Orbit integration in fully implicit, electrostatic particle-in-cell algorithms

Guangye Chen, Luis Chacón
Los Alamos National Laboratory, Los Alamos, NM 87545, USA
Daniel C. Barnes
Coronado Consulting, Lamy, NM 87540, USA

JIFT 2012 workshop on innovative methods in plasma particle simulations
Nov. 2-3, 2012 • Providence, RI
Work supported by ORNL LDRD, DOE FES, and LANL LDRD Programs
OUTLINE

• Particle-in-cell methods.
• Implicit and nonlinearly consistent, electrostatic PIC.
• Charge- and energy-conserving particle mover in a electric field
  – Finite difference method
  – Analytical method
• A GPU implementation of the particle mover
  – Roofline model;
  – Optimizations;
  – Performance.
• A CPU-GPU simulation of a multi-scale problem.
• Summary.
PIC method for kinetic plasma simulations

- PIC method solves *Vlasov–Maxwell* (or *Vlasov-Poisson*) equations using a particle-mesh formulation.

\[ \frac{\partial f}{\partial t} + v \cdot \nabla f + \frac{F}{m} \cdot \nabla_v f = 0 \]

- For simplicity, we focus on a 1D-1V *Vlasov-Poisson* (VP) system in this work.

\[ \begin{align*}
\mathbf{\dot{x}} &= \mathbf{v} \\
\mathbf{\dot{v}} &= \frac{\mathbf{F}}{m} \\
\mathbf{F} &= q(\mathbf{E} + \mathbf{v} \times \mathbf{B}) \\
\n\n\n\end{align*} \]

Maxwell equations solve for E, B field

\[ \begin{align*}
\n\n\n\n\n\end{align*} \]
Development of PIC methods

◆ Classical PIC methods (1950s) are explicit in time integration, which suffer numerical instabilities (CFL, finite-grid-instability).

◆ Early implicit PIC methods generally take semi-implicit, linearized approaches:
  – Moment method [Mason, Denavit, 1981][Brackbill and Forslund, 1982],

Long-time accuracy is hard to achieve using large time steps without energy conservation [Cohen, Langdon, and Hewett, 1989]
Our approach:

- Employs timesteps much larger than the explicit.
- Employs grid-sizes much larger than the Debye length.

Multiple time-scales:

- Dynamic time scale of field $>>$ particle orbit time scale ($\Delta t >> \Delta \tau$)
- Particle orbit time scale varies with species (e.g., $T_e >> T_i$)
- Particle orbit time scale varies within one species (passing particles vs. trapped particles)

Our goal is to develop an efficient and accurate, implicit, and nonlinearly consistent PIC algorithm (1D electrostatic as proof of principle, extensible to multiple-D, curvilinear geometry, and electromagnetic).
1D(1V) Vlasov-Ampere equations w periodic B.C.

\begin{align*}
\dot{x}_p - v_p &= 0, \\
\dot{v}_p - q_p \times \text{SM}(E)_p / m &= 0, \\
(p = 1, \ldots, N_p) \\
\end{align*}

\begin{align*}
\varepsilon_0 \dot{E}_i - \text{SM}(j_i) - \langle j \rangle &= 0 \\
(i = 1, \ldots, N_g) \\
\end{align*}

where \( \text{SM}(Q_i) = \frac{2Q_{i-1} + 4Q_i + 2Q_{i+1}}{4} \)

By Crank-Nicolson scheme:

\begin{align*}
\frac{x_p^{n+1} - x_p^n}{\Delta \tau^n} &= v_p^{n+1/2}, \\
\frac{v_p^{n+1} - v_p^n}{\Delta \tau^n} &= \frac{q_p}{m_p} \text{SM}(E) \left( x_p^{v+1/2} \right).
\end{align*}
Solving implicit time integration

• Newton-Raphson’s method for \( G(x) = 0 \):

\[
J \delta x_k = -G(x_k), \text{ where } J = \frac{\partial G}{\partial x}.
\]

• Jacobian-free approximation for Krylov subspace methods

\[
J(x)v = \frac{[G(x + \varepsilon v) - G(x)]}{\varepsilon} \quad \text{where } \varepsilon \text{ is a small parameter}.
\]

– No need to compute and store the Jacobian;
– Only need to form the nonlinear residual vector of the problem.

• Newton-Raphson’s method to achieve nonlinear convergence, and GMRES to solve the linear system, \( J \delta x = -G(x) \), in each Newton iteration.
Key algorithmic strategy

◆ The full residual $G(E,\{x_p,v_p\})$ is impractical to solve by JFNK.
  ➢ Too large storage requirement of nonlinear solver.
◆ **Nonlinear elimination** is necessary to make implicit PIC practical.

\[
\begin{align*}
G(E,\{x_p,v_p\}) \\
v_p &= v_p(E), \quad x_p = x_p(E), \\
\Downarrow \\
\tilde{G}(E) &= 0
\end{align*}
\]

◆ The required memory is comparable to that of a fluid approach.

G. Chen (gchen@lanl.gov) JIFT 2012
Self-adaptive particle mover

In a multi-scale problem, the timescale of the field can be much smaller than that of particles.

\[ \Delta t = \sum \Delta \tau^\nu \]

Resolving orbit is essential!

\[ \frac{x_p^{\nu+1} - x_p^\nu}{\Delta \tau^\nu} = \frac{v_p^{\nu+1} + v_p^\nu}{2}, \]

\[ \frac{v_p^{\nu+1} - v_p^\nu}{\Delta \tau^\nu} = \frac{q_p}{m_p} \sum_i \frac{E_i^n + E_i^{n+1}}{2} S\left(x_i - \frac{x_p^\nu + x_p^{\nu+1}}{2}\right). \]

Local-error-controlled; time-centered, 2nd order; unconditionally stable; charge-\& energy-conserving; equate VA \& VP algorithms.
Analytical solution of the equations of motion in a piecewise linear electric field

\[
\frac{dx_p}{dt} = v_p, \quad \frac{dv_p}{dt} = a_p, \quad \frac{d^2x_p}{dt^2} - \frac{da}{dx} x_p = 0.
\]

\[
x_p(t) = x_p^0 + v_p^0 t + \frac{1}{2} a_p^0 t^2,
\]

\[
\frac{da}{dx} = 0,
\]

\[
x_p(t) = x_p^0 \cos(\omega t) + v_p^0 \sin(\omega t),
\]

\[
\frac{da}{dx} < 0,
\]

\[
x_p(t) = x_p^0 \cosh(bt) + v_p^0 \sinh(bt),
\]

\[
\frac{da}{dx} > 0,
\]

\[
\omega = \sqrt{-\frac{da}{dx}} \quad b = \sqrt{\frac{da}{dx}}.
\]
Energy conservation

\[ \epsilon_0 \frac{E_{i+1/2}^{n+1} - E_{i+1/2}^n}{\Delta t} - j_{i+1/2}^{n+1/2} + \langle j_{i+1/2}^{n+1/2} \rangle = 0, \]

\[ j_{i+1/2}^{n+1/2} = \frac{1}{\Delta x \Delta t} \sum_p \int_0^{\Delta t} q_p S_1(x_{i+1/2} - x_p) v_p d\tau \]

\[ = \frac{1}{\Delta x \Delta t} \sum_p \int_{x_p}^{x_{i+1/2}} q_p S_1(x_{i+1/2} - x_p) dx_p, \]

\[ K_p^{n+1} - K_p^n = \sum_p \int_0^{\Delta t} \frac{d(m_p v_p^2/2)}{dt} dt = \sum_p \int_0^{\Delta t} q_p \sum_i E_{i+1/2}^{n+1/2} S(x_{i+1/2} - x_p) v_p dt = \]

\[ = \sum_i E_{i+1/2}^{n+1/2} \sum_p \int_0^{\Delta t} q_p S(x_{i+1/2} - x_p) v_p dt = \sum_i E_{i+1/2}^{n+1/2} j_{i+1/2}^{n+1/2} \Delta t \Delta x = \]

\[ = -\epsilon_0 \sum_i \frac{E_{i+1/2}^{n+1} + E_{i+1/2}^n}{2} (E_{i+1/2}^{n+1} - E_{i+1/2}^n) \Delta x = \]

\[ = -W_{E}^{n+1} + W_{E}^n, \]

\[ (K_p + W_{E}) |_{n+1}^{n} = 0. \]
Charge conservation

At any instant of time

\[
\frac{\partial \rho_i}{\partial t} = \frac{\partial \rho_i}{\partial x_p} \frac{\partial x_p}{\partial t} = \frac{1}{\Delta x} \sum_p \frac{\partial S_2(x_i - x_p)}{\partial x_p} q_p v_p = - \frac{1}{\Delta x} \sum_p \frac{\partial S_2(x_i - x_p)}{\partial x_i} q_p v_p =
\]

\[
= - \frac{1}{\Delta x} \sum_p S_1(x_{i+1/2} - x_p) - S_1(x_{i-1/2} - x_p) q_p v_p = - \frac{j_{i+1/2} - j_{i-1/2}}{\Delta x},
\]

Taking the time average

\[
\frac{1}{\Delta t} \int_{0}^{\Delta t} \left( \frac{\partial \rho_i}{\partial t} + \frac{j_{i+1/2} - j_{i-1/2}}{\Delta x} \right) dt = \frac{\rho_i^{n+1} - \rho_i^n}{\Delta t} + \frac{j_{i+1/2}^{n+1/2} - j_{i-1/2}^{n+1/2}}{\Delta x} = 0,
\]

Discrete Poisson equation:

\[
\varepsilon_0 \frac{E_{i+1/2}^{n+1} - E_{i-1/2}^{n+1}}{\Delta x} = \rho_i^{n+1}
\]
JFNK solver performance comparison

# of FE = # of Newton + # of GMRES
Hybrid implementation

Particle enslavement

Intel Xeon
X5460@3.16GHz
Single-core
theoretical peak
performance (SP)
25.2 GFLOPS

Nvidia GeForce
GTX 580@1.54GHz
many-core
theoretical peak
performance (SP)
1.58 TFLOPS

Maxwell eqs
$E^{n+1}$, $B^{n+1}$

Fluid moments
$\rho^{n+1}$, $u_\alpha^{n+1}$, $E_\alpha^{n+1}$, ...

Particle mover
$x^{p+1}$, $y^{p+1}$

Closure relations
$\Pi^{n+1}_\alpha$, $q^{n+1}_\alpha$, ...

Out

Convergence

Nonlinear iteration

G. Chen (gchen@lanl.gov) JIFT 2012
Fermi GPU architecture

- Support IEEE standards
- Support CUDA-C/C++, Fortran, Python, etc.
- Extreme multi-threading (~10,000 threads)
- 512 CUDA cores (16SMx32core/SM)
- High performance (TFLOPS peak, SP)
- Not all arithmetic operations are equally fast
- IEEE transcendental functions, division are slow
- Allow explicit access to fast memory
- Divergent control flows, memory collisions degrade performance
- Global memory is large, but slow
- CPU-GPU communication is slow

Basic Memory Management in PIC

- Read/write particles \{x_p, v_p, i_p\} from/to global memory
- Read E field from global memory through cache
- Current/charge first collected in fast memory, then written to global memory.
Roofline model

- Memory operations are very slow.
- Data reuse is important.


Balanced

Operational Intensity

Memory-bounded

Compute-bounded

Nvidia GPU (GTX 580)

- Fused mul-add
- Int mul-add
- Int add
- rsqrt, __fdividef (fast fun units)
- Sqrt (IEEE)
- / (IEEE)

Operational throughput (GOp/s)

Operational Intensity (Op/Byte)
Roofline model

- Memory operations are very slow.
- Data reuse is important.


Balanced

Operational Intensity

Memory-bounded

Compute-bounded

Nvidia GPU (GTX 580)

Fused mul-add

Int mul-add

Int add

rsqrt, __fdividef (fast fun units)

Sqrt (IEEE)

/ (IEEE)
Finite-difference particle orbit integration
self-adaptive, sub-stepping, charge-conserving

In each sub-timestep \( \nu \rightarrow \nu + 1 \)

\[
\frac{x_p^{\nu+1} - x_p^\nu}{\Delta \tau^\nu} = \frac{v_p^{\nu+1} + v_p^\nu}{2}
\]

\[
\frac{v_p^{\nu+1} - v_p^\nu}{\Delta \tau^\nu} = \frac{q_p}{m_p} \sum_i \frac{E_i^n + E_i^{n+1}}{2} S\left(x_i - \frac{x_p^\nu + x_p^{\nu+1}}{2}\right)
\]

- Local-error-controlled (time estimator).
- Implicit, time-centered, 2\(^{nd}\) order, unconditionally stable, non-dissipative (Crank-Nicolson method).
- Automatic charge-conservation (particle cell-crossing).

Resolving orbit is essential!
Re-thinking the algorithm for GPU performance

<table>
<thead>
<tr>
<th>while(1) {</th>
<th>Original (baseline)</th>
<th>Optimized</th>
<th>Principles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate sub-timestep</td>
<td>L2 norm, quadratic equation</td>
<td>L1 norm, split estimate w. abs and rel tol.</td>
<td>• Use fast operations.</td>
</tr>
<tr>
<td>Crank-Nicolson update</td>
<td>Picard iteration</td>
<td>Direct solve using fast div + correction</td>
<td>• Use fast memory.</td>
</tr>
<tr>
<td>Particle cell-crossing</td>
<td>Quadratic equation</td>
<td>Newton’s method</td>
<td>• Avoid memory collisions.</td>
</tr>
<tr>
<td>Collect current(VA)</td>
<td>Shared→global</td>
<td>Register→shared→global</td>
<td>• Use regular data-structure.</td>
</tr>
<tr>
<td>If(dt_p==dt) break;</td>
<td></td>
<td>Particle sort; Warp vote.all</td>
<td>• Load balance.</td>
</tr>
<tr>
<td>}; Collect charge(VP)</td>
<td></td>
<td></td>
<td>• Avoid divergent branches</td>
</tr>
</tbody>
</table>

Principles:
- Use fast operations.
- Use fast memory.
- Avoid memory collisions.
- Use regular data-structure.
- Load balance.
- Avoid divergent branches.
Optimize time-estimator

\[ \|l e(\Delta \tau)\|_2 < \varepsilon_a + \varepsilon_r \|r^0(\Delta \tau)\|_2 , \]

\[ \alpha \Delta \tau^2 - \beta \Delta \tau - \gamma^2 = 0 , \]

1. L2 norm \( \rightarrow \) L1 norm (no sqrt)
2. Split to two equations:

\[ \alpha \Delta \tau^2 = \gamma^2 , \]

\[ \alpha \Delta \tau = \beta , \]

3. Re-arrange:

\[ \Delta \tau = \max(\gamma d, \beta d^2) , \text{ with } d = \alpha^{-1/2} \]

4. Result: clock cycle 106 \( \rightarrow \) 28 (replaced 3 sqrt, 1 div, with 1 rsqrt).
Breakdown of run time of operations

- Factor of 3 overall improvement after optimizations
  - ✔ Absolute efficiency 20-25%
    = real ops/Absolute theoretical peak (=1.6TGOps)
  - ✔ Intrinsic efficiency 50-70%
    = real ops/theoretical peak of the algorithm (~600GOps)

- Memory operations are negligible.

- Atomic accumulations are expensive in VA (negligible in VP).
Sensitivity study of Performance & efficiency

Implicit Particle Mover Algorithm

Performance is most sensitive to timestep (large timestep $\rightarrow$ more work).
Strong scaling with number of threads

Consistent with Little’s law:

Needed parallelism = Latency × (# CUDA cores)

Instruction level latency (Fermi) ≈ 18 clock cycles

Scale up to theoretical limit
1D Long-timescale ion acoustic wave simulation

Mixed precision hybrid version
(CPU:DP, serial + GPU:SP, parallel) vs.

CPU-only (DP, serial) version

In this case, for the chosen parameters, parallel speed-up > 100.
(particles ~98% serial runtime)
Defect-correction for true double precision solution

- full DP CPU-GPU speedup=25.
- Newton’s method can converge to DP with iterations using SP.
  - Nonlinear residual in DP
  - Jacobian-vector product in SP
- In the IAW simulation, about 2/3 of particle mover calls in SP → speedup=40.
Summary

• Developed particle movers for implicit electrostatic PIC simulations.
• The adaptive, charge-conserving particle mover is compute-bounded.
• The implicit particle mover reaches 300-400 Gop/s on a GPU (SP). It reaches 20-25% absolute GPU efficiency, and 50-70% intrinsic efficiency.
• When combined with the field solver, we obtain a very efficient mixed-precision, hybrid CPU-GPU implementation for the newly developed charge- and energy-conserving implicit PIC algorithm.
• A long-timescale 1D ion acoustic wave simulation by the hybrid solver gains ~100x (vs. DP serial CPU version) without apparent loss of robustness or accuracy.