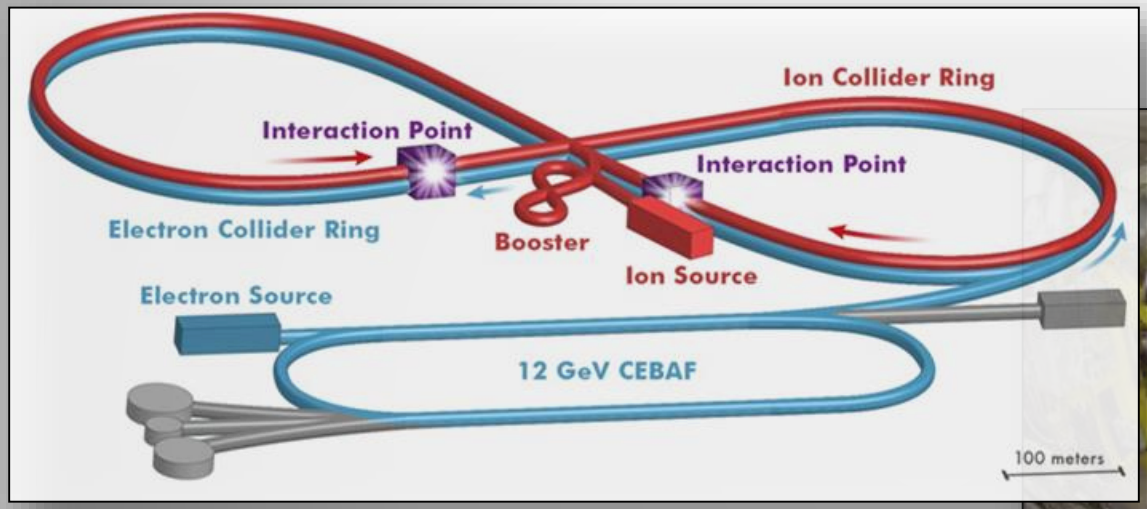
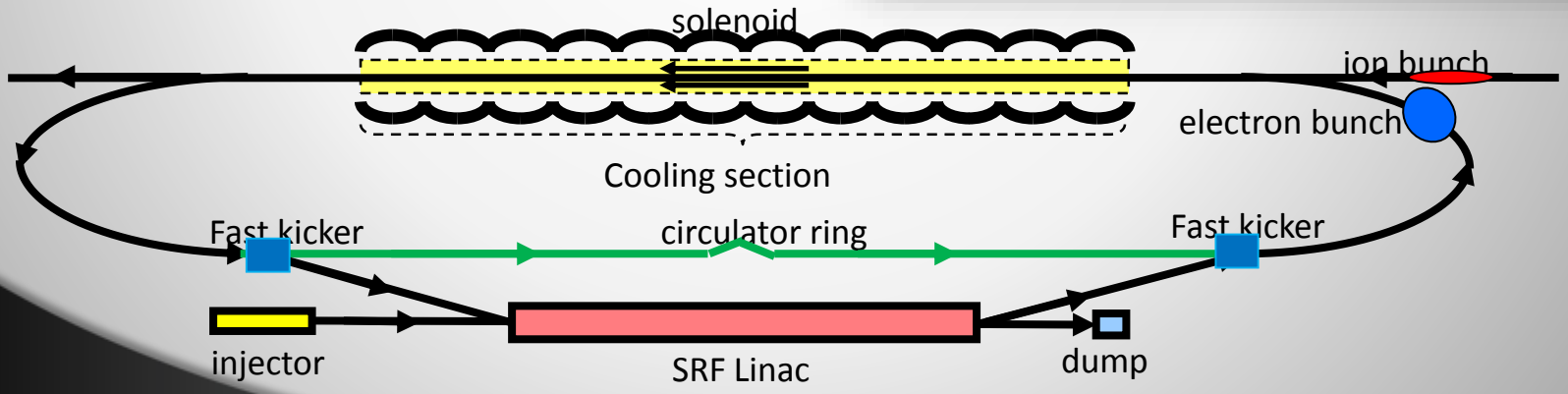
Electron Cooling for an Electron Ion Collider: Computational Methods and Code Development

Béla Erdélyi
Northern Illinois University

DOE-NP Accelerator R&D PI Meeting
November 14, 2016

Electron Cooling of Heavy Ions





Main Goals

- **First particle-based** proof of principle **demonstration** of electron cooling simulations
- **Develop a high-performance code** with capabilities in beam dynamics beyond cooling
- **Applications** to electron-ion colliders

Expenditures and Milestones

	FY10+FY11	FY12+FY13	FY14+FY15	FY16	TOTALS
Funds Allocated	0+56	55+52=107	50+54=104	50	$317K
Actual Costs to Date	56	107	104	50	$317K

	FY16	FY17
Quarter 1	Shared memory parallelization of FMM data structures and integration with parallel FMM	Electron cooling simulations in the JLEIC pre-booster at injection
Quarter 2	Variable order Picard integrator with automatic step size control parallelization	Electron cooling simulations in the pre-booster at extraction energy
Quarter 3	Binned time step implementation in parallel	Setup the re-circulator ring optics and bunched cooling in COSY
Quarter 4	Parallel PHAD integration, benchmarking and optimizations	Study electron beam dynamics in the re-circulator ring, set maximum useful turn limits

Cooling as an N-Body Problem

- Many **collective beam dynamics** effects can be cast in the form of an N-body problem: space charge, intra-beam scattering, electron cloud, beam-beam, beam-plasma, etc.
- **Electron cooling** is one of the most challenging:
  - Accurate analytical estimates are difficult to come by
  - Large particle numbers, but far from statistical limit
  - Both attractive and repelling forces
  - Close encounters matter
  - Relatively slow process

Algorithms in Scientific Computing

from *SIAM News*, Volume 33, Number 4

The Best of the 20th Century: Editors Name Top 10 Algorithms

By Barry A. Cipra

Algos is the Greek word for pain. *Algor* is Latin, to be cold. Neither is the root for *algorithm*, which stems instead from al-Khwarizmi, the name of the ninth-century Arab scholar whose book *al-jabr wa'l muqabalah* devolved into today's high school algebra textbooks. Al-Khwarizmi stressed the importance of methodical procedures for solving problems. Were he around today, he'd no doubt be impressed by the advances in his eponymous approach.

Some of the very best algorithms of the computer age are highlighted in the January/February 2000 issue of *Computing in Science & Engineering*, a joint publication of the American Institute of Physics and the IEEE Computer Society. Guest editors Jack Don-garra of the University of Tennessee and Oak Ridge National Laboratory and Fran-cis Sullivan of the Center for Comput-ing Sciences at the Institute for Defense Analyses put togeth-er a list they call the "Top Ten Algorithms of the Century."

"We tried to assemble the 10 al-gorithms with the greatest influence on the development and practice of science and engineering in the 20th century," Dongarra and Sullivan write. As with any top-10 list, their selections—and non-selections—are bound to be controversial, they acknowledge. When it comes to picking the algorithmic best, there seems to be no best algorithm.

Without further ado, here's the CiSE top-10 list, in chronological order. (Dates and names associated with the algorithms should be read as first-order approximations. Most algorithms take shape over time, with many contributors.)

1946: John von Neumann, Stan Ulam, and Nick Metropolis, all at the Los Alamos Scientific Laboratory, cook up the Metropolis algorithm, also known as the **Monte Carlo method**.

The Metropolis algorithm aims to obtain approximate solutions to numerical problems with unmanageably many degrees of freedom and to combinatorial problems of factorial size, by mimicking a random process. Given the digital computer's reputation for deterministic calculation, it's fitting that one of its earliest applications was the generation of random numbers.

In terms of widespread use, George Dantzig's simplex method is among the most successful algorithms of all time.

1947: George Dantzig, at the RAND Corporation, creates the **simplex method for linear programming**.

In terms of widespread application, Dantzig's algorithm is one of the most successful of all time: Linear programming dominates the world of industry, where economic survival depends on the ability to optimize within budgetary and other constraints. (Of course, the "real" problems of industry are often nonlinear; the use of linear programming is sometimes dictated by the computational budget.) The simplex method is an elegant way of arriving at optimal answers. Although theoretically susceptible to exponential delays, the algorithm in practice is highly efficient—which in itself says something interesting about the nature of computation.

1950: Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos, all from the Institute for Numerical Analysis at the National Bureau of Standards, initiate the development of **Krylov subspace iteration methods**.

These algorithms address the seemingly simple task of solving equations of the form $Ax = b$. The catch, of course, is that A is a huge $n \times n$ matrix, so that the algebraic answer $x = b/A$ is not so easy to compute. (Indeed, matrix "division" is not a particularly useful concept.) Iterative methods—such as solving equations of the form $Kx_{i+1} = Kx_i + b - Ax_i$ with a simpler matrix K that's ideally "close" to A—lead to the study of Krylov subspaces. Named for the Russian mathematician Nikolai Krylov, Krylov subspaces are spanned by powers of a matrix applied to an initial "remainder" vector $r_0 = b - Ax_0$. Lanczos found a nifty way to generate an orthogonal basis for such a subspace when the matrix is symmetric. Hestenes and Stiefel proposed an even niftier method, known as the conjugate gradient method, for systems that are both symmetric and positive definite. Over the last 50 years, numerous researchers have improved and extended these algorithms. The current suite includes techniques for non-symmetric systems, with acronyms like GMRES and Bi-CGSTAB. (GMRES and Bi-CGSTAB premiered in *SIAM Journal on Scientific and Statistical Computing*, in 1986 and 1992, respectively.)

Alston Householder

1951: Alston Householder of Oak Ridge National Laboratory formalizes the **decompositional approach to matrix computations**.

The ability to factor matrices into triangular, diagonal, orthogonal, and other special forms has turned out to be extremely useful. The decompositional approach has enabled software developers to produce flexible and efficient matrix packages. It also facilitates the analysis of rounding errors, one of the big bugbears of numerical linear algebra. (In 1961, James Wilkinson of the National Physical Laboratory in London published a seminal paper in the *Journal of the ACM*, titled "Error Analysis of Direct Methods of Matrix Inversion," based on the LU decomposition of a matrix as a product of lower and upper triangular factors.)

1957: John Backus leads a team at IBM in developing the **Fortran optimizing compiler**.

The creation of Fortran may rank as the single most important event in the history of computer programming: Finally, scientists (and others) could tell the computer what they wanted it to do, without having to descend into the netherworld of machine code. Although modest by modern compiler standards—Fortran I consisted of a mere 23,500 assembly-language instructions—the early compiler was nonetheless capable of surprisingly sophisticated computations. As Backus himself recalls in a recent history of Fortran I, II, and III, published in 1998 in the *IEEE Annals of the History of Computing*, the compiler "produced code of such efficiency that its output would startle the programmers who studied it."

1959–61: J.G.F. Francis of Ferranti Ltd., London, finds a stable method for computing eigenvalues, known as the **QR algorithm**.

Eigenvalues are arguably the most important numbers associated with matrices—and they can be the trickiest to compute. It's relatively easy to transform a square matrix into a matrix that's "almost" upper triangular, meaning one with a single extra set of nonzero entries just below the main diagonal. But chipping away those final nonzeros, without launching an avalanche of error, is nontrivial. The QR algorithm is just the ticket. Based on the QR decomposition, which writes A as the product of an orthogonal matrix Q and an upper triangular matrix R, this approach iteratively changes $A_i = QR$ into $A_{i+1} = RQ$, with a few bells and whistles for accelerating convergence to upper triangular form. By the mid-1960s, the QR algorithm had turned once-formidable eigenvalue problems into routine calculations.

1962: Tony Hoare of Elliott Brothers, Ltd., London, presents **Quicksort**.

Putting N things in numerical or alphabetical order is mind-numbingly mundane. The intellectual challenge lies in devising ways of doing so quickly. Hoare's algorithm uses the age-old recursive strategy of divide and conquer to solve the problem: Pick one element as a "pivot," separate the rest into piles of "big" and "small" elements (as compared with the pivot), and then repeat this procedure on each pile. Although it's possible to get stuck doing all $N(N-1)/2$ comparisons (especially if you use as your pivot the first item on a list that's already sorted!), Quicksort runs on average with $O(N \log N)$ efficiency. Its elegant simplicity has made Quicksort the pos-terchild of computational complexity.

James Cooley

John Tukey

1965: James Cooley of the IBM T.J. Watson Research Center and John Tukey of Princeton University and AT&T Bell Laboratories unveil the **fast Fourier transform**.

Easily the most far-reaching algo-rithm in applied mathematics, the FFT revolutionized signal processing. The underlying idea goes back to Gauss (who needed to calculate orbits of asteroids), but it was the Cooley–Tukey paper that made it clear how easily Fourier transforms can be computed. Like Quicksort, the FFT relies on a divide-and-conquer strategy to reduce an ostensibly $O(N^2)$ chore to an $O(N \log N)$ frolic. But unlike Quick- sort, the implementation is (at first sight) nonintuitive and less than straightforward. This in itself gave computer science an impetus to investigate the inherent complexity of computational problems and algorithms.

1977: Helaman Ferguson and Rodney Forcade of Brigham Young University advance an **integer relation detection algorithm**.

The problem is an old one: Given a bunch of real numbers, say $x_1, x_2, \ldots, x_n$, are there integers $a_1, a_2, \ldots, a_n$ (not all 0) for which $a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0$? For $n = 2$, the venerable Euclidean algorithm does the job, computing terms in the continued-fraction expansion of x_1/x_2. If x_1/x_2 is rational, the expansion terminates and, with proper unraveling, gives the "smallest" integers a_1 and a_2. If the Euclidean algorithm doesn't terminate—or if you simply get tired of computing it—then the unraveling procedure at least provides lower bounds on the size of the smallest integer relation. Ferguson and Forcade's generalization, although much more difficult to implement (and to understand), is also more powerful. Their detection algorithm, for example, has been used to find the precise coefficients of the polynomials satisfied by the third and fourth bifurcation points, $B_3 = 3.544090$ and $B_4 = 3.564407$, of the logistic map. (The latter polynomial is of degree 120; its largest coefficient is 257^{30}.) It has also proved useful in simplifying calculations with Feynman diagrams in quantum field theory.

1987: Leslie Greengard and Vladimir Rokhlin of Yale University invent the **fast multipole algorithm**.

This algorithm overcomes one of the biggest headaches of N-body simulations: the fact that accurate calculations of the motions of N particles interacting via gravitational or electrostatic forces (think stars in a galaxy, or atoms in a protein) would seem to require $O(N^2)$ computations—one for each pair of particles. The fast multipole algorithm gets by with $O(N)$ computations. It does so by using multipole expansions (net charge or mass, dipole moment, quadrupole, and so forth) to approximate the effects of a distant group of particles on a local group. A hierarchical decomposition of space is used to define ever-larger groups as distances increase. One of the distinct advantages of the fast multipole algorithm is that it comes equipped with rigorous error estimates, a feature that many methods lack.

What new insights and algorithms will the 21st century bring? The complete answer obviously won't be known for another hundred years. One thing seems certain, however. As Sullivan writes in the introduction to the top-10 list, "The new century is not going to be very restful for us, but it is not going to be dull either!"

Barry A. Cipra is a mathematician and writer based in Northfield, Minnesota.

Algorithms in Beam Physics

1970-2000

preconditioning
spectral methods
MATLAB
multigrid methods
IEEE arithmetic
nonsymmetric Krylov iterations
interior point methods
wavelets
fast multipole methods
automatic differentiation

… the dynamics of particle accelerators actually motivated the construction of the first symplectic integrators (Ruth 1983)

Combine:
Numerical integrators with automatic differentiation

hen
Oxford Computing Lab

Who invented the great numerical algorithms?

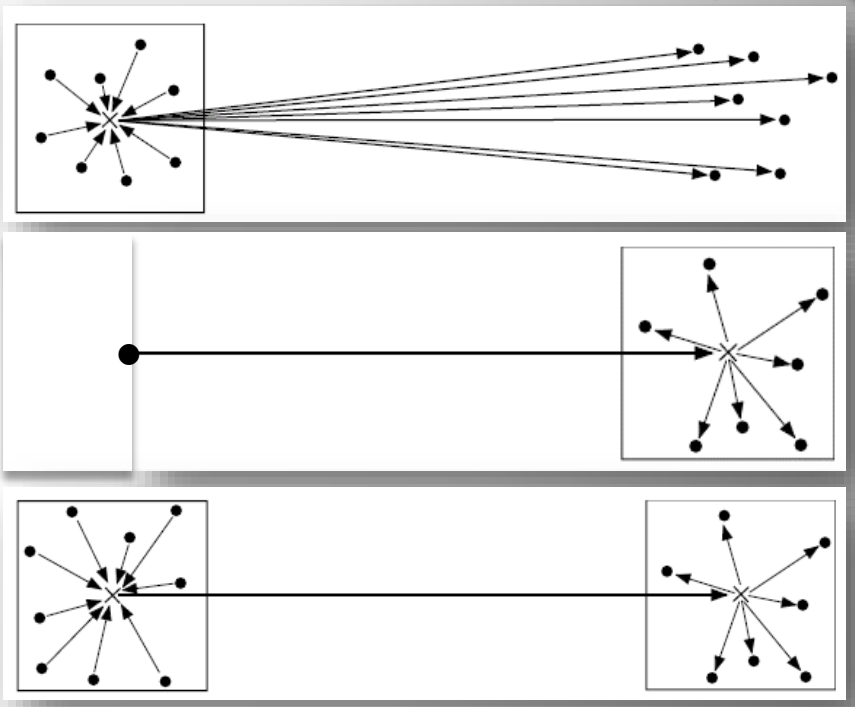
Combine:
fast multipole methods with automatic differentiation

Main Challenges of an N-Body Solver

- **Efficient Force Computation**
  - ✓ Adaptive hierarchical space decomposition
- **Accurate Time Stepper**
  - ✓ Variable high order, adaptive integrators with automatic steps size and order selection, and dense output
- **Ability to deal with very large N**
  - ✓ Distributed, high performance computing on hybrid architecture supercomputers
- **Ability to deal with long time-scale dynamics**
  - ✗ Time does not parallelize

Force Computation for N-Body

- **"Exact"** (scales as $O(N^2)$):
  - *Particle-particle*
- **"Fast"** methods (scale as $< O(N^2)$):
  - *Basis-function method*
    - Orthogonal polynomials
  - *Grid-based methods:*
    - PIC, particle-mesh, FFT-based
    - Multi-grid
  - *Hierarchical space decomposition:*
    - Tree: cell-particle
    - Cluster: particle-cell
    - Fast multipole: cell-cell



$$\vec{f}_i = \nabla \sum_{i=1}^{N} \mathbf{K}(\vec{r}_i, \vec{r}_j) q_j = \nabla \sum_{i=1}^{N} \mathbf{K}(\vec{r}_i - \vec{r}_j) q_j = \nabla \sum_{i=1}^{N} \mathbf{K}_{ij} q_j \qquad i = \overline{1, N}.$$

The Fast Multipole Method

Rokhlin and Greengard arguably provided the first numerically defensible method for reducing the N-body problem's computational complexity.

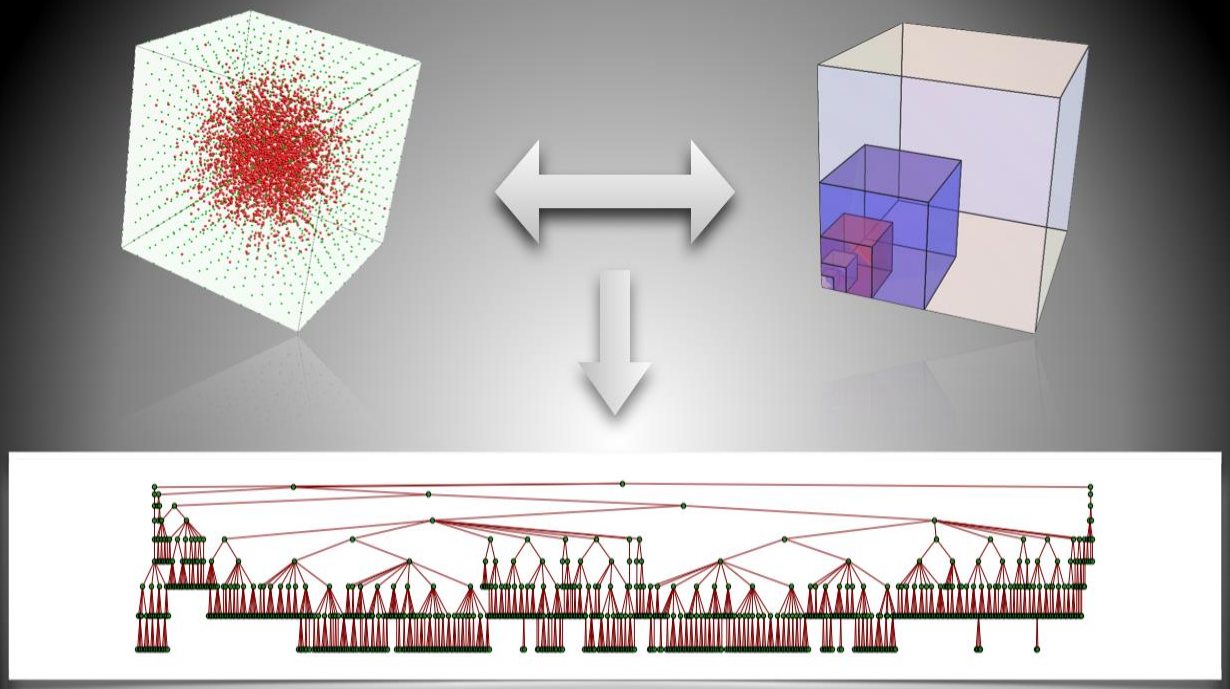
to $\mathcal{O}(N)$

Multipole methods and their descendants will be ubiquitous. – L.N. Trefethen (Oxford) Predictions for Scientific Computing Fifty Years From Now (Mathematics Today, 2000)

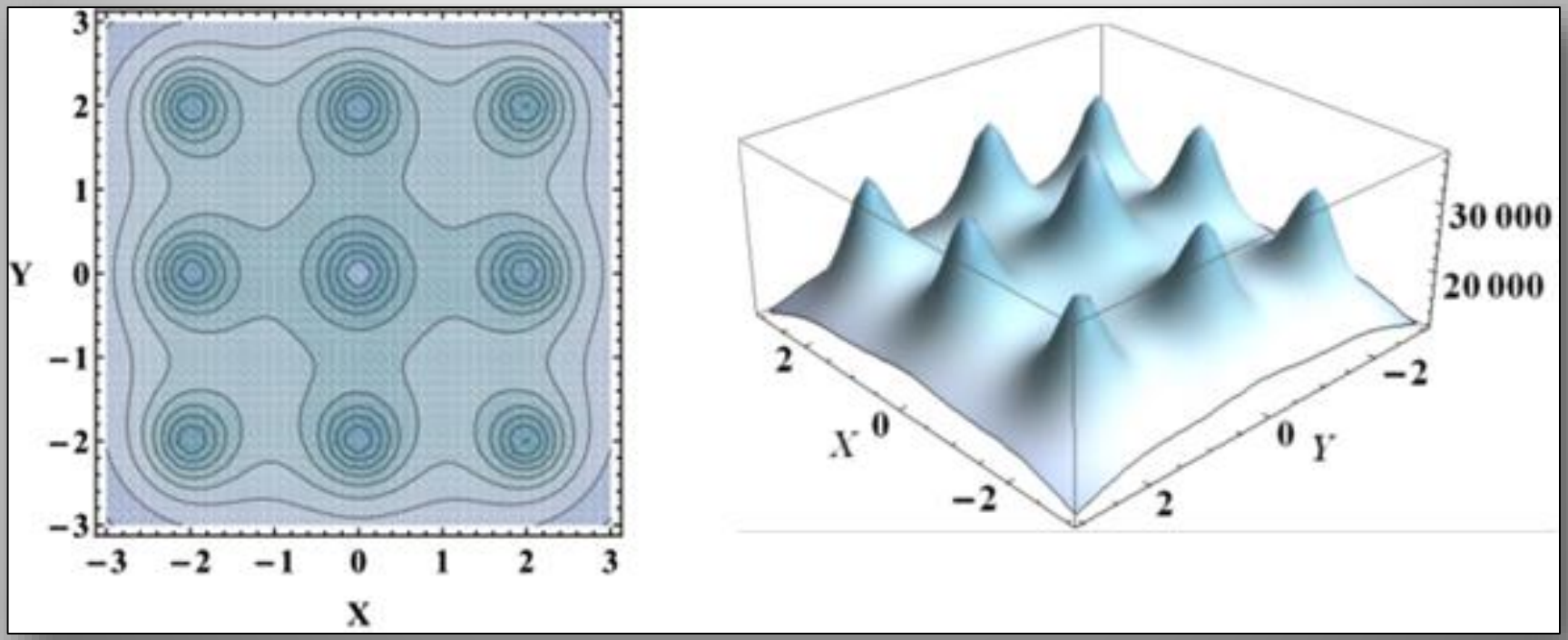
COMPUTING IN SCIENCE & ENGINEERING
JANUARY/FEBRUARY 2000

THE FAST MULTIPOLE ALGORITHM

A Picture Is Worth a Thousand Words



Multiple Gaussians in Linear Time and Memory



Time-Stepping Algorithms

- Low vs. high order
- Constant step size vs. adaptive
- Single step vs. multi-step
- Explicit vs. implicit
- Symplectic vs. non-symplectic
- Reversible vs. non-reversible
- Single level vs. hierarchical
- Serial vs. parallel

Time Stepping Best Practices

3.3 Block-step procedure

Let us examine the sequential procedure for one block-step.

(i) Obtain the next time for integration,

$$t_{\rm next} = \min_{1 \leqslant i \leqslant N} (t_{{\rm I},i} + \Delta t_{{\rm I},i}), \tag{4}$$

where $t_{{\rm I},i}$ and $\Delta t_{{\rm I},i}$ are the time of the last irregular force calculation and irregular time-step of particle i.

(ii) Make the active particle list for regular and irregular force calculation,

$$\mathbb{L}_{\rm act,R} = \{i \mid t_{{\rm R},i} + \Delta t_{{\rm R},i} = t_{\rm next}\}, \tag{5}$$

$$\mathbb{L}_{\rm act,I} = \{i \mid t_{{\rm I},i} + \Delta t_{{\rm I},i} = t_{\rm next}\}, \tag{6}$$

where the subscripts R and I denote regular and irregular terms. Here, $\{i \mid cond.\}$ defines a set of i such that it satisfies *cond.*

(iii) Predict all particles needed for force evaluation.

(iv) Calculate the irregular force and its time derivative for particle $i \in \mathbb{L}_{\rm act,I}$,

$$\mathbf{F}_{{\rm I},i} = \sum_{j \in \mathbb{L}_i} m_j \frac{\mathbf{R}_{ij}}{|\mathbf{R}_{ij}|^3}, \tag{7}$$

$$\dot{\mathbf{F}}_{{\rm I},i} = \sum_{j \in \mathbb{L}_i} m_j \left[\frac{\dot{\mathbf{R}}_{ij}}{|\mathbf{R}_{ij}|^3} - 3 \frac{(\mathbf{R}_{ij} \cdot \dot{\mathbf{R}}_{ij}) \mathbf{R}_{ij}}{|\mathbf{R}_{ij}|^5} \right]. \tag{8}$$

(v) Apply the corrector for the active irregular particles and decide the next time-step $\Delta t_{{\rm I},i}$.

(vi) Accumulate the regular force and its time derivative for each active regular particle $i \in \mathbb{L}_{\rm act,R}$ and construct the neighbour list,

$$\mathbf{F}_{{\rm R},i} = \sum_{j \neq i}^{N} \begin{cases} m_j \dfrac{\mathbf{R}_{ij}}{|\mathbf{R}_{ij}|^3} & (R_{ij,\min} > h_i) \\ 0 & (\text{otherwise}) \end{cases}, \tag{9}$$

$$\dot{\mathbf{F}}_{{\rm R},i} = \sum_{j \neq i}^{N} \begin{cases} m_j \left[\dfrac{\dot{\mathbf{R}}_{ij}}{|\mathbf{R}_{ij}|^3} - 3 \dfrac{(\mathbf{R}_{ij} \cdot \dot{\mathbf{R}}_{ij}) \mathbf{R}_{ij}}{|\mathbf{R}_{ij}|^5} \right] & (R_{ij,\min} > h_i) \\ 0 & (\text{otherwise}) \end{cases}, \tag{10}$$

$$\mathbb{L}_i = \{j \mid j \neq i, R_{ij,\min} < h_i\}. \tag{11}$$

(vii) Execute the regular corrector. Since the neighbour list $\mathbb{L}_i$ has been updated, the force polynomials should be corrected to reflect the difference between the old and new list.

Accelerating NBODY6 with Graphics Processing Units

Keigo Nitadori[1⋆] and Sverre J. Aarseth[2⋆]
[1] *Center for Computational Science, University of Tsukuba, 1-1-1, Tennodai, Tsukuba, Ibaraki 305-8577, Japan*
[2] *Institute of Astronomy, University of Cambridge, Madingley Road, Cambridge, CB3 0HA, UK*

Adaptive, Variable Order Integration

Proposition . *Assume that the function $h \mapsto x(t_m + h)$ is analytic on a disk of radius ρ_m, and that there exists a positive constant M_m such that*

$$|x_m^{[j]}| \approx \frac{M_m}{\rho_m^j}, \qquad \forall\, j \in \mathbb{N}.$$

Then, if the required accuracy ε tends to 0, the optimal value of h that minimizes the number of operations tends to

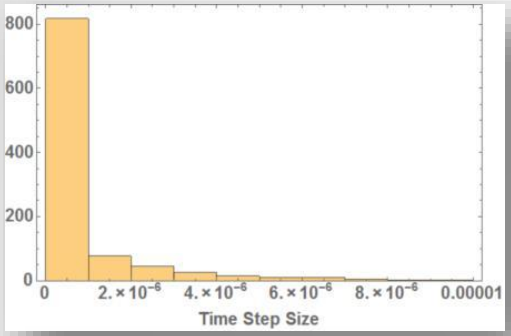
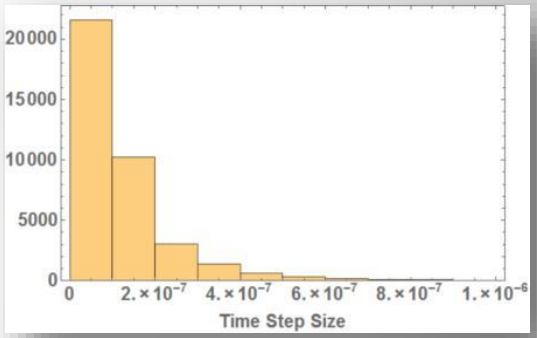
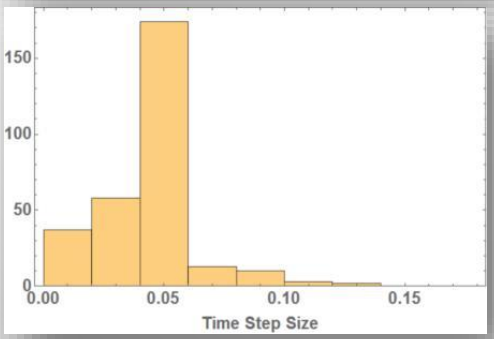
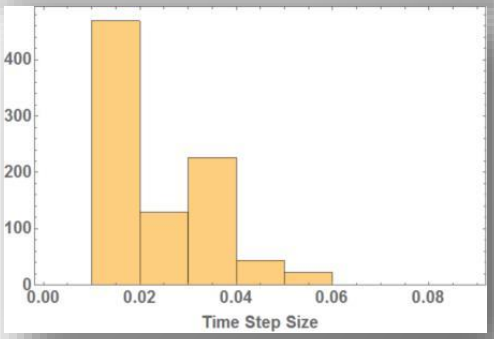
$$h_m = \frac{\rho_m}{e^2},$$

and the optimal order p_m behaves like

$$p_m = -\frac{1}{2} \ln\left(\frac{\varepsilon}{M_m}\right) - 1.$$

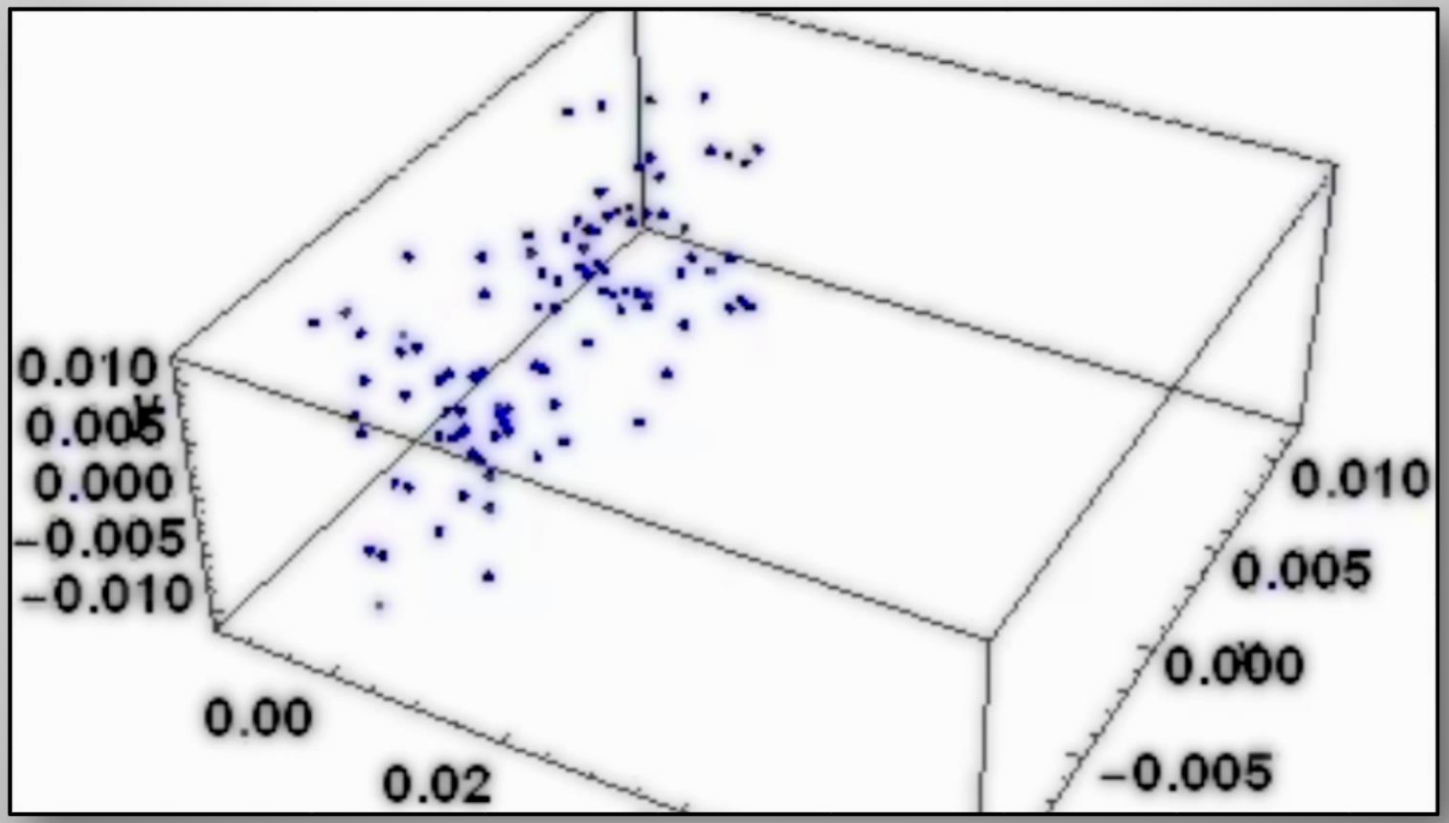
Simo, 2001

Examples

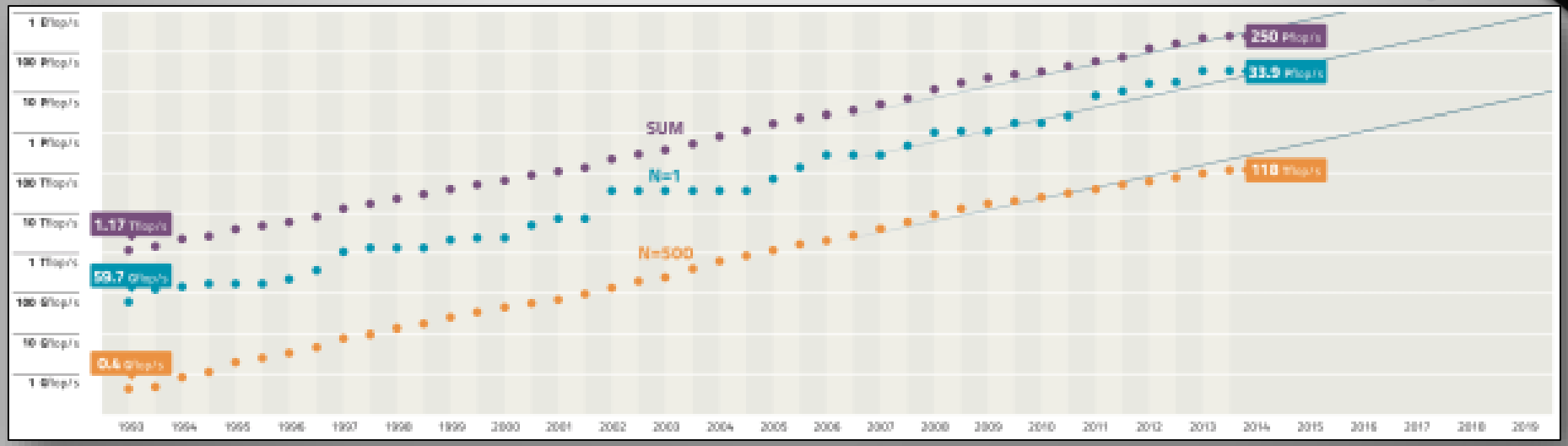
Energy	Time bins of equal time widths	Time bins of equal number of particles
7 TeV		
1 MeV		

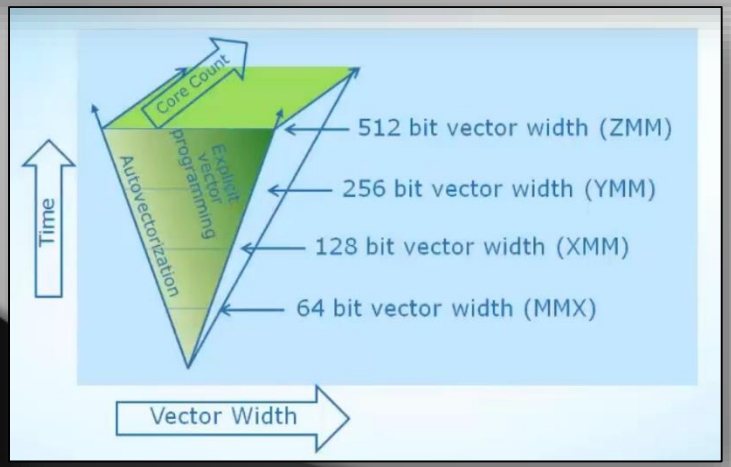
For accuracies of 10^{-7} and 10^{-16}, time step sizes used in each step are the same, while orders vary between 2 and 7.

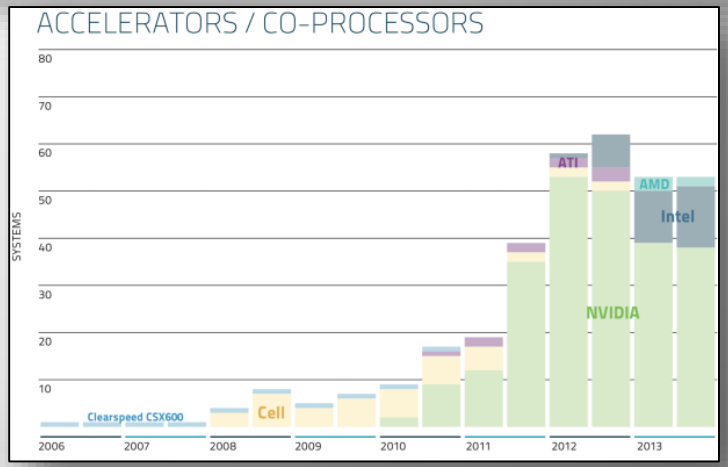
Simo Integrator



Hardware Trends







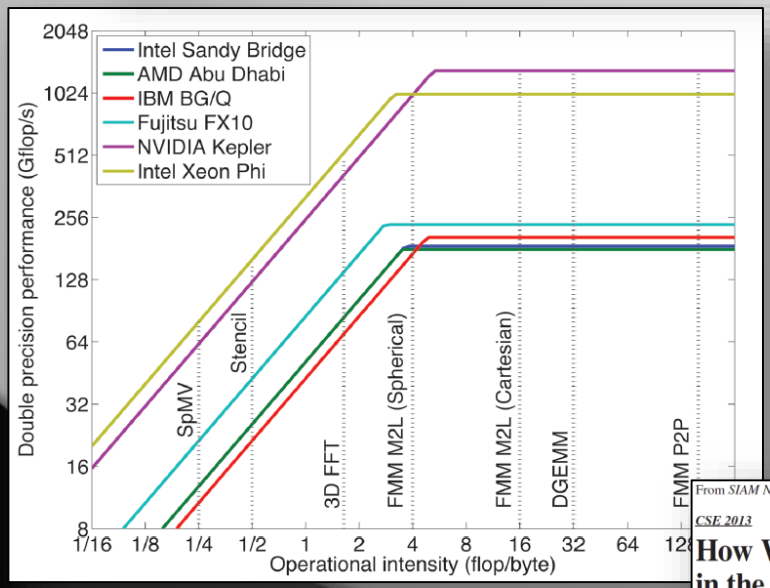
“Best” Algorithms

Comparison of scalable fast methods for long-range interactions

Phys. Rev. E **88**, 063308 – Published 19 December 2013

Axel Arnold, Florian Fahrenberger, Christian Holm, Olaf Lenz, Matthias Bolten, Holger Dachsel, Rene Halver, Ivo Kabadshow, Franz Gähler, Frederik Heber, Julian Iseringhausen, Michael Hofmann, Michael Pippig, Daniel Potts, and Godehard Sutmann

Our findings suggest that, depending on system size and desired accuracy, the FMM- and FFT-based methods are most efficient in performance and stability.

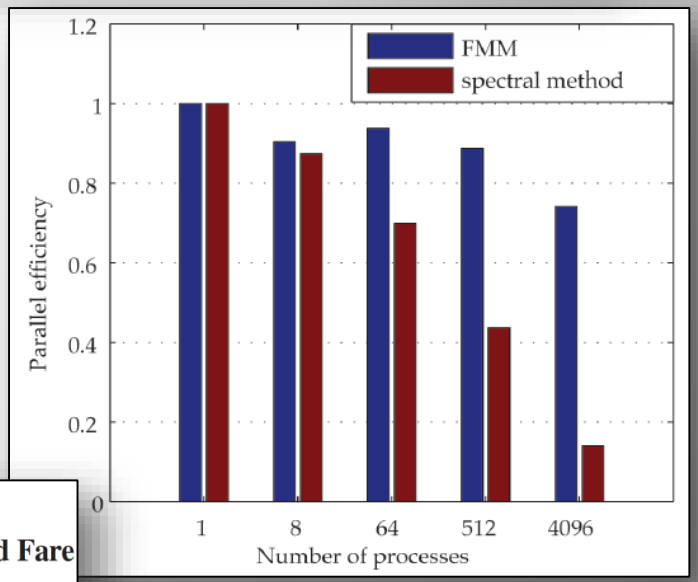



From *SIAM News*, Volume 46, Number 6, July/August 2013

CSE 2013

How Will the Fast Multipole Method Fare in the Exascale Era?

By Lorena A. Barba and Rio Yokota



FMM World Record

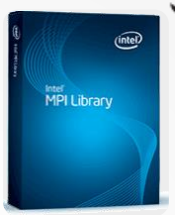
3,011,561,968,121 particles

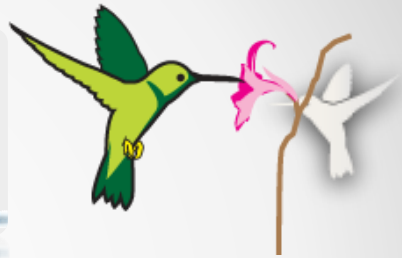
Method	Number of Cores: 1 (PC)	Number of Cores: 294,912 (JUGENE)
FMM	220 days	11 minutes
Direct	1 billion years	32,000 years

Credit: Jülich Supercomputing Centre (JSC)

Particles' High-order Adaptive Dynamics (PHAD)

We are developing a parallel code (PHAD) based on these new methods that will be the first one capable of particle-based simulations of electron cooling and other difficult beam dynamics phenomena with high fidelity, efficiently.





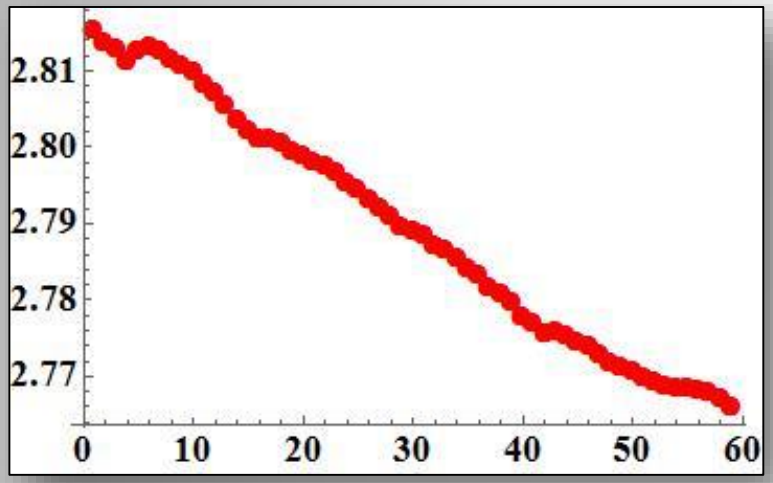
OpenCL

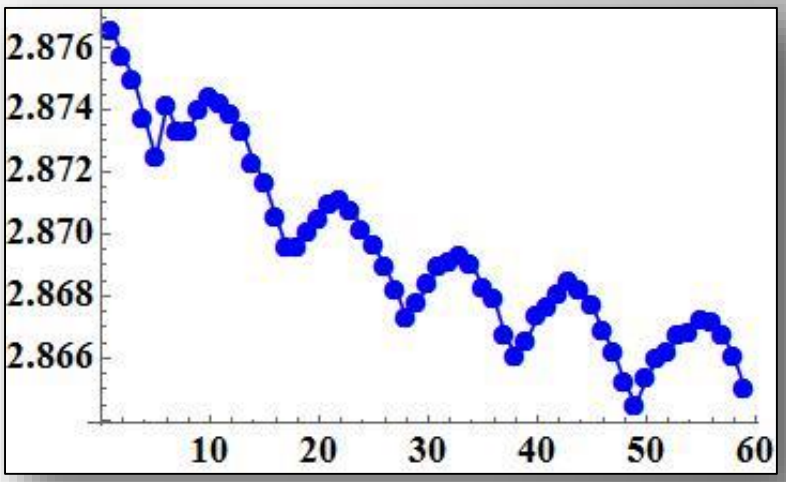
NORTHERN ILLINOIS UNIVERSITY
Center for Research Computing & Data
Division of Research and Innovation Partnerships

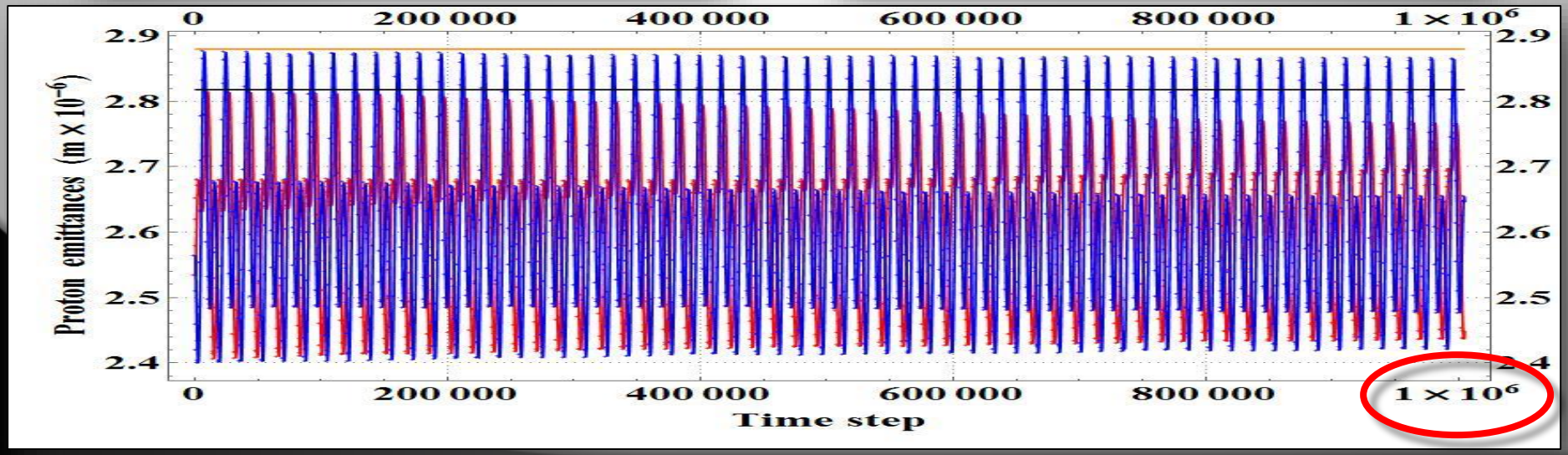
NORTHERN ILLINOIS UNIVERSITY
Beam Physics Code Repository
N. Illinois Center for Accelerator & Detector Development

http://niu.edu/beamphysicscode/

First Particle-Based Cooling Simulation







Summary and Conclusions

- **Computational beam physics** plays an important part in modeling and simulating electron cooling; designing, operating, and improving current and future particle accelerators and their performance
- **Algorithmic and hardware improvements multiply**, making high fidelity large-scale problems feasible
- **Fundamental algorithms** and methods are general enough to be adaptable/applicable to many other beam dynamics problems and different scientific fields:
- Current and next generation **high-performance computing** systems are well matched to these algorithms
- Entering a new phase of **high fidelity** *electron cooling simulations*