Institute for Advanced Study

The collisional N-body code REBOUND

Hanno Rein @ NAOJ, Tokyo, March 2012

REBOUND

A new open source collisional N-body code

Numerical Integrators

- We want to integrate the equations of motions of a particle

$$\dot{x} = v$$

$$\dot{v} = a(x, v)$$

- For example, gravitational potential

$$a(x) = -\nabla\Phi(x)$$

- In physics, these can usually be derived from a Hamiltonian

$$H = \frac{1}{2}p^2 + \Phi(x)$$

- Symmetries of the Hamiltonian correspond to conserved quantities

Numerical Integrators

- Discretization

$$\dot{x} = v \qquad \longrightarrow \qquad \Delta x = v\,\Delta t$$
$$\dot{v} = a(x, v) \qquad\qquad \Delta v = a(x, v)\,\Delta t$$

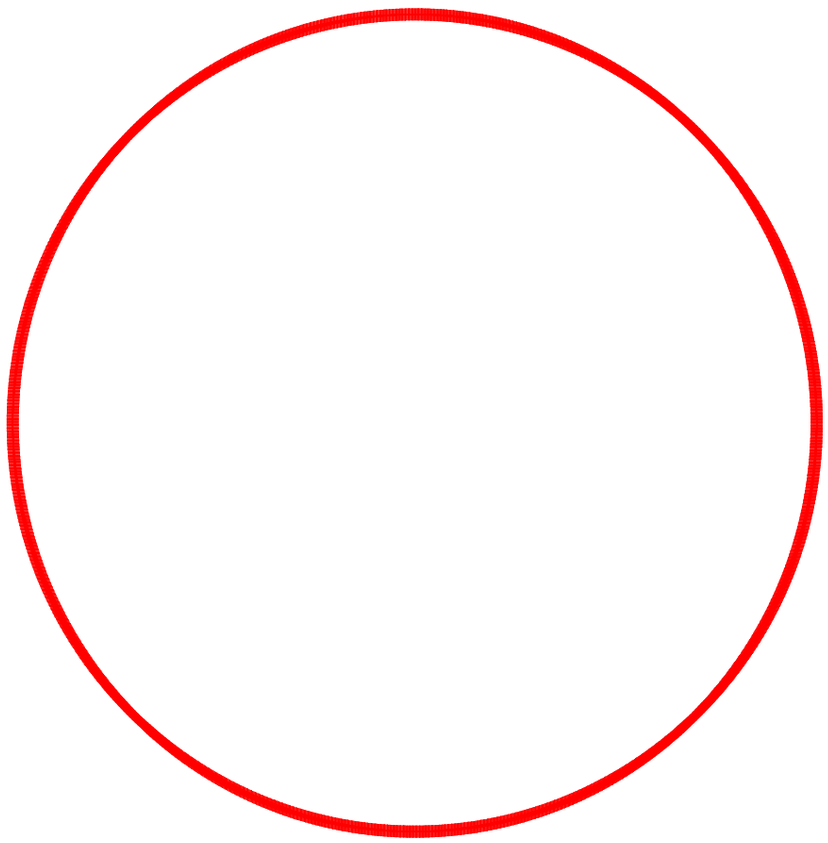
- Hamiltonian

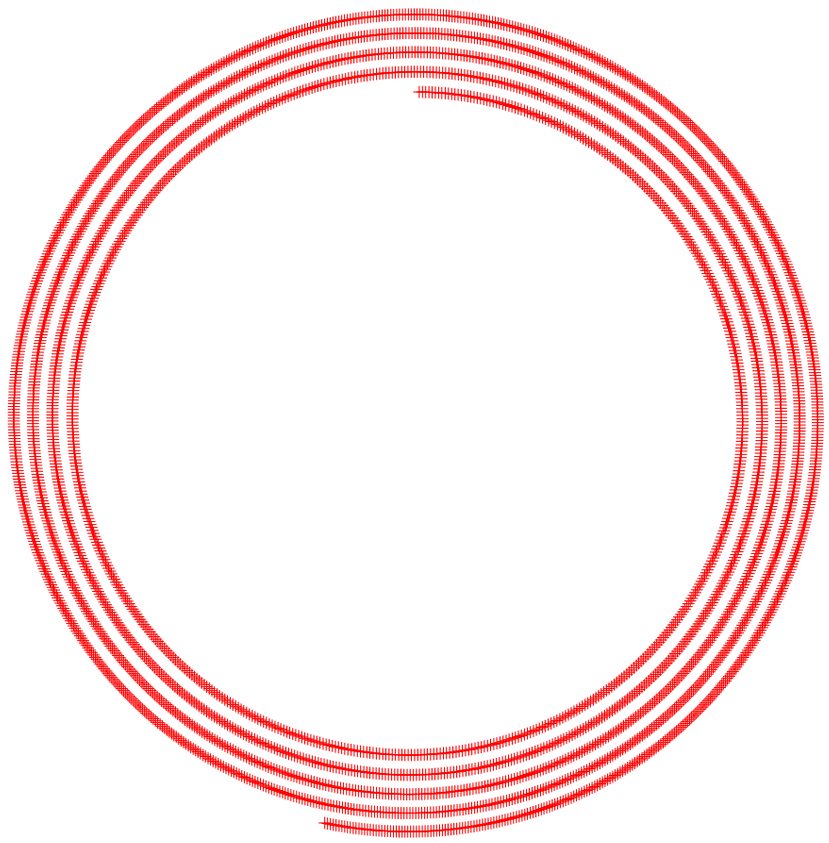
$$H = \frac{1}{2}p^2 + \Phi(x) \qquad \longrightarrow \qquad ?$$

- The system is governed by a 'discretized Hamiltonian', if and only if the integration scheme is symplectic.

- Why does it matter?

Symplectic vs non symplectic integrators





Mixed variable integrators

- So far: symplectic integrators are great.
- How can it be even better?
- We can split the Hamiltonian:

$$H = H_0 + \epsilon\, H_{\rm pert}$$

Integrate particle exactly with dominant Hamiltonian

Integrate particle exactly under perturbation Hamiltonian

- Switch back and forth between different Hamiltonians
- Often uses different variables for different parts
- Then:

$$\text{Error} = \epsilon\,(\Delta t)^{p+1}\ [H_0, H_{\rm pert}]$$

Example: Leap-Frog

$$H = \underbrace{\frac{1}{2}p^2}_{\text{Drift}} + \underbrace{\Phi(x)}_{\text{Kick}}$$

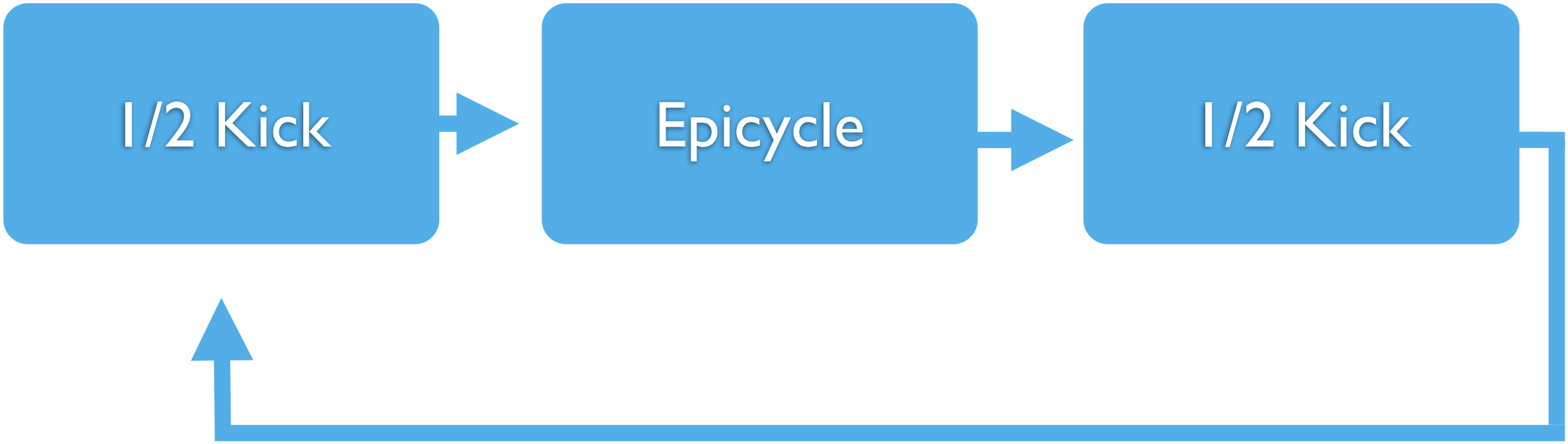
Example: SWIFT/MERCURY

$$H = \frac{1}{2}p^2 + \Phi_{\mathrm{Kepler}}(x) + \Phi_{\mathrm{Other}}(x)$$

Kepler | Kick

Example: Symplectic Epicycle Integrator

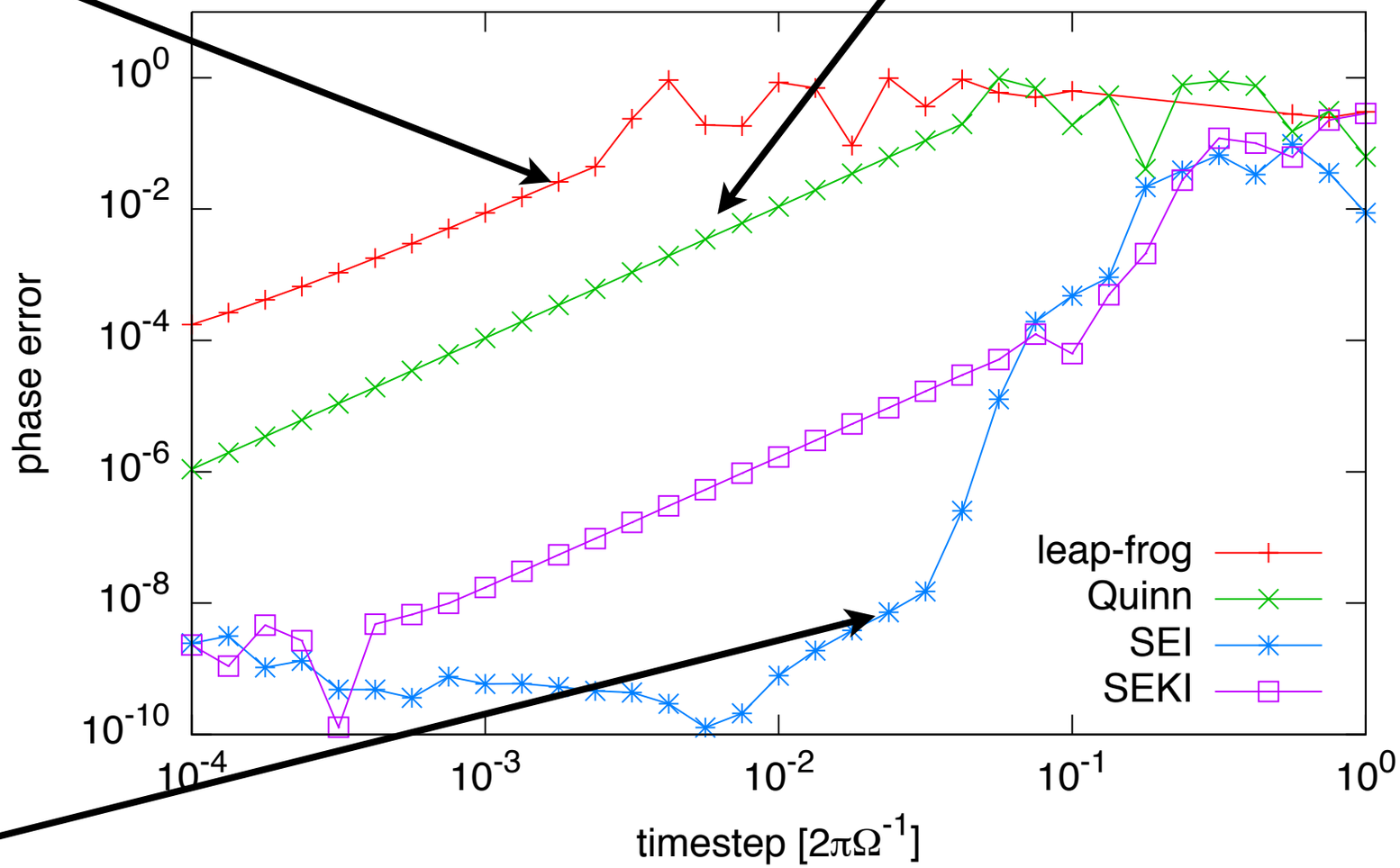
$$H = \underbrace{\frac{1}{2}p^2 + \Omega(p \times r)e_z + \frac{1}{2}\Omega^2\left[r^2 - 3(r \cdot e_x)^2\right]}_{\text{Epicycle}} + \underbrace{\Phi(r)}_{\text{Kick}}$$



10 Orders of magnitude better!

non-symplectic

symplectic



mixed variable, symplectic

Take home message V

symplectic integrators

=

awesome

REBOUND

- Multi-purpose N-body code
- Optimized for collisional dynamics
- Code description paper recently accepted by A&A
- Written in C, open source
- Freely available at http://github.com/hannorein/rebound

Astronomy & Astrophysics manuscript no. paper
October 20, 2011

© ESO 2011

REBOUND: An open-source multi-purpose N-body code for collisional dynamics

Hanno Rein[1] and Shang-Fei Liu[2,3]

[1] Institute for Advanced Study, 1 Einstein Drive, Princeton, NJ 08540
e-mail: rein@ias.edu
[2] Kavli Institute for Astronomy and Astrophysics, Peking University, Beijing 100871, P. R. China
[3] Department of Astronomy, Peking University, Beijing 100871, P. R. China
e-mail: liushangfei@pku.edu.cn

Submitted: 13 September 2011

ABSTRACT

REBOUND is a new multi-purpose N-body code which is freely available under an open-source license. It was designed for collisional dynamics such as planetary rings but can also solve the classical N-body problem. It is highly modular and can be customized easily to work on a wide variety of different problems in astrophysics and beyond.
REBOUND comes with three symplectic integrators: leap-frog, the symplectic epicycle integrator (SEI) and a Wisdom-Holman mapping (WH). It supports open, periodic and shearing-sheet boundary conditions. REBOUND can use a Barnes-Hut tree to calculate both self-gravity and collisions. These modules are fully parallelized with MPI as well as OpenMP. The former makes use of a static domain decomposition and a distributed essential tree. Two new collision detection modules based on a plane-sweep algorithm are also implemented. The performance of the plane-sweep algorithm is superior to a tree code for simulations in which one dimension is much longer than the other two and in simulations which are quasi-two dimensional with less than one million particles.
In this work, we discuss the different algorithms implemented in REBOUND, the philosophy behind the code's structure as well as implementation specific details of the different modules. We present results of accuracy and scaling tests which show that the code can run efficiently on both desktop machines and large computing clusters.

Key words. Methods: numerical – Planets and satellites: rings – Proto-planetary disks

1. Introduction

REBOUND is a new open-source collisional N-body code. This code, and precursors of it, have already been used in wide variety of publications (Rein & Papaloizou 2010; Crida et al. 2010; Rein et al. 2010, Rein & Liu in preparation; Rein & Latter in preparation). We believe that REBOUND can be of great use for many different problems and have a wide reach in astrophysics and other disciplines. To our knowledge, there is currently no publicly available code for collisional dynamics capable of solving the problems described in this paper. This is why we decided to make it freely available under the open-source license GPLv3[1].

Collisional N-body simulations are extensively used in astrophysics. A classical application is a planetary ring (see e.g. Wisdom & Tremaine 1988; Salo 1991; Richardson 1994; Lewis & Stewart 2009; Rein & Papaloizou 2010; Michikoshi & Kokubo 2011, and references therein) which have often a collision time-scale that is much shorter than or at least comparable to an orbital time-scale. Self-gravity plays an important role, especially in the dense parts of Saturn's rings (Schmidt et al. 2009). These simulations are usually done in the shearing sheet approximation (Hill 1878).

Collisions are also important during planetesimal formation (Johansen et al. 2007; Rein et al. 2010, Johansen et al. in preparation). Collisions provide the dissipative mechanism to form a planetesimal out of a gravitationally bound swarm of boulders.

REBOUND can also be used with little modification in situations where only a statistical measure of the collision frequency is required such as in transitional and debris discs. In such systems, individual collisions between particles are not modeled exactly, but approximated by the use of super-particles (Star Kuchner 2009; Lithwick & Chiang 2007).

Furthermore, REBOUND can be used to simulate classical body problems involving entirely collision-less systems. A plectic and mixed variable integrator can be used to follow trajectories of both test-particles and massive particles.

We describe the general structure of the code, how to obtain, compile and run it in Sect. 2. The time-stepping and our implementation of symplectic integrators are described in Sect. 3. The modules for gravity are described in Sect. 4 and algorithms for collision detection are discussed in Sect. 5. In Sect. 6, we present results of accuracy tests for various modules. We discuss the efficiency of the parallelization and results of scaling tests in Sect. 7. We finally summarize

2. Overview of the code structure

REBOUND is written entirely in C and conforms to the C99 standard. It compiles and runs on any modern UNIX system which supports the POSIX standard such as Linux distributions, Mac OSX. In its simplest form, REBOUND has no external libraries to compile.

Users are encouraged to install the GNU Scientific Library and libraries which enable real-time and interactive visualization. LIBPNG is required to automatically

[1] The full license is distributed together with REBOUND. It can also be downloaded from http://www.gnu.org/licenses/gpl.html.

REBOUND modules

Geometry
- Open boundary conditions
- Periodic boundary conditions
- Shearing sheet / Hill's approximation

Integrators
- Leap frog
- Symplectic Epicycle integrator (SEI)
- Wisdom-Holman mapping (WH)

Gravity
- Direct summation, $O(N^2)$
- BH-Tree code, O(N log(N))
- FFT method, O(N log(N))

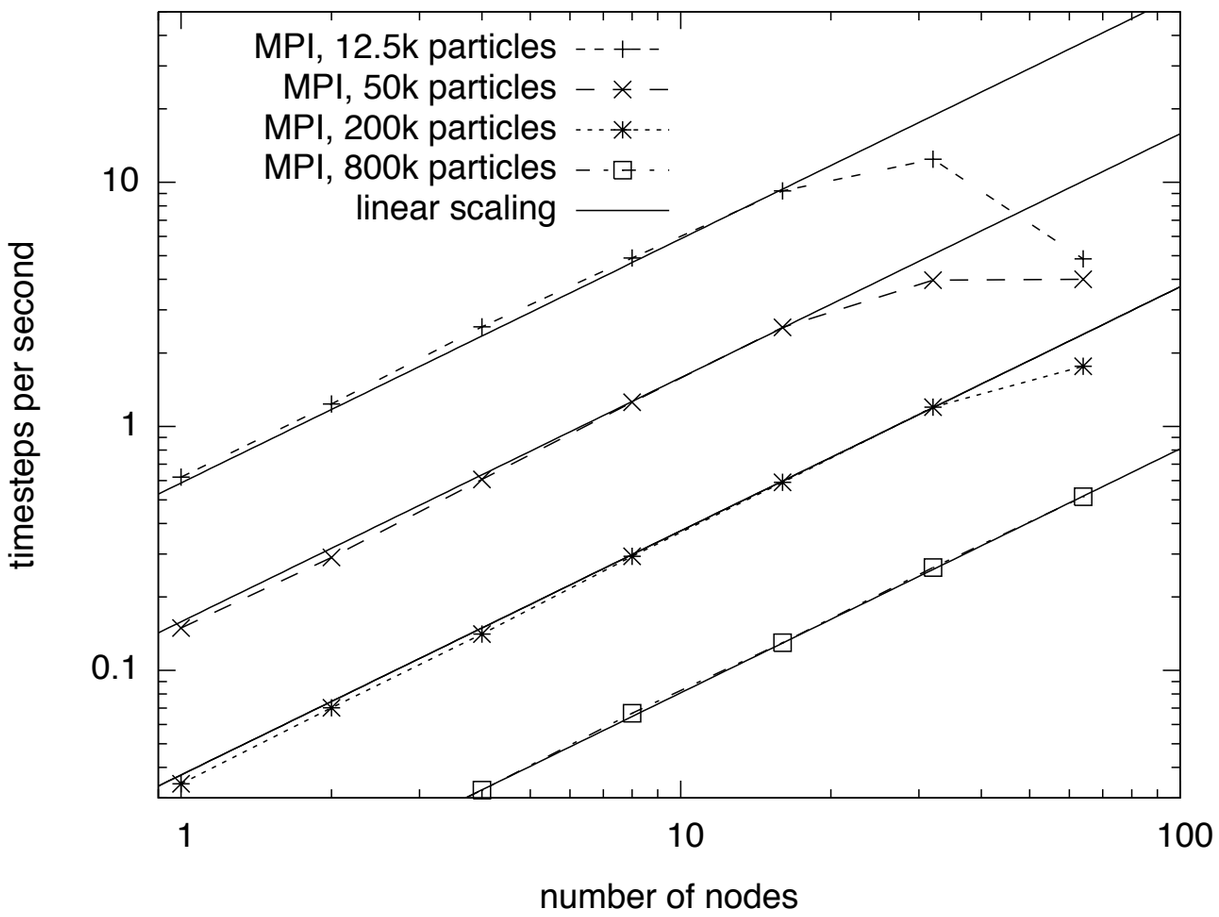
Collision detection
- Direct nearest neighbor search, $O(N^2)$
- BH-Tree code, O(N log(N))
- Plane sweep algorithm, O(N) or $O(N^2)$

Plane-sweep

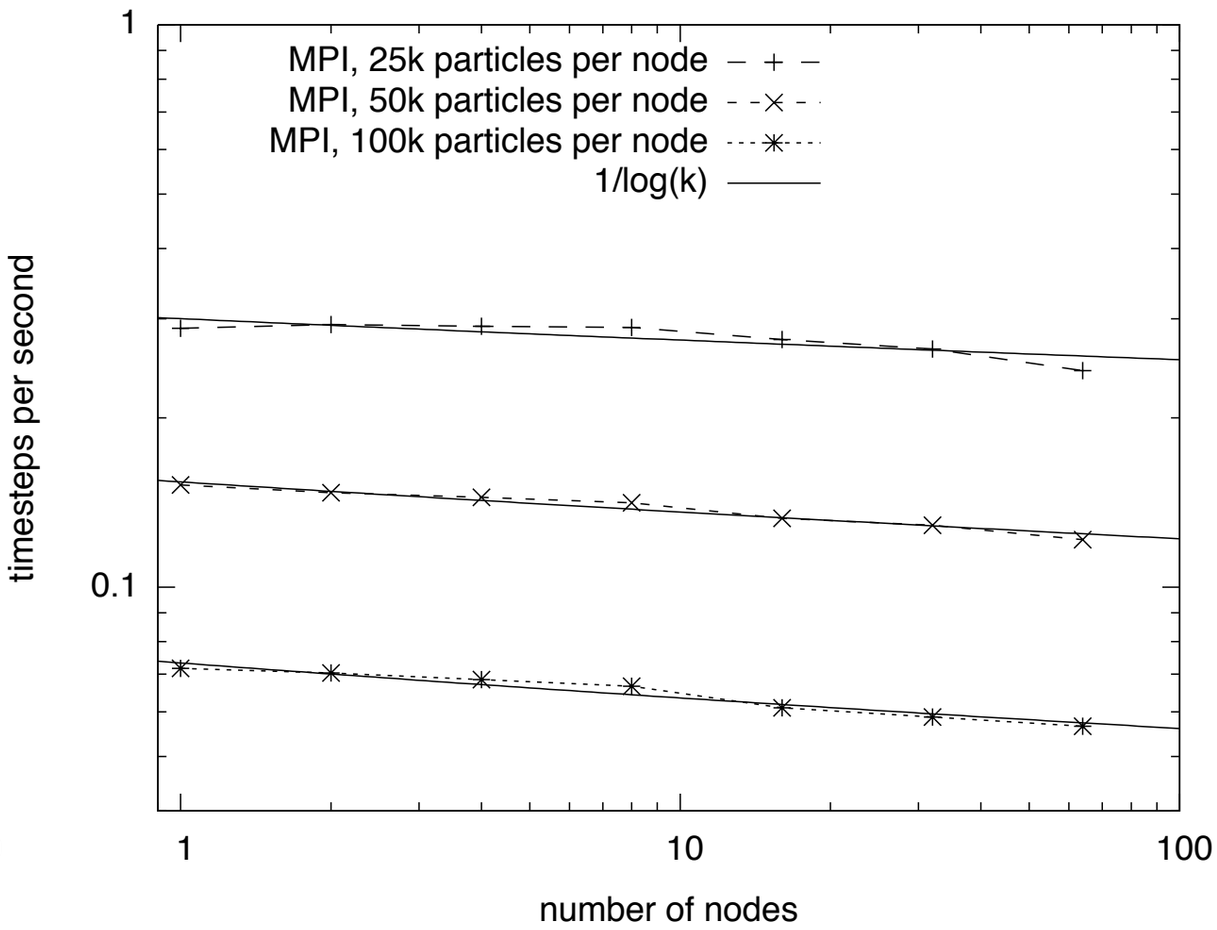
Blackboard

REBOUND scalings using a tree

strong



weak



DEMO

Take home message VII

Download REBOUND

Conclusions

Conclusions

REBOUND

N-body code, optimized for collisional dynamics, uses symplectic integrators

Open source, freely available, very modular and easy to use

http://github.com/hannorein/rebound