Multilevel Summation Method in NAMD and VMD

David Hardy
http://www.ks.uiuc.edu/Research/gpu/
NAIS: State-of-the-Art Algorithms for Molecular Dynamics
Multilevel Summation Method


- Fast algorithm for N-body electrostatics
- Calculates sum of smoothed pairwise potentials interpolated from a hierarchal nesting of grids
- Advantages over PME (particle-mesh Ewald) and/or FMM (fast multipole method):
  - Algorithm has linear time complexity
  - Allows non-periodic or periodic boundaries
  - Produces continuous forces for dynamics (advantage over FMM)
  - Avoids 3D FFTs for better parallel scaling (advantage over PME)
  - Permits polynomial splittings (no erf\(c()\) evaluation, as used by PME)
  - Spatial separation allows use of multiple time steps
  - Extends to other types of pairwise interactions (e.g., dispersion forces)
MSM Main Ideas

- Split the $1/r$ potential into a short-range cutoff part plus smoothed parts that are successively more slowly varying. All but the top level potential are cut off.
- Smoothed potentials are interpolated from successively coarser grids.
- Finest grid spacing $h$ and smallest cutoff distance $a$ are doubled at each successive level.

Split the $1/r$ potential

Interpolate the smoothed potentials

\[
\frac{1}{r} = \begin{cases} 
1/r & \text{short-range cutoff part} \\
+ & \text{smoothed parts that are successively more slowly varying}
\end{cases}
\]
Separation

Separation of length scales:

$$\frac{1}{r} = \left( \frac{1}{r} - g_a(r) \right) + g_a(r)$$

\underline{short-range} \quad \underline{smooth}

Requirements for smoothing function:

- \( g_a(r) = \frac{1}{r} \) for \( r \geq a \), short-range part vanishes beyond cutoff
- \( g_a(\sqrt{x^2 + y^2 + z^2}) \) and derivatives are slowly varying everywhere
- \( g_a \) has sufficient continuity
Approximation

Approximate smooth part on 3D grid with spacing $h$:

$$g_a(r, r') \approx \sum_m \phi_{h,m}(r) \, g_a(r_{h,m}, r')$$  \hspace{1cm} \text{interpolate source}$$

$$\approx \sum_m \phi_{h,m}(r) \left( \sum_n \phi_{h,n}(r') \, g_a(r_{h,m}, r_{h,n}) \right)$$  \hspace{1cm} \text{interpolate destination}$$

$$= \sum_m \sum_n \phi_{h,m}(r) \, g_a(r_{h,m}, r_{h,n}) \, \phi_{h,n}(r')$$

Nodal basis function $\phi_{h,m}$:
- continuously differentiable
- local support
Recursively apply *separation* and *approximation* using hierarchy of grids.

Separate smooth part:

$$g_a(r) = \left( g_a(r) - g_{2a}(r) \right) + g_{2a}(r)$$

- $g_a(r) - g_{2a}(r)$ vanishes for $r \geq 2a$
- $g_{2a}$ is more slowly varying than $g_a$

Approximate $g_{2a}$ on 3D grid of spacing $2h$:

$$g_{2a}(r_{h,m}, r_{h,n}) \approx \sum_i \sum_j \phi_{2h,i}(r_{h,m}) g_{2a}(r_{2h,i}, r_{2h,j}) \phi_{2h,j}(r_{h,n})$$

Double *cutoff* and *grid spacing* at each new grid level.
Matrix Formulation

\[ U = \frac{1}{2} \sum_i \sum_{j \neq i} \frac{q_i q_j}{\|r_j - r_i\|} = \frac{1}{2} q^T G q \]

Separation:

\[ G = \hat{G} + \tilde{G} \]

\[ \tilde{G}_{ij} = g_a(\|r_j - r_i\|), \quad \hat{G}_{ij} = \begin{cases} \frac{1}{\|r_j - r_i\|} - g_a(\|r_j - r_i\|), & \text{for } i \neq j, \\ -g_a(\|r_j - r_i\|), & \text{otherwise}, \end{cases} \]

Approximation:

\[ \tilde{G} \approx I_h^* G_h I_h^* \]

\[ (G_h)_{mn} = g_a(\|r_{h,n} - r_{h,m}\|), \quad (I_h^*)_{im} = \phi_{h,m}(r_i), \quad I_h^* = (I_h^*)^T \]

Hierarchy:

\[ G \approx \hat{G} + I_h^* \left( \hat{G}_h I_h^* + I_{2h}^*(G_{2h} I_{2h}^* I_h^*) \right) \]
Multilevel Algorithm

\[ q_\ell = I^\ell_{\ell-1} q_{\ell-1} \]
\[ e_\ell = G^\ell q_\ell \]
\[ q_2 = I^2_1 q_1 \]
\[ e_2 = \hat{G}_2 q_2 + I^2_3 e_3 \]
\[ q_1 = I^1_0 q_0 \]
\[ e_1 = \hat{G}_1 q_1 + I^1_2 e_2 \]
\[ q_0 = I^0_* q \]
\[ e_0 = \hat{G}_0 q_0 + I^0_1 e_1 \]
\[ e = \hat{G} q + I^0_* e_0 \approx G q \]

Computational work requires \( O(N) \) operations.
MSM Calculation

\[
\text{force} = \text{exact short-range part} + \text{interpolated long-range part}
\]

Computational Steps

- \(4h\)-grid
- \(2h\)-grid cutoff
- \(h\)-grid cutoff
- short-range cutoff
- potentials, forces
- positions, charges
- long-range parts

restriction

interpolation

prolongation
Application of MSM in VMD to Photosynthesis

Investigations of the chromatophore, a photosynthetic organelle

Electrostatic field of chromatophore model from multilevel summation method: computed with 3 GPUs (G80) in ~90 seconds, 46x faster than single CPU core in 1 hr, 10 min

Electrostatics needed to build full structural model, place ions, study macroscopic properties

Full chromatophore model will permit structural, chemical and kinetic investigations at a structural systems biology level
More Applications of MSM in VMD
Investigations of Satellite Tobacco Mosaic Virus (STMV) and “swine” flu virus

Investigation of drug (Tamiflu) resistance of the "swine" flu virus demanded **fast response!**
Calculating electrostatics for 20,000 trajectory frames, 27.8 hour job reduced to 1.1 hours
(Linux workstation with Quadro 5800)

Time averaged potential maps:
calculating electrostatics for thousands of trajectory frames,
1.5 hour job reduced to 3 minutes
(NCSA “AC” cluster)
MSM Potentials on the GPU

Accelerate **short-range cutoff** and **lattice cutoff** parts

Performance profile for 0.5 Å map of potential for 1.5 M atoms. Hardware platform is Intel QX6700 CPU and NVIDIA GTX 280.

<table>
<thead>
<tr>
<th>Computational steps</th>
<th>CPU (s)</th>
<th>w/ GPU (s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short-range cutoff</td>
<td>480.07</td>
<td>14.87</td>
<td>32.3</td>
</tr>
<tr>
<td>Long-range anterpolation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>restriction</td>
<td>0.16</td>
<td>1.36</td>
<td>36.4</td>
</tr>
<tr>
<td>lattice cutoff</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prolongation</td>
<td>0.17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>interpolation</td>
<td>3.47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>533.52</td>
<td>20.21</td>
<td>26.4</td>
</tr>
</tbody>
</table>

Multilevel summation of electrostatic potentials using graphics processing units.
MSM Calculation

force = exact short-range part + interpolated long-range part

Computational Steps

- positions, charges
- potentials, forces
- long-range parts
- exact short-range part
- interpolated long-range part
- computational steps
- short-range cutoff
- interpolation
- prolongation
- restriction
- 4h-grid cutoff
- 2h-grid cutoff
- h-grid cutoff
- anterpolation
Short-range Cutoff Summation

- Each lattice point accumulates electrostatic potential contribution from atoms within cutoff distance:
  
  \[
  \text{if } (r_{ij} < \text{cutoff}) \quad \text{potential}[j] \text{ += (charge}[i] / r_{ij}) \times s(r_{ij})
  \]

- Smoothing function \( s(r) \) is algorithm dependent
Short-range Cutoff Summation on GPU

- Atoms are spatially hashed into fixed-size bins
- CPU handles overflowed bins (GPU kernel can be aggressive)
- GPU thread block calculates corresponding region of potential map
- Bin/region neighbor checks costly; solved with universal lookup table

Each thread block cooperatively loads atom bins from surrounding neighborhood into shared memory for evaluation: GATHER

Global memory
- Potential map regions
- Bins of atoms

Shared memory
- atom bin

Constant memory
- Offsets for bin neighborhood
- Lookup table encodes “logic” of spatial geometry
Spatial Sorting of Atoms Into Bins

- Sort atoms into bins by their coordinates
- Each bin is sized to guarantee GPU memory coalescing
- Each bin holds up to 8 atoms, containing 4 FP values (coords, charge)
- Each lattice point gathers potentials from atom bins within cutoff
Using CPU to Improve GPU Performance

- GPU performs best when the work evenly divides into the number of threads / processing units

- Optimization strategy:
  - Use the CPU to “regularize” the GPU workload
  - Use fixed size bin data structures, with “empty” slots skipped or producing zeroed out results
  - Handle exceptional or irregular work units on the CPU while the GPU processes the bulk of the work
  - On average, the GPU is kept highly occupied to attain good fraction of peak performance
MSM Calculation

\[
\text{force} = \text{exact short-range part} + \text{interpolated long-range part}
\]

Computational Steps

- long-range parts
- restriction
- interpolation
- positions, charges
- potentials, forces
- short-range cutoff
- \( h \)-grid cutoff
- \( 2h \)-grid cutoff
- \( 4h \)-grid

prolongation

restriction
Lattice Cutoff Summation

- Potential summed from grid point charges within cutoff
- Uniform spacing enables distance-based interactions to be precomputed as stencil of “weights”
- Weights at each level are identical up to scaling factor (!)
- Calculate as 3D convolution of weights
  - stencil sizes range from 9x9x9 up to 23x23x23
Lattice Cutoff Summation on GPU

- Store weights in constant memory (padded up to next multiple of 4)
- Thread block calculates 4x4x4 region of potentials, stored contiguously for memory coalesced reads
- Pack all regions over all levels into 1D array (each level padded with zero-charge region)
- Store map of level array offsets in constant memory
- Kernel has thread block loop over surrounding regions of charge (load into shared memory)
- All grid levels are calculated concurrently, scaled by level factor (keeps GPU from running out of work at upper grid levels)

Each thread block cooperatively loads regions of grid charge into shared memory, multiply by weights from constant memory.

Global memory:
- Grid potential regions
- Grid charge regions

Shared memory:
- Subset of grid charge regions

Constant memory:
-Stencil of weights
Apply Weights Using Sliding Window

- Constant memory offers best performance when thread block collectively accesses the same location
- Read 8x8x8 grid charges (8 regions) into shared memory
- Window of size 4x4x4 maintains same relative distances
- Slide window by 4 shifts along each dimension