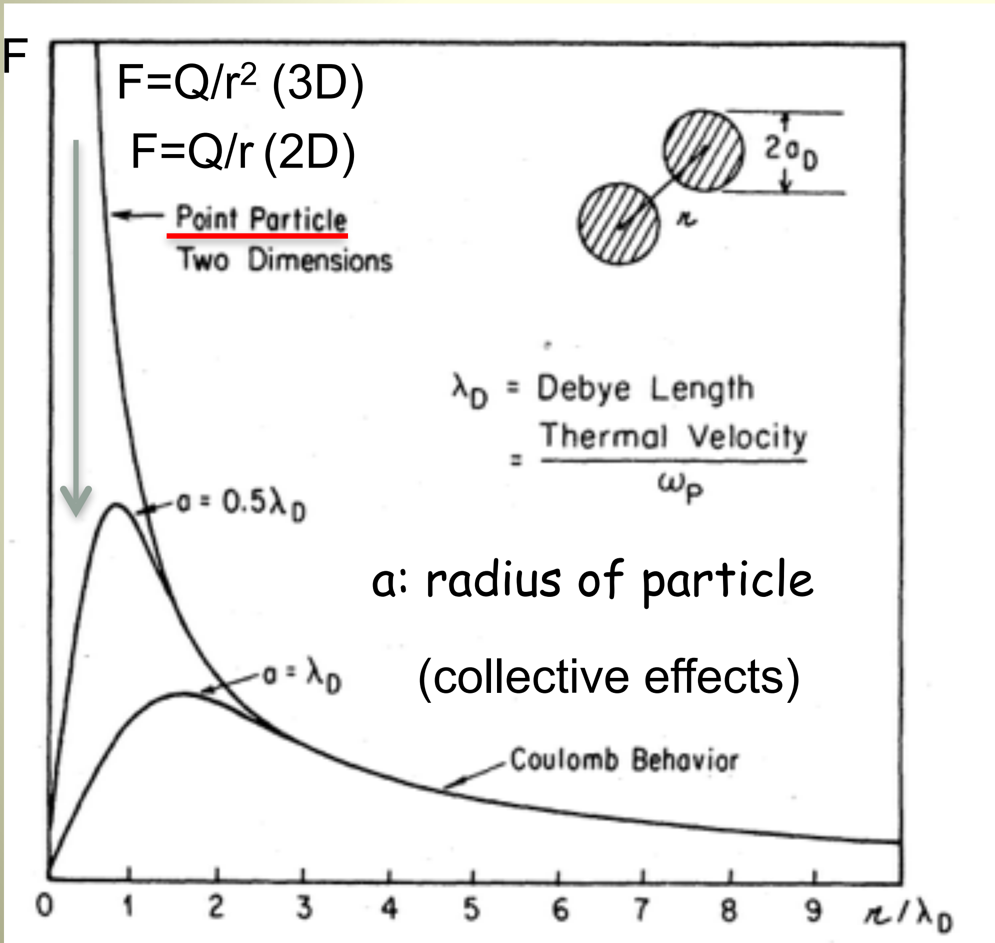
June 1, 2015

Context

- Collisionless plasmas
- Electrostatic codes
- Charge assignment and force interpolation and Integration of equations
- Filtering action
- Aliasing and Reducing noises
- Collisions
- Finite-size particle effects
- Restriction of time step and grid size
- Accuracy and stability of the time integration
- Current deposition seven-boundary move
- Current deposition ten-boundary move

Coulomb force and force law

Coulomb collision is reduced using finite-size particles

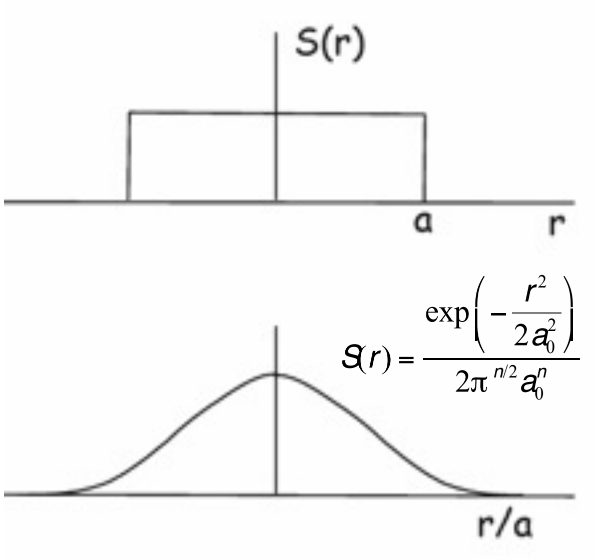


a: radius of particle

(collective effects)

$$S(r) = \frac{1}{V_n a^n}, \quad r < a$$

$$S(r) = 0, \quad r > a$$



$$S(r) = \frac{\exp\left(-\frac{r^2}{2a_0^2}\right)}{2\pi^{n/2} a_0^n}$$

(Dawson 1983)

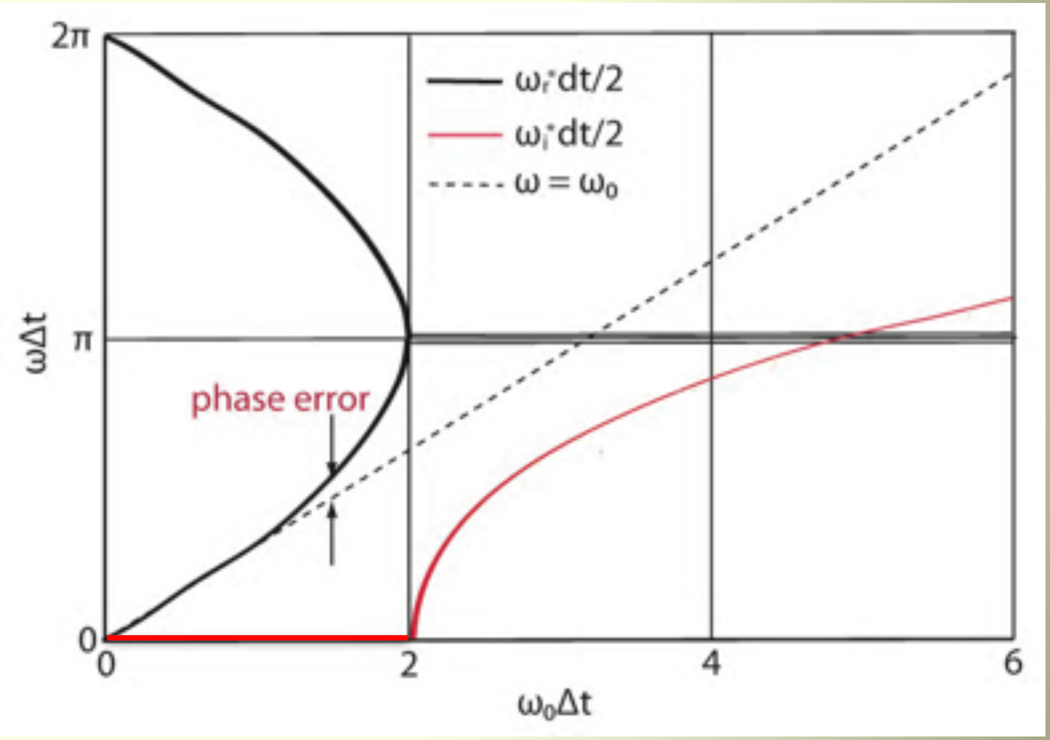
Particle mover accuracy: Simple harmonic motion test

$$\frac{d^2 x}{dt^2} = -\omega_0^2 x$$

$$x = Ae^{-i\omega t}$$

$$\frac{x^{t+\Delta t} - 2x^t + x^{t-\Delta t}}{dt^2} = -\omega_0^2 x^t$$

$$\sin\left[\omega \frac{\Delta t}{2}\right] = \pm\omega_0 \frac{\Delta t}{2}$$



Two major methods of calculating current

1. Spectral method (UPIC code) (note by Decyk)
   We will review this method in details later after we do handout exercises

2. Charge-conserving current deposit (Villasenor & Buneman 1992)
   We will review this method with Umeda's method later

Electrostatic codes

Time scales of the system >> light crossing time, static magnetic field

$$\nabla^2 \phi = -\rho(x)$$

$$E(x) = -\nabla\phi$$

$$F_i = q_i E(x_i)$$

Four major criteria to choose an Algorism
for integration of equation of motion

- **Convergence:** the numerical solution converges to the exact solution of the differential equation in the limit of Δt and Δx tend to zero
- **Accuracy:** the truncation error associated with approximating derivatives with differences
- **Stability:** depends on how total errors (including truncation error and round-off errors) grows in time
- **Efficiency:** the code needs to be efficient to handle large number of particles

Need asses two physical quantities to know how well the codes work

- **Dissipation:** The truncation error associated with approximating derivatives with differences causes the dissipation of some physical quantities
- **Conservation:** The truncation error also causes the deviation of the conservation law

Integration of equations of motion

The simple second order leapfrog achieves the best balanced between accuracy stability, and efficiency

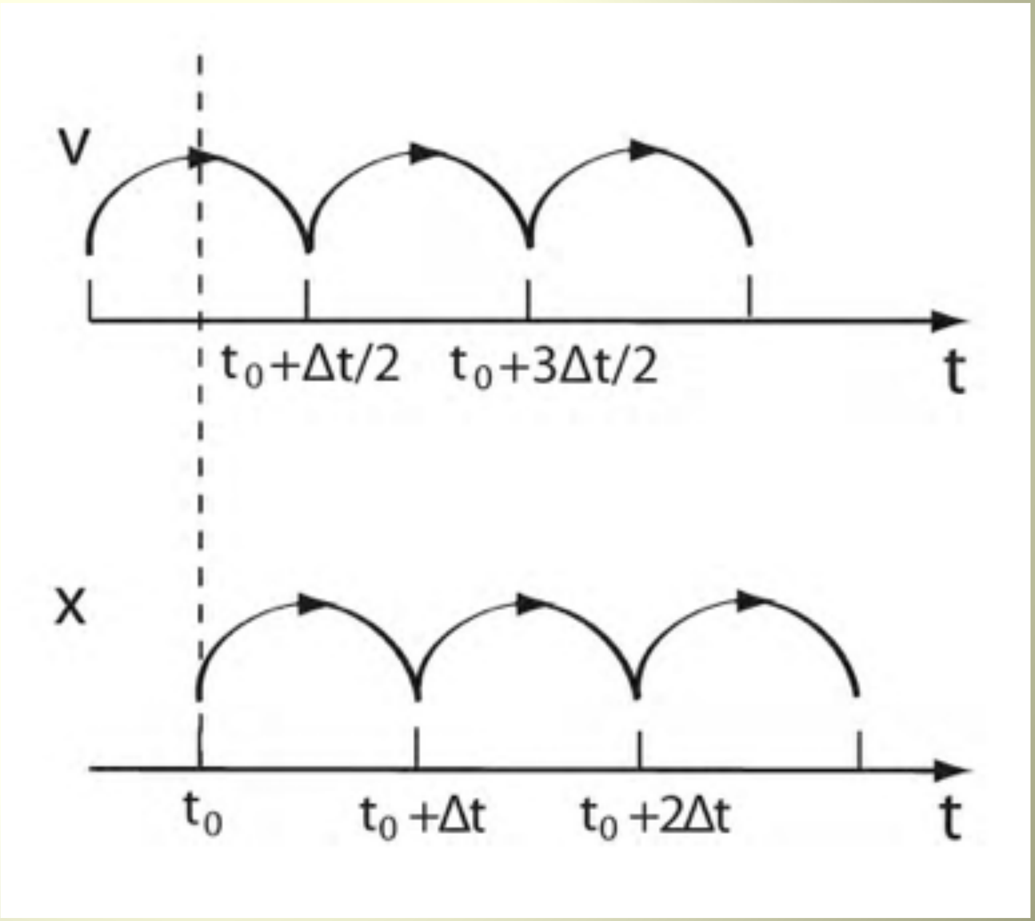
$$\frac{dx_i}{dt} = v_i$$

$$\frac{dv_i}{dt} = \frac{F_i}{m_i}$$

$$\downarrow$$

$$m_i \frac{v_i^{n+1/2} - v_i^{n-1/2}}{\Delta t} = F_i^n$$

$$\frac{x_i^{n+1} - x_i^n}{\Delta t} = v_i^{n+1/2}$$



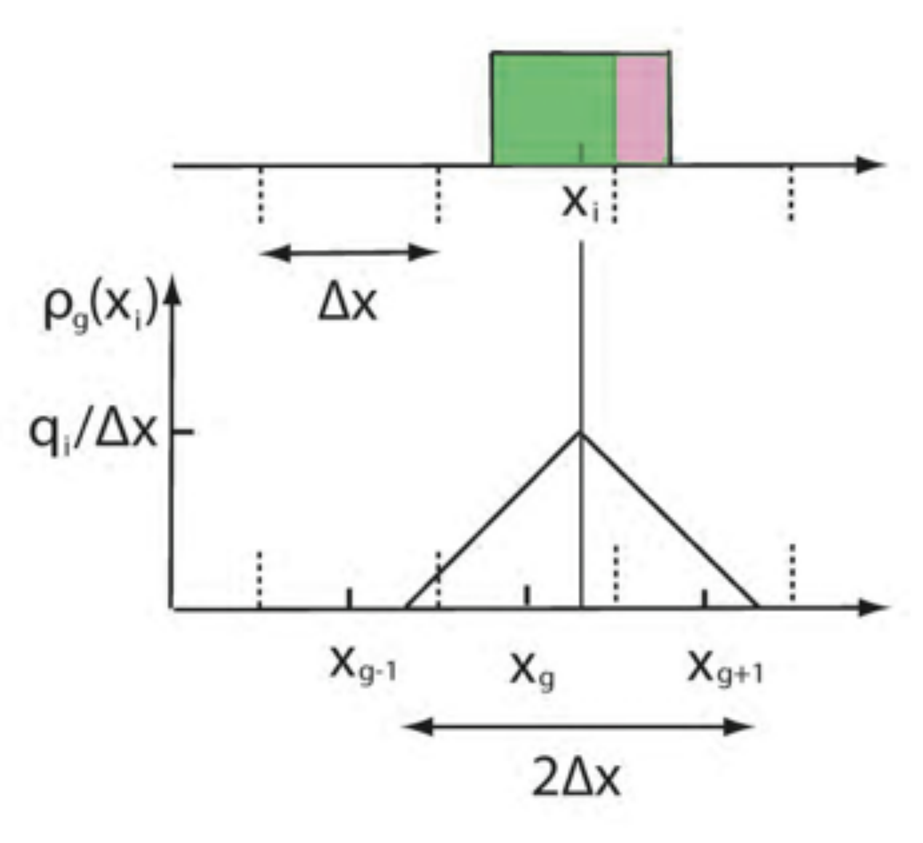
Charge assignment and force evaluation by cloud-in-cell in 1D

As assigned in 3D system the same interpolation scheme is used in 1D

$$\rho_g = \rho_i \frac{x_{g+1} - x_i}{\Delta x}$$

$$\rho_{g+1} = \rho_i \frac{x_i - x_g}{\Delta x}$$

$$F_x = q_i\left(\frac{x_{g+1} - x_i}{\Delta x}E_g + \frac{x_i - x_g}{\Delta x}E_{g+1}\right)$$



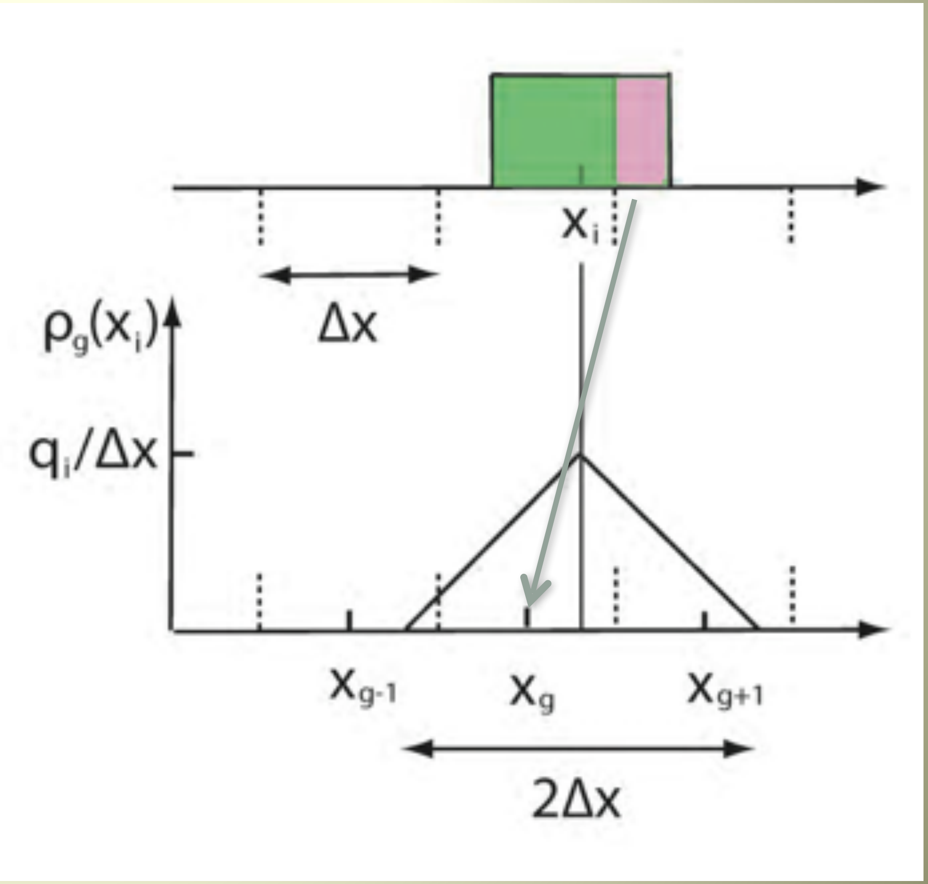
Charge assignment and force evaluation by cloud-in-cell in 1D

As assigned in 3D system the same interpolation scheme is used in 1D

$$\rho_g = \rho_i \frac{x_{g+1} - x_i}{\Delta x}$$

$$\rho_{g+1} = \rho_i \frac{x_i - x_g}{\Delta x}$$

$$F_x = q_i \left(\frac{x_{g+1} - x_i}{\Delta x} E_g + \frac{x_i - x_g}{\Delta x} E_{g+1} \right)$$



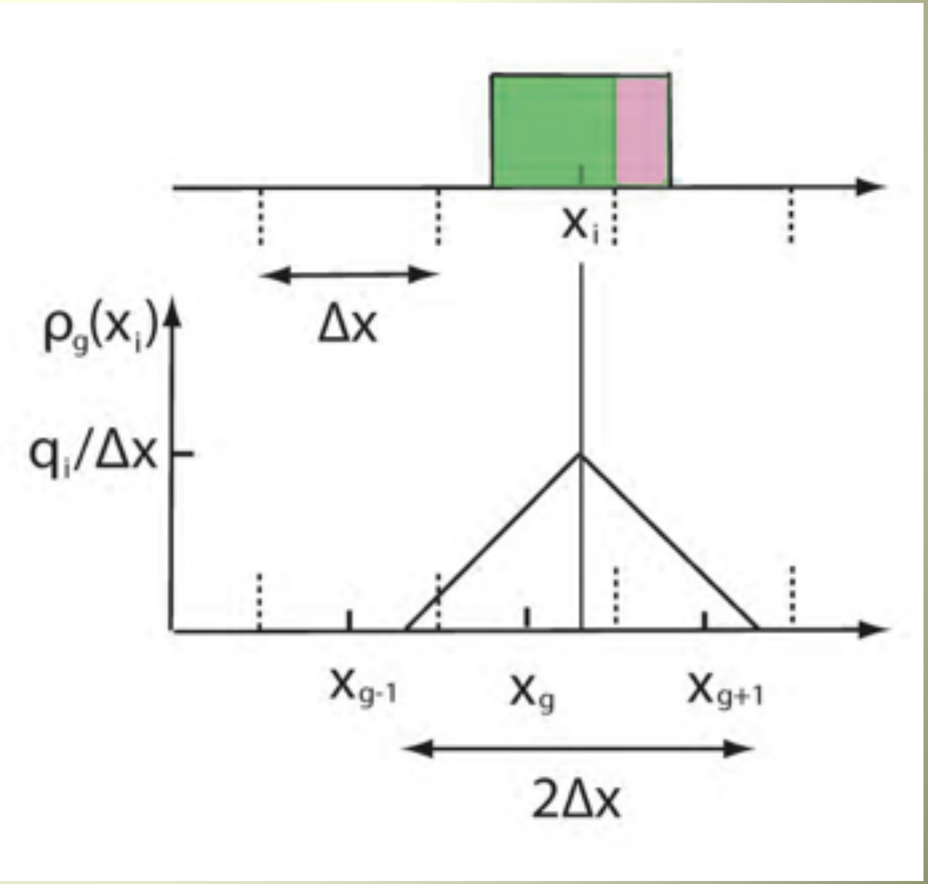
Charge assignment and force evaluation by cloud-in-cell in 1D

As assigned in 3D system the same interpolation scheme is used in 1D

$$\rho_g = \rho_i \frac{x_{g+1} - x_i}{\Delta x}$$

$$\rho_{g+1} = \rho_i \frac{x_i - x_g}{\Delta x}$$

$$F_x = q_i \left(\frac{x_{g+1} - x_i}{\Delta x} E_g + \frac{x_i - x_g}{\Delta x} E_{g+1} \right)$$



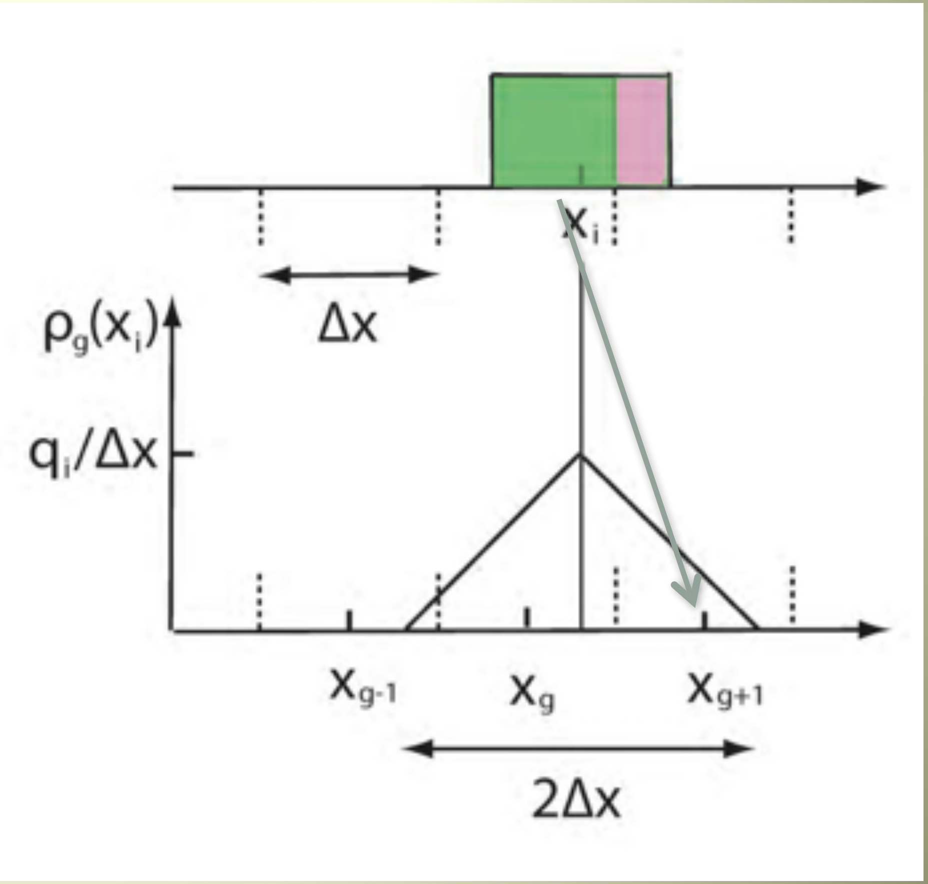
Charge assignment and force evaluation by cloud-in-cell in 1D

As assigned in 3D system the same interpolation scheme is used in 1D

$$\rho_g = \rho_i \frac{x_{g+1} - x_i}{\Delta x}$$

$$\rho_{g+1} = \rho_i \frac{x_i - x_g}{\Delta x}$$

$$F_x = q_i \left(\frac{x_{g+1} - x_i}{\Delta x} E_g + \frac{x_i - x_g}{\Delta x} E_{g+1} \right)$$



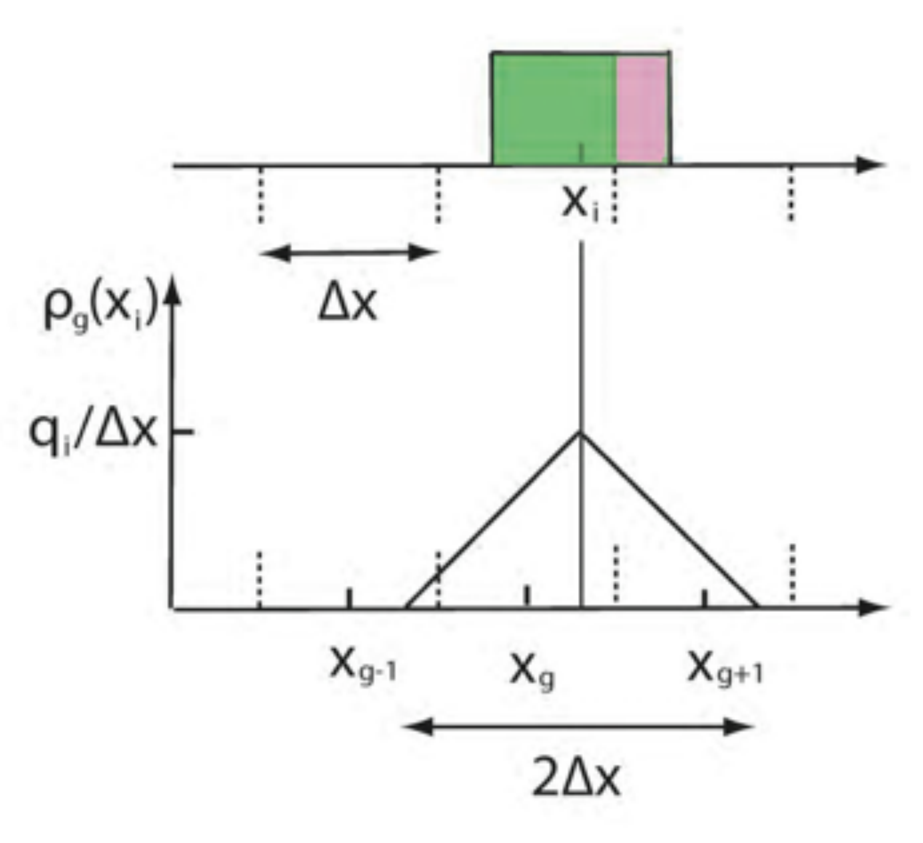
Charge assignment and force evaluation by cloud-in-cell in 1D

As assigned in 3D system the same interpolation scheme is used in 1D

$$\rho_g = \rho_i \frac{x_{g+1} - x_i}{\Delta x}$$

$$\rho_{g+1} = \rho_i \frac{x_i - x_g}{\Delta x}$$

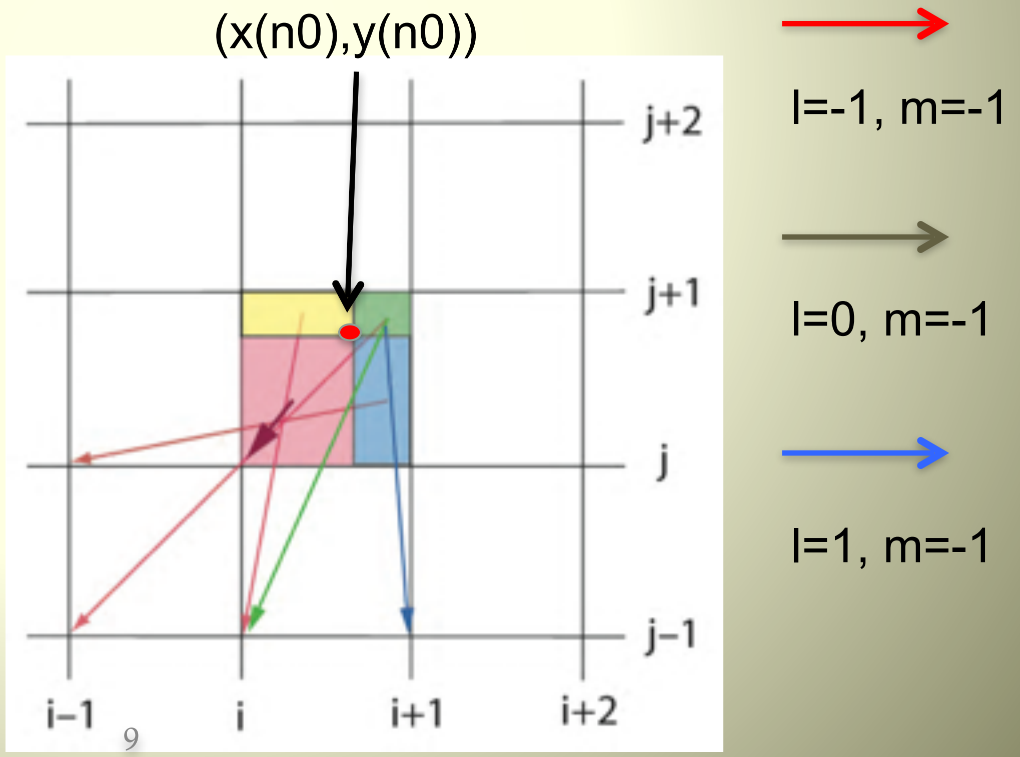
$$F_x = q_i \left(\frac{x_{g+1} - x_i}{\Delta x} E_g + \frac{x_i - x_g}{\Delta x} E_{g+1} \right)$$



Density assignment in 3D system (2D)

```
c for electrons
      do 3 n0=1,lecs
      i=x(n0)
      dx=x(n0)-i
      cx=1.-dx
      j=y(n0)
      dy=y(n0)-j
      cy=1.-dy
      k=z(n0)
      dz=z(n0)-k
      cz=1.-dz
C  Smoothing with the (.25,.5,.25)
profile in each dimension:
      sl=.5
      do 121 l=-1,1
      sl=.75-sl
      sr=.5
      do 121 m=-1,1
      sr=.75-sr
      sn=.5
      do 121 n=-1,1
      sn=.75-sn
      s=sl*sr*sn
```

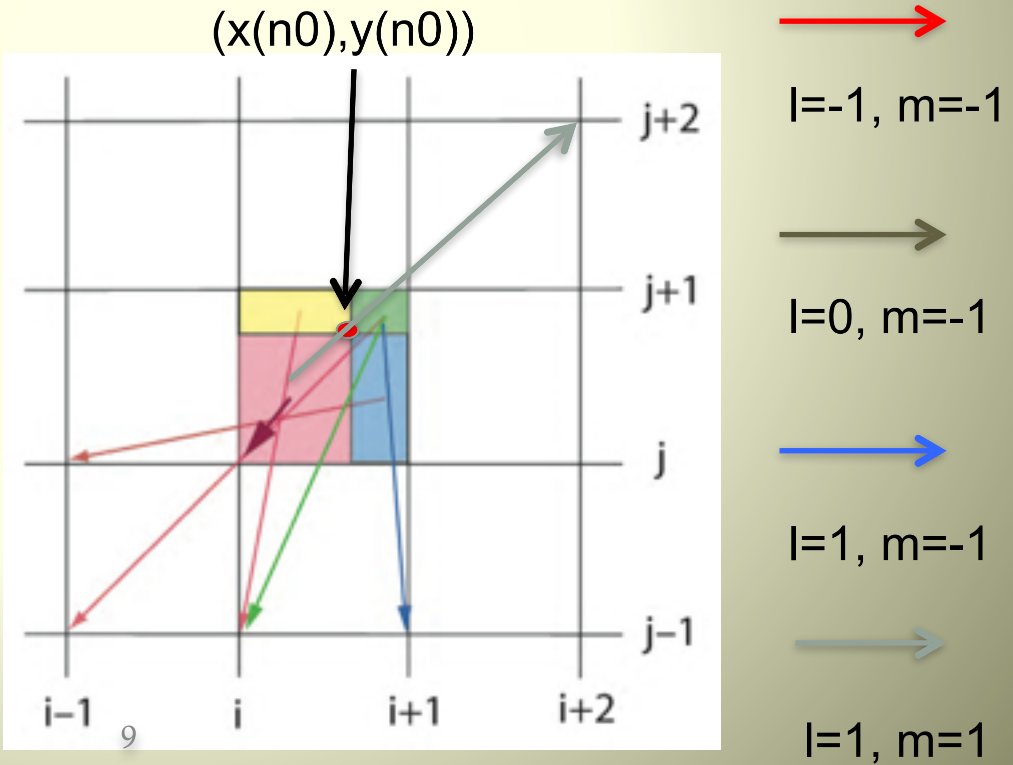
```
      rhe(i+l  ,j+m  ,k+n  )=rhe(i+l  ,j+m  ,k+n  )
+s*cx*cy*cz
      rhe(i+l+1,j+m  ,k+n  )=rhe(i+l+1,j+m  ,k+n  )
+s*dx*cy*cz
      rhe(i+l  ,j+m+1,k+n  )=rhe(i+l  ,j+m+1,k+n  )
+s*cx*dy*cz
      rhe(i+l+1,j+m+1,k+n  )=rhe(i+l+1,j+m+1,k+n  )
+s*dx*dy*cz
```



Density assignment in 3D system (2D)

```
c for electrons
      do 3 n0=1,lecs
      i=x(n0)
      dx=x(n0)-i
      cx=1.-dx
      j=y(n0)
      dy=y(n0)-j
      cy=1.-dy
      k=z(n0)
      dz=z(n0)-k
      cz=1.-dz
C  Smoothing with the (.25,.5,.25)
profile in each dimension:
      sl=.5
      do 121 l=-1,1
      sl=.75-sl
      sr=.5
      do 121 m=-1,1
      sr=.75-sr
      sn=.5
      do 121 n=-1,1
      sn=.75-sn
      s=sl*sr*sn
```

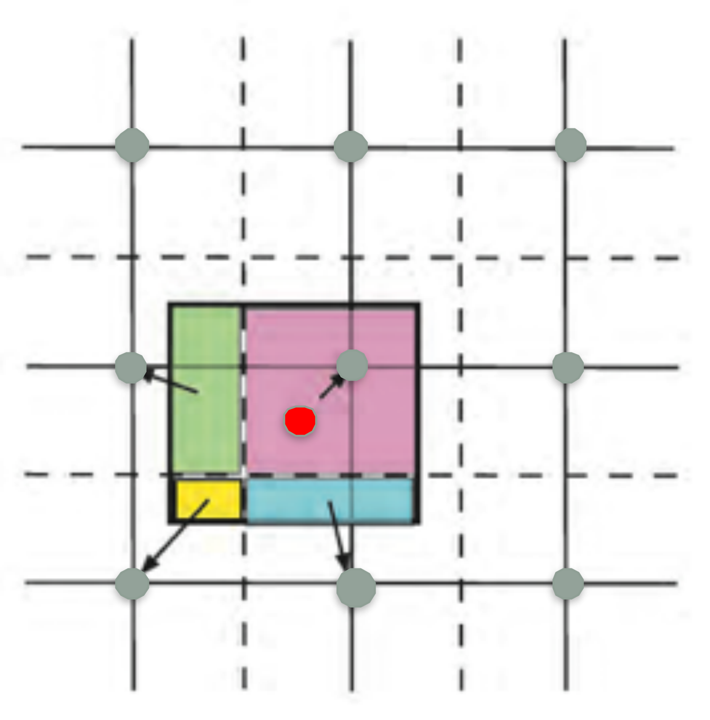
```
      rhe(i+l  ,j+m  ,k+n  )=rhe(i+l  ,j+m  ,k+n  )
+s*cx*cy*cz
      rhe(i+l+1,j+m  ,k+n  )=rhe(i+l+1,j+m  ,k+n  )
+s*dx*cy*cz
      rhe(i+l  ,j+m+1,k+n  )=rhe(i+l  ,j+m+1,k+n  )
+s*cx*dy*cz
      rhe(i+l+1,j+m+1,k+n  )=rhe(i+l+1,j+m+1,k+n  )
+s*dx*dy*cz
```

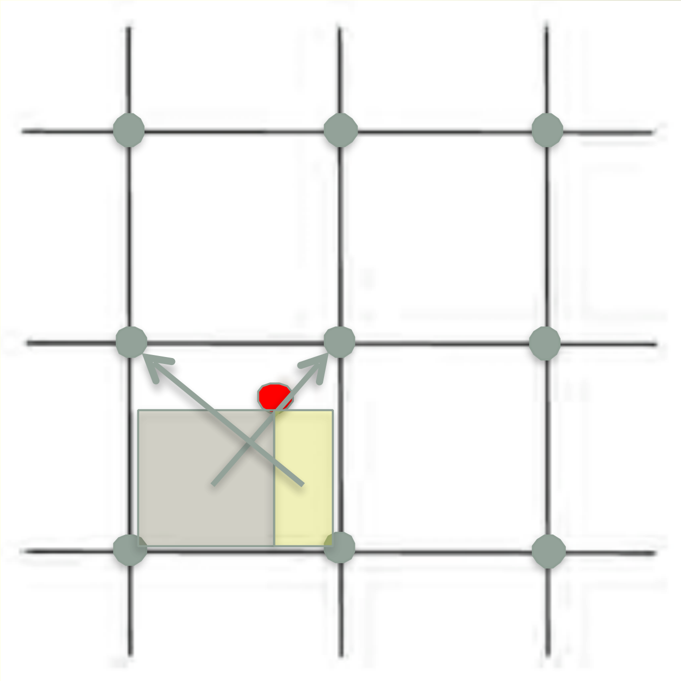


PIC Approach to Vlasov Equation

Lorentz-Force: $$\frac{d\vec{v}_j}{dt} = \frac{q_j}{m_j}(\vec{E} + \frac{\vec{v}_j \times \vec{B}}{c})$$

Solving Maxwell equations on grid





Charge assignment
(conserving charge current)

Force Interpolation

```
C -------------------------------------------------------------------
C
       subroutine Vol_Weighting(mxs,mys,mzs,lecg,rhe,eke,ueg,veg,weg,
     &                          xeg,yeg,zeg,fex,fey,fez)
        implicit none

        integer mxs,mys,mzs,mg,lecg
        real*4 xeg(lecg),yeg(lecg),zeg(lecg)
        real*4 ueg(lecg),veg(lecg),weg(lecg)
        real rhe(mxs,mys,mzs),eke(mxs,mys,mzs)
        real fex(mxs,mys,mzs),fey(mxs,mys,mzs),fez(mxs,mys,mzs)
        real*4 dx,dy,dz,cx,cy,cz
        real*4 sl,sr,sn,s
        integer i,j,k,l,m,n,n0

C  Volume weighting:

c       mhlec = maxhlf +lecs
        do 3 n0=1, lecg
c       if(n0.gt.mhlec.and.n0.lt.lecm) go to 3
```

```
      i=xeg(n0)
      dx=xeg(n0)-i
      cx=1.-dx
      j=yeg(n0)
      dy=yeg(n0)-j
      cy=1.-dy
      k=zeg(n0)
      dz=zeg(n0)-k
      cz=1.-dz
C  Smoothing with the (.25,.5,.25) profile in each dimension:
      sl=.5
      do 187 l=-1,1
      sl=.75-sl
      sr=.5
      do 187 m=-1,1
      sr=.75-sr
      sn=.5
      do 187 n=-1,1
      sn=.75-sn
      s=sl*sr*sn
C Calculating densities
```

```
      rhe(i+l  ,j+m  ,k+n  )=rhe(i+l  ,j+m  ,k+n  )+s*cx*cy*cz
      rhe(i+l+1,j+m  ,k+n  )=rhe(i+l+1,j+m  ,k+n  )+s*dx*cy*cz
      rhe(i+l  ,j+m+1,k+n  )=rhe(i+l  ,j+m+1,k+n  )+s*cx*dy*cz
      rhe(i+l  ,j+m  ,k+n+1)=rhe(i+l  ,j+m  ,k+n+1)+s*cx*cy*dz
      rhe(i+l  ,j+m+1,k+n+1)=rhe(i+l  ,j+m+1,k+n+1)+s*cx*dy*dz
      rhe(i+l+1,j+m  ,k+n+1)=rhe(i+l+1,j+m  ,k+n+1)+s*dx*cy*dz
      rhe(i+l+1,j+m+1,k+n  )=rhe(i+l+1,j+m+1,k+n  )+s*dx*dy*cz
      rhe(i+l+1,j+m+1,k+n+1)=rhe(i+l+1,j+m+1,k+n+1)+s*dx*dy*dz

c Calculating kinitic enegies
c      eke(i+l  ,j+m  ,k+n  )=eke(i+l  ,j+m  ,k+n  )
c     1 +(ueg(n0)*ueg(n0)+veg(n0)*veg(n0)+weg(n0)*weg(n0))*s*cx*cy*cz
c      eke(i+l+1,j+m  ,k+n  )=eke(i+l+1,j+m  ,k+n  )
c     1 +(ueg(n0)*ueg(n0)+veg(n0)*veg(n0)+weg(n0)*weg(n0))*s*dx*cy*cz
c      eke(i+l  ,j+m+1,k+n  )=eke(i+l  ,j+m+1,k+n  )
c     1 +(ueg(n0)*ueg(n0)+veg(n0)*veg(n0)+weg(n0)*weg(n0))*s*cx*dy*cz
c      eke(i+l  ,j+m  ,k+n+1)=eke(i+l  ,j+m  ,k+n+1)
c     1 +(ueg(n0)*ueg(n0)+veg(n0)*veg(n0)+weg(n0)*weg(n0))*s*cx*cy*dz
c      eke(i+l  ,j+m+1,k+n+1)=eke(i+l  ,j+m+1,k+n+1)
c     1 +(ueg(n0)*ueg(n0)+veg(n0)*veg(n0)+weg(n0)*weg(n0))*s*cx*dy*dz
c      eke(i+l+1,j+m  ,k+n+1)=eke(i+l+1,j+m  ,k+n+1)
c     1 +(ueg(n0)*ueg(n0)+veg(n0)*veg(n0)+weg(n0)*weg(n0))*s*dx*cy*dz
```

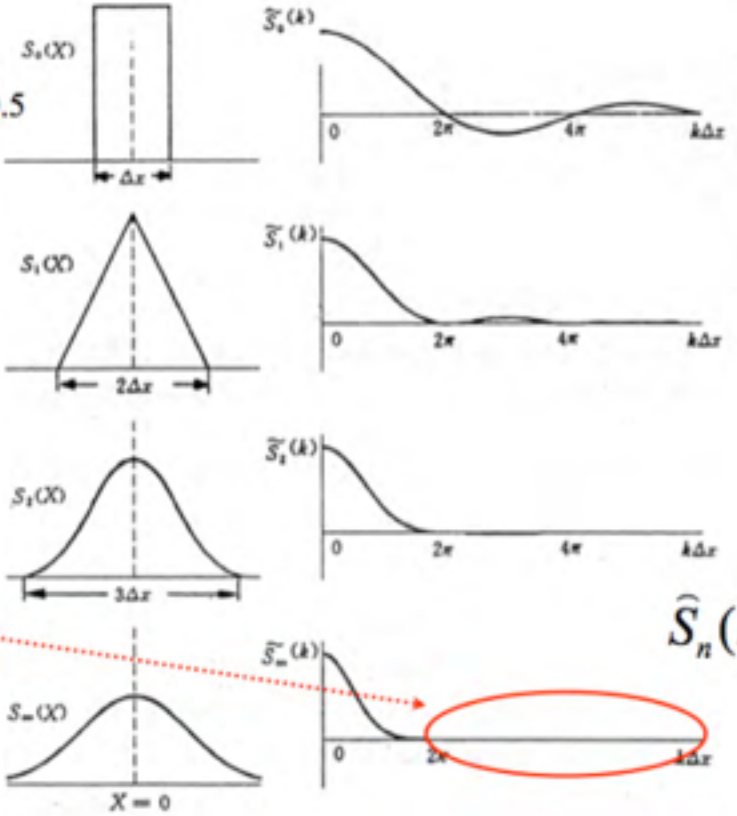
```
c     eke(i+l+1,j+m+1,k+n  )=eke(i+l+1,j+m+1,k+n  )
c    1 +(ueg(n0)*ueg(n0)+veg(n0)*veg(n0)+weg(n0)*weg(n0))*s*dx*dy*cz
c     eke(i+l+1,j+m+1,k+n+1)=eke(i+l+1,j+m+1,k+n+1)
c    1 +(ueg(n0)*ueg(n0)+veg(n0)*veg(n0)+weg(n0)*weg(n0))*s*dx*dy*dz
c
C Calculating flux (not velocity)
      fex(i+l  ,j+m  ,k+n  )=fex(i+l  ,j+m  ,k+n  )
     1        +ueg(n0)*s*cx*cy*cz
      fex(i+l+1,j+m  ,k+n  )=fex(i+l+1,j+m  ,k+n  )
     1        +ueg(n0)*s*dx*cy*cz
      fex(i+l  ,j+m+1,k+n  )=fex(i+l  ,j+m+1,k+n  )
     1        +ueg(n0)*s*cx*dy*cz
      fex(i+l  ,j+m  ,k+n+1)=fex(i+l  ,j+m  ,k+n+1)
     1        +ueg(n0)*s*cx*cy*dz
      fex(i+l  ,j+m+1,k+n+1)=fex(i+l  ,j+m+1,k+n+1)
     1        +ueg(n0)*s*cx*dy*dz
      fex(i+l+1,j+m  ,k+n+1)=fex(i+l+1,j+m  ,k+n+1)
     1        +ueg(n0)*s*dx*cy*dz
      fex(i+l+1,j+m+1,k+n  )=fex(i+l+1,j+m+1,k+n  )
     1        +ueg(n0)*s*dx*dy*cz
      fex(i+l+1,j+m+1,k+n+1)=fex(i+l+1,j+m+1,k+n+1)
     1        +ueg(n0)*s*dx*dy*dz
```

```
c
      fey(i+l  ,j+m  ,k+n  )=fey(i+l  ,j+m  ,k+n  )
     1        +veg(n0)*s*cx*cy*cz
      fey(i+l+1,j+m  ,k+n  )=fey(i+l+1,j+m  ,k+n  )
     1        +veg(n0)*s*dx*cy*cz
      fey(i+l  ,j+m+1,k+n  )=fey(i+l  ,j+m+1,k+n  )
     1        +veg(n0)*s*cx*dy*cz
      fey(i+l  ,j+m  ,k+n+1)=fey(i+l  ,j+m  ,k+n+1)
     1        +veg(n0)*s*cx*cy*dz
      fey(i+l  ,j+m+1,k+n+1)=fey(i+l  ,j+m+1,k+n+1)
     1        +veg(n0)*s*cx*dy*dz
      fey(i+l+1,j+m  ,k+n+1)=fey(i+l+1,j+m  ,k+n+1)
     1        +veg(n0)*s*dx*cy*dz
      fey(i+l+1,j+m+1,k+n  )=fey(i+l+1,j+m+1,k+n  )
     1        +veg(n0)*s*dx*dy*cz
      fey(i+l+1,j+m+1,k+n+1)=fey(i+l+1,j+m+1,k+n+1)
     1        +veg(n0)*s*dx*dy*dz
c
      fez(i+l  ,j+m  ,k+n  )=fez(i+l  ,j+m  ,k+n  )
     1        +weg(n0)*s*cx*cy*cz
      fez(i+l+1,j+m  ,k+n  )=fez(i+l+1,j+m  ,k+n  )
     1        +weg(n0)*s*dx*cy*cz
```

```
      fez(i+l  ,j+m+1,k+n  )=fez(i+l  ,j+m+1,k+n  )
     1       +weg(n0)*s*cx*dy*cz
      fez(i+l  ,j+m  ,k+n+1)=fez(i+l  ,j+m  ,k+n+1)
     1       +weg(n0)*s*cx*cy*dz
      fez(i+l  ,j+m+1,k+n+1)=fez(i+l  ,j+m+1,k+n+1)
     1       +weg(n0)*s*cx*dy*dz
      fez(i+l+1,j+m  ,k+n+1)=fez(i+l+1,j+m  ,k+n+1)
     1       +weg(n0)*s*dx*cy*dz
      fez(i+l+1,j+m+1,k+n  )=fez(i+l+1,j+m+1,k+n  )
     1       +weg(n0)*s*dx*dy*cz
      fez(i+l+1,j+m+1,k+n+1)=fez(i+l+1,j+m+1,k+n+1)
     1       +weg(n0)*s*dx*dy*dz
  187 continue
    3 continue
      return
      end subroutine Vol_Weighting
```

Filtering Action of Shape Functions

$$S_0(x) = \begin{cases} \dfrac{1}{\Delta x}, & -0.5 \le \dfrac{x}{\Delta x} \le 0.5 \\ 0, & \text{otherwise} \end{cases}$$



High-frequency components are filtered by a smooth shape function.

$$\hat{S}_0(k) = \sin\left(\tfrac{k\Delta x}{2}\right) \Big/ \tfrac{k\Delta x}{2}$$

$$\hat{S}_n(k) = [\hat{S}_0(k)]^{n+1}$$

$\downarrow$

$\hat{S}_n(k)$ decays with $k^{-(n+1)}$

Shape Functions of different orders

Corresponding Functions in Fourier Space

Integration of the field equations

$$E_x = -\frac{\partial \phi}{\partial x}$$

$$\frac{\partial E_x}{\partial x} = \rho$$

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

$\downarrow$ finit defference

$$E_j = \frac{\phi_{j-1} - \phi_{j+1}}{2\Delta x}$$

$$\frac{\phi_{j-1} - 2\phi_j + \phi_{j+1}}{(\Delta x)^2} = -\rho_j$$

$$\xrightarrow{\text{Fourier transform}}$$

$$\hat{E}(k_l) = -ik_l\hat{\phi}(k_l)$$

$$\hat{\phi}(k_l) = \frac{\hat{\rho}(k_l)}{k_l^2} \quad k_l = \frac{2\pi l}{L}$$

$$\uparrow \Delta x = 0$$

$$\hat{E}(k_l) = -iK_l\hat{\phi}(k_l)$$

$$\hat{\phi}(k_l) = \frac{\hat{\rho}(k_l)}{K_l^2}$$

$$K_l^2 = k_l^2\left(\frac{\sin(\frac{1}{2}k_l\Delta x)}{\frac{1}{2}k_l\Delta x}\right)^2$$

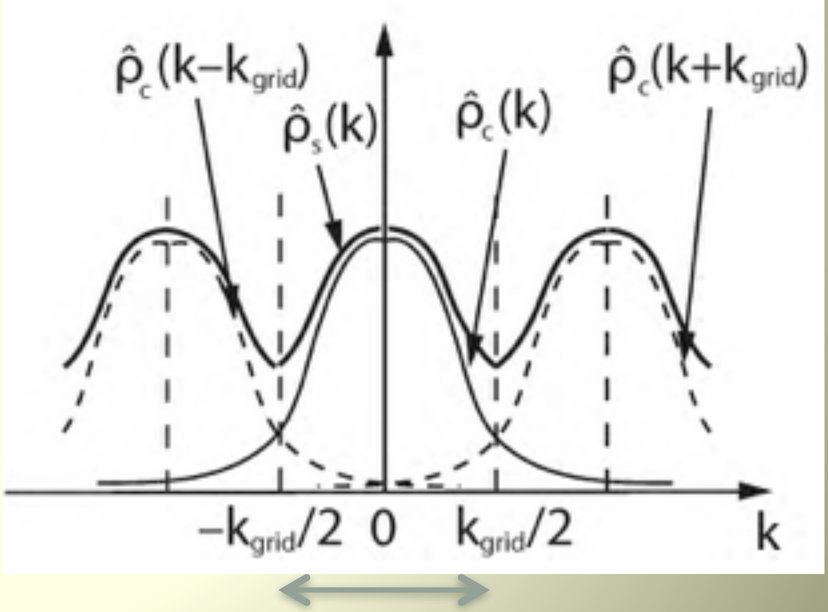
Aliasing

The spurious fluctuation which appears as result of the loss of displacement invariance, manifest themselves in ***k***-space as non-physical mode coupling, known as `aliasing’.

By introducing a mesh we reduced our representation of $\varrho(x)$ from a continuous representation $\varrho_c(x)$ to a sampled representation $\varrho_s(x)$.

$$\hat{\rho}_c(k) = \int_{-\infty}^{\infty} dx\, \rho_c(x) e^{-ikx}$$

$$\hat{\rho}_s(k) = \sum_{n=-\infty}^{n=\infty} \hat{\rho}_c(k + nk_{grid})$$



Principal zone

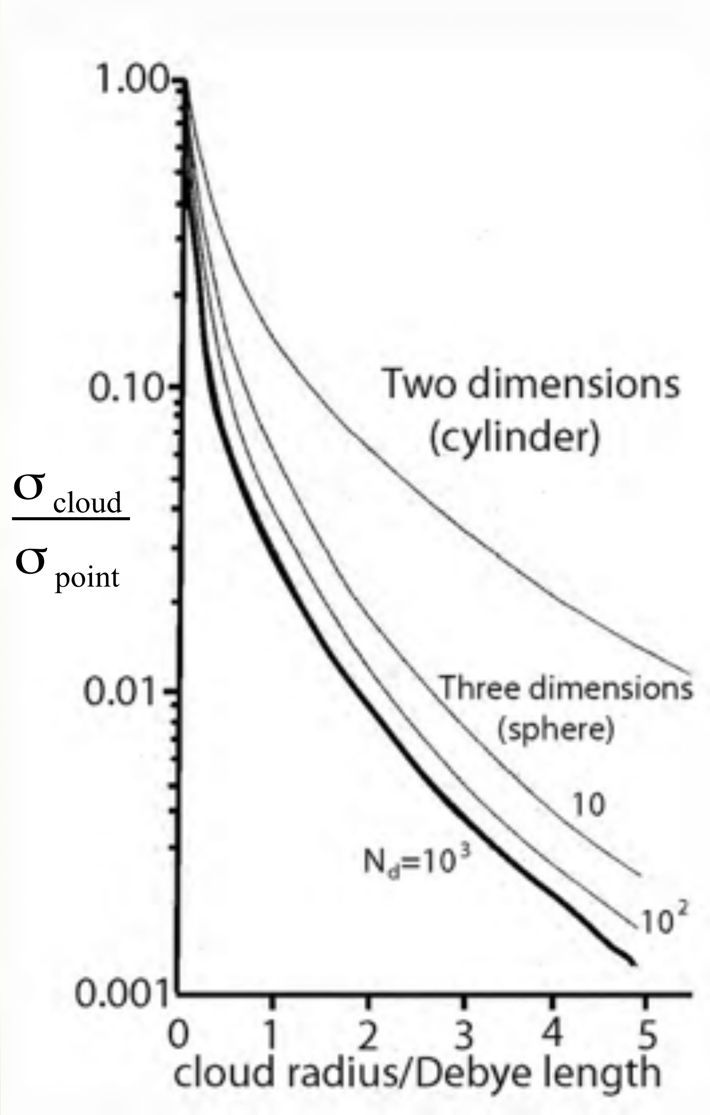
The extra contributions (from |n|>0) to inside the principal zone are called aliasing

Aliasing and reducing noise

- The spurious fluctuations of high frequency cause the noise and error in the main lobe, which might make the numerical system to be unstable.
- The high-k components of $S(k)$ is determined by the smoothness of $S(x)$; The high-k components of $n_c(k)$ is determined by the smoothness of $n(x)$, The number of particles.
- The major noise exists in the particle-in-cell method mainly comes from the aliasing effect. Two methods for reducing the aliasing effects:
  1. Increase the particle number.
  2. Increase the order of the shape function S(x).

Collisional effects

The ratio of the cross sections for finite–sized particles to that for point particles in in two and three dimensions (taken from Okuda and Birdsall 1970)



Examples of collision rates:
(a) two dimensions:

System $100\lambda_D \times 100\lambda_D$
$N = 3\times10^5$ particles
$n\lambda_D^2 = N_p=30$
particle radius $a = \lambda_D$
$\nu=R\omega_{pe}/16N_D\approx 2\times10^{-4}\omega_p$
(b) three dimensions:

System $50\lambda_D \times 50\lambda_D \times 50\lambda_D$
$N = 10^6$ particles
$n\lambda_D^3 = N_p=10$
particle radius $a = \lambda_D$
$\nu=R\omega_{pe}/16N_D\approx 10^{-3}\omega_p$

Finite-size particle effects

$$\frac{\partial f}{\partial t} + \boldsymbol{v}\cdot\frac{\partial f}{\partial \boldsymbol{r}} + \frac{F}{m}\cdot\frac{\partial f}{\partial \boldsymbol{v}} = 0$$

$$F(\boldsymbol{r}_j) = q\int S(\boldsymbol{r} - \boldsymbol{r}_j)\boldsymbol{E}(\boldsymbol{r})d^n r$$

$$\nabla\cdot\boldsymbol{E} = 4\pi q\int f(\boldsymbol{r}',\boldsymbol{v}')S(\boldsymbol{r} - \boldsymbol{r}')d^n r' d^n v'$$

Dispersion function with finite-size particles

$$\varepsilon(\boldsymbol{k},\omega) = 1 + \frac{\omega_p|S(\boldsymbol{k})|^2}{k^2}\int_{-\infty}^{\infty}\frac{\boldsymbol{k}\cdot\partial f_0/\partial \boldsymbol{v}}{(\omega - \boldsymbol{k}\cdot\boldsymbol{v} + i\nu)}d^n v$$

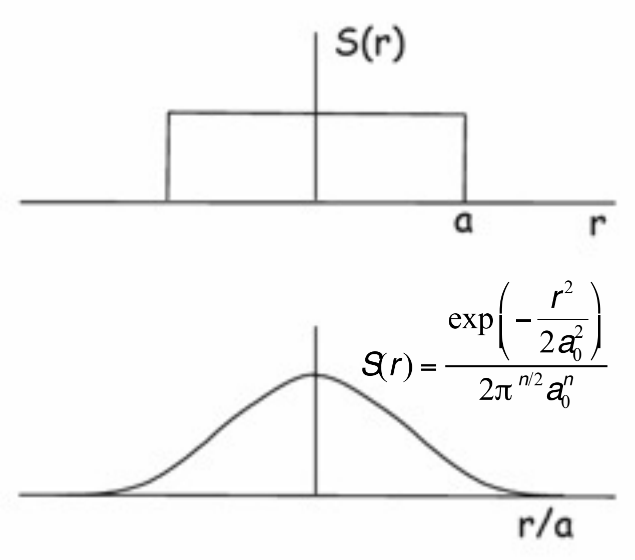
Plasma frequency is modified by smoothing

$$\omega^2(k) = \omega_p^2|S(k)|^2$$

Fourier space modification reduces collisions

$$S(r) = \frac{1}{V_n a^n}, \quad r < a$$

$$S(r) = 0, \quad r > a$$



$$S(r) = \frac{\exp\left(-\frac{r^2}{2a_0^2}\right)}{2\pi^{n/2}a_0^n}$$

$$|S(\boldsymbol{k})|^2 = \frac{e^{-k^2a^2}}{(2\pi)^{n_a n}}$$

Restrictions on time step and grid size

1. Courant condition (Cartesian coordinate) this condition comes from the electromagnetic code (light wave)

$$c\,dt < 1/\sqrt{1/dx_1^2 + 1/dx_2^2 + 1/dx_3^2}$$

2. $\omega_{\max} dt < 0.25 \quad \omega_{\max} = \max(\omega_{pe}, \omega_{ce})$

3. $v_{\max} dt < \min(dx_1, dx_2, dx_3)$

   particle move in one step < 1 cell (grid size)

4. More particles are better, however it takes more memory and computing time

Accuracy and stability of time integration

In vacuum $(\boldsymbol{E}, \boldsymbol{B}) = (\boldsymbol{E}_0, \boldsymbol{B}_0)\exp(i\boldsymbol{k}\bullet\boldsymbol{x} - i\omega t)$ $\boldsymbol{J}=0$ from Maxwell equations

$$\Omega \boldsymbol{B} = cK \times \boldsymbol{E}$$

$$\Omega \boldsymbol{E} = -cK \times \boldsymbol{B}$$

$$\Omega = \omega \left| \frac{\sin(\omega \Delta t/2)}{\omega \Delta t/2} \right|, \; K = k \left| \frac{\sin(k\Delta x/2)}{k\Delta x/2} \right|$$

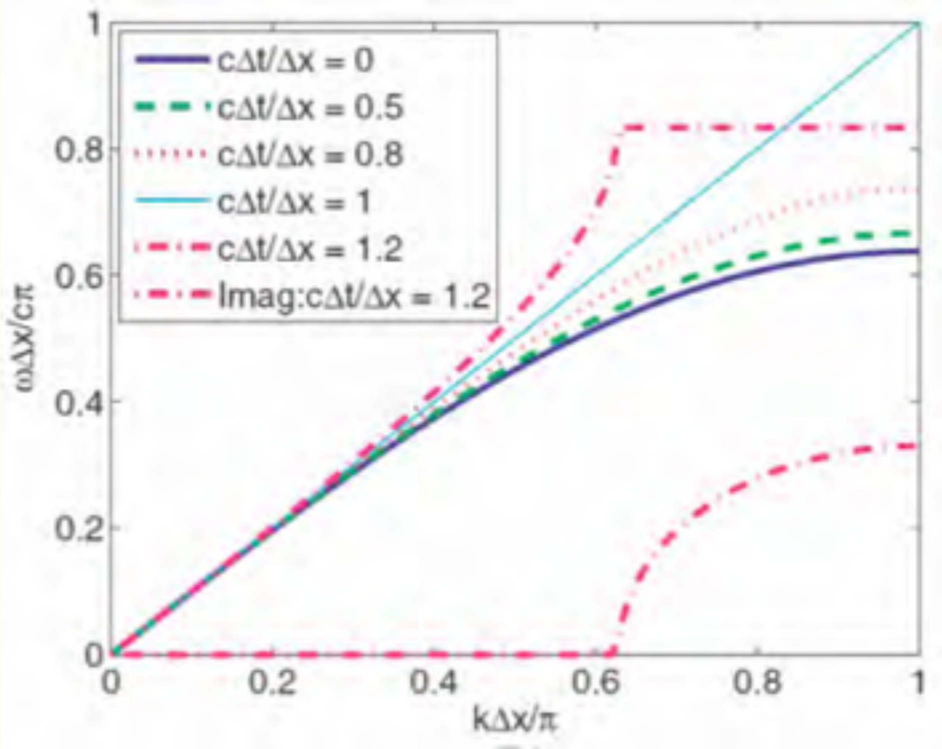
$$\Omega^2 = c^2 K^2$$

$$\left| \frac{\sin(\omega \Delta t/2)}{c\Delta t/2} \right|^2 = \left| \frac{\sin(k\Delta x_1/2)}{\Delta x_1/2} \right|^2 + \left| \frac{\sin(k\Delta x_2/2)}{\Delta x_2/2} \right|^2$$

$$\cos(\omega \Delta t) = \left(c\frac{\Delta t}{\Delta x} \right)^2 [\cos(k\Delta x) - 1] + 1$$

Courant-Levy stability criterion

$$\Delta t \le \frac{1}{c} \left(\sum_i \frac{1}{(\Delta x_i)^2} \right)^{-1/2}$$



Vacuum dispersion curve for leapfrog difference for wave equation

Relativistic particles which move faster than numerical speed of light cause numerical Cherenkov radiation in high wave-numbers

Calculation of vacuum dispersion solution (homework)

$$\cos(\omega\Delta t) = \left(c\frac{\Delta t}{\Delta x}\right)^2 [\cos(k\Delta x) - 1] + 1$$

$$\cos\left(\frac{\omega\Delta x}{c}\frac{cdt}{dx}\right) - 1 = \left(c\frac{\Delta t}{\Delta x}\right)^2 [\cos(k\Delta x) - 1]$$

$$\cos(\alpha y) - 1 = (\alpha)^2 [\cos(x) - 1]$$

$$y = \frac{\omega\Delta x}{c}, \qquad x = k\Delta x, \quad \alpha = \frac{cdt}{dx}$$

$$\frac{\cos(\alpha y) - 1}{(\alpha)^2} = \cos(x) - 1, \quad \cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n (x)^{2n}}{(2n)!}$$

$$-0.5y^2 + \cancel{\frac{1}{24} y^4 \alpha^2} + \cdot = \cos(x) - 1$$

$(\alpha = 0)$

$$y = \sqrt{2 \cdot (1 - \cos(x))}$$

Current deposit scheme (2-D)

$$\nabla \cdot E = 4\pi\rho, \quad \nabla \cdot \frac{\partial E}{\partial t} = 4\pi \frac{\partial \rho}{\partial t}, \quad \nabla \cdot (c\nabla \times B - 4\pi J) = 4\pi \frac{\partial \rho}{\partial t},$$

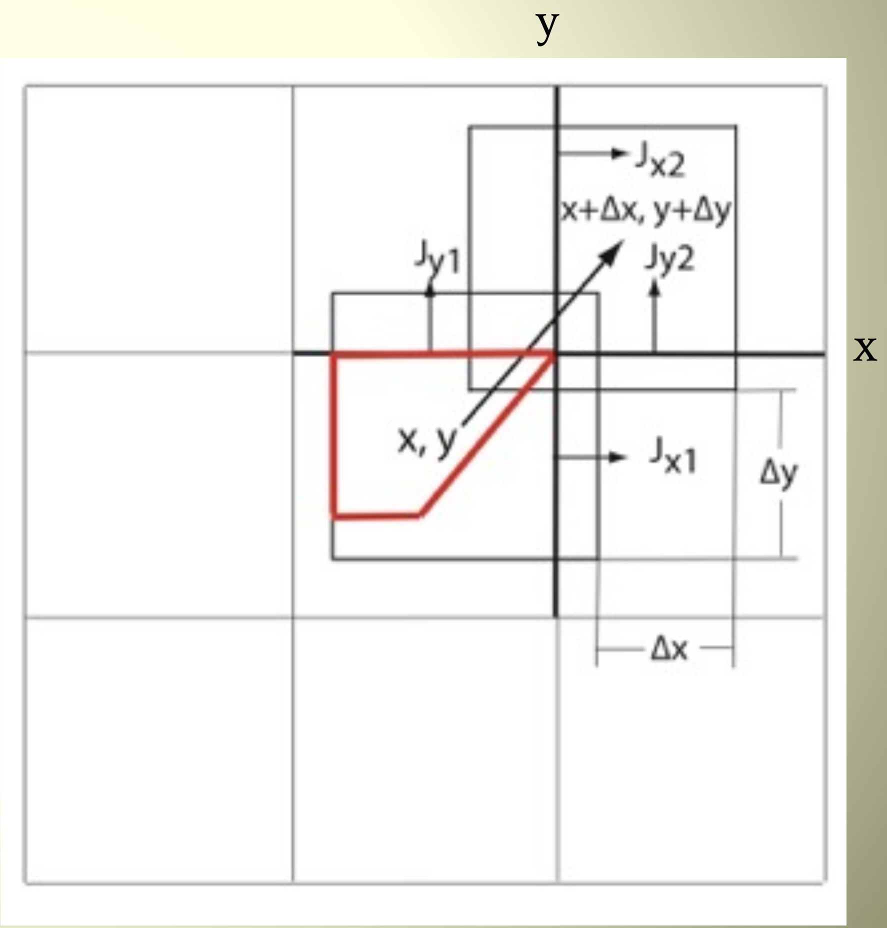
$$\frac{\partial \rho}{\partial t} = -\nabla \cdot J$$

$$J_{x1} = q\Delta x\left(\frac{1}{2} - y - \frac{1}{2}\Delta y\right)$$

$$J_{x2} = q\Delta x\left(\frac{1}{2} + y + \frac{1}{2}\Delta y\right)$$

→ $$J_{y1} = q\Delta y\left(\frac{1}{2} - x - \frac{1}{2}\Delta x\right)$$

$$J_{y2} = q\Delta y\left(\frac{1}{2} + x + \frac{1}{2}\Delta x\right)$$



Current deposit scheme (2-D)

$$\nabla\cdot E = 4\pi\rho,\quad \nabla\cdot\frac{\partial E}{\partial t} = 4\pi\frac{\partial\rho}{\partial t},\quad \nabla\cdot(c\nabla\times B - 4\pi J) = 4\pi\frac{\partial\rho}{\partial t},$$

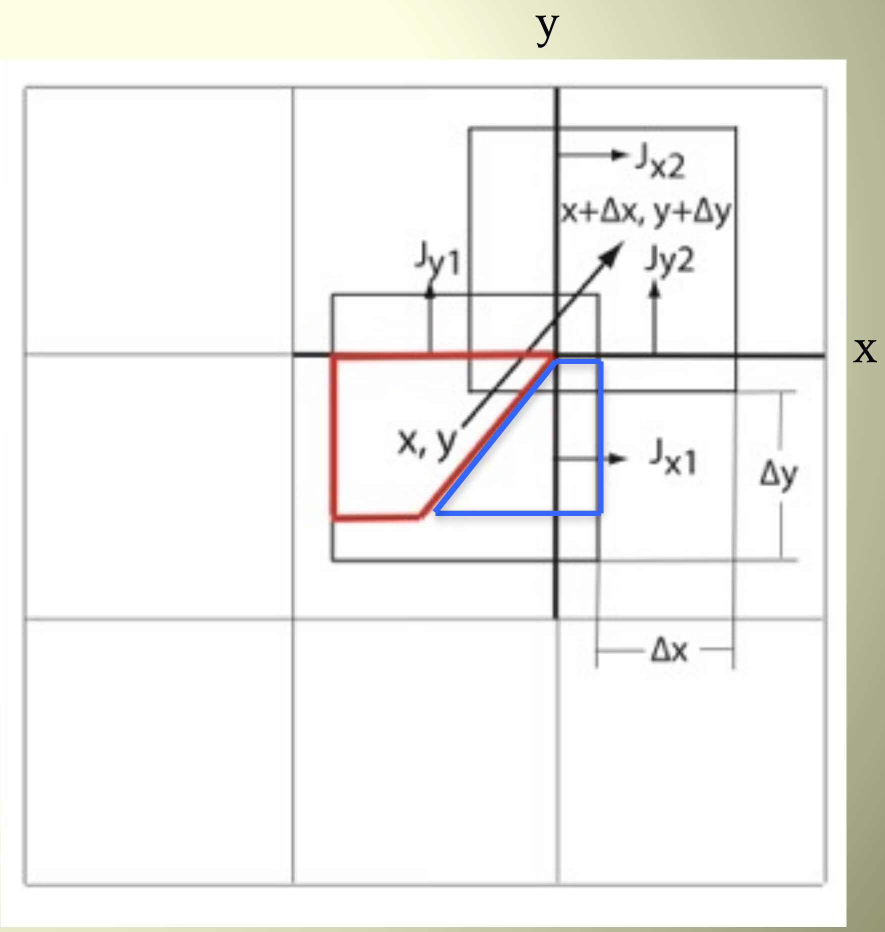
$$\frac{\partial\rho}{\partial t} = -\nabla\cdot J$$

$$J_{x1} = q\Delta x\left(\frac{1}{2} - y - \frac{1}{2}\Delta y\right)$$

$$J_{x2} = q\Delta x\left(\frac{1}{2} + y + \frac{1}{2}\Delta y\right)$$

→ $$J_{y1} = q\Delta y\left(\frac{1}{2} - x - \frac{1}{2}\Delta x\right)$$

→ $$J_{y2} = q\Delta y\left(\frac{1}{2} + x + \frac{1}{2}\Delta x\right)$$



Current deposition seven-boundary move

$\Delta x_1 = 0.5 - x,$

$\Delta y_1 = (\Delta y/\Delta x)\Delta x_1,$

$x_1 = -0.5,$

$y_1 = y + \Delta y_1,$

$\Delta x_2 = \Delta x - \Delta x_1,$

$\Delta y_2 = \Delta y - \Delta y_1$

$$J_{x1} = q\Delta x_1\left(\tfrac{1}{2} - y - \tfrac{1}{2}\Delta y_1\right)$$

$$J_{x2} = q\Delta x_1\left(\tfrac{1}{2} + y + \tfrac{1}{2}\Delta y_1\right)$$

$$J_{y1} = q\Delta y_1\left(\tfrac{1}{2} - x - \tfrac{1}{2}\Delta x_1\right)$$

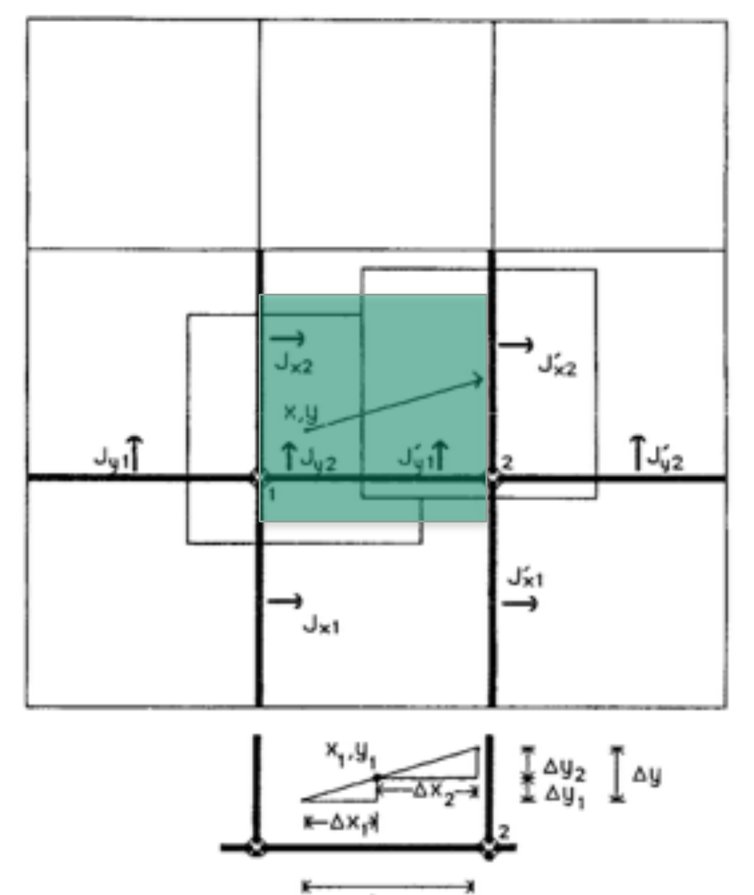
$$J_{y2} = q\Delta y_1\left(\tfrac{1}{2} + x + \tfrac{1}{2}\Delta x_1\right)$$

$$J'_{x1} = q\Delta x_2\left(\tfrac{1}{2} - y_1 - \tfrac{1}{2}\Delta y_2\right)$$

$$J'_{x2} = q\Delta x_2\left(\tfrac{1}{2} + y_1 + \tfrac{1}{2}\Delta y_2\right)$$

$$J'_{y1} = q\Delta y_2\left(\tfrac{1}{2} - x_1 - \tfrac{1}{2}\Delta x_2\right)$$

$$J'_{y2} = q\Delta y_2\left(\tfrac{1}{2} + x_1 + \tfrac{1}{2}\Delta x_2\right)$$



Current deposition ten-boundary move

$\Delta x_1 = 0.5 - x,$

$\Delta y_1 = (\Delta y / \Delta x)\Delta x_1,$

$x_1 = -0.5,$

$y_1 = y + \Delta y_{1,}$

$\Delta y_2 = 0.5 - y - \Delta y_1,$

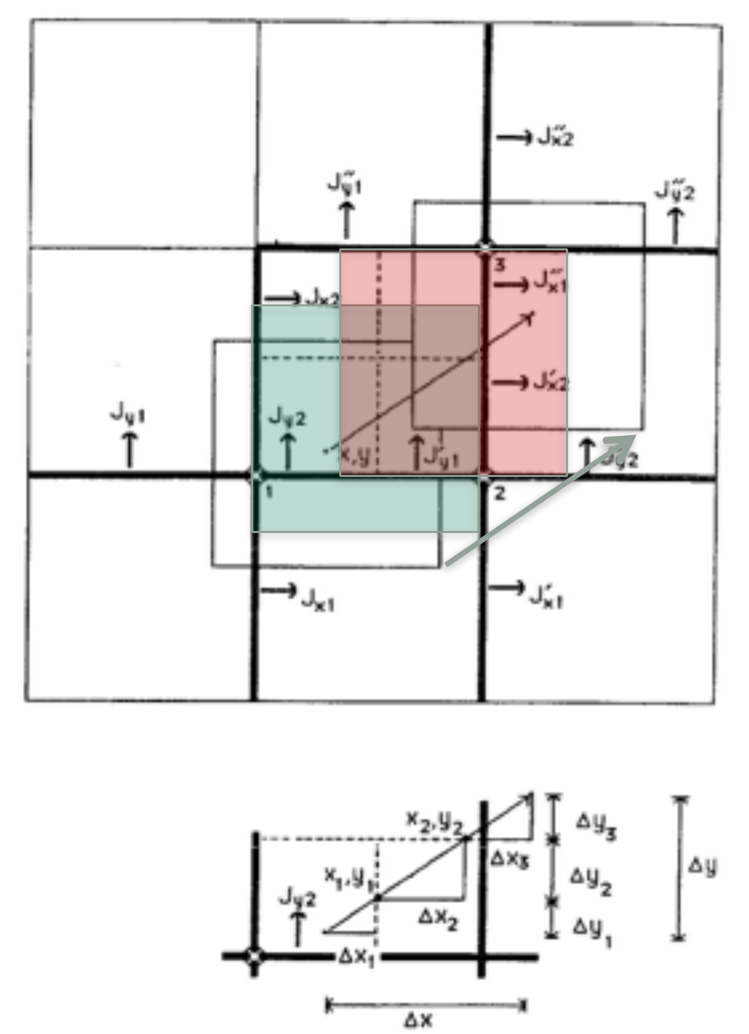
$\Delta x_2 = (\Delta x / \Delta y)\Delta y_2,$

$x_2 = \Delta x_2 - 0.5,$

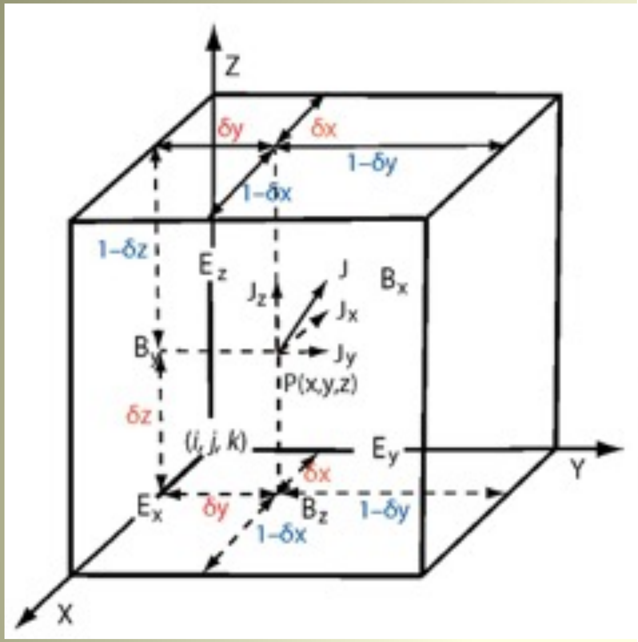
$y_2 = 0.5,$

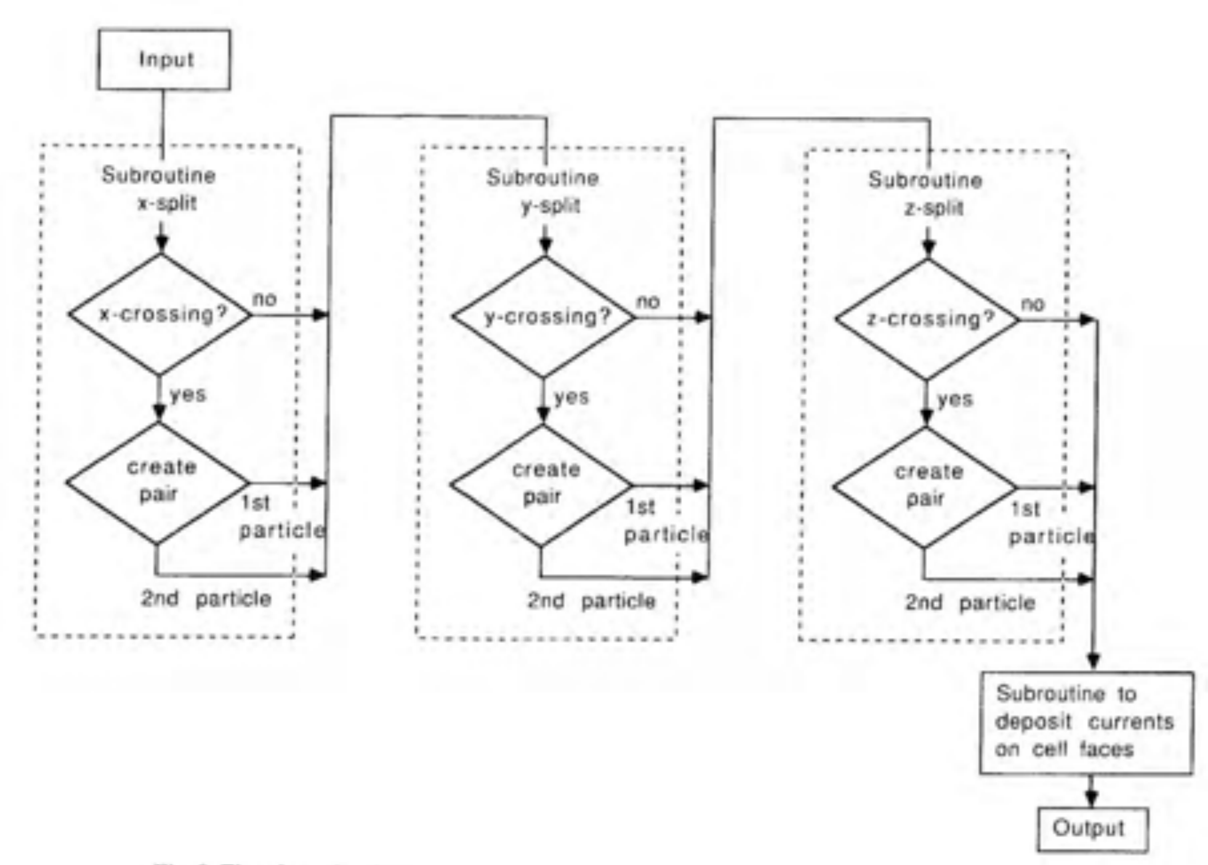
$\Delta x_3 = \Delta x - \Delta x_1 - \Delta x_2,$

$\Delta y_3 = \Delta y - \Delta y_1 - \Delta y_2$



Charge and current deposition

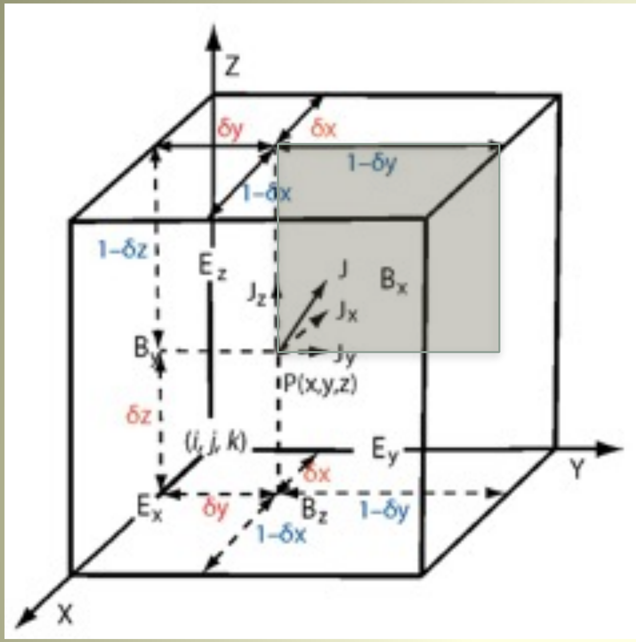


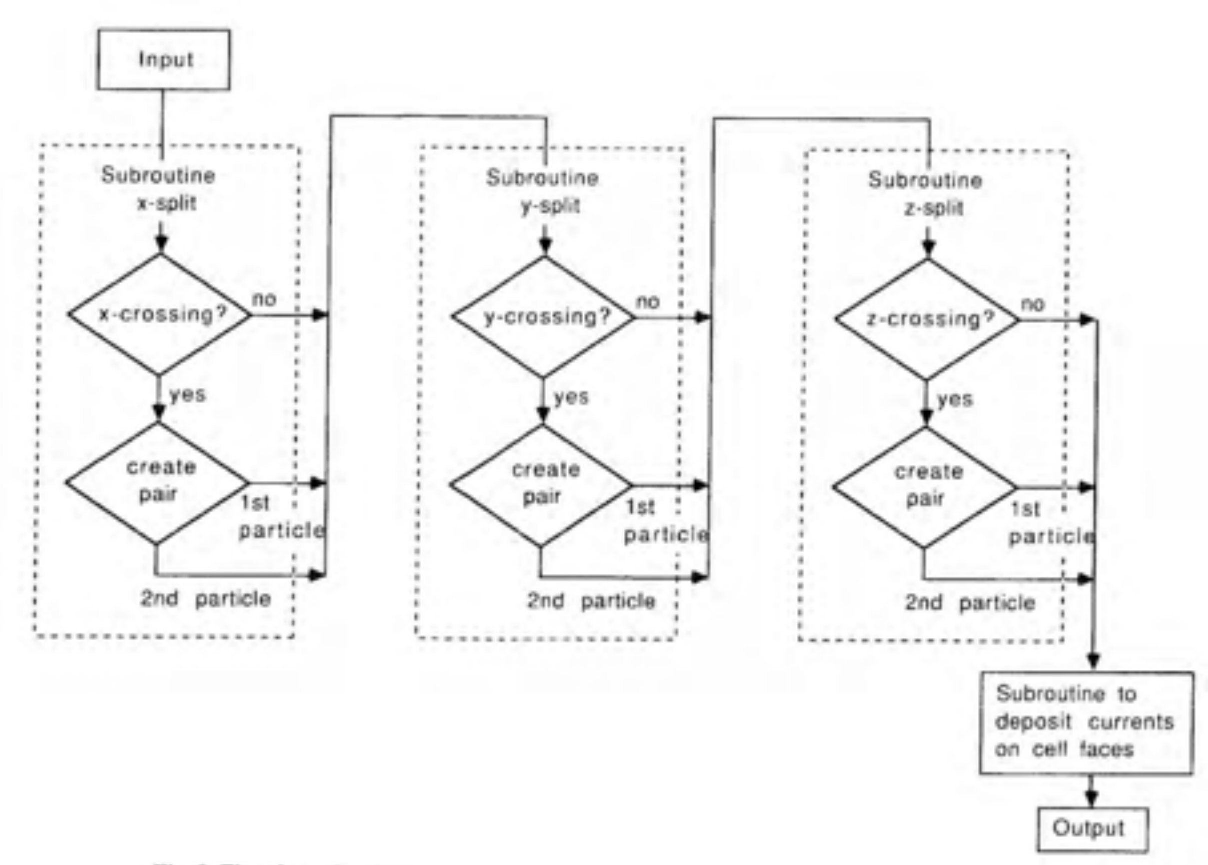


Current deposition can take as much time as the mover. More optimized deposits exist (Umeda 2003).

Charge conservation makes the whole Maxwell solver local and hyperbolic. Static fields can be established dynamically.

Charge and current deposition





Current deposition can take as much time as the mover. More optimized deposits exist (Umeda 2003).

Charge conservation makes the whole Maxwell solver local and hyperbolic.
Static fields can be established dynamically.