falcON

a Cartesian FMM for the low-accuracy regime

College Park, 28th April 2004


Walter Dehnen (Leicester)
walter.dehnen@astro.le.ac.uk
N-body simulations in astronomy

HCG87: a group of galaxies

ωCen: a globular cluster
properties of stellar systems

- simple physics: Newtonian gravity
- very inhomogeneous
  ⇒ large dynamic range
- dynamically young \( t_{\text{dyn}} \approx \text{Myr–Gyr} \)
- well approximated as ensembles of point masses
  ⇒ well described as Hamiltonian systems
  (⇒ need symplectic time integration)

\[
H = \sum_{i=1}^{N} \frac{m_i}{2} \left[ v_i^2 - \sum_{j \neq i} \frac{G m_j}{|x_i - x_j|} \right], \quad v_i = \dot{x}_i = \frac{p_i}{m_i}
\]

with \( N \approx 10^{5−20} \)

- equation of motion in continuum (mean-field) limit:

\[
0 = \frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \cdot v - \frac{\partial f}{\partial v} \cdot \frac{\partial \Phi}{\partial x}
\]

collisionless Boltzmann equation (CBE)

- \( f(x, v, t) \): distribution function (density in phase space)
- \( \Phi(x) \): mean-field gravitational potential

both are related via the Poisson equation:

\[
\nabla^2 \Phi(x) = 4\pi G \int d^3v \ f(x, v, t)
\]
two-body relaxation

How good is the continuum description?

▶ stellar encounters deflect trajectories
  ⇒ stellar orbits get randomized
  ⇒ Maxwellian velocity distribution

▶ two-body relaxation time:

\[ t_{\text{relax}} \approx 0.1 \frac{N}{\log N} t_{\text{dyn}} \]

1 collision-dominated stellar dynamics

▶ \( t_{\text{relax}} \ll \) age of system

⇒ continuum limit not applicable
⇒ must simulate Hamiltonian directly:
  ▶ force computation is \( \mathcal{O}(N^2) \)
    ⇒ computational effort limits \( N \ll 10^5 \)
  ▶ close encounters are important
    ⇒ time integration becomes tedious

2 collisionless stellar dynamics

▶ \( t_{\text{relax}} \gg \) age of system

⇒ continuum limit applicable
⇒ solve CBE & Poisson equation
‘collisionless’ \( N \)-body simulations

How to solve the CBE?

\[
0 = \frac{df}{dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \cdot v - \frac{\partial f}{\partial v} \cdot \frac{\partial \Phi}{\partial x}
\]

▷ \( f \) is 6D & very inhomogeneous
  ⇒ (Eulerian) grid methods are useless
  ⇒ Lagrangian method (‘method of characteristics’):

▷ sample \( N \) trajectories \( \{ \mu_i, x_i, v_i \} \) from \( f(x, v, t = 0) \)
▷ solve equations of motion \( \ddot{x}_i = -\nabla \Phi(x_i, t) \)
▷ CBE: \( \mu_i = \text{const} \) along trajectories
  ⇒ \( f(x, v, t) \) is represented by \( \{ \mu_i, x_i(t), v_i(t) \} \)
  ⇒ \( f \) is unknown
  ⇒ \textbf{moments} of \( f \) can be estimated
  ⇒ \( N \ll \mathcal{N} \) is \textit{numerical parameter}
  ⇒ artificial two-body relaxation
How to solve the Poisson equation?

\[ \nabla^2 \Phi(x) = 4\pi G \int d^3v f(x, v, t) \]

1 grid techniques (FFT, multigrid):
  ▶ fast: \( \mathcal{O}(n_{\text{grid}} \log n_{\text{grid}}) \)
  ▶ periodic (⇒ cosmology)
  ▶ problem: inhomogeneity (but: adaptive multigrid)

2 basic functions (using \( Y_{lm} \)):
  ▶ fast: \( \mathcal{O}(N n_{\text{basis}}) \)
  ▶ problems: central singularity, spherical symmetry

3 Greens-function approach:

\[ \Phi(x, t) = -G \int d^3x' d^3v \frac{f(x', v, t)}{|x - x'|} \]

▶ general & adaptive
▶ problem: \( f \) is unknown
⇒ estimate (\( \epsilon \): softening length)

\[ \Phi(x_i, t) \approx -\sum_{i \neq j} \frac{G \mu_j}{\sqrt{[x_i - x_j(t)]^2 + \epsilon^2}} \]

**force softening** to
▶ optimize force estimate (since \( f \) is unknown)
▶ suppress (unphysically) close encounters
⇒ force-**estimation error** (unavoidable)
true gravity of Hernquist model
estimation error with $N = 10^6$
computing the forces

- Greens-function approach → Hamiltonian:
  \[ H = \sum_{i=1}^{N} \frac{\mu_i}{2} \left( v_i^2 - \sum_{j \neq i} \frac{G \mu_j}{\sqrt{|x_i - x_j|^2 + \epsilon^2}} \right) \]

- how to evaluate \( \Phi \) & \( \nabla \Phi \)?
- can tolerate \textbf{approximation error} \( \ll \) \textbf{estimation error}
  \( \Rightarrow \) use \textbf{approximative} methods

1. direct summation (not approximative):
   - slow: \( \mathcal{O}(N^2) \) (but: GRAPE)
   - (unnecessarily) accurate
   - used in \textit{collisional N-body} codes

2. Barnes & Hut (1986) \textbf{tree code}:
   - use hierarchical tree (usually: oct-tree) \( \Rightarrow \) fully adaptive
   - fast(er): \( \mathcal{O}(N \log N) \)
   - most common method in astrophysics
   - violates Newton’s 3rd law
     \( \Rightarrow \) total momentum not conserved

3. traditional \textbf{fast multipole method (FMM)}:
   - use hierarchy of cartesian grids \( \Rightarrow \) not fully adaptive
   - compute gravity via spherical multipoles & complex \( Y_{lm} \)
     \( \Rightarrow \) numerics complicated & cumbersome
   - formally \( \mathcal{O}(N) \), \textbf{but}
     slower than \textbf{tree code} (for astrophysical applications, see Capuzzo-Colcetta & Miochi, 1998, JCP, \textbf{143}, 29)
approximation error with $N = 10^6$
details of the tree code

1 preparation phase
1.1 build a hierarchical tree of cubic cells
   ▶ cost: $\mathcal{O}(N \log N)$
1.2 pre-compute multipole moments etc

2 force computation: ‘tree-walk’
   ▶ for each body: compute force due to root cell
   ▶ to compute force from cell:
     if body is well-separated from cell:
       compute force from multipole moments
     otherwise
       sum forces from daughter cells (recursive)
   ▶ cost: $\mathcal{O}(\log N)$ per body $\Rightarrow \mathcal{O}(N \log N)$

▶ the tree code is wasteful:
  forces of neighbours are similar yet independently computed
details of the FMM

here I describe traditional Greengard & Rokhlin (1987) FMM

1 preparation phase
1.1 build a hierarchy of cartesian grids
1.2 pre-compute multipole moments etc (upward pass)

2 force computation

2.1 interactions
  on each grid level:
  ▶ perform ‘intermediate-field’ interactions:
    compute & accumulate multipoles of gravity field

2.2 downward pass
  ▶ pass field-multipoles down the hierarchy
  ▶ compute forces on finest grid

▶ theoretical $O(N)$ not demonstrated in practice
▶ not competitive with tree code in low-accuracy regime
details of falcON

- hybrid of tree code & FMM
- takes the better of each method

1 preparation phase (as for tree code)
1.1 build a hierarchical tree of cubic cells
   - cost: \( \mathcal{O}(N \log N) \)
1.2 pre-compute multipole moments etc

2 force computation

2.1 interaction phase
   - ‘catch’ all body-body interactions in well-separated node-node interactions:
     - if node-node interaction is executable
       execute it: accumulate field tensors
     - otherwise
       split it & continue with child iterations (recursive)
   - cost: (better than) \( \mathcal{O}(N) \), dominates

2.2 evaluation phase
   - pass field tensors down the tree
   - compute forces at body positions
   - cost: \( \mathcal{O}(N) \)

- \(~ 10\) times faster than tree code or FMM (at low accuracy)
**Numerics of falcON**

Wanted:

\[ \Phi(x_i) = - \sum_{j \neq i} \mu_j g(x_i - y_j), \]

Taylor expand \( g \) about \( R = x_0 - y_0 \)

\[ g(x - y) = \sum_{n=0}^{p} \frac{1}{n!} (x - y - R)^{(n)} \odot \nabla^{(n)} g(R) + R_p(g), \]

Insert & sum over source cell B

\[ \Phi_{B \rightarrow A}(x) = - \sum_{m=0}^{p} \frac{1}{m!} (x - x_0)^{(m)} \odot C^{m,p} + R_p(\Phi_{B \rightarrow A}) \]

\[ C^{m,p} = \sum_{n=0}^{p-m} \frac{(-1)^n}{n!} \nabla^{(n+m)} g(R) \odot M^n_{B}, \]

\[ M^n_{B} = \sum_{y_i \in B} \mu_i (y_i - y_0)^{(n)}. \]


\[ \sum_{m} : \text{evaluation of gravity, represented by the field tensors } C^{m,p}, \text{ at position } x \]

\[ \sum_{n} : \text{interaction between source cell B, represented by the multipoles } M^n_{B}, \text{ and the sink cell A.} \]

**Difference to tree code:**

- expansion in \( x \) (tree code: \( x \equiv x_0 \))
- mutuality of interactions
gravity between well-separated nodes

If $|R| > r_{A,\text{crit}} + r_{B,\text{crit}}$ with $r_{\text{crit}} = r_{\text{max}}/\theta$,

$\Rightarrow |x - y - R| < \theta |R| \ \forall \ x \in A, \ y \in B$ & Taylor series converges

force error of individual interaction:

$$|\nabla R_p(\Phi_{B \rightarrow A})| \leq \frac{(p + 1)\theta^p}{(1 - \theta)^2} \frac{M_B}{R^2}$$

$\lll \frac{\theta^{p+2}}{(1 - \theta)^2} r_{B,\text{max}}^{d-2} \propto \frac{\theta^{p+2}}{(1 - \theta)^2} M_B^{(d-2)/d}$

$\Rightarrow$ standard tree-code & FMM practice: $\theta = \text{const}$

$\Rightarrow$ relative error controlled

$\Rightarrow$ absolute error increases with $M_B$

$\Rightarrow$ total error dominated by few interactions with large cells

$\Rightarrow$ better:

$\Rightarrow$ balance absolute individual errors by $\theta = \theta(M)$ with

$$\frac{\theta^{p+2}}{(1 - \theta)^2} = \frac{\theta_{\text{min}}^{p+2}}{(1 - \theta_{\text{min}})^2} \left( \frac{M}{M_{\text{tot}}} \right)^{(2-d)/d}$$

$\Rightarrow$ reduce total error
mean (dashed) and 99 percentile (solid) relative force error

\[ \varepsilon \equiv \frac{|a_{\text{approx}} - a_{\text{exact}}|}{a_{\text{exact}}} \]

versus the CPU time (Pentium III/933Mhz in 2001) for a galaxy (left) and a group of galaxies (right), sampled with (total) \( N = 10^4 \) (top), \( N = 10^5 \) (middle), or \( N = 10^6 \) (bottom). We used either \( \theta = \text{const} \) (open triangles) or \( \theta = \theta(M) \) (solid squares). The symbols along each curve correspond, from left to right, to values for \( \theta \) or \( \theta_{\text{min}} \) of 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, and 0.8.
CPU time per body (Pentium III/500Mhz in 2000) versus \( N \) for a galaxy group.

**what complexity?**

▷ 8-folding \( N \) \( \Rightarrow \) \( N_I \rightarrow 8N_I + N_+ \) and thus:

\[
\frac{dN_I}{dN} \approx \frac{N_I \Delta \ln N_I}{N \Delta \ln N} \approx \frac{N_I}{N} + \frac{N_+}{N 8 \ln 8},
\]

with solution

\[
N_I = c_0 N + \frac{N}{8 \ln 8} \int \frac{N_+}{N^2} \, dN
\]

▷ B&H tree code: \( N_+ \propto N \)

\( \Rightarrow \) \( N_I \propto N \log N \)

▷ Here: \( N_+(N) \) grows sub-linear at large \( N \)

\( \Rightarrow \) \( N_I \propto N \)
CPU time per body (2001) versus \( N \) for various techniques. Note that there are differences in the hard- & software, stellar system, and accuracy requirements.

by 2003/2004: falcON is \( \sim 3 \) times faster, but GRAPE-5 tree not.
comparison with FMM

- comparing under same conditions (bodies uniform in a cube)

\[ E = \left[ \sum_{i} \left( \Phi_{i, \text{direct}} - \Phi_{i, \text{approx}} \right)^2 / \sum_{i} \Phi_{i, \text{direct}}^2 \right]^{1/2} \]

- low-accuracy regime: \( \sim 10 \) times faster:

  timing results (in seconds):

<table>
<thead>
<tr>
<th>( N )</th>
<th>( T_{\text{FMM}}^a )</th>
<th>( T_{\text{direct}}^a )</th>
<th>( E^a )</th>
<th>( T_{\text{falcON}}^b )</th>
<th>( T_{\text{direct}}^c )</th>
<th>( E^b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>13.3</td>
<td>233</td>
<td>( 7.9 \times 10^{-4} )</td>
<td>0.97</td>
<td>136</td>
<td>( 3.7 \times 10^{-4} )</td>
</tr>
<tr>
<td>50000</td>
<td>27.7</td>
<td>1483</td>
<td>( 5.2 \times 10^{-4} )</td>
<td>2.64</td>
<td>924</td>
<td>( 3.3 \times 10^{-4} )</td>
</tr>
<tr>
<td>200000</td>
<td>158</td>
<td>24330</td>
<td>( 8.4 \times 10^{-4} )</td>
<td>10.77</td>
<td>14694</td>
<td>( 3.4 \times 10^{-4} )</td>
</tr>
<tr>
<td>500000</td>
<td>268</td>
<td>138380</td>
<td>( 7.0 \times 10^{-4} )</td>
<td>29.42</td>
<td>91134</td>
<td>( 3.7 \times 10^{-4} )</td>
</tr>
<tr>
<td>1000000</td>
<td>655</td>
<td>563900</td>
<td>( 7.1 \times 10^{-4} )</td>
<td>58.34</td>
<td>366218</td>
<td>( 3.5 \times 10^{-4} )</td>
</tr>
</tbody>
</table>

\( a \) FMM; data from Table I of Cheng et al. (1999: JCP, 155, 468)

\( b \) falcON on a computer identical to that used by Cheng et al.

\( c \) our own implementation of direct summation on the same computer

- high-accuracy regime:

  falcON cannot compete with FMM

  \[ \Rightarrow \] accuracy & performance depend on both \( p \) & \( \theta \)

  \[ \Rightarrow \] FMM: fixed ‘\( \theta \)’, vary \( p \)

  \[ \Rightarrow \] falcON: fixed \( p = 3 \), vary \( \theta \)

  \[ \Rightarrow \] high accuracy requires higher order \( p \)
summary

- **falcON** = hybrid of tree code & FMM

  - new features:
    - explicitly exploits mutuality of gravity
      - reduces computational effort
      - requires novel tree-walking algorithm
      - conservation of Newton’s 3rd law
    - mass-dependent $\theta$
      - error balancing
      - reduces cost to **better** than $O(N)$
  
  - $\sim 10$ times faster than tree code or FMM

  - publicly available
more dogmas

▷ balance errors
  ⇒ reduce effort at given accuracy

▷ keep algorithm as simple as possible &
  as complicated as necessary
  ⇒ high-order may be unnecessary

▷ write efficient code
  ⇒ avoid cache misses
    ⇒ data structure design
  ⇒ write generic code
  ⇒ do not rely too much on compiler optimization
    ⇒ template metaprogramming