Bounded-Error Compression of Particle Data from Hierarchical Approximate Methods

Dow-Yung Yang* and Ananth Grama

(* Primary Author is a Student)
Computer Science Department,
Purdue University,
West Lafayette, IN 47907.
{yangdy, ayg}@cs.purdue.edu

Vivek Sarin

Department of Computer Science,
Texas A & M University,
College Station, TX 77843
sarin@cs.tamu.edu

Abstract

This paper presents an analytical and computational framework for the compression of particle data resulting from hierarchical approximate treecodes such as the *Barnes-Hut* and *Fast Multipole Methods*. Due to the approximations introduced by hierarchical methods, the position (as well as velocity and acceleration) of a particle can be bounded by a distortion radius. We develop storage schemes that maintain this distortion radii while maximizing compression. Our schemes make extensive use of spatial and temporal coherence of particle behavior and yield compression ratios higher than 12:1 over raw data, and 6:1 over gzipped (LZ78) raw data. We demonstrate that for uniform distributions with 100K particles, storage requirements can be reduced from 1200KB (100K $\times$ 12B) to about 99KB (under 1 byte per particle per timestep). This is significant because it enables faster storage/retrieval, better temporal resolution, and improved analysis. Our results are shown to scale from small systems (2K particles) to much larger systems (over 100K particles). The associated algorithm is optimal ($O(n)$) in both storage and computation with small constants.

1 Introduction and Motivation

Particle methods find application in a variety of domains ranging from molecular dynamics to astrophysics. Starting from an initial state, the system state is advanced by computing forces (such as Coulombic and Lennard-Jones) at each timestep and advancing the particles using a leapfrog method. A simple all-to-all force computation in an n particle system results in a complexity of $O(n^2)$ because of long-range Coulombic (or gravitational) forces. A number of approximation techniques have been explored to reduce this complexity. Of these, the prominent ones are lattice-

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

based methods and treecodes such as Barnes-Hut [3] and Fast Multipole Method (FMM) [9]. This paper focuses on compression of particle data resulting from simulations based on hierarchical treecodes. While considerable work has gone into compression techniques for both topology and geometry data [11, 2, 10, 5], the unique characteristics of the problem combined with theoretical error bounds provide us with unique opportunities for compression.

Multipole methods (both FMM and Barnes-Hut variants) use a truncated series approximation of charges within a localized region to estimate impact on well-separated sets of particles. The method of Barnes and Hut relies simply on particle-cluster interactions to achieve an $O(n \log n)$ computational bound for uniform particle distributions. The fast multipole method uses both particle-cluster and cluster-cluster interactions to achieve an $O(n)$ bound for uniform distributions. For non-uniform distributions, similar bounds can be obtained by using box-collapsing techniques of Callahan and Kosaraju [4] or the chaining techniques of Aluru and Gustafson [1].

The reduced complexity comes at the expense of approximations introduced by hierarchical treecodes. Specifically, it can be shown that the error in potential due to a set of charges circumscribed in a circle of radius r_s at a distance r from the center is bounded by:

$$\epsilon \leq \frac{A}{r - r_s} \left(\frac{r_s}{r} \right)^{p+1}, \tag{1}$$

where p is the degree of the truncated multipole series and $A = \sum_{j=1}^{k} |q_j|$ (i.e. sum of circumscribed charges) [8]. In [7], we show that the aggregate error in potential at a particle due to all charges in the Barnes-Hut method applied to a uniform particle system is given by:

$$\epsilon \leq A\alpha^{p+1}. \tag{2}$$

Here, α corresponds to the α criterion of the Barnes-Hut method and is generally chosen to be between 0.5 and 0.8. For the Fast Multipole Method, similar bounds can be derived with the constant α determined by the well-separatedness criterion of FMM. These proofs follow from bounds on number of interactions and their distance ratios. We use these results to develop variable degree multipoles to reduce this bound to:

$$\epsilon \leq \log(A)\alpha^{p+1}. \tag{3}$$

Similar bounds have been established for force computations using multipole methods as well [6]:

$$\epsilon_{\text{FORCE}} \leq A \times p\alpha^{p-1}. \tag{4}$$

These expressions have been simplified to demonstrate error behavior as functions of domain and simulation parameters. More accurate error expressions are presented in [6, 7].

Multipole methods are used to compute the forces and/or potentials at each timestep. These forces are used to advance particle positions using methods such as the leapfrog scheme. In a typical simulation, the positions and velocities associated with particles need to be stored and analyzed. For a small system with 20K particles, storing the positions (coordinate data) requires 12 bytes per particle (3 floats, 4 bytes/float) and approximately 240KB for the entire system per stored timestep. Due to this high storage requirement, data is typically stored once every k timesteps, where k is selected based on available storage, underlying phenomena being analyzed, and timestep size. This places severe restrictions on the post-processing operations that need to be performed on the data since phenomena at lower time-scales are completely lost in temporal sub-sampling.

In this paper, we present a set of schemes for compression of particle data that use the same hierarchical data structure as the potential estimation and force computation framework. These compression schemes rely on the fact that the approximate nature of the force computation phase introduces a distortion sphere around the particle. The particle may lie anywhere within this distortion sphere and still maintain accuracy bounds guaranteed by the hierarchical treecode. The compression schemes developed in this paper place the particle appropriately within the distortion sphere to maximize compression. They combine this with spatial and temporal coherence of particle behavior to

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

improve compression. Spatially proximate particles are likely to display coherent behavior. Similarly, particles are expected to demonstrate coherent behavior in successive timesteps. Using these, we develop a family of schemes that reduce the storage-per-particle to under 8 bits/particle per timestep in the best case. This corresponds to a compression ratio of over 12:1 over raw data and over 6:1 over gzipped raw data (LZ78). In addition to excellent compression ratios, we demonstrate the following desirable properties of our compression scheme:

- Bounded error rates – in particular, we maintain the error bounds of the original treecode.
- High compression and decompression rates – both $O(n)$ with small constants.
- Supports fast querying of intermediate frames – using intermediate MPEG-style I-frames, we can access any intermediate frame quickly and decompress the frame in $O(n)$ time. Sub-domain retrievals can be accomplished in $O(k)$ time where k is the number of particles in the sub-domain. All of these complexities are asymptotically optimal.
- Provides an in-built framework for analysis of spatial and temporal artifacts in data.

In Section 2, we describe a family of compression schemes based on our framework, Section 3 presents compression ratios and timings for these schemes, and the impact of proposed schemes is discussed in Section 4.

2 Compression Schemes for Particle Data

The basis for the compression schemes presented in this paper is the freedom to assign particles (quantize) to desirable points within a specified distortion sphere of radius ϵ. Here, ϵ is determined by the error in the multipole method. The following theorem derives the bound on the distortion radius for a particle:

Theorem 2.1 After n timesteps of size Δt, the distortion radius of a particle is bounded by

$$|E_n| \leq \frac{T^2 + T\Delta t}{2} \left[c_1 + \Delta t^2 c_2\right],$$

where $T = n\Delta t$ is the total time, and c_1 and c_2 are constants.

Proof Consider the Verlet-Leapfrog scheme to compute the position and velocity of each particle:

$$v_{k+1/2} = v_{k+1/2} + a_k \Delta t, \tag{5}$$

$$s_{k+1} = s_k + v_{k+1/2} \Delta t, \tag{6}$$

where s_k, v_k, and a_k denote the position, velocity, and acceleration, respectively, of a particle at time t_k after k timesteps of size Δt each, i.e., $t_k = k\Delta t$. The difference of (6) at t_{k-1} and t_k is

$$s_{k+1} - s_k = s_k - s_{k-1} + a_k \Delta t^2. \tag{7}$$

Taylor series expansion of the actual position $s(t_{k\pm1})$ around $s(t_k)$ is given as

$$s(t_{k\pm1}) = s(t_k) \pm \Delta t v(t_k) + \frac{\Delta t^2}{2} a(t_k) \pm \frac{\Delta t^3}{6} a'(t_k) + \frac{\Delta t^4}{24} a''(\tau_{k\pm1}),$$

where the last term represents the truncation error. Manipulating the sum of $s(t_{k+1})$ and $s(t_{k-1})$, we get

$$s(t_{k+1}) - s(t_k) = s(t_k) - s(t_{k-1}) + \Delta t^2 a(t_k) + \frac{\Delta t^4}{24} \left[a''(\tau_{k+1}) + a''(\tau_{k-1})\right]. \tag{8}$$

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

Defining the local error in position as

$$E_k = s_k - s(t_k),$$

the difference of (7) and (8) yields the recurrence

$$E_{k+1} - E_k = E_k - E_{k-1} + \Delta t^2 \left[a_k - a(t_k) + \frac{\Delta t^2}{24} \left[a''(\tau_{k+1}) + a''(\tau_{k-1}) \right] \right].$$

Next, we define a new variable:

$$E'_k = E_k - E_{k-1}.$$

Assuming there exist constants c_1 and c_2 such that

$$\begin{aligned} c_1 &\geq |a_k - a(t_k)|, \\ c_2 &\geq \frac{\Delta t^2}{24} \left| a''(\tau_{k+1}) +'' (\tau_{k-1}) \right|, \end{aligned}$$

we have

$$E'_{k+1} \leq E'_k + \Delta t^2 \left[c_1 + \Delta t^2 c_2 \right].$$

Thus,

$$E'_k \leq k \Delta t^2 \left[c_1 + \Delta t^2 c_2 \right],$$

which implies that

$$E_k \leq E_{k-1} + k \Delta t^2 \left[c_1 + \Delta t^2 c_2 \right].$$

Therefore, after n steps,

$$|E_n| \leq \frac{n(n+1)\Delta t^2}{2} \left[c_1 + \Delta t^2 c_2 \right]. \tag{9}$$

Equation 9 gives us an analytical bound on the distortion radius after n timesteps. For simulations over a fixed time period T, the error in position is given by

$$|E_n| \leq \frac{T^2 + T\Delta t}{2} \left[c_1 + \Delta t^2 c_2 \right],$$

which is proportional to $c_1 + \Delta t^2 c_2$. The dominant component of c_1 is the error in force computations arising from approximations in the multipole methods as given by Equation 4. □

We start with the description of a scheme for storing particle positions and then use the framework for storing particle displacements and difference in displacements (second order differences). Finally, we present a scheme that uses MPEG style index frames. Particle positions for interspersed non-index frames are computed differentially with respect to a trajectory interpolated over these index frames.

2.1 Framework for Bounded Distortion Multi-Dimensional Quantization

Given a set of particles with specified distortion radii, the basic scheme imposes an oct-tree structure over the domain. The refinement criteria for this oct-tree is that tree nodes are subdivided if they contain more than one particle [1], or the center of the node is outside the distortion radii of the particle in the node. Note that this refinement criteria is not the same as the refinement criteria for the FMM or Barnes-Hut tree. The refinement process is illustrated in Figure 1. The hierarchical structure used for compression is in fact a refinement of the Barnes-Hut or FMM tree and does not need to be explicitly constructed.

[1] This is not entirely necessary. If an oct contains two particles and falls within both of their distortion spheres, the simulation accuracy must be increased.

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

Once a hierarchical structure has been constructed, particles are assigned to leaf nodes that lie within distortion radii. The problem of representing particle positions now reduces to the problem of representing populated leaf nodes in the oct-tree. This is done by coding the path from the root to the leaf node. By associating a pre-defined ordering of children in the oct tree, we can associate 3 bits per level in the tree. We illustrate this process for a 2-D problem in Figure 1. In this case, we require only 2 bits per node since the tree is a quad tree. In the example, particle `a` is at level three in the tree and according to the predefined node ordering for children of a node, it is represented as `00 00 11`. This provides the basic quantization mechanism for our schemes.

2.1.1 Encoding Spatial Coherence

The next order of compression relies on spatial coherence of representation. Simply stated, particles that are spatially proximate are likely to share large prefixes in the path from root to leaf. This implies that if the particles are sorted in a proximity preserving order (such as Morton or Hilbert curves), then we can represent particle positions relative to the previous particle. Indeed, for improved cache performance as well as parallel performance, particles are typically sorted in a spatial order (such as a Morton order or a Peano-Hilbert order) for the FMM/Barnes-Hut computation and this requires no additional processing for compression.

The use of spatial coherence for improving compression ratios is illustrated in Figure 1, Time-step 0. The quantized representations for particles `a`, `b`, and `c` are given by `00 00 11`, `00 10`, and `00 11 01` respectively. Assuming that these particles are sorted in the order `a`, `b`, and then `c`, it is easy to see that particles `b` and `c` share the prefix `00` with particle `a`. Consequently, the prefix does not need to be stored for these and the representations for `b` and `c` are simply `10` and `11 01`. While this may not seem to be a significant improvement, in typical trees, the depth can be very high. For example, with a normalized domain of unit size in each dimension, a distortion radius of 10^{-3} would require up to 10 levels in the oct tree. In such cases with higher particle densities, significant improvements result from the use of spatial coherence in our quantization framework.

The performance of spatial coherence encoding is impacted by the fact that two spatially proximate particles may have significantly different prefixes based on which half of the tree they fall into. Trivially, two particles that lie on either side of a half split will have entirely different prefixes. In such cases, the spatial coherence framework does not provide any additional benefits over simple quantized representation. However, the number of such particles is a small fraction of the total number of particles.

2.1.2 Encoding Temporal Coherence

In addition to spatial coherence, compression rates can be further improved by considering temporal coherence; i.e. a particle is not expected to move significantly over a single timestep. Consequently, instead of storing the entire relative leaf location corresponding to a populated leaf node, we can store only the difference with respect to the previous location. This is stored as a sequence of XOR'ed least significant bits. This is illustrated in Figure 1, Time-step 1. The quantized positions of particles `a`, `b`, and `c` are given by `00 00 01`, `10`, and `11 11` respectively after coding spatial coherence. These representations can be reduced to `10` and `10` respectively for `a` and `c` since the rest of the prefix is shared with the previous timestep. Also, since the representation of `b` did not change between the two timesteps, it does not require any storage. It is easy to see from this simple illustration that the hierarchical framework when combined with spatial and temporal coherence is capable of yielding very high compression rates.

An important aspect that impacts the performance of time-coherence exploitation is the choice of boundary conditions in the simulation. Typical simulations are run with periodic boundary conditions; i.e., the simulation domain is infinitely replicated around itself. With such boundary conditions, a particle leaving the domain of simulation re-enters it on the far side. This causes a significant shift in the particle position and for these particles the differentially coded position requires the same amount of storage as a naive representation of the path from root to the leaf node. However,

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

the number of such particles is small and does not significantly degrade the performance of the compression scheme.

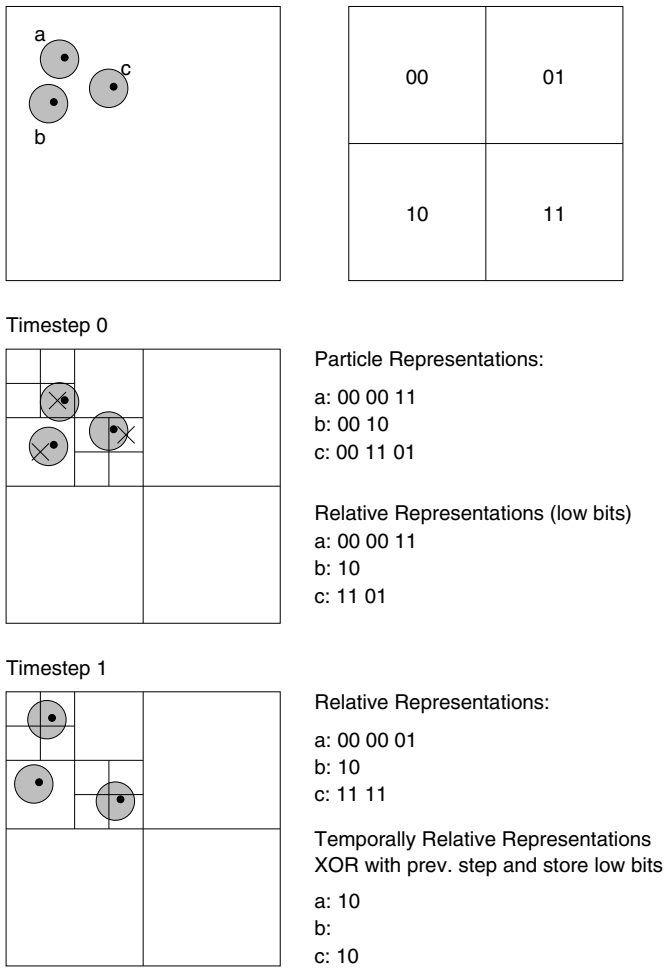


Figure 1: Illustration of compression of particle position data using distortion sphere, and spatial and temporal coherence.

2.1.3 Coding Particle Displacements and Higher Order Differences

It is possible to code particle displacements and higher order differences in the same framework as illustrated above. The driving intuition behind coding displacements between timesteps is that spatially proximate particles are expected to have coherent displacements. Furthermore, displacements of particles between timesteps are not expected to change abruptly. To code particle displacements in the above framework, the refinement criteria for the oct-tree must be modified slightly – a node is sub-divided only until its center is within prescribed distortion radii. The condition relating to single particle per leaf is not required. Furthermore, the distortion radius needs to be altered appropriately to maintain the same distortion radii for particle positions as before.

This compression process is illustrated in Figure 2. Between timesteps 0 and 1, particles `a`, `b`, and `c` are displaced by vectors `(0.1, 0.3)`, `(0.05, 0.25)`, and `(0.35, 0.15)` respectively. These values are now coded in our quantization framework as before. Since more than one particle can have the same displacement (but not the same position), multiple particles can be assigned to the same leaf node in the hierarchy. The hierarchy corresponding to the displacements is illustrated in Figure 2. The hierarchy is populated with vectors, each corresponding to the

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

displacement of a single particle. Populated leaf nodes in this hierarchy are now coded using spatial coherence.

An important consideration in this process is the choice of timestep data with respect to which differences are computed. Consider two vectors T_i and T_{i+1} of positions along with their error-bounded quantized counterparts T_i' and T_{i+1}'. To code T_{i+1}, its differences with respect to T_i' must be coded instead of vector T_i (as would be the obvious choice). This is because when uncompressing this data, we have access only to T_i' and not T_i. Therefore, the differences must be error bounded with respect to T_i' and not T_i, otherwise errors will accrue and exceed distortion bound. A similar technique is used for coding second order differences in particle positions.

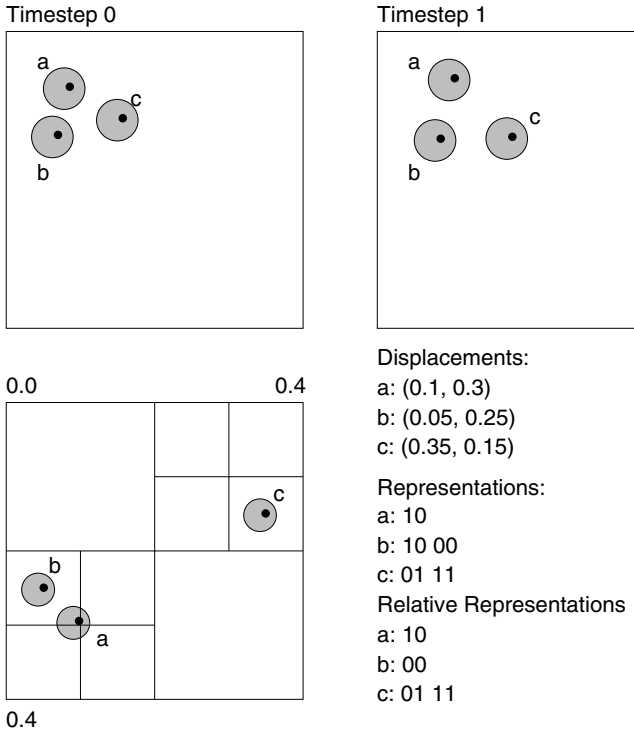


Figure 2: Illustration of compression of particle displacement data using distortion sphere and spatial coherence. Notice that the hierarchical domain is determined by the maximum displacement in any dimension and that the radius of distortion sphere is in fact smaller than for coding positions.

2.2 Index Time-Steps for Interpolated Encoding

An alternate scheme for compressing particle positions relies on using periodic index frames. Consider the illustration in Figure 3. This example shows a 1D trajectory of a single particle across nine timesteps. Errorbars on particle positions indicate the distortion radius introduced by the hierarchical method. In this example, every fourth frame is an index frame (marked by an I in the figure). A simple technique for coding particle positions in this framework uses the neighboring index frames to construct a polynomial that specifies particle positions at intermediate timesteps. If the polynomial approximation lies within the error-bars, no data needs to be stored for the particle. If the polynomial lies outside the errorbar, then the data can be differentially coded with respect to the polynomial. In the example in Figure 3, it is easy to see that there is no storage associated with timesteps 1, 2, 5, and 7 (0, 3, 6, and 9 are index timesteps). Only timesteps 4 and 8 require additional encoding. Clearly, this technique has the potential for significant compression.

We now look at the reduction in magnitude of quantities that need to be encoded because of the interpolation

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

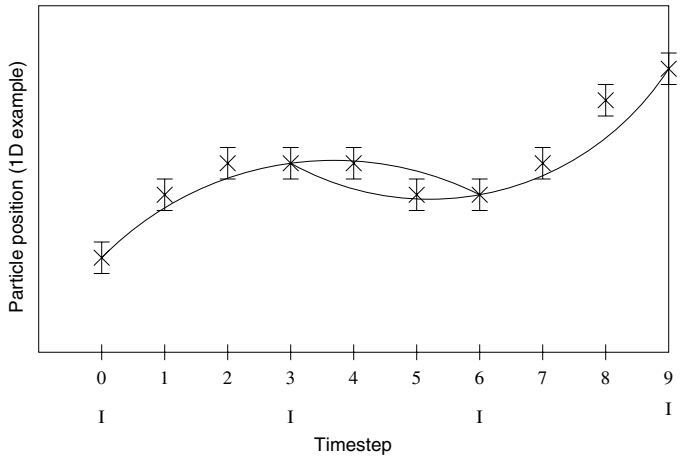


Figure 3: Using Index (I) timesteps for interpolating particle positions for intermediate timesteps.

framework above. In Figure 4, we illustrate the use of index frames with interpolation polynomials of varying degrees for a 2568 atom system (water). For the sake of illustration, we only show the encoding of the x coordinate. The magnitude of difference between the polynomial and actual position computed from the timestepping scheme applied to the hierarchical method is shown. In the extreme case, if each of the magnitudes in the figures were under, say, 10^{-2}, and the distortion radius was 10^{-2}, then no encoding is required for the intermediate timesteps. In Figure 4, we see that the magnitude of differences can be reduced very significantly by quadratic and cubic interpolation. We experimented with varying number of intermediate timesteps and present results for these in Section 3. Once particle trajectories have been interpolated, the data is differentially coded with respect to this trajectory. This differential coding is done using the hierarchical framework used for quantizing particle positions as before.

This approach of using index frames for compression has other benefits over simply coding first and second order differences in particle positions. Loss or corruption of data corresponding to any timestep in the original scheme renders all subsequent timestep data useless. Furthermore, if during the analysis phase, it is necessary to query an intermediate timestep, we would need to unroll all the timesteps leading up to the required step. This is expensive and undesirable. The I-frame approach alleviates the drawbacks of differentially coding timesteps and supports fast access to intermediate timestep data with limited unrolling. Note that this approach is similar in philosophy to that taken by MPEG coders for video compression.

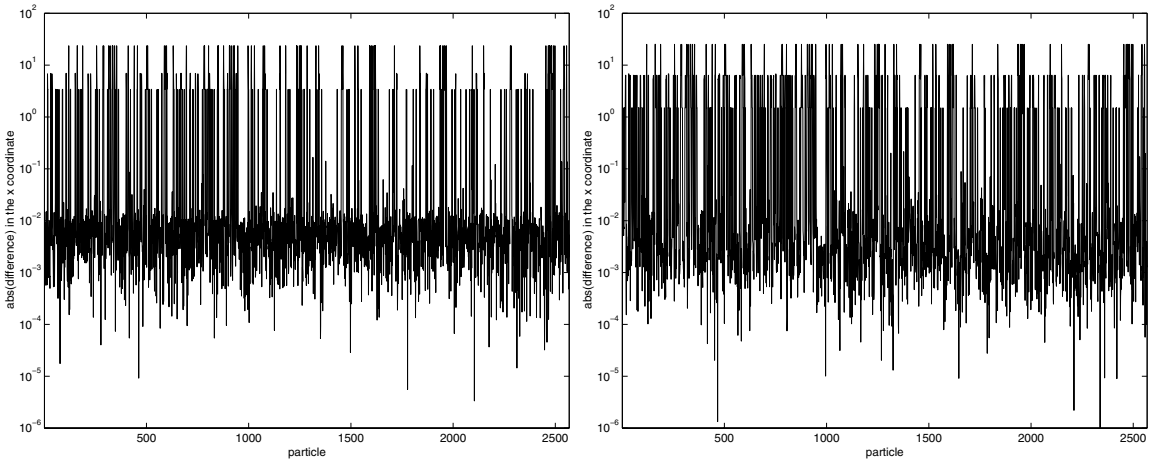
2.3 Performance of Compression Schemes

It is worth mentioning that the best compression for such applications is simply to store the initial state of the system. The decompressor is the simulation process itself. This highlights the tradeoff between computational overhead (speed of compression and decompression), and the compression ratio. It is important for any compression scheme to have the following properties:

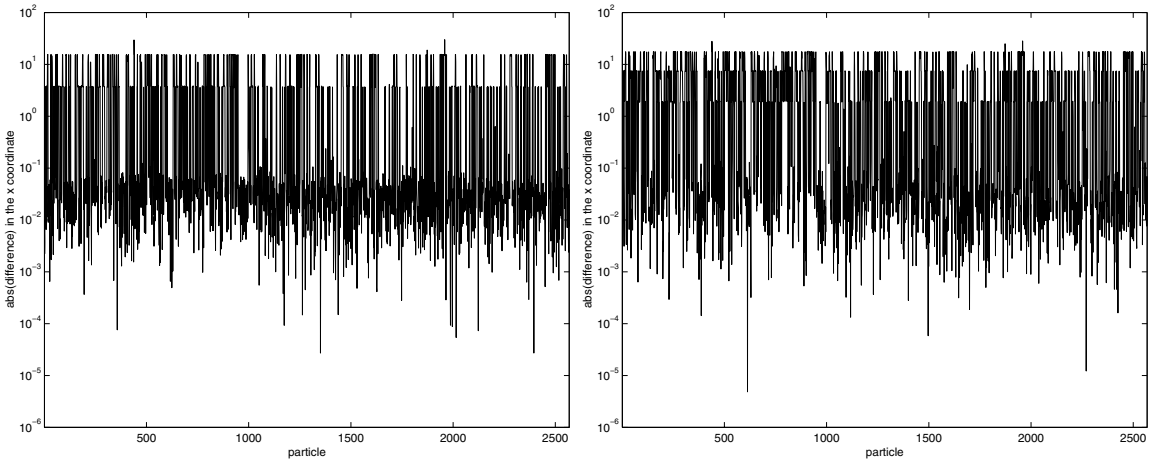
- Fast compression and decompression.
- High compression ratios.
- Data-structures that can build on existing hierarchies and support further analysis tasks on the data.

Our schemes address each of these requirements explicitly. The performance of the proposed schemes is a sensitive function of the distortion radius associated with the quantity being stored. While theoretical bounds exist on distortion

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE



Quadratic and cubic fitting: Graphs show magnitude of difference between interpolated and actual values. Each pair of index frames is separated by two interpolated timesteps. Differences are shown for the second interpolated timestep.



Four interpolated timesteps between each pair of index frames. Magnitude of differences are shown for the third interpolated timestep.

Figure 4: Use of index frames for interpolating intermediate timesteps.

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

radii, it is often observed that these bounds are loose and that hierarchical methods yield much better results in practice. For our compression schemes, we use distortion radii which are much smaller than theoretical bounds. For instance, for a charge-normalized system with charges totaling to 1, the error behavior is given by α^{p+1}. For $\alpha = 0.5$, an increase in multipole degree by one leads to halving of the error term. In the oct-tree structure for compression, each additional level corresponds to halving of the distortion radius. Therefore, the depth of the tree is expected to grow linearly with the multipole degree. For a compression tree depth of 10, the distortion radius is 2^{-11} and the corresponding storage per path is 30 bits (3 bits per level). Notice that without using any coherence (spatial or temporal), we can already achieve a 3-fold compression (down from 12 bytes) in storing positions. Spatial coherence can be coded either by sorting particles in a proximity order and coding differences with respect to previous particles or by using oct-tries (simple variants of binary-tries used in LZ78). We chose to use the former since the particles are already sorted in proximity order for force/potential estimation.

The compression algorithm can be easily parallelized (threaded) and incurs little parallel overhead. Subdomains assigned to processors can be independently compressed. It is assumed that the first particle is always coded independently (i.e. no differential coding). Subsequent particles are coded using spatial and temporal coherence. This is important because no serialization bottlenecks are added in the simulation / online analysis phases. This also provides a natural de-clustering of particle data across disk for improved parallel I/O performance. There is a slight loss of compression since each processor treats its sub-domain independently of the others; thus leading to loss of coherence. However, in practice, this loss of compression is negligible.

3 Experimental Results

We have implemented our compression framework in conjunction with a fast multipole based leapfrog particle dynamics scheme. We tested our schemes on various problems with sizes ranging from 2K to 100K particles and distributions ranging from uniform to Gaussian and multiple Gaussians. These distributions are illustrated in Figure 5. Uniform distributions are synthetically generated by locating particles randomly across the domain. Gaussian distributions are generated by altering the point density according to a Gaussian curve and generating the required point density randomly.

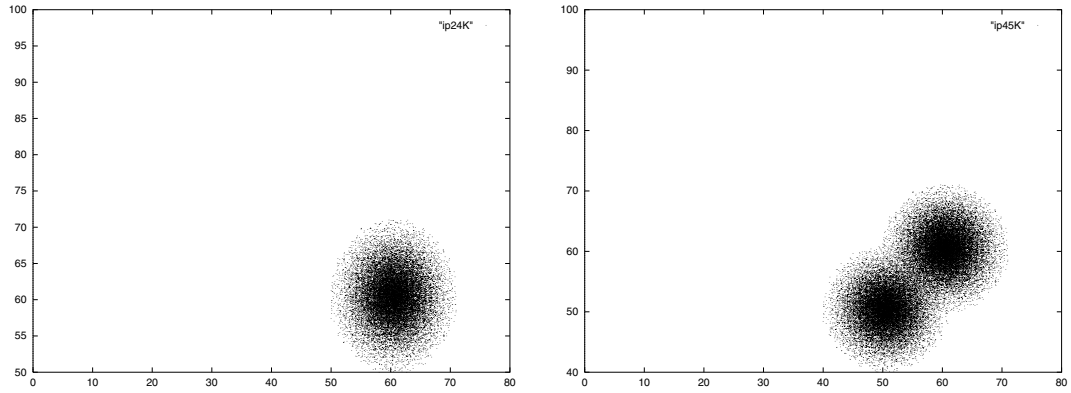


Figure 5: Sample distributions for experiments: (a) Gaussian (24,000 particles), and (b) Overlapped Gaussians (45,000 particles).

We present results of the basic compression scheme followed by the effect of various optimizations in Table 1. The following observations can be made from the table: coding particle position data using the hierarchical distortion radii proposed in the paper with spatial coherence yields up to a factor of three improvement in compression over

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

gzipped raw data. This corresponds to a compression ratio of six over raw data and can be seen to scale very well with problem size. Applying temporal coherence for coding the next timestep yields improvements in the range of 20% over spatial coherence. This is less than expected; however, it is due to the fact that the selected distortion radii is very small. Consequently, particles traverse significant number of leaf octs in the tree-space. The last two columns correspond to storing first and second order differences in particle positions in the hierarchical framework using only spatial coherence. It can be seen that storing second order differences yields the best storage of any scheme and yields compression ratios ranging from 4:1 to 12:1 over raw data and 2:1 to 6:1 over gzipped data.

Problem Size	Position Data gzipped (LZ78)	Position Data (Spat. Coh.) (Scheme 1)	2nd timestep (Temp. Coh.) (Scheme 2)	Displacement Data (Spat. Coh.) (Scheme 3)	2nd Order Diff. (Spat. Coh.) (Scheme 4)
2568	15289	10646	8453	8612	7602
10K	76123	38329	23007	18149	14314
20K	157426	66623	45649	33307	25761
100K	704018	219755	232050	135444	98820
Unstructured Distributions					
24K	241036	58106	107403	96550	64043
45K	457151	91784	182312	160333	107031

Table 1: Compression results for uniform particle distributions of varying sizes.

An interesting observation from the table is that compression improves as the number of particles is increased. This is because the error bound of multipole methods grows with the total charge in the system. Thus the distortion radii increases with increase in number of particles. Consequently, the compression ratio increases. This also points to an important fact that the accuracy of multipole methods needs to be increased (by changing the α parameter or multipole degree) as systems become larger. In that case, we expect the compression ratios to be largely independent of the size of particle system. For unstructured distributions, it can be seen from Table 1 that compression ratios are much lower than their structured counterparts. For example, for a 100K particle uniform distribution, the encoded space is 98820 bytes, whereas for a 45K particle unstructured distribution, it is 107031 bytes. This large difference is a result of the fact that the tree depth in unstructured distributions can be much higher. For this purpose, it may be necessary to use alternate hierarchical data structures. The discrepancy is also a function of the fact that we use the same distortion radii in both cases. In reality, the distortion radii for unstructured distributions is much higher for simple FMM/Barnes-Hut methods.

The relative improvements due to our schemes are illustrated in Figure 6. It is easy to see that our schemes yield excellent compression ratios in a fast and flexible framework.

In Table 2, we present results from compression schemes based on the use of index timesteps. We present encoded results for quadratic and cubic fitting for a number of distributions. The size of compressed data is computed as an average of one index timestep and all of the intermediate timesteps up to (but not including) the next timestep. This is done to ensure a fair comparison of various schemes. The number of intermediate (interpolated) timesteps is varied between 2 and 4. Several observations can be made from Table 2 and its comparison to Table 1. The compression ratio for uniform distributions improves with increasing number of intermediate frames. This is expected to saturate and then decrease as the number of intermediate frames is further increased. A comparison with Table 1 also reveals that this scheme yields the best compression of any scheme for smaller systems. For larger systems (100K), the second order difference scheme with spatial coherence outperforms the I-frame approach. The storage requirement of the scheme is almost linear in the number of particles and approaches 8 bits/particle/timestep. Furthermore, a quadratic fit

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

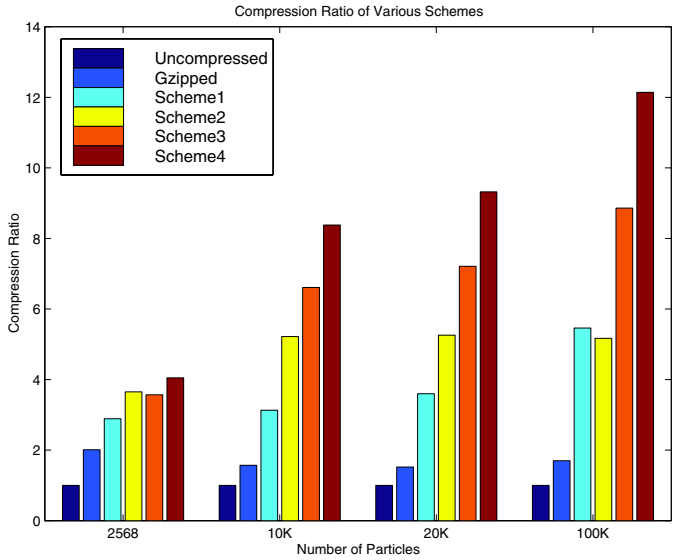


Figure 6: Comparison of compression rates of various schemes: Scheme 1 compresses particle positions with only spatial coherence, Scheme 2 compresses particle positions using both spatial and temporal coherence, Scheme 3 compresses particle displacements with spatial coherence, and Scheme 4 compresses second order differences with spatial coherence.

is largely adequate and higher order interpolations do not result in significant improvements in performance. In some cases, they result in slightly poorer performance. This is because cubic interpolants use the previous 4 I-frames for prediction and in doing so make potential mistakes.

Problem	I-FRAME SCHEME					
	Quadratic Fit			Cubic Fit		
	N of Interpolated Frames			N of Interpolated Frames		
Size	2	3	4	2	3	4
2568	6127	6382	6843	6325	6963	7669
10K	17521	14039	12050	16592	13429	11972
20K	34873	27955	24293	33097	27005	24406
100K	174257	141260	126841	166801	140221	134161

Table 2: Compression results for uniform particle distributions of varying sizes.

4 Discussion and Impact

The compression schemes presented in this paper reduce storage requirements of particle data by as much as a factor of 12 without any further loss of accuracy. Improvements using the I-frame approach pushes this compression rate closer to 15-20. This has several important implications for particle simulations: given a subsampling frequency, the I/O overheads can be reduced significantly, conversely, given a desired overhead factor, the sampling rate can be increased significantly for better analysis. It also facilitates remote access across networks where bandwidth is a bottleneck. The

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE

framework put forth in this paper provides a natural mechanism for clustering and quantifying cluster behavior. It can be used to identify high energy regions of the domain and other quantitative sub-domain features. It also supports segmented retrieval without expensive unrolling overheads in asymptotically optimal time.

While our schemes have achieved excellent results, we are exploring the following issues as a part of our current work: using frequency domain transforms for further improving compression ratios, incorporating analysis tools into our compression framework, and using our compression framework for supporting progressive visualization.

References

[1] Srinivas Aluru. Greengard's n-body algorithm is not O(n). *SIAM Journal on Scientific Computing*, 17(3):773–776, 1996.

[2] C. Bajaj, V. Pascucci, and G. Zhuang. Single resolution compression of arbitrary triangular meshes with properties. In *Proceedings of Data Compression Conference*, 1999.

[3] J. Barnes and P. Hut. A hierarchical o(n log n) force calculation algorithm. *Nature*, 324, 1986.

[4] P. B. Callahan and S. R. Kosaraju. A decomposition of multi-dimensional point-sets with applications to k-nearest-neighbors and n-body potential fields. In *Proceedings of 24th Annual ACM Symp. on Theory of Computing*, pages 546–556, May 1992.

[5] M. Deering. Geometry compression. In *Proceedings of SIGGRAPH'95*, pages 13–20, 1995.

[6] William D. Elliott. Revisiting the fast multipole algorithm error bounds. Technical Report TR94-008, Duke University, 1994. available from http://www.ee.duke.edu/Research/SciComp/Papers/TR94-008.html.

[7] Ananth Grama, Vivek Sarin, and Ahmed Sameh. On error and computation complexity of multipole methods. Technical report, Department of Computer Science, Purdue University, W. Lafayette, IN 47906, 1999. Accepted for publication in SIAM J. Sci. Comput.

[8] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. MIT Press, 1987.

[9] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comp. Physics*, 73:325–348, 1987.

[10] M. Levoy. Polygon-assisted jpeg and mpeg compression of synthetic images. In *Proceedings of SIGGRAPH'95*, pages 21–25, 1995.

[11] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84-115, 1998.

Proceedings of the ACM/IEEE SC99 Conference (SC'99)
1-58113-091-0/99 $ 17.00 © 1999 IEEE