What is Simulation?

• It is an art of taking approximations
• For example, to obtain the square root of 2, i.e.,
• We can take Newton’s Iteration Method

\[ x_{i+1} = \frac{x_i f'(x_i) - f(x_i)}{f'(x_i)} \]
for \( i=0 \ldots n \)

\[ f(x) = x^2 - 2 \]
\[ f'(x) = 2x \]

\[ x_0 = 1.3 \]
\[ x_1 = 1.4192 \]
\[ x_2 = 1.4140 \]
\[ x_3 = 1.4142 \]
Newton’s iteration method

\[ f'(x_0) \approx \frac{f(x_1) - f(x_0)}{x_1 - x_0} \]

According to Taylor’s series expansion

Assuming \( x_1 \) is a better solution to \( f(x) = 0 \), let \( f(x_1) = 0 \)

\[ x_1 = \frac{x_0 f'(x_0) - f(x_0)}{f'(x_0)} \implies \]

\[ x_2 = \frac{x_1 f'(x_1) - f(x_1)}{f'(x_1)} \implies \]

\[ x_3 = \frac{x_2 f'(x_2) - f(x_2)}{f'(x_2)} \implies \]

\[ x_n = \frac{x_{n-1} f'(x_{n-1}) - f(x_{n-1})}{f'(x_{n-1})} \]
What is Simulation?

• The better your first approximation (1.3) is, the more accurate your simulation will be.

• The more loops (computational time) you are using, the more accurate your simulation is.

• There is no such thing as a correct or wrong simulation because we are taking approximations from the beginning.

• There is good or bad simulation in terms of how skillfully you approximate reality.
What is Finite Difference?

Replacing the first, second or higher order of differentiatives by simple difference equations.

Taylor's Series Expansion states that any function value $f(x+h)$ at position $x+h$ can be expanded as a series of first, second and higher orders of differentiatives at position $x$. 

$$f(x-h)$$

$$f(x)$$

$$f(x+h)$$
What is Finite Difference?

1). \[ f(x + h) = f(x) + h f'(x) + \frac{h^2}{2!} f''(x) + \text{higher order terms} \]

2). \[ f(x - h) = f(x) - h f'(x) + \frac{(-h)^2}{2!} f''(x) + \text{higher order terms} \]
What is Finite Difference?

1). \[ f(x + h) = f(x) + hf'(x) + \frac{h^2}{2!} f''(x) + \text{ higher order terms} \]

Taking the first order and ignore the other terms, we have

\[ f(x + h) = f(x) + hf'(x) + O(h) \]

O(h) means the accumulate errors up to first order

After rearrangement, we have a finite difference of first order

\[ f'(x) = \frac{f(x + h) - f(x)}{h} \]

with an error of O(h)
with an error of $O(h)$

\[ f'(x) = \frac{f(x + h) - f(x)}{h} \]

with an error of $O(h)$

\[ f''(x) = \frac{f(x) - f(x - h)}{h} \]

with an error of $O(h^2)$

\[ f''(x) = \frac{f(x + h) - f(x - h)}{2h} \]

with an error of $O(h^2)$

\[ f'''(x) = \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} \]

with an error of $O(h^2)$
One Dimensional Plasma Simulation

Poisson’s equation: \[
\frac{\partial^2 \phi}{\partial x^2} = -\frac{n_I - n_e}{\varepsilon_o}
\]

Continuity equation: \[
\frac{\partial n_I}{\partial t} + \frac{\partial (n_I v_I)}{\partial x} = 0 \Rightarrow \frac{\partial n_I}{\partial t} + n_I \frac{\partial v_I}{\partial x} + v_I \frac{\partial n_I}{\partial x} = 0
\]

Momentum Conversion: \[
\frac{\partial v_I}{\partial t} + v_I \frac{\partial v_I}{\partial x} = \frac{q}{m_I} E
\]

Boltzmann’s electron equation: \[
n_e = n_o \exp\left(\frac{q\phi}{kT_e}\right)
\]
Poisson’s equation:
\[ \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} = -\frac{n_i - n_o \exp(q\phi_i / kT_e)}{\varepsilon_o} \]

Continuity equation:
\[ \frac{n_i^{j+1} - n_i^j}{f} + n_i^j \frac{v_i^j - v_{i-1}^j}{h} + v_i^j \frac{n_i^j - n_{i-1}^j}{h} = 0 \]

Momentum Conversion:
\[ \frac{v_i^{j+1} - v_i^{j-1}}{f} + v_i^j \frac{v_i^j - v_{i-1}^j}{h} = \frac{q}{m_i} E_i^j \]
At time $j$, solve the Poisson equation based on the ion density $n_i$ obtaining the electric field $E_i$ at each node.

Advance the ion density $n_i$ and ion velocity $v_i$ of each node to time step $j+1$ from the continuity and momentum equations.

Repeat the procedure until the end of the simulation.
Particle Simulation

• The state of the physical system (Plasma) is defined by the attributes of a finite ensemble of particles (ions and electrons).
• The evolution of the system is determined by the laws of interactions between the particles.

Plasma density = $1 \times 10^{10}$ cm$^{-3}$

In 1 cm$^3$ volume, we have $1 \times 10^{10}$ ions.

A double variable occupies 8 bytes of memory.
$1 \times 10^{10}$ ions need 80 GB of memory.

In 1 m$^3$ volume, we have $1 \times 10^{16}$ ions that
Need 80 PB of memory (peta = $10^{15}$).
Particles Simulation

• The state of the physical system (Plasma) is defined by the attributes of a finite ensemble of particles (ions and electrons).

In 1 cm$^3$ we have 1000 Super particles. Each Super particle represents 1x10$^7$ ions.

In 1 m$^3$ we have 1x10$^9$ Super particles. Need 8 GB of memory.
Particle Models

The particle-particle (PP) model:
Using the action at a distance formulation of the force law.

The particle-mesh (PM) model:
Treating the force as a field quantity and approximating it on a mesh.

The particle-particle--particle-mesh (PPPM or P³M) model:
A hybrid of the PP and PM models.
Particle-Particle (PP) Model

At some time $t$, we will calculate the forces of interaction for each particle.

For example, Coulomb’s force:

$$F_{ij} = \frac{q_i q_j (x_i - x_j)}{4\pi \varepsilon_o |x_i - x_j|^3}$$

The computational cost for 60 particles is $60 \times 59 = 3540$ operations.

The cost for a million particles is $(10^{12} - 10^6)$ operations per time step.
Particle-Mesh (PM) Model

We draw a mesh and distribute particles within the cells.

1. Assign charge (particles) to mesh.
2. Solve Poisson’s equation on the mesh.
   \[ \nabla^2 \phi = -\left(\frac{n_l - n_e}{\varepsilon_o}\right) \]
3. Compute forces from the mesh-defined potential and interpolate forces at the particle’s position.

For \( N_p \) particles and \( N \) nodes,
Operation count = \( \alpha N_p + \beta N \),
Where \( \alpha \) and \( \beta \) depend on the PM scheme being used.
Particle-Particle—Particle-mesh (PPPM) Model

The trick is to split the interparticle forces into two parts:

\[ F = F_{ij}^{sr} + F^m \]

\( F_{ij}^{sr} \) is the short range force and is non-zero for only a short distance.

\( F^m \) is the slow varying force represented on the mesh.

Operations count = \( \alpha N_p + \beta N + \gamma N_n N_p \)
where \( N_n \) is the number of neighbors.
Particle-In-Cell Weighting

In Particle-Mesh Model, we have to assign charge (particle) to the mesh and interpolate forces from the mesh back to the particle.

Particle-In-Cell refers to one of the weighting methods of assigning and interpolating.

Zero-order weighting: Nearest-Grid-Point or NGP.

First-order weighting: Particle-In-Cell or PIC.

Higher-order weighting: use of quadratic and cubic splines rounds off further the particle roughness in particle shape.
Nearest-Grid-Point (NGP)

Any particle of density $n_i$ within distance $\pm \Delta x/2$ from the mesh node will be assigned to the node.

Density of the mesh node

$$\Sigma n_i/(\Delta x)^2.$$  

NGP is rather noisy and a better weighting is needed.
Particle-In-Cell (PIC)

Any particle of density $n_i$ within distance $\pm \Delta x$ from the mesh node will be assigned to the node.

A proportion of the density belongs to the mesh node.

**Bilinear Interpolation**

$$n_{node} = n_i \frac{a}{a + b + c + d} = n_i \frac{a}{(\Delta x)^2}$$

Density of the mesh node

$$= \Sigma n_{node}/(\Delta x)^2.$$
Higher-Order Weighting

Particle Shape | One-Dimension | Two-Dimensions
---|---|---
Zero Order (NGP) | Square | Box
First Order (PIC) | Triangle | Pyramid
Higher Order | Quadratic and cubic splines | Something more symmetrical
Poisson’s equation: 

\[
\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} = -\frac{n_i - n_o \exp(q\phi_i/kT_e)}{\varepsilon_o}
\]

Particle-Mesh Method:

1. Assign charge (particles) to mesh
2. Solve Poisson’s equation on the mesh
3. Compute forces from the mesh-defined potential and interpolate forces at particle positions.
4. Update particle’s positions and repeat steps 1 to 4.
Demonstration of PIII Trench

---

By Dixon Kwok