An Energy- and Charge-conserving, Implicit, Electrostatic Particle-in-cell Algorithm[1]

L. Chacón

Oak Ridge National Laboratory
P.O. Box 2008, Oak Ridge, TN 37830

Collaborators:
Guangye Chen (ORNL),
D. C. Barnes (Coronado Consulting)

ICNSP 11
Sept 7-9, 2011
Long Branch, NJ

Work supported by ORNL's Laboratory Directed Research & Development (LDRD) Program.

[1] **Chen, Chacon, Barnes, JCP (2011)**

Luis Chacon, chaconl@ornl.gov

Outline

- Particle methods for plasma simulation (PIC)
- State of the art algorithm: explicit approach
- Status of implicit PIC: problems and limitations
- Our approach: energy and charge-conserving implicit PIC
  - Vlasov-Ampere vs. Vlasov-Poisson
  - Exact energy-conserving formulation
  - Exact charge-conserving mover
  - Momentum conservation error control: orbit adaptivity
- Implementation on GPU architectures

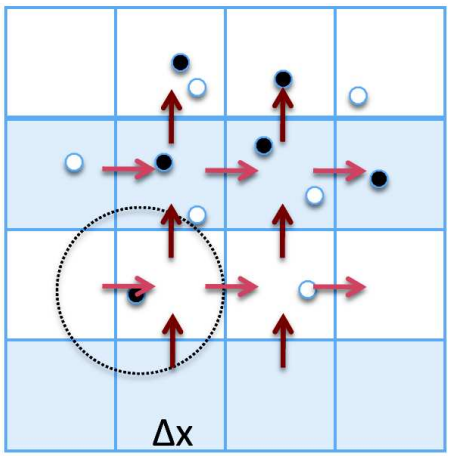
Particle-in-cell (PIC) methods for kinetic plasma simulation

$$\partial_t f + \mathbf{v}\cdot\nabla f + \frac{\mathbf{F}}{m}\cdot\nabla_v f = \left(\frac{\partial f}{\partial t}\right)_{col}$$

- Ignoring collisions$\Rightarrow$ Lagrangian solution by the **method of characteristics**:

$$f(\mathbf{x},\mathbf{v},t) = f_0\left(\mathbf{x} - \int_0^t dt\mathbf{v}, \mathbf{v} - \frac{1}{m}\int_0^t dt\mathbf{F}\right)\ ;\ \mathbf{x}(t=0) = \mathbf{x}_0\ ;\ \mathbf{v}(t=0) = \mathbf{v}_0$$

- PIC approach follows characteristics employing **macroparticles** (volumes in phase space)



$$f(\mathbf{x},\mathbf{v},t) = \sum_p \delta(\mathbf{x}-\mathbf{x}_p)\delta(\mathbf{v}-\mathbf{v}_p)$$

$$\begin{aligned}\dot{\mathbf{x}}_p &= \mathbf{v}_p\\ \dot{\mathbf{v}}_p &= \frac{q_p}{m_p}(\mathbf{E}+\mathbf{v}\times\mathbf{B})\end{aligned}$$

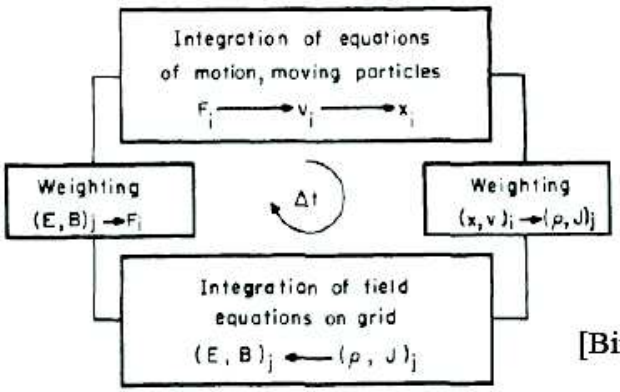
$$\begin{aligned}\partial_t\mathbf{B} + \nabla\times\mathbf{E} &= 0\\ -\mu_0\epsilon_0\partial_t\mathbf{E} + \nabla\times\mathbf{B} &= \mu_0\mathbf{j}\\ \nabla\cdot\mathbf{B} &= 0\\ \nabla\cdot\mathbf{E} &= \frac{e(n_i-n_e)}{\epsilon_0}\end{aligned}$$

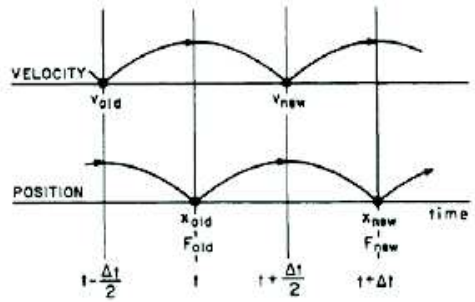
$$\boxed{\delta(\mathbf{x}-\mathbf{x}_p) \longrightarrow S(\mathbf{x}-\mathbf{x}_p)\ ;\ E_p = \sum_i E_i S(x_i - x_p)\ ;\ E_i = \sum_p E_i S(x_i - x_p)}$$

Luis Chacon, chaconl@ornl.gov

State-of-the-art *classical* PIC algorithm is explicit

- Classical explicit PIC approach "leap-frogs" particle positions and velocities, solves for fields after position update:





[Birdsall and Langdon, Plasma physics via computer simulation]

- Severe performance limitations:
  - ⇨ $\Delta x < \lambda_{Debye}$ (finite-grid instability: enforces a **minimum spatial resolution**)
  - ⇨ $\omega_{pe}\Delta t < 1$ (CFL-type instability: enforces a **minimum temporal resolution**)
  - ⇨ **Inefficient** for long-time, large-scale integrations
- In the presence of strong magnetic fields, gyro-averaging the Vlasov-Maxwell model can significantly ameliorate these limitations, but there are other issues (e.g. not asymptotic preserving, required order of expansion to capture some physical effects, treatment of nonlinear terms)

WE FOCUS ON ELECTROSTATIC PIC AS A PROOF OF PRINCIPLE

What about implicit PIC?

- Implicit PIC holds the promise of overcoming the difficulties and inefficiencies of explicit methods
- Exploration of implicit PIC started in the 1980s
  - Moment method [Mason, 1981; Brackbill, 1982]
  - Direct method [Friedman, Langdon, Cohen, 1981]
- Early approaches used linearized, semi-implicit formulations:
  - Lack of nonlinear convergence
  - Inconsistencies between particles and moments
  - Inaccuracies! $\longrightarrow$Plasma self-heating/cooling [Cohen, 1989]

Our goal is to explore the viability of a nonlinearly converged, fully implicit PIC algorithm

WHAT IS THE NATURE OF THE RESULTING FULLY-COUPLED ALGEBRAIC SYSTEM?
IS IT PRACTICAL TO INVERT?

Fully implicit PIC formulation

- A fully implicit formulation couples particles and fields non-trivially (integro-differential PDE):

$$\frac{f^{n+1}-f^n}{\Delta t}+\mathbf{v}\cdot\nabla\frac{f^{n+1}+f^n}{2} - \frac{q}{m}\nabla\frac{\Phi^{n+1}+\Phi^n}{2}\cdot\nabla_{\mathbf{v}}\frac{f^{n+1}+f^n}{2}=0$$

$$\nabla^2\Phi^{n+1} = \int d\mathbf{v}\, f^{n+1}(\mathbf{x},\mathbf{v},t)$$

- In PIC, f^{n+1} is sampled by a large collection of particles in phase space, $\{\mathbf{x},\mathbf{v}\}_p^{n+1}$.
  - There are N_p particles, each particle requiring $2\times d$ equations ($d\rightarrow$dimensions),
  - Field requires N_g equations, one per grid point.
- If implemented naively, an impractically large algebraic system of equations results:

$$\boxed{\mathbf{G}(\{\mathbf{x},\mathbf{v}\}_p^{n+1},\{\Phi^{n+1}\}_g)=0} \rightarrow \dim(\mathbf{G})=2dN_p+N_g\gg N_g$$

  - No current computing mainframe can afford the memory requirements
  - Algorithmic issues are showstoppers (e.g., how to precondition it?)
- An alternative strategy exists: nonlinear elimination (**particle enslavement**)

Particle enslavement (nonlinear elimination)

- Full residual $\mathbf{G}(\{x,v\}_p,\{\Phi\}_g)=0$ is impractical to solve
  - Very **large storage** requirements
  - **Inflexible** particle-orbit treatment (crucial for long-term accuracy)
- Alternatively, one can nonlinearly eliminate particle quantities so that they are not explicit unknowns:
  - Formally, particle equations of motion are functionals of the electrostatic potential:

$$x_p^{n+1}=x_p[\Phi^{n+1}]\ ;\ v_p^{n+1}=v_p[\Phi^{n+1}]$$

$$\mathbf{G}(\mathbf{x}_p{}^{n+1},\mathbf{v}_p{}^{n+1},\boldsymbol{\Phi}^{n+1})=\mathbf{G}(\mathbf{x}[\boldsymbol{\Phi}^{n+1}],\mathbf{v}[\boldsymbol{\Phi}^{n+1}],\boldsymbol{\Phi}^{n+1})=\tilde{\mathbf{G}}(\boldsymbol{\Phi}^{n+1})$$

Nonlinear residual can be *unambiguously* formulated in terms of electrostatic potential only!

- JFNK storage requirements are dramatically decreased, making it tractable:
  - Solver storage requirements $\propto N_g$, **comparable to a fluid simulation**
  - Particle quantities $\Rightarrow$ auxiliary variables: only a **single copy of particle population** needs to be maintained in memory throughout the nonlinear iteration

Jacobian-Free Newton-Krylov Methods

- After spatial and temporal discretization $\Rightarrow$ a large set of nonlinear equations: $\boxed{\vec{G}(\vec{x}^{n+1}) = \vec{0}}$
- Converging nonlinear couplings requires iteration: Newton-Raphson method:

$$\left.\frac{\partial \vec{G}}{\partial \vec{x}}\right|_k \delta\vec{x}_k = -\vec{G}(\vec{x}_k)$$

- Jacobian linear systems result, which require a linear solver $\Rightarrow$ Krylov subspace methods (GMRES)
  - Only require matrix-vector products to proceed.
  - Jacobian-vector product can be computed Jacobian-free:

$$\left(\frac{\partial \vec{G}}{\partial \vec{x}}\right)_k \vec{y} = J_k \vec{y} = \lim_{\epsilon \to 0} \frac{\vec{G}(\vec{x}_k + \epsilon \vec{y}) - \vec{G}(\vec{x}_k)}{\epsilon}$$

  - Krylov methods can be easily preconditioned: $P_k^{-1} \sim J_k^{-1}$

$$J_k P_k^{-1} P_k \delta\vec{x} = -\vec{G}_k$$

- In this study, we will use the **identity preconditioner**.

Luis Chacon, chaconl@ornl.gov

Field equation: Vlasov-Poisson vs. Vlasov-Ampere

- Nonlinear elimination procedure leads to $\mathbf{G}(\Phi) = 0$ (or $\mathbf{G}(E) = 0$)
- Two formulations are possible:

Vlasov-Poisson (VP)	Vlasov-Ampère (VA)
$\partial_t f + v\partial_x f + \frac{qE}{m}\partial_v f = 0$ $\partial_x E = \frac{\rho}{\epsilon_0}$ $E = -\partial_x \Phi$	$\partial_t f + v\partial_x f + \frac{qE}{m}\partial_v f = 0$ $\epsilon_0 \partial_t E + j = \langle j \rangle$
Two systems are equivalent in continuum, but not in the discrete.	
➤ Conventionally used in explicit PIC. ➤ Exact *local* charge conservation. ➤ Exact *global* momentum conservation. ➤ Unstable with orbit averaging in implicit context [Cohen and Freis, 1982].	➤ Exact *local* charge conservation. ➤ Exact *global* energy conservation. ➤ Suitable for orbit averaging. ➤ Can be extended to electromagnetic system.

- We will show, however, that an equivalent energy-conserving VP formulation exists.

Energy-conserving (EC) Vlasov-Ampère discretization

- Fully implicit Crank-Nicolson time discretization:

$$\varepsilon_0 \frac{E_i^{n+1} - E_i^n}{\Delta t} + \sum_p q_p v_p^{n+1/2} S(x_i - x_p^{n+1/2}) = 0$$

$$\frac{x_p^{n+1} - x_p^n}{\Delta t} = \frac{v_p^{n+1} + v_p^n}{2}$$

$$\frac{v_p^{n+1} - v_p^n}{\Delta t} = \frac{q_p}{m_p} \sum_i \frac{E_i^n + E_i^{n+1}}{2} S(x_i - x_p^{n+1/2})$$

In time:
centered, 2nd order;
implicit;
unconditionally stable;
non-dissipative.

- C-N enforces energy conservation to numerical round-off:

$$\sum_p \frac{m_p}{2}(v_p^{n+1} + v_p^n)(v_p^{n+1} - v_p^n) = -\sum_i \varepsilon_0 \frac{E_i^{n+1} - E_i^n}{\Delta t} \frac{E_i^{n+1} + E_i^n}{2} \Rightarrow \boxed{\sum_p \frac{1}{2} m v_p^2 + \sum_i \frac{1}{2} \varepsilon_0 E_i^2 = \textbf{const.}}$$

- As a result, the formulation does not suffer from finite-grid instabilities (normal mode analysis)
  - ⇨ Unconstrained spatial resolution: $\boxed{\Delta x \not< \lambda_D}$!!
- Energy conservation is only realized when particles and fields are nonlinearly converged:
  - ⇨ Requires a tight nonlinear tolerance

Luis Chacon, chaconl@ornl.gov

Algorithmic implementation details

- The nonlinear residual formulation $\mathbf{G}(E^{n+1})$ based on Vlasov-Ampere formulation is as follows:
  1. Input E (given by JFNK iterative method)
  2. Move particles (i.e., find $x_p[E]$, $v_p[E]$ by solving equations of motion)
     (a) Requires inner (local) nonlinear iteration: Picard (not stiff)
     (b) Can be as complicated as we desire (substepping, adaptivity, etc)
  3. Compute moments (current)
  4. Form Vlasov-Ampere equation residual
  5. return
- Because particle move is performed within function evaluation, we have much freedom.
- Rest of the talk will describe improvements in particle mover to ensure long-term accuracy
  - **Particle substepping and orbit averaging** (ensures orbit accuracy and preserves exact energy conservation)
  - **Exact charge conservation strategy** (a new charge-conserving particle mover)
  - **Orbit adaptivity** (to improve momentum conservation)

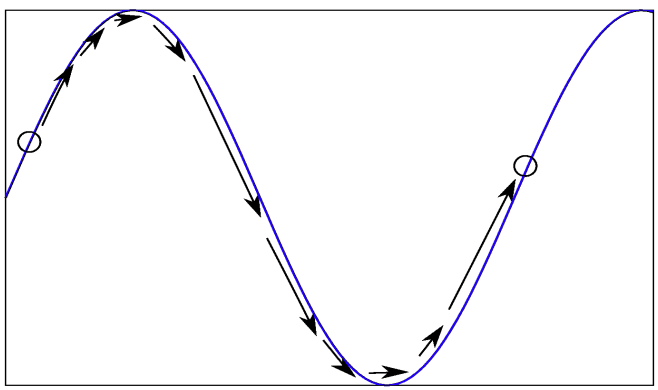
Luis Chacon, chaconl@ornl.gov

Particle orbit substepping

- In applications of interest, field time-scale (Δt) and orbit time-scale ($\Delta\tau$) can be well separated
  - Fields evolve *slowly* (dynamical time scale, Δt)
  - Particle orbits may still undergo *rapid change* ($\Delta\tau \ll \Delta t$)
- Particle orbits need to be resolved to **avoid large orbit integration errors**

Accurate orbit integration requires particle substepping!

- Field does not change appreciably: time-averaged value over long time scale is sufficient



$$\frac{x_p^{\nu+1} - x_p^{\nu}}{\Delta\tau} = v_p^{\nu+1/2}$$

$$\frac{v_p^{\nu+1} - v_p^{\nu}}{\Delta\tau} = \sum_i \underbrace{\frac{E_i^{n+1} + E_i^{n}}{2}}_{\text{slow}} S(x_i - x_p^{\nu+1/2})$$

Energy conservation and orbit averaging

- **Particle substepping breaks energy conservation**.
- Energy conservation theorem can be recovered by orbit averaging Ampère's law:

$$\epsilon_0 \partial_t E + j = \langle j \rangle \quad , \quad \frac{1}{\Delta t}\int_t^{t+\Delta t} d\tau [\cdots] \Rightarrow \epsilon_0 \frac{E^{n+1} - E^n}{\Delta t} + \bar{j} = \langle \bar{j} \rangle$$

- Orbit-averaged current is found as:

$$\bar{j} = \frac{1}{\Delta t}\int_t^{t+\Delta t} d\tau\, j \approx \frac{1}{\Delta t}\sum_p \sum_{\nu=1}^{N_\nu} q_p v_p S(x - x_p)\Delta\tau^\nu$$

- With these definitions, exact energy conservation is recovered:

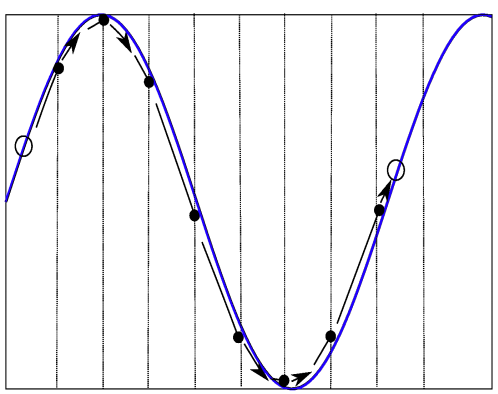
$$\sum_p \sum_\nu \frac{m_p}{2}(v_p^{\nu+1} + v_p^\nu)(v_p^{\nu+1} - v_p^\nu) = -\sum_i \epsilon_0 \frac{E^{n+1} - E^n}{\Delta t}\frac{E_i^{n+1} + E_i^n}{2}$$

$$\Rightarrow \sum_p \frac{1}{2} m_p v_p^2 + \sum_i \frac{1}{2}\epsilon_0 E_i^2 = \text{const.}$$

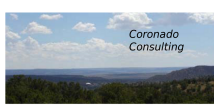
Exact charge conservation: charge-conserving particle mover

- Local charge conservation (enforced in the continuum by Gauss' law) is violated in discrete Vlasov-Ampère formulation.
- Local charge conservation is essential to ensure long-term accuracy of numerical algorithm
- Exact charge conservation requires a particle mover that satisfies a discrete charge continuity equation, $\partial_t \rho + \nabla \cdot \mathbf{j} = 0$ [Buneman 1968, Morse and Nielson, 1971]
  - ⇨ **Standard strategy** based on current redistribution when particle crosses boundary.
  - ⇨ In our context, current redistribution breaks energy conservation. **Need new strategy**.

Here, charge conservation is enforced by stopping particles at cell boundaries.



$$\left.\begin{aligned} \rho_{i+\frac{1}{2}} &= \sum_p q_p \frac{S_m(x - x_{i+\frac{1}{2}})}{\Delta x} \\ j_i &= \sum_p q_p v_p \frac{S_{m-1}(x - x_i)}{\Delta x} \\ S'_m(x) &= \frac{S_{m-1}(x+\frac{\Delta x}{2}) - S_{m-1}(x-\frac{\Delta x}{2})}{\Delta x} \end{aligned}\right\} \overset{(m=1,2)}{\Longrightarrow} [\partial_t \rho + \nabla \cdot \mathbf{j} = 0]_{i+\frac{1}{2}}^{n+\frac{1}{2}} = 0$$



Luis Chacon, chaconl@ornl.gov

Equivalence of VA and VP in the discrete form

- Exact local charge conservation allows a strict equivalence **between Vlasov-Ampere and Vlasov-Poisson in the discrete**:

$$\left.\begin{array}{ll} \text{VA}\rightarrow & \varepsilon_0 \dfrac{E_i^{n+1}-E_i^n}{\Delta t} + j_i^{n+1/2} = 0, \\ \text{CC}\rightarrow & \dfrac{\rho_{i+1/2}^{n+1}-\rho_{i+1/2}^{n}}{\Delta t} = -\dfrac{j_{i+1}^{n+1/2}-j_i^{n+1/2}}{\Delta x}, \\ \text{VP}^{\text{n}}\rightarrow & E_{i+1}^n - E_i^n = \rho_{i+1/2}^n \Delta x, \end{array}\right\} \Rightarrow E_{i+1}^{n+1} - E_i^{n+1} = \rho_{i+1/2}^{n+1}\Delta x. \leftarrow \text{VP}$$

THEREFORE, OUR APPROACH IS EQUIVALENT TO AN
exactly energy-conserving IMPLICIT **Vlasov-Poisson** FORMULATION.

- Properties of "equivalent" implicit Vlasov-Poisson:
  - **No orbit-averaging** is needed (charge density ρ is **not** orbit-averaged).
  - It **remains exactly energy conserving** (but one needs to accumulate orbit-averaged current to check)
- Why Vlasov-Ampere? Generalizes to the case with magnetic fields!

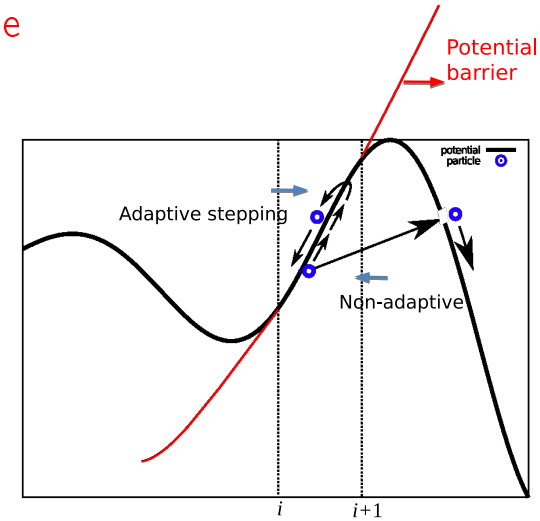
Luis Chacon, chaconl@ornl.gov

Momentum conservation: adaptive orbit integrator

- EC/CC PIC algorithm does not enforce momentum conservation exactly.
  - ⇨ **Controlling error** in momentum conservation is **crucial** for long-term accuracy
- Orbit integration errors can significantly affect momentum conservation: **particle tunneling**

- Adaptive orbit integration can be effective in suppressing particle tunneling and thus improve momentum conservation
- **Approach**: find $\Delta\tau$ to control local truncation error

$$\|l.e\| = \frac{q_p \Delta\tau^2}{2m} \left\| \begin{array}{c} E_p^0 \\ \frac{\partial E}{\partial x}\Big|_p v_p^0 \end{array} \right\| + H.O.T < \epsilon_0 + \epsilon_r \Delta\tau \left\| \begin{array}{c} v_p^0 \\ \frac{q E_p^0}{m} \end{array} \right\|$$



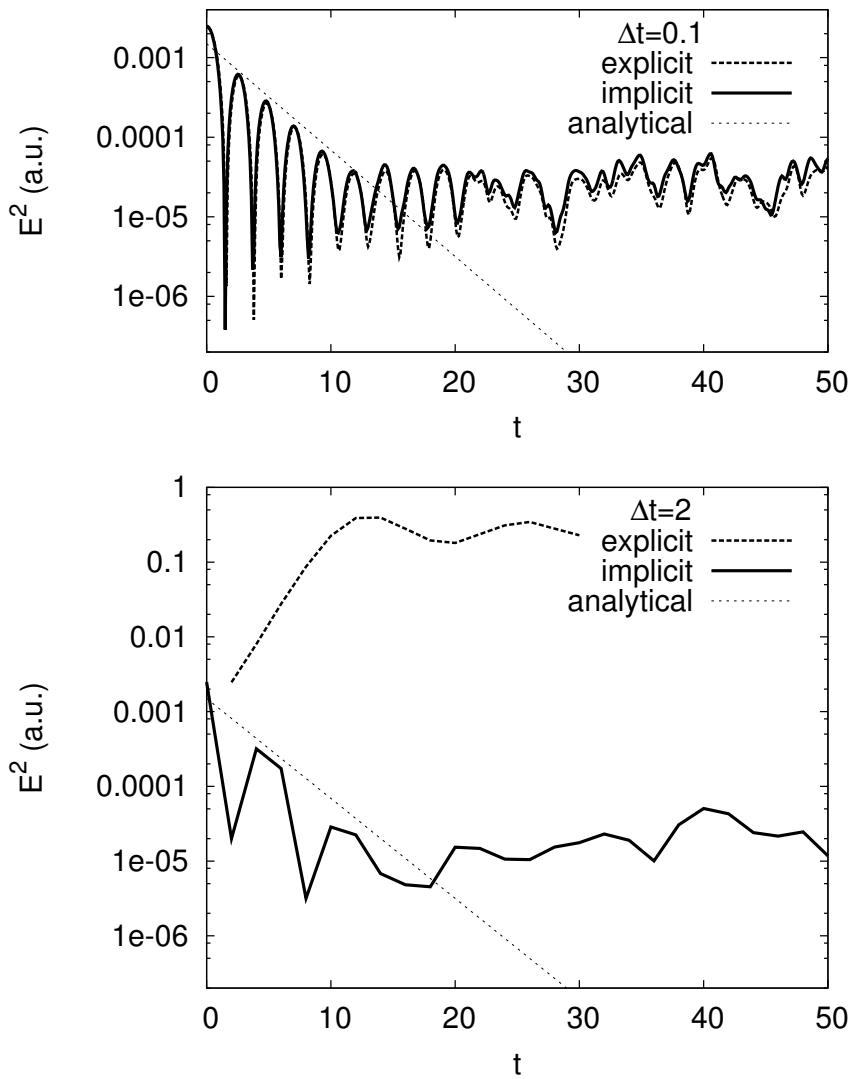
- Electric field gradient is estimated from cell-based gradient: $\frac{\partial E}{\partial x}\Big|_p \approx \frac{E_{i+1}-E_i}{\Delta x}$. **Provides potential barrier!**

- Quadratic equation is solved for $\Delta\tau$.
- **Particle is stopped at cell boundaries** to ensure charge conservation.

Luis Chacon, chaconl@ornl.gov

Landau damping test

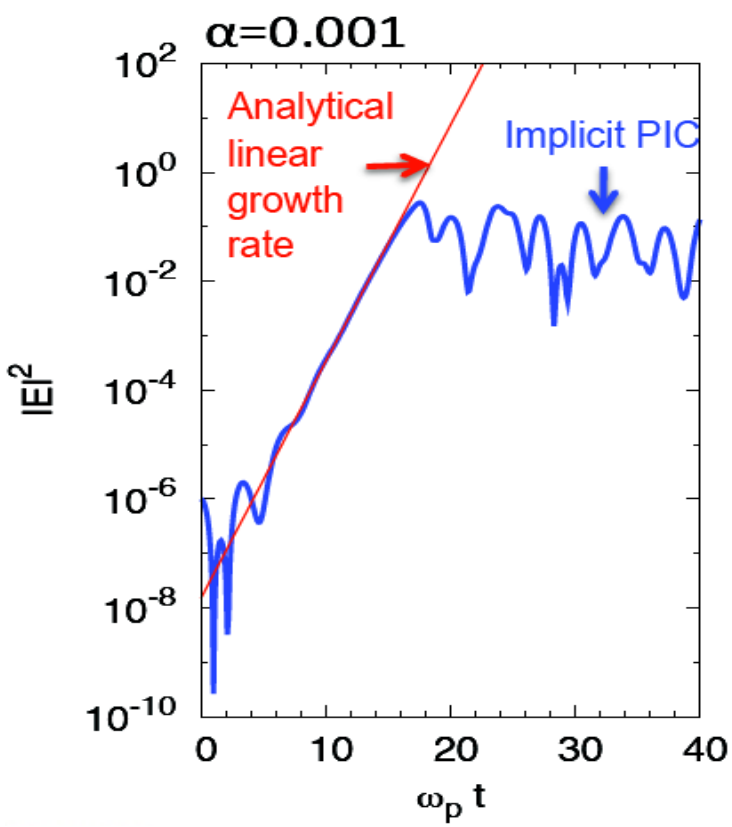
- Periodic domain [0,1], $N_x = 32$
- One-step energy-conserving (EC) Crank-Nicolson solver
- Single-species (electrons; cold uniform ion background), $N_p = 4 \times 10^4$
- Initial condition: $f(x, v, t = 0) = f_0[1 + \alpha \cos(kx)]$

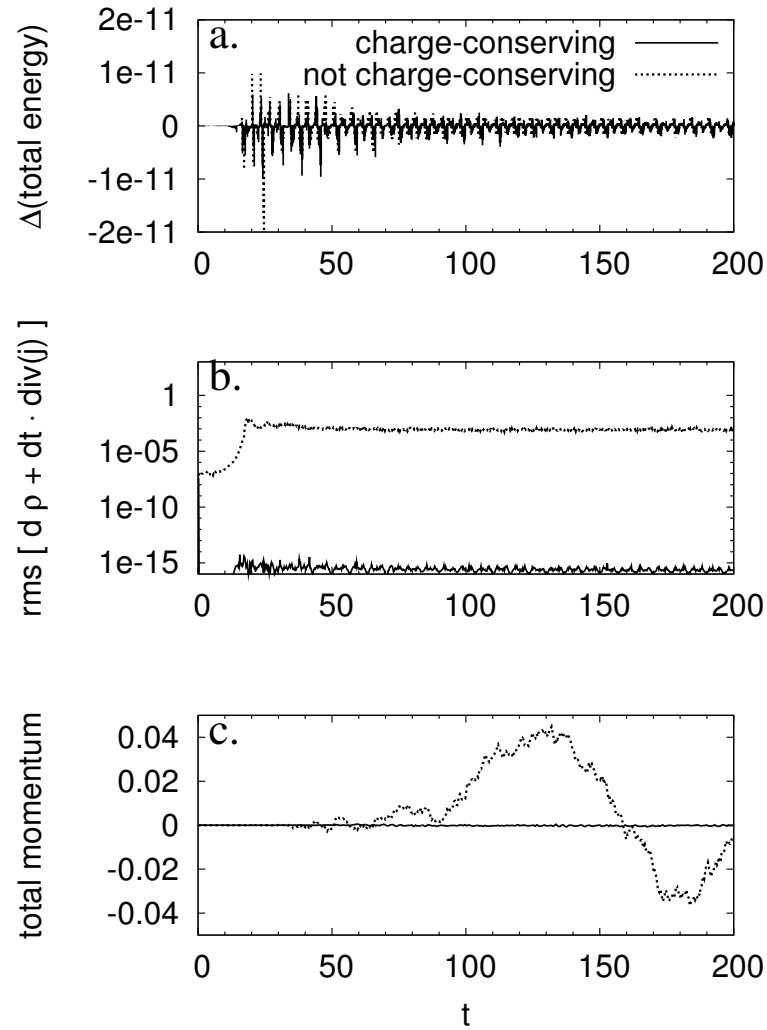


Luis Chacon, chaconl@ornl.gov

Two-stream test: impact of charge conservation

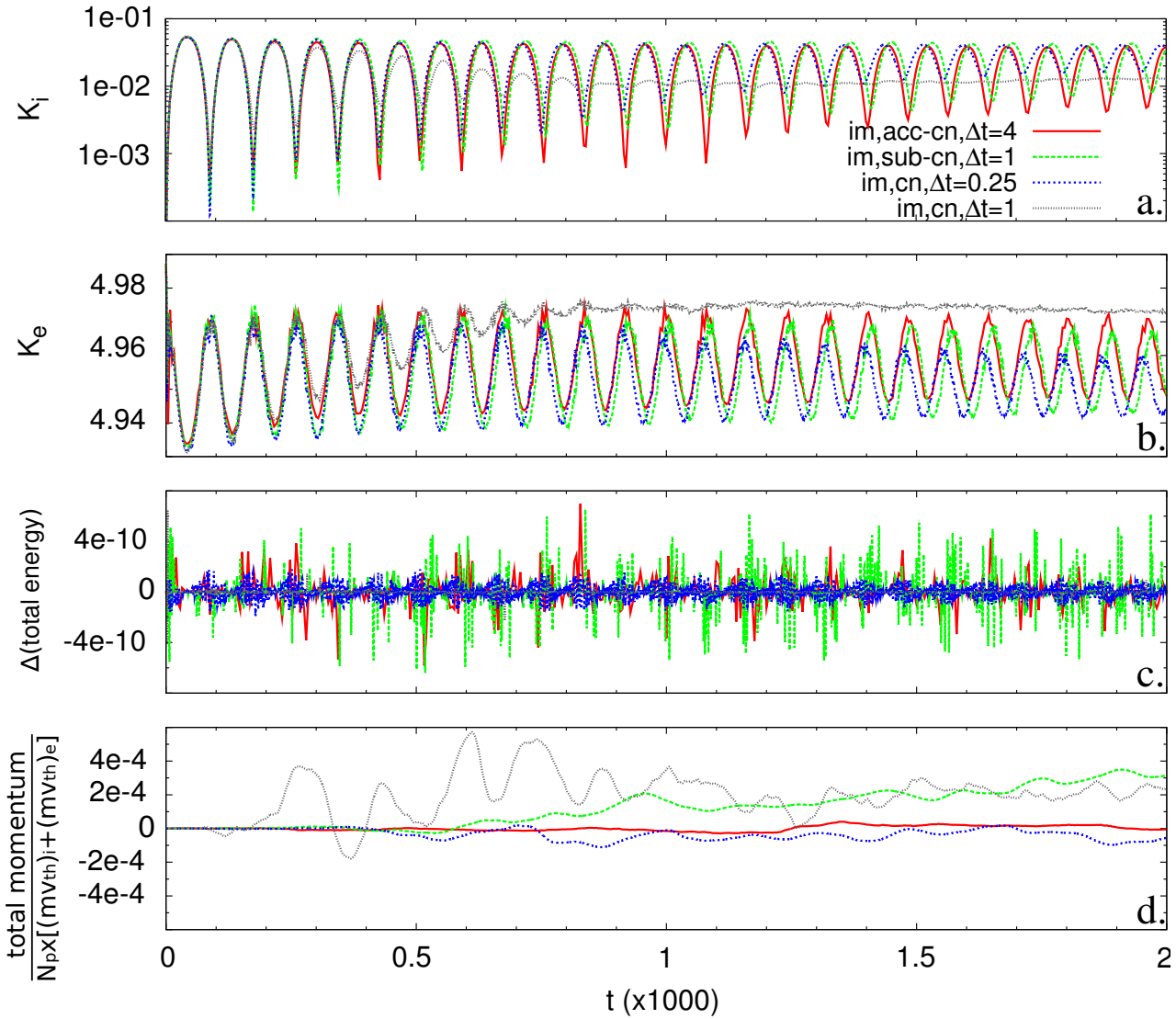
- Periodic domain [0,1], $N_x = 64$
- One-step EC Crank-Nicolson solver
- Single-species (electrons), $N_p = 10^4$
- $f(x, v, t = 0) = f_0[1 + \alpha \cos(kx)]$





Luis Chacon, chaconl@ornl.gov

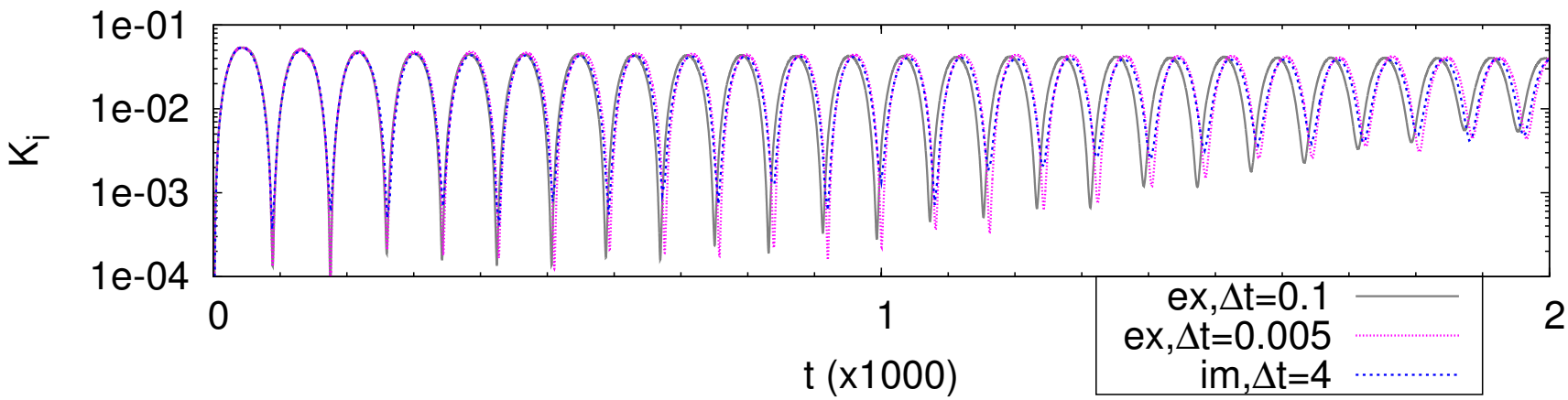
Ion acoustic wave (IAW): accuracy impact of different EC movers



Luis Chacon, chaconl@ornl.gov

IAW: explicit vs. implicit (accuracy)

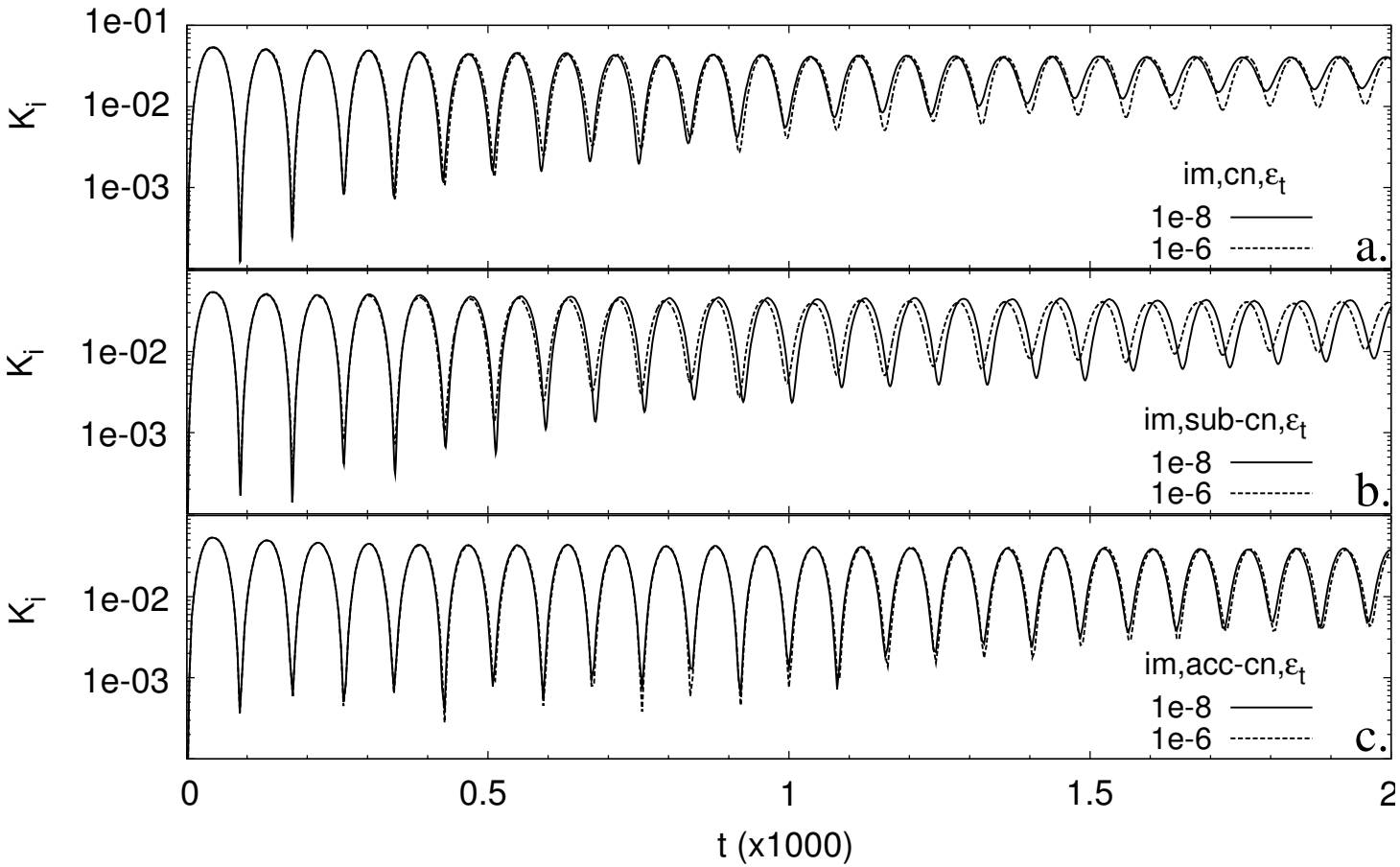
- Compare large-time-step implicit IAW vs explicit at CFL
- Found that explicit at CFL was not as accurate as implicit with $\Delta t \gg \Delta t_{CFL}$!!!



- CFL time-step is an "average" quantity (based on thermal velocity), and thus may still introduce inaccuracies in fast particles.

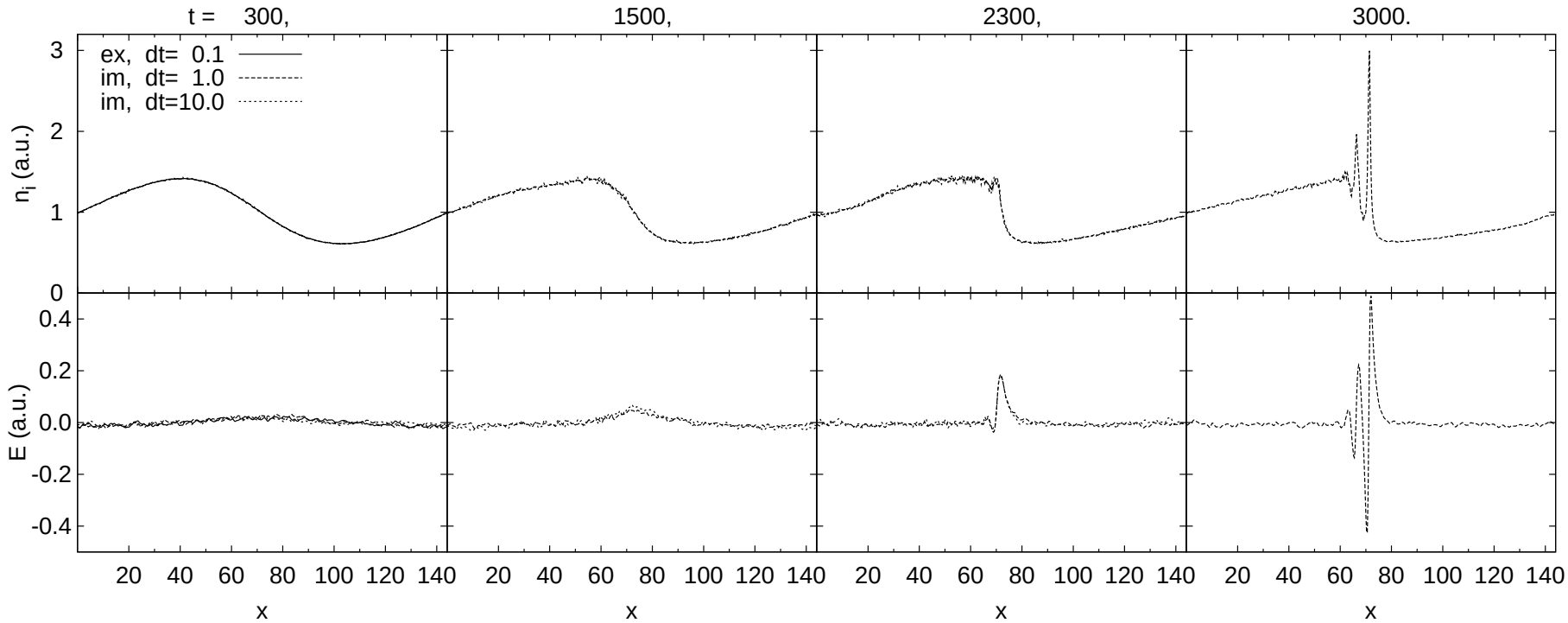
IAW: effect on nonlinear tolerance

- Exact energy conservation of implicit mover only holds for exact nonlinear solve
- It is of interest to understand robustness of mover when employing finite nonlinear tolerances



Adaptive-CC mover is the most robust!

Ion acoustic shock wave



- Propagating IAW with perturbation level $\epsilon = 0.4$, with 4000 particles/cell.
- Realistic mass ratio ($m_i/m_e = 2000$).
- Shock wave length scale$\sim$Debye length.

Luis Chacon, chaconl@ornl.gov

CPU gain potential of implicit PIC vs. explicit PIC

- Back-of-the-envelope estimate of CPU gain:

$$CPU \sim \left(\frac{T}{\Delta t}\right)\left(\frac{L}{\Delta x}\right)^d n_p C^{solver} \quad ; \quad \frac{C^{imp}}{C^{ex}} \sim N_{FE}\frac{\Delta t_{imp}}{\Delta \tau_{imp}} \quad ; \quad \frac{CPU_{ex}}{CPU_{imp}} \sim \left(\frac{\Delta x_{imp}}{\Delta x_{ex}}\right)^d \frac{\Delta \tau_{imp}}{\Delta t_{ex}}\frac{1}{N_{FE}}$$

- Using reasonable estimates:

$$n_p = \textbf{const.}$$

$$\Delta\tau_{im} \sim 0.1\frac{\Delta x_{im}}{v_{th}} \qquad (kv_{th}\Delta\tau<1)$$

$$\Delta t_{ex} \sim 0.1/\omega_{pe}$$

$$k\Delta x_{im} \sim 0.2$$

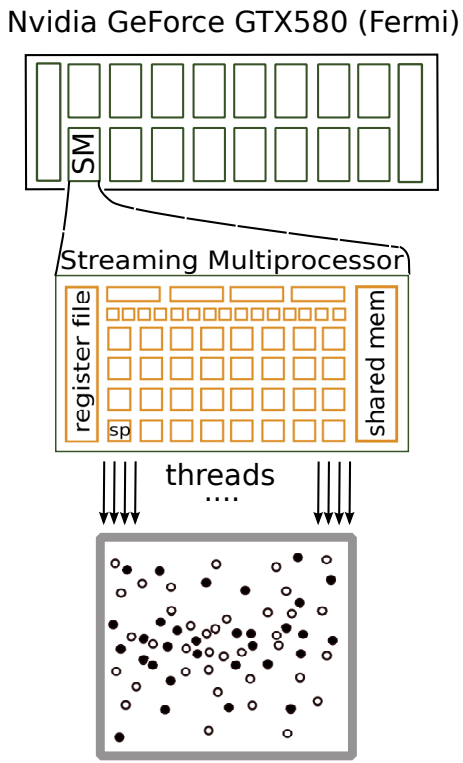
$$k\Delta x_{ex} \sim \lambda_D/3$$

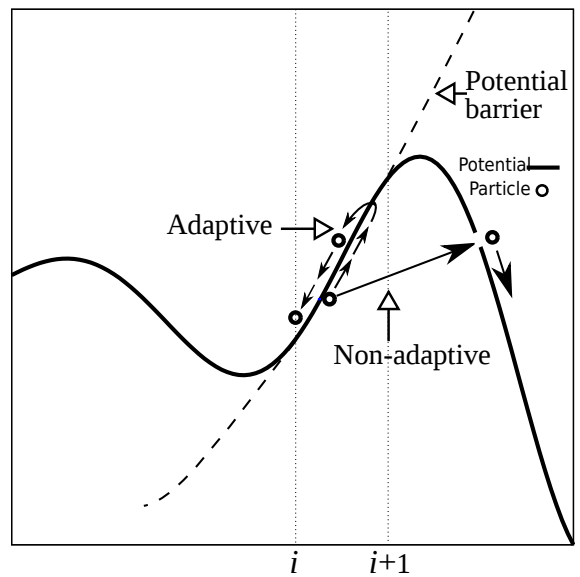
$$\Rightarrow \boxed{\frac{CPU_{im}}{CPU_{ex}} \sim \frac{1}{(k\lambda_D)^{d+1}}\frac{1}{N_{FE}} \quad >>1 \text{ if } k\lambda_D<<1.}$$

L	$k\lambda_D$	$\frac{N_x^{ex}}{N_x^{im}}$	$\frac{\Delta t_{im}}{\Delta t_{ex}}$	N_{FE}	$\frac{CPU_{ex}}{CPU_{im}}$
10	0.628	1	50	13.7	0.25
20	0.314	2	100	20	0.58
40	0.157	4	200	31.2	0.95
80	0.078	8	200	35.8	2.18
160	0.039	16	200	43.6	5.41
160	0.039	16	400	72.1	3.64
320	0.02	32	200	49.6	15.4
320	0.02	32	400	67.6	11.96

Implementation of ACC particle mover on GPU architectures

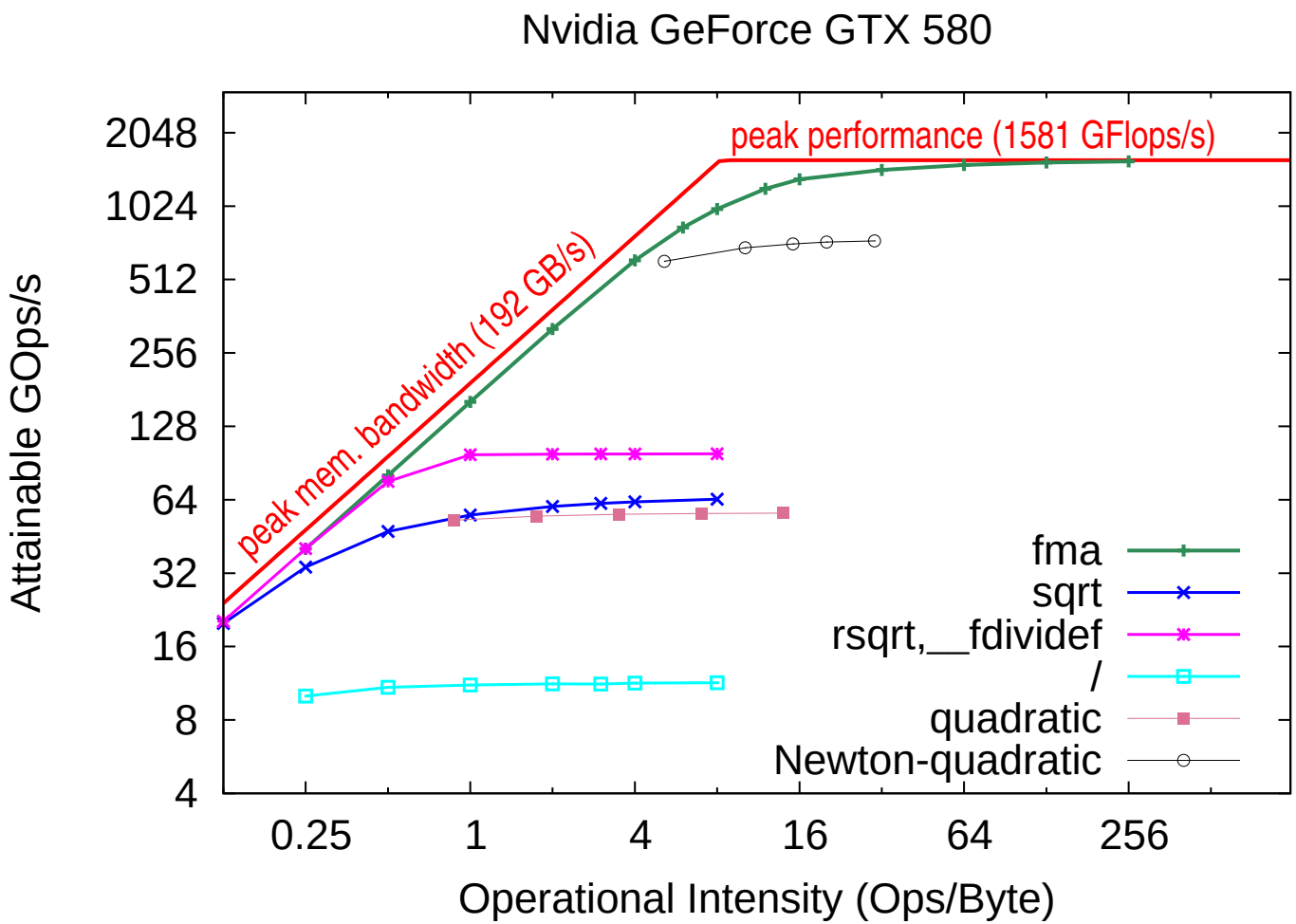
- Particle orbits are independent of each other ⇒ PIC algorithms are naturally data parallel.





- Potential performance killers for our implicit PIC ACC particle mover:
  - Particle motion is **self-adaptive** (orbit accuracy) ⇒**workload imbalances**.
  - Particles **stop at cell boundaries** (charge conservation) ⇒**dynamic control flows**.

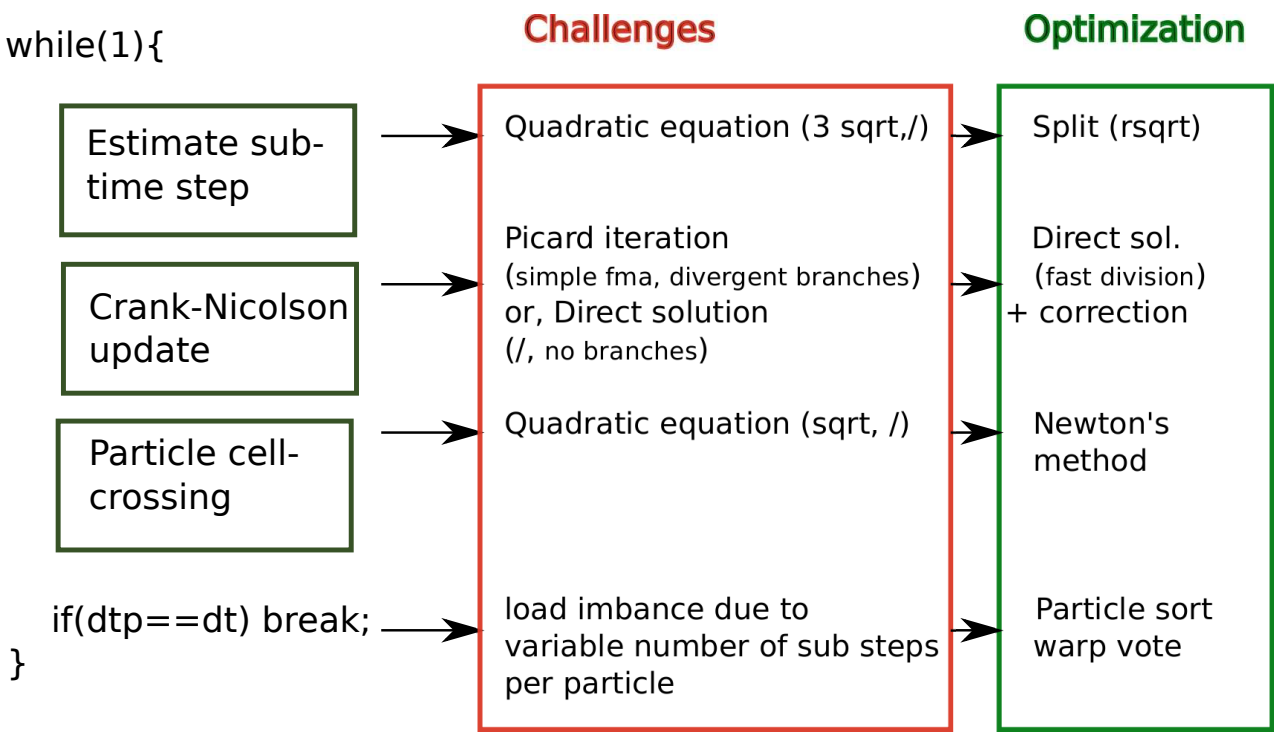
Algorithm optimization on GPU: roofline model[2]



[2] S. Williams, A. Waterman, and D. Patterson, *Comm. ACM*, **52** (94) 2009

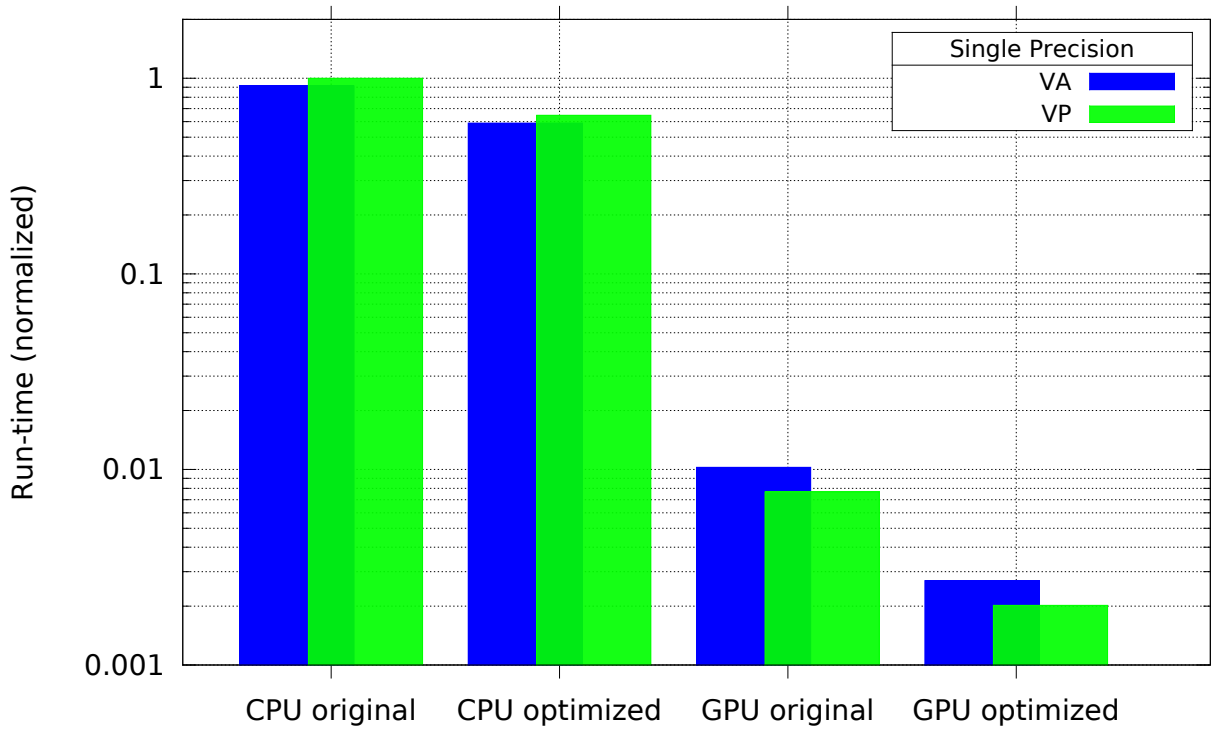
Optimization of ACC implicit particle mover

- Computationally intensive –> compute-bounded (vs. explicit schemes, typically memory-bounded)
- While loop introduces **control flow latencies and branch divergences**.
- Requires expensive operations **(sqrt, division)**, atomicAdd (for moment accumulation)



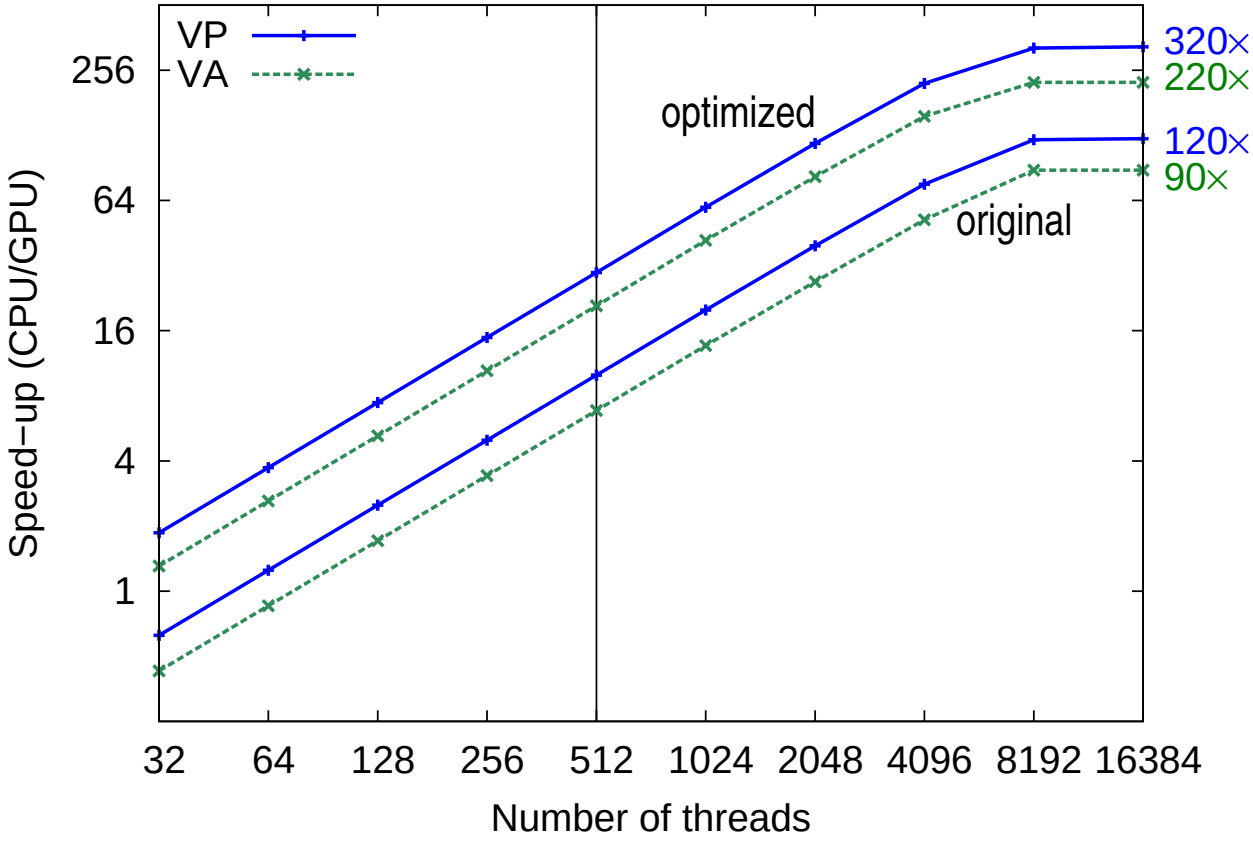
Luis Chacon, chaconl@ornl.gov

Performance results porting from CPU to GPU (single precision)



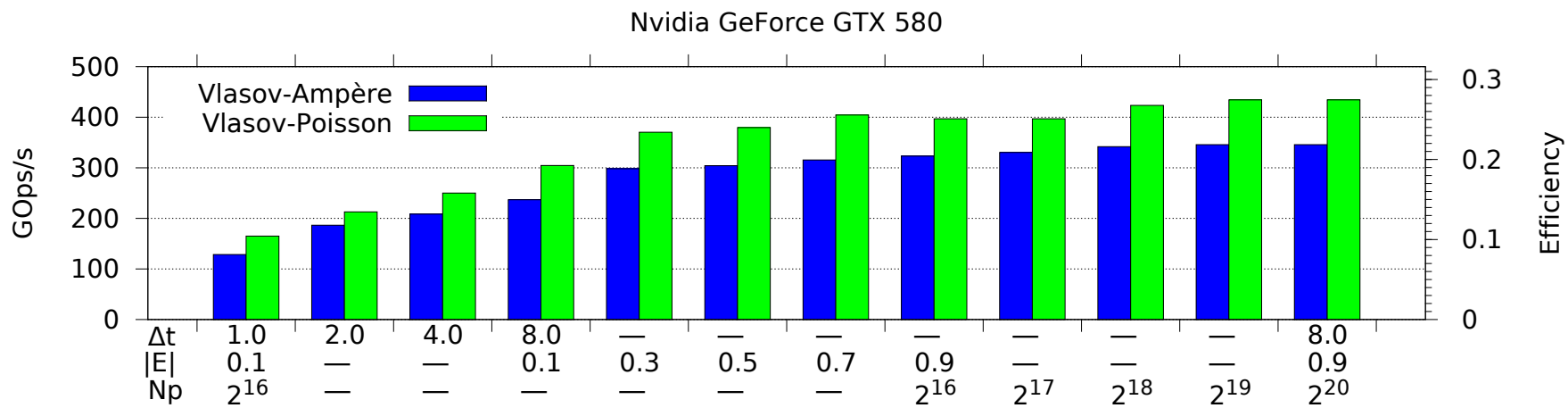
- Straightforward GPU implementation accelerates ∼100 times;
- Optimizations have larger effects on GPU; not all optimizations introduced are effective on CPU.
- GPU-CPU speedup $\sim 200 - 300$, depending on algorithm (VA, VP)

Scaling CPU-to-GPU speedup with number of threads



- Hardware limit is 512 threads (=32 cores/SMx16 SM/GPU) running concurrently;
- Large number of threads (≫512) are useful to hide latencies.

Sensitivity of GPU performance and efficiency



- All operations including floating, integer, and special functions are counted.
- Varied E, Δt, N_p to test performance sensitivity
  - ⇨ Performance is most sensitive to Δt: **more efficient for large Δt!**
- 300 to 400 GOps/s (20-30% efficiency of GPU peak) are obtained for large time steps, strong fields and many particles.

Summary and conclusions

- We have demonstrated, for the first time, a fully implicit, fully nonlinear PIC formulation that features:
  - **Exact charge conservation** (via a novel particle mover strategy).
  - **Exact energy conservation** (no particle self-heating or self-cooling).
  - **Adaptive particle orbit integrator** to control errors in momentum conservation.
- Central to our implementation is the concept of particle enslavement, which eliminates particle quantities as unknowns and affords us much freedom in the particle mover stage.
- The approach has been shown to be free of CFL and finite-grid numerical instabilities.
- As a result, the method is able to take time steps many times larger than explicit, and resolutions many times coarser.
- The method has much potential for efficiency gains vs. explicit, with the CPU speedup scaling as $(k\lambda_D)^{-d-1}/N_{FE}$.
- Key to realizing the potential of the approach is to minimize the number of nonlinear function evaluations.
  - This, in turn, requires preconditioning, which will be the subject of future work.
- We have ported the algorithm to GPU architectures (NVIDIA GeForce GTX 580)
  - **20-30% efficiency** of GPU peak (single precision)
  - **300× speedup** over a single Intel(R) Xeon(R) CPU X5460 @ 3.16GHz (single precision)