Parallel Hierarchial N-Body Methods

John K. Salmon

CRPC-TR90115
1990

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

CRPC-90-14 December 6, 1990

Parallel Hierarchical N-Body Methods*

John K. Salmon

C3P Physics 206-49
California Institute of Technology
Pasadena, California 91125

This work was supported in part by the NSF under Cooperative Agreement No. CCR-8809615.

PARALLEL HIERARCHICAL N-BODY METHODS

Thesis by
John K. Salmon

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena,California

1991

(Defended December 6, 1990)

TCaHoB, ttboai

Yuk Ha	Ben Finley	Chris Meisl	Andrea Mejia	Susan Gerhart
Ed Vail	Bill Jones	Cliff Kiser	Anna Jaeckel	Susan Ridgway
Kim Liu	Brad Scott	Clo Butcher	Anna Yeakley	Tanya Kurosky
Nye Liu	Bruce Bell	Curtis Ling	Anthony Chan	Walker Aumann
Alex Wei	Charles Hu	Dave Agabra	Craig Volden	Bill Gustafson
Dave Kim	Dan Frumin	Dave Krider	David Knight	Bonnie Wallace
Ian Agol	Dave Skeie	Dawn Sumner	Dawn Meekhof	Brian Kurkoski
Jim Bell	Deepi Brar	Fred Mallon	Dion Hawkins	Carmen Shepard
Ken Hahn	Di McMahon	Harold Zatz	Ed Llewellyn	Dana Pillsbury
Ken Wong	Ed Naranjo	Hoyt Hudson	James Daniel	Harry Catrakis
Tom Wahl	Erik Hille	Jack Profit	Jerry Carter	Jesus Mancilla
Viola Ng	Eugene Lit	Jared Smith	Josh Partlow	Kurth Reynolds
Alice Lee	Francis Ho	Joe Andrieu	Kevin Archie	Lee Hesseltine
BG Girish	Greg Harry	John Beahan	Laura Ekroot	Lisa Henderson
Ben Smith	Harvey Liu	John Butman	Lieven Leroy	Matthias Blume
Betty Pun	Irene Chen	Jung Carl Im	Lynne Hannah	Nancy Drehwing
Castor Fu	James Shih	Ken Andrews	Marc Reissig	Noam Bernstein
Chen Yuan	Jay Higley	Lynn Salmon	Mark Vaughan	Pam Katz Rosten
Chris Chu	Joe Shiang	Marc Turner	Mike Maxwell	Peter Ashcroft
Dan Kegel	John Houde	Mark Berman	Mike Pravica	Randy Levinson
Dan Ruzek	Karen Ross	Max Baumert	Nathan Inada	Richard Murray
Danny Chu	Katy Quinn	Mike Bajura	Shawn Hillis	Scott McCauley
Dave Cole	Ke-Ming Ma	Mike Nygard	Tim Hochberg	Stefan Marelid
Dave Hull	Ma Blacker	Mike Rigler	Tim Horiuchi	Stephanie Buck
Davy Hull	Mark Dinan	Mike Serfas	Tom Aldcroft	Andrew Lundsten
Doug Gray	Matt Tyler	Mike Yamada	Torry Lawton	Bengt Magnusson
Ed Nanale	Mike Nolan	Mimi Zucker	Tracy Ollick	Eric Hassenzahl
Fred Wong	Mike Ricci	Niel Brandt	Vincent Chow	Glenn Eyechaner
Greg Kopp	Minei Chou	Paul Amadeo	Alex Densmore	GloryAnne Yango
Jinha Kim	Pa Blacker	Pete Dussin	Barbara Chang	Imran Kizilbash
John Hart	Ray Bambha	Pete McCann	Betina Pavri	Mac Rhinelander
John Wall	Rob Padula	Pete Wenzel	Chris Cusanza	Margaret Carter
Kevin Kan	Steve Bard	Pui-Tak Kan	Deron Walters	Mike Guadarrama
Marc Abel	Steve Hwan	Randy Baron	Frank Lowther	Nikola Pavletic
Mark Huie	Susan Sheu	Rich Zitola	James Okamoto	Roderic Schmidt
Matt Kidd	Ted George	Rob Fatland	Jen Messenger	Simon Goldstein
Matt Penn	Ted Rogers	Robert Horn	John Erickson	Christina Garden
Mike Chwe	Vince Chen	Robert Lord	Ken Poppleton	Graham Macnamara
Peter Cho	Alex Rosser	Ron Goodman	Lushalan Liao	Jennifer Johnson
Po King Li	Allen Price	Shubber Ali	Margi Pollack	Mike Bronikowski
Steve Lew	Andy Miller	Steve Gomez	Mark Montague	Samantha Seaward
Tom Fiola	Anna George	Steve Hawes	Marty O'Brien	Sandra Blumhorst
Tom Nolan	Ben Holland	Sylvia Chin	Mike McDonald	Susan Sharfstein
Allan Wong	Bill Greene	Todd Walker	Philip Nabors	Konstantin Othmer
Andrew Hsu	Bill Newman	Tom Megeath	Randy Pollock	Suman Chakrabarti
Andy O'Dea	Blake Lewis	Tony Bladek	Rodney Kinney	Celina Mikolajczak
Arthur Ang	Brian Adams	Wurzel Keir	Rorke Haining	Marcos Phoniadakis
Ben Discoe	Chad Nelson	Alan Kulawik	Sue Hannaford	Bibi Jentoft-Nilsen

Acknowledgments

Throughout my many years as a graduate student, I have been fortunate to receive support and assistance from many people and organizations. This opportunity to publicly acknowledge their contributions is welcome.

First, I thank Professor Geoffrey Fox, my thesis advisor, for his encouragement and guidance during the long undertaking of this research project. During my tenure, he has provided me with an excellent working environment with the freedom to pursue topics that interest me.

Secondly, I thank Professor Thomas Prince, for stepping in as my advisor in the interim after Professor Fox's departure from Caltech. I look forward to continuing this relationship.

The work described in this thesis was the result of a fruitful collaboration. I certainly could not have done it without Peter Quinn, Wojciech Zurek, Craig Hogan and especially Mike Warren, who spent many, many hours helping with the development, testing, debugging and running of the parallel BH code.

I am indebted to Joshua Barnes for making his implementation of the algorithm available, in source code form, to the astrophysics community, myself included.

None of this work would have been possible without the staff which created and maintained a superb computational environment. Thanks to Chip Chapman, Heidi Lorenz-Wirzba, Mark Beauchamp, Charles DeBardeleben, Hiram Hunt, Doug Freyberger. I would also like to thank Mary Maloney and Terry Arnold for keeping the red tape at bay.

Many others have helped and inspired me. I am grateful for their friendship, as well as their direct contributions. Thank you Jon Flower, Adam Kolawa, Mark Muldoon, Shih-Chyuan Huang, Mark Johnson, Sean Callahan, Roy

Williams, Steve Otto, David Walker, Jeff Goldsmith, Sun Ra, Paul Stolorz, Clive Baillie and James Kuyper.

Throughout my stay at Caltech, I have been the recipient of generous financial support. The Shell Foundation supported me with a graduate fellowship. In addition, this work was supported by the National Science Foundation, and the Departement of Energy.

Very special thanks must go to the members of Blacker House for "engaging me in varied and interesting conversation."

Finally, I would like to thank my wife, Lynn, for her love, support, patience, and understanding. Words cannot express how grateful I am for her stubborn but helpful insistence that I really could complete this dissertation.

Abstract

Recent algorithmic advances utilizing hierarchical data structures have resulted in a dramatic reduction in the time required for computer simulation of N-body systems with long-range interactions. Computations which required $O(N^2)$ operations can now be done in $O(N \log N)$ or $O(N)$. We review these tree methods and find that they may be distinguished based on a few simple features.

The Barnes-Hut (BH) algorithm has received a great deal of attention, and is the subject of the remainder of the dissertation. We present a generalization of the BH tree and analyze the statistical properties of such trees in detail. We also consider the expected number of operations entailed by an execution of the BH algorithm. We find an optimal value for m, the maximum number of bodies in a terminal cell, and confirm that the number of operations is $O(N \log N)$, even if the distribution of bodies is not uniform.

The mathematical basis of all hierarchical methods is the multipole approximation. We discuss multipole approximations, for the case of arbitrary, spherically symmetric, and Newtonian Green's functions. We describe methods for computing multipoles and evaluating multipole approximations in each of these cases, emphasizing the tradeoff between generality and algorithmic complexity.

N-body simulations in computational astrophysics can require 10^6 or even more bodies. Algorithmic advances are not sufficient, in and of themselves, to make computations of this size feasible. Parallel computation offers, *a priori*, the necessary computational power in terms of speed and memory. We show how the BH algorithm can be adapted to execute in parallel. We use orthogonal recursive bisection to partition space. The logical communication structure that emerges is that of a hypercube. A local version of the BH tree is constructed in each processor by iteratively exchanging data along each edge of the logical hypercube.

We obtain speedups in excess of 380 on a 512 processor system for simulations of galaxy mergers with 180000 bodies. We analyze the performance of the parallel version of the algorithm and find that the overhead is due primarily to interprocessor synchronization delays and redundant computation. Communication is not a significant factor.

Table of Contents

Acknowledgments iv

Abstract vi

Table of Contents viii

List of Figures xii

1 Hierarchical Techniques and N-body Simulations 1

1.1 Categories of N-body simulations. 2

1.1.1 PP methods. 2

1.1.2 PM methods. 4

1.1.3 PPPM methods. 5

1.2 Tree methods 6

1.2.1 Appel's method. 6

1.2.2 Barnes and Hut's method 11

1.2.3 Greengard's method. 16

1.2.4 An illustrative example. 18

1.2.5 Other tree methods. 21

1.3 Categorization of tree methods. 23

1.4 Outline of dissertation. 24

2 Properties of the BH Tree. 27

2.1 Building the tree. 27

2.2 Notation. 33

2.3 Expected number of internal cells. 35

2.4 Probability that a cell is terminal. 36

2.5 Expected number of terminal cells. 37

2.6 Expected population of terminal cells. 38

2.7 Average depth of bodies. 40

2.8 Uniform distribution, i.e., $\bar{\rho}(x) = const.$ 44

2.9 Non-uniform distribution, i.e., $\bar{\rho}(x) \neq const.$ 51

2.9.1 Large N. 51
2.9.2 Bounds on C_{avg}. 52
2.9.3 Bounds on D_{avg}. 53
3 Multipole Expansions. 56
3.1 The multipole expansion. 56
3.2 General case: arbitrary Green's function. 57
3.3 Spherically symmetric Green's function. 58
3.4 A softened Newtonian potential. 62
3.5 Pure Newtonian potential. 64
4 Practical Aspects of the Barnes-Hut Algorithm. 68
4.1 Space requirement for storing multipoles. 68
4.2 Computing multipoles. 69
4.2.1 Direct summation. 69
4.2.2 Parallel axis theorem. 70
4.2.3 Hybrid algorithm. 72
4.2.4 Operation count. 74
4.3 Choosing $\vec{r}_\gamma$. 76
4.4 Evaluating $\phi_{\gamma(n)}(r)$ and $\vec{a}_{\gamma(n)}(r)$. 77
4.4.1 Arbitrary Green's function. 77
4.4.2 Spherically symmetric Green's function. 78
4.4.3 Newtonian potentials. 79
4.5 Choice of the terminal size parameter, m. 80
4.6 Running time. 82
4.6.1 A counter-example to the $O(N \log N)$ behavior. 83
4.6.2 Interaction volume. 84
4.6.3 Number of `BCInteractions`. 88
4.6.4 Number of `BBInteractions`. 89
4.6.5 Uniform distribution, i.e., $\bar{\rho}(x) = const.$ 92
4.6.6 Non-uniform distribution, i.e., $\bar{\rho}(x) \neq const.$ 98
4.7 Number of dimensions not equal to three. 102

5 Analysis of Errors in the BH Algorithm. 104
5.1 Error analysis. 104
5.1.1 Arbitrary Green's function. 106
5.1.2 Spherically symmetric Green's function. 108
5.1.3 A softened Newtonian potential. 109
5.1.4 Newtonian potential. 112
5.2 Statistical analysis of errors. 129
5.2.1 Uniform distribution in a sphere. 133
5.2.2 Uniform distribution in a cube. 134
6 Opening Criteria. 146
6.1 General form of the opening criterion. 146
6.2 Opening criteria in the literature. 148
6.3 The detonating galaxy pathology. 149
6.4 The cause of the problem. 153
6.5 Comparing opening criteria. 154
6.6 Avoiding the detonating galaxy pathology. 162
6.7 Optimizing the opening criterion computation. 163
7 Parallel Implementation of the BH Algorithm. 167
7.1 Domain decomposition, orthogonal recursive bisection. 169
7.1.1 Hypercube communication pattern. 172
7.1.2 Estimating work loads and bisection. 172
7.1.3 Choosing between $x > x_{split}$ or $x < x_{split}$. 176
7.1.4 Choosing which Cartesian direction to split. 178
7.2 Building a "locally essential" tree. 184
7.2.1 Data types for the parallel `BuildTree`. 188
7.2.2 Control structures for the parallel `BuildTree`. 191
7.3 The DomainOpeningCriterion function. 207
8 Performance in Parallel. 212
8.1 Definition of terms. 212
8.1.1 Computational overheads. 213

8.1.2 Memory overhead. 218
8.2 Performance data for the N-body program. 220
8.2.1 Time for a single timestep, T_{step}. 222
8.2.2 Estimating $T_{oneproc}$ and $T_{intrinsic}$. 222
8.2.3 Speedup and overhead. 227
8.2.4 Single-processor data requirement, $D_{oneproc}$. 231
8.2.5 Memory overheads, M_{copy} and M_{imbal}. 231
8.2.6 Natural grain size, q. 234
8.2.7 Complexity overhead, f_{cplx}. 240
8.2.8 Waiting overhead, f_{wait}. 246
8.2.9 Communication overhead, f_{comm}. 250
8.2.10 Load imbalance overhead, f_{imbal}. 253
8.3 Summary. 255
A Proof of bounds on C_{avg} and D_{avg}. 256
A.1 Bounds on C_{avg}. 256
A.1.1 Proof of Lemma 1. 257
A.1.2 Proof of Lemma 2. 260
A.1.3 Proof of Lemma 3. 262
A.1.4 Proof of Theorem 264
A.2 Bounds on D_{avg} 266
B Shell Formation Using 180k Particles... 268
C Primordial Density Fluctuations... 272
D Rotation of Halos in Open and Closed Universes... 279
E Correlation of QSO Absorbtion Lines... 296
F Cubix 309
References 317

List of Figures

1.1 An example of Appel's hierarchy of clumps. 8

1.2 The binary tree representation of an Appel hierarchy. 9

1.3 Expanded representation of a BH tree. 13

1.4 Flat representation of a BH tree 15

1.5 Representation of the hierarchy of levels in Greengard's method. . . 17

1.6 Earth-Sun system. 19

2.1 Iterative construction of BH tree. 29

2.2 Iterative construction of BH tree. 30

2.3 Iterative construction of BH tree. 31

2.4 Iterative construction of BH tree. 32

2.5 $< Pop >$ vs. λ_γ. 41

2.6 $< Pop > /m$ vs. λ_γ/m. 42

2.7 C_{avg} vs. N. 47

2.8 $\frac{mC_{avg}}{N}$ vs N. 48

2.9 D_{avg} vs. N. 49

2.10 $\frac{m8^{D_{avg}}}{N}$ vs. N 50

4.1 One dimensional pathological BH tree. 85

4.2 Labeling of regions around $\mathcal{V}_\gamma$. 87

4.3 Labeling of regions around $\mathcal{V}_\gamma$. 90

4.4 Boundary effects in estimating p_{mid}. 94

4.5 $\langle N_{bb} \rangle /Nm$ vs. N. 97

5.1 $K_1(\alpha, \beta)$ 117

5.2 $\left|\vec{\nabla} K_1(\alpha, \beta)\right|$ 118

5.3 $K_2(\alpha, \beta)$ 119

5.4 $\left|\vec{\nabla} K_2(\alpha, \beta)\right|$ 120

5.5 $K_4(\alpha, \beta)$ 121

5.6 $\left|\vec{\nabla} K_4(\alpha,\beta)\right|$ . . . 122

5.7 Geometry used in K_n^{sup} computations. . . . 124

5.8 Contour of $K_2^{sup}(r) = 0.01$ . . . 126

5.9 Contour of $K_2^{sup}(r) = 0.1$ . . . 127

5.10 Contour of $K_4^{sup}(r) = 0.1$ . . . 128

5.11 $\left|\left\langle\!\left\langle \phi_{\gamma(4)}(\hat{e}) \right\rangle\!\right\rangle\right|$ . . . 140

5.12 $\left|\left\langle\!\left\langle \vec{a}_{\gamma(4)}(\hat{e}) \right\rangle\!\right\rangle\right|$ . . . 141

5.13 $\left\langle\!\left\langle \phi^2_{\gamma(2)}(\hat{e}) \right\rangle\!\right\rangle^{\frac{1}{2}}$ . . . 142

5.14 $\left\langle\!\left\langle \left|\vec{a}_{\gamma(2)}(\hat{e})\right|^2 \right\rangle\!\right\rangle^{\frac{1}{2}}$ . . . 143

5.15 $\left\langle\!\left\langle \phi^2_{\gamma(3)}(\hat{e}) \right\rangle\!\right\rangle^{\frac{1}{2}}$ . . . 144

5.16 $\left\langle\!\left\langle \left|\vec{a}_{\gamma(3)}(\hat{e})\right|^2 \right\rangle\!\right\rangle^{\frac{1}{2}}$ . . . 145

6.1 A two-galaxy system. . . . 150

6.2 Pathological situation for BH `OpeningCriterion`. . . . 152

6.3 Worst case situation arising from BH `OpeningCriterion`. . . . 155

6.4 Geometry of $\mathcal{V}_{mid}$ with the original BH `OpeningCriterion`. . . . 157

6.5 Geometry of $\mathcal{V}_{mid}$ with the $(edge, Cart)$ `OpeningCriterion`. . . . 158

6.6 F_{OC} vs. θ . . . 160

6.7 Equivalent values of θ for different opening criteria . . . 161

7.1 An ORB domain decomposition. . . . 171

7.2 A one-dimensional decomposition . . . 177

7.3 A Gray code decomposition . . . 177

7.4 Domain Decomposition. . . . 179

7.5 A two-galaxy system. . . . 181

7.6 Memory and processor domain geometry . . . 182

7.7 A two-galaxy system with long-dim splits . . . 183

7.8 Data flow in a 16 processor system. . . . 189

7.9 Illustration of `FindParent` . . . 205

7.10 Illustration of `DomainOpeningCriterion`. 209
7.11 False positive from `DomainOpeningCriterion` 210
8.1 T_{step} vs. Number of processor. 223
8.2 T_{force} vs. N 225
8.3 Speedup vs. Number of processor. 228
8.4 Total overhead vs. N_{grain}. 230
8.5 Copied memory overhead, g_{copy} vs. N_{grain} 232
8.6 Imbalanced memory overhead, g_{imbal} vs. N_{grain} 233
8.7 N_{grain} vs. $max(d_{step})$ 235
8.8 Simplified interaction domain of a processor 236
8.9 Memory/N_{grain} vs. natural grain size, q. 239
8.10 Complexity overhead, f_{cplx} vs. N_{grain}. 241
8.11 Complexity time, T_{cplx} vs. copied memory, M_{copy}. 244
8.12 Complexity overhead, $N_{int}f_{cplx}$ vs. natural grain size, q. 245
8.13 Waiting overhead, f_{wait} vs. N_{grain}. 247
8.14 Empirical fit for waiting overhead, $\frac{f_{wait}*N_{int}}{\log_2(N_{proc})}$ vs. natural grain size, q.249
8.15 Communication overhead, f_{comm} vs. N_{grain}. 252
8.16 Load imbalance overhead, f_{imbal} vs. N_{grain}. 254

1. Hierarchical Techniques and N-body Simulations

Physical systems may be studied by computer simulation in a variety of ways. Simulations with digital computers are constrained by factors associated with the finite speed of computation and the finite size of the computer. Generally, these factors require that the first step in any computer simulation of a physical phenomenon be to develop a mathematical model of the phenomenon consisting of a finite number of discrete parts. The correspondence between the discrete components in the computer simulation and the physical phenomenon itself is completely arbitrary, limited only by the imagination of the scientist.

Often, the discretization is arrived at indirectly, by means of partial differential equations. First, one models the physical system by a set of partial differential equations. Then, one applies techniques of discrete approximation such as finite difference methods or finite element methods to recast the mathematical problem into a discrete form. An alternate, and less widely used, approach is to discretize the physical system into a finite set of "bodies" or "particles" which interact with one another as well as with externally applied "forces." The bodies carry some state information which is modified as the simulation proceeds, according to the interactions. Simulations based on this type of discretization are referred to as "N-body" simulations. Hockney and Eastwood[1] have extensively reviewed the techniques of N-body simulations, as well as applications in plasma physics, semiconductor device simulation, astrophysics, molecular dynamics, thermodynamics and surface physics. Applications also exist in fluid mechanics,[2, 3, 4] applied mathematics,[5] and undoubtedly other areas as well.

In the simplest case, the state information associated with each body may

consist of a position and a velocity, and the interaction between bodies is a model for some kind of force law. The dynamical state of the system is evolved by alternately adjusting the velocities based on accelerations that result from interparticle interactions, and the positions, based on the velocities.

1.1. Categories of N-body simulations.

In their monograph, Hockney and Eastwood[1] classify N-body methods broadly into three categories:

1. Particle-Particle (PP) methods.
2. Particle-Mesh (PM) methods.
3. Particle-Particle Particle-Mesh (PPPM or P^3M) methods.

We briefly review these methods to provide context for the discussion of tree methods which follows.

1.1.1. PP methods.

In PP methods, particles interact directly with one another. The basic control structure, which is repeated over and over as the simulation proceeds, is outlined in Codes 1.1 and 1.2.

```
ComputeAllInteractions
    for(each body, b)
        ComputeInteraction(of b with all other bodies)
    endfor
    for(each body b)
        UpdateInternalState(b)
    endfor
endfunc
```

Code 1.1. Function to compute all pairwise interactions among a set of bodies.

PP methods are particularly useful because of their simplicity. Often, the interaction follows directly from a well understood physical interaction with little or no approximation, e.g., the Newtonian force of attraction between massive

```
ComputeInteraction(b)
    for(each body, b_j ∋ b_j ≠ b )
        PairInteraction(b, b_j)
    endfor
endfunc
```

Code 1.2. Function to compute all interactions with a particular body, b.

bodies. The simplicity of Codes 1.1 and 1.2 allows for straightforward expression in a suitable computer language like C or FORTRAN. The simplicity also allows one to easily take advantage of "vectorization," whereby the full power of modern supercomputers can be efficiently used. Parallelization is also reasonably straightforward,[6] making it possible to use large parallel computing systems which derive their speed and cost-effectiveness from the assembly of numerous modest, autonomous processors into a single parallel system.

The most significant drawback of PP methods is their scaling for large numbers of bodies. An implementation following Codes 1.1 and 1.2 requires each body to interact with every other body in the system. Each time Code 1.1 is executed, the function `PairInteraction` is called $N(N-1)$ times, where N is the number of bodies in the simulation. Even if Code 1.2 is modified to take advantage of a commutative interaction, i.e., one for which `PairInteraction`(b_1, b_2) is equivalent to `PairInteraction`(b_2, b_1), then `PairInteraction` is executed only half as many times. Unfortunately, this does little to mitigate the rapid growth, proportional to N^2.

If the interaction between particles is short-range, then Code 1.2 can be replaced with Code 1.3. The loop in Code 1.3, which is restricted to bodies in a ball of radius r_{cut} around b, requires computation of only N_{neigh} pair interactions, where N_{neigh} is the number of bodies in the ball. For homogeneous systems, N_{neigh} is of $O(1)$, i.e., independent of N. Hockney and Eastwood describe data structures which allow one to select the N_{neigh} neighbors in a ball of fixed radius from the entire set of N bodies in time $O(N_{neigh})$. Thus, the entire calculation of

Code 1.1 requires only $O(NN_{Neigh})$ executions of `PairInteraction`. Of course, if N_{neigh} is very large, this may be little improvement over the the $O(N^2)$ behavior of Code 1.2.

If the interaction does not have a natural distance cutoff, PP methods are limited to situations in which $N < few \times 10^4$.

```
ComputeShortRangeInteraction(b)
    for(each body b_j ∋ Separation(b, b_j) < r_cut)
        PairInteraction(b, b_j)
    endfor
endfunc
```

Code 1.3. An alternative form of `ComputeInteraction`, applicable when the interaction is negligible beyond a distance of r_{cut}.

1.1.2. PM methods.

PM methods offer the advantage of $O(N)$ behavior for large N, even if the interaction cannot be neglected outside of some distance cutoff. However, they introduce approximations which may be problematical. PM methods are applicable when the interaction is expressible as the solution of a differential equation, discretized onto an auxiliary mesh. Electromagnetic and gravitational interactions have this property, with

$$\vec{a} = -\nabla\phi, \tag{1.1}$$

$$\nabla^2\phi = -4\pi G\rho. \tag{1.2}$$

The values of ρ on the mesh are computed from the locations of the bodies. Then, Poisson's Equation, Eqn. 1.2, is solved. Finally, a discrete approximation to the gradient in Eqn. 1.1 is evaluated, and the values of the acceleration, $\vec{a}$, are interpolated at the position of each body. The dominant contribution to the overall time required by PM methods is the solution of Poisson's Equation. Hockney and Eastwood discuss several methods for solution of Poisson's Equation. It is likely that these methods have been superseded in the intervening years by multigrid

methods,[7, 8] which converge in time $O(N_{mesh})$, with a remarkably small constant of proportionality, where N_{mesh} is the number of grid points in the discrete mesh.

Hockney and Eastwood discuss the relationship between N and N_{mesh}. Generally speaking, the two should be roughly comparable. Otherwise, laboriously obtained information is thrown away each time one moves back and forth between the mesh representation and the body representation.

The approximation introduced by the PM method generally results in depressing the strength of short-range interactions. The interaction between any bodies whose separation is less than a few mesh spacings will be significantly reduced from the PP case. In some circumstances, this "error" actually results in the simulation more faithfully representing the physical phenomena, e.g., collisionless plasmas. In any event, PM methods can only model phenomena on length scales larger than a few mesh spacings. Thus, the smallest interesting length scales dictate the required mesh spacing. If the system under study is highly inhomogeneous, with length scales extending over a few orders of magnitude, then the size of the mesh may become prohibitive.

1.1.3. PPPM methods.

Finally, Hockney and Eastwood discuss PPPM methods. These methods attempt to recover some of the accuracy lost by PM methods without reintroducing the $O(N^2)$ behavior of PP methods. In essence, a carefully designed short-range interaction is computed by PP methods, as in Code 1.3. This additional force is added to the usual PM interaction. The short-range component is an analytic approximation to the error introduced by the PM method. Since the error introduced by the PM method is of limited range, i.e., a few mesh spacings, the correction may be computed using Code 1.3 in time $O(NN_{neigh})$.

PPPM methods still encounter difficulty with highly inhomogeneous systems. When the range of length scales is large, one is faced with a choice between a very large mesh, or a very large average value of N_{neigh}. If N_{mesh} is comparable to

N, then N_{neigh} may be a significant fraction of N, leading to $O(N^2)$ behavior, albeit with a much smaller constant of proportionality than simple PP methods. Furthermore, the great simplicity of Code 1.2 which facilitated efficient vectorization and parallelization is lost for PPPM techniques. Although still possible, it requires considerably more effort to vectorize or parallelize a PPPM method.

1.2. Tree methods

Since Hockney and Eastwood's monograph was published, an entirely new class of particle simulation methods has emerged as an alternative to PP, PM or PPPM methods. These methods are characterized by an organization of particles into a hierarchy of clusters, which span the full range of length scales from the minimum interparticle spacing up to the diameter of the entire system. These methods are usually referred to as "tree methods" or "hierarchical methods" because of the data structures which are used.

1.2.1. Appel's method.

Hierarchical data structures were introduced into astrophysical N-body simulations by Appel.[9] It has been known for some time that the gravitational effect of a large group of bodies may often be approximated by the effect of a single object located at the center of mass.[10] Appel realized that one can use this fact to significantly reduce the computational complexity of the gravitational N-body iteration. He claimed to reduce the complexity to $O(N \log N)$, but subsequent work has shown that Appel's algorithm is asymptotically $O(N)$, although the difference may be of purely academic interest for practical values of N.[11] The basic idea is captured by the following example: when calculating the force of the Earth on an apple, it is not necessary to compute the effect of each and every atom on one another. To a very good approximation, we can take both the Earth and the apple to be point masses located at their respective centers-of-mass, and analyze the system as though it contains only two bodies. The details of how to make this approximation uniformly throughout a system of interacting bodies is the subject of the various hierarchical N-body methods that have recently received attention.

In Appel's terminology, bodies are collected into "clumps," which in turn are collected into larger clumps, and so on. The resulting data structure is a binary tree. Of course, there are many binary trees that can be constructed from a collection of bodies. A good tree will collect physically nearby bodies within the same branch. Figure 1.1 illustrates how a region of space might be divided into clumps which represent amorphous regions of space. Figure 1.2 shows the binary tree equivalent to the hierarchy of clumps shown in Figure 1.1. Of course, even given the constraint that clumps should correspond to compact regions of space, there is still a great deal of freedom in the choice of hierarchy. Appel builds the tree in two steps:

1. Build a "k-d tree" with the property that the bodies contained within the two descendants of a given node are separated by a plane parallel to one of the Cartesian axes, and have equal numbers of bodies. In other words, the left child contains all bodies below the median coordinate, and the right child contains all bodies above the median. The coordinate, i.e., x, y or z, alternates at successive levels of the tree.
2. Refine the tree, so that the nearest external clump to any clump is its parent. This is achieved with a local modification procedure that Appel calls a "grab." He points out the effectiveness of the "grab" procedure is difficult to analyze.

Once the tree is built, the acceleration of each clump is calculated by a recursive descent of the tree. Each node of the tree, i.e., each clump, only stores velocities and accelerations relative to its parent. The acceleration of a clump relative to its parent is due solely to the influence of its sibling. Thus, the acceleration of all the nodes in the tree can be computed by applying the recursive procedure, `ComputeAccel`, shown in Code 1.4, to the root of the tree.

So far, we have not made use of the approximation that well-separated clumps can be computed by treating them as point-masses. The subroutine `Interact`, shown in Code 1.5, uses an adjustable parameter, δ, and makes the point-mass approximation whenever the diameter of the larger of two clumps exceeds δ times the separation.

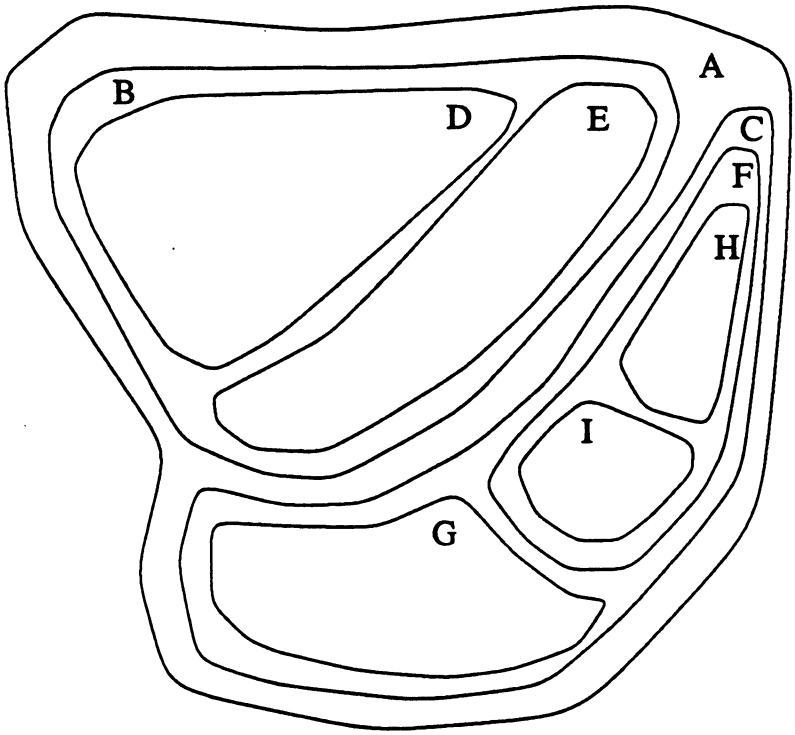


Figure 1.1. An example of Appel's hierarchy of clumps.

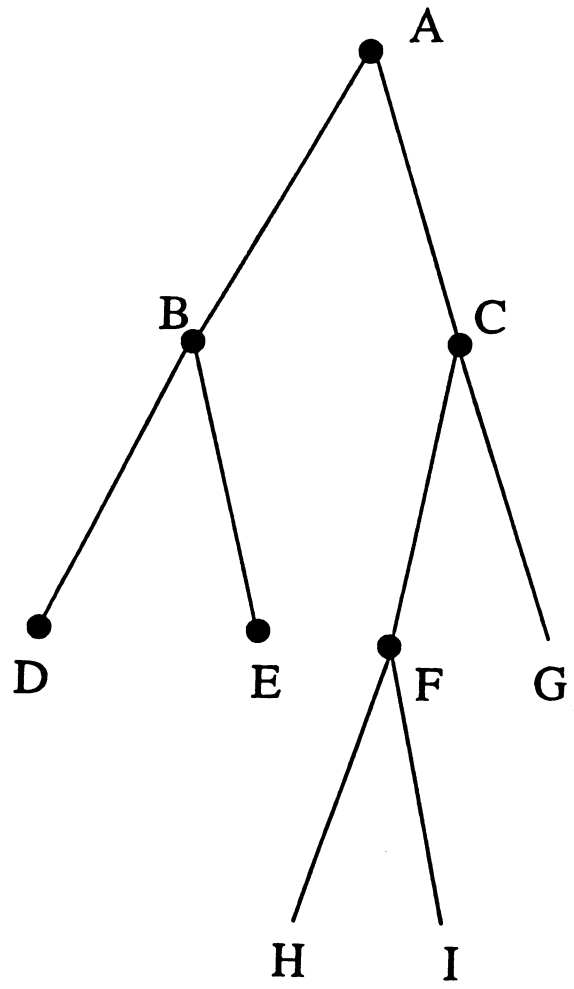


Figure 1.2. A binary tree equivalent to the collection of clumps in Figure 1.1.

```
ComputeAccel(node)
    ComputeAccel(RightChild(node))
    ComputeAccel(LeftChild(node))
    Interact(RightChild(node), LeftChild(node))
endfunc
```

Code 1.4. Appel's algorithm for computing the acceleration of a node in a tree.

```
Interact(A, B)
    Larger = larger of A and B
    Smaller = smaller of A and B

    if( Diameter(Larger) > δ * Separation(A, B) )
        Interact(RightChild(Larger), Smaller);
        Interact(LeftChild(Larger), Smaller);
    else
        Monopole(Larger, Smaller);
```

Code 1.5. Appel's algorithm for computing the interaction between two nodes in a clustering hierarchy. If the nodes are sufficiently well separated, then a monopole approximation is used. Otherwise, the interactions between their components are individually computed.

The parameter δ may be freely varied between 0 and 1. When δ equals zero, every pair of bodies interacts, in much the same way as in Code 1.2. For fixed, non-zero values of δ, Appel estimates the number of evaluations of the subroutine `Monopole` to be $O(N \log N)$, but Esselink [11] has shown that `Monopole` is actually executed only $O(N)$ times. Esselink's arguments are rather involved. The following simple discussion illustrates why Appel's algorithm is $O(N)$:

1. The nature of the recursion in Code 1.5 guarantees that clumps which interact via `Monopole`, are approximately the same size. The reason is that the recursive call to `Interact` always subdivides the larger of the two clumps. Its two daughters are unlikely to be very different in size from one another, or

from the smaller sibling.

2. Consider a particular clump, C. The clumps with which it interacts via `Monopole` are more distant than its size times δ^{-1}, and less distant than its size times $2\delta^{-1}$. If they were more distant than $2\delta^{-1}$, then they would have interacted with C's parent.

Assuming these two premises are valid, we conclude that the volume available for the clumps with which C interacts is limited to some fixed multiple of C's volume, while the clumps themselves are not smaller than some other fixed multiple of C's volume. Thus, there can be at most, $O(1)$ such clumps, independent of N. Since there are $2N-1$ clumps in a binary tree with N terminal nodes, and each is an argument to `Monopole` only $O(1)$ times, the total number of calls to `Monopole` must be $O(N)$.

Appel's algorithm has not been widely adopted. Appel himself recognized that the imposition of a hierarchical data structure on the physical system, and its associated approximations, might lead to non-physical, hierarchical artifacts in the final result. This would be highly undesirable in circumstances where the purpose of the simulation is to study the evolution of some physical clustering phenomenon, e.g., the growth of structure in the universe.[12] Furthermore, the somewhat chaotic and tangled structure of the binary tree after the application of the "grab" procedure makes the accuracy of the method difficult to analyze precisely. Finally, the restriction to the very crude monopole approximation requires a low value of δ, and a large number of executions of `Monopole` in order to achieve acceptable accuracy. It is possible, however, to adapt Greengard's[13] or Zhao's[14] formalism for higher order multipoles into Appel's framework.[15]

1.2.2. Barnes and Hut's method

Barnes and Hut's (BH) algorithm[16, 17] differs from Appel's in several important respects. First, the tree data structure is significantly different. Each internal cell in the tree corresponds to a cube in physical space, and has up to eight immediate descendants, corresponding to the eight smaller cubes obtained by splitting the

larger one in half along each of the three coordinate axes. In Appel's terminology, each such cube represents one "clump," with size given by the length of an edge. We shall call these cubes "cells." In addition, the tree structure is recomputed, *ab initio* for each iteration; there is nothing analogous to the "grab" procedure.

The second important difference is that BH compute accelerations only for bodies. The internal nodes of the tree are not dynamical objects that influence one another. They are merely data structures used in the computation of the forces on the bodies. In fact, it is possible to compute accelerations for arbitrary test bodies; even ones which are not included in the tree.

Finally, BH suggest, and Hernquist[18] elaborates upon the possibility of including quadrupole and higher order terms in the force calculation.

An example of the tree used by BH is shown in Figure 1.3. For simplicity, we shall usually render trees and other data structures as though we were considering two-dimensional simulations. The generalization to three dimensions is obvious. The figure makes clear the hierarchical nature of the structure, with each internal "node" or "cell" having exactly four descendants, each of exactly half the linear size of the parent. In three dimensions, of course, each cell has eight descendants. The root of the BH tree corresponds to the entire computational volume, i.e., it encompasses all of the bodies in the N-body simulation. As we shall see, the terminal nodes correspond to regions that contain one body. Thus, the BH tree contains groupings of objects over the entire range of length-scales present in the simulation. The tree is "adaptive" in the sense that it naturally extends to more levels in regions with high particle density and short length-scales.

Figure 1.4 shows a flattened representation of the same tree as Figure 1.3. In Figure 1.4 it is clear that the terminal nodes of the BH tree completely fill the computational volume with cubical (square) cells whose linear size is an integer power of $\frac{1}{2}$ times the full size of the volume. We shall use representations like Figure 1.4 repeatedly. It should be borne in mind that Figure 1.4 is merely shorthand for Figure 1.3. The internal nodes that are apparent in Figure 1.3 are always present. It is only the graphical representation of Figure 1.4 which obscures their

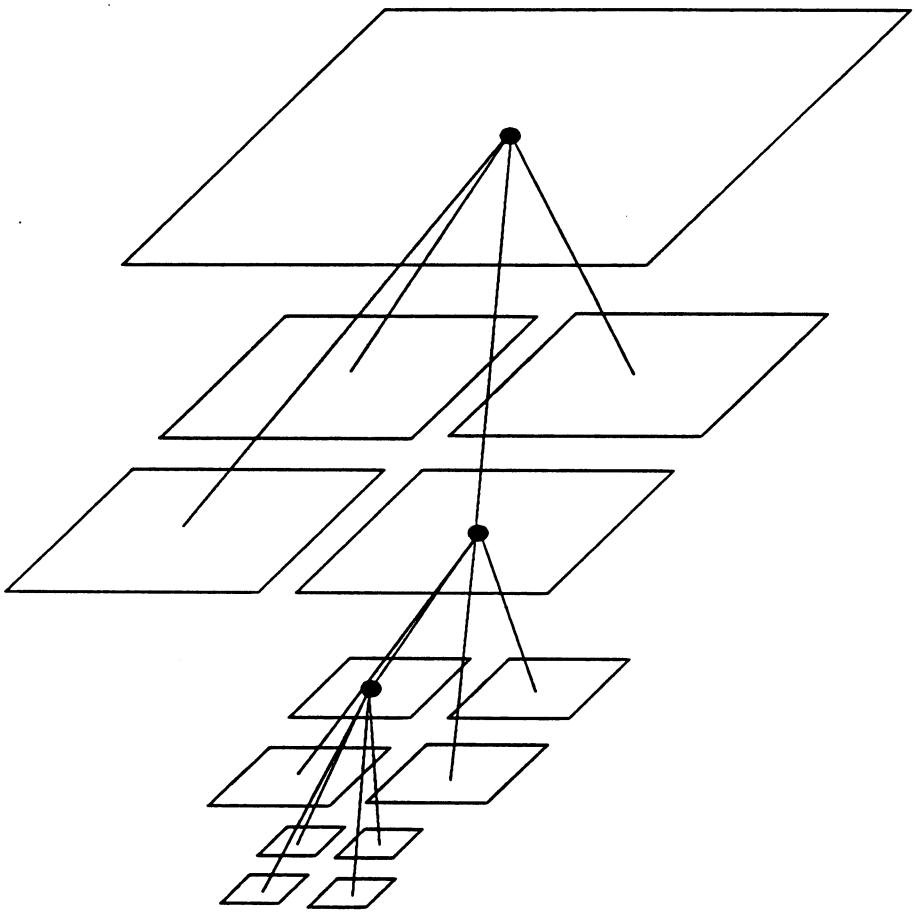


Figure 1.3. Expanded representation of a BH tree.

presence.

The BH algorithm proceeds as follows:

1. Build a BH tree with the property that each terminal node has exactly zero or one body within it.
2. Compute the mass and center-of-mass of each internal cell in the tree. Record this information so that each internal cell may be treated in Code 1.6 as a body with a mass and position.
3. For each body, traverse the tree, starting at the root, using the procedure in Code 1.6.

```
ComputeField(body, cell)
    if( cell is terminal )
        Monopole(body, cell)
    else if( Distance(cell, body)  > θ * Size(body))
        for( each child of cell )
            ComputeField(body, child)
        endfor
    else
        Monopole(body, cell)
    endif
endfunc
```

Code 1.6. Function `ComputeField` computes the interaction between a specified body, and all bodies lying in a cell.

As we shall see in Chapter 4, when ComputeField is called for every body in the tree, `Monopole` is executed $O(N \log N)$ times.

The BH algorithm can be generalized in several ways.

1. The tree can be constructed with a terminal-size parameter, m. Terminal nodes are required to have m or fewer bodies. BH consider $m = 1$. We consider the effect of $m \neq 1$ in Chapters 2 and 4.
2. The distribution of bodies within a cell can be modeled more accurately than as a point mass at the center-of-mass. It is common practice to compute the quadrupole moment and to replace the final call to `Monopole` in Code 1.6 with

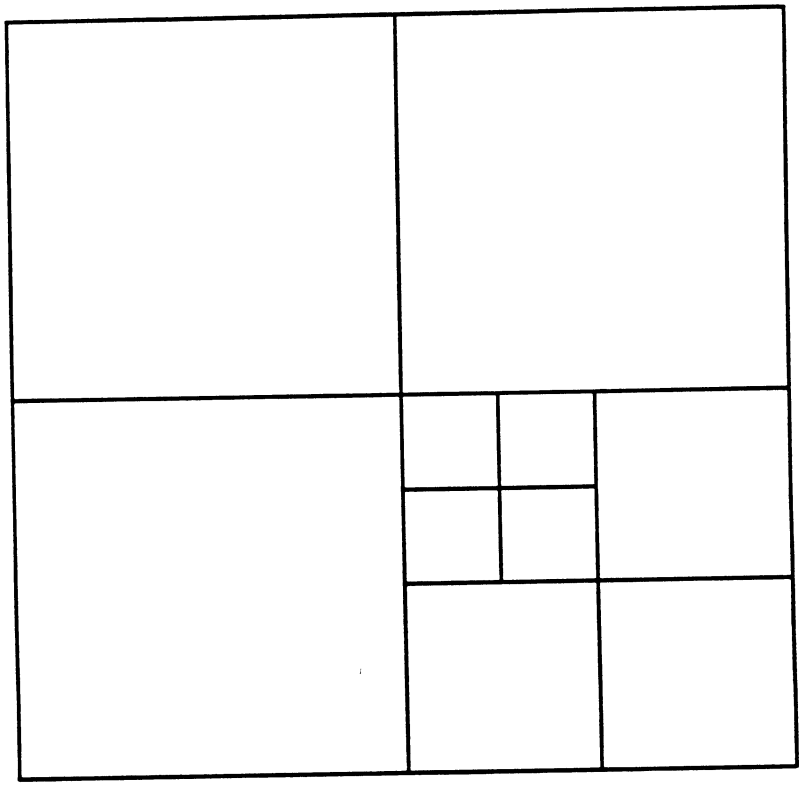


Figure 1.4. Flat representation of the same BH tree as in Figure 1.3.

a function that computes both monopole and quadrupole interactions.[18, 17, 19, 20]

3. The form of the interaction need not be Newtonian gravity. In Chapters 3, 4 and 5, we consider the applicability of the BH algorithm to other forms of interaction.
4. The form of the opening criterion, i.e., the predicate which determines whether the multipole approximation will be used for a given body and cell, may be modified. In addition to adjusting the parameter θ, one can contemplate alternatives which are both faster and more accurate. This topic is considered in Chapter 6.
5. It is possible to reformulate the algorithm to take advantage of "vector" supercomputers.[21, 22, 20]
6. The BH algorithm can be reformulated to take advantage of parallel computer hardware. This is discussed in Chapters 7 and 8.

1.2.3. Greengard's method.

Another algorithm has been introduced by Greengard and Rokhlin.[13, 23, 24] Greengard combines feature of Barnes' as well as Appel's formulation, and develops an extensive formalism based on spherical harmonics. As in Appel's method, accelerations are computed between internal nodes of the tree, leading to an overall time complexity of $O(N)$. On the other hand, the cells in the hierarchy are cubes which fill space completely, as in the BH method. Greengard has presented his algorithm in a static, rather than an adaptive form. The hierarchy in Greengard's method is a set of grids, each with twice as many cells in each direction as the layer above it. The structure is shown in Figure 1.5. With a representation of the data as in Figure 1.5, Greengard's method is not adaptive. The finest level of the hierarchy samples the system at the same level of detail, throughout the computational volume.

Greengard's method does not have a tunable parameter like θ that controls the descent of the tree or the number of interactions computed. In three dimensions,

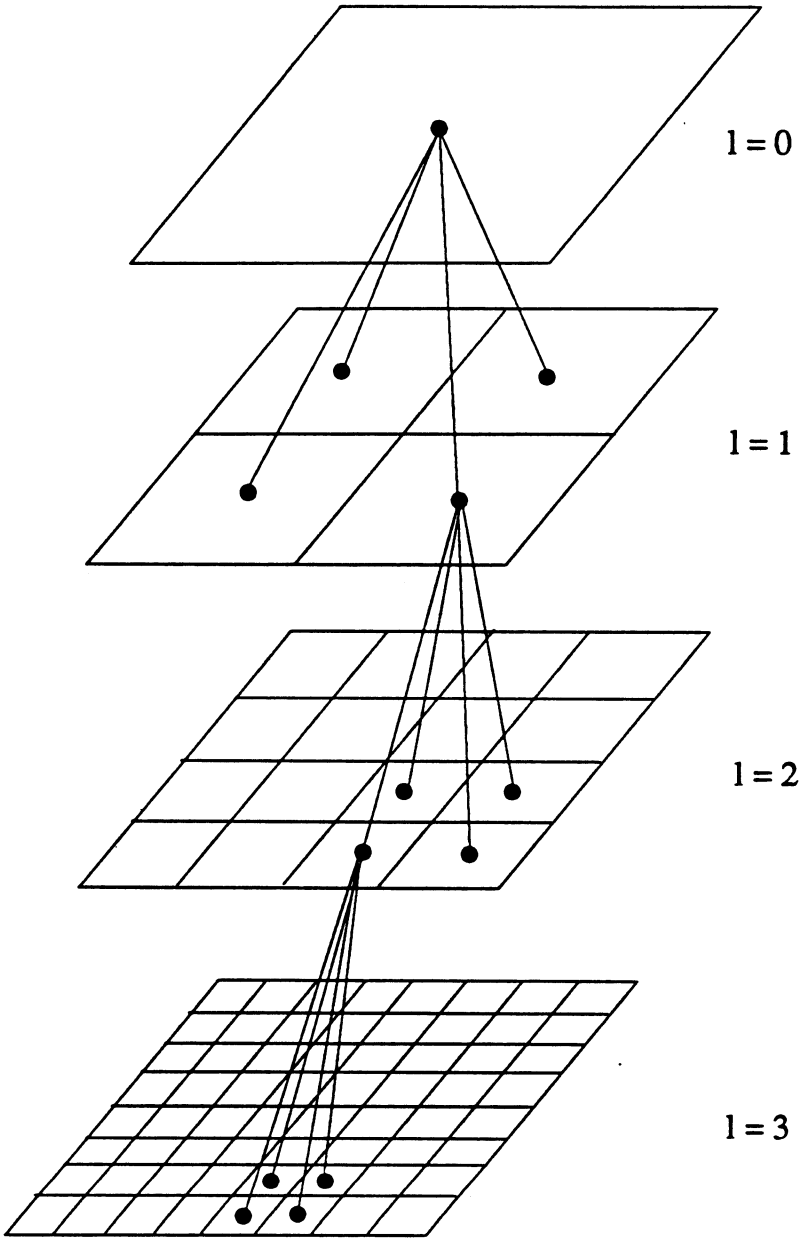


Figure 1.5. Representation of the hierarchy of levels in Greengard's method.

each cell interacts with $875 = 10^3 - 5^3$ nearby cells at the same level of the tree. Instead, Greengard controls errors by expanding the mass distribution in each cell to a fixed number of multipoles. He makes an extremely strict accounting of the errors introduced by approximating a distribution by a 2^p-pole multipole expansion, and concludes that the fractional error, ϵ, is bounded by

$$\epsilon < 2^{-p}. \tag{1.3}$$

Hence, he concludes that to obtain results accurate to within ϵ, one should carry out the expansion with

$$p \geq \lceil -log_2 \epsilon \rceil. \tag{1.4}$$

Greengard develops an extensive formalism for manipulating coefficients of spherical harmonics. The basic "interaction" in Greengard's algorithm consists of the translation of a multipole expansion from one center to another. For all but the simplest situations (which corresponds to the `Monopole` clump-clump interactions in Appel's algorithm) this is an extremely complex operation, requiring evaluation of high-order spherical harmonics and some p^4 floating point operations. Thus, despite its extremely promising $O(N)$ asymptotic behavior, the constant of proportionality which relates time to N is very large. Greengard estimates that some $875p^4N$ operations (some of them evaluations of high-order spherical harmonics) are required to compute potentials for N bodies. Greengard does not discuss the additional complexity needed to compute forces, except to remark that the gradients of the spherical harmonics may be computed analytically.

1.2.4. An illustrative example.

We have now briefly reviewed three hierarchical N-body algorithms currently in use. To illustrate the differences between these algorithms, consider how each would calculate the gravitational effect of the Sun on the Earth. Specifically, imagine that both the Earth and Sun are modelled as roughly spherical clouds each containing several thousand bodies. The geometry is shown in Figure 1.6.

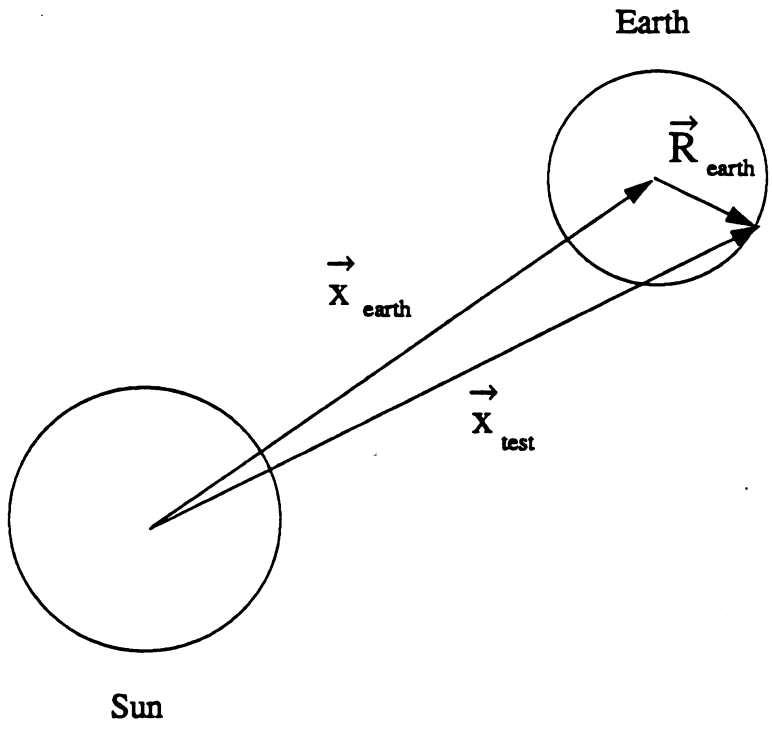


Figure 1.6. Earth-Sun system.

Appel would consider both the Earth and Sun to be "clumps" located at their respective centers-of-mass. The acceleration of all of the Earth bodies (neglecting interactions between pairs of Earth bodies) would be the same, given by

$$\vec{a} = -\frac{GM_{sun}\vec{x}_{earth}}{|x^3_{earth}|}. \tag{1.5}$$

In particular, there would be no tides, which arise from the difference between the Sun's gravitational attraction on the daytime and nighttime sides of the Earth.

If tides are important, Appel would recommend decreasing δ so that,

$$Diameter(Earth) > \delta Separation(Earth, Sun), \tag{1.6}$$

in which case the Earth would be subdivided into two or more sub-clumps before the monopole approximation was made. This would also have the effect of requiring subdivision of the Sun into many earth-sized clumps, each of which would require its own monopole interaction. Of course, there is no guarantee that the Earth would be split into the "correct" sub-clumps. They are as likely to be the Northern and Southern Hemispheres as to be Night and Day. Thus, the only safe procedure is to set

$$\delta \ll Diameter(Earth)/Separation(Earth, Sun). \tag{1.7}$$

Such a small value of δ, would, of course imply a very large number of executions of `Monopole`.

Now consider the BH method. In this case, the Sun would be treated as a point-mass at its center of mass. Each of the Earth bodies would be treated separately, with the monopole approximation being evaluated separately for each one. Tides raised by the Sun would be treated correctly because bodies on the nighttime side of the Earth would feel less force due to their larger distance from the center of the Sun. The correct behavior of solar tides is obtained with relatively little effort.

Finally consider Greengard's method. Again the mass distribution of the Sun is approximated, but now many terms (perhaps up to 2^{10}-pole) in the multipole expansion are retained. Of course, since the Sun is almost spherical, the magnitudes of most of the high-order terms would be very small. The field from the Sun is "translated" to the center of the Earth resulting in a new multipole expansion about the center of the Earth. The familiar solar tides arise from the contribution of the Sun's monopole term to the quadrupole term in the expansion about the center of the Earth. If we consider only the monopole term from the sun, then Greengard's formalism tells us

$$\begin{aligned}\phi(x_{test}) = &-\frac{GM_{sun}}{|\vec{x}_{earth}|}\sqrt{4\pi}Y_0^0(\theta,\phi) + \frac{GM_{sun}}{|vxe|^2}|R_{earth}|\sqrt{\frac{4\pi}{3}}Y_1^0(\theta,\phi) \\ &- \frac{GM_{sun}}{|\vec{x}_{earth}|^3}|R_{earth}|^2\sqrt{\frac{4\pi}{5}}Y_2^0(\theta,\phi) + \cdots,\end{aligned} \tag{1.8}$$

where Y_l^m are the spherical harmonics, θ is the angle between $\vec{R}_{earth}$ and $\vec{x}_{earth}$. The azimuthal angle, ϕ, happens to be irrelevant in this case. Greengard does not compute explicit error bounds for the gradient of ϕ, but we may compute it analytically as:

$$\begin{aligned}-\vec{\nabla}\phi(x_{test}) = &-\frac{GM_{sun}}{|\vec{x}_{earth}|^2}\vec{\nabla}\left(|R_{earth}|\sqrt{\frac{4\pi}{3}}Y_1^0(\theta,\phi)\right) \\ &+ \frac{GM_{sun}}{|\vec{x}_{earth}|^3}\vec{\nabla}\left(|R_{earth}|^2\sqrt{\frac{4\pi}{5}}Y_2^0(\theta,\phi)\right) - \cdots,\end{aligned} \tag{1.9}$$

The first term in Eqn. 1.9, gives rise to the usual acceleration of all Earth particles uniformly toward the Sun, while the second gives the tidal variations, which depend on the angle θ.

1.2.5. Other tree methods.

There are a number of other hierarchical N-body methods in the literature. In this section, we provide a very brief overview of them.

Jernigan and Porter[25, 26, 27] describe a method which treats the hierarchy as a dynamical object, in which the velocity and acceleration of a node in the tree is always with respect to its parent. They use a binary tree similar to Appel's. In order to improve on the accuracy of Appel's method, they transform the equations of motion for each pair of siblings into a "regularized" form.

Pépin, Chua, Leonard, and Winckelmans[15, 28] have described yet another hierarchical method. The simulation of interacting "vortex particles" as models of fluid flow shares the same long-range difficulties as gravitational N-body simulations. They use a binary tree, similar to that used by Appel, and a multipole expansion similar to Greengard's two-dimensional formalism. Pépin has shown that the method can be parallelized, achieving good efficiency on up to 32 processors.[15] This work specifically treats two-dimensional problems, for which Greengard's formalism is considerably simplified. In principle, three-dimensional problems may be treated by adopting either Greengard's or Zhao's formalism for translation of three-dimensional multipoles.

Zhao[14] has described an $O(N)$ algorithm that is very similar in structure to Greengard's method. The difference lies in the representation of the multipole expansion. Where Greengard uses a formalism based on spherical harmonics, Zhao uses a formalism based on Taylor series and Cartesian coordinates. Otherwise, the structure of the algorithms is identical. Zhao and Johnsson[29] have demonstrated that the algorithm may be adapted for execution on the Connection Machine, a SIMD system with 65536 single-bit processors and 2048 floating point units. Their results are only for uniform distributions of bodies. They do not consider problems of load balancing or implementation of adaptive hierarchies.

Katzenelson[30] presents a unified framework which encompasses both Greengard's algorithm and a non-adaptive version of Barnes' algorithm, as well as an outline for implementation on the Connection Machine. He also does not consider the problem of load balance or the need for adaptive structures when the distribution of bodies in space is highly non-uniform.

Benz, Bowers, Cameron and Press[31, 32, 33] have proposed yet another

method with similarities to Appel's method and BH. As in Appel's method, the hierarchy is constructed as a binary tree. The construction is slightly different, however, based on a technique for finding "mutual nearest neighbor" pairs. This technique seems to eliminate some of the uncertainties associated with Appel's "grab" procedure. The force calculation now proceeds in a manner very similar to the BH algorithm. The tree is traversed once for each body, applying an opening criterion at each node which determines whether the quadrupole approximation is adequate, or, conversely, whether the interaction must be computed with the daughters of the node. The method requires $O(N \log N)$ executions of the basic quadrupole interaction subroutine to compute forces on all bodies. Makino [20] has found that the force evaluation is approximately as time-consuming as the BH algorithm, for a given level of accuracy, but the construction of the tree takes roughly ten times as long.

1.3. Categorization of tree methods.

Obviously, a large number of tree methods have been suggested, with diverse scalings, data structures, mathematical foundations, etc. Nevertheless, many also share features in common. Table 1.1 categorizes the different methods according to the following criteria:

Tree type: The methods use either binary trees or octrees. The octrees are generally easier to construct, as they do not require searching for neighbors, computing medians, etc. Since they do not impose any external rectangular structure, the binary trees may introduce fewer artifacts into the simulation. There is little evidence one way or the other.

Multipoles formalism: Methods make use of monopole, quadrupole or arbitrarily high order multipoles. When high order methods are used, they are either based on spherical harmonics or Taylor series expansions.

Adjustable opening criterion: Some methods have an adjustable parameter, e.g., θ, which controls the opening of nodes in the tree during traversals.

Adaptive: A method is considered adaptive if the hierarchy extends to deeper

levels in regions in which bodies are present with high spatial densities. In general, those methods which are not adaptive can be made so by altering the fundamental data structure from an "1-D array of 3-D arrays" to an octree.

Scaling: The methods are either $O(N \log N)$ or $O(N)$. As $\log N$ grows so slowly the difference between the asymptotic speeds for very large N may be of little practical significance. For practical values of N, the "constants" are at least as important as the functional form of the scaling.

Interaction type: Some of the methods compute interactions between nodes of the hierarchy, while others only compute interactions between isolated bodies and nodes of the hierarchy. All methods compute some interactions between pairs of bodies. The methods which allow nodes to interact have a scaling behavior of $O(N)$, while those which have only body-node interactions scale as $O(N \log N)$. Generally, node-node interactions are extremely complex. We distinguish between the different algorithms based on the most complex type of interaction allowed, i.e., either body-body (B-B), body-node (B-N) or node-node (N-N).

Table 1.1. Classification of tree methods according to criteria described in the text.

	Appel	BH	GR	Zhao	Pépin	Benz	JP
Binary/Oct-tree	B	O	O	O	B	B	B
Multipoles	Mono	Quad	Sphr.	Tayl.	Sphr.	Quad	Mono
Adjustable OC	Yes	Yes	No	No	Yes	Yes	Yes
Adaptive	Yes	Yes	No	No	Yes	Yes	Yes
Scaling	N	$N \log N$	N	N	N	$N \log N$	$N \log N$
Interactions	B-B	B-N	N-N	N-N	N-N	B-N	B-N

1.4. Outline of dissertation.

The remainder of this dissertation will be concerned with various aspects of the BH algorithm. We concentrate on the BH algorithm for several reasons:

1. It has received, by far, the most attention in the astrophysics community and

will likely be the most reliable algorithm for use in "real" scientific simulations.

2. Joshua Barnes kindly provided a copy of his version of the code which was written in the C programming language, facilitating portability to Caltech's parallel computers.

3. The firm distinction between the force calculation and the tree construction phase makes effective parallelization much easier. Although by no means impossible to parallelize, Appel's and Greengard's algorithms require considerably more bookkeeping because of the need to store and compute interactions for non-terminal objects in their hierarchies. It is likely that load balance will be considerably harder for either of these algorithms, in comparison to BH.

4. The asymptotic rates, $O(N \log N)$ vs. $O(N)$, do not tell the whole story. Based on Greengard's estimate of $1752N$ pair interactions per time-step[13] (pg. 70–71), and our empirical observation that the BH algorithm with $\theta = 0.8$ requires approximately 15–35 $N \log_2 N$ pair interactions per timestep, the asymptotic regime in which Greengard's algorithm is expected to be superior is near $N \approx 10^{15}$ — well beyond any planned simulations. It is worth noting that the exact value of the crossover point depends exponentially on the precise constants that precede the $O(N \log N)$ and $O(N)$ factors. The turnover point can be changed by many orders of magnitude by relatively small adjustments in parameters or algorithms.

In Chapter 2, we investigate the statistical properties of the BH tree. We present rigorous estimates for the average depth of the tree, the average number of cells, etc. These results are not unexpected, but only heuristic arguments have been presented before. In Chapter 3, we develop some of the mathematics of the multipole expansions used in the BH algorithm. Multipole expansions, of course, are not new, but the fact that the BH algorithm can be applied to non-Newtonian potentials has not been generally recognized. In Chapter 4, we consider some aspects of the algorithm itself. In particular, how much memory and time are required to execute it. Again, the results are not unexpected, but it is reassuring

to obtain rigorous estimates of the expected performance of the algorithm. Especially interesting is the fact that the performance is only weakly effected by highly non-uniform distributions of bodies. In Chapter 5, we return to the multipole expansion, and compute error bounds for various approximations. These error bounds are useful for controlling the accuracy of simulations, and for choosing a strategy for selectively using the multipole approximation. In Chapter 6, we consider the issue of opening criteria, i.e., the question of when the multipole approximation may be used, in some detail. We discuss a serious flaw in the usual procedure, and consider some alternatives that do not suffer from the same difficulty. In Chapter 7, we address the issue of parallel computation. We find that the BH algorithm can be formulated so that it may be executed on a parallel computer. Adaptive load balancing, and a strategy for constructing only a small fraction of the BH tree in each processor is necessary to efficiently use a large number of independent processors. In Chapter 8, we analyze the parallel algorithm in some detail. A general framework for identifying the sources of inefficiency in a parallel algorithm is described, and applied to the parallel BH algorithm. In contrast to the majority of parallel scientific algorithms, the inefficiency in the parallel BH algorithm is not related to interprocessor communication. Most of the overhead is due to processor synchronization and redundant calculations. Appendix A contains detailed proofs of some assertions in Chapter 2. Appendix B contains a reprint on the formation of galactic shells. Appendix C contains a reprint on formation of galaxy halos in an expanding universe. Appendix D contains a reprint on the relationship between the local mass density and the rotation of halos arising in N-body simulations. Appendix E contains a reprint of a paper on the statistics of QSO absorption line systems. Finally, Appendix F contains a reprint of a paper on Cubix, a programming tool which was used for all of the parallel programs described in this dissertation.

2. Properties of the BH Tree.

In this chapter we investigate some statistical properties of the BH tree. Such quantities as the average depth of the tree and the number of non-terminal nodes will be important in the discussions of performance, so we begin by studying the tree itself.

2.1. Building the tree.

We define a generalized BH tree by the following properties:

1. The cells partition space into an octree of cubical sub-cells, so that each cell has eight descendants of equal size.
2. No terminal node of the tree contains more than m bodies. BH consider the case, $m = 1$.
3. Any node of the tree which contains m or fewer bodies is a terminal node, i.e., it is not further subdivided.

There are many ways to construct such a data structure from a list of bodies. One method is to begin with an empty tree and to examine each body in turn, modifying the tree as necessary to accommodate it. To add a body, it is necessary to find the most refined element of the current tree which contains the body, and then to add the body directly to that element, either by refining it further or by simply inserting the body. Barnes describes a recursive procedure which performs the above steps. In the following code fragments, we will encounter two distinct data types:

Bodies: These contain a position, a mass, a velocity, and perhaps other physical data which is carried by the bodies in the simulation.

Cells: These represent cubical volumes of space. Cells which are internal have somewhat different storage requirements and properties from cells which are

terminal.

Cells may be further subdivided into two types:

Internal Cells: These are the internal nodes of the BH tree. Each Cell contains a set of multipoles which may be used to approximate the effect of all the bodies which are descendants of the cell. In addition, a Cell contains pointers to up to eight direct descendants.

Terminal Cells: These are the terminal nodes of the BH tree. As we shall see in Chapter 4, if m is chosen correctly, then terminal cells need not store a multipole expansion. They simply store the header for a linked list of bodies and enough information to specify the size and location of the region of space enclosed by the cell.

```
Insert(Body, Cell)
    if( Cell is not terminal )
        if( Cell has a Child which encloses Body )
            Insert(Body, Child)
        else
            NewChild(Body, Cell)
        endif
    else if( Cell contains fewer than m bodies )
        InsertDirectly(Body, Cell)
        return;
    else if( Cell contains exactly m bodies )
        NewCell=a new, non-terminal cell with
            eight empty children
        for( oldbody = each body in Cell )
            Insert(oldbody, NewCell)
        endfor
        Insert(Body, NewCell)
        replace Cell with NewCell
    endif
endfunc
```

Code 2.1. Procedure for inserting a new body into an existing BH tree.

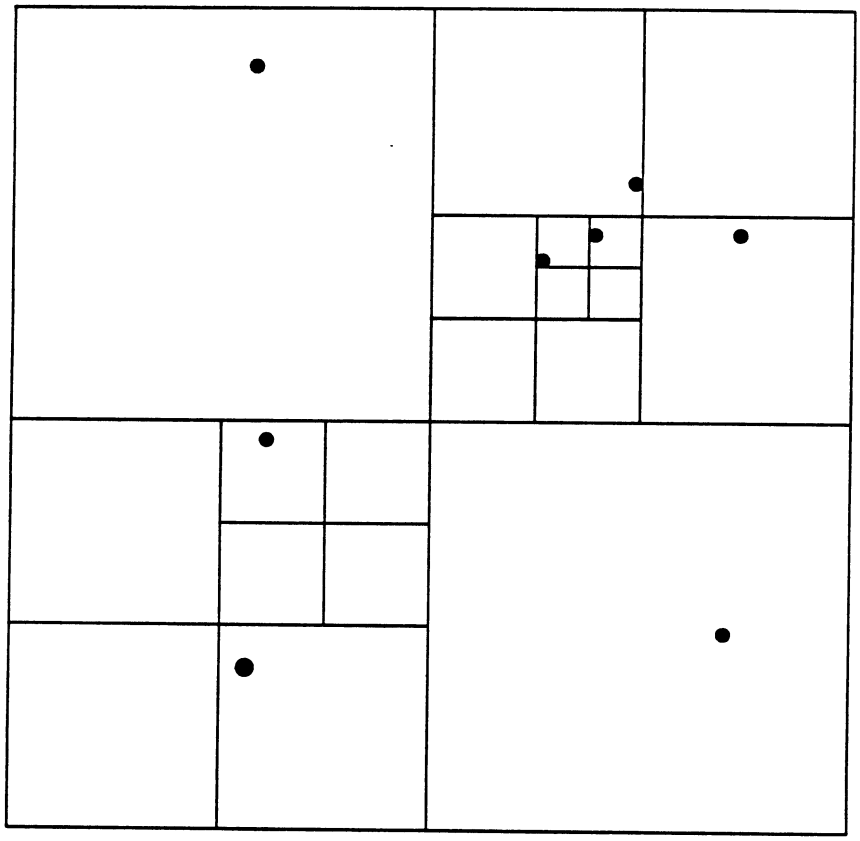


Figure 2.1. Snapshot of an $m = 1$ BH tree after eight bodies have been inserted using Code 2.1.

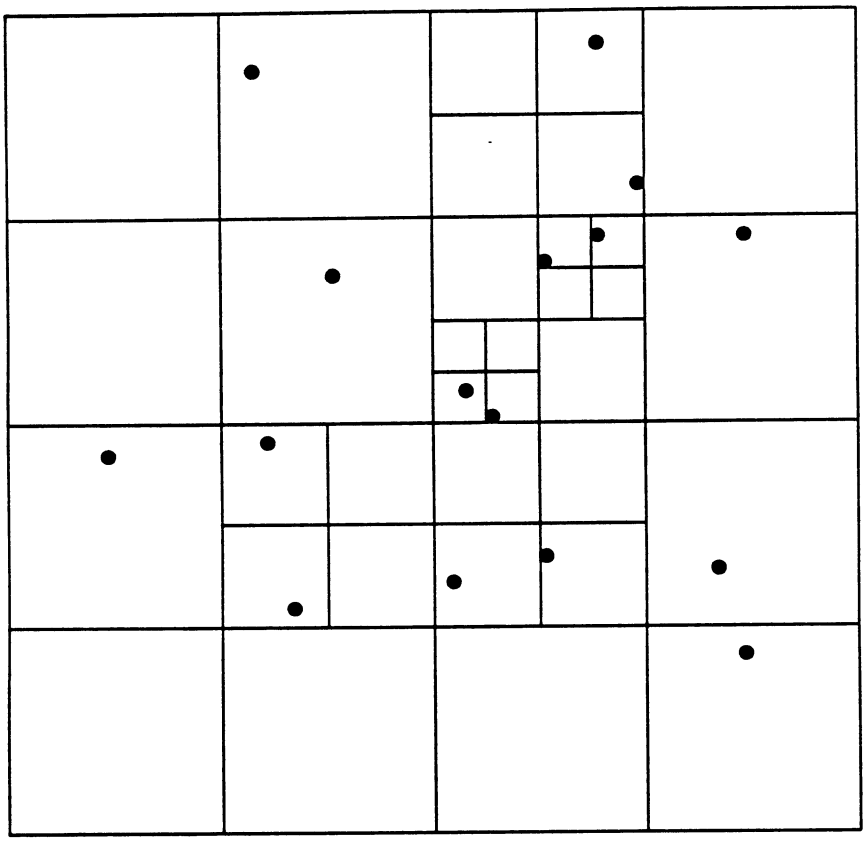


Figure 2.2. Snapshot of an $m = 1$ BH tree after 16 bodies have been inserted using Code 2.1.

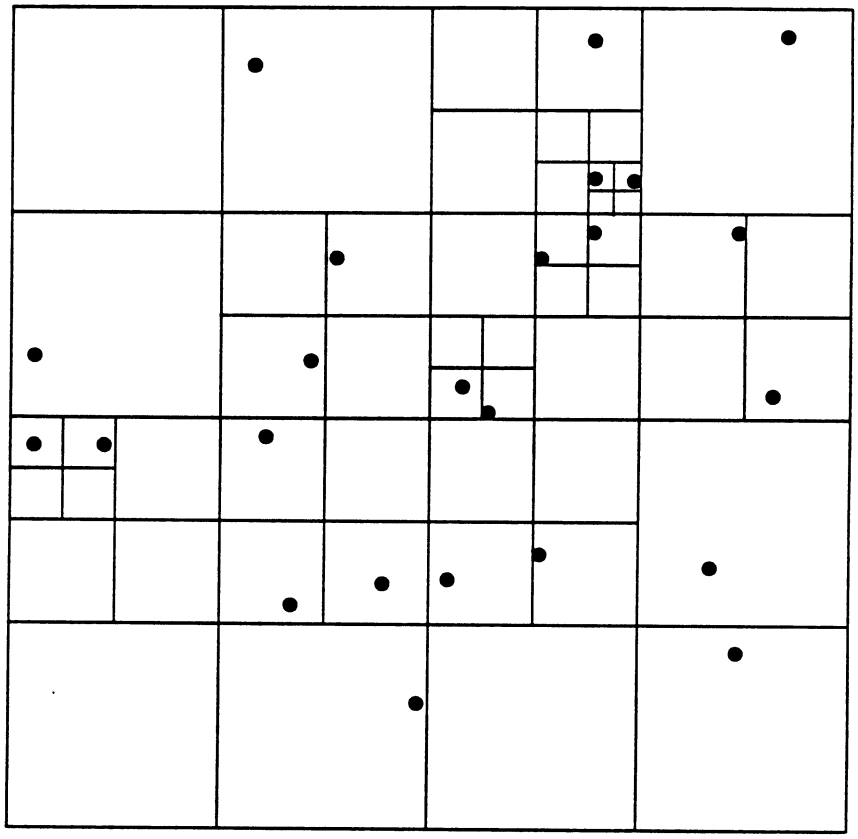


Figure 2.3. Snapshot of an $m = 1$ BH tree after 24 bodies have been inserted using Code 2.1.

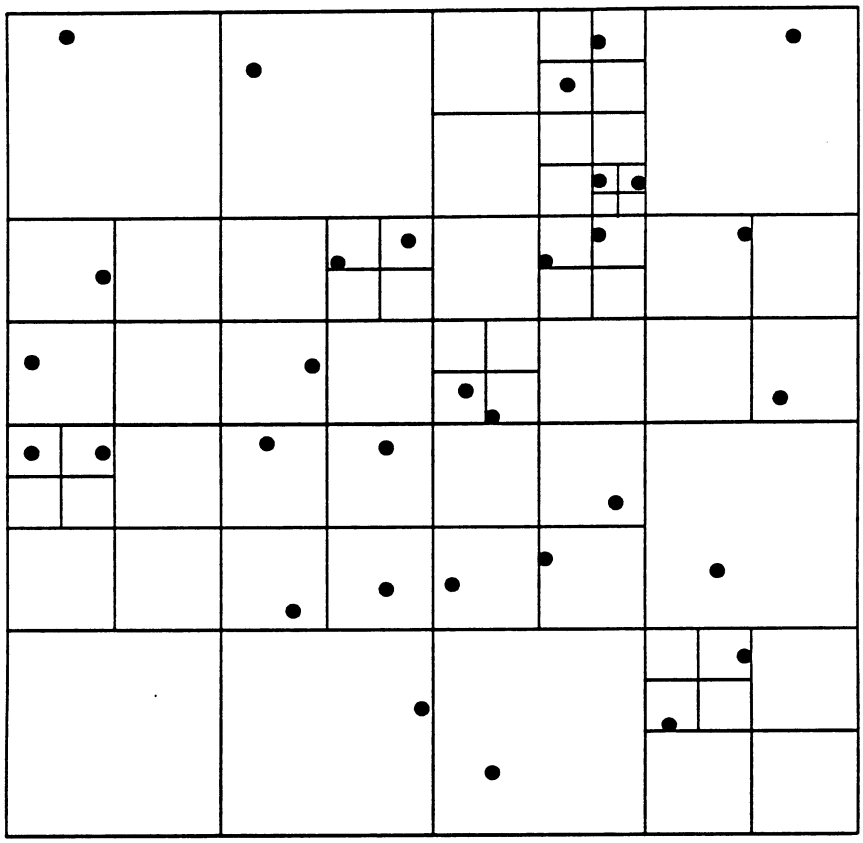


Figure 2.4. Snapshot of an $m = 1$ BH tree after 32 bodies have been inserted using Code 2.1.

The function `Insert` in Code 2.1 is essentially equivalent to Barnes' method of inserting a body into a cell. The function `InsertDirectly`, which is used by `Insert`, simply adds a body to the list of bodies associated with a terminal cell. The function `NewChild`, creates a new terminal node which contains only one `Body`, and links it into the tree as a child of `Cell`.

Figures 2.1 through 2.4 show the evolution of a two-dimensional tree built according to the method of Code 2.1. Notice that the sequence shown in these figures would be very different had we inserted bodies in a different order. Nevertheless, the tree that finally results from this procedure is unique.

Once the tree is in place, the centers-of-mass and multipole moments of each of the internal nodes can be computed using a general form of the parallel axis theorem. We will consider multipole moments and the parallel axis theorem in some detail in Chapter 4. We now turn to the statistical properties of the tree itself.

2.2. Notation.

We begin with a finite cubical volume, V_0, representing all of space, i.e., no body falls outside of this volume, and an underlying probability density function $\bar{\rho}(x)$. Now consider a collection of N coordinates (or bodies), $\{x_1, \ldots, x_N\}$, independent, identically distributed random variables, with probability density $\bar{\rho}(x)$. We are interested in statistical properties of the BH tree constructed from these bodies. For example, how deep is it, how many internal nodes does it contain, and how many operations are entailed by an application of the BH force calculation algorithm, on average?

We begin with some notation. We shall label cubical cells with a subscript γ. Every possible cell is labeled by some γ - those which are in the BH tree, as well as those which are not.

Every cell, except the root, has a unique parent, which we denote by $\uparrow(\gamma)$ In addition, every cell has an associated depth, $d(\gamma)$, which is the number of its

ancestors. We note that all cells at the same depth have the same volume,

$$V_\gamma = V_0 8^{-d(\gamma)}, \tag{2.1}$$

and there are 8^d cells with depth equal to d. We define $\mathcal{L}_d$ as the set of cells at level d, i.e.,

$$\mathcal{L}_d = \{\gamma | d(\gamma) = d\}. \tag{2.2}$$

For any cell, p_γ is the probability that a particular body lies within $\mathcal{V}_\gamma$,

$$p_\gamma = \int_{\mathcal{V}_\gamma} \bar{\rho}(x) d^3x. \tag{2.3}$$

The probability that $\mathcal{V}_\gamma$ will contain exactly i bodies is given by the binomial distribution function,

$$B_{N,i}(p_\gamma) = \binom{N}{i} p_\gamma^i (1 - p_\gamma)^{N-i}. \tag{2.4}$$

We also define the cumulative binomial distribution and its complement,

$$C_{N,m}(p_\gamma) = \sum_{j=0}^{m} B_{N,j}(p_\gamma) \tag{2.5}$$

and

$$D_{N,m}(p_\gamma) = 1 - C_{N,m}(p_\gamma) = \sum_{j=m+1}^{N} B_{N,j}(p_\gamma). \tag{2.6}$$

The functions, $C_{m,N}$ and $D_{m,N}$ are related to the incomplete beta function.[34] For two disjoint volumes, γ_1 and γ_2, the probability that there are exactly i_1 bodies in $\mathcal{V}_{\gamma_1}$ and exactly i_2 bodies in $\mathcal{V}_{\gamma_2}$ is given by

$$B_{N,i_1,i_2}(p_{\gamma_1}, p_{\gamma_2}) = \binom{N}{i_1 i_2} p_{\gamma_1}^{i_1} p_{\gamma_2}^{i_2} (1 - p_{\gamma_1} - p_{\gamma_2})^{N-i_1-i_2}. \tag{2.7}$$

We shall also use the limit

$$\lim_{p \to 0} D_{N,m}(p) = p^{m+1} \binom{N}{m+1}. \tag{2.8}$$

2.3. Expected number of internal cells.

The probability that a cell, γ, is an internal node of the BH tree is exactly the probability that it contains more than m bodies, i.e.,

$$Prob(\gamma \text{ is internal}) = D_{N,m}(p_\gamma). \tag{2.9}$$

Thus, the expected number of internal nodes in the tree is given by

$$\begin{aligned} C_{avg} &= \sum_{\gamma} D_{N,m}(p_\gamma) \\ &= \sum_{d=0}^{\infty} \sum_{\gamma \in \mathcal{L}_d} D_{N,m}(p_\gamma) \\ &= \sum_{d=0}^{\infty} G_{N,m}(d), \end{aligned} \tag{2.10}$$

where we have defined

$$G_{N,m}(d) = \sum_{\gamma \in \mathcal{L}_d} D_{N,m}(p_\gamma). \tag{2.11}$$

As long as $\bar{\rho}$ is bounded, we have

$$p_\gamma \leq \rho_{max} V_0 8^{-d(\gamma)}, \tag{2.12}$$

and hence, using Eqn. 2.8 and the fact that there are 8^d cells in $\mathcal{L}_d$,

$$\lim_{d \to \infty} G_{N,m}(d) = O(8^{-md}). \tag{2.13}$$

Therefore, the sum in Eqn. 2.10 clearly converges. In fact, it converges very rapidly, and we will have no problems evaluating it numerically in Section 2.8.

2.4. Probability that a cell is terminal.

Now we ask for the probability that a cell, $\mathcal{V}_\gamma$, is terminal, and that it has exactly a bodies, denoted by $Prob(Pop(\mathcal{V}_\gamma) = a)$. In order for a cell to be terminal, its parent must be internal. Thus,

$$\begin{aligned}
&Prob(Pop(\gamma) = a) \\
&\quad = Prob(\ \mathcal{V}_\gamma \text{ contains } a \text{ bodies AND } \mathcal{V}_{\uparrow\gamma} \text{ contains } > m \text{ bodies}) \\
&\quad = Prob(\mathcal{V}_\gamma \text{ contains } a \text{ bodies}) \\
&\qquad\quad - Prob(\mathcal{V}_\gamma \text{ contains } a \text{ bodies AND } \mathcal{V}_{\uparrow\gamma} \text{ contains } \leq m \text{ bodies}) \\
&\quad = Prob(\mathcal{V}_\gamma \text{ contains } a \text{ bodies}) \\
&\qquad - \sum_{j=0}^{m-a} Prob(\mathcal{V}_\gamma \text{ contains } a \text{ bodies AND } (\mathcal{V}_{\uparrow\gamma} - \mathcal{V}_\gamma) \text{ contains } j \text{ bodies}).
\end{aligned} \tag{2.14}$$

We can now use the binomial distribution functions defined in Eqn. 2.7 to obtain

$$Prob(Pop(\gamma) = a) = B_{N,a}(p_\gamma) - \sum_{j=0}^{m-a} B_{N,a,j}(p_\gamma, p_{\uparrow\gamma} - p_\gamma). \tag{2.15}$$

The probability that a cell is terminal, regardless of how many bodies it contains, is clearly a summation over Eqn. 2.15,

$$\begin{aligned}
Prob(\mathcal{V}_\gamma \text{is terminal}) &= \sum_{a=0}^{m} \left(B_{N,a}(p_\gamma) - \sum_{j=0}^{m-a} B_{N,a,j}(p_\gamma, p_{\uparrow\gamma} - p_\gamma) \right) \\
&= C_{N,m}(p_\gamma) - \sum_{a=0}^{m} \sum_{j=0}^{m-a} B_{N,a,j}(p_\gamma, p_{\uparrow\gamma} - p_\gamma).
\end{aligned} \tag{2.16}$$

Now observe that

$$\sum_{a=0}^{m} \sum_{j=0}^{m-a} = \sum_{s=0}^{m} \sum_{a=0}^{s} \qquad \text{where} \quad s = a + j, \tag{2.17}$$

and also, by the binomial theorem

$$\sum_{a=0}^{s} B_{N,a,s-a}(p_\gamma, p_{\uparrow\gamma} - p_\gamma) = B_{N,s}(p_{\uparrow\gamma}). \tag{2.18}$$

From Eqns. 2.16, 2.17, and 2.18, we obtain

$$\begin{aligned} Prob(\mathcal{V}_\gamma \text{is terminal}) &= C_{N,m}(p_\gamma) - \sum_{s=0}^{m}\sum_{a=0}^{s} B_{N,a,s-a}(p_\gamma, p_{\uparrow\gamma} - p_\gamma) \\ &= C_{N,m}(p_\gamma) - \sum_{s=0}^{m} B_{N,s}(p_{\uparrow\gamma}) \\ &= C_{N,m}(p_\gamma) - C_{N,m}(p_{\uparrow\gamma}) \\ &= D_{N,m}(p_{\uparrow\gamma}) - D_{N,m}(p_\gamma). \end{aligned} \tag{2.19}$$

2.5. Expected number of terminal cells.

We now estimate the expected number of terminal cells, T_{avg} in a BH tree with N bodies as

$$\begin{aligned} T_{avg} &= \sum_{\gamma} Prob(\mathcal{V}_\gamma \text{is terminal}) \\ &= \sum_{d=0}^{\infty}\sum_{\gamma\in\mathcal{L}_d} Prob(\mathcal{V}_\gamma \text{is terminal}) \\ &= \sum_{d=0}^{\infty}\sum_{\gamma\in\mathcal{L}_d} (D_{N,m}(p_{\uparrow\gamma}) - D_{N,m}(p_\gamma) \\ &= \sum_{d=1}^{\infty} 8G_{N,m}(d-1) - G_{N,m}(d) \\ &= 7\sum_{d=0}^{\infty} G_{N,m}(d) \\ &= 7C_{avg}. \end{aligned} \tag{2.20}$$

Thus, the expected number of terminal cells is exactly seven times the expected number of internal cells. This is not a surprising result. In general, if an octree has

N_{term} terminal nodes, then we can expect approximately $N_{term}/8$ nodes which are parents of terminal nodes, $N_{term}/64$ nodes which are grandparents, etc. The total number of nodes will be the geometric sum,

$$\frac{N_{term}}{7} = \frac{N_{term}}{8} + \frac{N_{term}}{64} + \cdots \tag{2.21}$$

Equation 2.20 verifies that the loose reasoning of Eqn. 2.21 is indeed exactly correct when treating expectations of cell populations. We note that the sums in this section include terminal nodes with zero bodies. In practice, such empty terminal nodes will not require storage space, so the results are not of great significance.

2.6. Expected population of terminal cells.

The expected population of $\mathcal{V}_\gamma$ is also a summation over Eqn. 2.15,

$$< Pop(\mathcal{V}_\gamma) >= \sum_{a=1}^{m} aProb(Pop(\mathcal{V}_\gamma) = a). \tag{2.22}$$

Note that we are considering the population of a cell to be zero if it is non-terminal.

We now make use of the following identities:

$$aB_{N,a}(p) = NpB_{N-1,a-1}(p) \tag{2.23}$$

$$aB_{N,a,j}(p,q) = NpB_{N-1,a-1,j}(p,q). \tag{2.24}$$

Thus,

$$\begin{aligned} < Pop(\mathcal{V}_\gamma) > &= Np_\gamma \sum_{a=1}^{m} \left(B_{N-1,a-1}(p_\gamma) - \sum_{j=0}^{m-a} B_{N-1,a-1,j}(p_\gamma, p_{\uparrow\gamma} - p_\gamma) \right) \\ &= Np_\gamma \left(C_{N-1,m-1}(p_\gamma) - \sum_{a=0}^{m-1} \sum_{j=0}^{m-1-a} B_{N-1,a,j}(p_\gamma, p_{\uparrow\gamma} - p_\gamma) \right). \end{aligned} \tag{2.25}$$

Again, using Eqns. 2.17 and 2.18, we obtain

$$\begin{aligned} < Pop(\mathcal{V}_\gamma) > &= Np_\gamma \left(C_{N-1,m-1}(p_\gamma) - \sum_{s=0}^{m-1} \sum_{a=0}^{s} B_{N-1,a,s-a}(p_\gamma, p_{\uparrow\gamma} - p_\gamma) \right) \\ &= Np_\gamma \left(C_{N-1,m-1}(p_\gamma) - \sum_{s=0}^{m-1} B_{N-1,s}(p_{\uparrow\gamma}) \right) \\ &= Np_\gamma \left(C_{N-1,m-1}(p_\gamma) - C_{N-1,m-1}(p_{\uparrow\gamma}) \right) \\ &= Np_\gamma \left(D_{N-1,m-1}(p_{\uparrow\gamma}) - D_{N-1,m-1}(p_\gamma) \right). \end{aligned} \tag{2.26}$$

This result tells us the expectation value of the number of "terminal bodies" in a cell whose integrated probability is p_γ and whose parent's integrated probability is $p_{\uparrow\gamma}$. For very large p_γ, the expectation is small because it is very unlikely that the cell is terminal. For very small p_γ, the expectation is also small because it is unlikely that any bodies fall in the cell at all.

We can quantify these observations if we assume that N is very large, and that p_γ is very small, but that $Np_\gamma = \lambda_\gamma$. Under these conditions, the binomial distribution goes over to the Poisson distribution,

$$B_{N,i}(p_\gamma) \approx P_i(\lambda_\gamma) = \frac{\lambda_\gamma^i}{i!} e^{-\lambda_\gamma}. \tag{2.27}$$

Furthermore, we assume that $\bar{\rho}(x)$ is almost constant over the volume $\mathcal{V}_{\uparrow\gamma}$, in which case

$$p_{\uparrow\gamma} = 8p_\gamma. \tag{2.28}$$

Under these conditions, Eqn. 2.26 becomes

$$< Pop(\mathcal{V}_\gamma) >= \lambda_\gamma \sum_{i=0}^{m-1} \left(P_i(\lambda_\gamma) - P_i(8\lambda_\gamma) \right). \tag{2.29}$$

Figure 2.5 shows a plot of $< Pop(\mathcal{V}_\gamma) >$ as a function of λ_γ, for m ranging from 1 through 5. The maximum of the expected population is approximately

proportional to m. This is because m is the maximum number of bodies that can reside in a terminal cell. As m increases, we expect proportionally more bodies in terminal cells. Furthermore, the value of λ_γ which gives rise to the maximum expected population is also approximately equal to m. The reason is that for significantly higher λ_γ, cells are likely not to be terminal, while for significantly smaller λ_γ, cells are likely to have low populations, or not to have parents which are internal cells. Figure 2.6 shows the same data as Figure 2.5, but with the axes rescaled to reflect the fact that the heights and widths of the curves in Figure 2.5 are approximately proportional to m.

From Figure 2.6, we see that the expected number of terminal bodies in a cell peaks when λ_γ is slightly less than m. Furthermore, even for "resonant" values of λ_γ, close to the peak, the expected population of $\mathcal{V}_\gamma$ is slightly less than about $m/2$. Thus we conclude that in a BH tree with a maximum of m bodies per node, we can expect, on average, slightly fewer than $m/2$ bodies in the terminal nodes. This should not be a surprising result. Consider the fact that the probability that a given node, not necessarily terminal, with $\lambda_\gamma \approx m$ has m bodies is not too much different from the probability that it has $m+1$. In the latter case, it will likely give rise to eight smaller terminal cells, with an average population of only $(m+1)/8$. Thus, almost-empty terminal cells are about as likely as almost-full terminal cells, and it isn't surprising that the average population of terminal cells is near to $m/2$.

2.7. Average depth of bodies.

Now we can ask for the expected population of all cells with depth equal to d. Clearly, this is just a sum over all appropriate γ of Eqn. 2.26,

$$< Pop(depth = d) >= \sum_{\gamma \in \mathcal{L}_d} N p_\gamma (D_{N-1,m-1}(p_{\uparrow\gamma}) - D_{N-1,m-1}(p_\gamma)). \qquad (2.30)$$

Equation 2.30 may be simplified by noting that

$$p_\alpha = \sum_{\gamma \ni \uparrow(\gamma)=\alpha} p_\gamma, \qquad (2.31)$$

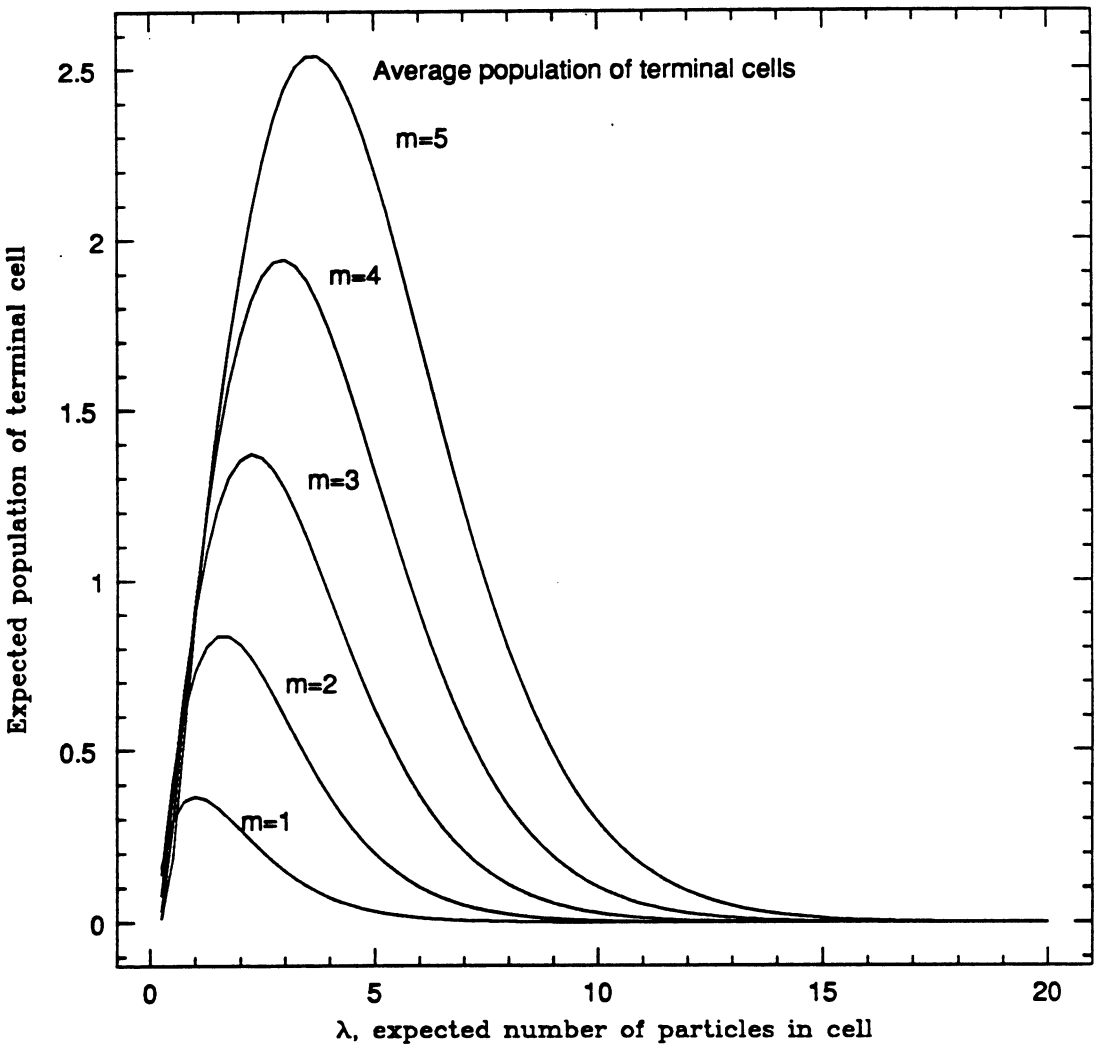


Figure 2.5. The expected value of the number of bodies in a cell with $Np_\gamma = \lambda_\gamma$.

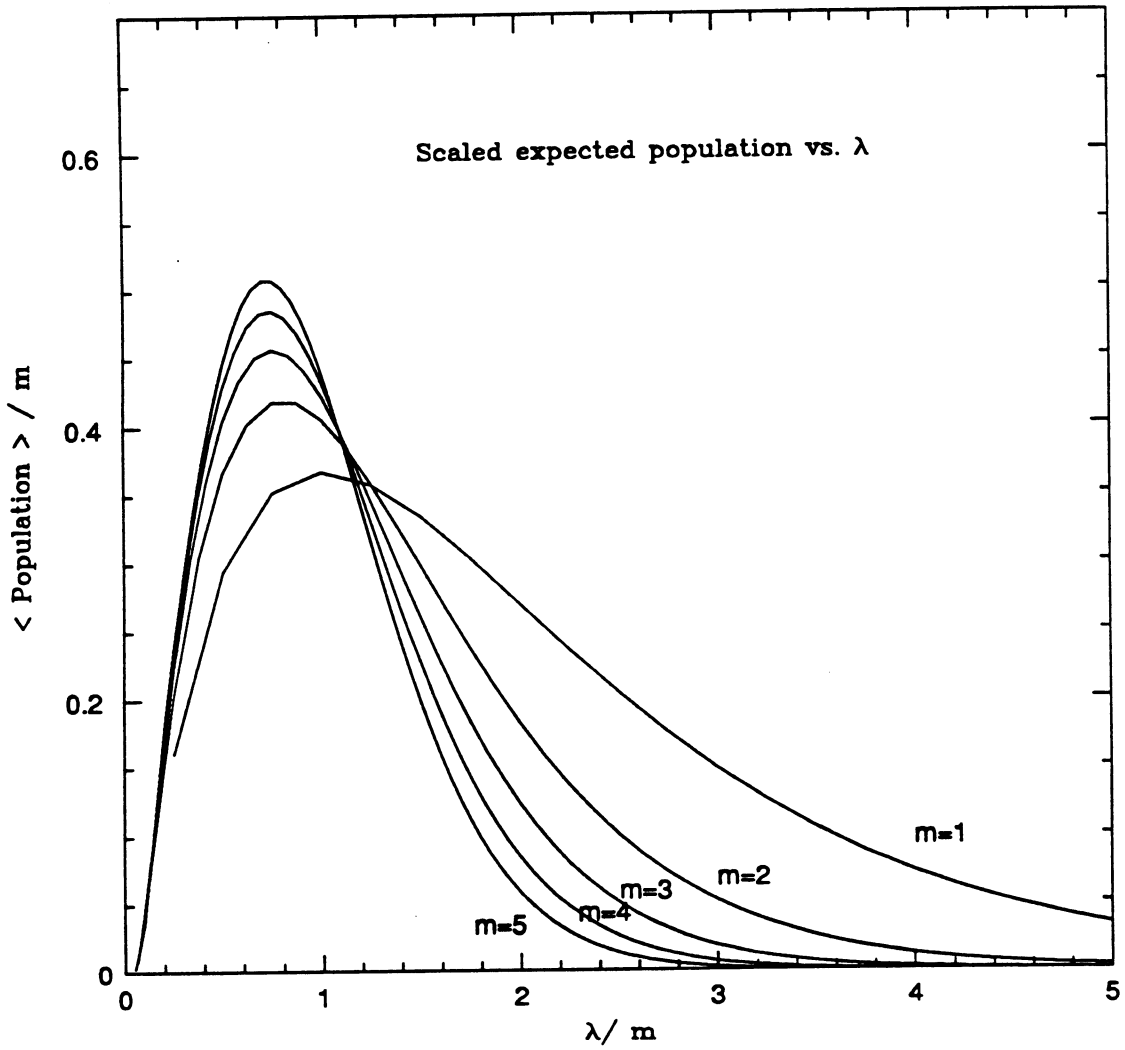


Figure 2.6. The expected value of the number of bodies in a cell with $Np_\gamma = \lambda_\gamma$, with both the ordinate and abcissa scaled by m.

i.e., for a given cell, the sum of the probabilites of its children is equal to its own probability. Thus,

$$\begin{aligned} < Pop(depth = d) > &= N \left(\sum_{\gamma \in \mathcal{L}_{d-1}} p_\gamma D_{N-1,m-1}(p_\gamma) - \sum_{\gamma \in \mathcal{L}_d} p_\gamma D_{N-1,m-1}(p_\gamma) \right) \\ &= N \left(F_{N-1,m-1}(d-1) - F_{N-1,m-1}(d) \right), \end{aligned} \tag{2.32}$$

where we define

$$F_{N,m}(d) = \sum_{\gamma \in \mathcal{L}_d} p_\gamma D_{N,m}(p_\gamma). \tag{2.33}$$

We can check our calculation by computing the sum of the expected population, over all values of d,

$$\begin{aligned} < \text{Total population} > &= \sum_{d=0}^{\infty} < Pop(depth = d) > \\ &= N \left(\sum_{d=1}^{\infty} (F_{N-1,m-1}(d-1) - F_{N-1,m-1}(d)) \right) \\ &= N \lim_{D \to \infty} \left(\sum_{d=1}^{D} (F_{N-1,m-1}(d-1) - F_{N-1,m-1}(d)) \right) \\ &= N \lim_{D \to \infty} (F_{N-1,m-1}(0) - F_{N-1,m-1}(D)) \\ &= N \left(1 - \lim_{D \to \infty} F_{N-1,m-1}(D) \right) \\ &= N. \end{aligned} \tag{2.34}$$

The last identity follows from the fact that

$$F_{N,m}(d) = O(8^{-md}) \tag{2.35}$$

by the same argument as led to Eqn. 2.13.

The total population is exactly as it should be. By a very roundabout method, we have determined that the expected number of bodies in a BH tree constructed from N bodies is precisely N.

The average depth of a body in a BH tree may also be obtained from Eqn. 2.31,

$$\begin{aligned} D_{avg} &= \frac{1}{N}\sum_{d=0}^{\infty} d < Pop(depth = d) > \\ &= \lim_{D\to\infty}\sum_{d=1}^{D} d(F_{N-1,m-1}(d-1) - F_{N-1,m-1}(d))) \\ &= \lim_{D\to\infty}(\sum_{d=0}^{D-1} F_{N-1,m-1}(d) - DF_{N-1,m-1}(D)) \\ &= \sum_{d=0}^{\infty} F_{N-1,m-1}(d) \\ &= \sum_{\gamma} p_\gamma D_{N-1,m-1}(p_\gamma). \end{aligned} \tag{2.36}$$

The value of $F_{N,m}(d)$ depends on the details of the probability distribution, $\bar{\rho}$. We cannot make further progress analyzing the average depth of the BH tree, or the average number of cells it contains, unless we make assumptions about $\bar{\rho}$.

2.8. Uniform distribution, i.e., $\bar{\rho}(x) = const.$

If the distribution of bodies, $\bar{\rho}(x)$, is constant, we can explicitly evaluate $F_{N,m}(d)$ and $G_{N,m}(d)$ and then numerically evaluate C_{avg} and D_{avg}. The results are interesting in their own right, as approximately uniform distributions of bodies are used to describe the early universe in cosmological simulations. More importantly, we can extrapolate the results for a uniform distribution to situations with a non-uniform distribution, but a sufficiently large value of N.

We now consider the situation in which $\bar{\rho}$ is a uniform distribution, i.e., a body is equally likely to appear anywhere in the volume, $\mathcal{V}_0$. Since $\bar{\rho}(x)$ is a probability density, we have

$$\bar{\rho}(x) = \frac{1}{V_0}, \tag{2.37}$$

$$p_\gamma = 8^{-d(\gamma)}, \tag{2.38}$$

and there are exactly 8^d cells with depth d. Equation 2.11 becomes

$$G_{N,m}(d) = 8^d(D_{N,m}(8^{-d})), \tag{2.39}$$

and Eqn. 2.33 becomes

$$F_{N,m}(d) = D_{N,m}(8^{-d}). \tag{2.40}$$

Now we use Eqns. 2.10 and 2.36 and obtain

$$C_{avg} = \sum_{d=0}^{\infty} 8^d D_{N,m}(8^{-d}) \tag{2.41}$$

and

$$D_{avg} = \sum_{d=0}^{\infty} D_{N-1,m-1}(8^{-d}). \tag{2.42}$$

We can verify that they are well defined if we make use of the limit, Eqn. 2.8. In fact, Eqn. 2.8 tells us that the sums converge very rapidly and can eaily be evaluated numerically given values N and m.

Figure 2.7 shows C_{avg} computed according to Eqns. 2.41 and 2.10. Based on the figure one is tempted to conjecture that

$$C_{avg} < \frac{Nc_m}{m} \qquad \text{for all } N. \tag{2.43}$$

A plot of $\frac{mC_{avg}}{N}$ is shown in Figure 2.8, in which the abscissa is remarkably constant over a large range in N.

Figure 2.9 shows D_{avg} computed according to Eqns. 2.42. Again, the figure suggests a very tight bound,

$$D_{avg} < \log_8 \frac{d_m N}{m} \qquad \text{for all } N, \tag{2.44}$$

which is substantiated in Figure 2.10 by a plot of $\frac{m}{N}8^{D_{avg}}$ vs. N. Proving these conjectures is rather involved, and is the subject of Appendix A.

We show values for c_m and d_m for several values of m in Table 2.1.

Table 2.1. Values of c_m and d_m as defined in Equations 2.43 and 2.44.

m	c_m	d_m
1	0.50	5.2
2	0.52	4.1
3	0.55	4.0
4	0.58	4.1
5	0.63	4.2
20	0.78	5.2

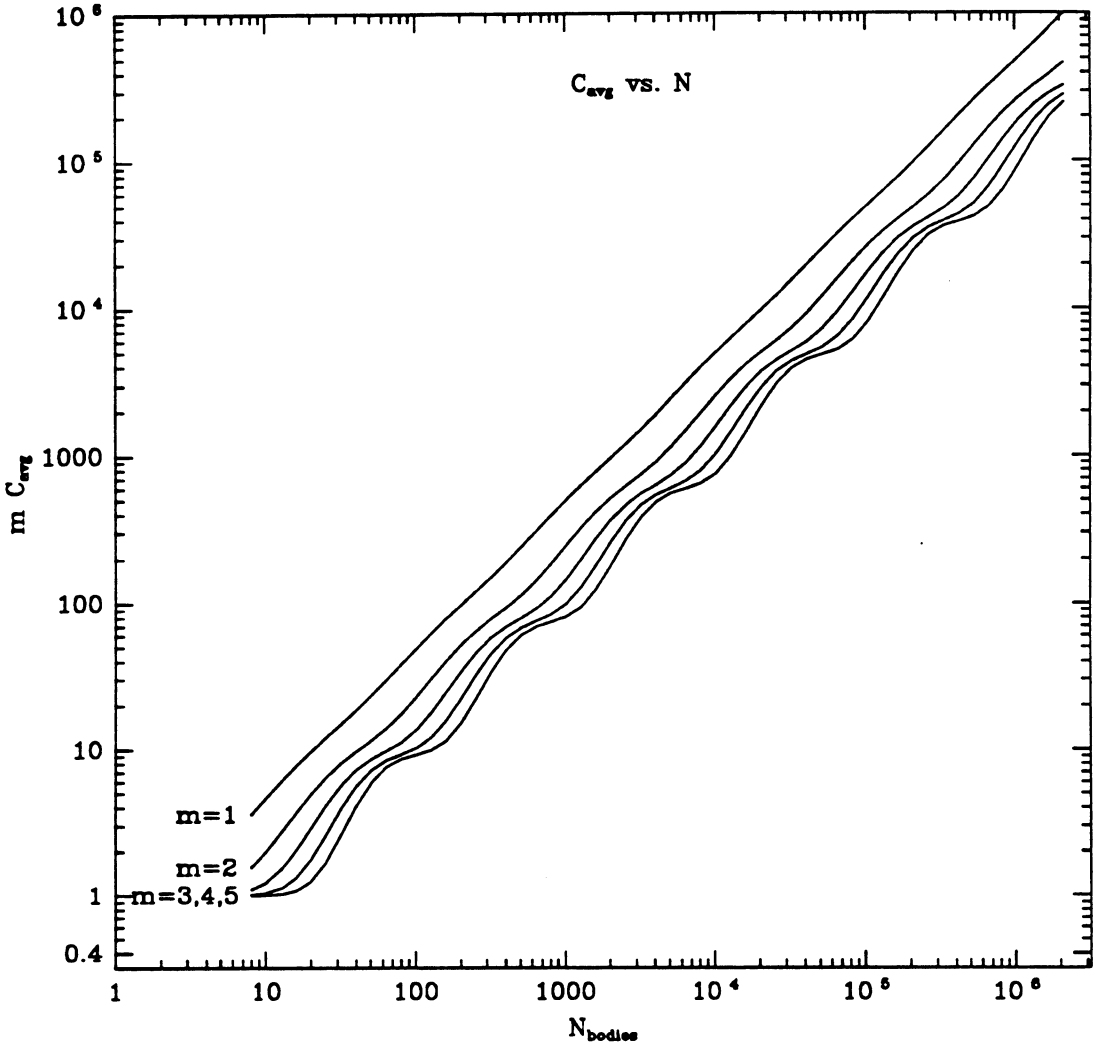


Figure 2.7. Plot of C_{avg} vs. N. for several values of m.

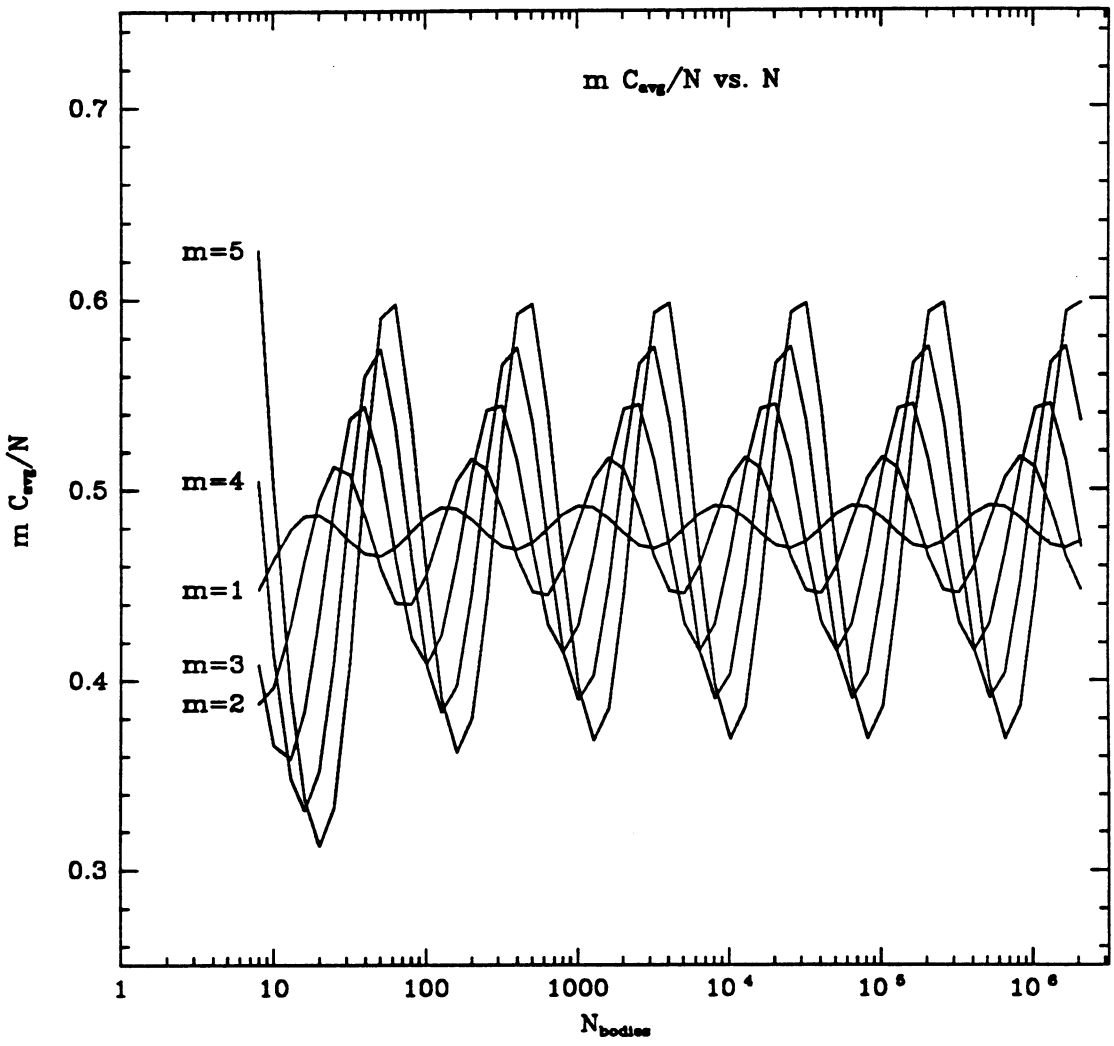


Figure 2.8. Plot of $\frac{mC_{avg}}{N}$ for several values of m. Note how C_{avg} is approximately proportional to $\frac{N}{m}$.

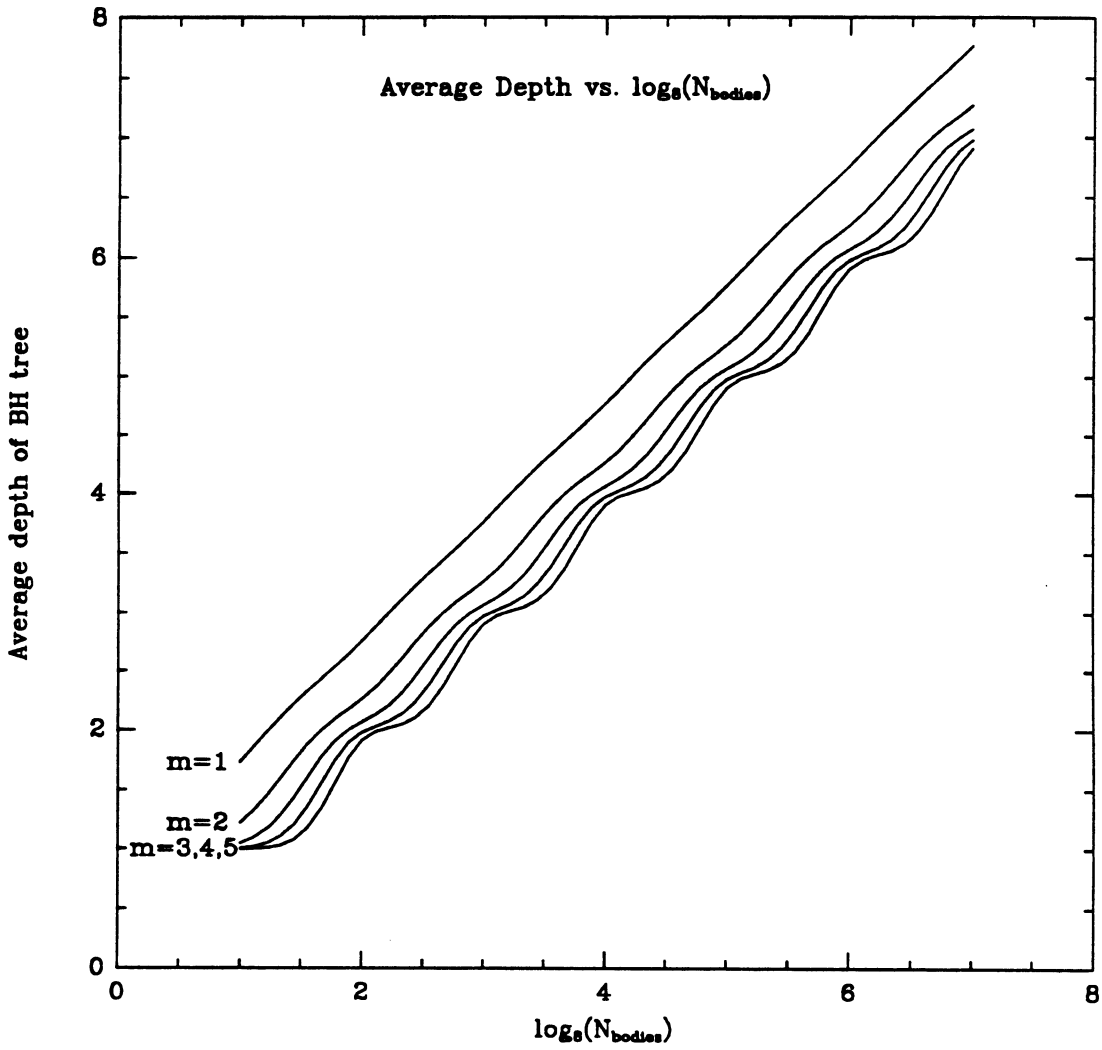


Figure 2.9. Plot of D_{avg} vs. N. for several values of m.

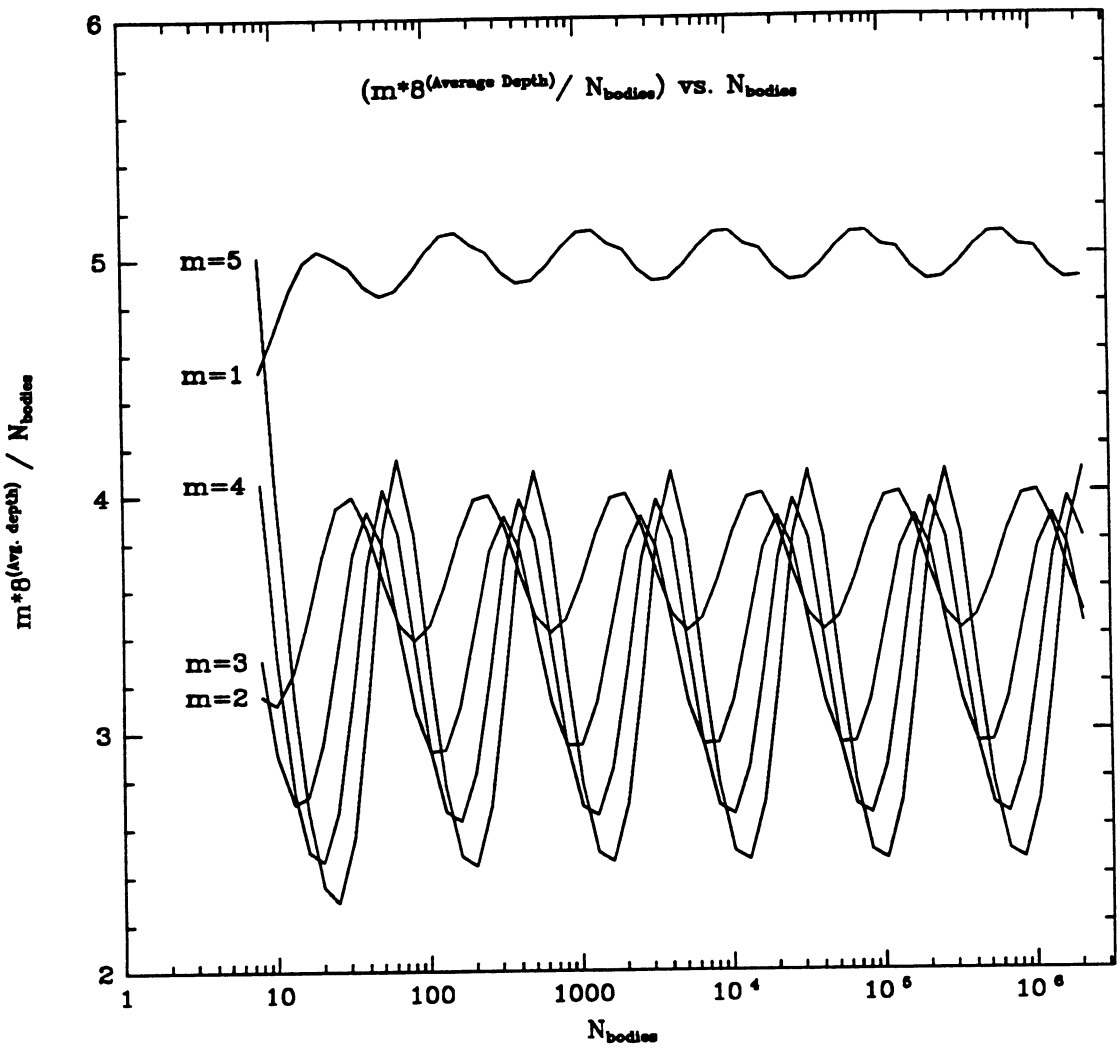


Figure 2.10. Plot of $\frac{m}{N}8^{D_{avg}}$ vs. N for several values of m.

2.9. Non-uniform distribution, i.e., $\bar{\rho}(x) \neq const.$

Now we return to distributions of bodies that are not constant. We can no longer explicitly evaluate quantities like $F_{N,m}(d)$ and $G_{N,m}(d)$, so our conclusions cannot be as specific as in the previous section. However, we will find that if N is reasonably large, then the bounds of previous section apply with minor modifications, even if $\bar{\rho}(x)$ is not constant. The exact condition on the magnitude of N is simply a precise statement of the idea that N must be large enough so that a collection of N samples drawn from the distribution, $\bar{\rho}(x)$, is sufficient to accurately reconstruct the distribution, $\bar{\rho}(x)$. This condition is always satisfied by practical N-body simulations.

2.9.1. Large N.

We start with the assumption that $\bar{\rho}$ is well-behaved, i.e., bounded with bounded derivatives, etc., everywhere in the region $\mathcal{V}_0$. Then, to any desired level of accuracy, we can find a set of disjoint cells that partition $\mathcal{V}_0$ such that $\bar{\rho}$ is approximately constant over each cell. Call this set $\mathcal{S}$. We can precisely characterize $\mathcal{S}$ by

$$\begin{aligned} &\sum_{\gamma \in \mathcal{S}} p_\gamma = 1, \\ &\forall \gamma \in \mathcal{S}: \quad \bar{\rho}(x) \text{ is approximately constant over } \mathcal{V}_\gamma. \end{aligned} \tag{2.45}$$

We designate by $|\mathcal{S}|$ the number of elements in $\mathcal{S}$.

Now, we compute the expected number of bodies in each member of $\mathcal{S}$,

$$N_\gamma = N p_\gamma. \tag{2.46}$$

The condition we need for the results of this section is that

$$N_\gamma^{\frac{1}{2}} \gg 1 \qquad \gamma \in \mathcal{S}. \tag{2.47}$$

That is, in every region over which $\bar{\rho}$ is approximately constant, we can expect a substantial number of bodies. The purpose of the one-half power in Eqn. 2.47 is so that the number of bodies that actually fall within each $\gamma \in \mathcal{S}$ will be a good

estimate of N_γ. In the following, we will neglect all terms smaller than $N_\gamma^{-\frac{1}{2}}$. We shall refer to a distribution $\bar{\rho}$ which satisfies these conditions as "measurable with N samples." If $\bar{\rho}$ is measurable, then it can be accurately reconstructed from a statistical realization with N samples, simply by approximating:

$$\bar{\rho}(x) \approx \frac{N_\gamma}{NV_{gamma}} \qquad \text{for } x \in \mathcal{V}_\gamma. \tag{2.48}$$

2.9.2. Bounds on C_{avg}.

It is well known that the binomial distribution, $B_{N,i}(p)$, has mean value, Np, and variance $Np(1-p)$. Thus, in each cell in the set $\mathcal{S}$, we can expect to find

$$Np_\gamma \pm (Np_\gamma(1-p_\gamma))^{\frac{1}{2}} = N_\gamma \left(1 \pm (1-p)^{\frac{1}{2}} N_\gamma^{-\frac{1}{2}}\right) \tag{2.49}$$

bodies. Based on Eqn. 2.47, we can neglect the last term in Eqn. 2.49 and conclude that there will be almost exactly N_γ bodies in each of the elements of $\mathcal{S}$. We can now treat each of the cells, $\gamma \in \mathcal{S}$, as a separate and independent BH tree. By construction, these cells have constant $\bar{\rho}$, and a specified number of bodies. We can now simply calculate the number of bodies in the entire BH tree,

$$\begin{aligned} C_{avg}(N) = &\sum_{\gamma \in \mathcal{S}} C_{avg}(N_\gamma) \\ &+ \#\text{ cells resulting from assembly of the elements of } \mathcal{S}. \end{aligned} \tag{2.50}$$

The elements of $\mathcal{S}$ are disjoint and cover all of $\mathcal{V}_0$, taken together, they form the terminal nodes of an octree (quadtree in two dimensions). The number of additional nodes which constitute their ancestors in the entire BH tree must be less than $\frac{|\mathcal{S}|}{7}$. Thus, we have

$$C_{avg}(N) = \frac{|\mathcal{S}|}{7} + \sum_{\gamma \in \mathcal{S}} C_{avg}(N_\gamma). \tag{2.51}$$

From Eqn. 2.43, we have

$$C_{avg}(N) < \frac{|\mathcal{S}|}{7} + \sum_{\gamma \in \mathcal{S}} \frac{c_m N_\gamma}{m}, \tag{2.52}$$

and, since the elements of $\mathcal{S}$ are a disjoint partition of V_0, we have

$$C_{avg}(N) < \frac{c_m N}{m} + \frac{|\mathcal{S}|}{7}. \tag{2.53}$$

2.9.3. Bounds on D_{avg}.

We now turn to the average depth of bodies when the probability density is not constant. Again, we express $D_{avg}(N)$ in terms of a sum over the elements of $\mathcal{S}$,

$$D_{avg}(N) = \frac{1}{N} \sum_{\gamma \in \mathcal{S}} N_\gamma (D_{avg}(N_\gamma) + \text{depth}(\gamma)). \tag{2.54}$$

First, we note that the depth of a cell, γ, may be expressed in terms of the logarithm of its volume,

$$depth(\gamma) = -\log_8 \frac{V_\gamma}{V_0}. \tag{2.55}$$

We can make use of Eqn. 2.44, and Eqn. 2.54 becomes

$$D_{avg}(N) < \frac{1}{N} \sum_{\gamma \in \mathcal{S}} N_\gamma \left(\log_8 \left(\frac{d_m N_\gamma}{m} \right) - \log_8 (\frac{V_\gamma}{V_0}) \right). \tag{2.56}$$

Now, we recall that

$$N_\gamma = N p_\gamma, \tag{2.57}$$

so Eqn. 2.56 becomes

$$D_{avg}(N) < \log_8 \left(\frac{d_m N}{m} \right) + \sum_{\gamma \in \mathcal{S}} p_\gamma \log_8 \left(p_\gamma \frac{V_0}{V_\gamma} \right). \tag{2.58}$$

The sum in Eqn. 2.58 has some interesting properties. First, we note that it may be approximated by an integral over the volume V_0. Since $\bar{\rho}$ is approximately constant over each $\mathcal{V}_\gamma$, we have

$$p_\gamma \approx \bar{\rho}(x_\gamma) V_\gamma. \tag{2.59}$$

We define the quantity, H, as the sum in Eqn. 2.58, and in light of Eqn. 2.59, we have

$$H \equiv \sum_{\gamma \in \mathcal{S}} p_\gamma \log_8 \left(p_\gamma \frac{V_0}{V_\gamma} \right) \approx \int_{\mathcal{V}_0} \bar{\rho}(x) \log_8 \left(\bar{\rho}(x) V_0 \right) d^3x, \tag{2.60}$$

and

$$D_{avg}(N) < \log_8 \left(\frac{d_m N}{m} \right) + H. \tag{2.61}$$

The integral in Eqn. 2.60 is similar to the "entropy" of the probability distribution, $\bar{\rho}$. Thus, we might expect that it has a minimum when $\bar{\rho}$ is constant. This is indeed the case. One can easily show, using the method of Lagrange multipliers that H has an extremum when

$$\frac{V_\gamma}{p_\gamma} = V_0, \tag{2.62}$$

in which case, we trivially have

$$H = 0. \tag{2.63}$$

Furthermore, the matrix of second-derivatives of H, i.e., its Hessian, is positive definite when Eqn. 2.62 is satisfied. Thus, we conclude that

$$H \geq 0, \tag{2.64}$$

and equality only holds if Eqn. 2.62 is satisfied.

In practice, H is usually a fairly small constant. For example, the Jaffe model[35] cut off at $R = \Lambda r_0$, which is used as a benchmark in Chapter 8 and which is used for the initial configurations in Appendix C, has a density profile of

$$\bar{\rho}(r) = \begin{cases} \frac{\Lambda+1}{\Lambda} \frac{1}{4\pi r_0^3} \frac{r_0^4}{r^2(r+r_0)^2}, & \text{if } r < \Lambda r_0; \\ 0, & \text{otherwise.} \end{cases}$$

It is a simple exercise to compute the "entropy,"

$$H_{Jaffe} = \left(\frac{4+3\Lambda}{\Lambda} \right) \log_8(1+\Lambda) - \log_8(3e^2). \tag{2.65}$$

For large Λ, H grows only logarithmically with Λ,

$$\lim_{\Lambda \to \infty} H_{Jaffe} = \log_8 \left(\frac{\Lambda^3}{3e^2} \right), \tag{2.66}$$

while for small Λ, H_{Jaffe} approaches a constant,

$$\lim_{\Lambda \to \infty} H_{Jaffe} = \log_8 \left(\frac{e^2}{3} \right) \approx 0.43. \tag{2.67}$$

The benchmarks in Chapter 8 were run with $\Lambda = 10$. The corresponding value of H is

$$H_{Jaffe}(\Lambda = 10) \approx 2.43. \tag{2.68}$$

Clearly, the magnitude of H is not very large.

For practical values of N, in the range 10^4–10^6, H is considerably smaller than $\log_8(d_m N/m)$, so the correction due to the non-uniformity of the distribution does not greatly affect the average depth of the tree.

3. Multipole Expansions.

In this section, we treat some of the mathematical issues that arise in the application of clustering techniques to potential problems. The analysis will proceed in several steps. First, we will develop a multipole expansion for a very general class of potential problems. Then we will specialize these results for particular forms of the potential, proceeding from completely general to spherically symmetric, to a softened Newtonian potential, to a strictly Newtonian potential. As we shall see, the strictly Newtonian potential has some important mathematical features which simplify both the mathematical analysis as well as the computational problem.

3.1. The multipole expansion.

The computational N-body problem is usually an approximation to a continuous physical system or a discrete physical system with a very much larger number of degrees of freedom (simulations of globular clusters are a notable exception). In a large elliptical galaxy, there may be 10^{11} stars, sufficiently many that the densities and distributions of stars may be treated as a continuum. In the continuum limit, the problem is described by a Green's function, $G(r)$, a density field, $\rho(\vec{r})$, and a potential, $\phi(\vec{r})$, related by

$$\phi(r) = \int_{\mathcal{V}} G(\vec{r} - \vec{r}')\rho(\vec{r}')d^3r', \tag{3.1}$$

where $\mathcal{V}$ is the volume over which the density is known to be non-zero. The density and Green's function are considered "inputs" and the potential, or perhaps its gradient,

$$\vec{a} \equiv -\vec{\nabla}\phi, \tag{3.2}$$

are the desired "output."

First, we note that the expression for ϕ in terms of ρ is linear in ρ. If we partition space into disjoint regions, $\mathcal{V}_\gamma$ such that

$$\mathcal{V} = \bigcup_\gamma \mathcal{V}_\gamma. \tag{3.3}$$

Then

$$\phi(r) = \sum_\gamma \phi_\gamma(r), \tag{3.4}$$

$$\phi_\gamma(r) = \int_{\mathcal{V}_\gamma} G(\vec{r} - \vec{x})\rho(\vec{x})d^3x. \tag{3.5}$$

3.2. General case: arbitrary Green's function.

Now let's choose a point, $\vec{r}_\gamma$, which is not necessarily inside $\mathcal{V}_\gamma$, and do a Taylor series expansion of $G(\vec{r} - \vec{x})$,

$$\begin{aligned} G(\vec{r} - \vec{x}) =& G((\vec{r} - \vec{r}_\gamma) - (\vec{x} - \vec{r}_\gamma)) \\ =& G|_{\vec{r}-\vec{r}_\gamma} - (x - r_\gamma)^i \partial_i G|_{\vec{r}-\vec{r}_\gamma} \\ &+ \frac{1}{2}(x - r_\gamma)^i (x - r_\gamma)^j \partial_i \partial_j G|_{\vec{r}-\vec{r}_\gamma} \\ &- \frac{1}{6}(x - r_\gamma)^i (x - r_\gamma)^j (x - r_\gamma)^k \partial_i \partial_j \partial_k G|_{\vec{r}-\vec{r}_\gamma} + \cdots \end{aligned} \tag{3.6}$$

Inserting this expression into the integral in Eqn. 3.5, the dependence of the integrand on the value of r can be factored out, and we have the following,

$$\begin{aligned} \phi_\gamma(r) =& M_{\gamma(0)} G|_{\vec{r}-\vec{r}_\gamma} - M^i_{\gamma(1)} \partial_i G|_{\vec{r}-\vec{r}_\gamma} \\ &+ \frac{1}{2} M^{ij}_{\gamma(2)} \partial_i \partial_j G|_{\vec{r}-\vec{r}_\gamma} - \frac{1}{6} M^{ijk}_{\gamma(3)} \partial_i \partial_j \partial_k G|_{\vec{r}-\vec{r}_\gamma} \cdots \end{aligned} \tag{3.7}$$

The gradient of the potential, $a_{\gamma i}$, is given by

$$\begin{aligned} a_{\gamma i}(r) =& - M_{\gamma(0)} \partial_i G|_{\vec{r}-\vec{r}_\gamma} + M^j_{\gamma(1)} \partial_i \partial_j G|_{\vec{r}-\vec{r}_\gamma} \\ &- \frac{1}{2} M^{jk}_{\gamma(2)} \partial_i \partial_j \partial_k G|_{\vec{r}-\vec{r}_\gamma} \\ &+ \frac{1}{6} M^{jkl}_{\gamma(3)} \partial_i \partial_j \partial_k \partial_l G|_{\vec{r}-\vec{r}_\gamma} - \cdots, \end{aligned} \tag{3.8}$$

where

$$M^{i_1\cdots i_n}_{\gamma(n)} = \int_{\mathcal{V}_\gamma} d^3x (x - r_\gamma)^{i_1}(x - r_\gamma)^{i_2}\ldots(x - r_\gamma)^{i_n}\rho(x) \tag{3.9}$$

are the multipole moments of the mass distribution in $\mathcal{V}_\gamma$. In the following, we will need to refer to the individual terms of the expansion, separately. Thus,

$$\phi_{\gamma(n)} = \frac{(-1)^n}{n!} M^{i_1\cdots i_n}_{\gamma(n)} \partial_{i_i}\cdots\partial_{i_n} G|_{\vec{r}-\vec{r}_\gamma}, \tag{3.10}$$

$$\vec{a}_{\gamma(n)i_0} = \frac{(-1)^{n+1}}{n!} M^{i_1\cdots i_n}_{\gamma(n)} \partial_{i_0}\partial_{i_i}\cdots\partial_{i_n} G|_{\vec{r}-\vec{r}_\gamma}, \tag{3.11}$$

and Eqns. 3.7 and 3.8 can be written succinctly as

$$\phi_\gamma(r) = \sum_{n=0}^{\infty} \phi_{\gamma(n)}(r), \tag{3.12}$$

$$\vec{a}_\gamma(r) = \sum_{n=0}^{\infty} \vec{a}_{\gamma(n)}(r). \tag{3.13}$$

If the multipole moments are known, then Eqns. 3.12 and 3.13 tell us how to evaluate the potential $\phi(r)$ and its gradient at any point, r, (assuming that the Taylor series converges). We have characterized the continuous distribution of mass $\rho(r)$ inside $\mathcal{V}_\gamma$ by a countable set of multipole tensors, $M_{\gamma(n)}$. If we have the $M_{\gamma(n)}$ in hand, then in order to evaluate the potential $\phi_\gamma(r)$, we need only evaluate G and its derivatives with one argument, $\vec{r} - \vec{r}_\gamma$, and perform some summations over indices. In many cases, this is vastly simpler than integrating G over the volume, $\mathcal{V}_\gamma$.

3.3. Spherically symmetric Green's function.

Although it is useful to have a completely general formulation of the method, in practice, Green's functions often obey special properties which make both the algorithm and the associated analysis much simpler. We begin our process of specialization by investigating spherically symmetric Green's functions.

If the Green's function is rotationally invariant, we can write it as

$$G(\vec{r}) = g(|\vec{r}|) = f(r^2/2), \tag{3.14}$$

from which follow its derivatives:

$$\begin{aligned}
\partial_i G =& f'(r^2/2) r_i, \\
\partial_i \partial_j G =& f''(r^2/2) r_i r_j + f'(r^2/2)\delta_{ij}, \\
\partial_i \partial_j \partial_k G =& f'''(r^2/2) r_i r_j r_k + f''(r^2/2)(r_i \delta_{jk} + 2permutations), \\
\partial_i \partial_j \partial_k \partial_l G =& f''''(r^2/2) r_i r_j r_k r_l + f'''(r^2/2)(r_i r_j \delta_{kl} + 5perm.), \\
& + f''(r^2/2)(\delta_{ij}\delta_{kl} + 2perm.), \\
\partial_i \partial_j \partial_k \partial_l \partial_m G =& f'''''(r^2/2) r_i r_j r_k r_l r_m + f''''(r^2/2)(r_i r_j r_k \delta_{lm} + 9perm.), \\
& + f'''(r^2/2)(r_i \delta_{jk}\delta_{lm} + 14perm.), \\
& \ldots .
\end{aligned} \tag{3.15}$$

If we define the "normalized" n^{th} derivative,

$$g^{(n)}(r) \equiv \frac{r^{2n}}{g(r)} \frac{d^n}{dx^n} f(x)|_{x=r^2/2}, \tag{3.16}$$

then the derivatives of G may be written explicitly in terms of inverse powers of $|\vec{r} - \vec{r}_\gamma|$ and the unit-vector pointing in the direction of $\vec{r} - \vec{r}_\gamma$. First, we define,

$$d \equiv |\vec{r} - \vec{r}_\gamma|, \tag{3.17}$$

$$\hat{e} \equiv \frac{\vec{r} - \vec{r}_\gamma}{d}. \tag{3.18}$$

In terms of d and $\hat{e}$, we have

$$\begin{aligned}\partial_{i_1}\ldots\partial_{i_n}G|_{\vec{r}-\vec{r}_\gamma} = \frac{g(d)}{d^n}\Big(&g^{(n)}(d)e_{i_1}\ldots e_{i_n}\\ &+ g^{(n-1)}(d)\Big(\delta_{i_1 i_2}e_{i_3}e_{i_4}\ldots e_{i_n}+\cdots \quad \binom{n}{2}\textit{permutations}\Big)\\ &+ g^{(n-2)}(d)\Big(\delta_{i_1 i_2}\delta_{i_3 i_4}e_{i_5}\ldots e_{i_n}+\cdots \quad \frac{1}{2!}\binom{n}{2\,2}\textit{permutations}\Big)\\ &+\cdots\Big).\end{aligned} \tag{3.19}$$

Alternatively, we can write Eqn. 3.19 as

$$\partial_{i_1}\ldots\partial_{i_n}G|_{\vec{r}-\vec{r}_\gamma} = \frac{g(d)}{d^n}\sum_{m=0}^{\leq\frac{n}{2}} g^{(n-m)}(d) \tag{3.20}$$

$$\times\left(\delta_{i_1 i_2}\cdots\delta_{i_{2m-1}i_{2m}}e_{i_{2m+1}}\cdots e_{i_n}+\cdots\frac{n!}{2^m(n-2m)!m!}\,\text{permutations}\right).$$

In order to insert Eqn. 3.20 into Eqns. 3.7 and 3.8, we introduce the following notation for the trace of the multipole tensor $M_{\gamma(n)}$ with l remaining indices,

$$M^{(l)i_1\cdots i_l}_{\gamma(n)} \equiv \delta_{i_{l+1}i_{l+2}}\cdots\delta_{i_{n-1}i_n}M^{i_1 i_2\cdots i_n}_{\gamma(n)}, \tag{3.21}$$

and the following for the contraction of $M^{(l)}_{\gamma(n)}$ with the "m^{th}" power of the unit-vector, $\hat{e}$,

$$\left\langle M^{(l)}_{\gamma(n)}|e^{(m)}\right\rangle^{i_1\cdots i_{l-m}} = M^{(l)i_1\cdots i_l}_{\gamma(n)}e_{i_{l-m+1}}\cdots e_{i_l}. \tag{3.22}$$

Note that under rotations, $\left\langle M^{(l)}_{\gamma(n)}|e^{(l)}\right\rangle$ transforms as a scalar and $\left\langle M^{(l)}_{\gamma(n)}|e^{(l-1)}\right\rangle$ transforms as a vector.

With this notation, Eqns. 3.7 and 3.8 become

$$\begin{aligned}\phi_{\gamma(n)}(r) =& \frac{(-1)^n g(d)}{d^n n!}\Big(\Big\langle M^{(n)}_{\gamma(n)} \Big| e^{(n)} \Big\rangle g^{(n)}(d) \\ &+ \binom{n}{2} \Big\langle M^{(n-2)}_{\gamma(n)} \Big| e^{(n-2)} \Big\rangle g^{(n-1)}(d) \\ &+ \frac{1}{2!}\binom{n}{2\,2} \Big\langle M^{(n-4)}_{\gamma(n)} \Big| e^{(n-4)} \Big\rangle g^{(n-2)}(d) \\ &+ \frac{1}{3!}\binom{n}{2\,2\,2} \Big\langle M^{(n-6)}_{\gamma(n)} \Big| e^{(n-6)} \Big\rangle g^{(n-3)}(d) \\ &+ \cdots \Big)\end{aligned} \tag{3.23}$$

and

$$\begin{aligned}\vec{a}_{\gamma(n)}(r) =& \frac{(-1)^{n+1} g(d)}{d^{n+1} n!}\Big(\Big\langle M^{(n)}_{\gamma(n)} \Big| e^{(n)} \Big\rangle g^{(n+1)}(d)\hat{e} \\ &+ n g^{(n)}(d) \Big\langle M^{(n)}_{\gamma(n)} \Big| e^{(n-1)} \Big\rangle \\ &+ \binom{n}{2} \Big\langle M^{(n-2)}_{\gamma(n)} \Big| e^{(n-2)} \Big\rangle g^{(n)}(d)\hat{e} \\ &+ \binom{n}{2\,1} g^{(n-1)}(d) \Big\langle M^{(n-2)}_{\gamma(n)} \Big| e^{(n-3)} \Big\rangle \\ &+ \frac{1}{2!}\binom{n}{2\,2} \Big\langle M^{(n-4)}_{\gamma(n)} \Big| e^{(n-4)} \Big\rangle g^{(n-1)}(d)\hat{e} \\ &+ \frac{1}{2!}\binom{n}{2\,2\,1} g^{(n-2)}(d) \Big\langle M^{(n-4)}_{\gamma(n)} \Big| e^{(n-5)} \Big\rangle \\ &+ \cdots \Big).\end{aligned} \tag{3.24}$$

Or, following Eqn. 3.20,

$$\phi_{\gamma(n)} = \frac{(-1)^n g(d)}{d^n n!} \sum_{m=0}^{m \le n/2} \frac{n!}{m! 2^m (n-2m)!} \Big\langle M^{(n-2m)}_{\gamma(n)} \Big| e^{(n-2m)} \Big\rangle g^{(n-m)}(d) \tag{3.25}$$

and

$$\vec{a}_{\gamma(n)} = \frac{(-1)^{n+1} g(d)}{d^{n+1} n!} \sum_{m=0}^{m \le n/2} \frac{n!}{m! 2^m (n-2m)!} \tag{3.26}$$

$$\left(\left\langle M_{\gamma(n)}^{(n-2m)} \middle| e^{(n-2m)} \right\rangle g^{(n+1-m)}(d) \hat{e} \right.$$

$$\left. + (n-2m) \left\langle M_{\gamma(n)}^{(n-2m)} \middle| e^{(n-2m-1)} \right\rangle g^{(n-m)}(d) \right).$$

3.4. A softened Newtonian potential.

We are now ready to consider a particular functional form for the Green's function. Often, astrophysical N-body simulations use a non-Newtonian potential which lacks the singularity at the origin, and makes time integration of the N-body equations considerably easier. The "Plummer" model is frequently used:

$$g(r) = -\frac{1}{R}, \tag{3.27}$$

$$R \equiv (r^2 + \epsilon^2)^{1/2}. \tag{3.28}$$

This Green's function behaves like Newtonian gravity for separations much greater than ϵ, but for small separations, the potential is not allowed to go to infinity. In physical terms, it is the potential energy around a "Plummer" sphere, i.e., a density distribution of the form,

$$\rho_{\mathbf{plummer}}(r) = \frac{3}{4\pi} \frac{\epsilon^2}{(r^2 + \epsilon^2)^{5/2}}. \tag{3.29}$$

It is a useful modification of the true potential because it eliminates the singularity at the origin, and limits the maximum potential energy, and hence orbital frequency, of a pair of bodies.

In the notation of Eqn. 3.15,

$$f(x) = -(2x + \epsilon^2)^{-1/2}. \tag{3.30}$$

The derivatives of this particular $f(x)$ are easy to evaluate by induction,

$$f^{(n)}(x) = -(2n-1)!!(-1)^n (2x+\epsilon^2)^{-(2n+1)/2} = -(2n-1)!!(-1)^n R^{-(2n+1)} \tag{3.31}$$

and

$$g^{(n)}(r) = (2n-1)!!(-1)^n \left(\frac{r}{R}\right)^{2n}. \tag{3.32}$$

We also define the "softened unit-vector", $\vec{h}$,

$$\vec{h} = \hat{e}\left(\frac{r}{R}\right) = \frac{\vec{r}}{(r^2+\epsilon^2)^{1/2}}. \tag{3.33}$$

Inserting Eqn. 3.32 into Eqns. 3.25 and 3.26, we immediately have

$$\phi_{\gamma(n)}(r) = \frac{g(d)}{R^n} \sum_{m=0}^{m\leq n/2} \frac{(-1)^m(2n-2m-1)!!}{m!2^m(n-2m)!} \left\langle M_{\gamma(n)}^{(n-2m)} | h^{(n-2m)} \right\rangle \tag{3.34}$$

$$\begin{aligned} \vec{a}_{\gamma(n)}(r) = & \frac{g(d)}{R^{n+1}} \sum_{m=0}^{m\leq n/2} \frac{(-1)^m(2n-2m-1)!!}{m!2^m(n-2m)!} \\ & \times \left((2n-2m+1) \left\langle M_{\gamma(n)}^{(n-2m)} | h^{(n-2m)} \right\rangle \vec{h} \right. \\ & \left. - (n-2m) \left\langle M_{\gamma(n)}^{(n-2m)} | h^{(n-2m-1)} \right\rangle \right). \end{aligned} \tag{3.35}$$

For reference, the summations and combinatoric factors are presented for the first few values of n:

$$\begin{aligned} \phi_{(0)} &= -R^{-1} M_{(0)}, \\ \phi_{(1)} &= -R^{-2} \left\langle M_{\gamma(1)}^{(1)} | h^{(1)} \right\rangle, \\ \phi_{(2)} &= -R^{-3} \frac{1}{2} \left(3 \left\langle M_{\gamma(2)}^{(2)} | h^{(2)} \right\rangle - \left\langle M_{\gamma(2)}^{(0)} | h^{(0)} \right\rangle \right), \\ \phi_{(3)} &= -R^{-4} \frac{1}{6} \left(15 \left\langle M_{\gamma(3)}^{(3)} | h^{(3)} \right\rangle - 9 \left\langle M_{\gamma(3)}^{(1)} | h^{(1)} \right\rangle \right), \\ \phi_{(4)} &= -R^{-5} \frac{1}{24} \left(105 \left\langle M_{\gamma(4)}^{(4)} | h^{(4)} \right\rangle - 90 \left\langle M_{\gamma(4)}^{(2)} | h^{(2)} \right\rangle + 9 \left\langle M_{\gamma(4)}^{(0)} | h^{(0)} \right\rangle \right), \end{aligned} \tag{3.36}$$

$$
\begin{aligned}
\vec{a}_{(0)i} &= -R^{-2} M_{(0)} \vec{h}, \\
\vec{a}_{(1)i} &= -R^{-3}(3\left\langle M^{(1)}_{\gamma(1)} | h^{(1)} \right\rangle)\vec{h} - \left\langle M^{(1)}_{\gamma(1)} | h^{(0)} \right\rangle), \\
\vec{a}_{(2)i} &= -R^{-4}\frac{1}{2}(15\left\langle M^{(2)}_{\gamma(2)} | h^{(2)} \right\rangle \vec{h} - 6\left\langle M^{(2)}_{\gamma(2)} | h^{(1)} \right\rangle \\
&\qquad - 3\left\langle M^{(0)}_{\gamma(2)} | h^{(0)} \right\rangle \vec{h}), \\
\vec{a}_{(3)i} &= -R^{-5}\frac{1}{6}\Big(105\left\langle M^{(3)}_{\gamma(3)} | h^{(3)} \right\rangle \vec{h} - 45\left\langle M^{(3)}_{\gamma(3)} | h^{(2)} \right\rangle \\
&\qquad - 45\left\langle M^{(1)}_{\gamma(3)} | h^{(1)} \right\rangle \vec{h} + 9\left\langle M^{(1)}_{\gamma(3)} | h^{(0)} \right\rangle \Big), \\
\vec{a}_{(4)i} &= -R^{-6}\frac{1}{24}\Big(945\left\langle M^{(4)}_{\gamma(4)} | h^{(4)} \right\rangle \vec{h} - 420\left\langle M^{(4)}_{\gamma(4)} | h^{(3)} \right\rangle \\
&\qquad - 630\left\langle M^{(2)}_{\gamma(4)} | h^{(2)} \right\rangle \vec{h} + 180\left\langle M^{(2)}_{\gamma(4)} | h^{(1)} \right\rangle \\
&\qquad + 45\left\langle M^{(0)}_{\gamma(4)} | h^{(0)} \right\rangle \vec{h}\Big).
\end{aligned}
\tag{3.37}
$$

Equations 3.36 and 3.37 are implemented directly in the `PairInteraction` phase of the BH algorithm. It is worth noting that these equations are slightly different from those used by Hernquist,[18] who, apparently, failed to account for the terms involving the trace of the various multipoles. This almost certainly explains his results which indicate a failure of the second order correction for separations approaching ϵ.

3.5. Pure Newtonian potential.

Now let us take $\epsilon = 0$. Thus, we have finally specialized to the case of pure Newtonian gravity. The Green's function now has the following important property, away from the origin

$$\vec{\nabla}^2 G = 0; \qquad \vec{r} \neq 0. \tag{3.38}$$

In component notation,

$$\delta^{ij}\partial_i\partial_j G(r) = 0; \qquad \vec{r} \neq 0. \tag{3.39}$$

Since derivatives may be taken in any order, we immediately have

$$\delta^{i_\alpha i_\beta}\partial_{i_1}\partial_{i_2}\cdots\partial_{i_n}G(r)=0; \qquad \vec{r}\neq 0; \quad \alpha,\beta\leq n. \tag{3.40}$$

This allows us to add arbitrary multiples of the Kronecker-delta to any of the multipoles, without changing the value of $\phi_{(n)}$ or $\vec{a}_{(n)}$, computed according to Eqns. 3.10 and 3.11. That is, if we define

$$Q^{i_1\ldots i_n}_{(n)} \equiv M^{i_1\ldots i_n}_{(n)} - (\delta^{i_1 i_2}C^{i_3\ldots i_n}_{(n)} + \text{permutations}), \tag{3.41}$$

where $C_{(n)}$ is any fully symmetric rank $n-2$ tensor whatsoever, then

$$\phi_{(n)} = \frac{(-1)^n}{n!}Q^{i_1\ldots i_n}_{(n)}\partial_{i_1}\cdots\partial_{i_n}G|_{\vec{r}-\vec{r}_\gamma}, \tag{3.42}$$

$$a_{(n)i_0} = \frac{(-1)^{n+1}}{n!}Q^{i_1\ldots i_n}_{(n)}\partial_{i_0}\cdots\partial_{i_n}G|_{\vec{r}-\vec{r}_\gamma}. \tag{3.43}$$

In particular, we may choose $C_{(n)}$ so that $Q_{(n)}$ is completely trace-free, on any pair of indices. We shall refer to the tensor, $Q_{(n)}$, that results from such a choice of $C_{(n)}$ as the "reduced multipole tensor." Since $M_{(n)}$ is symmetric under interchange of indices, and Eqn. 3.41 is manifestly symmetric by construction, $Q_{(n)}$ is a completely trace-free symmetric tensor of rank n. $C_{(n)}$ is readily evaluated in terms of $M_{(n)}$ by simply taking traces of both sides of Eqn. 3.41. We show the results of this procedure for $C_{(2)}$ through $C_{(6)}$:

$$\begin{aligned}
C_{(2)} &= \frac{1}{3}M^{(0)}_{\gamma(2)},\\
C^{i}_{(3)} &= \frac{1}{5}M^{(1)i}_{\gamma(3)},\\
C^{ij}_{(4)} &= \frac{1}{7}(M^{(2)ij}_{\gamma(4)} - \frac{1}{10}M^{(0)}_{\gamma(4)}\delta^{ij}),\\
C^{ijk}_{(5)} &= \frac{1}{9}(M^{(3)ijk}_{\gamma(5)} - \frac{1}{14}(M^{(1)i}_{\gamma(5)}\delta^{jk} + 2\,\text{permutations}))),\\
C^{ijkl}_{(6)} &= \frac{1}{11}(M^{(4)ijkl}_{\gamma(6)} - \frac{1}{18}((M^{(2)ij}_{\gamma(6)} - \frac{1}{21}M^{(0)}_{\gamma(6)}\delta^{ij})\delta^{kl} + 5\,\text{permutations})).
\end{aligned} \tag{3.44}$$

The reduced multipole moments are extremely useful, since they make it much easier to evaluate $\phi_{(n)}$ and $a_{(n)}$. First we replace $M_{(n)}$ with $Q_{(n)}$ in Eqns. 3.10 and 3.11. Equations 3.25 and 3.26 follow just as they did before, only with M replaced by Q. But we know that the trace of Q, on any pair of indices, vanishes by explicit construction. In other words,

$$Q^{(l)}_{\gamma(n)} = 0; \qquad l < n. \tag{3.45}$$

This means that all but the leading terms in Eqns. 3.25 and 3.26 vanish, and we have

$$\phi_{\gamma(n)}(r) = \frac{(-1)^n g(d)}{d^n n!} \left\langle Q_{\gamma(n)} \middle| e^{(n)} \right\rangle g^{(n)}(d), \tag{3.46}$$

$$\vec{a}_{\gamma(n)}(r) = \frac{(-1)^{n+1} g(d)}{d^{n+1} n!} \Big(\left\langle Q_{\gamma(n)} \middle| e^{(n)} \right\rangle g^{(n+1)}(d)\hat{e} + n g^{(n)}(d) \left\langle Q_{\gamma(n)} \middle| e^{(n-1)} \right\rangle \Big). \tag{3.47}$$

Using the explicit expressions for the $g^{(n)}$, we have

$$\phi_{\gamma(n)}(r) = -\frac{(2n-1)!!}{n!} \frac{1}{d^{n+1}} \left\langle Q_{\gamma(n)} \middle| e^{(n)} \right\rangle, \tag{3.48}$$

$$\vec{a}_{\gamma(n)}(r) = -\frac{(2n-1)!!}{n!} \frac{1}{d^{n+2}} \left((2n+1) \left\langle Q_{\gamma(n)} \middle| e^{(n)} \right\rangle \hat{e} - n \left\langle Q_{\gamma(n)} \middle| e^{(n-1)} \right\rangle \right). \tag{3.49}$$

Recall from Eqns. 3.17 and 3.18 that the vector, $\hat{e}$, is a unit-vector, and points in the direction of the vector, $\vec{r} - \vec{r}_\gamma$, and that d is the magnitude of $\vec{r} - \vec{r}_\gamma$.

Equations 3.48 and 3.49 give an explicit form for the multipole approximation

for a Newtonian potential. We tabulate the first few cases for reference:

$$
\begin{aligned}
\phi_{(0)} &= -R^{-1} M_{(0)}, \\
\phi_{(1)} &= -R^{-2} \left\langle Q_{\gamma(1)} | e^{(1)} \right\rangle, \\
\phi_{(2)} &= -R^{-3} \frac{3}{2} \left(\left\langle Q_{\gamma(2)} | e^{(2)} \right\rangle \right), \\
\phi_{(3)} &= -R^{-4} \frac{5}{2} \left(\left\langle Q_{\gamma(3)} | e^{(3)} \right\rangle \right), \\
\phi_{(4)} &= -R^{-5} \frac{35}{8} \left(\left\langle Q_{\gamma(4)} | e^{(4)} \right\rangle \right), \\
\phi_{(5)} &= -R^{-5} \frac{63}{8} \left(\left\langle Q_{\gamma(5)} | e^{(5)} \right\rangle \right), \\
\phi_{(6)} &= -R^{-5} \frac{231}{16} \left(\left\langle Q_{\gamma(6)} | e^{(6)} \right\rangle \right),
\end{aligned}
\tag{3.50}
$$

$$
\begin{aligned}
\vec{a}_{(0)i} &= -R^{-2} M_{(0)} \hat{e}, \\
\vec{a}_{(1)i} &= -R^{-3} (3 \left\langle Q_{\gamma(1)} | e^{(1)} \right\rangle) \hat{e} - \left\langle Q_{\gamma(1)} | e^{(0)} \right\rangle), \\
\vec{a}_{(2)i} &= -R^{-4} \frac{3}{2} (5 \left\langle Q_{\gamma(2)} | e^{(2)} \right\rangle \hat{e} - 2 \left\langle Q_{\gamma(2)} | e^{(1)} \right\rangle), \\
\vec{a}_{(3)i} &= -R^{-5} \frac{5}{2} \left(7 \left\langle Q_{\gamma(3)} | e^{(3)} \right\rangle \hat{e} - 3 \left\langle Q_{\gamma(3)} | e^{(2)} \right\rangle \right) \\
\vec{a}_{(4)i} &= -R^{-6} \frac{35}{8} \left(9 \left\langle Q_{\gamma(4)} | e^{(4)} \right\rangle \hat{e} - 4 \left\langle Q_{\gamma(4)} | e^{(3)} \right\rangle \right), \\
\vec{a}_{(5)i} &= -R^{-6} \frac{63}{8} \left(11 \left\langle Q_{\gamma(4)} | e^{(4)} \right\rangle \hat{e} - 5 \left\langle Q_{\gamma(4)} | e^{(3)} \right\rangle \right), \\
\vec{a}_{(6)i} &= -R^{-6} \frac{231}{16} \left(13 \left\langle Q_{\gamma(4)} | e^{(4)} \right\rangle \hat{e} - 6 \left\langle Q_{\gamma(4)} | e^{(3)} \right\rangle \right),
\end{aligned}
\tag{3.51}
$$

We shall discuss methods of evaluating these expressions in Chapter 4.

4. Practical Aspects of the Barnes-Hut Algorithm.

We are now ready to analyze the Barnes-Hut algorithm in some detail. In this chapter, we attempt to bridge the gap between the the data structures and mathematics of Chapters 2 and 3 and the BH algorithm, as implemented on a serial computer. Issues that arise from parallel computers will be deferred until Chapter 7. Of particular interest are methods for computing the multipole moments of a system, the amount of storage space required to store the components, the number of operations necessary to compute the multipoles themselves and to compute the multipole approximations derived in Chapter 3. The BH algorithm is described in Chapter 1, and a method for constructing a BH tree is described in Chapter 2. Here, we discuss the form of the data in the BH tree, and how it is used.

4.1. Space requirement for storing multipoles.

Before we analyze the algorithm for computing the multipoles, we need to know how much space will be required to store them. Each multipole of order n is a rank n fully symmetric tensor. Such a tensor is completely specified by only a small fraction of its components. We need to store only the following components,

$$M_{(n)}^{i_1 i_2 \cdots i_n} \ni 3 \geq i_1 \geq i_2 \geq \cdots \geq i_n \geq 1. \tag{4.1}$$

Then, when any of the components is needed in an expression such as Eqn. 3.7, we can permute the desired indices to obtain the ordering of Eqn. 4.1 and find the stored element. A simple combinatoric argument reveals that there are exactly

$$\binom{n+2}{2} = \frac{(n+2)(n+1)}{2} \tag{4.2}$$

distinct components to $M_{(n)}$, and storage is required for each of them. In order to

store all multipoles up to some maximum order, p, we will need storage equal to

$$\sum_{n=0}^{p} \binom{n+2}{2} = \binom{p+3}{3}. \tag{4.3}$$

Thus, the storage requirement for the whole tree of multipoles is

$$C_{avg}\binom{p+3}{3} = O(p^3), \tag{4.4}$$

where C_{avg} is the number of internal cells in the tree. We found in Chapter 2 that C_{avg} is proportional to N. Equation 4.3 is a special case of a very useful identity [36] which will be needed several times in the following,

$$\sum_{0 \le k \le r} \binom{r-k}{m}\binom{s+k}{n} = \binom{r+s+1}{m+n+1}. \tag{4.5}$$

Equation 4.4 is an example of the case when $m = 0$, $n = 2$, $r = p$, $s = 2$, and $b = 0$.

4.2. Computing multipoles.

Two approaches are available for computing the multipole moments of the cells in the BH tree. The methods have different behavior with respect to the number of bodies in the cell, and the highest order multipole required. As we shall see, it is easy to select between the two methods as the calculation proceeds, according to which will produce the result in the shortest time.

4.2.1. Direct summation.

One possibility is to simply identify all the bodies in each cell, and compute the multipole moments as

$$M^{i_1 \cdots i_n}_{\gamma(n)} = \sum_{x_p \in \mathcal{V}_\gamma} m_p (x_p^{i_1} - r_\gamma^{i_1}) \cdots (x_p^{i_n} - r_\gamma^{i_n}). \tag{4.6}$$

Let us investigate how many operations are required to compute the multipole of one cell using Eqn. 4.6. Equation 4.6 requires computing $\binom{n+2}{2}$ terms. If the

terms are computed in order of increasing n, then each new term is the product of a component of $\vec{x} - \vec{r}_\gamma$ and a previously computed term. Thus, to compute all multipoles up to order, p, for a single particle, and a single $\vec{r}_\gamma$, using Eqn. 4.6 requires

$$\sum_{n=0}^{p} \binom{n+2}{2} = \binom{p+3}{3} \tag{4.7}$$

multiplications. Since each body is computed independently, the total number of operations required to compute a multipole by direct summation is

$$N_\gamma \binom{p+3}{3}, \tag{4.8}$$

where N_γ is the number of bodies contained within $\mathcal{V}_\gamma$.

4.2.2. Parallel axis theorem.

An alternative to direct summation of all bodies contained in a node, is to compute the multipole moment of a node using only the information available in its eight children. If we had a procedure which would tell us the multipole moments of a cell, given the moments of its children, then we could proceed with a bottom-up traversal of the tree and compute the multipole moments of each cell from the information already computed for its children. The appropriate mathematical expression is a generalized parallel axis theorem.[37]

We note that the multipole moments act as tensors under rotations, and obey some useful transformation rules under translation, as well. If $M_{\gamma(n)}$ are the multipoles of a mass distribution defined around a point, $\vec{r}_\gamma$, and $M'_{\gamma(n)}$ represents the same distribution, defined around the point, $\vec{r}'_\gamma = \vec{r}_\gamma + \vec{y}$, then by definition,

$$M'^{i_1 i_2 \ldots i_n}_{\gamma(n)} = \int_{\mathcal{V}} \rho(x)(x - r_\gamma - y)^{i_1} \cdots (x - r_\gamma - y)^{i_n}. \tag{4.9}$$

Factoring out powers of y we get

$$M'^{i_1 i_2 \ldots i_p}_{\gamma(p)} = M^{i_1 i_2 \ldots i_p}_{\gamma(p)} - (y^{i_1} M^{i_2 \ldots i_p}_{\gamma(n-1)} + perm.) + (y^{i_1} y^{i_2} M^{i_3 \ldots i_p}_{\gamma(n-2)} + perm.) - \ldots \tag{4.10}$$

Furthermore, the multipole moments of two distributions, defined around the same point, may be added term-wise to obtain the multipole expansion of the combined mass distributions. Equation 4.10 is a generalization of the Parallel Axis Theorem. For $p = 2$, and $\vec{r}_\gamma$ located at the center-of-mass of the $\mathcal{V}_\gamma$, Eqn. 4.10 reduces to

$$M'^{ij}_{(2)} = M^{ij}_{(2)} + y^i y^j M_{(0)}. \tag{4.11}$$

Note that for the purposes of defining the multipole moments, there is no logical reason why the point, $\vec{r}_\gamma$, must lie in the region, $\mathcal{V}_\gamma$. Thus, Eqn. 4.10 can be used to "translate" the multipoles of several disjoint volumes, e.g., the daughter cells of a cell, to a common point, e.g., the $\vec{r}_\gamma$ of the parent, where they can be added, term-wise, to give the multipole of the mass in the set of volumes.

In order to compare Eqn. 4.10 with direct summation, we analyze how many operations are required to translate a set of multipole moments up through some order, p, using Eqn. 4.10. To implement Eqn. 4.10 for all orders up to p, we first compute all monomials formed by repeated products of the components of the vector, $\vec{y}$, up to order, p. As with the multipole moments themselves, there are exactly $\binom{n+2}{2}$ such monomials of order, n, making a total of $\binom{p+3}{3}$ up to order, p. We rewrite Eqn. 4.10 schematically as follows,

$$M_{\gamma(p)} = \sum_{m=0}^{m=n} (\text{Product of } m \text{ components of } y) \otimes M_{\gamma(n-m)}. \tag{4.12}$$

Let us assume that all of the "products of m components of y" are available. There are exactly $\binom{m+2}{2}$ such products. Each of them must be multiplied with each of the components of $M_{\gamma(n-m)}$, and added into the result, giving us a total operation count of

$$\sum_{m=0}^{m=n} \binom{m+2}{2}\binom{n-m+2}{2} = \binom{n+5}{5}. \tag{4.13}$$

Each term is evaluated as a single product and is added to a stored component of $M_{\gamma(n)}$. Thus, the number of operations required to translate $M_{\gamma(n)}$ is equal to $2\binom{n+5}{5}$.

To translate all multipoles up through order p, using Eqn. 4.10 requires

$$\sum_{n=0}^{n=p} 2\binom{n+5}{5} = 2\binom{p+6}{6} \tag{4.14}$$

operations. We note that this result is entirely independent of the number of bodies within $\mathcal{V}_\gamma$.

4.2.3. Hybrid algorithm.

The best way to compute all the multipoles in a BH tree is a hybrid of the two methods just described. A recursive routine, which traverses the hierarchical tree of volumes from the bottom up is shown in Code 4.1. For each child of a given node, either `TranslateMultipoles` is called once, or `BodyMulti` is called repeatedly to produce the desired result.

```
GenerateMultipoles(cell)
    for(each child of cell)
        GenerateMultipoles(child)
    endfor
    ChooseRgamma(cell)
    if(cell is terminal)
        cell.nbody = number of bodies in cell
    else
        for(each child of cell)
            TranslateMultipole(cell, child)
        endfor
    endif
endfunc
```

Code 4.1. Function `GenerateMultipoles` to fill in multipole moments of the cells in a BH tree.

The information needed to choose $\vec{r}_\gamma$ may not be known until the multipoles of the daughter cells have been computed, so the call to `ChooseRgamma` is delayed until after the multipole moments of the daughter cells have been computed.

```
TranslateMultipole(cell, child)
    if( child.nbody < N_cutoff or child is terminal )
        DirectSum(cell, child)
    else
        ParAxis(cell, child)
    endif
    cell.nbody += child.nbody
endfunc
```

Code 4.2. Function `TranslateMultipole` to compute the contribution of the contents of a child to multipole moment of a cell.

```
DirectSum(cell, child)
    if( child is terminal )
        for( each body, b, in child )
            BodyMulti(cell, b)
            cell.nbody += 1
        endfor
    else
        for( each grandchild of child )
            DirectSum(cell, grandchild)
        endfor
    endif
endfunc
```

Code 4.3. Function `DirectSum` to compute the contribution of a child by direct summation of all contained bodies, using Eqn. 4.6.

The two functions, `BodyMulti` and `ParAxis`, are direct implementations of Eqns. 4.6 and 4.10. Code 4.1 contains an if-clause which decides whether to use Eqn. 4.6 or 4.10. based on the number of bodies contained in the child which is being translated. The reason for this is that application of Eqn. 4.10 requires constant time, while application of Eqn. 4.6 requires time proportional to the number of bodies contained within the child. There is a particular value of the number of bodies in the child, N_{cutoff}, above which Eqn. 4.10 is faster and below which Eqn. 4.6 is faster. The exact value of N_{cutoff}, of course, depends on details of the implementation. We may estimate the value of N_{cutoff} from the operation

counts implied by Eqns. 4.6 and 4.10.

$$N_{cutoff}\binom{p+3}{3} \approx 2\binom{p+6}{6}, \tag{4.15}$$

or

$$N_{cutoff} \approx \frac{1}{10}\binom{p+6}{3}. \tag{4.16}$$

In practice, N_{cutoff} should be determined empirically from timing data, but Eqn. 4.16 may be used if timing data is unavailable. In most of our production calculations, $p = 2$, and Eqn. 4.16 suggests $N_{cutoff} \approx 3$. In fact, we used $N_{cutoff} = 1$, which allows for some simplification of Code 4.3. This choice is almost certainly not optimal, but is not so far from optimal as to contribute significantly to total running time.

4.2.4. Operation count.

We now estimate the total running time when `GenerateMultipoles` is applied to an entire BH tree. Clearly, the mass and position of each Body effects the multipole moment of each cell which is an ancestor of the Body. For ancestors with fewer than N_{cutoff} descendants, the effect is computed by `DirectSum`. For ancestors with more than N_{cutoff} descendants, the effect is implicitly computed by `ParAxis`.

We first estimate the number of times `BodyMulti` is executed. Recall from Section 2.6, that each terminal node of the tree is expected to have approximately $\frac{m}{2}$ bodies. Furthermore, as we ascend the tree, we can expect to find eight times as many siblings at each higher level. Therefore, the expected number of parents with fewer than N_{cutoff} descendents is approximately

$$\log_8\left(\frac{2N_{cutoff}}{m}\right), \tag{4.17}$$

and the total number of times `BodyMulti` is executed is estimated to be

$$N\log_8\left(\frac{2N_{cutoff}}{m}\right). \tag{4.18}$$

In order to estimate the number of times `ParAxis` is executed, we simply need to estimate the number of cells which have more than N_{cutoff} descendants. The Parallel Axis Theorem is applied to each such cell exactly once, to translate the multipole from the $\vec{r}_\gamma$ of the cell to the $\vec{r}_\gamma$ of its parent. The number of cells with more than N_{cutoff} descendants is exactly equal to the number of non-terminal cells in a tree constructed with the parameter m, set equal to N_{cutoff}. In Section 2.9 we found that the expected number of non-terminal cells in a BH tree is

$$C_{avg} = \frac{c_m N}{m}. \tag{4.19}$$

Thus, the expected number of internal nodes in a tree with $m = N_{cutoff}$, and hence, the expected number of executions of `ParAxis` is approximately

$$\frac{c_{N_{cutoff}} N}{N_{cutoff}}. \tag{4.20}$$

Table 2.1 tabulates values of c_m for small values of m. The value of c_m increases slowly with increasing m, but for $m < 20$, the value of c_m is well below unity.

We let $T_{ParAxis}$ be the time required to execute `ParAxis` and $T_{BodyMulti}$ be the time required to execute `BodyMulti`, and we assume that N_{cutoff} has been chosen optimally, so that

$$N_{cutoff} = \frac{T_{ParAxis}}{T_{BodyMulti}}. \tag{4.21}$$

Then the total time spent in `GenerateMultipoles` is

$$\begin{aligned} T_{GenerateMultipoles} &\approx c_{N_{cutoff}} \frac{N}{N_{cutoff}} T_{ParAxis} + N \log_8 \left(\frac{2N_{cutoff}}{m} \right) T_{BodyMulti} \\ &= N T_{BodyMulti} \left(c_{N_{cutoff}} + \log_8 \left(\frac{2N_{cutoff}}{m} \right) \right). \end{aligned} \tag{4.22}$$

Thus, the time to construct the multipole tree is linear in N, with a constant of proportionality, i.e., $T_{BodyMulti}$, that scales with the maximum multipole order,

p, as

$$\binom{p+3}{3}. \tag{4.23}$$

4.3. Choosing $\vec{r}_\gamma$.

We must first choose an $\vec{r}_\gamma$ and then find the corresponding multipole moments for each cell in the tree. Most hierarchical N-body methods take $\vec{r}_\gamma$ to be the center-of-mass of the cell.[9, 16, 32, 38] This choice has the advantage of causing the $n = 1$ multipole, i.e., the dipole moment, to vanish identically. That is

$$\vec{r}_\gamma = \frac{1}{M_{(0)}} \int_{\mathcal{V}} \vec{x}\rho(x)d^3x = \frac{1}{M_0} \sum_{x_p \in \mathcal{V}_\gamma} m_p \vec{x_p}, \tag{4.24}$$

implies

$$M^i_{(1)} = \int_{\mathcal{V}} (x^i - r^i_\gamma)\rho(x)d^3x = 0. \tag{4.25}$$

This choice appears to have the disadvantage that the magnitude of higher multipoles might be larger than with some other choice of $\vec{r}_\gamma$. Greengard[13] takes $\vec{r}_\gamma$ to be in the geometric center of each cell. Such a choice affords the best strict bound on the magnitude of an arbitrary term in the multipole expansion. This is evident from Chapter 5, where the maximum error is seen to depend on a power of

$$b \equiv \sup_{x \in \mathcal{V}_\gamma} |\vec{x} - \vec{r}_\gamma|. \tag{4.26}$$

The quantity b, and hence the maximum possible error, is minimized by placing $\vec{r}_\gamma$ in the geometric center of $\mathcal{V}_\gamma$, like Greengard, rather than at the center of mass of $\mathcal{V}_\gamma$, as is more common. On the other hand, the magnitude of the quadrupole moment is minimized by taking $\vec{r}_\gamma$ to be the center of mass. If the calculation will be to limited to low orders, e.g., only terms up to quadrupole will be computed, then it is best to use the center of mass for $\vec{r}_\gamma$. If the calculation will be include high order multipoles then it is best to follow Greengard and use the geometric center of the $\mathcal{V}_\gamma$. The center of mass is especially good when one terminates the

multipole expansion after only one term, in which case, one gets the dipole term "for free," and the leading correction is the quadrupole term.

The cost of computing the center of mass of each cell in the tree is easily accounted for. The computation of the center of mass is comparable in structure and expense to the computation of the dipole moment (compare Eqn. 4.24 with Eqn. 4.6, $n = 1$). Thus, we have a situation in which we either compute the center of mass, but we don't compute $M_{(1)}$, because of Eqn. 4.25, or we compute $M_{(1)}$, but need not compute the center of mass. In either case, the work has already been accounted for in the analysis of Section 4.2.

4.4. Evaluating $\phi_{\gamma(n)}(\boldsymbol{r})$ and $\vec{a}_{\gamma(n)}(\boldsymbol{r})$.

Once a suitable set of multipole moments has been computed and stored, the BH algorithm computes $\phi_{\gamma(n)}(r)$ and $\vec{a}_{\gamma(n)}(r)$ repeatedly for different values of r and for the various cells, $\mathcal{V}_\gamma$. We now estimate the number of operations in each such body-cell interaction, i.e., in each evaluation of Eqns. 3.10, 3.25, or 3.48. As with the mathematical derivation, certain assumptions about the Green's function and its derivatives play a significant role in the results of this section. We treat the same cases as in Chapter 3.

4.4.1. Arbitrary Green's function.

Since we make no assumptions about the Green's function, there is no choice but to evaluate Eqn. 3.10 exactly as written. The right-hand side of each of these equations is a sum of 3^n terms, each of which is of unknown complexity (i.e., it includes the evaluation of one of the n^{th} partial derivatives of the Green's function.) Computing Eqn. 3.10 for each order n up through p, thus requires at least

$$\sum_{n=0}^{p} 3^n \approx \frac{3^{p+1}}{2} \tag{4.27}$$

multiplications, and a like number of evaluations of the derivative of the Green's function.

4.4.2. Spherically symmetric Green's function.

As we saw in the mathematical analysis, it is possible to make considerable simplifications if the Green's function is spherically symmetric. Now, we must evaluate Eqn. 3.25 for a particular value of d and $\hat{e}$. A detailed accounting of the number of operations implied by Eqn. 3.25 is not particularly useful, as a practical implemenation of any particular Green's function can certainly be optimized once the properties of the Green's function and its derivatives are known. Nevertheless, it is worthwhile to obtain a bound on the number of operations required to compute repeated inner products like $\left\langle M^{(m)}_{\gamma(n)} | e^{(m)} \right\rangle$. Let us assume that we have first computed all possible outer products like

$$e_x^{n_x} e_y^{n_y} e_z^{n_z} \text{ where, } \qquad n_x + n_y + n_z = m. \tag{4.28}$$

Then the computation of the inner product, $\left\langle M^{(m)}_{\gamma(n)} | e^{(m)} \right\rangle$, requires only a summation of products of a component of $M^{(m)}_{\gamma(n)}$ with one of the outer products of components of $\hat{e}$. There are precisely $\binom{m+2}{2}$ such terms, so the number of operations required to compute $\left\langle M^{(m)}_{\gamma(n)} | e^{(m)} \right\rangle$ is proportional to

$$\binom{m+2}{2}. \tag{4.29}$$

From Eqn. 3.25 the computation of $\phi_{\gamma(n)}$ is a sum of $n/2$ terms, each of which consists of a combinatoric factor, an evaluation of $g^{(m)}(d)$ and an inner product like $\left\langle M^{(m)}_{\gamma(n)} | e^{(m)} \right\rangle$. If we assume the time spent evaluating $n/2$ instances of $g^{(m)}(d)$ and the combinatoric factor is negligible, then the total number of operations is

$$\sum_{m=0}^{<n/2} \binom{n-2m+2}{2}. \tag{4.30}$$

It is easy to show that this expression is always less than

$$\frac{1}{2}\binom{n+4}{3}, \tag{4.31}$$

indicating that the evaluation of $\phi_{\gamma(n)}$ requires time cubic in n.

Inspection of Eqn. 3.26 reveals that terms which are combined to compute $\vec{a}_{\gamma(n)}$ have a very similar formal structure to those used in the computation of $\phi_{\gamma(n)}$. As there are two separate terms on the right-hand side of Eqn. 3.26, we expect that the computation of $\vec{a}_{\gamma(n)}$ will be about twice as costly as that for $\phi_{\gamma(n)}$.

We may now use Eqn. 4.31 to estimate the total work associated with computing $\phi_\gamma(n)$ and $\vec{a}_{\gamma(n)}$ for all values of n up through p. The total work requires less than

$$\frac{1}{2}\sum_{n=0}^{p}\binom{n+4}{3} = \frac{1}{2}\binom{p+5}{4} \tag{4.32}$$

operations, indicating that, in general, a full multipole interaction up through order p, for a spherically symmetric Green's function, requires time proportional to p^4.

We have assumed, in this analysis, that the products of components of $\hat{e}$, as in Eqn. 4.28, were "pre-computed" before attempting to evaluate expressions like $\left\langle M^{(m)}_{\gamma(n)} \middle| e^{(m)} \right\rangle$. In a complete evaluation of all orders through p, we can compute the products of elements of $\hat{e}$ only once, and ammortize the cost over all orders, n. The products of the components of $\hat{e}$ are essentially the same as the products necessary to compute the multipole moments of a point mass. In Section 4.2, we found that this calculation required approximately

$$2\binom{p+3}{3} \tag{4.33}$$

operations. It is clear that, for a general spherically symmetric Green's function, the "pre-computation" of products of $\hat{e}$ is a negligible expense, compared to the computation of the ϕ_γ and $\vec{a}_{\gamma(n)}$.

4.4.3. Newtonian potentials.

If the Green's function represents a softened Newtonian potential, i.e., Eqn. 3.27, the derivatives, $g^{(n)}(d)$, are particularly easy to evaluate. In fact, all dependence

on $g^{(n)}(d)$ in Eqns. 3.34 and 3.35 is obviously contained in the "softened unit-vector," $\vec{h}$. It is clear that the considerations of the previous section apply, except that the unit-vector, $\hat{e}$, should be replaced by the "softened unit-vector," $\vec{h}$, in all expressions like $\left\langle M_{\gamma(n)}^{(m)} | h^{(m)} \right\rangle$. In this way, only combinatoric factors and expressions like $\left\langle M_{\gamma(n)}^{(m)} | h^{(m)} \right\rangle$ need to be evaluated, and the total time required for a full evaluation of all multipoles up through order, p is proportional to

$$\binom{p+5}{4}. \tag{4.34}$$

The pure Newtonian case allows for even more simplification, however. Equations 3.48 and 3.49 contain only two distinct inner products, $\left\langle Q_{\gamma(n)}^{(n)} | e^{(n)} \right\rangle$ and $\left\langle Q_{\gamma(n)}^{(n)} | e^{(n-1)} \right\rangle$. The combinatorial terms are also considerably simplified. The total number of operations required to compute both $\phi_\gamma(n)$ and $\vec{a}_{\gamma(n)}$ in the purely Newtonian case, assuming that products of components of $\hat{e}$ are pre-computed, is proportional to

$$\binom{n+2}{2}, \tag{4.35}$$

and the time required to evaluate terms up through order p is proportional to

$$\sum_{n=0}^{p} \binom{n+2}{2} = \binom{p+3}{3}. \tag{4.36}$$

In this case, the time required to pre-compute products of components of $\hat{e}$ is not negligible, contributing approximately the same number of operations as the computation of the inner products, $\left\langle Q_{\gamma(n)}^{(n)} | e^{(n)} \right\rangle$. Nevertheless, the total operation count is proportional to p^3.

4.5. Choice of the terminal size parameter, m.

We have just seen that the computation of the interaction between a body and a set of multipoles can be rather costly. It is safe to say that a multipole interaction

will generally be more time consuming than a simple body-body interaction, i.e., a single evaluation of the Green's function. This leads us to construct multipole expansions only if there are a sufficient number of particles to achieve a net decrease in computation time.

If we have a set of N_γ bodies in the volume, $\mathcal{V}_\gamma$, as well as a set of multipoles, $M_{\gamma(n)}$, we can compute the field at a point, $\vec{r}$, in two ways:

$$\phi(r) = \sum_{i=1}^{N_\gamma} G(\vec{r} - \vec{x}_i) \tag{4.37}$$

or

$$\phi(r) \approx \sum_{n=0}^{p} \phi_{\gamma(n)}(r). \tag{4.38}$$

Let the time required to evaluate Eqn. 4.37 be

$$N_\gamma T_{b-b}, \tag{4.39}$$

and the time required to evaluate Eqn. 4.38 be

$$T_{b-m}. \tag{4.40}$$

For any particular implementation, we can determine the ratio of the cost of a body-multipole interaction to the cost of a body-body interaction,

$$q \equiv \frac{T_{b-m}}{T_{b-b}}. \tag{4.41}$$

Clearly, it is never advantageous to use Eqn. 4.38 unless

$$N_\gamma > q. \tag{4.42}$$

Thus, it is a waste of time and space to compute and store a multipole expansion for any collection of bodies with fewer than q elements. Furthermore, it is desirable to have a multipole expansion for any collection of bodies with greater than q

```
ComputeAllFields()
    for( each body at which forces are required )
        body.φ = 0
        body.a⃗ = 0
        ComputeField(body, root of tree)
    endfor
endfunc
```

Code 4.4. Function `ComputeAllFields` to compute an approximation to ϕ and $\vec{a}$ for each body in simulation, by recursive descent of a BH tree.

elements. We recall from the definition of the BH tree that terminal nodes of the tree are guaranteed to have m or fewer elements. If we set

$$m = \lfloor q \rfloor , \tag{4.43}$$

then terminal nodes of the tree need never have multipole expansions, and internal nodes of the tree, which are guaranteed to contain more than m bodies, should always contain multipole expansions. Close examination of Code 4.1 reveals that multipole moments are never computed for terminal nodes, and they are always computed for internal nodes. A smaller value of m would lead to the computation of useless multipole moments, while a larger value of m would lead us to use Eqn. 4.37 on occasions when Eqn. 4.38 would be preferable.

4.6. Running time.

Once the tree is complete, we use the algorithm in Code 4.4 to compute all potentials and accelerations. Code 4.4 calls `ComputeField` which traverses the BH tree once for each Body. If a terminal node is encountered during the traversal, then a `BBInteraction` is computed for each of the bodies contained within the terminal node. If the `OpeningCriterion` is satisfied for an internal node, then the traversal continues with its descendants. Finally, if the `OpeningCriterion` fails for an internal node, then the multipole approximation is acceptable, `BCInteraction` is executed to compute the interaction between the body and the multipole expansion of the node. The recursive procedure is also shown in Code 4.4.

```
ComputeField(body, cell)
    if( cell is terminal )
        for( each fieldbody in cell )
            BBInteraction(body, fieldbody)
        endfor
    else if( OpeningCriterion(cell, body) )
        for( each child of cell )
            ComputeField(body, child)
        endfor
    else
        BCInteraction(body, cell)
    endif
endfunc
```

Code 4.5. Function `ComputeField` computes the interaction between a specified body, and all bodies lieing in a cell. `BBInteraction` computes the interaction between a specific pair of bodies, while `BCInteraction` computes an interaction between a body and the multipole expansion stored in a cell.

We now consider how many times `BBInteraction` and how many times `BMinteraction` are called during one execution of `ComputeAllFields`. The "fact" that the BH algorithm is $O(N \log N)$ has been reported repeatedly in the literature.[16, 38, 19] The literature only contains, however, some very rough justifications for this claim. In this section, we use some of the machinery developed in Chapter 2 to attack the running time of the BH algorithm with somewhat more rigor than has been done before. We shall see that the $O(N \log N)$ claim is, in fact, justified for any collection of bodies drawn independently from an underlying distribution, $\bar{\rho}$. It is not, however, guaranteed to be true in all circumstances, as the following counter-example demonstrates.

4.6.1. A counter-example to the $O(N \log N)$ behavior.

Consider a one-dimensional arrangement of N bodies on the unit line, at positions

$$x_n = 2^{-n}; \qquad 1 \leq n \leq N. \tag{4.44}$$

A one-dimensional BH tree constructed for these points is shown in Figure 4.1. The bodies are shown on a line-segment in Figure 4.1 and the BH tree is shown above it in the form of a binary tree. of shorter line-segments. The tree is highly unbalanced. The right child of any node is terminal, while the left child of all nodes except one is internal. This unbalanced structure follows from the positions of the bodies, which clump exponentially closely near the origin.

Now consider the interactions that are computed in the course of traversing the tree to find the potential at body j. Body j is actually inside j internal cells, so no matter what the `OpeningCriterion`, at least j internal cells will be opened, and $j-1$ `BBInteractions` will be executed (with all bodies, $n < j$). Even if we disregard what happens for $n > j$, we see that there are at least

$$\begin{aligned} N_{bb} &= \sum_{j=1}^{N}(j-1) \\ &= N(N-1)/2 \end{aligned} \tag{4.45}$$

body-body interactions computed. Thus, we have a counterexample to the claim that the BH algorithm is strictly $O(N \log N)$.

Nevertheless, this example is clearly contrived. The density of bodies in space grows exponentially near the origin, and as N increases, new bodies are not distributed in the same way as the old ones. Thus, there is no underlying distribution function, $\bar{\rho}$, which could plausibly give rise to bodies distributed in this way for arbitrarily large N. It is this pathology which causes the BH algorithm to fail.

4.6.2. Interaction volume.

Although the BH algorithm proceeds by treating each body separately, after the tree is built, analysis of the timing is easier if we focus on the cells. Rather than asking how many cells does each body interact with, we ask how many bodies does each cell interact with. Since we also want to know how many `BBInteractions` are computed, we also ask if a cell is terminal, then how many `BBInteractionsare` computed with the bodies contained in the cell.

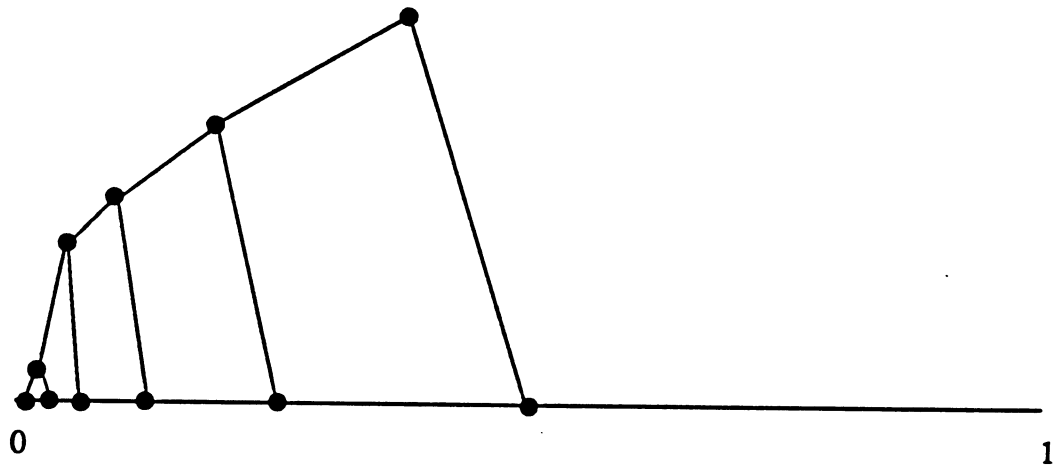


Figure 4.1. One dimensional pathological BH tree.

We proceed as we did in Chapter 2 with a set of N "bodies," located at N independent, identically distributed random positions, restricted to a finite, cubical volume, $\mathcal{V}_0$. We also imagine a complete hierarchy of BH cells beneath $\mathcal{V}_0$, extending to arbitrarily small volumes, and we select an arbitrary cell, $\mathcal{V}_\gamma$, somewhere in the hierarchy. We do not, a priori, assume that $\mathcal{V}_\gamma$ contains any of the bodies, or even that it is represented in the BH tree constructed from the set of bodies.

Around $\mathcal{V}_\gamma$, we identify a set of other regions which will be useful in the following discussions. Figure 4.2 shows, schematically, the region around $\mathcal{V}_\gamma$. $\mathcal{V}_{\uparrow\gamma}$ is the immediate parent of $\mathcal{V}_\gamma$ in the BH tree. $\mathcal{V}_{near}$ is the region around $\mathcal{V}_\gamma$, for which `OpeningCriterion` succeeds, but not including $\mathcal{V}_\gamma$ itself, i.e., it is the region in which the multipole approximation cannot be used for $\mathcal{V}_\gamma$. $\mathcal{V}_{far}$ is the region around the parent of $\mathcal{V}_\gamma$ for which the `OpeningCriterion` fails (for the parent), i.e., it is the region in which the multipole approximation is always used for the parent. $\mathcal{V}_{mid}$ is the region between $\mathcal{V}_{near}$ and $\mathcal{V}_{far}$, i.e., it is the only region in which the multipole approximation for $\mathcal{V}_\gamma$ will be used.

Now, let us assume that there are n_γ bodies in $\mathcal{V}_\gamma$, n_{mid} bodies in $\mathcal{V}_{mid}$, etc., and that the integrals of the probability density, $\bar{\rho}$, are

$$\begin{aligned} p_\gamma &= \int_{\mathcal{V}_\gamma} \bar{\rho}(x) d^3x, \\ p_{mid} &= \int_{\mathcal{V}_{mid}} \bar{\rho}(x) d^3x, \\ &\cdots \end{aligned} \tag{4.46}$$

Then the probability of such a configuration of bodies is given by a multinomial distribution, which we write formally as

$$P_N(n_1, p_1; n_2, p_2; \ldots) = \binom{N}{n_1 n_2 \cdots} p_1^{n_1} p_2^{n_2} \ldots (1 - p_1 - p_2 - \ldots)^{N - n_1 - n_2 - \ldots} \tag{4.47}$$

If there is no possibility of confusion, we will use the following shorter notation

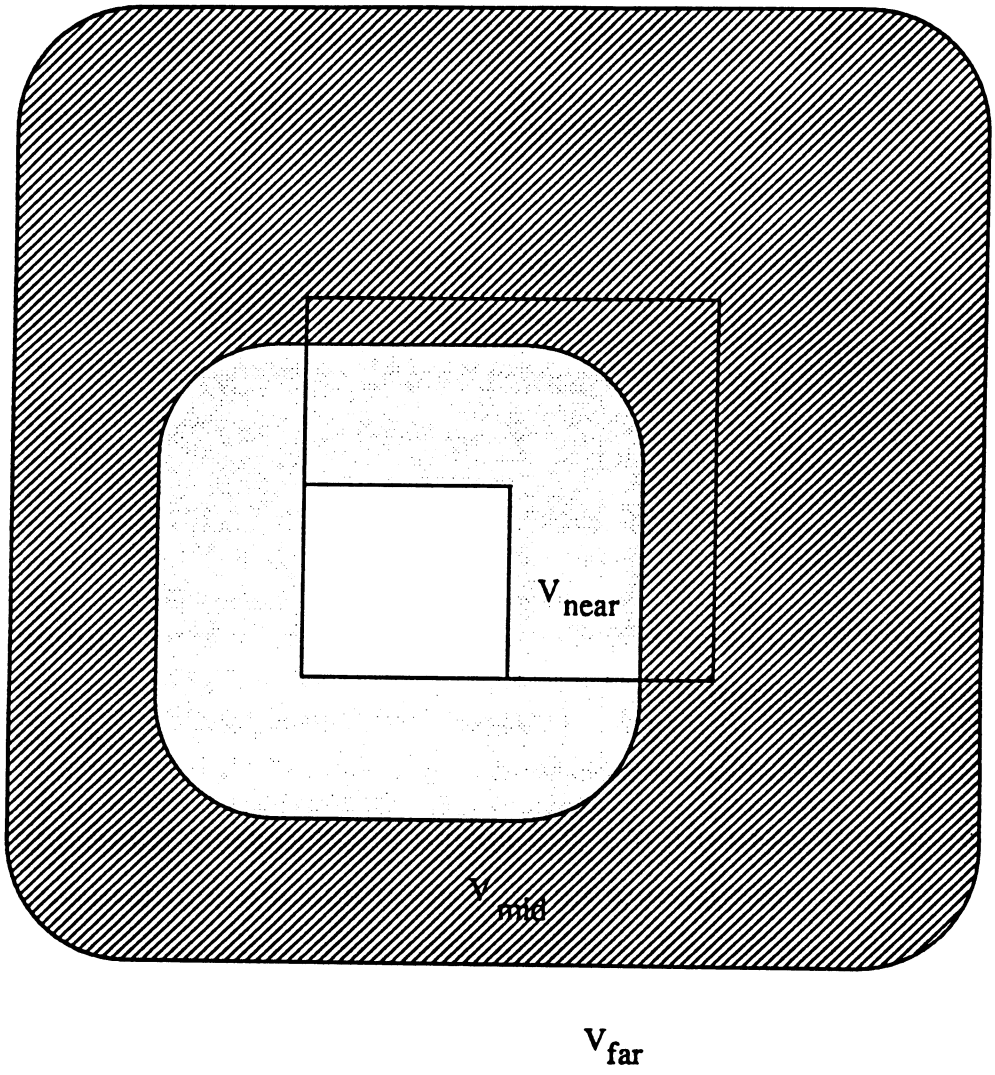


Figure 4.2. Labeling of regions around $\mathcal{V}_\gamma$ for computation of `BCInteractions`.

for the multinomial distribution function,

$$P_N(n_1, n_2, \ldots) = P_N(n_1, p_1; n_2, p_2; \ldots). \tag{4.48}$$

The multinomial distribution function obeys some useful identities:

$$\begin{aligned}
&n_1 P_N(n_1, \ldots) = N p_1 P_{N-1}(n_1 - 1, \ldots), \\
&\sum_{n_1=0}^{N} P_N(n_1, \ldots) = P_N(\ldots), \\
&\sum_{n_1=0}^{m} P_N(n_1, p_1; m - n_1, p_2; \ldots) = P_N(m, (p_1 + p_2); \ldots), \\
&\sum_{n=0}^{m} P_N(n, p) = C_{N,m}(p), \\
&\sum_{n=m+1}^{N} P_N(n, p) = D_{N,m}(p) = 1 - C_{N,m}(p),
\end{aligned} \tag{4.49}$$

where $C_{m,N}$ and $D_{m,N}$ are defined in Chapter 2. They are, in turn, related to the incomplete beta function.[34]

4.6.3. Number of `BCInteractions`.

Now consider a particular cell, $\mathcal{V}_\gamma$. The number of `BCInteractions` in which $\mathcal{V}_\gamma$ is the cell is simply the number of bodies in the $\mathcal{V}_{mid}$, i.e., n_{mid}. Bodies in $\mathcal{V}_{far}$ cannot interact with $\mathcal{V}_\gamma$ because their traversal would be terminated before reaching $\mathcal{V}_\gamma$. Similarly, bodies in $\mathcal{V}_{near}$ or in $\mathcal{V}_\gamma$ itself cannot interact with $\mathcal{V}_\gamma$ because the `OpeningCriterion` succeeds, and traversal continues with the descendants of $\mathcal{V}_\gamma$. Thus, the number of `BCInteractions` with $\mathcal{V}_\gamma$ is

$$N_{bc\gamma} = \begin{cases} n_{mid}, & n_\gamma > m; \\ 0, & \text{otherwise.} \end{cases} \tag{4.50}$$

There are no body-cell interactions unless $\mathcal{V}_\gamma$ is a bona fide internal cell, containing more than m bodies. Now, we can compute the expected number of body-cell

interactions, simply by summing over the probability distribution and using the identities in Eqn. 4.49,

$$\begin{aligned}\langle N_{bc\gamma}\rangle &= \sum_{n_\gamma > m} \sum_{n_{mid}} n_{mid} P_N(n_\gamma, n_{mid}) \\ &= \sum_{n_\gamma > m} \sum_{n_{mid}} N p_{mid} P_{N-1}(n_\gamma, n_{mid}) \\ &= N p_{mid} D_{N-1,m}(p_\gamma).\end{aligned} \tag{4.51}$$

We can now sum Eqn. 4.51 over all cells, $\mathcal{V}_\gamma$, and obtain the expected number of interactions with all cells in the tree,

$$\begin{aligned}\langle N_{bc}\rangle &= \sum_\gamma \langle N_{bc\gamma}\rangle \\ &= N \sum_\gamma p_{mid} D_{N-1,m}(p_\gamma).\end{aligned} \tag{4.52}$$

4.6.4. Number of BBInteractions.

Counting the average number of BBInteractions is slightly more complicated. First, we divide the space around $\mathcal{V}_\gamma$ into a slightly different set of disjoint regions, as shown in Figure 4.3. $\mathcal{V}_{sib}$ is the set of "siblings" of $\mathcal{V}_\gamma$, i.e., those cells which share a common parent, ane $\mathcal{V}_\phi$ is the region which is "near" to the parent of $\mathcal{V}_\gamma$. The number of bodies in each region, e.g., n_{sib} and the integrated probabilities, e.g., p_{sib}, are defined just as in the previous section.

Each cell is assigned a certain number of BBInteractions. If the cell is not terminal, then it is assigned no BBInteractions. Otherwise, the cell is assigned n_γ BBInteractions for each body which encounters the cell during its tree traversal. Since bodies do not interact with themselves, we subtract one interaction if the body which encountered the cell is actually contained within the cell. The regions shown in Figure 4.3 have been constructed so that bodies which fall within $\mathcal{V}_\gamma$, $\mathcal{V}_{sib}$ and $\mathcal{V}_\phi$ are guaranteed to encounter $\mathcal{V}_\gamma$ during their tree traversal, and all other

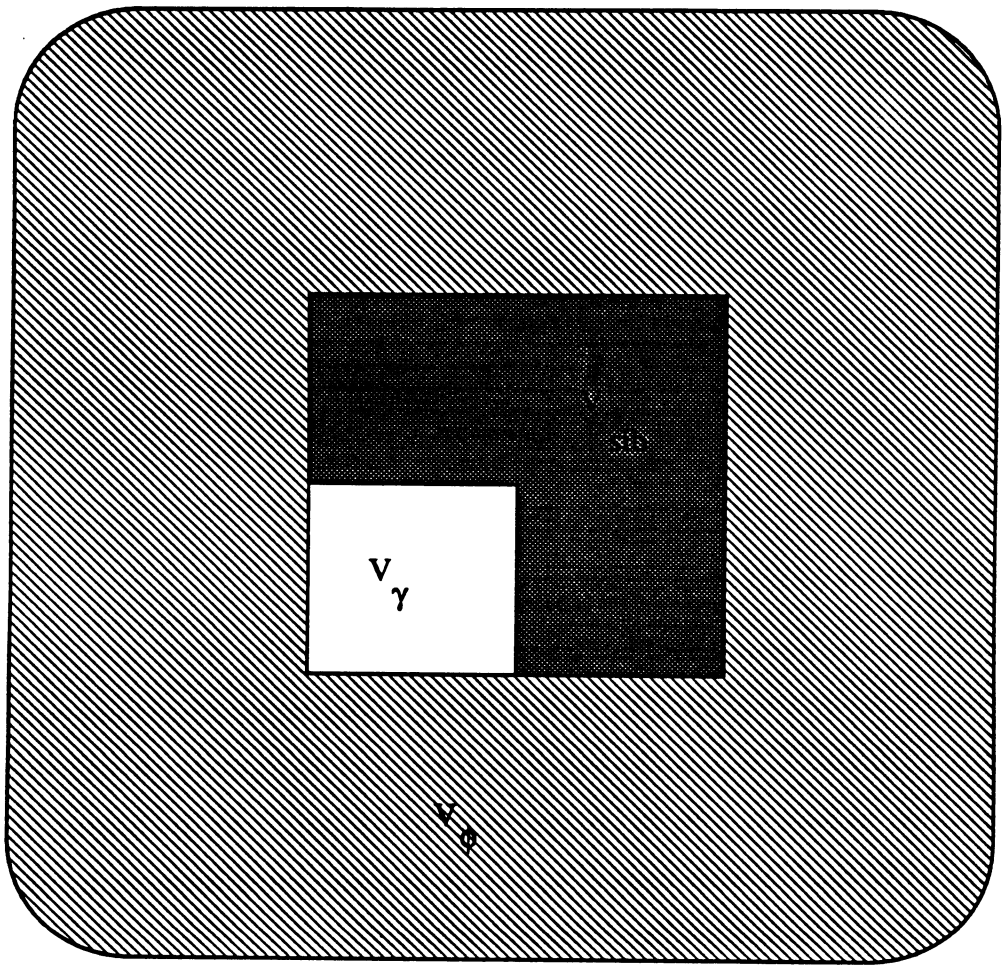


Figure 4.3. Labeling of regions around $\mathcal{V}_\gamma$ for computation of `BBInteractions`.

bodies are guaranteed not to. Thus, the number of `BBInteractions` associated with cell $\mathcal{V}_\gamma$ is

$$N_{bb\gamma} = \begin{cases} n_\gamma(n_\gamma - 1 + n_{sib} + n_\phi), & n_\gamma + n_{sib} > m; \\ 0, & \text{otherwise.} \end{cases} \tag{4.53}$$

To compute the expected number of interactions, we simply sum over all possible values of n_γ, n_{sib} and n_ϕ,

$$\langle N_{bb\gamma} \rangle = \sum_{n_\gamma=0}^{m} \sum_{n_{sib}>m-n_\gamma}^{N} \sum_{n_\phi=0}^{N} n_\gamma(n_\gamma - 1 + n_{sib} + n_\phi) P_N(n_\gamma, n_{sib}, n_\phi). \tag{4.54}$$

Using the identities in Eqn. 4.49, and the fact that

$$p_{\uparrow\gamma} = p_\gamma + p_{sib}, \tag{4.55}$$

the sums in Eqn. 4.54 can be eliminated and we obtain

$$\begin{aligned} \langle N_{bb\gamma} \rangle =& N(N-1)p_\gamma \\ & \times \Big(p_\gamma \left(C_{N-2,m-2}(p_\gamma) - C_{N-2,m-2}(p_{\uparrow\gamma}) \right) \\ & + p_\phi \left(C_{N-2,m-1}(p_\gamma) - C_{N-2,m-1}(p_{\uparrow\gamma}) \right) \\ & + p_{sib} \left(C_{N-2,m-1}(p_\gamma) - C_{N-2,m-2}(p_{\uparrow\gamma}) \right) \Big). \end{aligned} \tag{4.56}$$

Now, we sum over all possible cells, $\mathcal{V}_\gamma$, to obtain the total expected number of `BBInteractions`,

$$\begin{aligned} \langle N_{bb} \rangle =& N(N-1) \sum_\gamma p_\gamma \\ & \times \Big(p_\gamma \left(C_{N-2,m-2}(p_\gamma) - C_{N-2,m-2}(p_{\uparrow\gamma}) \right) \\ & + p_\phi \left(C_{N-2,m-1}(p_\gamma) - C_{N-2,m-1}(p_{\uparrow\gamma}) \right) \\ & + p_{sib} \left(C_{N-2,m-1}(p_\gamma) - C_{N-2,m-2}(p_{\uparrow\gamma}) \right) \Big). \end{aligned} \tag{4.57}$$

We may use the following identities, all of which represent absolutely convergent sums, to rewrite the sum in Eqn. 4.57:

$$\begin{aligned}
\sum_{\gamma} {p_\gamma}^2 C_{N-2,m-2}(p_{\uparrow\gamma}) &= \sum_{\gamma} C_{N-2,m-2}(p_\gamma) \sum_{\alpha \in Child(\gamma)} {p_\alpha}^2, \\
\sum_{\gamma} p_\gamma p_\phi C_{N-2,m-1}(p_{\uparrow\gamma}) &= \sum_{\gamma} p_{near} p_\gamma C_{N-2,m-1}(p_\gamma), \\
\sum_{\gamma} p_\gamma p_{sib} C_{N-2,m-2}(p_{\uparrow\gamma}) &= \sum_{\gamma} \sum_{\alpha \in Child(\gamma)} p_\alpha (p_\gamma - p_\alpha) C_{N-2,m-2}(p_\gamma), \\
&= \sum_{\gamma} C_{N-2,m-2}(p_\gamma) \left({p_\gamma}^2 - \sum_{\alpha \in Child(\gamma)} {p_\alpha}^2 \right).
\end{aligned} \tag{4.58}$$

The result is

$$\begin{aligned}
\langle N_{bb} \rangle &= N(N-1) \sum_{\gamma} p_\gamma (p_\phi + p_{sib} - p_{near}) C_{N-2,m-1}(p_\gamma) \\
&= N(N-1) \sum_{\gamma} p_\gamma p_{mid} C_{N-2,m-1}(p_\gamma).
\end{aligned} \tag{4.59}$$

4.6.5. Uniform distribution, i.e., $\bar{\rho}(x) = const.$

We now have two formal expressions, Eqns. 4.59 and 4.52, for the expected number of of `BBInteractions` and `BCInteractions` that are executed when all bodies traverse the BH tree according to Code 4.4. Despite the fact that we required p_{sib}, p_ϕ, etc. to derive Eqns. 4.59 and 4.52, the only probabilities that appear in the final results are p_γ and p_{mid}. The expected number of interactions depends implicitly on the underlying distribution of bodies, $\bar{\rho}(x)$, through the quantities p_γ and p_{mid}. Before we consider the general case of an arbitrary $\bar{\rho}(x)$, it is useful to treat the special case of a uniform distribution, i.e.,

$$\begin{aligned}
\bar{\rho}(x) &= \frac{1}{V_0}, \\
p_\gamma &= \frac{V_\gamma}{V_0}, \\
p_{mid} &\leq \frac{V_{mid}}{V_0} = \frac{V_{mid}}{V_\gamma} p_\gamma.
\end{aligned} \tag{4.60}$$

The inequality appears in Eqn. 4.60 because for some cells, $\mathcal{V}_\gamma$, the associated "middle" region, $\mathcal{V}_{mid}$ extends outside of $\mathcal{V}_0$. The situation is illustrated in Figure 4.4.

In Chapter 2, we defined the depth of a cell, $d(\gamma)$, and the set of cells which comprise a "level" of the BH tree, $\mathcal{L}_d$. Since all cells on a level have the same volume, they also have the same p_γ. The sums over all cells in Eqns. 4.59 and 4.52 can be restated as sums over levels, d. Furthermore, the ratio, V_{mid}/V_γ, is entirely independent of γ. It depends only on the `OpeningCriterion` function used during the tree traversal. We shall discuss `OpeningCriteria` in detail in Chapter 6. For now, we shall characterize the `OpeningCriterion` with the constant

$$F_{OC} = \frac{V_{mid}}{V_\gamma}. \tag{4.61}$$

Finally, we note that on each level, there are exactly p_d^{-1} separate cells.

4.6.5.1. Number of `BCInteractions`.

Thus, Eqn. 4.52 may be rewritten as

$$\langle N_{bc} \rangle \leq N F_{OC} \sum_{d=0}^{\infty} D_{N-1,m}(p_d) \qquad p_d = 8^{-d}. \tag{4.62}$$

Equation 4.62 is closely related to the average depth of bodies in the BH tree, D_{avg}, computed in Chapter 2, where we found that D_{avg} is tightly bounded by the logarithm of N. In fact, using Eqns. 2.42, 2.44 and 4.62, we have

$$\langle N_{bc} \rangle \leq F_{OC} N D_{avg}(m+1) \leq F_{OC} N \log_8 \left(\frac{d_{m+1} N}{m+1} \right). \tag{4.63}$$

The values of d_m are tabulated in Table 2.1.

4.6.5.2. Number of `BBInteractions`.

The situation for N_{bb} is again somewhat more complicated. The number of `BBInteractions` is not related to a previously computed statistic about the tree. Nevertheless, we can estimate the value of $\langle N_{bb} \rangle$. Again using the fact that all cells

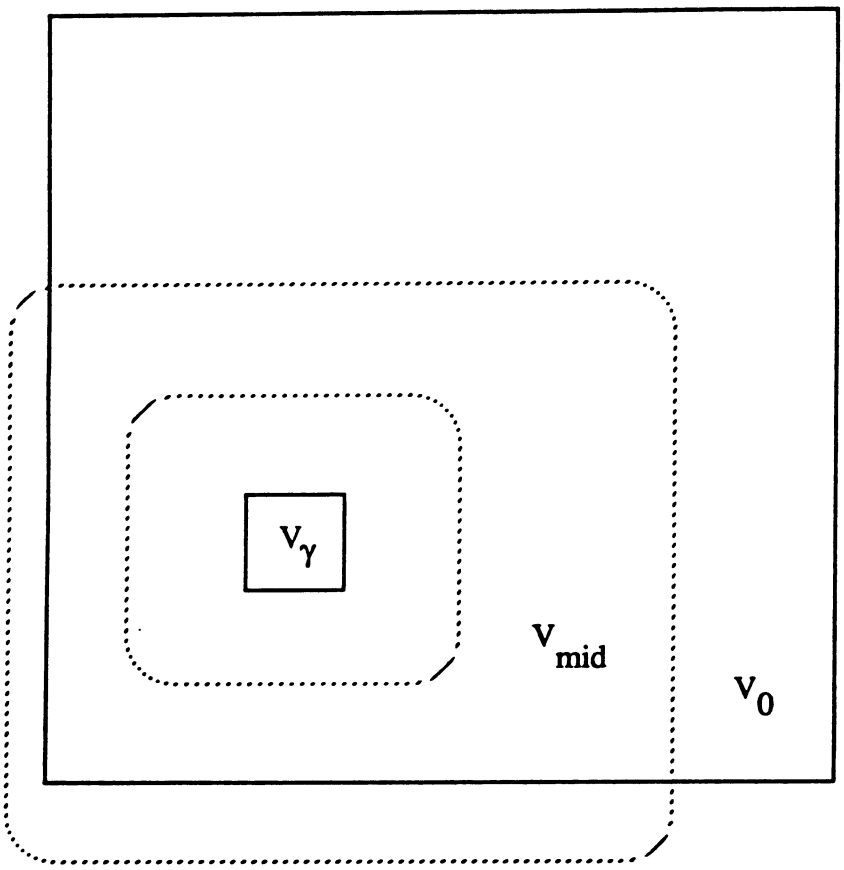


Figure 4.4. Cell $\mathcal{V}_\gamma$ for which $\mathcal{V}_{mid}$ extends outside of $\mathcal{V}_0$.

on a level are equivalent, we have

$$\langle N_{bb} \rangle \leq N(N-1) F_{OC} \sum_{d=0}^{\infty} p_d C_{N-1,m-1}(p_d); \qquad p_d = 8^{-d}. \tag{4.64}$$

The cumulative binomial distribution, $C_{N,m}$, has the following properties which allow us bound the summation in Eqn. 4.64,

$$\begin{aligned} C_{N,m}(p) &\propto \exp(-Np); \qquad Np \gg m, \\ C_{N,m}(p) &\approx 1; \qquad Np \ll m. \end{aligned} \tag{4.65}$$

For small values of d, i.e., large values of Np_d, the terms in Eqn. 4.64 are limited by the exponentially small value of $C_{N-1,m-1}$. For large values of d, the terms are limited by the exponentially small values of p_d itself. The only values of d for which the terms of Eqn. 4.64 are non-neglible are those for which

$$(N-1)p_d \approx m. \tag{4.66}$$

The terms for which Eqn. 4.66 holds have magnitudes approximate equal to

$$NmO(1), \tag{4.67}$$

where the $O(1)$ accounts for the small number of terms that satisfy the approximate equality, Eqn. 4.66, and the fact that $C_{N-1,m-1}(p_d)$ is of order unity. Thus, we conclude that, for a uniform distribution of bodies,

$$\langle N_{bb} \rangle = F_{OC} NmO(1). \tag{4.68}$$

We can, of course, simply compute the summation in Eqn. 4.64 numerically for specific values of N and m. The exponential decrease of p_d for large values of d assures us that the sum will converge rapidly. The result of numerical evaluation of Eqn. 4.64 is shown in Figure 4.5. In light of the limiting behavior indicated by Eqn. 4.68, Figure 4.5 shows the ratio of N_{bb} to Nm. Based on the arguments above, and Figure 4.5, we conjecture that

$$\langle N_{bb} \rangle \leq F_{OC} Nmb_m, \tag{4.69}$$

where values of b_m are presented in Table 4.1. A rigorous proof of Eqn. 4.69 can almost certainly be constructed along the lines of the proof in Appendix A concerning C_{avg}. Values of b_m in Eqn. 4.69 are shown in Table 4.1.

Table 4.1. Values of b_m as defined in Eqn. 4.69, estimated from numerical evaluation of Eqn. 4.64 over the range $8 \leq N \leq 2 \times 10^6$.

m	b_m
1	0.52
2	0.55
3	0.59
4	0.62
5	0.64
20	0.79

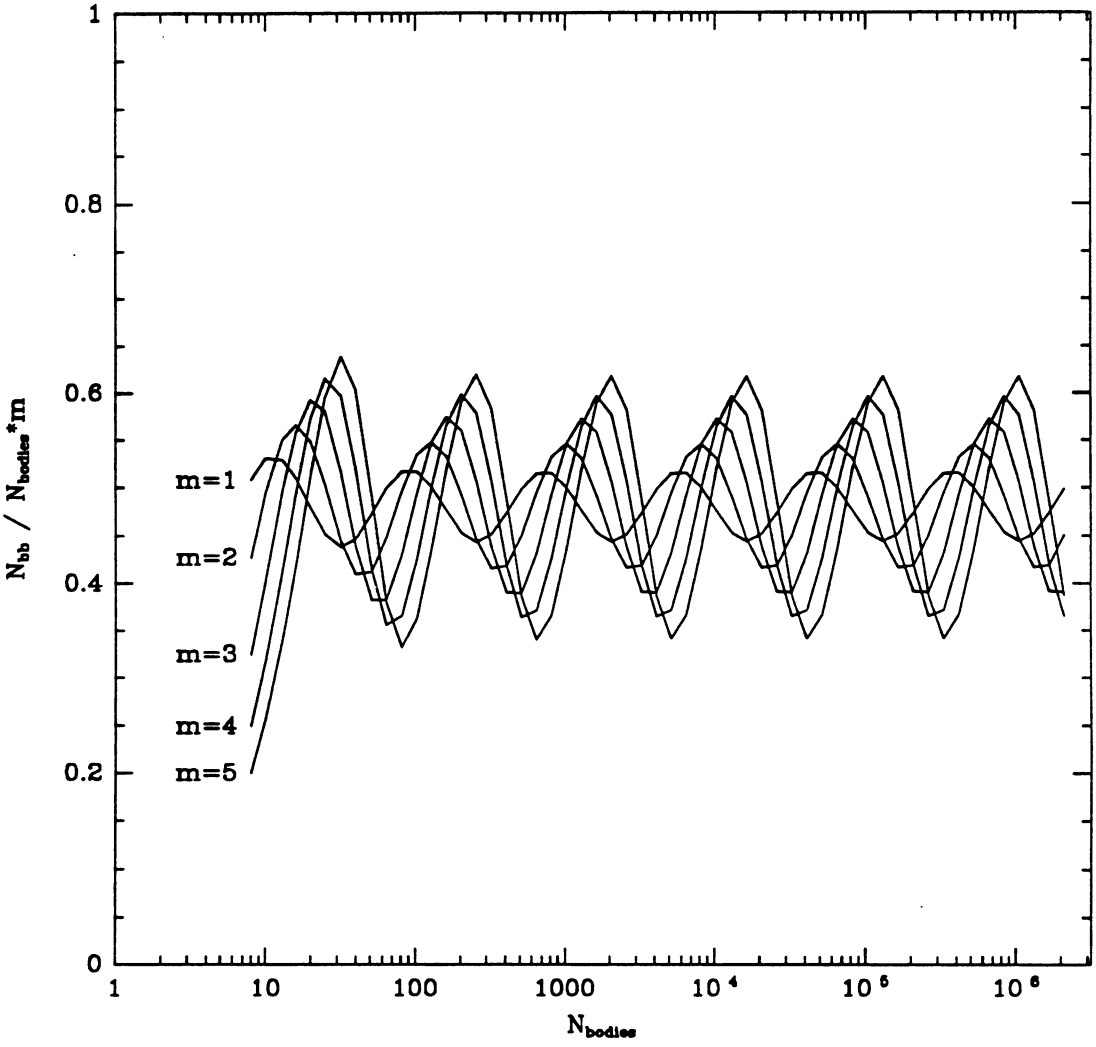


Figure 4.5. $\langle N_{bb} \rangle / Nm$ vs. N.

4.6.6. Non-uniform distribution, i.e., $\bar{\rho}(x) \neq const.$

In practice, one does not simulate systems of constant particle density. The purpose of numerical simulations is to study systems which are too complicated to treat by other means. These are precisely the systems which deviate significantly from uniformity. (Uniform systems can be treated analytically, and almost-uniform systems can often be treated by analytic means.) Thus, it is important to assess the performance of the algorithm for situations in which the distribution of bodies in space is non-uniform.

Fortunately, a non-uniform distribution of bodies has only a weak adverse affect on the performance of the BH algorithm. In Chapter 2, we analyzed some statistical properties of BH trees with non-uniform particle distributions. We defined the "measurability" of a distribution, and were able to estimate the properties of the tree under the constraint that $\bar{\rho}(x)$ be measurable. We shall use exactly the same strategy in this section, and the results will be valid under very similar conditions on N and $\bar{\rho}(x)$

Recall that for a distribution to be "measurable," we must have a finite set of cells, $\mathcal{S}$, with the properties:

$$\begin{aligned} &\sum_{\gamma \in \mathcal{S}} p_\gamma = 1, \\ &\forall \gamma \in \mathcal{S}: \quad \bar{\rho}(x) \text{ is approximately constant over } \mathcal{V}_{mid},\ \mathcal{V}_{near} \text{ and } \mathcal{V}_\gamma \\ &\forall \gamma \in \mathcal{S}: \quad (N p_\gamma)^{\frac{1}{2}} \gg 1. \end{aligned} \tag{4.70}$$

The conditions in Eqn. 4.70 are slightly stronger than those in Eqn. 2.45 because Eqn. 4.70 requires that $\bar{\rho}$ be approximately constant over a larger region, namely $\mathcal{V}_{mid}$, $\mathcal{V}_{near}$ and $\mathcal{V}_\gamma$. Equation 2.45 only required that $\bar{\rho}$ be approximately constant over $\mathcal{V}_\gamma$.

Nevertheless, for any distribution function, $\bar{\rho}$, which is continuous and bounded, there is always a value of N and a set, $\mathcal{S}$, which satisfies Eqn. 4.70. The results of this section are valid when the conditions of Eqn. 4.70 are satisfied.

Now we follow the same approach as was used in Chapter 2 and use the results developed in the previous section for each of the constant-density cells in $\mathcal{S}$. We begin by formally writing the summation over all cells, γ, as two separate sums,

$$\sum_{\gamma} = \sum_{\sigma \in \mathcal{S}} \sum_{\gamma \in Desc(\sigma)} + \sum_{\gamma \in Anc(\mathcal{S})}, \tag{4.71}$$

where $Anc(\mathcal{S})$ is the set of all cells which are ancestors of cells in $\mathcal{S}$ and $Desc(\sigma)$ is the set of cells which are descendants of σ.

4.6.6.1. Number of BBInteractions.

It is somewhat simpler to estimate the number of BBInteractions in the non-uniform case, so we first treat $\langle N_{bb} \rangle$, which is given by Eqn. 4.59. Using Eqn. 4.71, we have

$$\langle N_{bb} \rangle \leq \left(\sum_{\sigma \in \mathcal{S}} \sum_{\gamma \in Desc(\sigma)} + \sum_{\gamma \in Anc(\mathcal{S})} \right) N(N-1) p_\gamma p_{mid} C_{N-2,m-2}(p_\gamma). \tag{4.72}$$

Each cell in $\mathcal{S}$ has very nearly N_γ bodies in it, and very nearly constant $\bar{\rho}$ throughout the cell, and in its immediate surroundings. Thus, the sum over the descendants of σ in Eqn. 4.72 may be replaced by Eqn. 4.69,

$$\langle N_{bb} \rangle \leq \left(\sum_{\gamma \in \mathcal{S}} b_m F_{OC} N_\gamma \right) + N(N-1) \sum_{\gamma \in Anc(\mathcal{S})} p_\gamma p_{mid} C_{N-2,m-1}(p_\gamma). \tag{4.73}$$

Now we can make use of the properties of the cumulative binomial distributions. For N satisfying Eqn. 4.70, and $p_\gamma \in \mathcal{S}$, $C_{N-2,m-1}(p_\gamma)$ is exponentially small. Thus, the second summation in Eqn. 4.73 is completely negligible and we have

$$\begin{aligned} \langle N_{bb} \rangle &\leq b_m F_{OC} \sum_{\gamma \in \mathcal{S}} N_\gamma \\ &= b_m F_{OC} N. \end{aligned} \tag{4.74}$$

Equation 4.74 is identical to Eqn. 4.69. All that has changed is the domain of applicability. Eqn. 4.74 applies as long as N is large enough so that $\bar{\rho}$ is measurable, according to the definition of Eqn. 4.70.

4.6.6.2. Number of BCInteractions.

Computing the number of BCInteractions is slightly more complicated. We begin with Eqn. 4.52, and again use Eqn. 4.71. Just as in the previous section, every cell in $\mathcal{S}$ has very nearly constant $\bar{\rho}$. Thus, the sum over the descendants of σ may be replaced by Eqn. 4.63. The result is:

$$\langle N_{bc} \rangle \leq \left(F_{OC} \sum_{\gamma \in \mathcal{S}} N_\gamma \log_8 \left(\frac{d_{m+1} N_\gamma}{m+1} \right) \right) + N \sum_{\gamma \in Anc(\mathcal{S})} p_{mid} D_{N-1,m}(p_\gamma). \quad (4.75)$$

The cumultative binomial distributions that appear in Eqn. 4.75, $D_{N-1,m}(p_\gamma)$, are exponentially close to unity, so we can eliminate them for simplicity,

$$\langle N_{bc} \rangle \leq \left(F_{OC} \sum_{\gamma \in \mathcal{S}} N_\gamma \log_8 \left(\frac{d_{m+1} N_\gamma}{m+1} \right) \right) + N \sum_{\gamma \in Anc(\mathcal{S})} p_{mid}. \quad (4.76)$$

The final summation in Eqn. 4.76 is:

$$\sum_{\gamma \in Anc(\mathcal{S})} p_{mid} \leq \sum_{\gamma \in Anc(\mathcal{S})} F_{OC} p_\gamma \quad (4.77)$$

Now we make use of the fact that the elements of $\mathcal{S}$ cover all of $\mathcal{V}_0$. Thus,

$$\sum_{\gamma \in Anc(\mathcal{S})} p_\gamma = \sum_{\gamma \in Anc(\mathcal{S})} \sum_{\beta \in \mathcal{S}} p_{\gamma \cap \beta} = \sum_{\beta \in \mathcal{S}} \sum_{\gamma \in Anc(\mathcal{S})} p_{\gamma \cap \beta} = \sum_{\beta \in \mathcal{S}} p_\beta depth(p_\beta). \quad (4.78)$$

We can use Eqns. 4.78 and 4.77 in Eqn. 4.76, and obtain:

$$\langle N_{bc} \rangle \leq F_{OC} \sum_{\gamma \in \mathcal{S}} N_\gamma \left(\log_8 \left(\frac{d_{m+1} N_\gamma}{m+1} \right) + depth(p_\gamma) \right). \quad (4.79)$$

We now follow the same reasoning as in Chapter 2, with

$$depth(p_\gamma) = \log_8 \left(\frac{V_0}{V_\gamma} \right). \quad (4.80)$$

After some simple algebraic manipulations,

$$\langle N_{bc} \rangle \leq F_{OC} N \left(\log_8 \left(\frac{d_{m+1} N}{m+1} \right) + H \right), \tag{4.81}$$

where H was defined in Chapter 2,

$$H = \sum_{\gamma \in \mathcal{S}} p_\gamma \log_8 \left(\frac{p_\gamma V_0}{V_\gamma} \right). \tag{4.82}$$

In Chapter 2 we noted that H is closely related to the "entropy" of the underlying distribution, $\bar{\rho}$. Since $\bar{\rho}$ is approximately constant over all the elements of $\mathcal{S}$, the sum can be rewritten as an integral, and we have

$$H \approx \int_{\mathcal{V}_0} \bar{\rho}(x) \log_8(\bar{\rho}(x) V_0) dV. \tag{4.83}$$

H is a characteristic of the underlying distribution.

In Section 2.9.3 we showed that the

$$H \geq 0, \tag{4.84}$$

where equality only holds for a constant distribution. Thus, the performance of the BH algorithm should degrade somewhat as $\bar{\rho}$ departs from uniformity. Nevertheless, H does not depend on N, and so it does not influence the assymptotic "order" of the expected number of `BCInteractions`,

$$\langle N_{bc} \rangle \leq O(N \log_8 N). \tag{4.85}$$

We showed in Section 2.9.3 that for practical values of N, in the range 10^4–10^6, H is considerably smaller than $\log_8(d_{m+1} N/m + 1)$, so the correction due to the non-uniformity of the distribution has only a small effect on the total number of `BCInteractions`.

4.7. Number of dimensions not equal to three.

The results of this chapter apply only to systems in three spatial dimensions.

If the number of dimensions, d, is not equal to three ($d = 2$ being of greatest physical interest), then the number of components in $M_{\gamma(n)}$ is given by

$$\binom{n+d-1}{d-1}. \tag{4.86}$$

The following results are obtained by exactly the same logic as their counterparts in the previous section.

The total storage required for all moments up through order p is

$$\binom{n+d}{d}. \tag{4.87}$$

The number of operations required to translate the components of $M_{\gamma(n)}$ is

$$\binom{n+2d-1}{2d-1}. \tag{4.88}$$

The number of operations required to translate all moments up through order p is

$$\binom{p+2d}{2d}. \tag{4.89}$$

The number of operations required to compute potentials for an arbitrary Green's function requires approximately

$$\frac{d^{p+1}}{d-1} \tag{4.90}$$

multiplications and a similar number of evaluations of derivatives of the Green's function.

If the Green's function is spherically symmetric, then the total work in evaluating ϕ_γ and $\vec{a}_{\gamma(n)}$ through order p is

$$\binom{p+d+2}{d+1}. \tag{4.91}$$

If the Green's function is Newtonian, then the number of operations is further reduced to

$$\binom{p+d}{d}. \tag{4.92}$$

The total number of `BBInteractions` in a system of d dimensions is still proportional to Nm. The constants of proportionality will be different from those in Table 4.1, however.

Finally, the number of `BCInteractions` will satisfy

$$\langle N_{bc} \rangle \leq F_{OC} N \left(\log_{2^d} \left(\frac{N d_{m+1}}{m+1} \right) + \frac{3}{d} H \right), \tag{4.93}$$

but the precise values of d_m will be different from those in Table 2.1.

5. Analysis of Errors in the BH Algorithm.

5.1. Error analysis.

In Chapter 3, we saw how to carry out a multipole expansion for an arbitrary Green's function, and several interesting special cases. Such an expansion is only useful, of course, if it provides an adequate approximation to the desired result. We need to investigate the conditions under which the multipole expansion converges, and further, obtain estimates for the magnitude of the error introduced by terminating the expansion after, say, n terms. Some of the results of this section have been reported by Greengard.[13] Barnes and Hut[17] have also obtained similar results. The formalism we develop here allows us to treat non-Newtonian Green's functions, which cannot easily be incorporated into Greengard's analysis. Furthermore, we derive error bounds for the gradient of the potential, which are new, and which suggest that the gradient of the potential is considerably harder to evaluate than the potential itself.

To carry out this program, we must replace the implied infinite summations in Eqns. 3.7 and 3.8 with finite summations and firm error terms. This is accomplished by returning to the Taylor expansion with error term, [39]

$$G((\vec{r}-\vec{r}_\gamma)-(\vec{x}-\vec{r}_\gamma)) = \sum_{m=0}^{n} \frac{(-1)^m}{m!}(x-r_\gamma)^{i_1}\cdots(x-r_\gamma)^{i_m}\partial_{i_1}\cdots\partial_{i_m}G|_{\vec{r}-\vec{r}_\gamma} \quad (5.1)$$

$$+\frac{1}{n!}(x-r_\gamma)^{i_1}\cdots(x-r_\gamma)^{i_{n+1}}\int_0^1 dt(1-t)^n\partial_{i_1}\cdots\partial_{i_{n+1}}G|_{\vec{r}-\vec{r}_\gamma-t(\vec{x}-\vec{r}_\gamma)}.$$

From Eqn. 5.1, we obtain

$$\phi_\gamma(r) = \sum_{m=0}^{n}\phi_{\gamma(n)}(r) + \Phi_{\gamma(n)}(r), \quad (5.2)$$

where

$$\Phi_{\gamma(n)}(r) = \int_{\mathcal{V}_\gamma} \rho(x) K_n(x) d^3x, \tag{5.3}$$

and

$$K_n(x) = \frac{(-1)^{n+1}}{n!}(x-r_\gamma)^{i_1}\cdots(x-r_\gamma)^{i_{n+1}} \int_0^1 dt(1-t)^n \partial_{i_1}\cdots\partial_{i_{n+1}} G|_{\vec{r}-\vec{r}_\gamma - t(\vec{x}-\vec{r}_\gamma)}. \tag{5.4}$$

We may also express the remainder in the "Lagrange form" as

$$\begin{aligned} K_n(x) = \frac{(-1)^{n+1}}{(n+1)!}&(x-r_\gamma)^{i_1}\cdots(x-r_\gamma)^{i_{n+1}} \\ &\times \partial_{i_1}\cdots\partial_{i_{n+1}} G|_{\vec{r}-\vec{r}_\gamma - t(\vec{x}-\vec{r}_\gamma)} \qquad 0 \le t \le 1, \end{aligned} \tag{5.5}$$

or in the "Cauchy form" as

$$\begin{aligned} K_n(x) = \frac{(-1)^{n+1}}{n!}&(1-t)^n(x-r_\gamma)^{i_1}\cdots(x-r_\gamma)^{i_{n+1}} \\ &\times \partial_{i_1}\cdots\partial_{i_{n+1}} G|_{\vec{r}-\vec{r}_\gamma - t(\vec{x}-\vec{r}_\gamma)} \qquad 0 \le t \le 1. \end{aligned} \tag{5.6}$$

In both of these forms, t represents an unknown value in the range

$$0 \le t \le 1. \tag{5.7}$$

Since we are interested in the magnitude of $\Phi_{\gamma(n)}$, we take the absolute values of both sides of Eqn. 5.3,

$$\begin{aligned} \left|\Phi_{\gamma(n)}(r)\right| &\le \int_{\mathcal{V}_\gamma} |\rho(x)|\,|K_n(x)|\,d^3x \\ &\le K_n^{sup} \int_{\mathcal{V}_\gamma} |\rho(x)|\,d^3x, \end{aligned} \tag{5.8}$$

where

$$K_n^{sup} = \sup_{x\in\mathcal{V}_\gamma} |K_n(x)|. \tag{5.9}$$

If we further assume that $\rho(x)$ is non-negative, then

$$\left|\Phi_{\gamma(n)}(r)\right| \leq K_n^{sup} M_{\gamma(0)}. \tag{5.10}$$

The gradient of K_n is also of interest. It contains information about the residual error after n terms have been used to approximate $\vec{a}(r)$. Taking the gradient of Eqn. 5.2, we have

$$\vec{a}(r) = \sum_{m=0}^{n} \vec{a}_{\gamma(n)}(r) + \vec{A}_{\gamma(n)}(r), \tag{5.11}$$

$$\vec{A}_{\gamma(n)}(r) = -\int_{\mathcal{V}_\gamma} \rho(x)\vec{\nabla}K_n(x)d^3x. \tag{5.12}$$

Following the same reasoning as above, we conclude that

$$\left|\vec{A}_{\gamma(n)}(r)\right| \leq \vec{\nabla}K_n^{sup} M_{\gamma(0)}. \tag{5.13}$$

Clearly, if we can place firm bounds on the magnitude of $K_n(x)$ and its gradient, then we will be able to place correspondingly firm bounds on the error introduced by truncating the multipole expansion after only n terms. In the following sections, we investigate bounds on the magnitude of $K_n(x)$ under various assumptions about the form of the Green's function, $G(x)$.

5.1.1. Arbitrary Green's function.

We begin by assuming that the region, $\mathcal{V}_\gamma$, is bounded and that its "radius," measured from r_γ, is b,

$$b \equiv \sup_{\vec{x}\in\mathcal{V}_\gamma} |\vec{x} - \vec{r}_\gamma|. \tag{5.14}$$

As we shall see, the error bounds that we develop for $\Phi_{\gamma(n)}$ are expressed as a power of b over a length-scale introduced by the geometry and the Green's function.

With a completely arbitrary Green's function, there is little we can do except apply dimensional arguments. In order for dimensions to agree, we must be able to write the n^{th} derivative of the Green's function as

$$\left|\partial_{i_1}\cdots\partial_{i_{n+1}} G|_{\vec{r}-\vec{r}_\gamma - t(\vec{x}-\vec{r}_\gamma)}\right| = \frac{G|_{\vec{r}-\vec{r}_\gamma}}{|D_{n+1}|^{n+1}}, \tag{5.15}$$

where D has the dimensions of length, and depends on the precise value of $\vec{r} - \vec{r}_\gamma - t(\vec{x} - \vec{r}_\gamma)$. Let us suppose, now, that with $\vec{r}$ fixed, and with t and x allowed to range through their respective domains, that the magnitude of the the derivative of G is bounded. That is, that there exists a non-zero function, $D_{n+1}(\vec{r})$, such that

$$\left|\partial_{i_1} \cdots \partial_{i_{n+1}} G|_{\vec{r}-\vec{r}_\gamma - t(\vec{x}-\vec{r}_\gamma)}\right| \leq \frac{G|_{\vec{r}-\vec{r}_\gamma}}{|D_{n+1}|^{n+1}}. \tag{5.16}$$

We may now take the absolute value of Eqn. 5.5 and apply Eqns. 5.14 and 5.16, noting that there are 3^{n+1} implied terms in the Einstein summation over the indices $i_1, i_2, \ldots$. We obtain

$$|K_n| \leq \frac{|G|_{\vec{r}-\vec{r}_\gamma}|}{(n+1)!}\left(\frac{3b}{|D_{n+1}|}\right)^{n+1}, \tag{5.17}$$

and, by Eqn. 5.8,

$$\begin{aligned} |\Phi_{\gamma(n)}(r)| &\leq \frac{M_{\gamma(0)}\,|G|_{\vec{r}-\vec{r}_\gamma}|}{(n+1)!}\left(\frac{3b}{|D_{n+1}|}\right)^{n+1}, \\ |\Phi_{\gamma(n)}(r)| &\leq |\phi_{\gamma(0)}(r)|\,\frac{1}{(n+1)!}\left(\frac{3b}{|D_{n+1}|}\right)^{n+1}. \end{aligned} \tag{5.18}$$

Thus, we have a very general result. The error in the multipole expansion after n terms is bounded by the leading term in the expansion times the $n+1$ power of the ratio of the radius of $\mathcal{V}_\gamma$ to a length-scale, D_{n+1}. The length-scale depends on the details of the Green's function, and the geometry of $\mathcal{V}_\gamma$ and $\vec{r}$. The convergence of the Taylor series depends on whether $|D_{n+1}|$ is sufficiently small for large n. It is clear from Eqn. 5.18, that a sufficient condition for the eventual convergence of the Taylor series, Eqn. 5.2 is

$$\lim_{n\to\infty} |D_n| > 3b. \tag{5.19}$$

5.1.2. Spherically symmetric Green's function.

Now, we investigate what further progress can be made if the Green's function is spherically symmetric. We shall use the following identities:

$$d_t \equiv |\vec{r} - \vec{r}_\gamma - t(\vec{x} - \vec{r}_\gamma)|, \tag{5.20}$$

$$\hat{e}_t \equiv \frac{\vec{r} - \vec{r}_\gamma - t(\vec{x} - \vec{r}_\gamma)}{d_t}, \tag{5.21}$$

$$c \equiv |\vec{x} - \vec{r}_\gamma|, \tag{5.22}$$

$$\mu_t \equiv \frac{(\vec{x} - \vec{r}_\gamma) \cdot \hat{e}_t}{c}. \tag{5.23}$$

If we use Eqn. 3.20 for the derivative of G in Eqn. 5.4, and make use of the definitions, Eqns. 5.20 through 5.23, we obtain

$$K_{n-1} = \int_0^1 dt(1-t)^{n-1} \left(\frac{c}{d_t}\right)^n g(d_t) \sum_{m=0}^{m \le \frac{n}{2}} \frac{n(-1)^n}{2^m (n-2m)!m!} g^{(n-m)}(d_t) \mu_t^{n-2m}. \tag{5.24}$$

The equivalent Cauchy form of Eqn. 5.24 is

$$K_{n-1}(x) = (-1)^n g(d_t) \left(\frac{c}{d_t}\right)^n (1-t)^{n-1} \sum_{m=0}^{m \le \frac{n}{2}} \frac{n}{2^m (n-2m)!m!} g^{(n-m)}(d_t) \mu_t^{n-2m}. \tag{5.25}$$

It is clear, from Eqn. 5.14 that

$$c \le b, \tag{5.26}$$

and from Eqns. 5.20 through 5.24 that

$$|\mu_t| \le 1. \tag{5.27}$$

Thus, Eqn. 5.25 implies

$$|K_{n-1}(x)| \le n\,|g(d_t)| \left(\frac{b}{d_t}\right)^n (1-t)^{n-1} \sum_{m=0}^{m \le \frac{n}{2}} \frac{1}{2^m (n-2m)!m!} \left|g^{(n-m)}(d_t)\right|, \tag{5.28}$$

and, finally, setting $t = 0$, as that provides us with an upper bound,

$$|K_{n-1}(x)| \leq n\,|g(d_t)| \left(\frac{b}{d_{min}}\right)^n \sup_{z \in \mathcal{V}_\gamma'} \sum_{m=0}^{m \leq \frac{n}{2}} \frac{1}{2^m (n-2m)! m!} \left| g^{(n+1-m)}(|\vec{r} - \vec{z}|) \right|, \tag{5.29}$$

where we have defined $\mathcal{V}_\gamma'$ to be the "shadow" of $\mathcal{V}_\gamma$,

$$\mathcal{V}_\gamma' = \{z \mid z = \vec{r}_\gamma + t(\vec{x} - \vec{r}_\gamma), t \in [0,1], x \in \mathcal{V}_\gamma\}, \tag{5.30}$$

and d_{min} is the shortest distance from the point, $\vec{r}$, to the region, $\mathcal{V}_\gamma'$.

The normalized derivatives, $g^{(n)}$, are dimensionless, although they may well depend on some internal length-scale. Nevertheless, we have found that d_{min} is a natural unit of length, which is much more definite than the unspecified D_n of Eqn. 5.17. We now have a very powerful result. For a spherically symmetric Green's function, the magnitude of the error term is bounded by the $n+1$ power of the radius of $\mathcal{V}_\gamma$ divided by the distance from $\vec{r}$ to the shadow of $\mathcal{V}_\gamma$. Nevertheless, we still have an arbitrary function, $g(r)$, and its derivatives in our expression for $\Phi_{\gamma(n)}$. In order to make more specific claims, we must specify a precise functional form for $g(r)$.

5.1.3. A softened Newtonian potential.

We now return to the example of the softened Newtonian potential. With a completely specified Green's function, we can, in principle, evaluate $\Phi_{\gamma(n)}$ for any particular values of $\vec{x}$, $\vec{r}_\gamma$ and $\vec{r}$. Our first step is to simplify the expression for K_n, and to cast it in terms of as few independent variables as possible.

We simply use Eqn. 3.32 in Eqn. 5.24, and obtain

$$K_{n-1}(x) = n \int_0^1 g(d_t) \left(\frac{c}{R_t}\right)^n (1-t)^{n-1} \sum_{m=0}^{m \leq n/2} \frac{(-1)^m (2n-2m-1)!!}{m! 2^m (n-2m)!} \nu_t^{n-2m}, \tag{5.31}$$

where

$$R_t = (d_t^2 + \epsilon^2)^{\frac{1}{2}}, \tag{5.32}$$

$$\nu_t = \mu_t \frac{d_t}{R_t} \leq \mu_t \leq 1. \tag{5.33}$$

The summation in Eqn. 5.31 is precisely identical to the Legendre polynomial, $P_n(\nu_t)$, [34] Thus, we have

$$K_n(x) = (n+1)\left(-\frac{1}{d}\right)\alpha^{n+1}\int_0^1 \left(\frac{d}{R_t}\right)^{n+2}(1-t)^n P_{n+1}(\nu_t). \tag{5.34}$$

We find that K_n may be computed in terms of μ, α and β, where

$$\alpha \equiv \frac{c}{d}, \tag{5.35}$$

$$\beta \equiv \frac{\epsilon}{d}, \tag{5.36}$$

$$\mu \equiv \frac{\hat{e}\cdot(\vec{x}-\vec{r}_\gamma)}{c}. \tag{5.37}$$

Simple algebraic manipulations reveal that

$$\frac{R_t}{d} = \left(1 - 2\mu\alpha t + \alpha^2 t^2 + \beta^2\right)^{\frac{1}{2}}, \tag{5.38}$$

$$\nu_t = (\mu - \alpha t)\frac{d}{R_t}. \tag{5.39}$$

The situation is now well-defined enough so that it is worthwhile to compute the gradient of the K_n. A somewhat tedious exercise in algebra reveals that

$$\vec{\nabla}K_n(x) = -\frac{1}{d^2}\alpha^{n+1}(n+1)\int_0^1 dt(1-t)^n\left(\frac{d}{R_t}\right)^{n+3} \tag{5.40}$$
$$\left((n+2)P_{n+1}(\nu_t)\vec{h}_t - (\hat{x}_\gamma - \nu_t\vec{h}_t)P'_{n+1}(\nu_t)\right),$$

where

$$\vec{h}_t \equiv \frac{d_t}{R_t}\hat{e}_t = \frac{d}{R_t}(\hat{e} - \alpha t\hat{x})$$
$$\hat{x}_\gamma = \frac{\vec{x}-\vec{r}_\gamma}{c}. \tag{5.41}$$

We can express the gradient of K_n as a sum of two orthogonal parts, one parallel to $\hat{e}$ and one perpendicular,

$$\vec{\nabla}K_n(x) = -\frac{(n+1)\alpha^{n+1}}{d^2}(K_e\hat{e} + K_\perp(\hat{x} - \mu\hat{e})), \tag{5.42}$$

where

$$K_e(x) = \int_0^1 dt(1-t)^n \left(\frac{d}{R_t}\right)^{n+4} \left((n+2)(1-\mu\alpha t)P_{n+1}(\nu_t) - \left(\frac{d}{R_t}\right)^2 \left(\alpha t(1-\mu^2)+\beta^2\right) P'_{n+1}(\nu_t) \right) \tag{5.43}$$

$$K_\perp(x) = \int_0^1 dt(1-t)^n \left(\frac{d}{R_t}\right)^{n+3} \left(\left(1+\left(\frac{d}{R_t}\right)^2 (\mu-\alpha t)\right) P'_{n+1}(\nu_t) + (n+2)\alpha t \frac{d}{R_t} P_{n+1}(\nu_t) \right). \tag{5.44}$$

With explicit expressions for K_n and its gradient, we can now quantitatively address several issues regarding the multipole series.

5.1.3.1. Convergence of the multipole series.

First, we investigate the conditions under which the multipole series eventually converges as n grows arbitrarily large. Upon noting that the Legendre polynomials and their derivatives are bounded, inspection of Eqns. 5.34 and 5.40 reveals that $K_n(x)$ and its gradient approach zero with increasing n, if

$$\frac{(1-t)d\alpha}{R_t} < 1; \qquad \text{for } 0 < t < 1. \tag{5.45}$$

Using Eqn. 5.38, we find that Eqn. 5.45 is equivalent to

$$\alpha^2 < 1 + \beta^2. \tag{5.46}$$

We note that Eqns. 5.22, 5.35 and 5.14 imply

$$\alpha < \frac{b}{d}, \tag{5.47}$$

so the following is sufficient to guarantee Eqn. 5.46, and hence convergence of the series:

$$d^2 > b^2 - \epsilon^2. \tag{5.48}$$

Thus, the multipole expansion of a softened Newtonian potential converges for all points, $\vec{r}$, outside a ball of radius, $(b^2 - \epsilon^2)^{\frac{1}{2}}$, centered on the point, $\vec{r}_\gamma$.

Knowledge of the convergence of a series is of little practical use because one rarely has the facilities to explicitly evaluate the sum of an infinite number of terms. In practice, it is much more useful to understand the behavior of K_n and its gradient for fixed, reasonably small, values of n. In order to simplify the analysis, and the presentation of results, we shall defer this discussion until we treat the case of pure Newtonian gravity, in which case the parameter β vanishes, and there are only two free parameters to adjust, α and μ.

5.1.4. Newtonian potential.

The situation for a purely Newtonian potential is essentially the same as for the softened version. For a purely Newtonian potential, ϵ, or equivalently β, may be taken to be zero in any of the equations in the previous section. This implies that R_t may be replaced by d_t, and ν_t may be replaced by μ_t. Thus,

$$K_n(x) = (n+1)\left(-\frac{1}{d}\right)\alpha^{n+1}\int_0^1 \left(\frac{d}{d_t}\right)^{n+2}(1-t)^n P_{n+1}(\mu_t). \tag{5.49}$$

5.1.4.1. Bounds on $\Phi_{\gamma(n)}$.

It is easy to place bounds on Eqn. 5.49. First we note that the Legendre polynomials are bounded by

$$P_n(\mu) \leq 1 \qquad |\mu| \leq 1 \tag{5.50}$$

and

$$|P_n(\mu)| = 1 \qquad \text{iff } |x| = 1. \tag{5.51}$$

Furthermore, we observe that if $\mu = 1$, we have the following identities:

$$\begin{aligned} \frac{d}{d_t} &= \frac{1}{1-\alpha}, \\ \mu_t &= 1, \end{aligned} \tag{5.52}$$

and finally, we observe that for fixed α and t,

$$d_t(\mu) < d_t(\mu = 1) \qquad |\mu| < 1. \tag{5.53}$$

Based upon Eqns. 5.49, 5.50 and 5.53, we conclude that the maximum possible value of K_n is obtained by setting $\mu = 1$. That is

$$K_n(\alpha, \mu) \leq K_n(\alpha, \mu = 1). \tag{5.54}$$

The integral expression in Eqn. 5.49, with $\mu = 1$ becomes

$$K_n(x)|_{\mu=1} = -\frac{1}{d}(n+1)\alpha^{n+1} \int_0^1 dt \frac{(1-t)^n}{(1-\alpha t)^{n+2}}, \tag{5.55}$$

which is readily evaluated as

$$K_n(x)|_{\mu=1} = -\frac{1}{d}\frac{\alpha^{n+1}}{1-\alpha}. \tag{5.56}$$

We now combine Eqns. 5.47, 5.10 and 5.56 to obtain

$$\left|\Phi_{\gamma(n)}(r)\right| \leq \frac{M_{\gamma(0)}}{d}\frac{\left(\frac{b}{d}\right)^{n+1}}{1-\frac{b}{d}} = \left|\phi_{\gamma(0)}\right|\frac{\left(\frac{b}{d}\right)^{n+1}}{1-\frac{b}{d}}. \tag{5.57}$$

Equation 5.57 tells us that the error after n multipole terms is bounded by a constant times the magnitude of the first multipole term. The constant is a function of the ratio of the maximum extent of $\mathcal{V}_\gamma$ to d, the distance from $\vec{r}_\gamma$ to the point at which the field is evaluated. Essentially the same result as Equation 5.57 has been obtained by Greengard [13] using a different formalism, which does not treat the case of non-Newtonian potentials.

5.1.4.2. Bounds on $\vec{A}_{\gamma(n)}$.

The expression for $\left|\vec{\nabla}K_n(x)\right|$ cannot be separated into pieces, each of which is maximized at $\mu = 1$, as we did for the expression for K_n. Nevertheless, we conjecture that the upper bound on $\left|\vec{\nabla}K_n(x)\right|$ also occurs when $\mu = 1$. This conjecture is verified empirically for $n < 4$ by the figures in the following subsection. When $\mu = 1$, the expression for $\vec{\nabla}K_n(x)$ is greatly simplified,

$$\vec{\nabla}K_n(x)_{\mu=1} = -\frac{\hat{e}}{d^2}(n+1)(n+2)\alpha^{n+1} \int_0^1 dt \frac{(1-t)^n}{(1-\alpha t)^{n+3}}. \tag{5.58}$$

The integral may be explicitly evaluated, and we obtain

$$\vec{\nabla} K_n(x)_{\mu=1} = -\frac{\hat{e}}{d^2}\frac{\alpha^{n+1}}{(1-\alpha)^2}\left(n+2-\alpha(n+1)\right). \tag{5.59}$$

If we accept Eqn. 5.59 as an upper bound for $\left|\vec{\nabla} K_n(x)\right|$, then we may use it in Eqn. 5.13 to obtain

$$\left|\vec{A}_{\gamma(n)}\right| \leq \frac{M}{d^2}\frac{\alpha^{n+1}}{(1-\alpha)^2}\left(n+2-\alpha(n+1)\right) = \left|\vec{a}_{\gamma(0)}\right|\frac{\alpha^{n+1}}{(1-\alpha)^2}\left(n+2-\alpha(n+1)\right). \tag{5.60}$$

With Eqn. 5.47, we have

$$\left|\vec{A}_{\gamma(n)}\right| \leq \left|\vec{a}_{\gamma(0)}\right|\frac{\left(\frac{b}{d}\right)^{n+1}}{(1-\frac{b}{d})^2}\left(n+2-\frac{b}{d}(n+1)\right). \tag{5.61}$$

Comparing Eqn. 5.61 to Eqn. 5.57, we note some important differences. Equation 5.61 contains an additional factor of $(1-b/d)^{-1}$, and another factor approximately proportional to n. We can expect the fractional error in the acceleration after n multipole terms to be substantially greater than the error in the potential. Thus if a given number of multipole terms provides a desired level of accuracy in the computation of potentials, it will not necessarily provide the same level of accuracy in the computation of accelerations. The best choice of parameters for computing the potential is not necessarily the best for computing the acceleration, to a specified level of accuracy. This is an important consideration in the application of Greengard's algorithm to the computation of accelerations. Greengard computes only errors in the potential, and estimates the number of multipoles based on those errors. If accelerations are required, then additional analysis is required to determine the magnitude of the errors. The result will certainly resemble Eqn. 5.61, and will likely imply that even more multipole terms are needed for the accurate computation of accelerations than for the computation of potentials.

5.1.4.3. Values of $K_n(\alpha, \mu)$.

Equations 5.57 and 5.61 give firm upper bound on the error in the potential after n multipole terms. Recall that the true error is actually an integral of the mass density weighted by the function, $K_n(x)$. In the worst case, the mass density may be concentrated at the point, x, which gives rise to the maximum magnitude of K_n, and hence, Eqns. 5.57 and 5.61. In general, however, the mass density will be more evenly distributed, and it is useful to gain an understanding of the behavior, $K_n(x)$, and its gradient over their entire domain. Figures 5.1, 5.3 and 5.5 are contour plots of $K_n(\alpha, \mu)$ with contours every 0.05 from -0.5 to 0.5. The factor of $-\frac{1}{d}$ has been factored out, so that the contours represent a dimensionless quantity. Similarly, Figures 5.2, 5.4 and 5.6 are contour plots of $\left|\vec{\nabla} K_n(\alpha, \mu)\right|$, with contours every 0.05 from 0.0 to 0.5. The factor of $\frac{1}{d^2}$ has been eliminated, so the contours represent a dimensionless quantity. These figures provide a picture of the behavior of K_n over its entire domain.

Several points are worth noting in these plots.

1. As μ ranges from -1 to 1, the value of K_n changes sign. Averaged over μ, for any given fixed value of α, the error vanishes. This will be demonstrated analytically in Section 5.2.
2. For a given value of α, the typical value of $K_n(x)$ is much less than the maximum value, which occurs at $\mu = 1$. This suggests that the worst-case analysis that leads to the error bound of Eqn. 5.57 may be very pessimistic. The typical error (in the potential) may be much less than the maximum error.
3. The same is true, but to a much lesser extent, for the gradient of K_n. Thus, we can again expect that the statistically typical case will be somewhat better than the worst-case of Eqn. 5.61, but the improvement will not be as great as for the potential.
4. These graphs may be used to estimate the behavior of a particular choice of opening criterion in the Barnes-Hut algorithm. As we saw in Chapter 4, the

OpeningCriterion controls the range of values of α for which the multipole approximation is used. Once the range of α values is determined, Figures 5.-5 through 5.0 may be consulted to estimate the errors that will be introduced by multipole approximations of given order. We will return to this in Chapter 6.

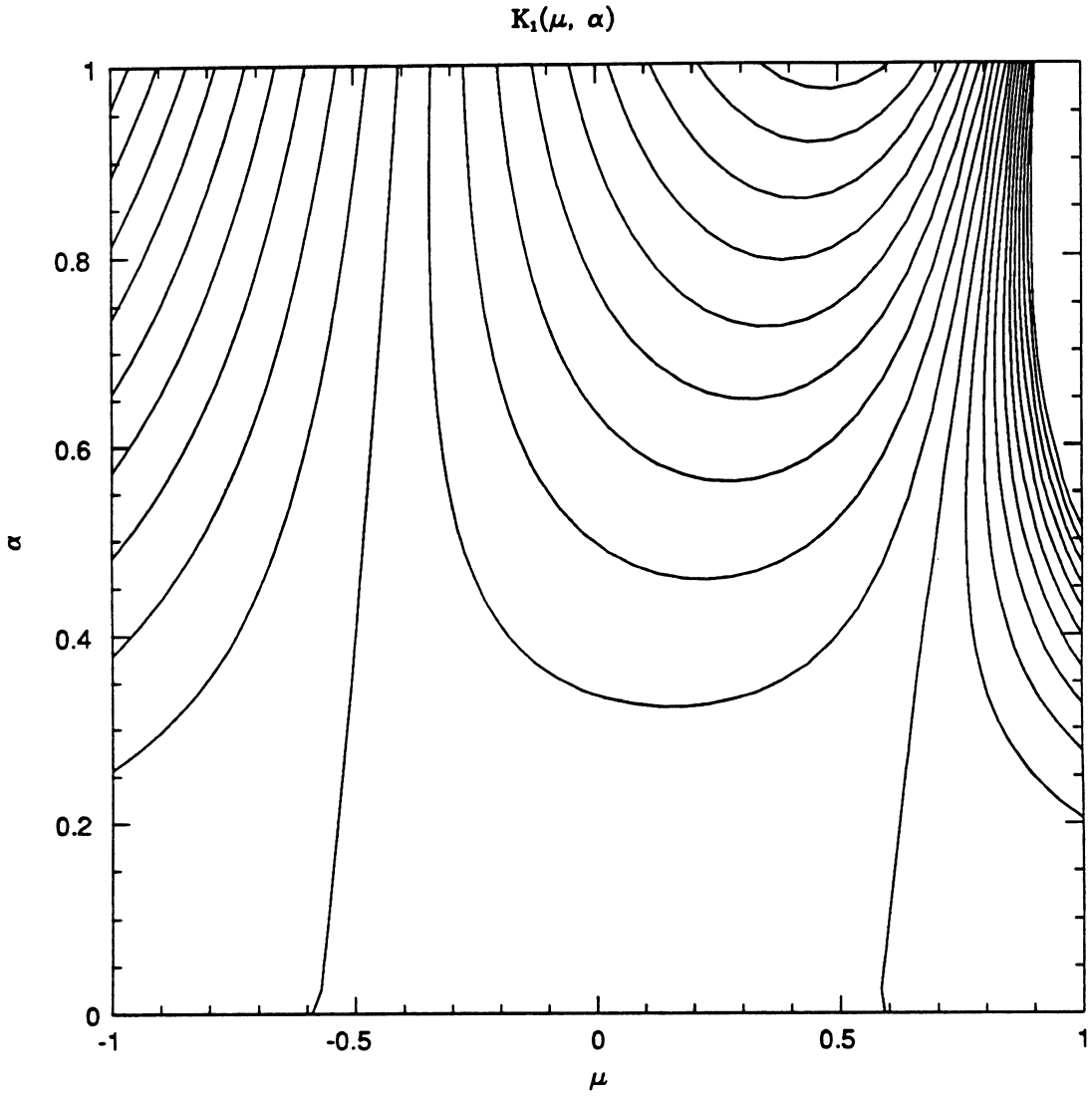


Figure 5.1. Magnitude of the error in the field, ϕ, after $n = 1$ multipole terms.

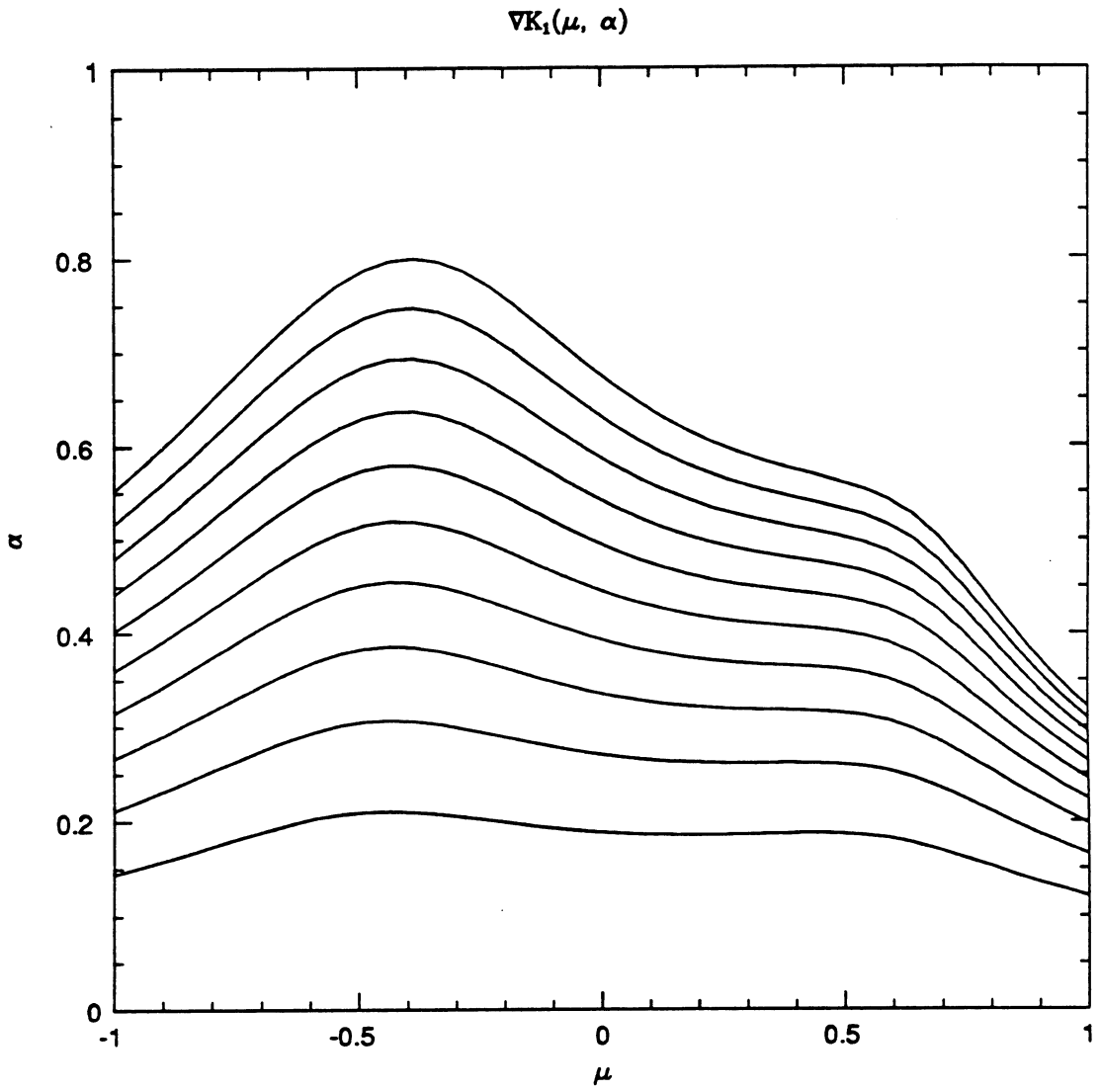


Figure 5.2. Magnitude of the error in the acceleration, $-\vec{\nabla}\phi$, after $n = 1$ multipole terms.

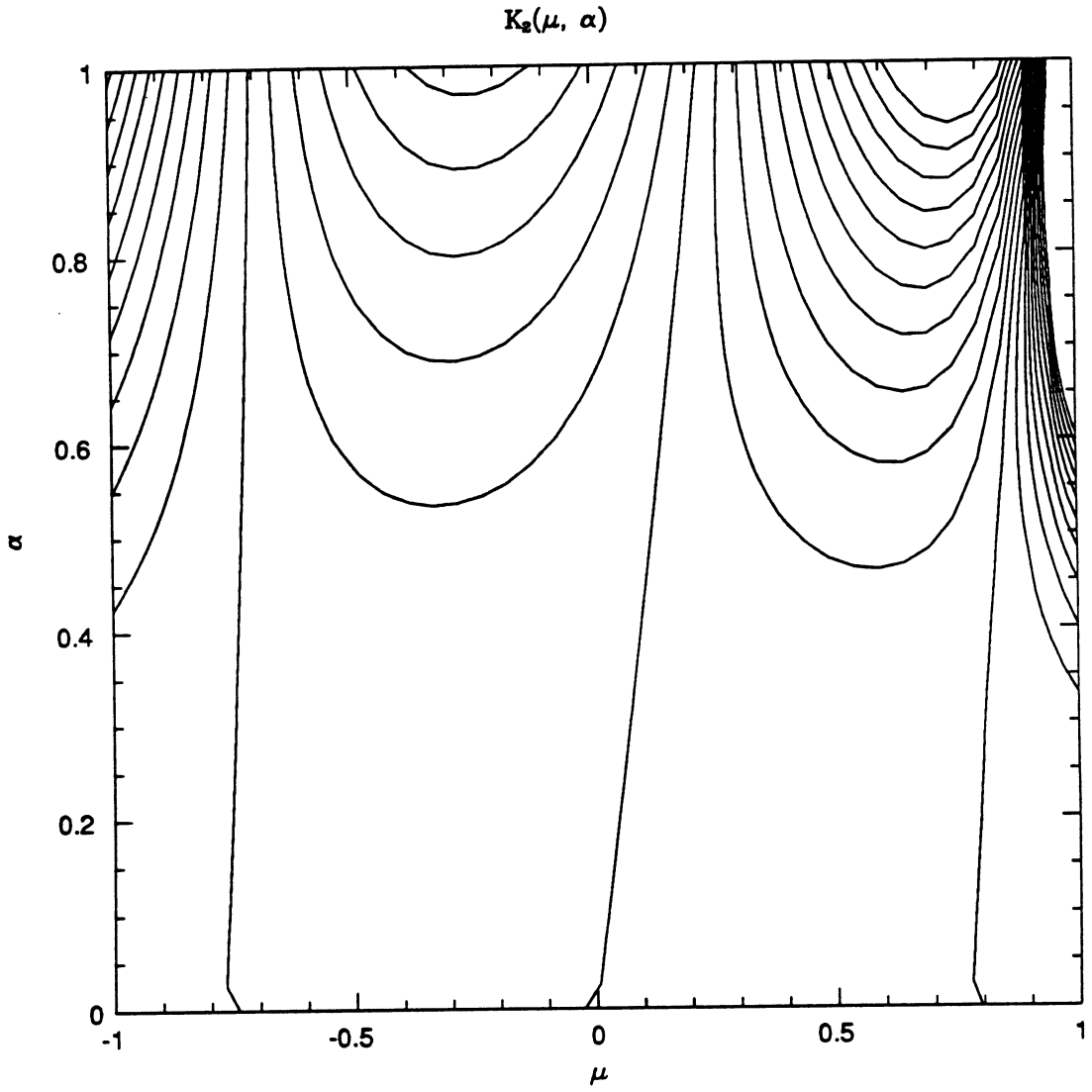


Figure 5.3. Magnitude of the error in the field, ϕ, after $n = 2$ multipole terms.

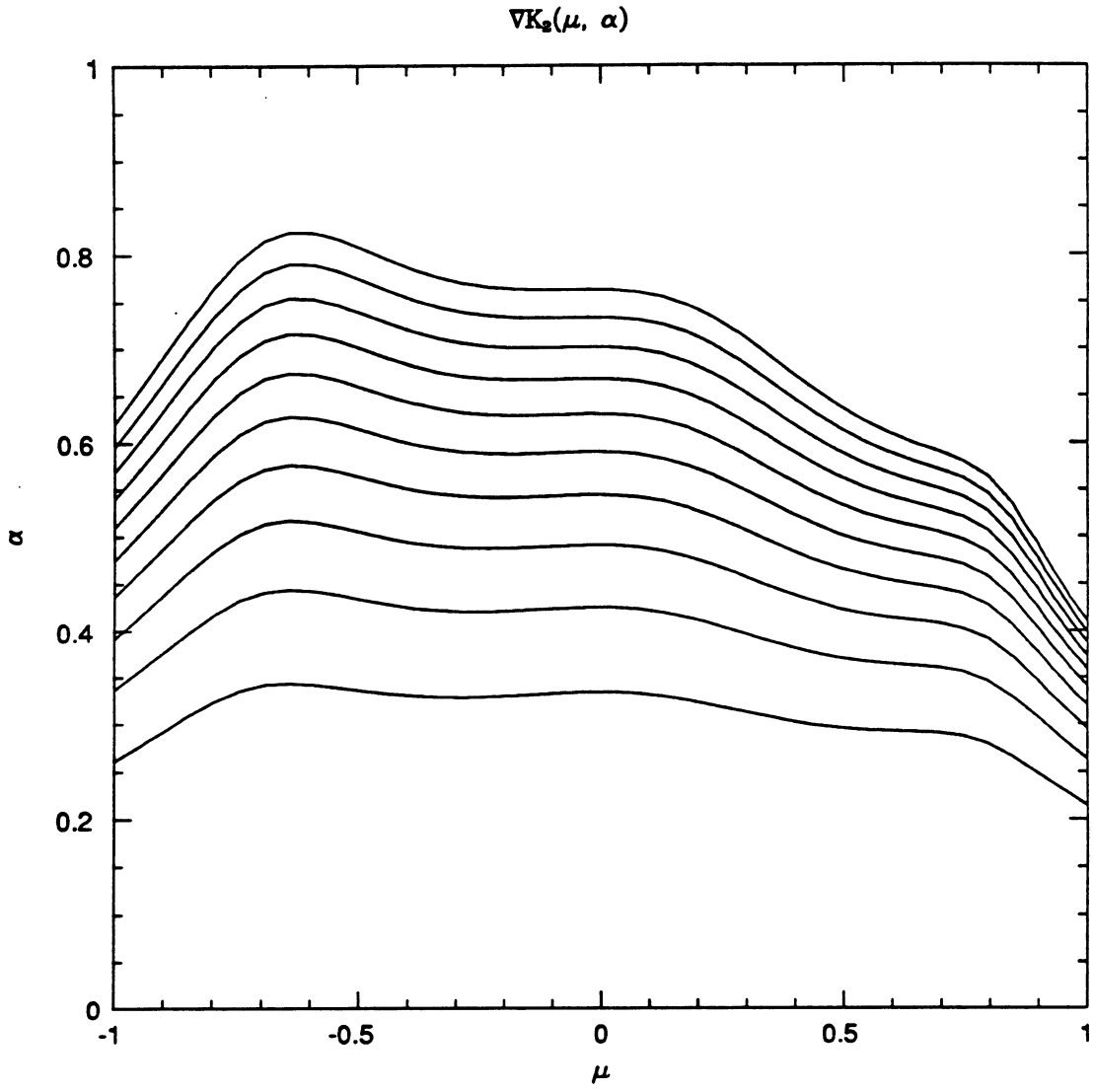


Figure 5.4. Magnitude of the error in the acceleration, $-\vec{\nabla}\phi$, after $n = 2$ multipole terms.

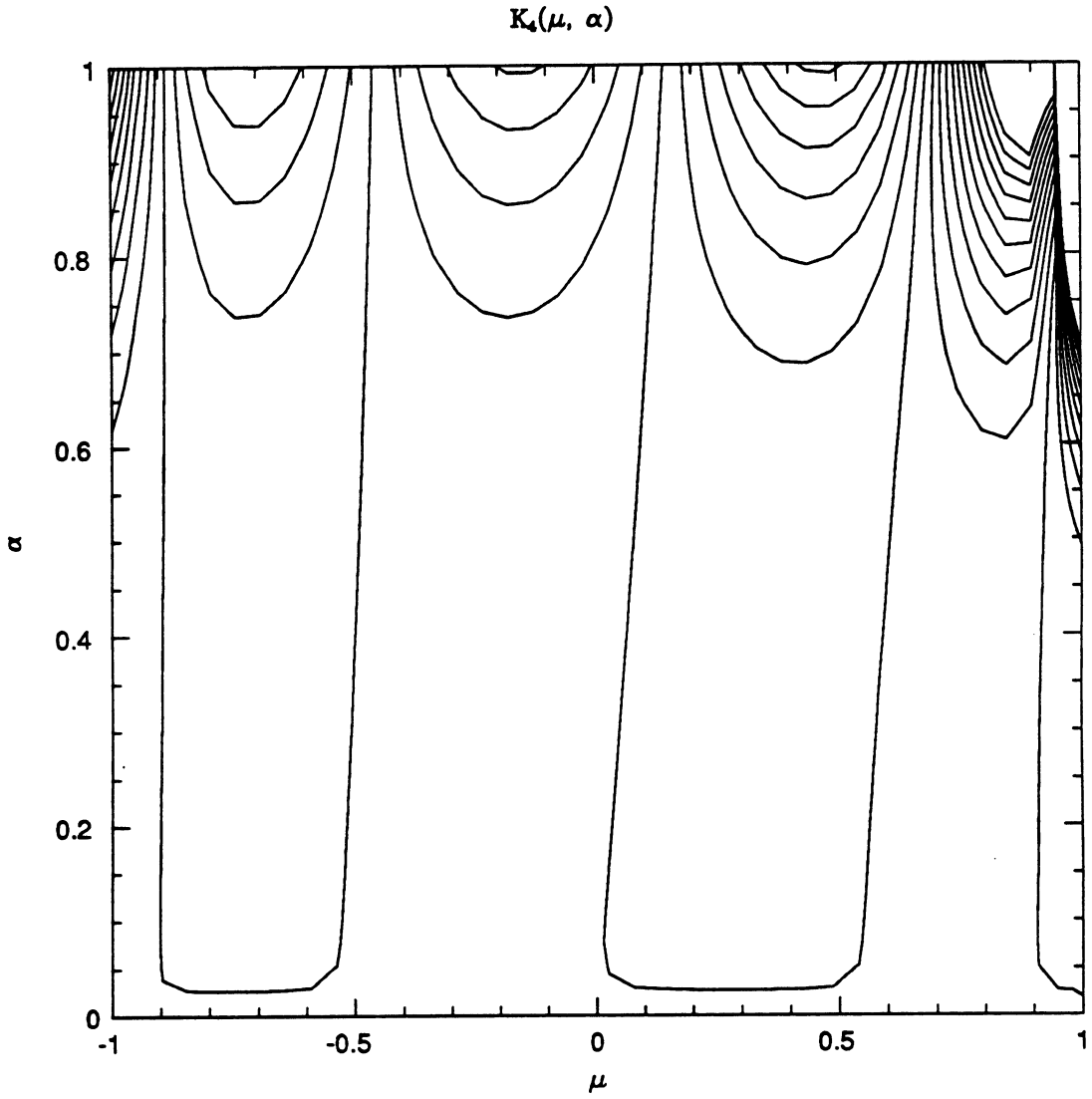


Figure 5.5. Magnitude of the error in the field, ϕ, after $n = 4$ multipole terms.

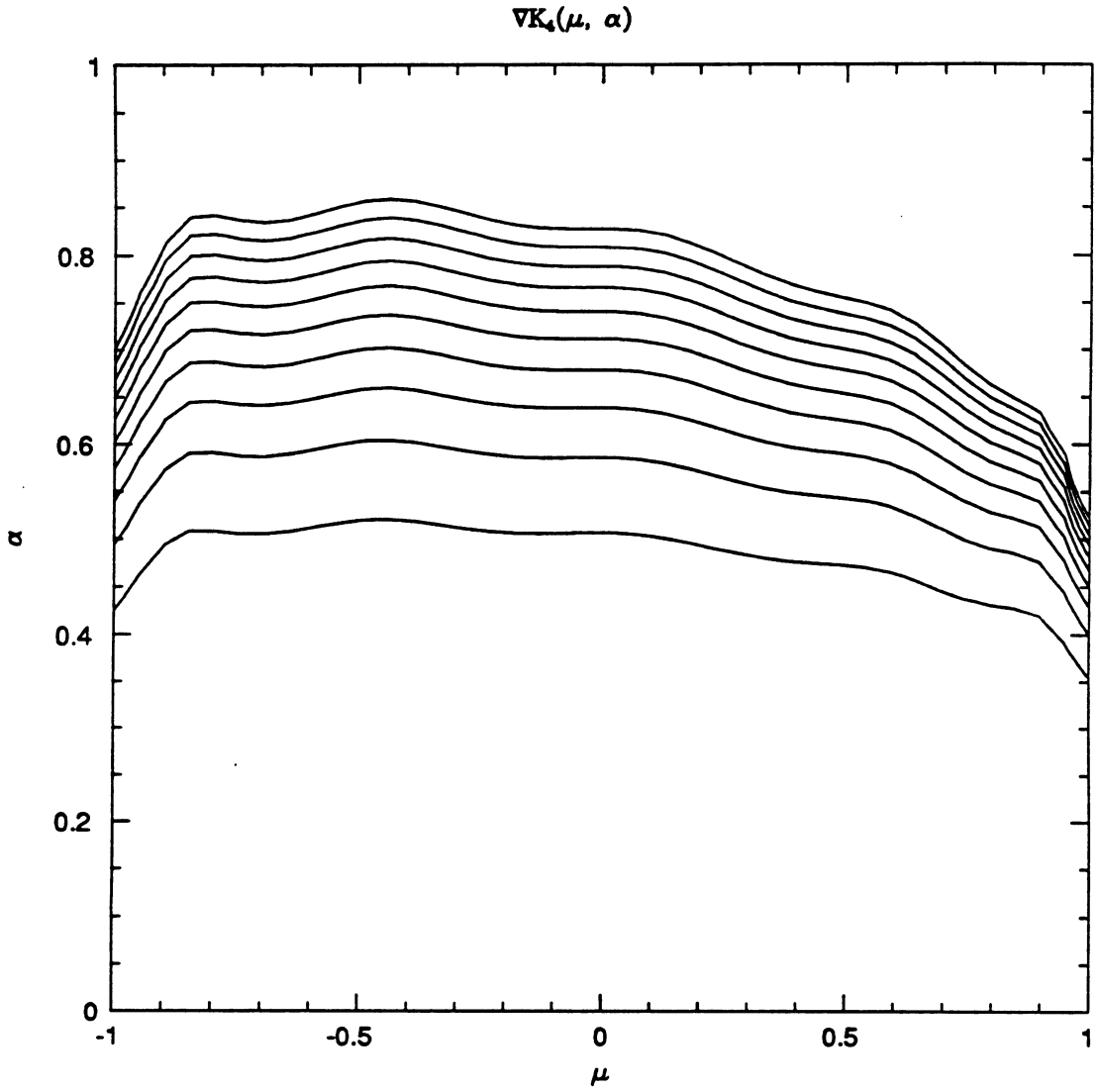


Figure 5.6. Magnitude of the error in the acceleration, $-\vec{\nabla}\phi$, after $n = 4$ multipole terms.

5.1.4.4. Contours of constant K_n^{sup}.

Another way of visualizing the error function K_n^{sup} is to plot three dimensional contours on which the value of K_n^{sup} is constant. These contours are particularly useful because they help us to formulate the appropriate criterion for deciding if a multipole approximation is valid or not. Computing K_n^{sup} for a particular value of $\vec{r}$ is a non-trivial task, naively requiring a search of the entire interior of $\mathcal{V}_\gamma$ until the point x with the maximum value is determined. We can simplify the task by observing that K_n is a strictly increasing function of α, which is proportional to the magnitude of x. Thus, we can immediately restrict our search to the boundary of $\mathcal{V}_\gamma$. We can restrict the search further by observing that for any fixed value of μ and $\vec{r}$, there will be a unique maximum value of α, which corresponds to the point x on the surface of $\mathcal{V}_\gamma$ with

$$\frac{\vec{x} \cdot \hat{e}}{|\vec{x}|} = \mu, \tag{5.62}$$

$$|\vec{x}| \quad \text{maximum.} \tag{5.63}$$

The geometry is illustrated in Figure 5.7. There are two points on the boundary of the square which satisfy Eqn. 5.62. Of these, x_1 has the larger value of α. Thus, in Figure 5.7,

$$\alpha_{max} = \frac{|\vec{x}_1 - \vec{r}_\gamma|}{|\vec{r} - \vec{r}_\gamma|} \tag{5.64}$$

In three dimensions, the set of points satisfying Eqn. 5.62 will be a deformed circle, on the surface of the cube.

Thus, searching $\mathcal{V}_\gamma$ for a maximum value of $K_n(x)$ is reduced to searching the range of μ for a maximum value of

$$K_n(\mu, \alpha_{max}(\mu)). \tag{5.65}$$

Of course, α_{max} depends on $\hat{e}$ as well as the geometry of $\mathcal{V}_\gamma$.

Figures 5.8 through 5.10 show contours of constant K_n^{sup} for $\mathcal{V}_\gamma$, a cubical box with sides of length b. These figures are useful for determining an appropriate

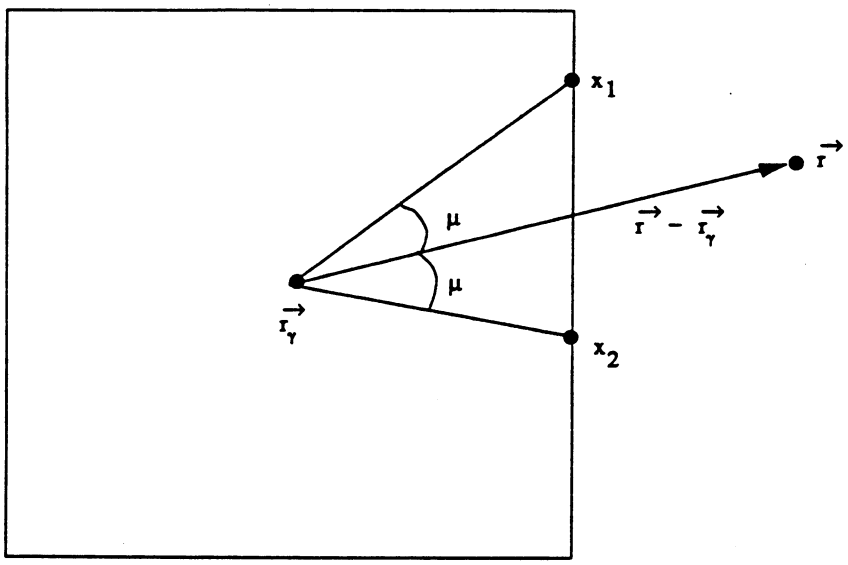


Figure 5.7. Geometry used in K_n^{sup} computations. The value of $c_{max}(\mu)$ is $|\vec{x}_1 - \vec{r}_\gamma|$. In two dimensions only two distinct points on the boundary, x_1 and x_2, satisfy the condition on the angle μ. In three dimensions, the set of points is topologically a circle. Nevertheless, there is a unique value of c_{max} analagous to x_1.

`OpeningCriterion`, as in Chapter 6, because they represent the region of space that must be excluded if one wishes to guarantee a certain accuracy from the multipole approximation. It is interesting that the surfaces in these figures are not at all round. They follow, more-or-less, the shape of the cubical volume $\mathcal{V}_\gamma$. We shall return to this observation in Chapter 6.

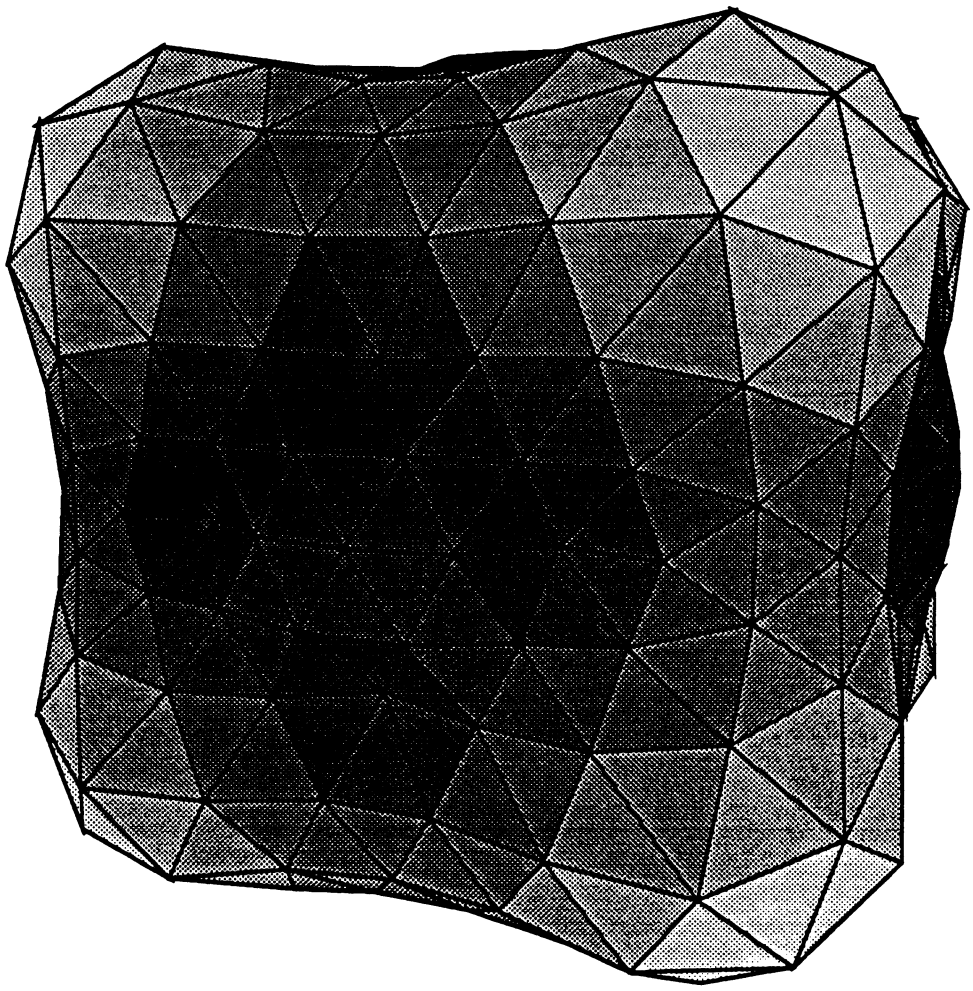


Figure 5.8. Contour of $K_2^{sup}(r) = 0.01$, maximum radius $= 4.33b$.

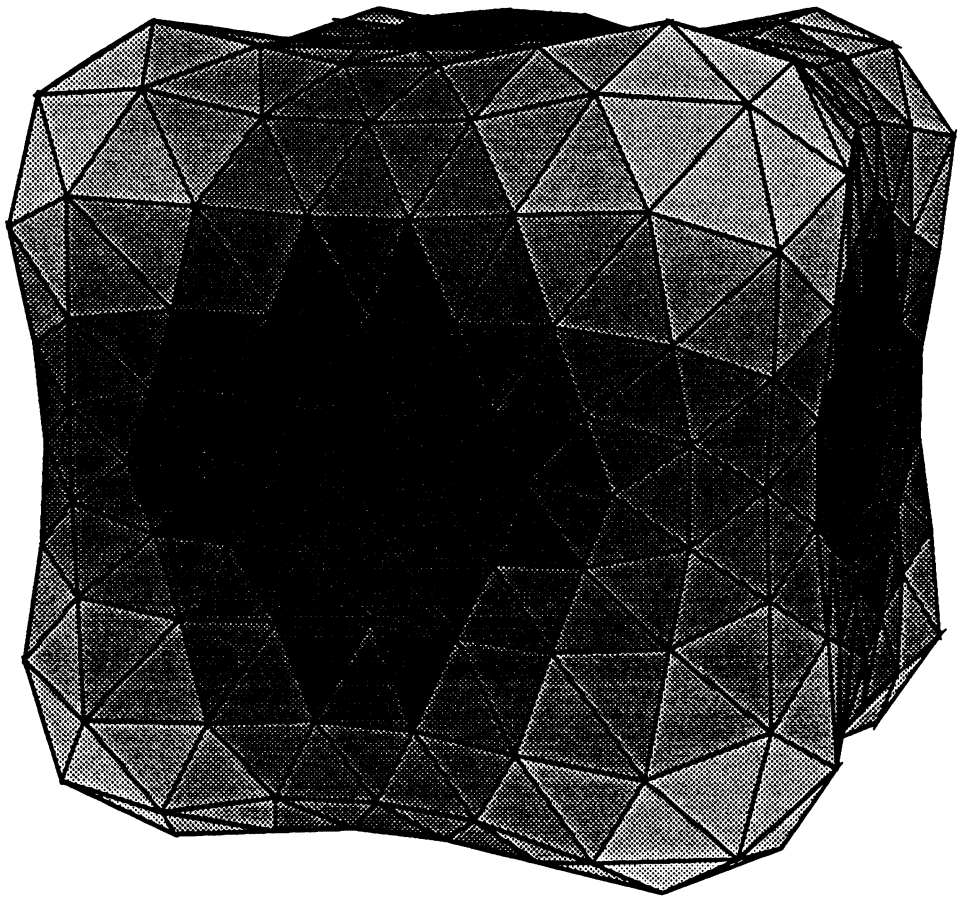


Figure 5.9. Contour of $K_2^{sup}(r) = 0.1$, maximum radius = $2.20b$.

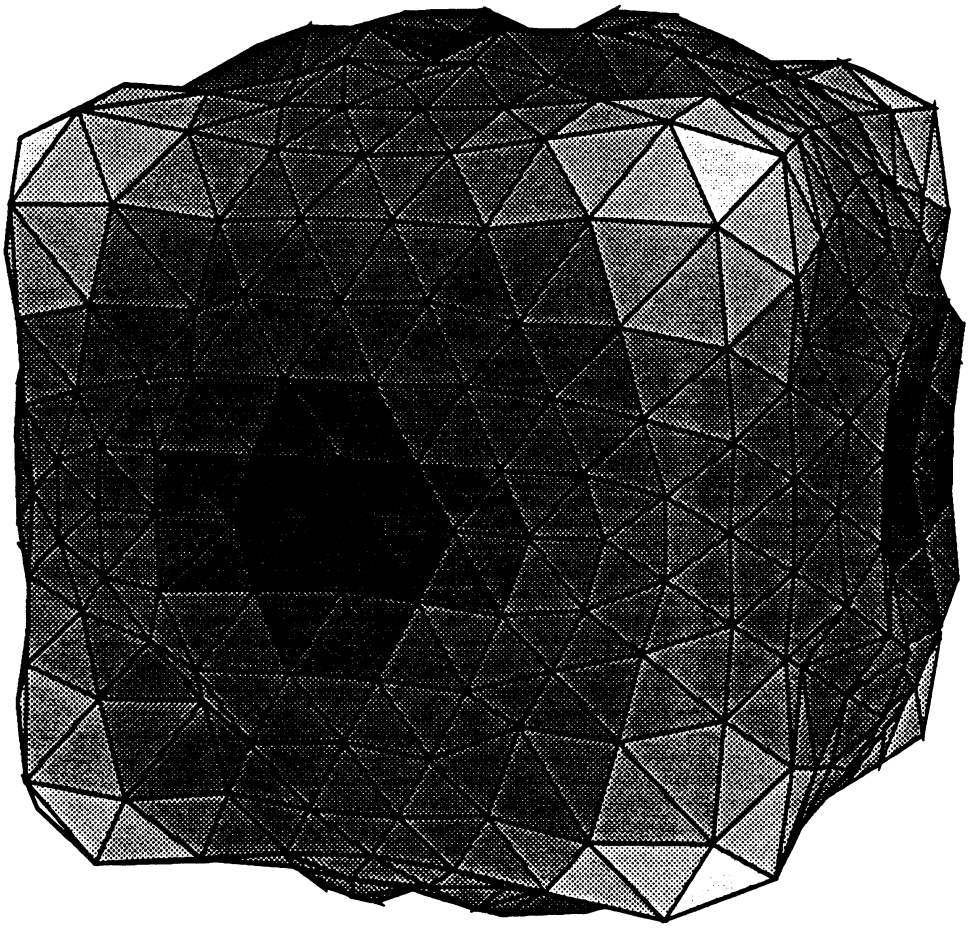


Figure 5.10. Contour of $K_4^{sup}(r) = 0.1$, maximum radius $= 1.60b$.

5.2. Statistical analysis of errors.

In the previous section we analyzed the multipole approximation and developed analytic expressions for the errors introduced by replacing the integral expression, Eqn. 3.1, with a multipole expansion. That analysis lead to a firm upper limit on the errors, Eqns. 5.57 and 5.61. We also noted that over most of its domain, the function, $K_n(\alpha, \mu)$ is much smaller than its maximum value, suggesting that the typical error may be much smaller than the upper bound. In this section we investigate some statistical properties of the multipole expansion. It has not proved possible to obtain useful results from a statistical treatment of the complete error term, i.e., $\Phi_{\gamma(n)}$ and $\vec{A}_{\gamma(n)}$. Instead, we estimate the mean and variance of each of the terms in the multipole expansion separately. Thus, in order to use the results of this section to select parameters for an algorithm that uses the multipole approximation, one looks at the mean and variance of the first few terms which are not computed by the algorithm. If these terms are acceptably small, then the affect of neglecting all higher terms will almost certainly still be acceptable.

Although the statistical analysis may be carried out for the general Green's function, definite results do not emerge until a specific form of the Green's function is chosen. In this section, we will restrict ourselves to the Newtonian potential.

Two different types of statistical averages will be studied. The first is an average over possible configurations of the density field, ρ. We treat density fields made up of collections of point particles, so this type of average is really an average over particle positions. If f is some function of the density field configuration, then we shall denote the mean of f over possible density field configurations by

$$\langle f \rangle . \tag{5.66}$$

The second type of average is over measurment position, i.e., the point, $\vec{r}$, at which the field and gradient are to be evaluated. Since the Green's function of interest is spherically symmetric, the angular and radial dependence of the errors generally decouple. It is useful to compute averages over a solid angle, i.e., the unit-vector, $\hat{e}$. If g is a function of $\vec{r}$, then the average of g over possible values of

the unit-vector, $\hat{e}$, is denoted by

$$\langle g \rangle_{\hat{e}} \equiv \frac{1}{4\pi} \int d\Omega_{\hat{e}} g(r, \hat{e}). \tag{5.67}$$

Suppose we have a density field characterized by N mass points distributed at random throughout the region $\mathcal{V}_\gamma$. Then, the density is given by

$$\rho(x) = \sum_{p=0}^{N} m_p \delta^3(\vec{x} - \vec{x}_p). \tag{5.68}$$

Let us assume that the position of each point, i.e., $\vec{x}_p$, is an independent, identically distributed random variable with probability distribution function, $\bar{\rho}(x)$. For the remainder of this section, we shall take $\vec{r}_\gamma$ to be the origin, which greatly simplifies much of the algebra. As Eqn. 3.1 is translation invariant, this simplification entails no loss of generality. In addition, we shall take the Green's function to be Newtonian.

Since the particles are independently distributed, we can use the following statements about the expectation value of a function of particle positions,

$$\langle f(x_p) \rangle = \int_{\mathcal{V}_\gamma} \bar{\rho}(x) f(x) d^3x \tag{5.69}$$

and

$$\begin{aligned} \langle f(x_p) g(x_q) \rangle = \delta_{pq} \int_{\mathcal{V}_\gamma} f(x) g(x) \bar{\rho}(x) d^3x \\ + (1 - \delta_{pq}) \int_{\mathcal{V}_\gamma} f(x) \bar{\rho}(x) d^3x \int_{\mathcal{V}_\gamma} g(y) \bar{\rho}(y) d^3y. \end{aligned} \tag{5.70}$$

The total mass inside $\mathcal{V}_\gamma$, M, is given by

$$M = \sum_{p=1}^{N} m_p. \tag{5.71}$$

The multipole moments (and their traces) of this mass distribution are

$$M_{\gamma(n)}^{(n-2m)i_1 i_2 \cdots i_{n-2m}} = \sum_{p=1}^{N} m_p \left|x_p\right|^{2m} x_p^{i_1} \cdots x_p^{i_{n-2m}}. \tag{5.72}$$

By straightforward application of Eqn. 3.34, we obtain

$$\phi_{(\gamma n)}(\vec{r}) = \left(\frac{-1}{d^{n+1}}\right) \sum_{p=1}^{N} m_p \sum_{m=0}^{m \leq n/2} \frac{(-1)^m (2n-2m-1)!!}{m! 2^m (n-2m)!} \mu_p^{n-2m} \left|x_p\right|^n, \tag{5.73}$$

where

$$\mu_p = \frac{\vec{x}_p \cdot \vec{r}}{|\vec{x}_p|\,|\vec{r}|}. \tag{5.74}$$

The summation in Eqn. 5.73 is the Legendre polynomial, $P_n(\mu_p)$,[34] so Eqn. 5.73 may be written as

$$\phi_{\gamma(n)}(\vec{r}) = \left(-\frac{1}{d^{n+1}}\right) \sum_{p} m_p P_n(\mu_p) \left|x_p\right|^n. \tag{5.75}$$

Applying Eqn. 5.69 we obtain

$$\left\langle\!\left\langle \phi_{\gamma(n)}(r) \right\rangle\!\right\rangle = \phi_{\gamma(0)} \int_{\mathcal{V}_\gamma} P_n(\mu) \left|x\right|^n \bar{\rho}(x) d^3x \equiv \overline{\phi}_{\gamma(n)}(r). \tag{5.76}$$

In other words, the expected magnitude of the n^{th} multipole correction to a collection of N mass points is precisely equal to the magnitude of the n^{th} multipole correction to the underlying probability distribution that governs the position of those mass points.

A similar result follows for the acceleration at the point, r, due to the collection of mass points. Taking the gradient of Eqn. 5.75, we obtain

$$\vec{a}_{\gamma(n)}(r) = -\frac{1}{d^{n+2}} \sum_{p} m_p \left(\left((n+1)P_n(\mu_p)\right) \hat{e} - (1-\mu_p^2)^{\frac{1}{2}} P_n'(\mu_p) \hat{x}_{\perp p} \right), \tag{5.77}$$

$$\left\langle\left\langle \vec{a}_{\gamma(n)}(r)\right\rangle\right\rangle = -\left|\vec{a}_{\gamma(0)}\right|\left(\frac{1}{d^n}\right)$$
$$\int_{\mathcal{V}_\gamma} d^3x\bar{\rho}(x)\left|x\right|^n\left(((n+1)P_n(\mu))\,\hat{e}-(1-\mu^2)^{\frac{1}{2}}P_n'(\mu)\hat{x}_\perp\right),$$
$$\equiv \bar{\bar{a}}_{\gamma(n)} \tag{5.78}$$

where

$$\hat{x}_\perp \equiv \frac{\hat{x}-\mu\hat{e}}{(1-\mu^2)^{\frac{1}{2}}} \tag{5.79}$$

is a unit-vector perpindicular to $\hat{e}$, in the plane of $\hat{e}$ and $\hat{x}$. The result is similar to that of Eqn. 5.76. The expectation of the acceleration from the set of N mass points is again exactly that which one expects from the underlying distribution that governs the particle positions.

In order to estimate fluctuations in the n^{th} multipole term due to the finite number of discrete mass points, we compute the mean square values of the potential and acceleration. Applying Eqn. 5.70, we obtain

$$\left\langle\left\langle \phi^2_{\gamma(n)}(r)\right\rangle\right\rangle = \frac{1}{N_{eff}}\phi^2_{\gamma(0)}\frac{1}{d^{2n}}\int_{\mathcal{V}_\gamma}\left(P_n(\mu)\left|x\right|^n\right)^2\bar{\rho}(x)d^3x+\left(1-\frac{1}{N_{eff}}\right)\bar{\phi}^2_{\gamma(n)}(r) \tag{5.80}$$

and

$$\left\langle\left\langle \left|\vec{a}_{\gamma(n)}(r)\right|^2\right\rangle\right\rangle = \left(1-\frac{1}{N_{eff}}\right)\left|\bar{\bar{a}}_{\gamma(n)}\right|^2 \tag{5.81}$$
$$+\frac{1}{N_{eff}}\left|\vec{a}_{\gamma(0)}\right|^2\int_{\mathcal{V}_\gamma} d^3x\bar{\rho}(x)\left|x\right|^{2n}\left((n+1)^2P_n^2(\mu)+(1-\mu^2)P_n'^2(\mu)\right),$$

where N_{eff} is the effective number of particles in the sample,

$$N^2_{eff} = \frac{M^2}{\sum_p m_p^2}. \tag{5.82}$$

We now have two contributions to the mean square of the n^{th} order potential and acceleration. There is a term proportional to the square of the n^{th} multipole

of the undrerlying probability distribution, and a second "statistical" term proportional to N_{eff}^{-1} times the monopole term times a function of order unity, which depends on the underlying probability distribution.

5.2.1. Uniform distribution in a sphere.

Let us now consider the situation in which $\mathcal{V}_\gamma$ is a sphere of radius, b, and the underlying probability distribution, $\bar{\rho}(x)$, is constant.

The orthogonality of the Legendre polynomials allow us to explicitly evaluate the integrals of the previous section very easily. We use the identities:

$$\begin{aligned} \int_{-1}^{1} d\mu P_n(\mu) P_{n'}(\mu) &= \frac{2\delta_{nn'}}{2n+1}, \\ \int_{-1}^{1} d\mu (1-\mu^2)(P_n'(\mu))^2 &= \frac{2n(n+1)}{2n+1}, \end{aligned} \tag{5.83}$$

to obtain

$$\begin{aligned} \left\langle\!\left\langle \phi_{\gamma(n)}(r) \right\rangle\!\right\rangle &= 0 \qquad n > 0, \\ \left\langle\!\left\langle \vec{a}_{\gamma(n)}(r) \right\rangle\!\right\rangle &= 0 \qquad n > 0, \end{aligned} \tag{5.84}$$

and

$$\left\langle\!\left\langle \phi^2_{\gamma(n)}(r) \right\rangle\!\right\rangle = \frac{1}{N_{eff}} \phi^2_{\gamma(0)} \left(\frac{b}{d}\right)^{2n} \frac{3}{(2n+1)(2n+3)}, \tag{5.85}$$

$$\left\langle\!\left\langle \left|\vec{a}_{\gamma(n)}(r)\right|^2 \right\rangle\!\right\rangle = \frac{1}{N_{eff}} \left|\vec{a}_{\gamma(0)}\right|^2 \left(\frac{b}{d}\right)^{2n} \frac{3(n+1)}{(2n+3)}. \tag{5.86}$$

Thus, when the region, $\mathcal{V}_\gamma$, is a sphere, and N bodies are distributed uniformly and randomly inside the sphere, only the statistical term contributes to the mean square magnitude of the n^{th} multipole. The magnitude of this term contains a factor of N_{eff}^{-1}, which makes the mean square estimate of the error considerably less than the worst-case estimates from the previous section. This result for spherical systems suggests that in situations where the distribution of matter is reasonably uniform, the multipole approximation may, in fact, be much better than the estimates of the worst-case situation.

5.2.2. Uniform distribution in a cube.

Let us now consider the situation in which $\mathcal{V}_\gamma$ is a cube of linear dimension l, and the underlying probability distribution is uniform, i.e.,

$$\bar{\rho} = \frac{1}{l^3}. \tag{5.87}$$

Note that l is related to the "radius" of $\mathcal{V}_\gamma$, b, by

$$l = \frac{2}{\sqrt{3}} b. \tag{5.88}$$

Since the volume, $\mathcal{V}_\gamma$ is expressed most naturally in Cartesian coordinates, the algebra is greatly simplified if we return to the Cartesian expansion in terms of $Q_{\gamma(n)}$. Taking expectation values of Eqns. 3.48 and 3.49,

$$\begin{aligned}
\left\langle\!\left\langle \phi_{\gamma(n)}(r) \right\rangle\!\right\rangle &= \phi_{\gamma(0)} \frac{(2n-1)!!}{n!} \frac{\left\langle\!\left\langle \left\langle Q_{\gamma(n)} \middle| e^{(n)} \right\rangle \right\rangle\!\right\rangle}{M_{\gamma(0)} d^n} \\
\left\langle\!\left\langle \vec{a}_{\gamma(n)}(r) \right\rangle\!\right\rangle &= -\left|\vec{a}_{\gamma(0)}\right| \frac{(2n-1)!!}{n!} \frac{\left\langle\!\left\langle \left((2n+1) \left\langle Q_{\gamma(n)} \middle| e^{(n)} \right\rangle \hat{e} - n \left\langle Q_{\gamma(n)} \middle| e^{(n-1)} \right\rangle \right) \right\rangle\!\right\rangle}{M_{\gamma(0)} d^n}
\end{aligned}, \tag{5.89}$$

and, for the rms fluctuations we have

$$\begin{aligned}
\left\langle\!\left\langle \phi^2_{\gamma(n)}(r) \right\rangle\!\right\rangle &= \left(\phi_{\gamma(0)}\right)^2 \left(\frac{(2n-1)!!}{n!} \right)^2 \frac{\left\langle\!\left\langle \left\langle Q_{\gamma(n)} \middle| e^{(n)} \right\rangle^2 \right\rangle\!\right\rangle}{\left(M_{\gamma(0)} d^n\right)^2} \\
\left\langle\!\left\langle \left|\vec{a}_{\gamma(n)}(r)\right|^2 \right\rangle\!\right\rangle &= \left|\vec{a}_{\gamma(0)}\right|^2 \left(\frac{(2n-1)!!}{n!} \right)^2 \\
&\qquad \frac{\left\langle\!\left\langle \left(n^2 \left| \left\langle Q_{\gamma(n)} \middle| e^{(n-1)} \right\rangle \right|^2 + (2n+1) \left\langle Q_{\gamma(n)} \middle| e^{(n)} \right\rangle^2 \right) \right\rangle\!\right\rangle}{\left(M_{\gamma(0)} d^n\right)^2}.
\end{aligned} \tag{5.90}$$

Since the tensor, $Q_{\gamma(n)}$, is obtained by linear combination of the components of $M_{\gamma(n)}$, we begin by evaluating the expectation of $M_{\gamma(n)}$. Returning to the definition of $M_{\gamma(n)}$, and using Eqn. 5.69 we obtain

$$\left\langle\!\left\langle M^{i_1 \cdots i_n}_{\gamma(n)} \right\rangle\!\right\rangle = M_{\gamma(0)} \overline{M}^{i_1 \cdots i_n}_{\gamma(n)}, \tag{5.91}$$

where

$$\overline{M}^{i_1\cdots i_n}_{\gamma(n)} = \int_{\mathcal{V}_\gamma} x^{i_1}\cdots x^{i_n}\bar{\rho}(x)d^3x, \tag{5.92}$$

and from Eqn. 5.70 we also obtain

$$\left\langle\!\left\langle M_{\gamma(n)}\otimes M_{\gamma(n)}\right\rangle\!\right\rangle = M^2_{\gamma(0)}\left(\frac{1}{N_{eff}}\overline{M}_{\gamma(2n)} + \left(1-\frac{1}{N_{eff}}\right)\overline{M}_{\gamma(n)}\otimes\overline{M}_{\gamma(n)}\right). \tag{5.93}$$

Now we define a linear operator, $L_{(n)}$, which maps $M_{\gamma(n)}$ into $Q_{\gamma(n)}$, and recall that in general, the order of linear combinations and expectation values may be interchanged. Thus,

$$Q_{\gamma(n)} = L_{(n)}M_{\gamma(n)}, \tag{5.94}$$

and hence,

$$\begin{aligned}\left\langle\!\left\langle Q_{\gamma(n)}\right\rangle\!\right\rangle &= L_{(n)}\left\langle\!\left\langle M_{\gamma(n)}\right\rangle\!\right\rangle \\ \left\langle\!\left\langle Q_{\gamma(n)}\otimes Q_{\gamma(n)}\right\rangle\!\right\rangle &= (L_{(n)}\otimes L_{(n)})\left\langle\!\left\langle M_{\gamma(n)}\otimes M_{\gamma(n)}\right\rangle\!\right\rangle.\end{aligned} \tag{5.95}$$

Putting all this together, we have

$$\begin{aligned}\left\langle\!\left\langle Q_{\gamma(n)}\right\rangle\!\right\rangle &= M_{\gamma(0)}L_{(n)}\overline{M}_{\gamma(n)} = M_{\gamma(0)}\overline{Q}_{\gamma(n)} \\ \left\langle\!\left\langle Q_{\gamma(n)}\otimes Q_{\gamma(n)}\right\rangle\!\right\rangle &= M^2_{\gamma(0)}\left(\frac{1}{N_{eff}}(L_{(n)}\otimes L_{(n)})\overline{M}_{\gamma(2n)} + \left(1-\frac{1}{N_{eff}}\right)\overline{Q}\otimes\overline{Q}\right).\end{aligned} \tag{5.96}$$

For the case of a uniform distribution, $\bar{\rho}$, in a cube of dimension, l, the integrals that appear in the definition of $\overline{M}_{\gamma(n)}$ can be explicitly evaluated,

$$\left\langle\!\left\langle \overline{M}^{i_1\cdots i_n}_{\gamma(n)}\right\rangle\!\right\rangle = \begin{cases} \frac{l^n}{2^n}\frac{1}{(n_x+1)(n_y+1)(n_z+1)} & n_x, n_y, n_z \text{ even;} \\ 0, & \text{otherwise,}\end{cases} \tag{5.97}$$

where n_x, n_y and n_z are the number of indices from the set, $i_1\cdots i_n$, which are x, y and z, respectively. We note that the expression is completely symmetric with respect to exchange of any two indices, as well as with respect to any permutation

of x, y, and z. It is clear from Eqn. 5.97 that the expectation of $\overline{M}_{\gamma(n)}$, and hence, $\overline{Q}_{\gamma(n)}$, vanishes if n is odd.

We also introduce the notation,

$$e_n \equiv e_x^n + e_y^n + e_z^n, \tag{5.98}$$

which will be used repeatedly in the equations which follow.

Using Eqns. 3.41, 5.22 and 5.97, we easily find that the lowest multipole for which the expectation value does not vanish is $n = 4$, in which case we have

$$\begin{aligned} \left\langle\!\left\langle \overline{Q}^{xxxx}_{\gamma(4)} \right\rangle\!\right\rangle &= -\frac{l^4}{300}, \\ \left\langle\!\left\langle \overline{Q}^{xxyy}_{\gamma(4)} \right\rangle\!\right\rangle &= \frac{l^4}{600}. \end{aligned} \tag{5.99}$$

Thus, the fourth order multipole is the lowest order which does not vanish when averaged over particle configurations. The magnitude of the average value of the fourth order corrections to the potential and acceleration around a uniform cube are obtained from Eqns. 5.89 and 5.99,

$$\left\langle\!\left\langle \phi_{\gamma(4)}(r) \right\rangle\!\right\rangle = -\phi_{\gamma(0)} \left(\frac{l}{d}\right)^4 \frac{7}{960} \left(3 - 5(e_x^4 + e_y^4 + e_z^4)\right) \tag{5.100}$$

$$\left\langle\!\left\langle \vec{a}_{\gamma(4)}(r) \right\rangle\!\right\rangle = -\frac{M}{d^2} \left(\frac{l}{d}\right)^4 \frac{7}{192} \left(\hat{x} e_x \left(3 + 4e_x^2 - 9(e_x^4 + e_y^4 + e_z^4)\right) + \hat{y} \cdots\right). \tag{5.101}$$

The magnitude of the acceleration is given by

$$\begin{aligned} \left|\left\langle\!\left\langle \vec{a}_{\gamma(4)}(r) \right\rangle\!\right\rangle\right| &= \frac{M}{d^2} \left(\frac{l}{d}\right)^4 \frac{7}{192} \Big(9 - 30(e_x^4 + e_y^4 + e_z^4) \\ &\qquad + 16(e_x^6 + e_y^6 + e_z^6) + 9(e_x^4 + e_y^4 + e_z^4)^2\Big)^{\frac{1}{2}} \\ &= \frac{M}{d^2} \left(\frac{l}{d}\right)^4 \frac{7}{192} \left(12 - 30e_4 + 4e_6 + 18e_8\right). \end{aligned} \tag{5.102}$$

The maximum magnitudes of the previous expressions occur when $\hat{e}$ points in one of the cartesian directions, in which case

$$\left|\left\langle\left\langle \phi_{\gamma(4)}(r)\right\rangle\right\rangle\right|_{max} = \left|\phi_{\gamma(0)}\right| \left(\frac{l}{d}\right)^4 \frac{7}{480} \tag{5.103}$$

$$\left|\left\langle\left\langle \vec{a}_{\gamma(4)}(r)\right\rangle\right\rangle\right|_{max} = \left|\vec{a}_{\gamma(0)}\right| \left(\frac{l}{d}\right)^4 \frac{7}{96}. \tag{5.104}$$

The small numerical factor of 7/480 in Eqn. 5.103 indicates that the magnitude of $\phi_{\gamma(4)}$, although non-zero will be, on average, quite small in comparison with the monopole term for any sensible value of l/d. The numerical factor of 7/96 that appears in Eqn. 5.104 is considerably larger, indicating that the fourth-order correction to the acceleration is not likely to be completely negligible, unless l/d is sufficiently small. This is further evidence that analysis of the potential only can be misleading if one intends to compute accelerations to a desired degree of accuracy.

Although the average of the n^{th} order multipole vanishes for $n < 4$, the rms fluctuations are non-zero for smaller values of n. The magnitude of these fluctuations contains a factor of N_{eff}^{-1}, and hence, are most significant when the number of bodies in the cell is small. For any given order, the magnitudes of the fluctuations may be calculated from Eqns. 5.90 and 5.96. The results of a rather tedious calculation are tabulated here.

The details are rather cumbersome, even for small values of n. The full angular dependence of the rms values of the potential and acceleration are as follows:

$$\begin{aligned}
\left\langle\left\langle \phi^2_{\gamma(1)}\right\rangle\right\rangle &= \frac{\left|\phi^2_{\gamma(0)}\right|}{N_{eff}} \left(\frac{l}{d}\right)^2 \left(\frac{1}{12}\right), \\
\left\langle\left\langle \left|\vec{a}_{\gamma(1)}\right|^2\right\rangle\right\rangle &= \frac{\left|\vec{a}_{\gamma(0)}\right|^2}{N_{eff}} \left(\frac{l}{d}\right)^2 \left(\frac{1}{2}\right),
\end{aligned} \tag{5.105}$$

$$\left\langle\!\left\langle \phi^2_{\gamma(2)} \right\rangle\!\right\rangle = \frac{\left|\phi^2_{\gamma(0)}\right|}{N_{eff}} \left(\frac{l}{d}\right)^4 \left(\frac{13-9e_4}{480}\right),$$
$$\left\langle\!\left\langle \left|\vec{a}_{\gamma(2)}\right|^2 \right\rangle\!\right\rangle = \frac{\left|\vec{a}_{\gamma(0)}\right|^2}{N_{eff}} \left(\frac{l}{d}\right)^4 \left(\frac{47-15e_4}{160}\right), \tag{5.106}$$

$$\left\langle\!\left\langle \phi^2_{\gamma(3)} \right\rangle\!\right\rangle = \frac{\left|\phi^2_{\gamma(0)}\right|}{N_{eff}} \left(\frac{l}{d}\right)^6 \left(\frac{1000e_6-1515e_4+647}{40320}\right),$$
$$\left\langle\!\left\langle \left|\vec{a}_{\gamma(3)}\right|^2 \right\rangle\!\right\rangle = \frac{\left|\vec{a}_{\gamma(0)}\right|^2}{N_{eff}} \left(\frac{l}{d}\right)^6 \left(\frac{14000e_6-15795e_4+17007}{80640}\right). \tag{5.107}$$

The preceding formulas are quite formidable, but really just define some rather simple functions on the unit sphere. The functions obey cubic symmetry, so it is easy to find their maximum values. It is also easy to compute their values when averaged over the entire unit-sphere. When integrated over the unit-sphere, the quantity, e_n, has mean

$$\langle e_n \rangle_{\dot{e}} = \frac{3}{n+1}. \tag{5.108}$$

Thus, the relevant features of the functions appearing above are:

$$\left\langle\!\left\langle \phi^2_{\gamma(1)} \right\rangle\!\right\rangle_{max,avg} = \frac{\left|\phi^2_{\gamma(0)}\right|}{N_{eff}} \left(\frac{l}{d}\right)^2 \left(\frac{1}{12},\frac{1}{12}\right),$$
$$\left\langle\!\left\langle \left|\vec{a}_{\gamma(1)}\right|^2 \right\rangle\!\right\rangle_{max,avg} = \frac{\left|\vec{a}_{\gamma(0)}\right|^2}{N_{eff}} \left(\frac{l}{d}\right)^2 \left(\frac{1}{2},\frac{1}{2}\right), \tag{5.109}$$

$$\left\langle\!\left\langle \phi^2_{\gamma(2)} \right\rangle\!\right\rangle_{max,avg} = \frac{\left|\phi^2_{\gamma(0)}\right|}{N_{eff}} \left(\frac{l}{d}\right)^4 \left(\frac{1}{48},\frac{19}{1200}\right),$$
$$\left\langle\!\left\langle \left|\vec{a}_{\gamma(2)}\right|^2 \right\rangle\!\right\rangle_{max,avg} = \frac{\left|\vec{a}_{\gamma(0)}\right|^2}{N_{eff}} \left(\frac{l}{d}\right)^4 \left(\frac{21}{80},\frac{19}{80}\right), \tag{5.110}$$

$$\left\langle\!\left\langle \phi^2_{\gamma(3)} \right\rangle\!\right\rangle_{max,avg} = \frac{\left|\phi^2_{\gamma(0)}\right|}{N_{eff}} \left(\frac{l}{d}\right)^6 \left(\frac{1139}{181440},\frac{583}{141120}\right),$$
$$\left\langle\!\left\langle \left|\vec{a}_{\gamma(3)}\right|^2 \right\rangle\!\right\rangle_{max,avg} = \frac{\left|\vec{a}_{\gamma(0)}\right|^2}{N_{eff}} \left(\frac{l}{d}\right)^6 \left(\frac{3803}{20160},\frac{451}{2688}\right). \tag{5.111}$$

We can also plot these functions in three dimensions. The plots in Figures 5.11 through 5.16 were obtained by plotting a surface whose distance from the origin is proportional to the function evaluated at the corresponding point on a sphere. The shading from dark to light is also proportional to the value of the function being plotted. Plots such as appear in Figures 5.11 through 5.16 fail to provide a measure of the overall scale of the function. This information is available from Eqns. 5.109 through 5.111.

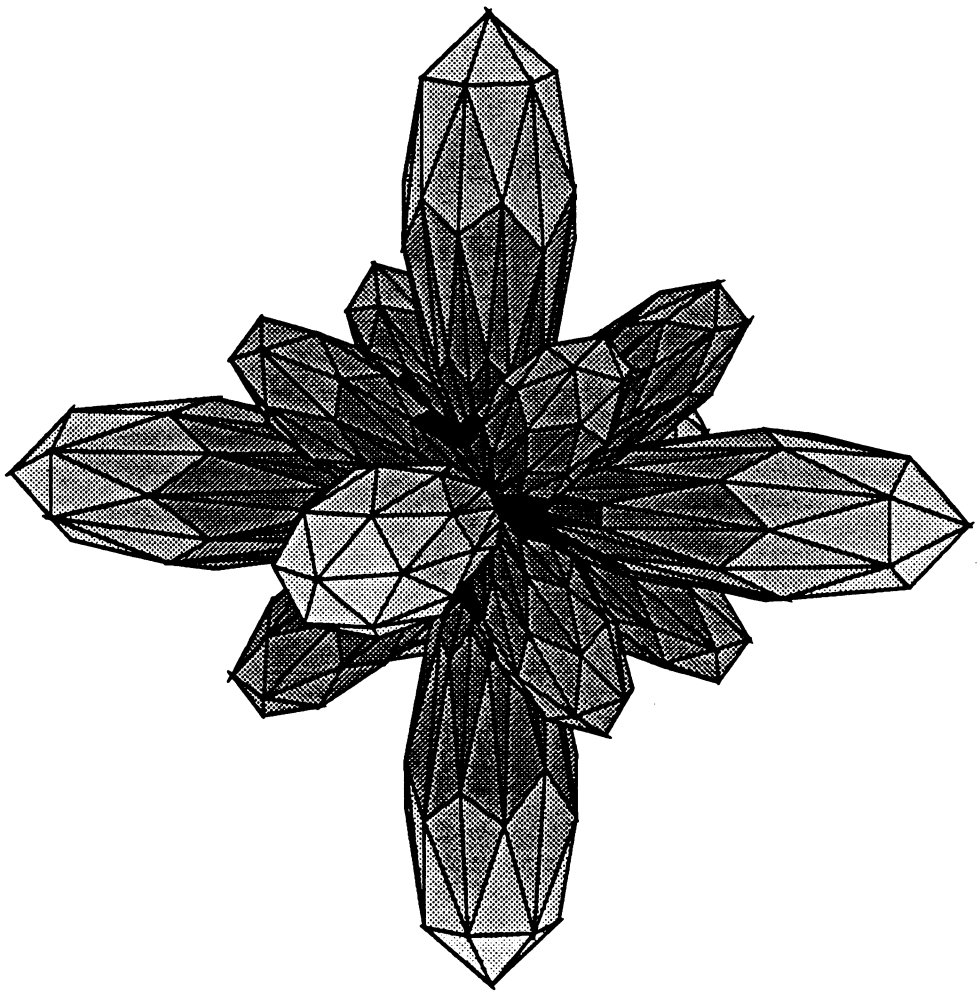


Figure 5.11. Radius proportional to $\left|\left\langle\!\left\langle \phi_{\gamma(4)}(\hat{e}) \right\rangle\!\right\rangle\right|$.

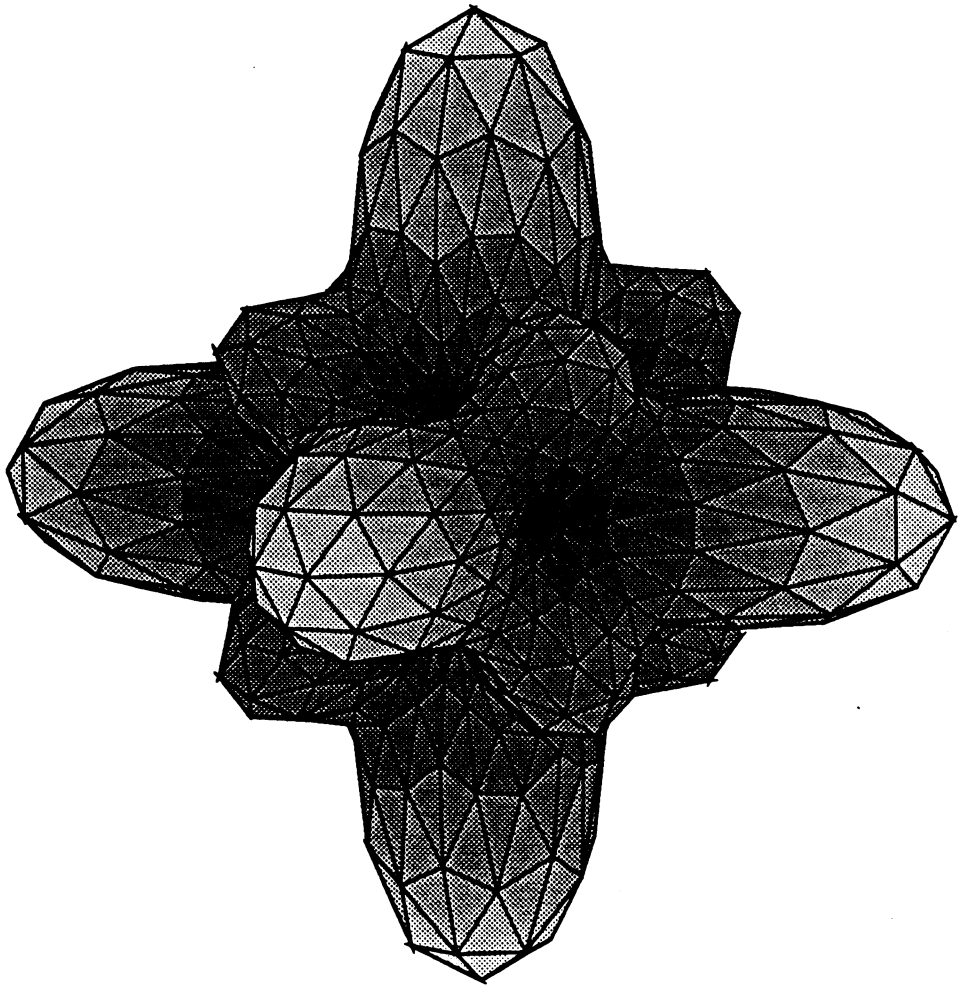


Figure 5.12. Radius proportional to $\left|\left\langle\!\left\langle \vec{a}_{\gamma(4)}(\hat{e}) \right\rangle\!\right\rangle\right|$.

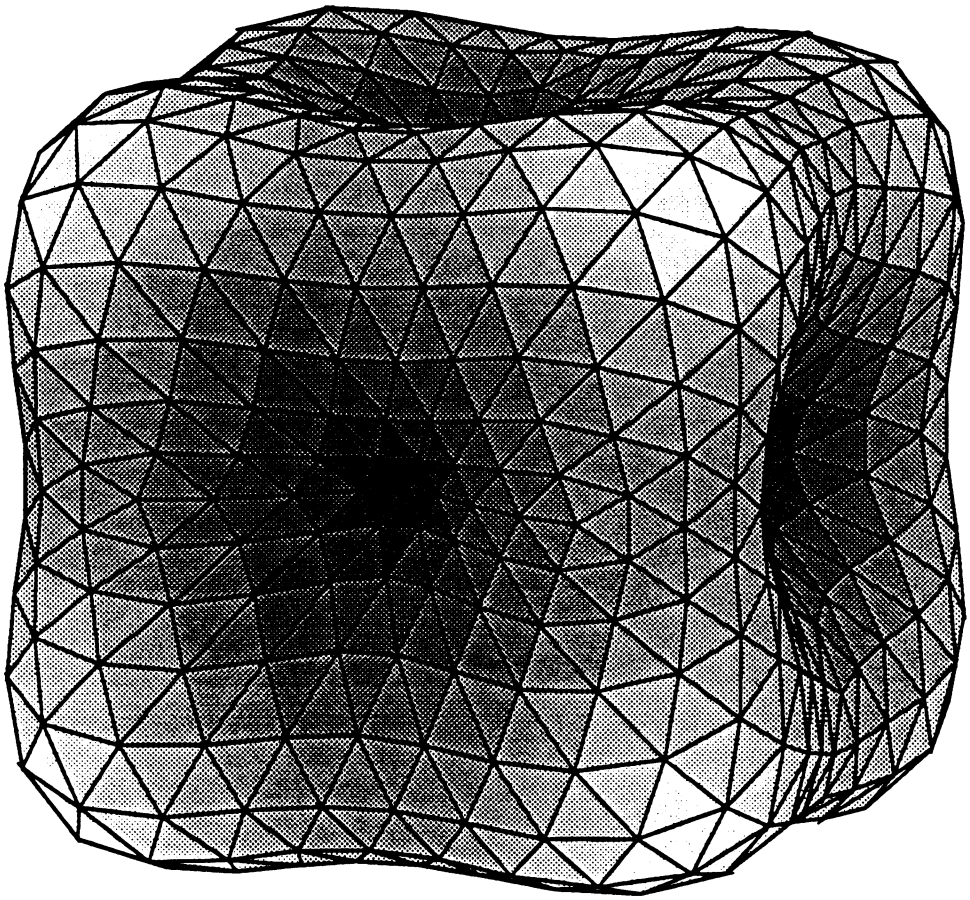


Figure 5.13. Radius proportional to $\left\langle\!\left\langle \phi^2_{\gamma(2)}(\hat{e}) \right\rangle\!\right\rangle^{\frac{1}{2}}$.

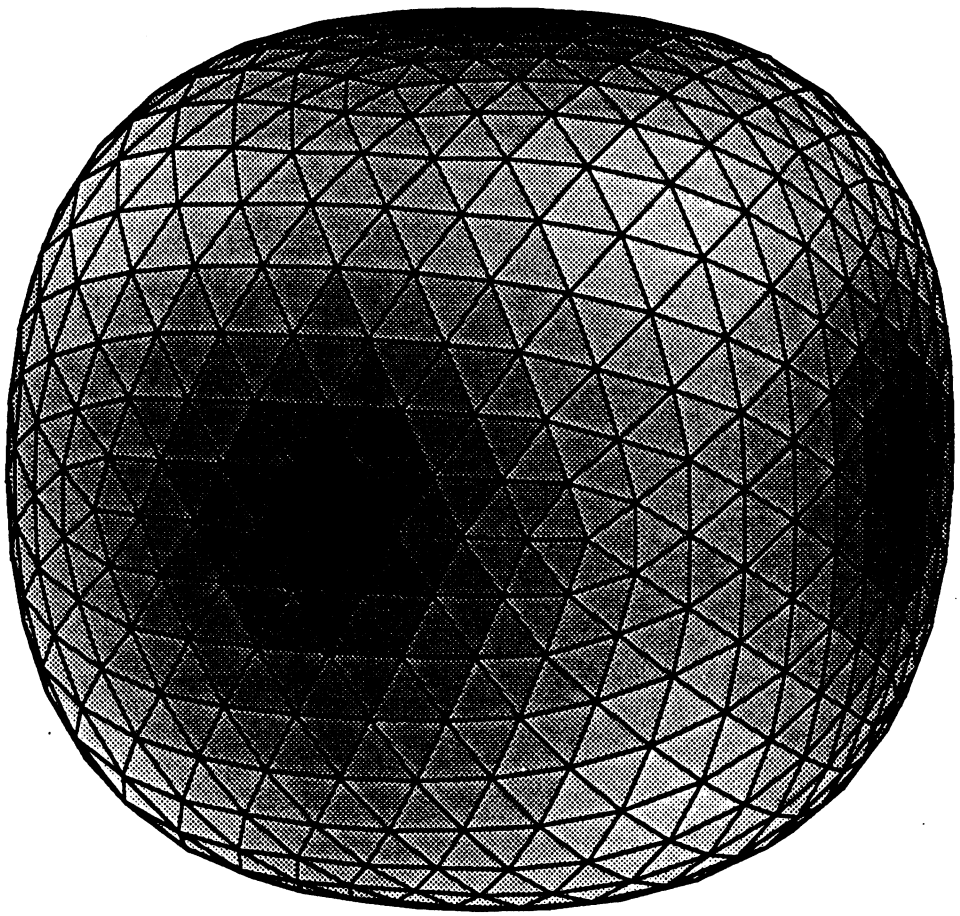


Figure 5.14. Radius proportional to $\left\langle\!\left\langle \left|\vec{a}_{\gamma(2)}(\hat{e})\right|^2 \right\rangle\!\right\rangle^{\frac{1}{2}}$.

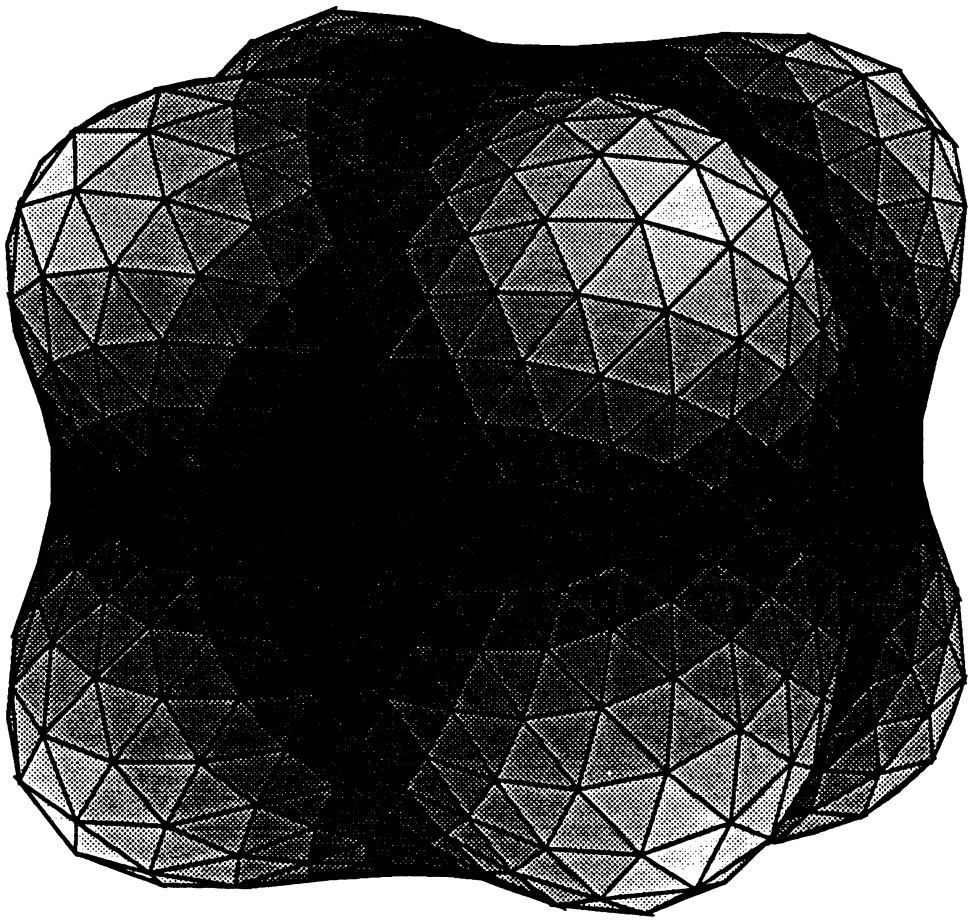


Figure 5.15. Radius proportional to $\left\langle\!\left\langle \phi^2_{\gamma(3)}(\hat{e}) \right\rangle\!\right\rangle^{\frac{1}{2}}$.

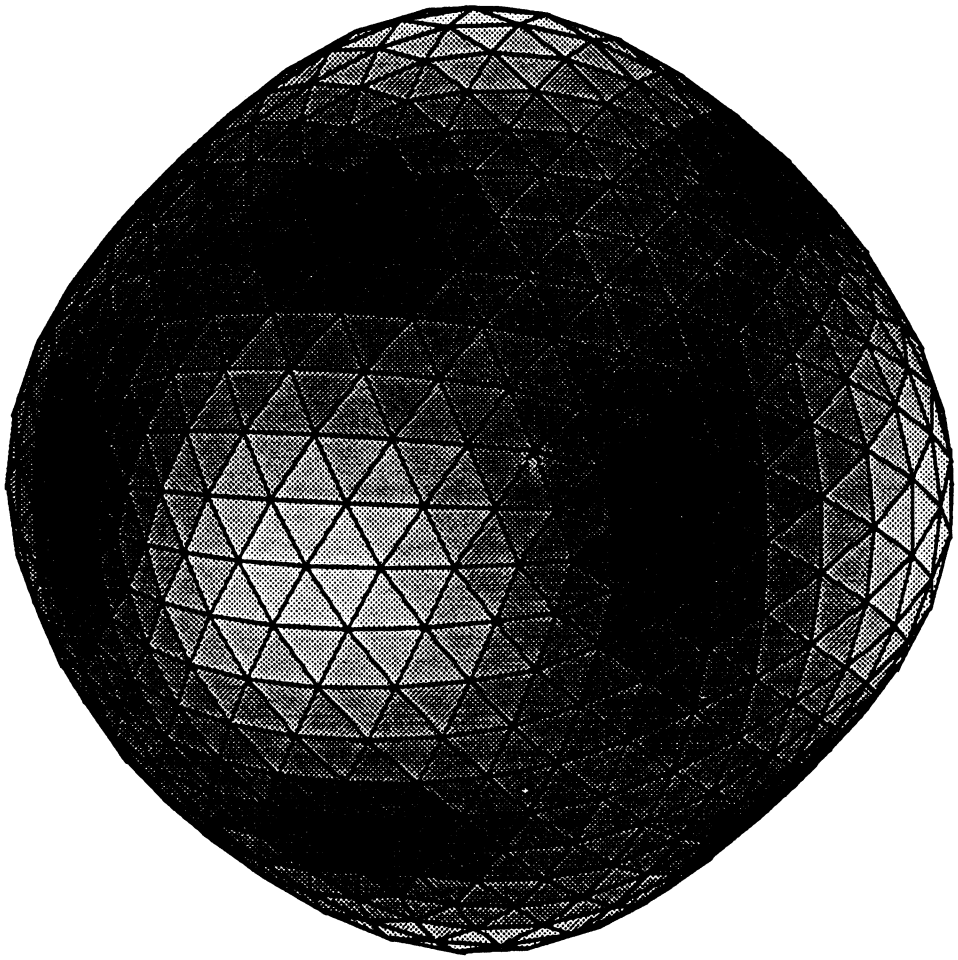


Figure 5.16. Radius proportional to $\left\langle\!\left\langle \left|\vec{a}_{\gamma(3)}(\hat{e})\right|^2 \right\rangle\!\right\rangle^{\frac{1}{2}}$.

6. Opening Criteria.

We saw in Chapter 4, that the performance of the BH algorithm depends critically on the `OpeningCriterion` used when traversing the tree. The choice of `OpeningCriterion` is constrained by the requirement that the approximations be sufficiently accurate for whatever scientific purposes are intended. We cannot make the `OpeningCriterion` too strict, however, or we will derive no benefit from the BH tree at all. Thus, the `OpeningCriterion` must be sufficiently conservative that whenever the multipole approximation is used, sufficient accuracy results, but it must also allow enough multipole approximations to achieve acceptable performance. The basic routine responsible for tree traversal appears in Code 6.1.

```
ComputeField(body, cell)
    if( cell is terminal )
        for( each tbody in cell )
            BodyBodyInteraction(body, tbody)
        endfor
    else if( OpeningCriterion(cell, body) )
        for( each child of cell )
            ComputeField(body, child)
        endfor
    else
        BodyCellInteraction(body, cell)
    endif
endfunc
```

Code 6.1. Function `ComputeFields` to compute an approximation to ϕ and $\vec{a}$ for a body, by recursive descent of a BH tree.

6.1. General form of the opening criterion.

In Chapter 5 we computed numerous error estimates for the multipole

approximation. A common feature of all the error estimates (both maximum and statistical, and both potential and acceleration) is that they are increasing functions of the ratio

$$\left(\frac{b}{d}\right), \tag{6.1}$$

where b measures the size of $\mathcal{V}_\gamma$, and d measures the distance from $\vec{r}_\gamma$ to the body. In general, the approximation is completely unreliable for $\frac{b}{d} > 1$, and the error goes to zero as

$$\lim_{\frac{b}{d}\to 0} |\text{error}| \propto \left(\frac{b}{d}\right)^{p+1}, \tag{6.2}$$

where p is the highest order multipole considered.

The functional form of the error term suggests an `OpeningCriterion` of the form shown in Code 6.2. This general form for the `OpeningCriterion` relies on an additional function, $D(cell, body)$, and an adjustable parameter, θ. $D(cell, body)$ may be loosely interpreted as the distance from the body to $\vec{r}_\gamma$. It is related to the quantity, $d = |\vec{x} - \vec{r}_\gamma|$, but it need not be identical to d. The optimal form for d would have contours that follow the constant-error contours of Figures 5.8 through 5.10. On the other hand, there is little benefit in following the contours of Figures 5.8 through 5.10 precisely. The function, $D(cell, body)$, should be as simple as possible, while still maintaining contours shaped roughly like those in Figures 5.8 through 5.10.

```
OpeningCriterion(cell, body)
    return b/θ > D(cell, body)
endfunc
```

Code 6.2. General form of the `OpeningCriterion` function.

The adjustable parameter, θ, in Code 6.2 controls the accuracy of the simulation. Small values of θ imply high accuracy, and less willingness to use the multipole approximations, while large values of θ imply lower accuracy, with a greater reliance on multipole approximations.

We shall consider six distinct `OpeningCriteria` in this chapter. We may select one of two options for $D(cell, body)$:

1. The distance from the body to $\vec{r}_\gamma$.
2. The distance from the body to the edge of $\mathcal{V}_\gamma$.

In addition, we may select one of three options for the metric used to compute the distance:

1. The usual Cartesian metric, i.e., $(\delta x^2 + \delta y^2 + \delta z^2)^{\frac{1}{2}}$.
2. The Manhattan or L_1 metric, i.e., $|\delta x| + |\delta y| + |\delta z|$.
3. The L_∞ metric, i.e., $\max(|\delta x|, |\delta y|, |\delta z|)$.

We shall use subscripts on θ to indicate which `OpeningCriterion` is under consideration. The subscript BH indicates Barnes' original `OpeningCriterion` which uses distance to $\vec{r}_\gamma$ and a Cartesian metric, i.e.,

$$\theta_{BH} = \theta_{\vec{r}_\gamma, Cart}. \tag{6.3}$$

6.2. Opening criteria in the literature.

The original Barnes-Hut `OpeningCriterion` makes the following choice for the `OpeningCriterion`,

$$D(cell, body) = \text{distance from } \vec{r}_\gamma \text{ to the body.} \tag{6.4}$$

This particular choice of strategies has been studied in some detail.[18, 17] Nevertheless, there is a serious problem with this `OpeningCriterion` which has gone unnoticed. As we shall see, other strategies offer superior accuracy for the same performance.

Some work has been done on alternative `OpeningCriteria`. Makino has suggested an alternative formulation of the `OpeningCriterion`, based on statistical arguments. [20] He notes the factor of N_{eff}^{-1}, as well as the factors, $\left|\phi_{\gamma(0)}\right|^2$ or $\left|\vec{a}_{\gamma(0)}\right|^2$, which are proportional to N_γ^2 in Eqns. 5.85 and 5.86. Thus, he minimizes the "error" based on the assumption that the error is proportional to $N_\gamma^{1/2}$. Makino points out, however, that there may be difficulties with large cells and

warns against a naive application of the method. The reason for the failure of convergence is contained in Eqns. 5.76 and 5.78, which tell us that when the underlying distribution of matter is not perfectly uniform, we no longer have the benefit of the extra factor of N^{-1}. Assuming that the distribution of matter is exactly uniform leads to an overly optimistic error estimate which explains Makino's observation that "convergence of the multipole expansion is not guaranteed for large cells, since the criterion leads to a too-large opening criterion for large cells."

Barnes has described yet another `OpeningCriterion`,[22] which is motivated primarily by the desire to eliminate conditionals from the tree traversal routine in order to facilitate vectorization. The resulting `OpeningCriterion` is applied to groups of bodies, rather than one at a time, as in Code 6.2, and all bodies in a group either interact with a cell, or not, according to the result of a single execution of `OpeningCriterion`. The `OpeningCriterion` is the usual BH one, except that the distance is calculated from the boundary of the group of bodies to the $\vec{r}_\gamma$ of the cell. Groups of bodies are defined, effectively, as terminal nodes of a BH tree with $m = n_{crit}$, where n_{crit} is comparable to the vector length of the target architecture. Barnes compares the alternative `OpeningCriterion` with the original one and concludes that the choice can be made on the basis of computational performance, i.e., from a strictly numerical point of view there is little evidence to recommend one over the other. Unfortunately, the modified `OpeningCriterion` remains susceptible to the systematic source of error described in the next section.

6.3. The detonating galaxy pathology.

Before we investigate alternatives to `OpeningCriteria`, we consider a serious difficulty with the original formulation. We found, quite by accident, that the BH `OpeningCriterion` with $\theta \geq 0.58$ can result in very significant systematic errors that can completely destroy a simulation.

To understand the difficulty with the original `OpeningCriterion`, consider the situation illustrated in Figure 6.1. A small secondary galaxy is moving toward

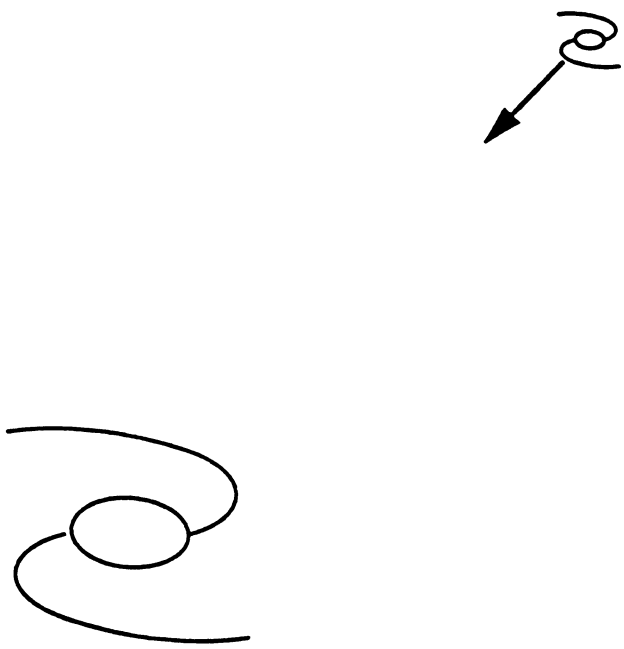


Figure 6.1. A two-galaxy system that may exhibit the detonating galaxy pathology.

a massive "primary" system which is stationary at the lower left corner of one of the hierarchical cells. Unless θ_{BH} is very small, the internal dynamics of the small system will be severely disrupted by systematic errors introduced into the force calculation.

Consider the situation when the evolution reaches the configuration shown in Figure 6.2. Much of the secondary system has passed into the cell P. The center-of-mass of cell P, however, has not moved appreciably, and remains near the lower left corner. Thus, when the opening criterion is applied for the body marked with an X, one finds that the distance to the center-of-mass of cell P is about $\sqrt{2}$. For any value of $\theta_{BH} > \frac{1}{\sqrt{2}}$, cell P will not be opened. All the bodies in cell P, including those in the secondary galaxy, e.g., the body Y, will be represented by a multipole expansion around $\vec{r}_\gamma$, near the lower left corner of P. The result is that the gravitational attraction of the part of the core of the secondary galaxy that has passed into P will be greatly reduced on body X, and the secondary galaxy, if it was gravitationally bound on its own, will lose a considerable amount of its self-gravity. In practice, the secondary galaxy appears to detonate as it crosses the cell boundary. For this reason, we shall refer to this problem as the "detonating galaxy pathology."

Note that the point X is outside of cell P, so this problem is distinct from the "self-interaction" problem discussed, and solved, by Hernquist [18] and Barnes,[17] which may be treated by modifying the opening criterion so that a cell which contains the body whose potential is being calculated is always opened. In effect, Hernquist and Barnes have corrected the problem for bodies like Y, which are inside cell P, but not for bodies like X.

It is noteworthy that this pathology actually arose in a "real" simulation before we were aware of its existence, so it may not be "solved" by asserting that it will not occur in practice. The simulation in question was of a physical system much like that shown in Figure 6.1. It consisted of two "galaxies" on a headon collision orbit inclined at 45° to the Cartesian axes. Interestingly, Barnes has used

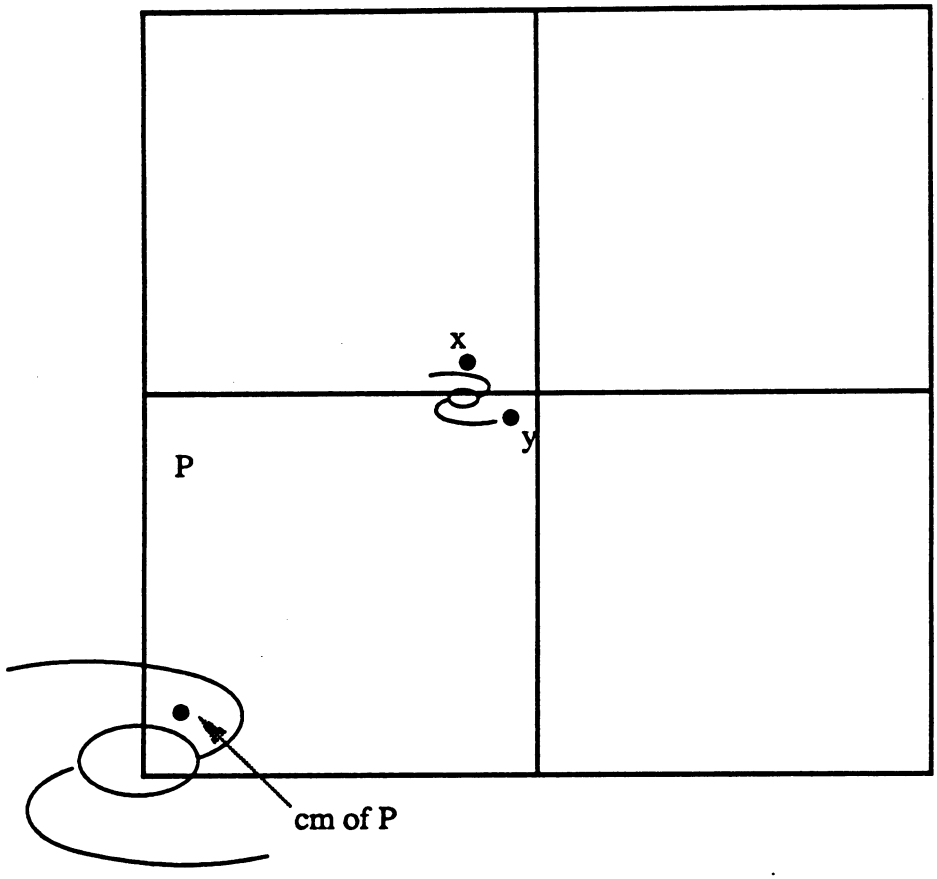


Figure 6.2. Pathological situation for BH `OpeningCriterion`.

a similar system to verify the correctness of his approach in an extensive series of tests.[17, 22] The galaxies in Barnes' test case were of equal mass and on a headon collision course parallel to one of the Cartesian axes. In this configuration, the pathology does not arise, and Barnes was able to "verify" the accuracy of the algorithm. Makino[20] has similarly verified the accuracy of his alternative `OpeningCriterion` on an even simpler system–a single Plummer sphere of bodies. Again, the pathology did not arise.

6.4. The cause of the problem.

The fundamental cause of the detonating galaxy pathology is easily understood. It stems from the proximity of the point X, to the cell P, which is treated by a multipole expansion. Clearly, as the point X becomes arbitrarily close to the boundary of cell P, the multipole expansion of P fails to account for the very-near-neighbor interactions that are possible between X and bodies inside P.

It is worth noting that this behavior is "predicted" by Eqns. 5.57 and 5.61, which give strict upper bounds on the remaining error after the n^{th} multipole term. We can write Eqn. 5.57 as

$$\left|\Phi_{\gamma(n)}(r)\right| \leq \frac{M_{\gamma(0)}}{d-b}\left(\frac{b}{d}\right)^{n+1}, \tag{6.5}$$

and Eqn. 5.61 as

$$\left|\vec{A}_{\gamma(n)}\right| \leq \frac{M_{\gamma(0)}}{(d-b)^2}\left(\frac{b}{d}\right)^{n+1}\left(n+2-\frac{b}{d}(n+1)\right). \tag{6.6}$$

The factor of $(d-b)$ appearing in the denominator of both of these equations has a simple geometric interpretation, shown in Figure 6.3. As the test point X approaches the boundary of $\mathcal{V}_\gamma$, the maximum possible error increases as an inverse power of the distance from the point to the boundary, i.e., $(d-b)^{-1}$. In fact, the factors of $(d-b)^{-1}$ in the potential and $(d-b)^{-2}$ in the acceleration,

correspond to the magnitude of the error that would result from a body placed inside the cell, near the boundary, a distance of $(d - b)$ from the test point.

For values of the ratio, $\frac{b}{d}$, which are not vanishingly small, we must include the term proportional to $(d-b)^{-1}$ in our estimate of the error, if we hope to obtain a predictable level of accuracy.

In order to constrain our `OpeningCriteria`, we compute the minimum possible distance between the boundary of a cell and a body which interacts with the multipole approximation of that cell. Unless this distance is greater than zero, there is no guarantee that the multipole expansion will produce accurate results, and the detonating galaxy pathology may arise. The worst case is illustrated in Figure 6.3, which shows a cell with center-of-mass near one of the corners and a sphere (circle in the two-dimensional figure) of radius $b\theta^{-1}$ outside of which the multipole approximation may be used. From Figure 6.3, we see that for any value of $\theta > 1/\sqrt{3} \approx 0.58$, there may be no distance at all between a test point and the boundary of a multipole approximated cell. This is the basis of the claim that the original BH `OpeningCriterion` is unreliable for $\theta > 1/\sqrt{3}$.

6.5. Comparing opening criteria.

Before we explore the relative merits of various `OpeningCriteria` mentioned in Section 6.1, we need a method for comparing one with another. We found in Chapter 4 that the total number of interactions computed in an N-body simulation is proportional to the volume of the region, $\mathcal{V}_{mid}$, which satisfies the `OpeningCriterion` of the parent of a cell, but fails the `OpeningCriterion` of the cell itself. The quantity, F_{OC}, was defined to be the ratio of $\mathcal{V}_{mid}$ to $\mathcal{V}_{\gamma}$, and is the only characteristic of the `OpeningCriterion` which influences the total number of interactions computed.

We know that, in some cases, the BH `OpeningCriterion` implies largely incorrect approximations, and that, perhaps, some other `OpeningCriteria` will not share the same difficulty. Unfortunately, we lack a quantitative measure of this phenomenon. In lieu of any statement like: "The alternative `OpeningCriterion`

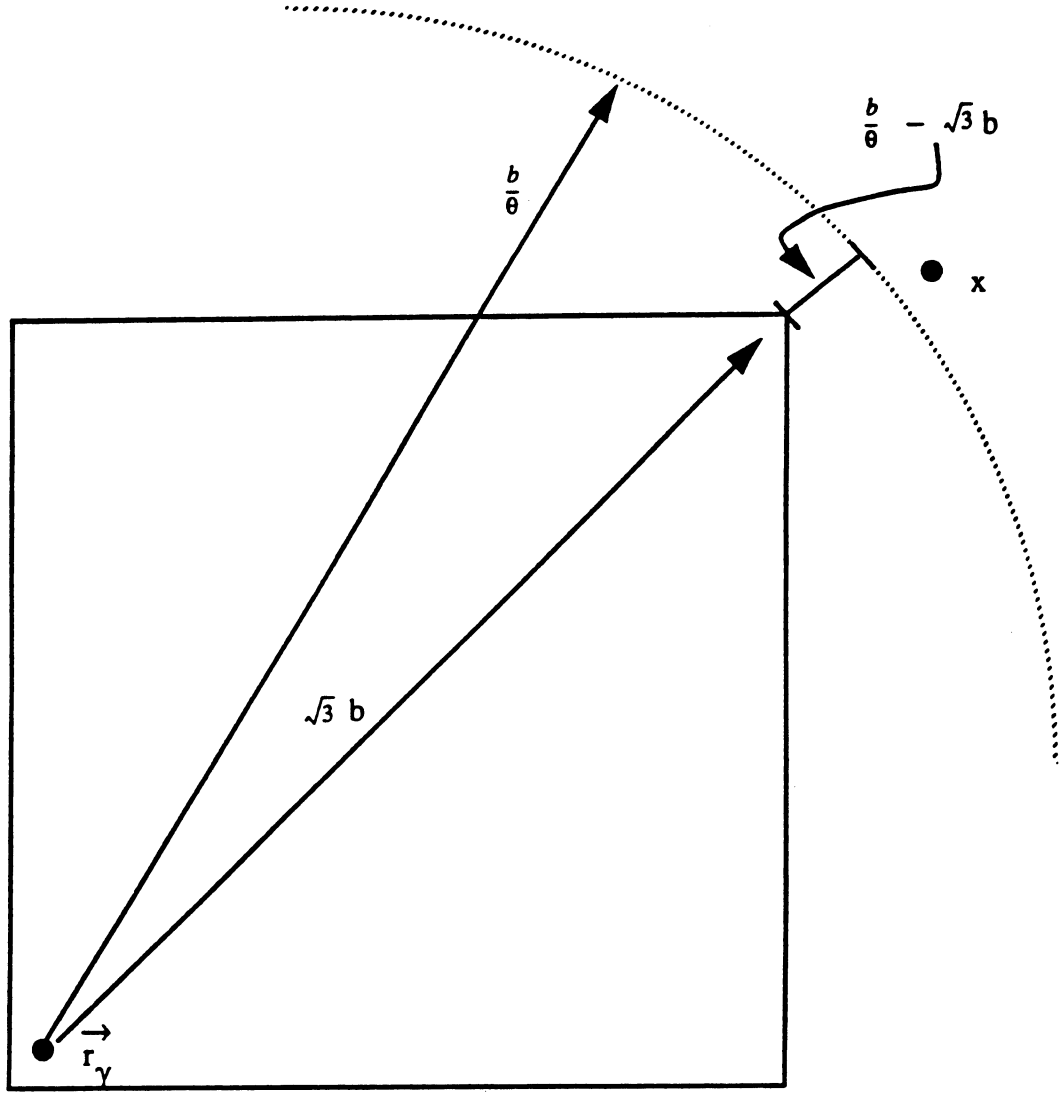


Figure 6.3. Worst case situation arising from BH `OpeningCriterion`. The distance from the test point, x to the corner of the cell is only $b/\theta - \sqrt{3}b$.

results in an average 20% improvement in the error for a fixed number of computed interactions," we adopt the simple hypothesis that the approximations implied by each `OpeningCriteria` are equally good. Then, we can find a value for θ_{alt} which leads to the same number of interactions as any given value of θ_{BH}. We assume that a simulation carried out with an alternative `OpeningCriterion` and this equivalent value of θ will be at least as accurate as the simulation using the original BH `OpeningCriterion`. The detonating galaxy pathology inherent in the BH opening criterion suggests that it is not a particularly good choice for a benchmark. Nevertheless, there is considerable evidence that it performs well, at least when the detonating galaxy pathology does not arise. Thus, despite its problems, we use the BH opening criterion as a benchmark with "known" numerical properties,[17] and we compare the alternative `OpeningCriteria` to it.

With these caveats in mind, we turn to computation of the values of F_{OC} for the alternative `OpeningCriteria` under consideration. The region, $\mathcal{V}_{mid}$, corresponding to the original BH `OpeningCriterion` is shown in Figure 6.4. The value of F_{OC} is clearly

$$F_{OC} = \frac{4\pi}{3}\left(\frac{2}{\theta_{BH}}\right)^3 - \frac{4\pi}{3}\left(\frac{1}{\theta_{BH}}\right)^3 = \frac{28\pi}{3\theta_{BH}^3}. \tag{6.7}$$

In Figure 6.5, we show the $\mathcal{V}_{mid}$ arising from a Cartesian metric and an *edge* distance function. The value of F_{OC} is

$$F_{OC} = 7\left(1 + \frac{6}{\theta_{edge,Cart}} + \frac{3\pi}{\theta_{edge,Cart}^2} + \frac{4\pi}{3\theta_{edge,Cart}^3}\right). \tag{6.8}$$

It is a simple exercise in solid geometry to calculate the value of F_{OC} for the remaining `OpeningCriteria`:

$$F_{OC} = \frac{28}{3\theta_{\vec{r}_\gamma,Man}^3}, \tag{6.9}$$

$$F_{OC} = \frac{56}{\theta_{\vec{r}_\gamma,L_\infty}^3}, \tag{6.10}$$

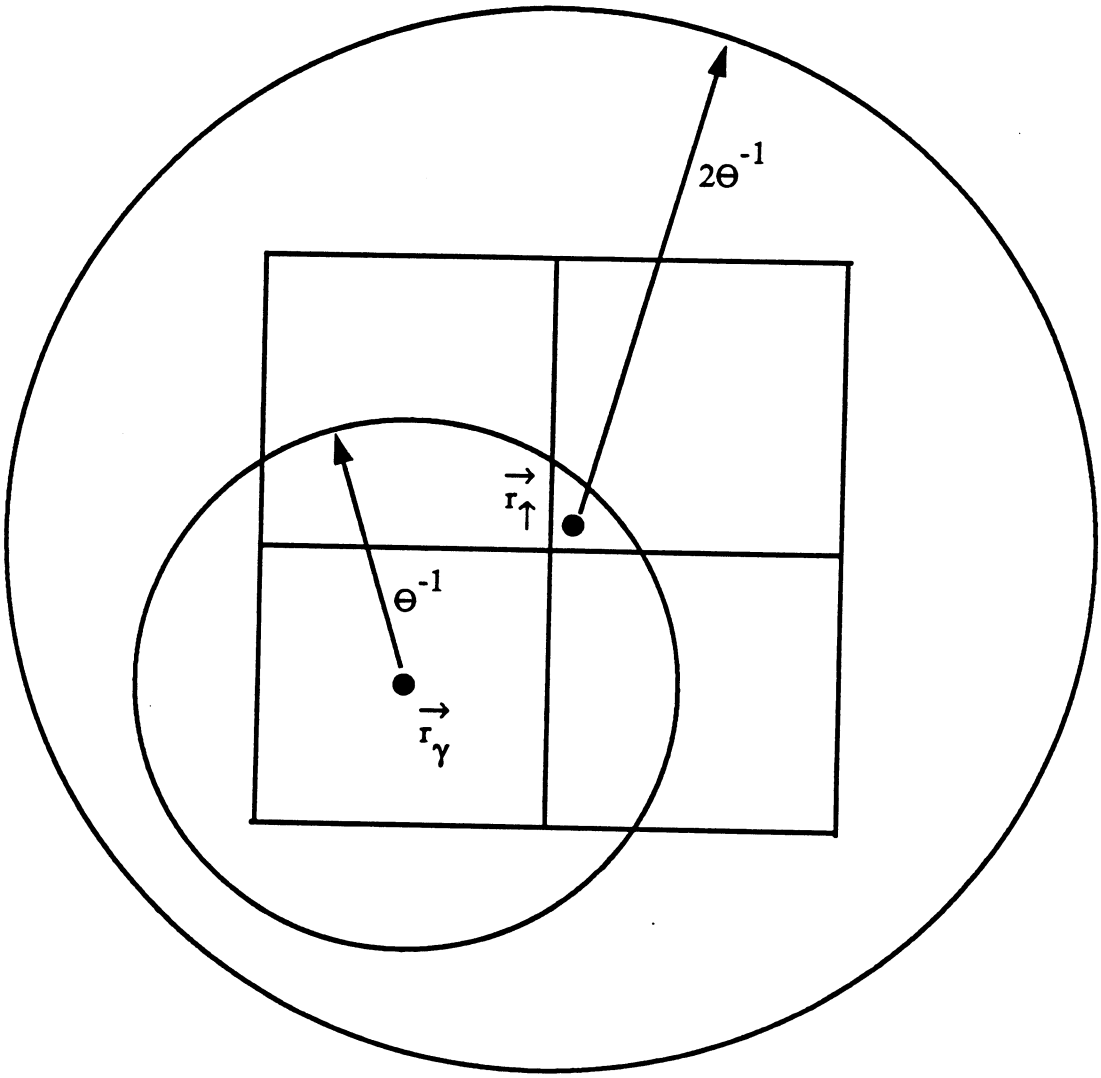


Figure 6.4. Geometry of $\mathcal{V}_{mid}$ with the original BH `OpeningCriterion`.

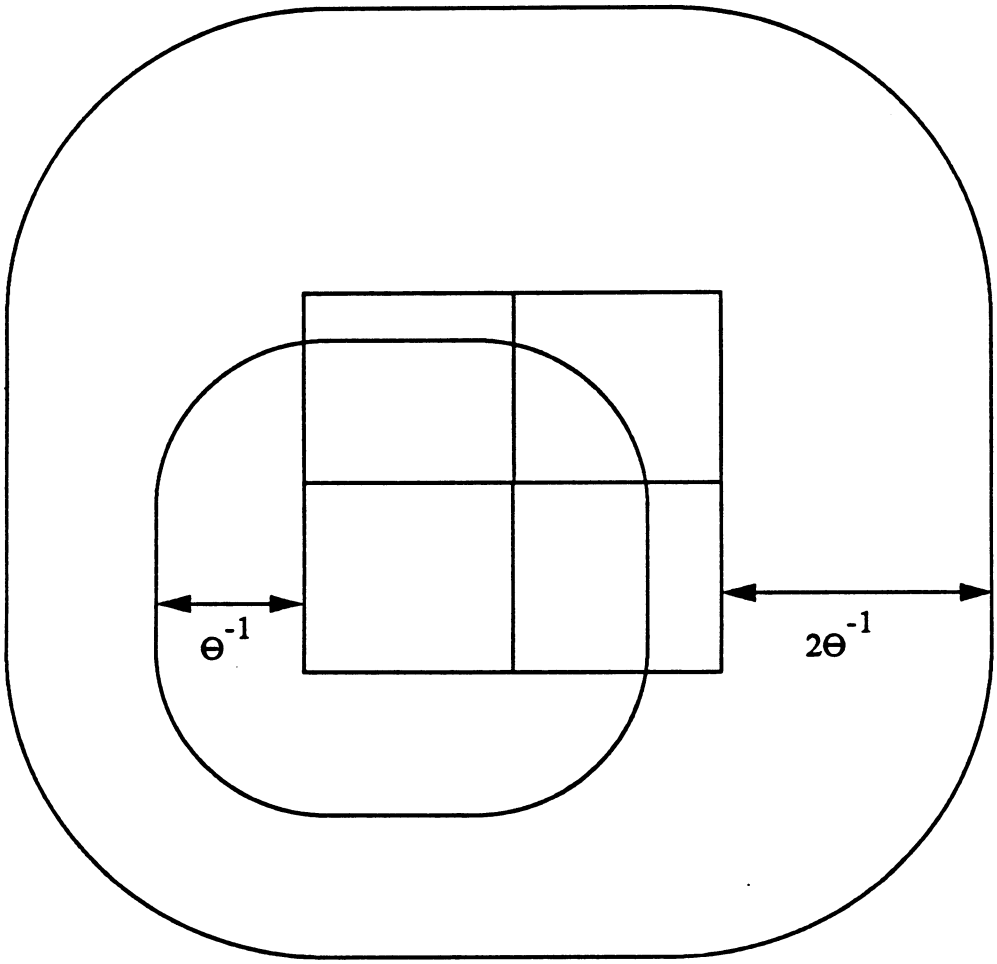


Figure 6.5. Geometry of $\mathcal{V}_{mid}$ with the $(edge, Cart)$ `OpeningCriterion`.

$$F_{OC} = 7\left(1 + \frac{6}{\theta_{edge,Man}} + \frac{6}{\theta^2_{edge,Man}} + \frac{4}{3\theta^3_{edge,Man}}\right), \tag{6.11}$$

$$F_{OC} = 7\left(1 + \frac{2}{\theta_{edge,L_\infty}}\right)^3. \tag{6.12}$$

The functions in Eqns. 6.7 through 6.12 are plotted in Figure 6.6.

In addition, we can set the values of F_{OC} in Eqns. 6.7 through 6.12, equal to one another, and obtain equivalent values of θ, by which one can compare different `OpeningCriteria`. Since we have adopted the BH `OpeningCriterion` as our benchmark, we relate the new `OpeningCriteria` to it. The results for three cases are easy to state explicitly as

$$\theta_{\bar{r}_\gamma,Man} \approx \pi^{-1/3}\theta_{BH}, \tag{6.13}$$

$$\theta_{\bar{r}_\gamma,L_\infty} \approx \left(\frac{6}{\pi}\right)^{1/3}\theta_{BH}, \tag{6.14}$$

$$\theta_{edge,L_\infty} \approx \frac{2\theta_B H}{\left(\frac{4}{3}\right)^{1/3} - \theta_{BH}}. \tag{6.15}$$

For the other two, we refer to Figure 6.7, which graphically represents equivalent values of θ.

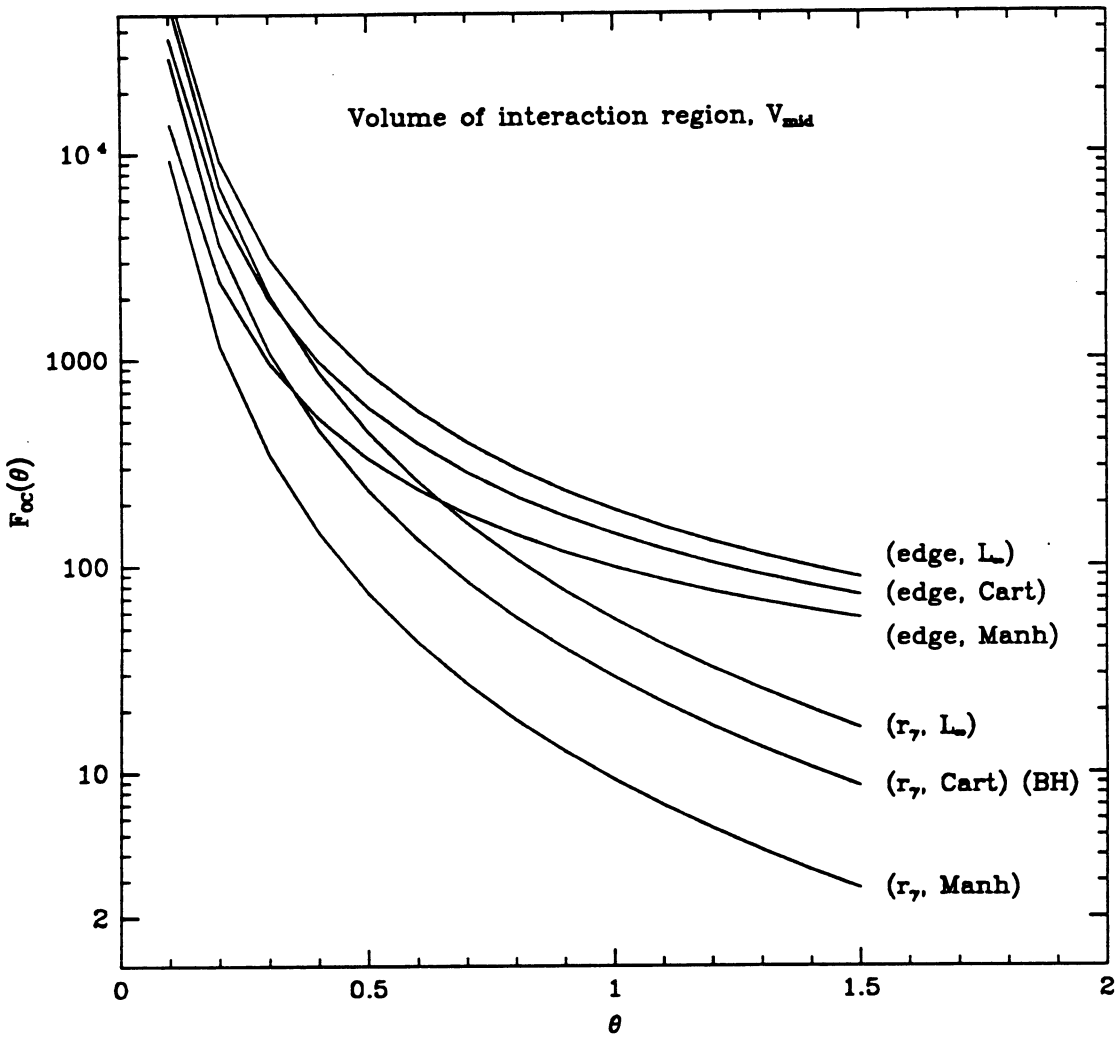


Figure 6.6. The volume of $\mathcal{V}_{mid}$ plotted against θ for the six `OpeningCriteria` discussed in this chapter.

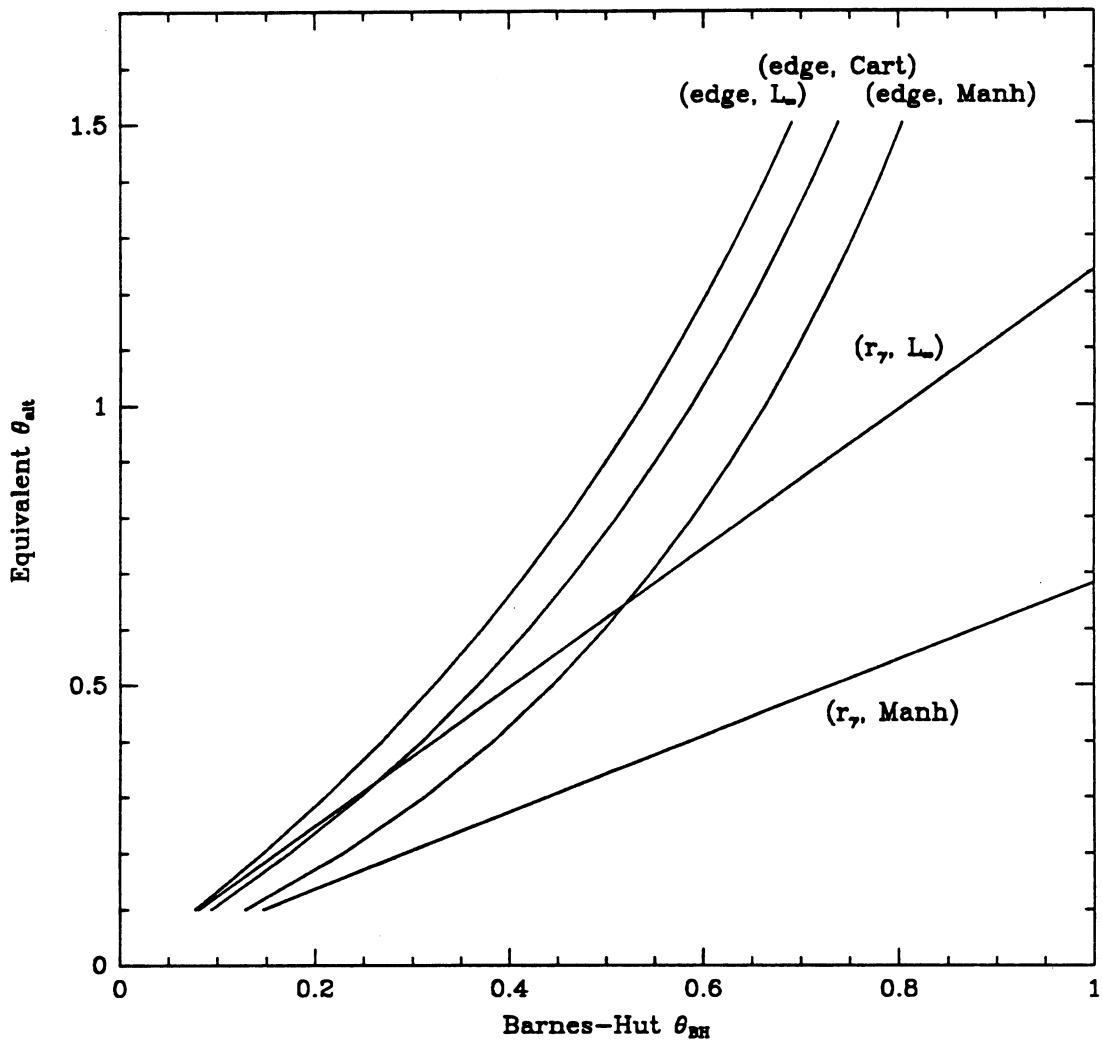


Figure 6.7. Equivalent values of θ for different `OpeningCriteria`. θ_{BH} is shown on the horizontal axis, and equivalent values of θ for the other five `OpeningCriteria` are plotted.

6.6. Avoiding the detonating galaxy pathology.

It is clear that we can eliminate the detonating galaxy pathology by ensuring that there is a finite separation between a body and a cell which fails the `OpeningCriterion` for that body. When the `OpeningCriterion` uses a $D(cell, body)$ which measures the distance from a body to the edge of a cell, then the separation is, by construction, always greater than zero. Thus, any of the edge-type `OpeningCriteria` are not susceptible to the detonating galaxy pathology. If distances are measured from the body to $\vec{r}_\gamma$, then only certain values of θ guarantee that bodies are well separated from unopened cells. The values are

$$\theta_{\vec{r}_\gamma, Cart} < \frac{1}{\sqrt{3}}, \tag{6.16}$$

$$\theta_{\vec{r}_\gamma, Man} < \frac{1}{3}, \tag{6.17}$$

$$\theta_{\vec{r}_\gamma, L_\infty} < 1. \tag{6.18}$$

For each of these three, we can compute the value of F_{OC}:

$$F_{OC}(\vec{r}_\gamma, Cart) > 28\pi\sqrt{3}, \tag{6.19}$$

$$F_{OC}(\vec{r}_\gamma, Man) > 252, \tag{6.20}$$

$$F_{OC}(\vec{r}_\gamma, L_\infty) > 56. \tag{6.21}$$

These values are useful because they give us an indication of the "maximum safe operating speed" for the given `OpeningCriterion`. For example, it is almost certainly a bad idea to use a $(\vec{r}_\gamma, Man)$ `OpeningCriterion` with $F_{OC} < 252$. As this value is even higher than the corresponding value for the BH `OpeningCriterion`, there is little to recommend the Manhattan metric. On the other hand, with $\theta_{\vec{r}_\gamma, L_\infty} = 1$, F_{OC} is only 56. A simulation with such an `OpeningCriterion` will execute at approximately the same overall speed as a BH `OpeningCriterion` with $\theta_{BH} \approx 0.8$. Values of θ_{BH} that large exhibit the detonating galaxy pathology, so the $(\vec{r}_\gamma, L_\infty)$ metric is preferred in this case.

6.7. Optimizing the opening criterion computation.

The opening criterion is evaluated many times during the course of a simulation. In fact, a careful analysis of the algorithm shows that it is evaluated approximately as many times as the pair-wise force computation. It is critical to choose an opening criterion that can be evaluated very rapidly, as it can have an impact on the overall performance of the algorithm.

Our first observation in this regard refers to the Cartesian distance between two points in space. Typically, evaluation of the square-root is a very time consuming operation, and an obvious "simplification" of the opening criteria is to eliminate the square-root, and compare the squared distance with the square of the diameter of the cell, i.e., modify Code 6.2 to be:

```
OpeningCriterion(cell, body)
    return (b/θ)^2 > Dsquared(cell, body)
endfunc
```

Code 6.3. Modified form of the `OpeningCriterion` function for Cartesian metrics.

A second observation is that when evaluating an `OpeningCriterion` one does not strictly need to evaluate $D(cell, body)$ completely. In many cases, it is possible to determine that the `OpeningCriterion` has failed before all three Cartesian directions have been accumulated to obtain the "distance." One begins with a "cutoff" distance, b/θ, and decreases the value appropriately for δx, δy and δz. If the cutoff ever falls below zero, then FALSE can be returned immediately.

In Codes 6.4 through 6.6, we illustrate these ideas for three of the `OpeningCriteria`. In order to be as explicit as possible, we present C language implementations. These C programs are organized to make translation into a concise set of machine instructions as simple as possible. The predicates within the `if` statements are comparisons of the result of an arithmetic operation with zero. This type of operation corresponds to commonly available "conditional branch" instructions.

```
cutoff = b*thetainv;
cutoff *= cutoff;
d = x-xgamma;
if( (cutoff -= d*d) < 0 )   return FALSE;
d = y-ygamma;
if( (cutoff -= d*d) < 0 )   return FALSE;
d = z-zgamma;
if( (cutoff -= d*d) < 0 )   return FALSE
return TRUE
```

Code 6.4. C language code fragment to compute an `OpeningCriterion` based on a Cartesian distance to $\vec{r}_\gamma$.

```
cutoff = b*thetainv;
if( (d=x-xgamhi) > 0 ){
    if((cutoff -= d) < 0)   return FALSE;
}else if( (d -= b) < 0){
    if((cutoff += d) < 0)   return FALSE;
}
if( (d=y-ygamhi) > 0 ){
    if((cutoff -= d) < 0)   return FALSE;
}else if( (d -= b) < 0){
    if((cutoff += d) < 0)   return FALSE;
}
if( (d=z-zgamhi) > 0 ){
    if((cutoff -= d) < 0)   return FALSE;
}else if( (d -= b) < 0){
    if((cutoff += d) < 0)   return FALSE;
}
return TRUE;
```

Code 6.5. C language code fragment to compute an `OpeningCriterion` based on a Manhattan metric to the edge of a cell, of size `b` and with maximum Cartesian coordinates `xgamhi`, `ygamhi` and `zgamhi`.

The L_∞ and Manhattan metrics contain fewer multiplications, but more conditional branches than the Cartesian metric. All three methods require approximately the same number of additions and subtractions. It is likely that the L_∞

```
cutoff = b*thetainv;
if( (d=x-xgamhi) > 0 ){
    if((cutoff - d) < 0)    return FALSE;
}else if( (d -= b) < 0){
    if((cutoff + d) < 0)    return FALSE;
}
if( (d=y-ygamhi) > 0 ){
    if((cutoff - d) < 0)    return FALSE;
}else if( (d -= b) < 0){
    if((cutoff + d) < 0)    return FALSE;
}
if( (d=z-zgamhi) > 0 ){
    if((cutoff - d) < 0)    return FALSE;
}else if( (d -= b) < 0){
    if((cutoff + d) < 0)    return FALSE;
}
return TRUE;
```

Code 6.6. C language code fragment to compute an `OpeningCriterion` based on an L_∞ metric to the edge of a cell, of size `b` and with maximum Cartesian coordinates `xgamhi`, `ygamhi` and `zgamhi`.

and Manhattan metrics will be somewhat faster, so they are recommended for that reason.

However, the Manhattan metric when combined with the $\vec{r}_\gamma$ distance function, is only reliable for $\theta_{\vec{r}_\gamma, Manh} < 1/3$. Equation 6.20 tells us that the value of F_{OC} at the maximum acceptable value of θ is even larger than the BH case. Thus, we reject this particular case.

The contours of constant error, shown in Figures 5.8 through 5.10 are distinctly "boxy" in appearance. This suggests that the contours of $\mathcal{V}_{mid}$ should also be boxy. The three metrics we consider, Manhattan, Cartesian and L_∞ have isosurfaces that are octohedra, spheres and cubes, respectively. The closest approximation to the shapes in Figures 5.8 through 5.10 is obtained with the cubic isosurfaces derived from the L_∞ norm. Thus, the most promising choice of

`OpeningCriteria` uses the L_∞ norm and either the *edge* or $\vec{r}_\gamma$ distance function. As we shall see in Chapter 7, the *edge* distance is advantageous for parallelization because the result of the `OpeningCriterion` is determined purely by the geometry of the cell, and is independent of its contents, which influence $\vec{r}_\gamma$.

7. Parallel Implementation of the BH Algorithm.

We now turn to issues that arise in the implementation of the BH algorithm on a parallel processor. In particular, we are concerned with multiple instruction stream, multiple data stream, message passing, tightly coupled, asynchronous distributed memory, large grain size, homogeneous, highly parallel processors. Taking each of these adjectives in turn, we have:

Multiple instruction stream

Each processor executes instructions from its own instruction stream, regardless of the instructions executed by its neighbors.

Multiple data stream

Each processor addresses its own data space, separate from that addressed by the other processors in the system.

Message passing

Data is transmitted between processors in discrete messages, in contrast to the use of shared memory.

Tightly coupled

The message passing system is an integral part of the hardware and software environment, allowing the user to treat the system as a unified whole, rather than a collection of autonomous servers and clients.

Asynchronous

The processors have independent clocks, and synchronize only during transmission of messages.

Distributed memory

The memory is distributed amongst the processors, and is not commonly accessible. Each processor's data space is uniquely its own.

Large grain size

The individual processors are "large," in the sense of being capable of significant computations. A precise definition of large is problematical, but a good operational definition is that a "large" processor is as powerful as the current state of the art in scientific workstations.

Highly parallel

There are many, up to a few thousand, processors in the system. This is in contrast to sequential processors or systems which can be configured with only a handful of processors.

Homogeneous

The individual processors have identical cpus and equal amounts of memory. This allows the same program to be loaded into each processor. A program may determine the processor on which it is running, and other aspects of the parallel environment by making operating system calls.

For some of the above categories, the fact that we are targeting a particular instance does not rule out using the same algorithm on the alternatives. Certainly, if the algorithm performs well on a distributed memory machine, then it will perform at least as well on a shared memory machine. One can always elect not to share the memory. Similarly, if a machine has only eight processors, it does not qualify as "highly parallel," but there is no reason that the algorithm would not perform well.

Highly parallel, distributed memory computers are capable of extremely high overall performance. It is likely that in the near future, distributed memory parallel machines will outperform the fastest vector supercomputers. One can contemplate extremely large N-body simulations on such machines, which would be prohibitively expensive on more common vector supercomputers. Thus, this chapter is devoted to understanding issues related to parallelization of the BH algorithm.

The parallelization proceeds in three steps. First, the bodies are divided amongst the processors. Second, a representation of the BH tree is constructed

in each processor. Finally, the bodies in each processor traverse the tree in that processor exactly as in the sequential algorithm.

7.1. Domain decomposition, orthogonal recursive bisection.

The first problem is usually referred to as "decomposition."[6] We seek a way to "decompose" or map the set of bodies into the set of processors that make up the parallel computer. The simplest decompositions is the so-called "domain" decomposition. In a "domain" decomposition, the physical space of the simulation is partitioned and each processor manages the degrees of freedom associated with a contiguous portion of the underlying physical space. In our case, the underlying physical space is the three-dimensional space in which the bodies move. Domain decompositions are generally susceptible to "load imbalance," a situation in which the computational loads assigned to individual processors are not exactly equal, resulting in the the parallel computation, as a whole, executing at the speed of the most heavily burdened processor.

Thus, we must use a domain decomposition to assign bodies to processors. We use "orthogonal recursive bisection," ORB,[40] to partition the particles among processors. The object of ORB is to partition space into a structure similar to Figure 7.1, assigning all the particles in one partition to one processor. The goal is to choose a partition in which the amount of work in each domain is equal. The basic steps in ORB are outlined in Code 7.1.

Orthogonal recursive bisection repeatedly splits sets of processors in half until there is only one processor in each set. Each time a set of processors is divided in half, ORB partitions the remaining domain between the two subsets. When there is only one processor in each set, each processor has a spatial domain associated with it, and taken together, the spatial domains of all the processors partition the entire physical space. An outline of the ORB procedure is shown in Code 7.1. At each iteration of the loop in Code 7.1, the physical domain is partitioned by selecting a particular Cartesian direction, and a value of the the corresponding Cartesian coordinate. Space is partitioned into the region "above" the coordinate

```
ORB( )
    set the current processor subset to all processors.
    dowhile( more than 1 processor in the current subset )
        choose one of the Cartesian coordinates to split.
        find a splitting coordinate for the bodies
            in the current  processor subset.
        decide if ''this'' processor
            will be above or below the split.
        place those bodies which are on the other side of
            the split in a buffer.
        exchange the buffer with a processor on the other
            side of the split.
        incorporate the newly received bodies into the
            local list of bodies.
        update the current processor subset so that it
            contains only those processors on the same
            side of the split as ''this'' processor.
    enddo
endfunc
```

Code 7.1. Function `ORB`, which performs an orthogonal recursive bisection to obtain a decomposition of bodies into processors.

and the region "below." It is possible to generalize ORB to allow more general partitions, For example, one could allow partitions along arbitrary planes, rather than only those perpendicular to the coordinate axes. Such a partition would lead to processor domains which are general convex polygons, rather than rectangular parallelepipeds. This would present difficulties for the `DomainOpeningCriterion` function discussed in Section 7.3, but is, in principle, possible.

We shall find that the tree-building algorithm exchanges data between the subsets defined by the splitting coordinates of the ORB. The communication proceeds by iterating over each ORB split. For each one, every processor must exchange data with another processor on the other side of the split, i.e., in the other processor subset. If we require that for each split, every processor in one subset communicates with a unique processor in the other subset, then the pattern

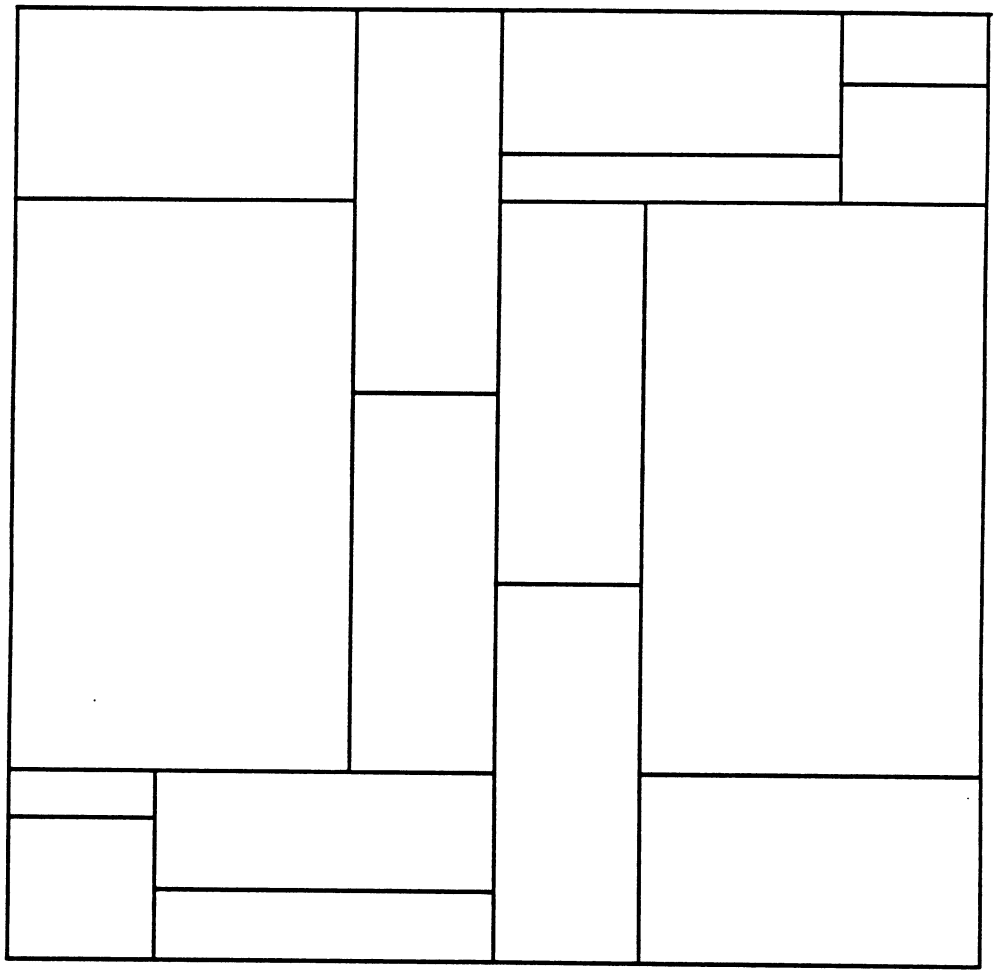


Figure 7.1. A division of space in to rectangular domains, as are produced by orthogonal recursive bisection.

of communication is identical to that of a hypercube. Thus, the most natural architecture on which to implement ORB, and the parallel BH algorithm, is a hypercube. This does not imply that hardware configured as a hypercube is necessary for the algorithm to work. It simply means that the communication paths correspond to those in a hypercube, but those data paths may be realized with hardware or via routing through some other network.

7.1.1. Hypercube communication pattern.

Let us consider the hypercube defined by the ORB further. In a hypercube of $\log_2(N_{proc})$ dimensions, each processor is connected to $\log_2(N_{proc})$ neighbors. A numbering scheme exists, in which processors are numbered from 0 through $N_{proc} - 1$. Each processor is connected to the $\log_2(N_{proc})$ other processors whose numbers, when represented in binary, differ in exactly one bit. Conversely, connections can be assigned a "channel" number, which is the unique bit which differs in the two processor numbers on the two ends of the connection.[6]

The loop in Code 7.1 can now be identified with a loop over channel numbers. For each channel number, ORB determines a Cartesian direction to split, a value of the Cartesian coordinate which bisects the data, and which half of the processors are assigned to which portion of the data.

7.1.2. Estimating work loads and bisection.

In order to apply ORB, one must be able to estimate the work associated with a given spatial domain (rectangular parallelepiped). Otherwise, there would be no way to find a bisection of the work into two equal parts at each iteration. In the N-body program, this is a simple matter. During each time-step, the work associated with each particle can be recorded (either by reading the system clock, or simply by counting interactions). The nature of the differential equation solver used for the time-integration requires that the system's gross features not change dramatically from one timestep to the next. The work associated with a body is roughly proportional to the logarithm of the local density of bodies in its neighborhood. Clearly, if any body's local particle density fluctuates rapidly then

the time-step is much too large. Thus, the work necessary to compute the force on a particle in the last timestep is guaranteed to be an excellent estimate of the work in the subsequent timestep. This gives us the desired relationship between the work assigned to a processor and the bodies that lie within the processor's domain. Given a domain, we can estimate the work associated with it by identifying the bodies that lie within it, and adding up their associated work values from the last time-step. To choose a splitting coordinate, x_{split}, we can define a function, $W(x)$, which corresponds to the fraction of work in the portion of the current `Domain` below the coordinate, x,

$$W(x) = \frac{\text{Work(sub-domain below } x)}{\text{Work(whole domain)}}. \tag{7.1}$$

The median coordinate, x_{split}, is the root of the equation,

$$W(x_{split}) = 0.5, \tag{7.2}$$

which is easily found with a few iterations of any general purpose root-finder.

A method for evaluating $W(x)$ is shown in Code 7.2. First, each processor in the current `Domain` sums the work associated with particles it controls. Then these partial sums must be transmitted and added so that each processor in the subset has the value associated with the entire subset. This implementation is not strictly optimal, as it fails to use information learned on previous evaluations. In particular, it could be improved by partially sorting the bodies, each time it is called, into those above X and those below X. Then, subsequent evaluations could restrict the loop over bodies to a much smaller set. This optimization has not been implemented because the total time spent in the naive `WeightFrac` is not significant compared to other phases of the algorithm.

This function `WeightFrac` implicitly contains interprocessor communication, via the global aggregation function, `Combine`. The functionality of `Combine` is provided as a system call in some software environments, e.g., ExPress, CrossIII. In any event, it is easy to implement in terms of the more basic send/receive primitives.[6] The second argument to `Combine` is a function of two arguments which

returns the result of some commutative and associative operation. Combine returns the result of applying the operator to all the arguments in a ProcessorSubset. The result of Combine is identical in all of the processors in a ProcessorSubset.

It is useful to recall that a great deal of accuracy is not required in the root-finder. Our experience indicates that locating the root with accuracy greater than a few percent is a waste of time, as the predictive power of the last timestep's estimates is only accurate to a few percent anyway. Finally, we note that on the very first timestep, the work load associated with the last timestep is not known. Lacking any knowledge to the contrary, we simply assume that all bodies require an equal amount of work, which leads us to assign an equal number of bodies to each processor.

```
WeightFrac(bodylist, X, CartDirection)

    WorkAbove = 0
    WorkBelow = 0
    for ( each Body in Bodylist )
        if( Coordinate of Body in CartDirection > X )
            WorkAbove += Work(Body)
        else
            WorkBelow += Work(Body)
        endif
    WorkAbove = Combine(WorkAbove, ADD, ProcessorSubset)
    WorkAbove = Combine(WorkBelow, ADD, ProcessorSubset)
    return WorkBelow/(WorkAbove+WorkBelow)
endfunc
```

Code 7.2. Function WeightFrac which evaluates the function, $W(x)$, used to find the a splitting coordinate, x_{split}.

Unfortunately, the total computational effort is not entirely contained in the force calculation. In parallel, a significant fraction (see Section 8.2, f_{cplx}) of the computation is spent building the tree and computing multipoles. This work load is not so neatly assigned to individual bodies, as the size and complexity of the

tree needed by a given processor depends on the distribution of bodies outside of the domain of the processor. In addition, this work load is not exactly balanced by the requirment that the force calculation work load be balanced. It is easy to compensate for this discrepancy by seeking a division of the force-calculation work load which does not assign exactly half to each `ProcessorSubset`. Although the exact amount of tree-build work associated with a given domain is not easily predicted a priori, we can again make use of the fact that the structure of the tree, and hence, the expense of building it, may not change dramatically from one timestep to the next. Thus, if the load was not balanced on the last timestep, then we can approach a more equitable distribution by seeking a splitting coordinate, x_{split}, which is not exactly the median of the distribution of work, $W(x)$. Thus, our "median finder" no longer seeks a root of Eqn. 7.2, but instead seeks a root of

$$W(x_{split}) = p, \tag{7.3}$$

where p is a "percentile" which is adjusted dynamically.

We define the load imbalance on the last timestep as

$$I_{old} = \frac{\text{Total time in } \texttt{ProcessorSubset} \text{ assigned below } x_{split}}{\text{Total time in both } \texttt{ProcessorSubsets}} - 0.5. \tag{7.4}$$

We compute p_{new} by

$$\begin{aligned} p_{new} &= p_{old} - I_{old}\omega, \\ 0 \leq \omega &\leq 1. \end{aligned} \tag{7.5}$$

The constant, ω, is arbitrary, and is chosen to prevent the percentile from oscillating because of overcorrections. We have used $\omega = 0.75$. Although such large fluctuations have not been observed, it seems prudent to also limit the range of allowed percentiles to some fixed range like

$$0.25 < p_{new} < 0.75. \tag{7.6}$$

Occasionally, during the course of the calculation, the decomposition undergoes gross changes. These occur when choice of splitting directions changes and

also on the second timestep when information about the work associated with each body becomes available. Whenever such changes in the decomposition occur, the assumptions underlying Eqn. 7.5 are invalid, and it is best to revert to a simple median, i.e., $p_{new} = 0.5$.

7.1.3. Choosing between $x > x_{split}$ or $x < x_{split}$.

Let us assume that the bit (channel) corresponding to each split starts with the most significant bit, and progresses toward less significant bits for each split. The exact order is of little importance, but a definite order aids in understanding the process. Recall that for each bit, half of the processors in the current `Domain` will have a 1 in the corresponding location in their processor number, and half of the processors will have a 0.

An obvious method for determining whether a processor falls in the $x > x_{split}$ subdomain is simply to assign all those processors with a 1 in the corresponding processor number bit to the $x > x_{split}$ subdomain, and vice versa. This assignment is equivalent to a simple linear assignment of processors to domains, as illustrated by Figure 7.2. This decomposition has the drawback that neighboring processors are not near to one another in the presumed hypercube topology. For example, for data to travel from processor 011 to processor 100, it must pass through processors 111 and 101. This problem can be resolved by numbering the processors according to a Gray code, rather than the "counting" code of Figure 7.2.[41, 6]

A Gray code can be incorporated into ORB by changing the correspondence between the value of the bit corresponding to the channel and the boolean value, `Above`, corresponding to whether this processor is assigned to the subset above the split point or vice versa. If, for each channel and in each processor, we make the decision according to

$$\texttt{Above} = \texttt{Above}\ \textbf{Xor}\ \texttt{Bit}, \tag{7.7}$$

then the decomposition of eight processors is as in Figure 7.3. Evidently, this is a Gray code mapping similar to that described by Fox et al[6]. It is not hard to show by induction that the assignment of processors to domains determined in

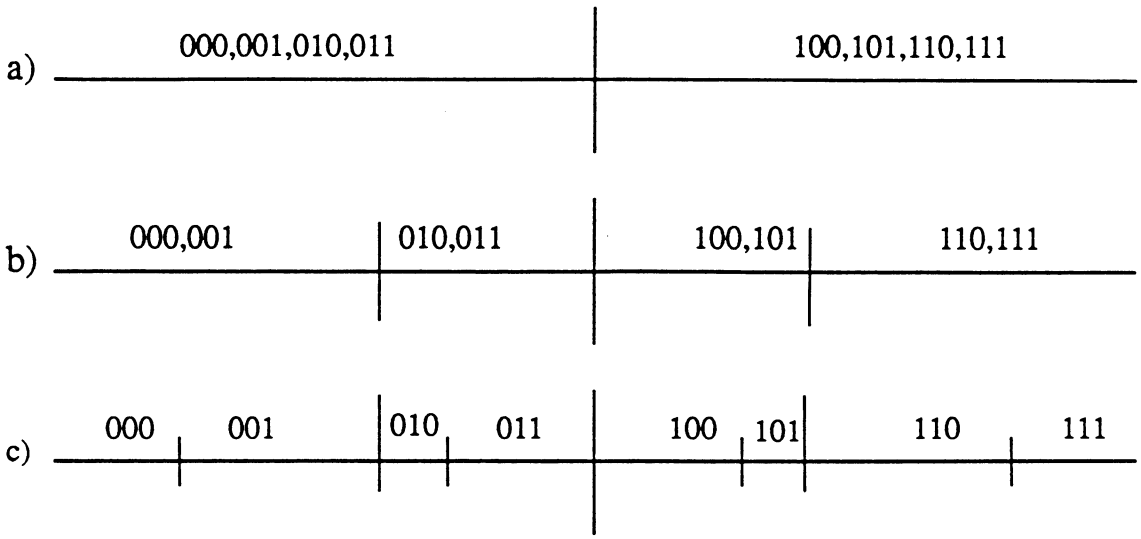


Figure 7.2. A one dimensional decomposition: a) after channel 2 is split, b) after channel 1 is split, c) after channel 0 is split.

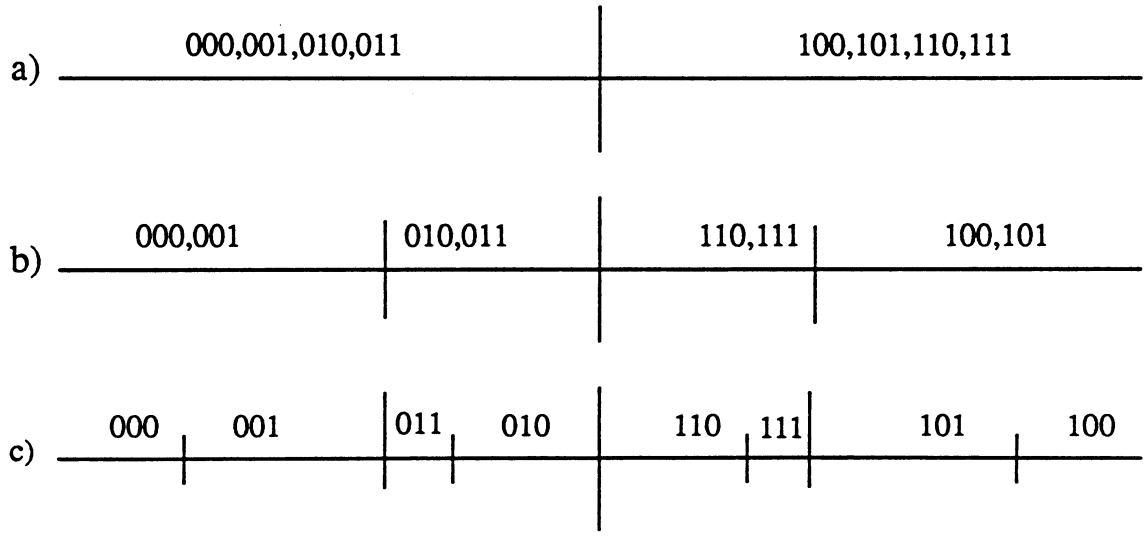


Figure 7.3. A Gray code decomposition, using ORB on eight processors: a) after channel 2 is split, b) after channel 1 is split, c) after channel 0 is split.

this way always gives rise to a Gray code, as long as the number of processors is an integral power of two.

7.1.4. Choosing which Cartesian direction to split.

We now turn to the question of which Cartesian dimension to split corresponding to each channel of the hypercube. There are two considerations which affect the decision. First, we must consider the desire to have nearby domains assigned to processors which are relatively close in the hypercube. This can be accomplished by selecting the dimension to split without reference to the current domain or processor subset, e.g., in three Cartesian dimensions,

$$\text{Direction} = \text{channel mod } 3. \tag{7.8}$$

This is the strategy used in Figure 7.1. In Figure 7.4, we show the same set of domains, with processor numbers assigned by the Gray code method of Eqn. 7.7. The resulting pattern of domains is always equivalent to a rectangular grid, in the sense that row and column numbers for domains are well defined. If the processor assignment decisions are made according to a separate Gray code criterion for each Cartesian direction, then adjacent processors in the same row or column will be hypercube neighbors. Unfortunately, there is no guarantee that the logical rows and columns represent neighboring domains in the physical space of the decomposition. That is, the domains that border a given domain in physical space may not be in the same logical row or column.

Figure 7.5 shows a domain decomposition that might arise from a system containing two well separated "galaxies." The large border between processors 0110 and 1010, does not correspond to a hypercube connection. Even in this fairly pathological situation, however, most of the domain borders still correspond to hypercube channels.

It is important to remember that the structure represented in Figure 7.4 and that represented in Figures 2.1 through 2.4 are logically separate entities. The latter show the data structure used by the BH algorithm, while Figure 7.4

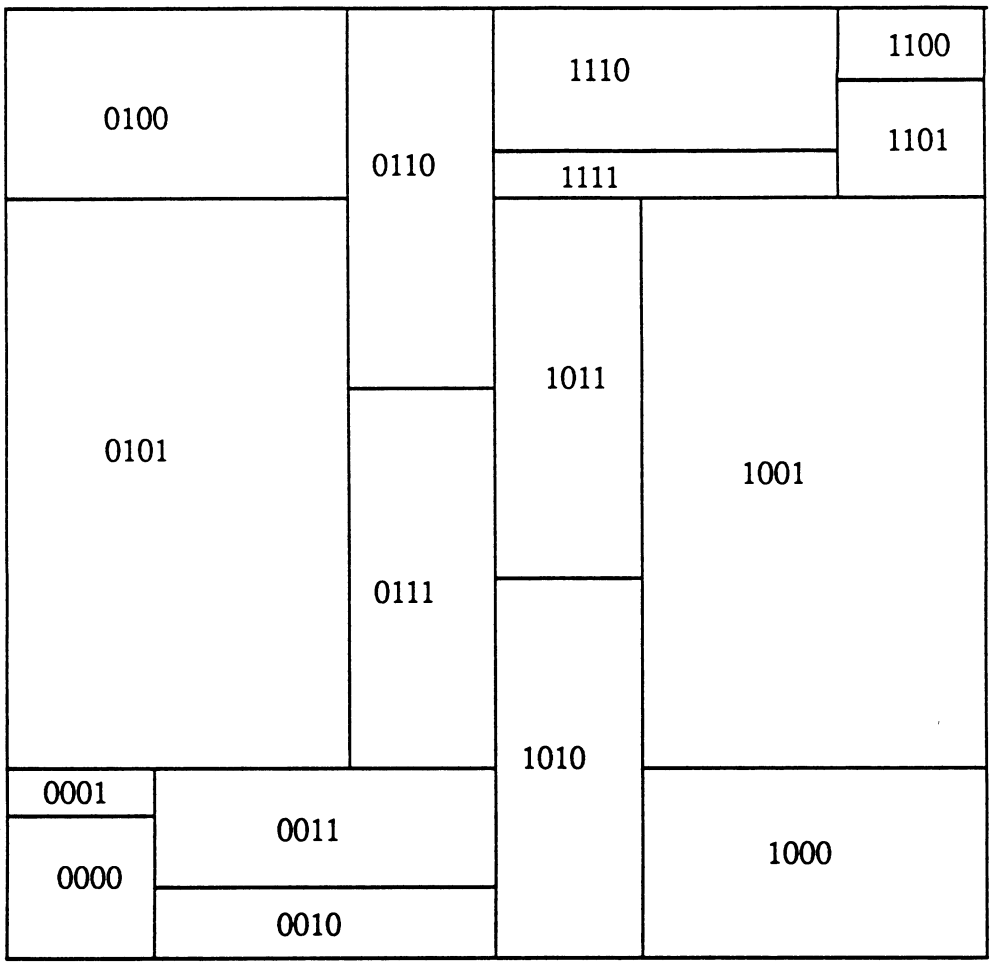


Figure 7.4. A typical domain decomposition, with processor numbers assigned by the Gray Code method of Eqn. 7.7.

is a picture of a domain decomposition which assigns bodies to processors. The processor boundaries in Figure 7.4 do not necessarily correspond to the boundaries of cells in the BH tree.

It is clear from Figure 7.5 that when `FindSplit` is based on a median (or percentile) finder, the domains that result can become very long and thin. The reason is that the density of bodies, and hence, workload, in physical space can be highly inhomogeneous. Splitting the work in half can easily split physical space into two regions of widely disparate sizes. The smaller piece will generally be much longer than it is wide. This tendency adversely affects the memory required by each processor. The reason is that a processor must allocate enough memory for the bodies in its domain, as well as all for other cells and bodies that are used in the force calculations for the bodies in its own domain. For illustrative purposes, let's imagine that a processor's domain is of dimension, L_x, L_y and L_z in each of the three Cartesian directions. Furthermore, let's assume that space must be allocated in the processor for all bodies and cells within L_{int} of the boundary of the domain in any Cartesian dimension. Then the total memory is approximately proportional to

$$M \propto (L_x + 2L_{int})(L_y + 2L_{int})(L_z + 2L_{int}). \tag{7.9}$$

Figure 7.6 illustrates the situation in two dimensions. It is easy to show that with the volume, $V = L_x L_y L_z$, fixed, memory is minimized by a cubical geometry with $L_x = L_y = L_z$. Furthermore, if the domain is interpreted as being associated with a `ProcessorSubset` containing two processors, and it is split along, for example, L_x, then the total memory required by the two processors will be

$$M_x \propto (L_x + 4L_{int})(L_y + 2L_{int})(L_z + 2L_{int}). \tag{7.10}$$

Splitting along the largest of the three Cartesian directions leads to the smallest total memory requirement. Thus, if we wish to reduce the total memory required by the parallel implementation, we should split domains along the dimension corresponding to the maximum extent of the current Domain, at each

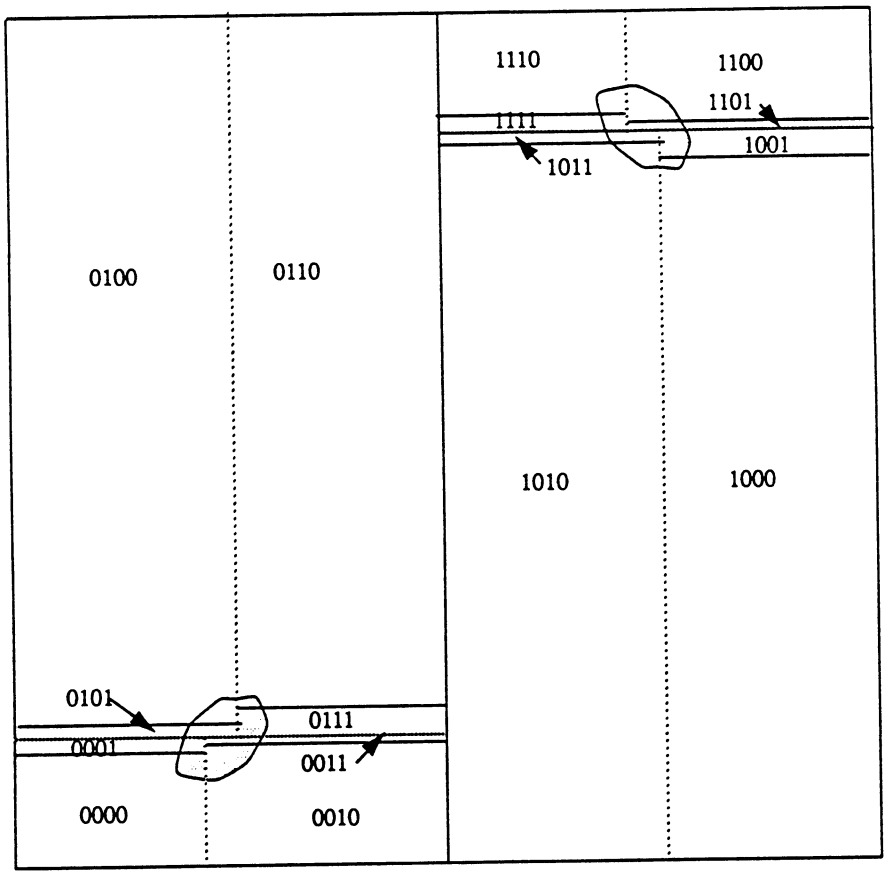


Figure 7.5. A two-galaxy system which gives rise to processor domains with high aspect ratios.

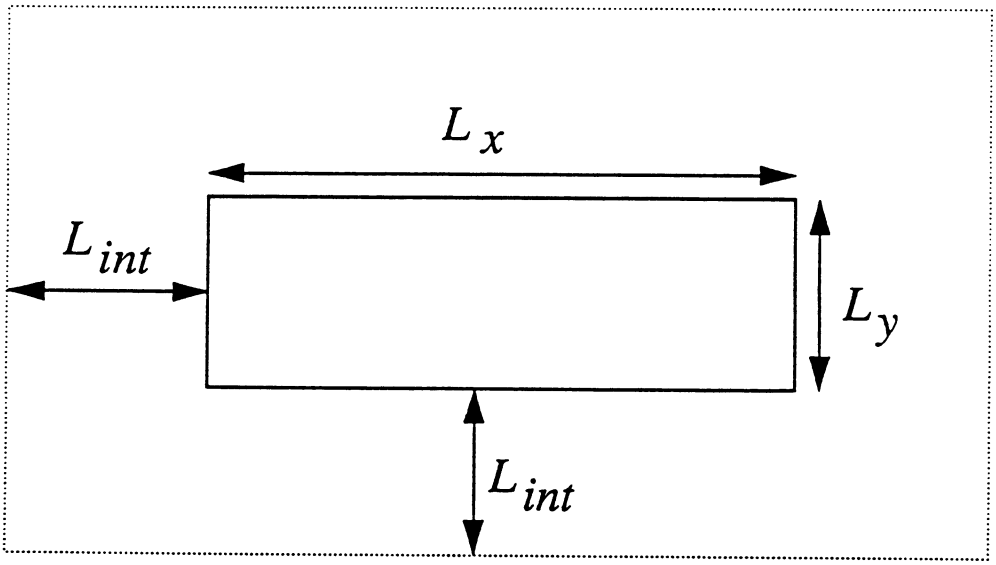


Figure 7.6. Relationship between memory and the geometry of a processor's domain. The processor's domain is L_x by L_y, but it must have roughly enough memory to store the data in a volume which is L_{int} larger in each direction.

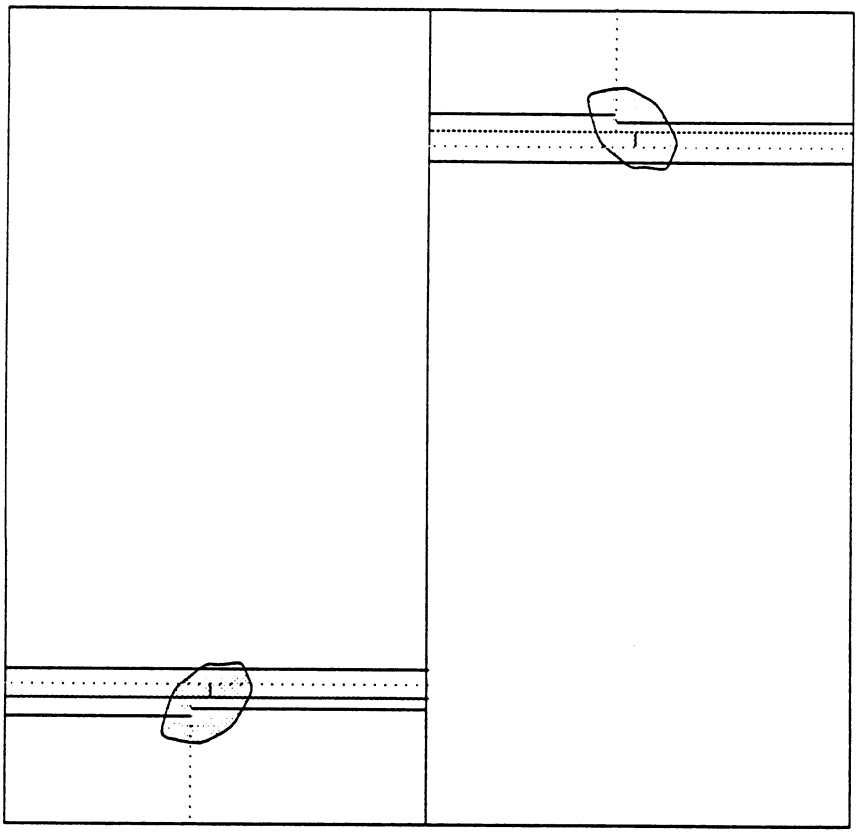


Figure 7.7. The same two-galaxy system as in Figure 7.5, but the decomposer always splits the longest Cartesian direction.

step in ORB. Figure 7.7 shows the domains that result from splitting the two galaxy example of Figure 7.5 along the longest dimension. The domains have somewhat smaller aspect ratios, but the correspondence of hypercube neighbors with physical neighbors is all but lost.

As we shall see from the detailed timings, a significant fraction of the computational overhead associated with the parallel implementation is proportional to the amount of memory required by each processor. This overhead is much greater than the overhead associated with data communication. Thus, we adopted the strategy of splitting in the longest dimension in ORB. One additional complication is that the procedure for load balancing, which uses information about the quality of load balance on the last time-step, works best when the splitting directions remain the same from one time-step to the next. For this reason, we introduced a small amount of "inertia" so that the splitting coordinate changes only if the extent of the domain in its longest direction is more than 20% larger than the extent in the direction that was split on the last timestep.

7.2. Building a "locally essential" tree.

After the decomposition phase, each processor has the bodies that it must compute the forces on. It does not, however, have enough information to compute those forces. Clearly, information must be exchanged which will allow each processor to evaluate the forces on each body. It is important to realize that the nature of the hierarchical algorithm allows each processor to neglect most (but not all) of the tree data. This ability to make due with only a subset of the entire data set is, in fact, the essence of the domain decomposition. If it were not for this fact, the domain decomposition would be a waste of time, because every processor would need to construct the entire BH tree, and there would be no reason whatsoever to restrict each processor to evaluating the forces on particles in a restricted spatial domain. We shall refer to the subset of the tree which is necessary for the evaluation of forces in a processor's domain as the "locally essential" subset. It is interesting that the existence and exploitation of a "locally essential" subset of a global data set is

common to many algorithms which may be parallelized loosely synchronously.[6]

After the decomposition phase, each processor controls a region of space, and must compute forces for all of the bodies within that region. Imagine that each processor obtained a copy of the entire hierarchical tree. Each body, when it traverses the tree, as in Code 4.5, will access only a subset of the tree. In fact, since the bodies are restricted to a limited domain, most of the tree is completely unnecessary for any of the bodies managed by a single processor. This is a fortunate result, because without it, we would have to store the entire tree in each processor. Large simulations would be impossible on distributed memory parallel computers because there would not be enough local memory for each processor.

```
DomainOpeningCriterion( Domain, Cell )
    if ( there exists a TestPoint within Domain such that
        OpeningCriterion(TestPoint, Cell) is TRUE )
        return TRUE
    else
        return FALSE
    endif
endfunc
```

Code 7.3. Function `DomainOpeningCriterion`, which tests whether a call to `ComputeField` for a body at any point in a domain could require that the children of the cell be accessed.

Let us assert the existence of a procedure, `DomainOpeningCriterion`, which tells us whether any body in a specified rectangular domain could possibly require inspection of the descendants of a given cell or conversely, whether the cell may be treated as terminal, for the purposes of any body which lies in the domain. `DomainOpeningCriterion` is shown schematically in Code 7.3. If such a function exists, then we can use it to test cells in a "local" BH tree to determine whether they are needed by other processors. The function `BuildTree` in Code 7.4 uses this idea to iteratively build up a local tree which eventually contains precisely the data needed to evaluate the fields at any position within a processor's own

```
BuildTree(bodylist)
    Tree = EmptyTree
    BuildTreeLocal(bodylist, Tree)
    for( each bisection created by DomainDecomp )
        Traverse the Tree and queue any data which may be
            necessary to the domain on the other
            side of the bisector.
        Traverse the Tree again, and delete any data which
            can never be necessary on ''this''
            processor's side of the bisector.
        Exchange queued data with corresponding processor
            on other side of bisector, and merge it
            into the Tree.
    endfor
    Do any necessary housekeeping to finish the tree.
endfunc
```

Code 7.4. Function `BuildTree` which constructs a locally essential representation of a tree.

rectangular domain.

Recall that ORB produced a set of bisectors. The first one divides the universe into two halves, each of which is a rectangular region of space. Half of the processors are assigned to one of the domains, and half are assigned to the other. Then each of those domains is split again into two more rectangular regions, and half of each subset of processors is assigned to each of the new domains. This process repeats until there is one rectangular domain per processor. At every stage of the ORB, a processor identifies itself with one or the other of the two domains that were created. The function `BuildTree` uses this fact to organize the way it transmits data and constructs a locally essential tree.

After each iteration of the `for` loop in `BuildTree`, every processor is a member of a subset of processors which have not yet communicated any information amongst one another. Because of the nature of ORB, this subset forms a subcube in the hypercube topology. Furthermore, the spatial domains assigned to the

processors in this subset, when taken together, form a rectangular region.

At the end of each loop iteration in `BuildTree`, all the information that may be necessary for a force calculation anywhere in a processor's current subdomain was either originally stored on one of the processors in the subset, or has been transmitted to one of the processors in the processor's current subset by a prior "exchange/merge" operation. This assertion is easy to prove by induction. It is obviously true before the first iteration because all relevant data is certainly available somewhere in the entire ensemble. If it is true before an iteration, then it must also be true after, because any necessary data is either already in a processor in the subset or in a processor which will exchange data with a processor in the subset during the iteration. The first tree traversal in `BuildTree` guarantees that the data will be identified and transmitted to one of the processors in the subset. The `DomainOpeningCriterion` function is used during this tree traversal to identify which data needs to be transmitted.

After data has been exchanged between each of the bisectors in `BuildTree`, each processor is in a subset that consists only of itself. Thus, based on the assertion of the last paragraph, we conclude that each processor has obtained all the data necessary for force calculations anywhere in its domain. This is precisely the desired, locally essential data set. The pattern of communication exhibited by `BuildTree` is similar to that of Furmanski's "Crystal Accumulator."[42] The crystal accumulator also consists of an iteration over hypercube channels, with interleaved operations which modify the transmitted data. In the case of `BuildTree`, the interleaved operations are quite complex, consisting of merging the new data into the BH tree followed by a traversals of the tree to identify the next batch of data to send, followed by another traversal to prune unnecessary data.

Figure 7.8 shows schematically how some data may travel around a 16-processor system. The second tree traversal in `BuildTree` conserves a processor's memory by reclaiming data which was transitted through a processor, but which is not needed by the processor itself, or any other member of its current subset. The body sent from processor 0110 through 1110 and 1010 to 1011 in Figure 7.8

would likely be deleted from processor 1110's tree during the pruning on channel 2, and from 1010's tree during the pruning on channel 0.

7.2.1. Data types for the parallel `BuildTree`.

In order to implement the tree traversals and the merge operations implied by `BuildTree`, we will need to distinguish between some new types of cells and bodies. We will also need to be able to set some boolean flags on a per-cell or per-body basis. Thus, we introduce the following data types:

Local Body

> Abbreviated `LBody`, these are identical to the bodies manipulated by the sequential code fragments. They contain a position, velocity, mass, and perhaps other data of physical interest. The position of an `LBody` is always within a processor's spatial domain.

Foreign Body

> Abbreviated `FBody`, these are bodies which have been transmitted into a processor because they may be needed for a force calculation of one of the `Lbodies`. It contains a position and a mass, but lacks a velocity, and other physical data.

Leaf Cell

> Abbreviated `LCell`, these are identical to the terminal cells manipulated by the sequential code fragments. They are really just the head of a list of `Bodies`. They contain no multipole information because such information would never be used in lieu of a direct sum over the list of `Bodies`, as discussed in Section 4.5.

Internal Cell

> Abbreviated `ICell`, these are cells internal to the BH tree. They contain multipole expansions, as well as pointers to up to eight daughter cells. They are identical to the internal cells in the sequential code fragments.

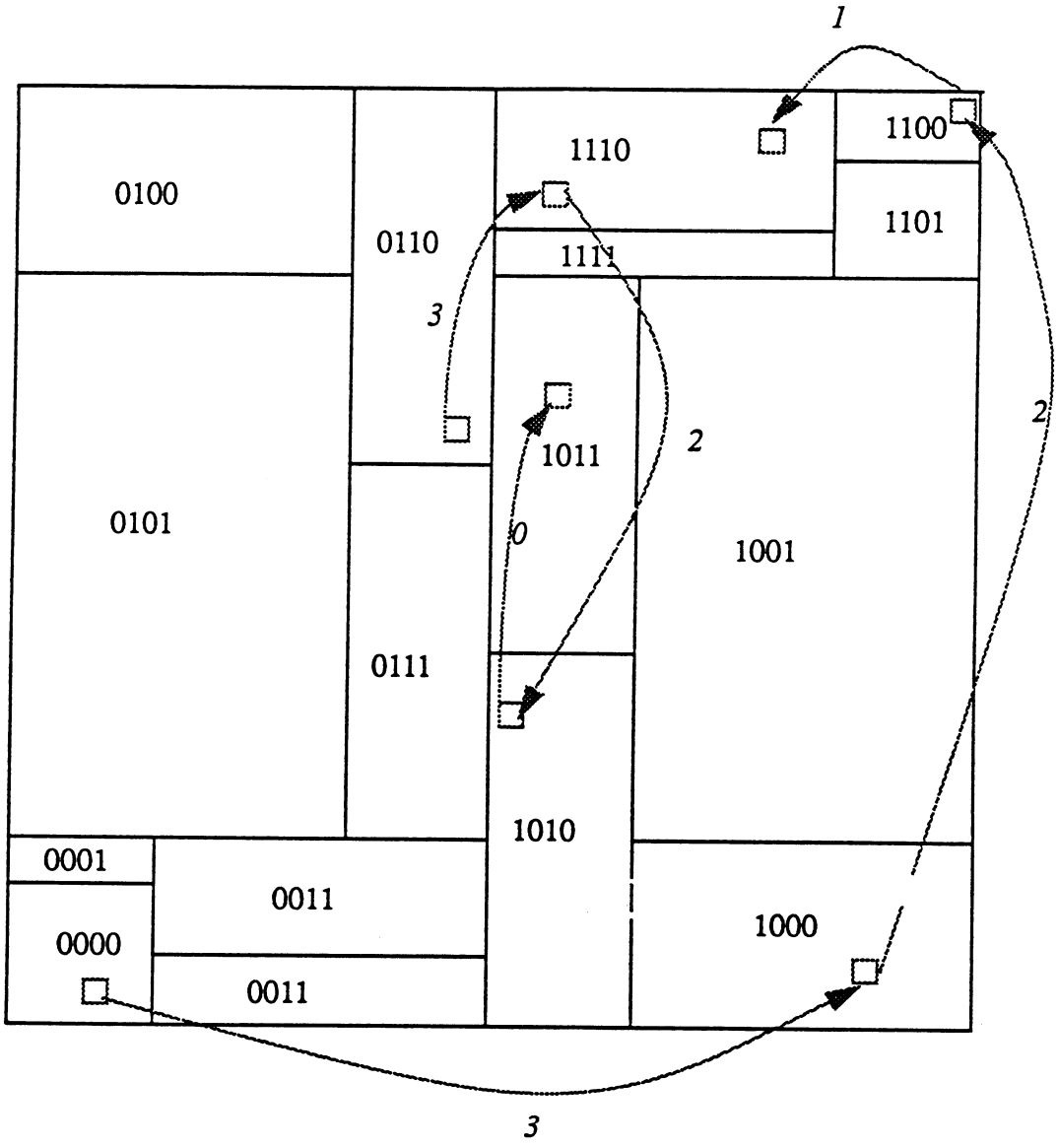


Figure 7.8. Data flow in a 16 processor system. Arrows indicate the flow of data. Arrows are labeled by hypercube channel number.

Multipole Terminal Cell

Abbreviated `TCell`, these are also terminal cells in the BH tree, but they contain a multipole expansion to represent their contents. The region represented is guaranteed to be outside the processor's domain, and it is furthermore guaranteed that no force evaluation in the processor's domain could ever require "looking inside" a `TCell`.

We will assume that there is some form of memory management software that efficiently allocates and reclaims single objects of each of these types. The functions `NewTCell`, etc., in the Code fragments below, and the function `FreeMemory` constitute our abstraction of the memory management system. We will not assume the existence of any "garbage collection" system, so we must be careful to explicitly return the memory associated with any object that becomes disconnected from the data structures.

Some additional information must be recorded on a per-cell basis. This information is:

Mpoles Correct

Internal cells have a bit which identifies whether the multipoles stored with this cell are correctly computed to correspond with the positions and multipoles of its descendants. `TCells` always have a correct multipole, and `LCells` have no multipole moments at all.

Transmission Pending

`ICells`, `TCells` and `LCells` have a bit which indicates that they will soon be transmitted, and should not be removed from memory by a tree-pruning operation.

Free when Sent

Another bit is used to indicate that the space used by an `ICell`, a `TCell` or the `Bodies` in an `LCell` should be returned to the memory management system after the data exchanging routines have communicated the data stored in the memory. This bit will be set during the pruning operation when the `TransmissionPending` bit is detected.

Nbody

The number of bodies contained in an `ICell` is useful for optimizing the computation of multipole moments for `ICells`.

ContainsTCells

In addition to the number of bodies, a bit which indicates whether any terminal descendants of an `ICell` are `TCells` is also necessary for optimizing the computation of multipole moments.

7.2.2. Control structures for the parallel `BuildTree`.

With these data types and flags, we have sufficient framework to describe the components of the parallel `BuildTree` algorithm.

7.2.2.1. Selecting data for transmission.

```
TraverseAndQueue ( Cell )
    if( Cell is type LCell or TCell )
        EnqueueForSending(Cell)
    else if( !DomainOpeningCriterion( Cell, OtherDomain ) )
        EnqueueForSending(Cell)
    else
        for( each child of Cell )
            TraverseAndQueue( child )
        endfor
    endif
endfunc
```

Code 7.5. Function `TraverseAndQueue`

It is clear from Codes 7.4 and 7.5, that the `TraverseAndQueue` function starts at the root of the tree and descends toward the leaves, selecting a fraction of the nodes for special treatment. The nodes which are enqueued satisfy two properties:

1. Their parent satisfies the `DomainOpeningCriterion` test for the domain `OtherDomain`, meaning that the node itself may be required by the processor subset on the other side of the current bisector.

```
EnqueueForSending( Cell )
    if( Cell is type ICell )
        ComputeMpoles(Cell)
    endif

    Set(TransmissionPending, Cell)
    record a pointer to Cell in a list to be scanned
        during the communication phase.
endfunc
```

Code 7.6. Function `EnqueueForSending`

2. The node itself is either terminal, or it fails the `DomainOpeningCriterion` test for the domain `OtherDomain`, meaning that the processor subset on the other side of the bisector can never need to "look inside" the node.

These are exactly the objects that must be transmitted to the other processor subset. Thus, `TraverseAndQueue` in Code 7.4 selects the right objects for transmission.

Cells that are enqueued for transmission pass through the `EnqueueForSending` function immediately upon being identified. In addition to the mundane task of storing a pointer to the cell, (or its constituent Bodies, in the case of an `LCell`), `EnqueueForSending` guarantees that the multipole information stored in `ICells` is up-to-date by calling `ComputeMpoles`. It ensures that the multipoles are correct at this early stage because a significant amount of manipulation of the tree will ensue between the time a cell is enqueued and the time it is eventually transmitted. `ComputeMpoles` must be called early enough that we can be sure that the tree has not been disturbed by "pruning" or insertion of information from other processors, before the multipoles are calculated.

7.2.2.2. Computing multipoles.

The function `ComputeMpoles` is shown in Code 7.7. It is similar to the sequential function analyzed in Code 4.1, but there are some important differences. The most significant difference is that in parallel, we are only computing multipoles when

they are needed. The `MultipolesCorrect` bit signals that the multipoles stored in a cell are up-to-date with respect to the descendants of the cell. We must take care not to use the parallel axis theorem to translate the multipoles of a cell unless the multipoles are correct. Another complication is that the leaves of the tree can contain `TCells` as well as `Bodies`. When the descendants of a cell include one or more `TCells`, it cannot be advantageous to use direct summation to move its multipole moment, because the `TCell` would require a call to `ParAxis`, equal in expense to the one call to `ParAxis` that would translate the entire cell. Thus, we must take care that the optimization discussed in Section 4.2, which conditionally uses direct summation rather than the parallel axis theorem is never applied to cells that contain a `TCell`. The flag `ContainsTCells`, which tells us whether a cell has descendants of type `TCell`, is set, along with the value of `Nbody` in the insertion functions in Codes 7.15 and 7.19.

```
ComputeMpoles(Cell)
    if(IsSet(MpolesCorrect, Cell))
        return
    ZeroMpole(Cell)
    for( each Child of Cell )
        TranslateMpoles(Child, Cell)
    endfor
    Set(MultipolesCorrect, Cell)
endfunc
```

Code 7.7. Function `ComputeMpoles`, which recursively calls `TranslateMpoles` to determine the multipole moments of a cell.

7.2.2.3. Pruning the tree.

Several considerations enter into the process of pruning the tree. First, of course, we must only prune data that we will never need again. Unnecessary cells have two properties:

1. They fail the `DomainOpeningCriterion` test applied to the domain of the processor's own subset of processors.

```
TranslateMpoles(Child, Cell)
    if( Child is type LCell )
        for(each Body in Child)
            BodyMpole(Body, Cell)
        endfor
    else if( Child is type TCell )
        ParAxis(Child, Cell)
    else if(IsSet(MpoleCorrect, Child) AND
        (IsSet(ContainsTCells, Child) OR Nbody(Child)>N_cutoff)
        ParAxis(Child, Cell)
    else
        for(each Grandchild of Child)
            TranslateMultipoles(Grandchild, Cell)
        endfor
    endif
endfunc
```

Code 7.8. Function `TranslateMpoles` computes the multipole moments of a cell either by direct summation of the contained bodies, or by use of the parallel axis theorem (Section 4.2).

2. They are not terminal. The function `TraverseAndPrune` in Code 7.9 selects precisely the nodes that satisfy these two conditions and applies the function `PruneDescendants` to them.

The function `PruneDescendants` is applied only to `ICells`. Its purpose is to recursively delete all of the descendants of the node, returning their memory to the memory management system for later use. Since the data "beneath" the node is about to be destroyed, this is the last opportunity to compute a correct multipole expansion for that data. Thus, before destroying the data, it must first ensure that the multipole moment stored in the cell is up-to-date. Finally, the `ICell` should be replaced by a `TCell` since it is now terminal and contains a multipole expansion to represent its contents. The process of recursively deleting the descendants of a node is complicated by the fact that a node scheduled for deletion may also be

```
TraverseAndPrune( Cell )
    if( Cell is type ICell )
        if ( !DomainOpeningCriterion( Cell, MyDomain ) )
            PruneDescendants(Cell)
        else
            for( each Child of Cell )
                TraverseAndPrune( Child )
            endfor
        endif
    endif
endfunc
```

Code 7.9. Function `TraverseAndPrune` uses `DomainOpeningCriterion` to identify sub-trees which are no longer needed.

scheduled for transmission. The function `ZapCell` in Code 7.12 solves this problem by testing the `TransmissionPending` bit. If it is set, then the `FreeWhenSent` bit is also set, and the deletion of the cell or body is deferred until after the contents of the body or cell has been safely transmitted. If it is not set, then there is no reason not to delete the cell or body immediately. If the `FreeWhenSent` bit is set in any body or cell, then we can be sure that that body or cell has been completely disconnected from the tree. There is no "path" from the root of the tree to it, and we may safely delete it immediately upon its transmission, without fear of leaving a "dangling pointer" in the tree.

7.2.2.4. Communication systems.

Communication systems on parallel machines are extremely diverse, and any particular set of assumptions is likely to be violated by some system. This makes it difficult to abstract a set of communication primitives that map naturally onto a variety of software/hardware combinations. The communication requirements of the algorithm are extremely simple. We need the ability to queue a fairly large number of fixed size blocks for transmission, the ability to execute a simple function, `SendFilter`, after each block is safely on its way, and the ability to process

```
PruneDescendants( Cell )
    ComputeMpoles(Cell)
    for( each Child of Cell )
        FreeSubtree(Cell)
    endfor
    Replacement = NewTCell()
    copy multipoles from Cell to Replacement.
    replace Cell in the tree with Replacement.
endfunc
```

Code 7.10. Function `PruneDescendants` calls FreeSubtree to recursively delete a sub-tree.

```
FreeSubtree(Cell)
    if( Cell is type ICell )
        for(each Child of Cell)
            FreeSubtree(Child)
        endfor
    endif

    ZapCell(Cell)
endfunc
```

Code 7.11. Function `FreeSubtree`, a simple recursive function to destroy a sub-tree.

the blocks one-at-a-time upon receipt. `SendFilter` clears the `TransmissionPending` bit and conditionally calls `FreeMemory` for each object once it is safely en route to its destination. It is shown in Code 7.13.

It is not necessary for the communication system to overlap the processes of transmission and input processing, but the algorithm can take advantage of such a system if it is available. It is easy to implement these requirements in any communication system that supports blocking, point-to-point communication, e.g., CrOSIII, ExPress, the Cosmic Kernel, Vertex, etc. Furthermore, the algorithm is structured so that communication may take place in a few very large data transfers (one per channel), or in many smaller transfers, as dictated by the availability of

```
ZapCell( Cell )
    if( IsSet(TransmissionPending, Cell) )
        Set(FreeWhenSent, Item)
    else
        if( Cell is type LCell )
            for( each Body in Cell )
                FreeMemory(Body)
            endfor
        endif
        FreeMemory( Item )
    endif
endfunc
```

Code 7.12. Function `ZapCell`, which is used by `FreeSubtree` and later to destroy the contents of a cell. If the cell's `TransmissionPending` bit is set, then deletion is defered until after the contents have been sent.

```
SendFilter( Cell )
    arrange for hardware to transmit the data in the Cell.
        if the Cell an LCell, this means send the Bodies,
        otherwise it means send the multipole moments.
    Clear(TransmissionPending, Cell)
    if ( IsSet(FreeWhenSent, Cell) )
        FreeMemory(Body)
    else
endfunc
```

Code 7.13. Filter to apply to outgoing data.

communication buffers and the details of communication latency and throughput.

In Code 7.5, we built a queue of pointers to Cells and Bodies which required transmission. This queue contains Cells of all types. In order to transmit the data in an `ICell` or in a `TCell`, we must send the entire multipole expansion stored in the cell, as well as the $\vec{r}_\gamma$ around which the multipole expansion was computed. In the case of an `ICell`, it is not necessary to send information about the eight children or other information related to the structure of the tree. Transmission of

an LCell implies transmission of the positions and velocities of the Bodies stored in it. For LBodies, it is not necessary to send the velocity or other information related to the time evolution. Thus, the stream of outgoing data is made up of two fundamentally different types of objects, TCells, i.e., multipole expansions, and FBodies, i.e., positions and masses. The details of the communication and memory management systems dictate how we should go about sending this data. It is possible to split the transmission into two separate phases, exchanging the multipole data during the first phase and the body data during the second. Alternatively, it is equally easy to mix the data types in the same stream, inserting a very small amount of control information so the receiving processor can decode what is being received. Also, it is easy to buffer the data in relatively small pieces, placing less demand on the memory resources used by the communication system, or one could send the data in a single very large transfer, minimizing the relative importance of communication latency with respect to throughput. Finally, we note that a significant fraction of the cells which are transmitted will have the FreeWhenSent bit set. The memory used by these cells or bodies may be recovered and immediately reused to store the incoming data stream as soon as the data has been safely delivered into the communication system. Thus, if memory is extremely tight, it is possible to: 1) reorder the export queue so that all bodies with the FreeWhenSent bit set are sent first, and 2) buffer the communication, or overlap the communication with the processing of incoming data so that FreeWhenSent objects can be recovered to make space for the incoming data objects and memory can be used most efficiently. The choices between these strategies all depend on details of the hardware and software and must be guided by considerations particular to a specific machine. The algorithm takes some care to allow, for example, newly received data objects to be inserted into the tree before all of the enqueued objects have been transmitted, so the decision can be made on the basis of efficiency. The algorithm performs correctly in any case.

We assume that the communication system has been set up to call the functions ReceiveFBodyFilter and ReceiveTCellFilter for each FBody and TCell

which is received. This does not imply a communication model which interrupts "regular" processing to process each incoming item. Nor does it exclude such a model, although some additional effort will be necessary to isolate "critical sections" in the insertion functions. Two obvious critical sections occur where the `FreeWhenSent` bit is set in Codes 7.12 and 7.20. The communication model may be implemented simply by buffering the incoming and outgoing data in two large arrays, exchanging them with a blocking communication call, and scanning the incoming arrays, calling the appropriate `ReceiveFilter` for each item. We found that memory was a very precious commodity on the NCube system, and adopted a slightly more sophisticated system in which the incoming and outgoing arrays are of order a few hundred objects in size, and blocking communication routines are called repeatedly to transfer the whole data set. Only pointers to the actual data objects were copied during the `TraverseAndQueue` phase of the algorithm, and since many of the transmitted data items have the `FreeWhenSent` bit set, we found that we could run very close to the edge of memory, utilizing the space just freed by the outgoing data when processing the next incoming buffer.

The choice of buffer size was motivated by the competing desires that the buffers be large enough so that communication speed not be adversely affected by latency, and small enough that they consume a negligible fraction of the available memory. On the NCube, buffers of about ten thousand bytes, or a few hundred data objects satisfy both these requirements.

The two filters that process incoming bodies are shown in Code 7.14 through Code 7.20. They are very similar to the function `Insert` in Code 2.1, which performs an identical function in the sequential algorithm. Insertion of new data into the tree is complicated by three features not present in the sequential situation.

1. The possibility that `Cells` in the tree are scheduled for subsequent transmission.
2. The presence of `TCells`, both in the incoming stream of items and in the tree into which we are inserting items.
3. The fact that incoming `TCells` have a specific size and location in the tree

from which they came, and the tree into which we insert them may not be constructed to that level yet.

The first difficulty is handled by checking the `TransmissionPending` bit in `Cells` into which new data is being inserted. If we try to insert a new item (a `Body` or a `TCell`) into an original `TCell` or `LCell` with the `TransmissionPending` bit set, then we must take care to make an exact copy of the original and substitute the copy into the tree, severing the original from the tree. Then we set the `FreeWhenSent` bit in the original element so its memory is reclaimed after it is no longer needed. This way, items which have been queued for transmission are sent in their original form. If we mistakenly modified a `TransmissionPending` item before it was sent, for example, by merging an incoming `Body` into an outgoing `TCell`, then we would effectively be returning the `Body` back to the processor from which it came, hidden inside the `TCell`. The function `ReplaceInTree` performs most of the necessary housekeeping required to sever a `Cell` from the tree and replace it with a new one.

`TCells` in the tree are `Cells` which are guaranteed to be unopenable by any processor in the current (or subsequent) processor subset. Hence, if we encounter a `TCell` at any time when we are traversing the tree looking for a place to insert a new object (`Body` or `TCell`), we can simply add the new object to the multipole expansion already in the `TCell`. When the new object is a `Body`, this is accomplished with `BodyMpole`, and when it is a `TCell` it is accomplished with `ParAxis`. These are precisely the same functions that are used in Code 7.7 to compute multipole expansions in the first place. The mathematics underlying these functions was described in Chapter 3.

A further complication arises from the fact that the `TCells` which are transmitted have a specific size and $\vec{r}_\gamma$ which is implicitly located at the center of their cubical volume. In order to insert a `TCell` into the tree, we must be certain that it represents exactly the same cell that it represented in its original processor. Figure 7.9 shows a situation in which a `TCell` is received by a processor which has not previously refined its tree to the point at which the `TCell` must be located.

The function FindParent finds the immediate parent of a TCell, if it exists in the tree, and if it does not, it refines the tree sufficiently so that it does.

```
ReceivedFBodyFilter(Bodydata)
    NewB = NewFBody()
    Copy Bodydata into NewB.
    InsertBody(NewB, Root)
endfunc

ReceivedTCellFilter(TCelldata)
    NewC = NewTCell()
    Copy TCelldata into NewC
    InsertTCell(NewC, Root)
endfunc
```

Code 7.14. Functions to execute for each FBody or TCell in the incoming data stream.

```
InsertBody(Body, Cell)
    if( Cell is type ICell )
        InsertBodyICell( Body, Cell )
    else if( Cell is type LCell )
        InsertBodyLCell( Body, Cell )
    else if( Cell is type TCell )
        InsertBodyTCell( Body, Cell )
    endif
endfunc
```

Code 7.15. Function to execute for every FBody received.

```
InsertBodyICell(Body, ICell)
    ClearBit( MpoleCorrect, ICell )
    if(ICell has a Child which contains Body)
        InsertBodyCell( Body, Child )
    else
        Child = NewLCell()
        make Child a child of ICell, correctly sized
            and positioned to contain Body.
        InsertBodyLCell( Body, New )
    endif
endfunc
```

Code 7.16. Function `InsertBodyICell` to insert a Body into an ICell.

```
InsertBodyTCell( Body, TCell )
    if( IsSet(TransmissionPending, TCell) )
        New = NewTCell()
        ReplaceInTree( New, TCell )
    else
        New = TCell
    endif

    BodyMpole(Body, New)
endfunc
```

Code 7.17. Function `InsertBodyTCell` to insert a Body into a TCell.

```
InsertBodyLCell( Body, LCell )
    if( IsSet(TransmissionPending, LCell) )
        New = NewLCell()
        ReplaceInTree(New, LCell)
    else
        New = LCell
    endif
    if( New has fewer than m Bodies )
        add LCell directly to the list of bodies in New
    else
        Newer = NewICell()
        ReplaceInTree(Newer, New)
    endif
endfunc
```

Code 7.18. Function `InsertBodyLCell` to insert a `Body` into an `LCell`.

```
InsertTCell(NewCell, Cell)
    Parent = FindParent(NewCell, Cell)
    if( Parent already has a Child at the location of NewCell )
        if( Child is type TCell )
            ParAxis(NewCell, Child)
        else
            child must be type LCell.
            for( each OldBody in LCell )
                BodyMpole(OldBody, Cell)
            endfor
        endif
        ZapCell(Child)
    else
        make NewCell a child of Parent.
    endif
endfunc
```

Code 7.19. Function to execute for every `TCell` that is received.

```
ReplaceInTree( New, Old )
    modify the parent of Old so that New is its child
        in place of Old.
    if( Old is type LCell )
        if( IsSet(TransmissionPending, Old) )
            for( each Body in Old )
                Copy = a copy of Body
                InsertBody( Copy, New )
            endfor
            SetBit(FreeWhenSent, Old)
        else
            for( each Body in Old )
                InsertBody( Body, New )
                disconnect Body from Old
            endfor
            ZapCell(Old)
        endif
    else
        copy multipole information from Old to New.
        ZapCell(Old)
    endif
endfunc
```

Code 7.20. Function `ReplaceInTree` which substitutes a new cell for an old one in the tree, taking care to either remove the old cell, or set the `FreeWhenSent` bit.

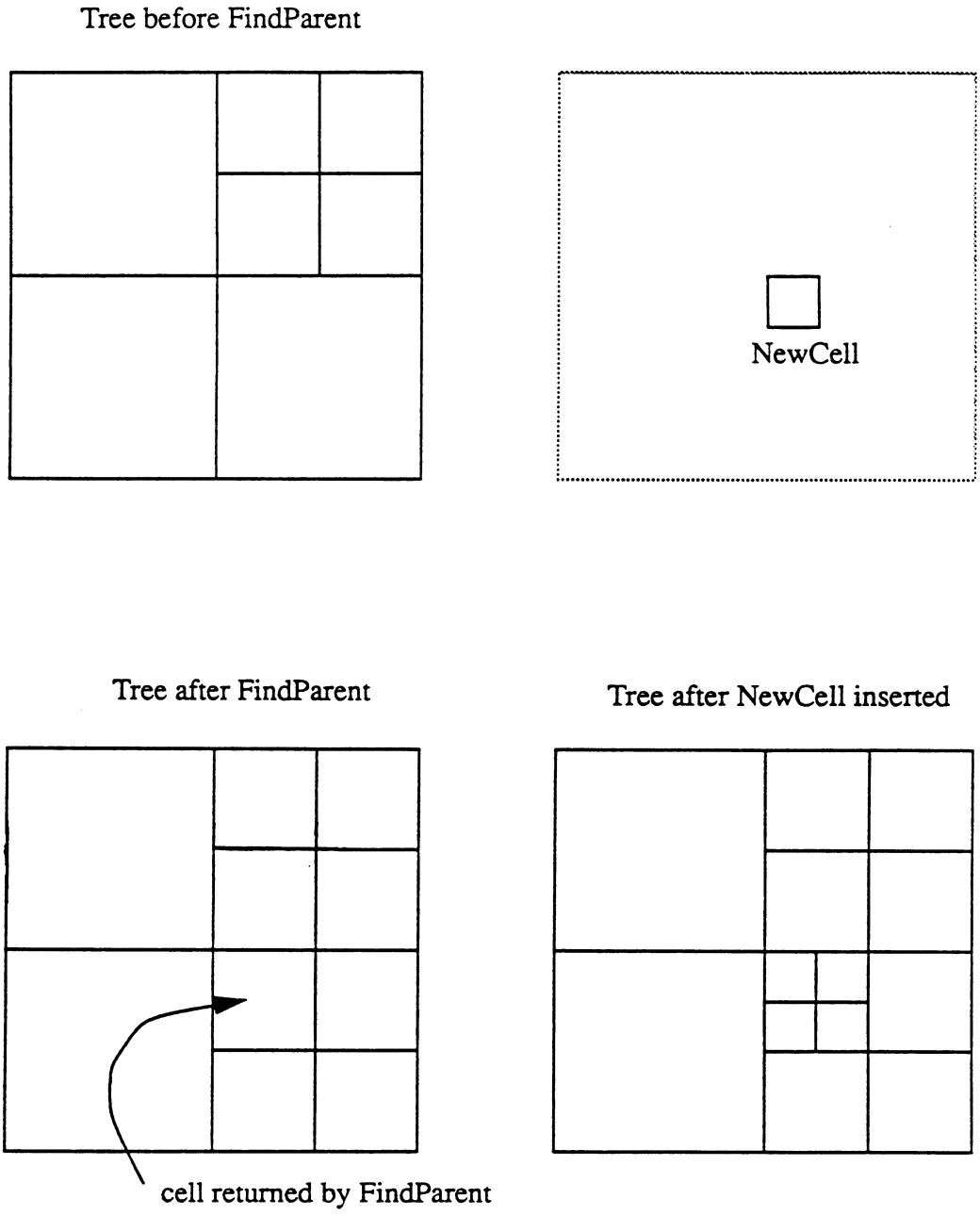


Figure 7.9. `FindParent` must create the parent of `NewCell` if it does not already exist.

```
FindParent( NewCell, Ancestor )
    SetBit(ContainsTCells, Ancestor)
    if( Ancestor is exactly twice as large as NewCell )
        return Ancestor
    else
        if( Ancestor has a Child which contains NewCell )
            if( Child is type ICell )
                Sub = Child
            else
                Sub = Refine(Child);
            endif
        else
            Sub = NewICell()
            make Sub a child of Ancestor containing NewCell
        endif
    endif
    return FindParent( NewCell, New )
endfunc
```

Code 7.21. Function `FindParent` to locate the immediate parent of a received `TCell`, or to create it if it does not exist.

7.3. The DomainOpeningCriterion function.

We are now in a position to understand why the *edge* type opening criteria are, in addition to being numerically superior to the BH criterion, also easier to parallelize.

The performance of the parallel algorithm depends on a fast and accurate implementation of the `DomainOpeningCriterion`. Implementation of a `DomainOpeningCriterion` is complicated by the fact that the complete contents of a `Cell` are not known to the processor which must evaluate the `DomainOpeningCriterion` for the `Cell`. Nevertheless, the `DomainOpeningCriterion` is required to produce an answer which tells us whether the final, complete `Cell` will be openable by test-points in the given domain. This is only possible because the correctness of the algorithm is assured if the `DomainOpeningCriterion` returns "false-positives," but no "false-negatives," i.e., it is acceptable for the the `DomainOpeningCriterion` to tell us that a cell is openable, even though it is not. Such errors lead to unnecessary data communication, and memory usage because the descendants of the falsely openable cell are communicated and stored, but never used. False-negatives, on the other hand, would lead to errors in the field calculation, or at least differences between the serial and parallel program which would be hard to characterize. A falsely negative return from `DomainOpeningCriterion` would assure a processor that a particular cell would never be opened, and would instruct the processor to send only the multipole expansion, but not the detailed contents. During the force evaluation, the processor would try to inspect the descendants of the cell, which would not be available.

Although the final contents of a `Cell` are not known when `DomainOpeningCriterion` is called, the geometry of the cell, i.e., its size and position in space, is known. In fact, the geometry of the cell and the geometry of the domain are all that `DomainOpeningCriterion` can use to determine an answer.

Consider the implementation of `DomainOpeningCriterion` when the `OpeningCriterion` follows the original BH prescription, i.e., $\vec{r}_\gamma$ is the center-of-mass of the cell, distance is measured from the test-point to $\vec{r}_\gamma$ and the size of a cubical

cell is given by the length of an edge. At the time DomainOpeningCriterion is called, it is impossible to know what the final center-of-mass of the cell will be. All that can be assumed is that the center-of-mass will lie somewhere within the cell. Thus, following the rule that false-positives are allowed, but false-negatives are not, the DomainOpeningCriterion must be implemented as in Code 7.22. The geometry is shown in Figure 7.10.

```
DomainOpeningCriterion( Domain, Cell )
    if ( θ× Distance from nearest point in Cell
        to nearest point in Domain < size of cell )
        return TRUE
    else
        return FALSE
    endif
endfunc
```

Code 7.22. DomainOpeningCriterion for the original Barnes-Hut opening criterion.

When used with the BH OpeningCriterion, the DomainOpeningCriterion of Code 7.22 results in a very large number of false-positives. Figure 7.11 shows an example of how a false-positive DOC can arise. In the figure, the distance computed between the cell and the processor domain, as computed by DomainOpeningCriterion is much less than the true distance, which is known only after $\vec{r}_\gamma$ is computed. When the OpeningCriterion is applied to the distance from the domain to the edge of the cell, the test passes, but when it is applied to the true distance, the test fails.

One way to avoid false-positives in the DomainOpeningCriterion is to use an OpeningCriterion which only relies on the geometry of the cell. Fortunately, the *edge*-based OpeningCriteria from Chapter 6 satisfy this property. In Chapter 6, we introduced *edge*-based OpeningCriteria as a means to eliminate a source of systematic error from the algorithm. Interestingly, the DomainOpeningCriterion for the *edge*-based OpeningCriteria is precisely the same as for the BH opening

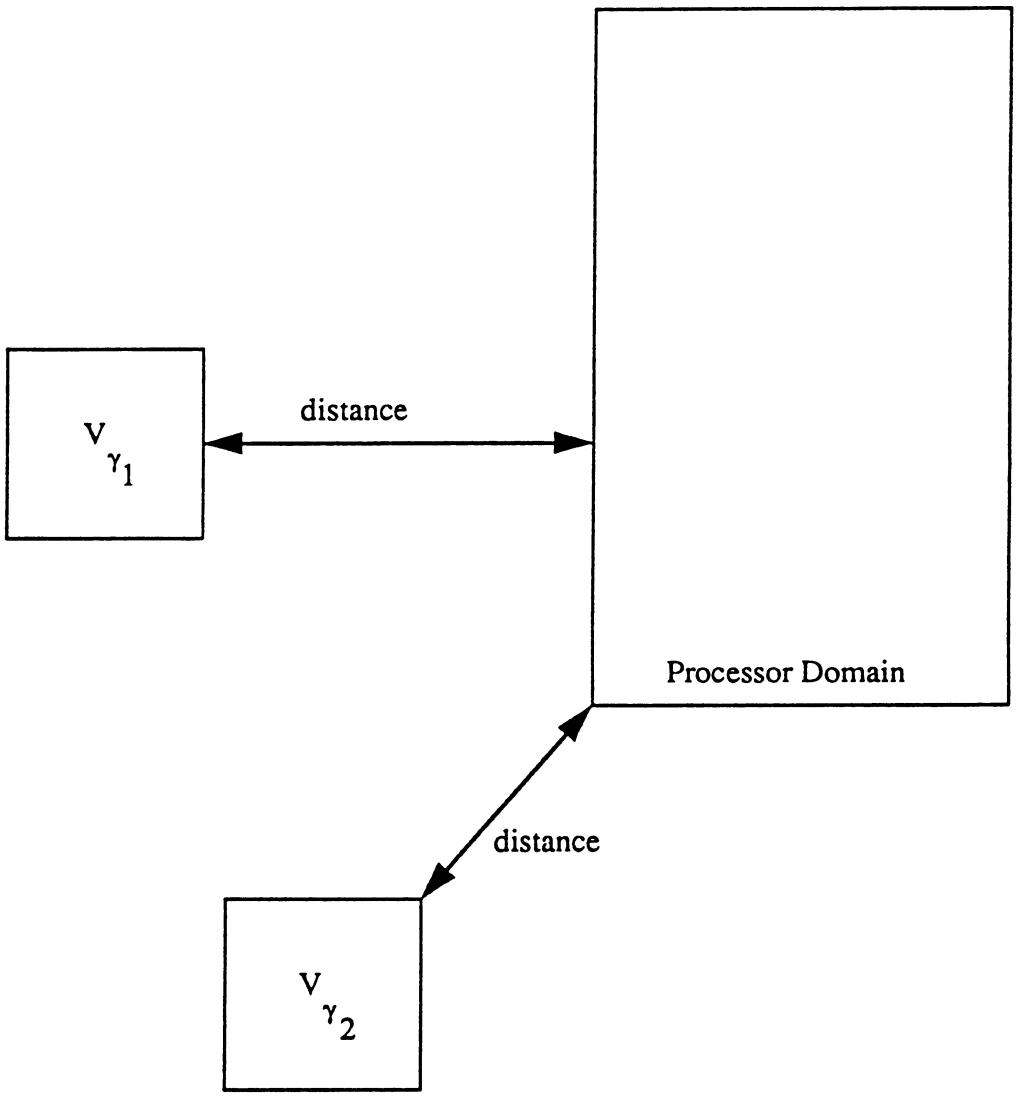


Figure 7.10. The distance used by the `DomainOpeningCriterion` is computed by finding the shortest distance between the processor domain and the boundary of $\mathcal{V}_\gamma$.

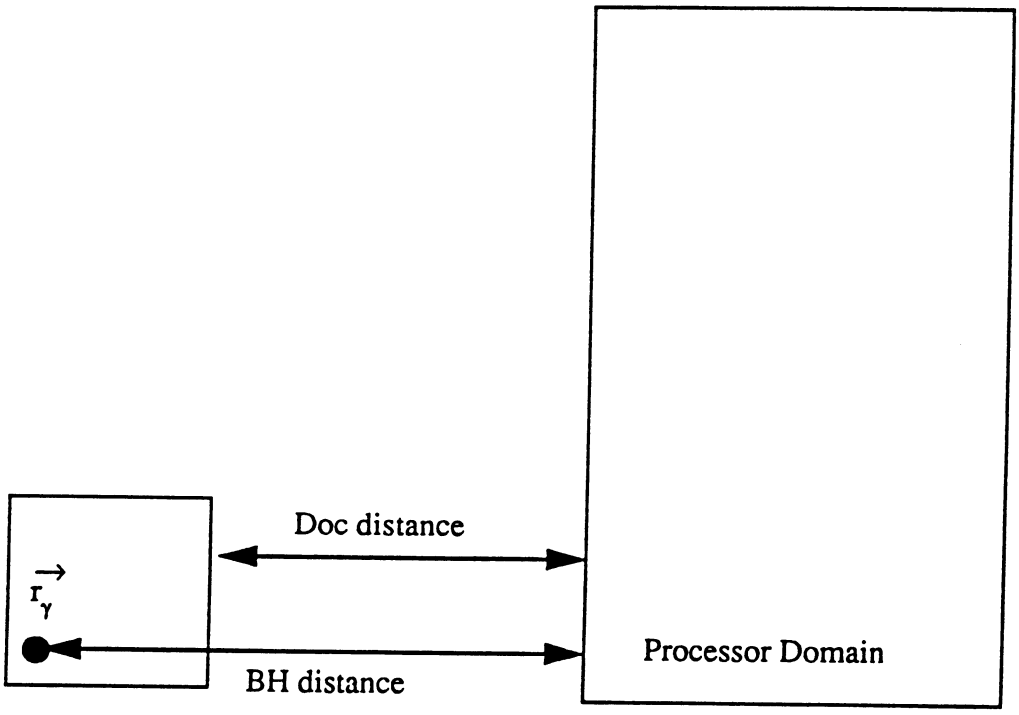


Figure 7.11. The `DomainOpeningCriterion` can return a false positive result when the true distance from the $\vec{r}_\gamma$ to the domain is much greater than the pessimistic assumption made by `DomainOpeningCriterion`.

criterion, i.e., it is exactly as shown in Code 7.22. The distance is still measured from the boundary of the cell to the boundary of the domain. Thus, we have only one `DomainOpeningCriterion`, which is used for either the both types of `OpeningCriteria`.

There are two reasons why the *edge* `OpeningCriteria` perform considerably better in parallel. First, there are no false-positive returns from `DomainOpeningCriterion`. The reason is that the distance used by `DomainOpeningCriterion` is precisely the same as the distance used by the `OpeningCriterion` itself. The discrepancy shown in Figure 7.11 cannot arise when the `OpeningCriterion` measures distances to the edge of the cell. Second, we recall that a given degree of accuracy is obtained with a much larger value of θ when the *edge* `OpeningCriteria` are used, vis-á-vis the BH opening criterion. (See Figures 6.6 and 6.7). Since both methods use the same `DomainOpeningCriterion`, the overhead associated with tree building, storing and computing multipoles, etc., depends only on the value of θ, but is independent of which `OpeningCriterion` is used. Since θ is much larger when *edge* `OpeningCriteria` are used, the tree building overhead is much smaller, and the performance of the parallel implementation is much better, both in terms of memory and cpu performance.

8. Performance in Parallel.

It is possible to evaluate numerous measures of the performance. Among them are "efficiency," "speedup," "wall clock time," "largest practical problem." All of these may be related to absolute problem size, "grain size," number of processors, and a multi-dimensional space of "tuning parameters," e.g., θ, and hardware parameters, e.g., communication bandwidth. All this makes it almost impossible to produce an entirely satisfactory analysis of parallel performance.

The timing data reported here was obtained from the 512 processor NCube system at Caltech. The NCube was used because it had the largest number of processors of any readily available distributed memory machine. Good performance on smaller numbers of processors is considerably easier to obtain. It was our experience while developing the parallel algorithm that testing an algorithm on 8 or even 64 processors often failed to illustrate the difficulties that will be encountered when running on highly parallel systems with several hundred processors. It is hoped that the lessons learned from the experience with the 512 processor NCube prepare us for systems with another factor of ten more processors and perhaps a factor of one thousand in total performance. The total system performance of the NCube barely qualifies it as a supercomputer, but the large number of processors allows us to extrapolate performance to the large parallel supercomputers that will be available in the very near future.

8.1. Definition of terms.

Before studying the details of the performance characteristics of the parallel N-body program, we devote a section to definitions. The quantities defined in this section will be analyzed in the subsequent sections. The performance measures that we define in this section are not at all specific to the particular program

we are studying. They can be used systematically to investigate the performance of any loosely synchronous parallel program. Several of these quantities, most notably "speedup" and to a lesser extent "efficiency," have been reported so widely as to be almost cliches. In the following sections, we develop a consistent set of quantities which, when measured for a variety of different configurations of N_{proc} and problem size, can provide a very detailed understanding of a parallel program.

8.1.1. Computational overheads.

First, we will assume that we are capable of measuring elapsed time on a per-processor basis. That is, every processor in an ensemble may access a timer without interfering with, or interacting with any other processor in the ensemble. Whether or not there is a true, global system clock is irrelevant for our purposes. All we are concerned with is the ability to measure elapsed time, not absolute time. Accordingly, any measurement of time is recorded in an array of N_{proc} values, one for each processor. Such per-processor times will be denoted with the lower-case letter, t. In the following, we will consider measurements of the elapsed time for a number of different operations. These times will depend on the number of bodies, N, and their exact locations, as well as the number of processors, N_{proc}, and system parameters like θ, etc. We have used three fundamentally different simulations, and averaged the values over several (usually 10) timesteps to arrive at the values reported in the following sections.

Our N-body simulation is typical of large-scale scientific calculations in that a full calculation consists of a loop over a large number of basic calculations. In our case, the basic calculation is to compute the forces on each body in the system, and to advance the positions and velocities of each body in the system by one unit of simulated time. Let t_{step} be the elapsed time required for a single step in a simulation. The existence of a basic unit of calculation is crucial to the development in the remainder of this section. In any large calculation, as long as one can define an appropriate t_{step}, then all of the other quantities defined below may be computed as well. In other situations, t_{step} may be the time required to

perform a matrix inversion, a single cycle of relaxation, an FFT, etc. In any case, it is the time required for the basic unit of calculation in the algorithm.

The maximum value of t_{step}, over the ensemble of processors, is of primary importance. It is equal to the time required for the entire ensemble of processors to carry out the basic computation. In our case, it is the time required for the entire ensemble to advance the positions and velocities by one timestep. We define

$$T_{step} = \max(t_{step}). \tag{8.1}$$

In Eqn. 8.1, T_{step} is a single number, obtained from the array t_{step}. In the following, quantities defined with an uppercase T will be single numbers representing some statistic, e.g., *max*, *average*, applied to the array, t. We shall use the statistic $slow(t_x)$, which is defined to be the value of t_x on the processor with the largest value of t_{step}, i.e., the slowest overall processor. Often, $slow(t_x)$ is the same as $\max(t_x)$, because the slowest processor happens to have the largest value of t_x, but the two statistics are defined quite differently.

It is useful to separate the time, t_{step}, into component pieces,

$$t_{step} = t_{work} + t_{comm} + t_{wait}, \tag{8.2}$$

where

t_{comm} is the portion of t_{step} devoted to transferring data in or out of each processor.

t_{wait} is the portion of t_{step} spent idle waiting for some type of interprocessor synchronization to take place. Since we are using a synchronous communication system, waiting occurs whenever one processor is ready to send/receive data, but its neighbor is not yet ready. Other communication systems allow processing to continue, for example after data has been queued for transmission. When such systems are used, waiting still occurs when a processor waits for data to be received, or at explicit synchronization points, which are often necessary when such communication systems are used.

t_{work} is the remainder of t_{step}, presumably devoted to some form of computation.

With these definitions, we may also define

$$T_{step} = T_{imbal} + T_{work} + T_{comm} + T_{wait}, \tag{8.3}$$

where

$$\begin{aligned} T_{imbal} &= slow(t_{work} + t_{comm}) - avg(t_{work} + t_{comm}), \\ T_{work} &= avg(t_{work}), \\ T_{comm} &= avg(t_{comm}), \\ T_{wait} &= slow(t_{wait}). \end{aligned} \tag{8.4}$$

We may further subdivide the load imbalance into components related to communication, waiting and work,

$$T_{imbal} = T_{iwork} + T_{icomm}, \tag{8.5}$$

where

$$\begin{aligned} T_{iwork} &= slow(t_{work}) - avg(t_{work}), \\ T_{icomm} &= slow(t_{comm}) - avg(t_{comm}). \end{aligned} \tag{8.6}$$

Note that T_{wait} is defined using the *slow* statistic, rather than the *avg* statistic, so there is no associated "waiting load imbalance." The reason is that there is little to be gained from separating the waiting time into "average" and "imbalanced" components. Synchronization delays represent a form of "microscopic" load imbalance anyway. It is not useful to consider the "average microscopic load imbalance" separately from the "imbalance of the microscopic load imbalance."

All of the quantities so far defined are directly measurable by inserting suitable calls to a time-reporting function into a program. Statistics like *max*, *avg* and *slow* are easily computed in parallel using utility functions which perform global aggregation over the ensemble of processors. Some software systems, e.g., ExPress,[43] provide utilities, e.g., `excombine`, which facilitate these aggregations, but they are simple enough to construct even if the native software does not provide them. We point out that it is not particularly "efficient" to compute an

average of N_{proc} values on an N_{proc} processor system. Nevertheless, the time is negligible, measured in msec, at most, and the diagnostic information is almost certainly worth the expense. As long as the overhead associated with invoking the timer is small compared to the times in question, we have a perfectly adequate method for obtaining the various T_{xxx} quantities.

Finally, we divide T_{work} into components that distinguish between the intrinsic work, $T_{intrinsic}$, associated with calculation and the additional time, T_{cplx}, which was necessary to process the data in parallel because of additional complexity in the parallel algorithm,

$$T_{work} = T_{cplx} + T_{intrinsic}. \tag{8.7}$$

In Eqn. 8.7, $T_{intrinsic}$ is related to the time, $T_{oneproc}$, an optimal single processor implementation of the algorithm would require to execute the same basic computational unit,

$$T_{intrinsic} = \frac{T_{oneproc}}{N_{proc}}. \tag{8.8}$$

Evaluation of $T_{oneproc}$ may be problematical. Distributed memory parallel computers usually have limited per-processor memory. Almost certainly, a simulation which utilizes the full resources of the entire ensemble will simply not fit within a single processor. Current designs of MIMD parallel processors do not allow data sets to exceed the size of available dynamic memory, i.e., there is no virtual memory, and even if there were, a timing made on a single processor "thrashing" in virtual memory would not accurately reflect $T_{oneproc}$ anyway. Thus, there is no direct way to measure $T_{oneproc}$, except for relatively small simulations. In order to obtain values for $T_{oneproc}$ for large simulations, we will need to extrapolate from smaller simulations which fit on a single processor.

Once $T_{oneproc}$ is known, a host of other other statistics may be defined. First, the speedup, S, of the implementation is defined by

$$S = \frac{T_{oneproc}}{T_{step}}. \tag{8.9}$$

The speedup is the ratio of the time a single processor would spend on a step to the time that the ensemble would spend. One measure of the effectiveness of a parallel program is how close its speedup is to the number of processors employed. Thus, we are led to the "efficiency," ϵ,

$$\epsilon = \frac{S}{N_{proc}} = \frac{T_{oneproc}}{N_{proc} T_{step}} = \frac{T_{intrinsic}}{T_{step}}. \tag{8.10}$$

Although useful, the efficiency lacks simple algebraic properties. We cannot say "the efficiency is the sum of X and Y and Z." A quantity related to the efficiency, but with superior algebraic properties is the "overhead," f, where

$$f = \frac{1}{\epsilon} - 1 = \frac{T_{step} - T_{intrinsic}}{T_{intrinsic}}. \tag{8.11}$$

Using Eqns. 8.3 and 8.7, we can define

$$T_{ohead} = T_{step} - T_{intrinsic} = T_{imbal} + T_{cplx} + T_{comm} + T_{wait}, \tag{8.12}$$

and thus, we have

$$f = f_{imbal} + f_{cplx} + f_{comm} + f_{wait}, \tag{8.13}$$

where

$$\begin{aligned} f_{imbal} &= \frac{T_{imbal}}{T_{intrinsic}}, \\ f_{cplx} &= \frac{T_{cplx}}{T_{intrinsic}}, \\ f_{comm} &= \frac{T_{comm}}{T_{intrinsic}}, \\ f_{wait} &= \frac{T_{wait}}{T_{intrinsic}}. \end{aligned} \tag{8.14}$$

We can also define f_{iwork} and f_{icomm} as

$$\begin{aligned} f_{iwork} &= \frac{T_{iwork}}{T_{intrinsic}}, \\ f_{icomm} &= \frac{T_{icomm}}{T_{intrinsic}}, \end{aligned} \tag{8.15}$$

and, from Eqn. 8.5, we have

$$f_{imbal} = f_{iwork} + f_{icomm}. \tag{8.16}$$

Equations 8.13, 8.14, 8.15 and 8.16 are precisely the algebraic relations that are lacking for the efficiency. We have defined a single number, the total overhead, f, which measures the effectiveness of a parallel program. In addition, we can say that f is identically equal to a sum of components, each readily identified with a particular aspect of the parallel implementation. If we understand the components of the overhead of an algorithm, i.e., the dependence on N and N_{proc}, and the reason for it, we are well on our way to fully understanding the parallel algorithm, and can predict how it will perform in a situation with, for example, more or faster processors, faster communication or a larger data set.

8.1.2. Memory overhead.

Often, memory considerations are completely overlooked when considering the performance of a parallel program. Nevertheless, effective utilization of memory resources may be as important as effective utilization of processing resources. Just as one asks the question, "Given N_{proc} processors, each capable of x operations per second, how large a system can I simulate?" One can also ask, "Given N_{proc} processors, each with y words of storage, how large a system can I simulate?" This is especially true of algorithms with $O(N \log N)$ or better performance. Such algorithms may be applied to extremely large data sets in reasonable time, making available memory, rather than cpu cycles, the limiting resource. Just as we defined overheads for processor utilization, above, we now define the overhead associated with memory utilization.

First, we define m_{step} to be the memory requirement on each processor. Memory is naturally divided into that used for code and that used for data. On a MIMD processor, the code is replicated in each and every processor. Thus, one of the hidden costs of using very many processors is the "wasted" memory that results from many identical copies of the same executable code. It is convenient to separate

this easily analyzed source of memory overhead from those specific to a particular algorithm. Thus, we write

$$m_{step} = d_{step} + c, \tag{8.17}$$

where c is the memory required on each processor by the executable code, and d_{step} is that required for the data.

Then, we define M_{step} to be the largest memory requirement amongst all of the processors,

$$M_{step} = max(m_{step}). \tag{8.18}$$

If we make the conservative assumption that all processors in the parallel ensemble have the same amount of available memory, and that it cannot be shared or repartitioned dynamically, then M_{step} is the amount of per-processor memory required to run the simulation for a timestep. If any less memory were available on each processor, then at least one of the processors would run out of memory.

We now define a "one-processor" memory requirement, $M_{oneproc}$, which is the memory used by a single processor implementation of the algorithm. Just as above, we split the one-processor memory into that required for the executable code, $C_{oneproc}$ and that required for the data, $D_{oneproc}$,

$$M_{oneproc} = D_{oneproc} + C_{oneproc}. \tag{8.19}$$

If the parallel algorithm were perfectly efficient, memory-wise, it would require no more memory than the single processor implementation, but that memory would be divided amongst N_{proc} processors. Thus, we define the intrinsic memory requirement, by analogy with $T_{intrinsic}$ as

$$M_{intrinsic} = \frac{M_{oneproc}}{N_{proc}}, \tag{8.20}$$

and the memory overhead, M_{ohead}, as

$$M_{ohead} = M_{step} - M_{intrinsic}. \tag{8.21}$$

Just as with the computational overhead, T_{ohead}, the total memory overhead can be separated into pieces whose sum is equal to M_{ohead}. Thus,

$$M_{ohead} = M_{code} + M_{copy} + M_{imbal}, \tag{8.22}$$

where

$$\begin{aligned} M_{code} &= C - \frac{C_{oneproc}}{N_{proc}}, \\ M_{copy} &= avg(d_{step}) - D_{intrinsic}, \\ M_{imbal} &= max(d_{step}) - avg(d_{step}). \end{aligned} \tag{8.23}$$

Now, we can define dimensionless overheads simply by dividing each of the quantities in Eqn. 8.23 by an appropriate single-processor quantity. Because of its extremely simple dependence on N in the BH algorithm, we elect to use $D_{intrinsic}$ as the "unit of memory" by which we define the various memory overheads. Thus, we have the following dimensionless memory overheads, g_{xxx},

$$\begin{aligned} g_{code} &= \frac{M_{code}}{D_{intrinsic}}, \\ g_{copy} &= \frac{M_{copy}}{D_{intrinsic}}, \\ g_{imbal} &= \frac{M_{imbal}}{D_{intrinsic}}. \end{aligned} \tag{8.24}$$

Thus, we have quantitative measures of how "efficiently" we are using memory in the parallel system, completely analogous to our measures of how "efficiently" we are using cpu cycles.

8.2. Performance data for the N-body program.

Now that the necessary definitions are out of the way, we can investigate the sources of overhead in the parallel N-body code. All of the plots in this section are derived from a set of timing runs in which three distinct "models types" were constructed with varying numbers of particles. The model types were:

1. Particles distributed uniformly, at random, throughout a sphere of unit radius. This distribution is expected to perform well because load imbalance should be minimal. It is typical of the initial stages of astrophysical simulations of cosmological density perturbations. In such simulations, small fluctuations grow via gravitational instability. Usually the perturbations start out quite small, and from a purely computational viewpoint, the particles are almost uniformly distributed. Uniform model data are all connected by dashed lines in the following figures.
2. Particles distributed at random, according to a "spherical Jaffe profile," [35] cut off at $r = \Lambda r_0$ in which the number density of particles inside radius Λr_0 is

$$\rho(r) = \frac{N}{4\pi r_0^3} \frac{\Lambda + 1}{\Lambda} \frac{r_0^4}{r^2 (r + r_0)^2}, \tag{8.25}$$

   where r_0 is the half-mass radius. The cutoff was at $\Lambda = 10$. The Jaffe profile ($\Lambda = \infty$) is an analytic density profile that resembles that of real elliptical galaxies. The very sharp density profile at the center of the Jaffe models leads to difficulties with load balance. This model is a "worst case," since the density is very sharply peaked in one place, and relatively smooth elsewhere. It is typical of astrophysical simulations which study the dynamics of single galaxies. Jaffe model data are all connected by solid lines in the following plots.
3. Particles chosen at random from a "real" simulation, consisting of two interacting galaxies. The full simulation had 180000 bodies, in two Jaffe galaxies in orbit around one another. This model is typical of astrophysical simulations of galaxy interactions. Merger model data are all connected by dotted lines in the following plots.

All of the timing runs were carried out on the Ncube/10 at Caltech. Each processor of this system has 512 kbytes of memory and a proprietary custom cpu with builtin communication channels. The simulations were run using the *edge*-distance opening criterion with a Cartesian metric, and $\theta_{edge,Cart} = 1.4$. Each

benchmark run lasted from between 7 to 10 timesteps. The various t_{xxx} were measured directly on each processor using the Ncube system call `ntime`. The t_{xxx} values were aggregated into T_{xxx} values at the end of each timestep, and the values were recorded. Finally, the recorded values were averaged, excluding the first two timesteps to eliminate "transients" associated with reading the data file and the poor decomposition on the first two timesteps, which are carried out without knowledge about the work associated with each particle. The resulting averages are displayed in the plots in the following sections.

8.2.1. Time for a single timestep, T_{step}.

We begin with the "bottom line." Figure 8.1 shows T_{step} plotted against the number of processors, for various data sets. Figure 8.1 provides direct information on how long a simulation will actually take on a particular hardware configuration. If one is simply interested in the speed of a simulation, and not the underlying nature of the algorithm, or the parallelism achieved, Figure 8.1 contains the relevant data. Clearly, the T_{step} is decreasing as more processors are added, up to 512 processors, but many of the data sets have reached a point of diminishing returns. If the parallel implementation were "perfect," then each factor of two in number of processors would correspond to a factor of one half in T_{step}. Clearly, this is not the case and it is important to understand the nature of the overheads that prevent perfect speedup.

8.2.2. Estimating $T_{oneproc}$ and $T_{intrinsic}$.

In order to estimate how efficiently we are using a parallel machine, and to compute the various overheads and speedups, we need to estimate $T_{oneproc}$ and $T_{intrinsic}$. As we discussed in Section 8.1, it is not possible to simply run the program on a single processor for any but the smallest data sets. Thus, we must estimate $T_{oneproc}$ by extrapolating results from a few relatively small simulations. One way to do this would be to develop an accurate model of $T_{oneproc}$ as a function of N. Unfortunately, the accessible range of N on a single processor only extends up to about 2000 bodies. Over this range, it is difficult to accurately distinguish

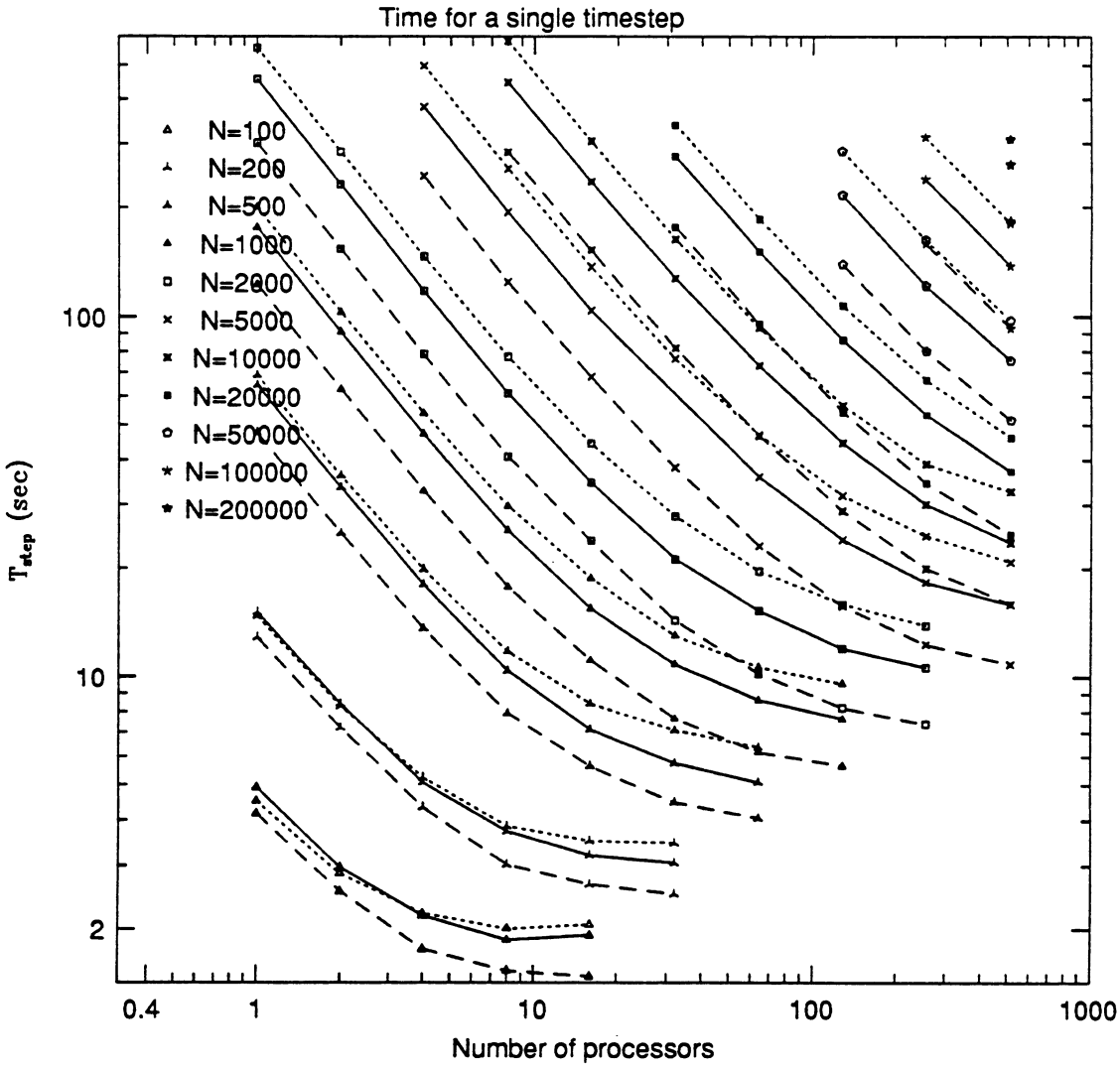


Figure 8.1. T_{step} vs. Number of processors.

terms proportional to $N \log N$ from those proportional to small powers of N. An accurate extrapolation to several hundred thousand bodies would be difficult.

If we could find a subset of the basic computational unit which

1. is independent of the number of processors, and
2. provides an accurate estimate of $T_{oneproc}$,

then we could use the time for that subset to estimate $T_{oneproc}$ for larger data sets.

In the N-body program, the time spent evaluating forces after the tree is constructed very nearly satisfies these conditions. Inspection of the code reveals that in an ensemble of processors, the force on each body is evaluated in exactly one processor. By design, the interactions that each body encounters in a single force calculation are exactly identical to those encountered in a sequential implementation. Furthermore, there is absolutely no synchronization or communication during the entire force calculation phase, in contrast to the tree construction phase. Thus, the time spent evaluating forces, after the tree is constructed, satisfies the first condition. We demonstrate this empirically in Figure 8.2, which shows $\sum_{proc} t_{force}$ plotted against the number of bodies in the simulation. Notice how the graphs with different numbers of processors lie almost exactly on top of one another, verifying condition 1.

It is interesting that the time spent in force evaluation, for a fixed number of particles, is a decreasing function of the number of processors. That is, more processors appear to do slightly less work than a single processor. The reason is that in a multi-processor system, many of the interactions are between `Bodies` and `MTCells`. In contrast, on a single processor, the equivalent interactions are between `Bodies` and `ICells`. Each interaction with an `ICell` implies that the `OpeningCriterion` function returned `False`, while there was no need to execute `OpeningCriterion` at all in the parallel case when an `MTCell` was encountered during tree traversal. The effect is small but noticeable, and suggests that there may be sequential methods which reduce the number of calls to `OpeningCriterion`. In fact, Warren[44] has devised a version of the algorithm that completely

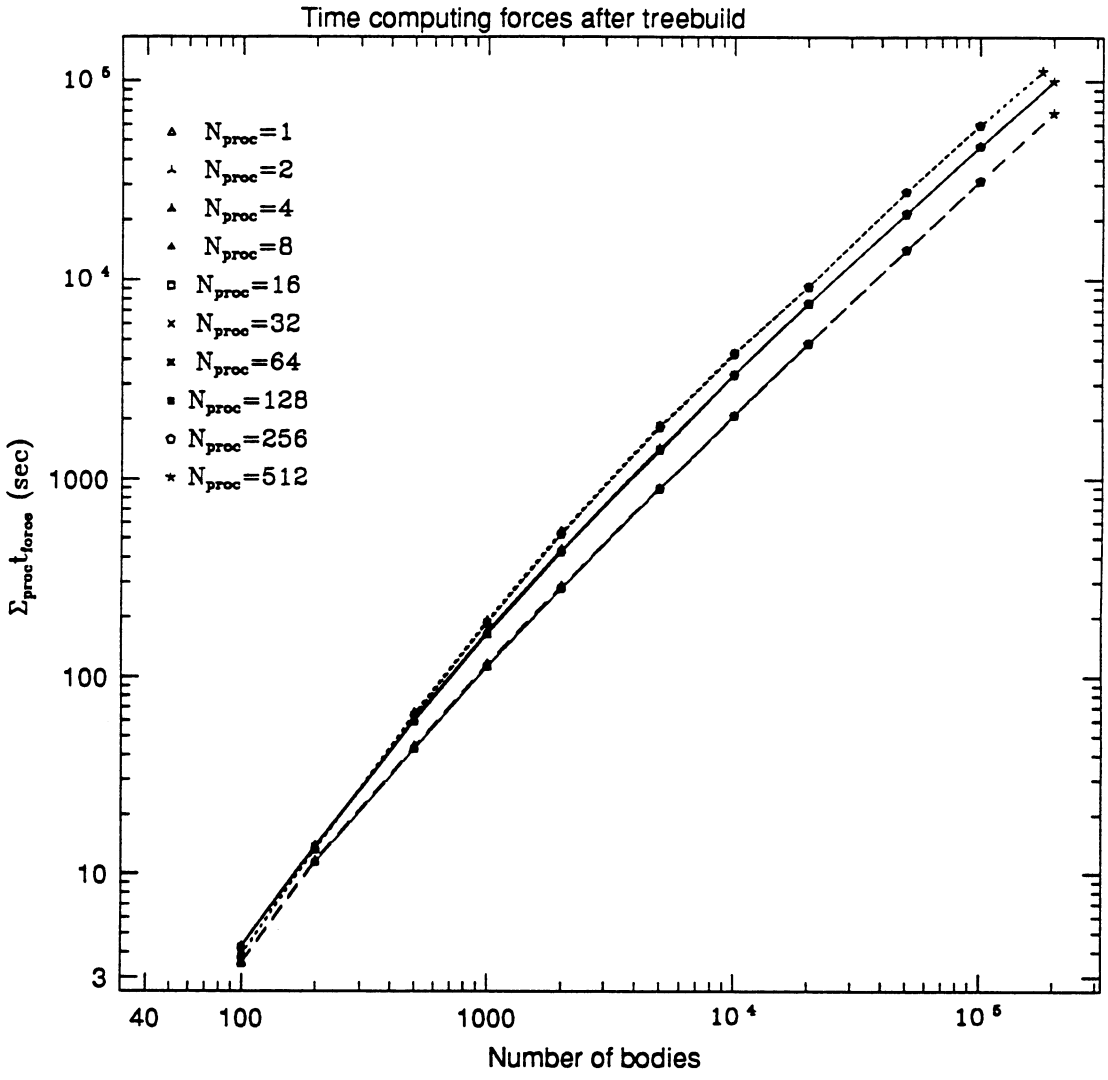


Figure 8.2. T_{force} vs. N.

eliminates explicit traversal of the tree, in favor of hash table lookup, and hence, eliminates all calls to `OpeningCriterion`, and Barnes[22] has described a similar mechanism which he uses to allow vectorization of the tree traversal on vector supercomputers. The improvement in overall performance in our situation is likely to be small, but the approach may have other advantages.

A "parallelism invariant" subset of the program does us no good unless we can also use it to estimate $T_{oneproc}$. In fact, we find that to within the accuracy of our time measurements, we have the following relationship,

$$T_{oneproc} = T_{force}|_{N_{proc}=1} + cN, \tag{8.26}$$

where c depends on the type of simulation,

$$\begin{aligned} c &= 0.0060\,\text{sec} \quad && \text{Jaffe sphere,} \\ c &= 0.0059\,\text{sec} && \text{Uniform sphere,} \\ c &= 0.0063\,\text{sec} && \text{Merger data.} \end{aligned} \tag{8.27}$$

Figure 8.2 and Eqn. 8.26 allow us to estimate $T_{oneproc}$ as

$$T_{oneproc} = \sum_{processors} t_{force} + cN \tag{8.28}$$

and

$$T_{intrinsic} = avg(t_{force}) + cN_{grain}, \tag{8.29}$$

where we have defined the "grain-size," N_{grain}, as

$$N_{grain} = \frac{N}{N_{proc}}. \tag{8.30}$$

8.2.3. Speedup and overhead.

Now that we have a good estimate of $T_{intrinsic}$, we can compute speedups, efficiencies, etc. We begin with Figure 8.3, which shows the speedup achieved by the parallel machine, plotted against the number of processors. The curves in Figure 8.3 are typical of parallel programs on MIMD processors. The speedup is approximately linear with the number of processors for a while, but it begins to flatten out at some point beyond which adding additional processors does not appreciably improve performance. On the other hand, as the problem size is increased, speedup typically increases. In other words, the point at which the curves in Figure 8.3 flatten out is a function of problem size, with larger problems admitting of greater parallelism. This behavior is seen again and again in parallel applications.[45, 46] In fact, this observation is so common, that it leads us to plot grain size, defined in Eqn. 8.30, on the horizontal axis in a number of subsequent plots. It is often found that overhead (or equivalently, efficiency or speedup) depends on N and N_{proc} only through the single parameter, N_{grain}.[6] As we shall see, things are not so simple for us, but the idea of studying how various components of the overhead depend on grain size remains an important one.

Efficiency is closely related to speedup. In fact, according to Eqn. 8.11, there is no new information in Figure 8.4, which displays overhead plotted against grain size. It is clear from Figure 8.4 that the overhead is a fairly rapidly decreasing function of grain-size, and a fairly slowly growing function of N_{proc}. The largest simulations practical on the 512 processor suffer from overheads below 35% which is quite acceptable, corresponding to a speedup in excess of 380. Nevertheless, it is of interest to know the sources of the overhead, and to see if it can be substantially reduced, or if the different categories of overhead all have the same general dependence on increasing grain-size and processor number. We shall see that that in the important regime of large numbers of bodies and large numbers of processors, all the overhead is mostly due to f_{wait} and f_{cplx}. The other components, i.e., f_{comm}

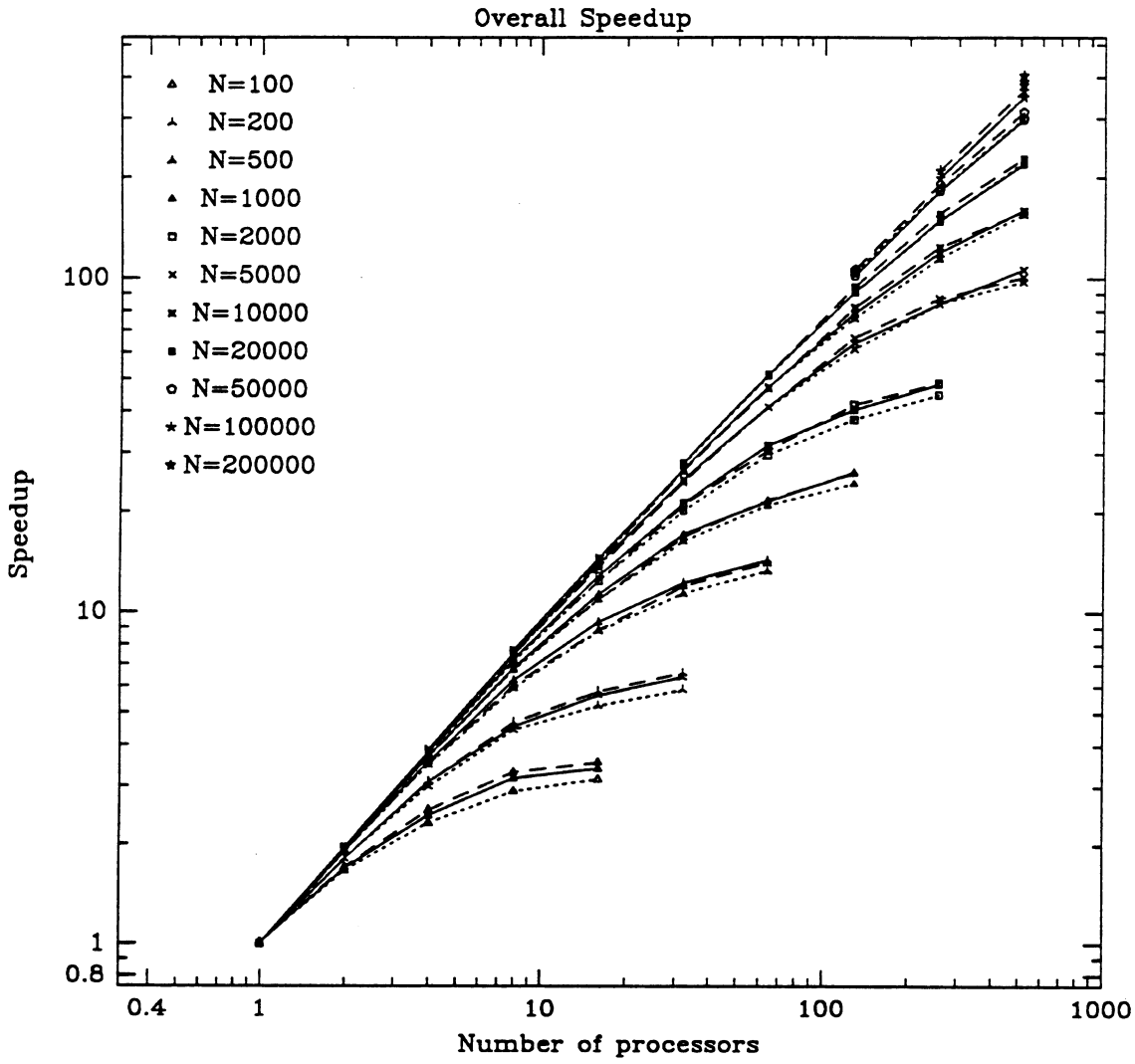


Figure 8.3. Speedup vs. Number of processors.

and f_{imbal}, are negligible.

Furthermore, all of the components of the overhead obey the same general trend with increasing grain-size and processor number. There is no indication that with substantially increased N_{proc} that the overhead will be much worse than at $N_{proc} = 512$, and there is considerable evidence that with increasing grain size (as is available on more modern MIMD processors) that all types of overhead will be considerably lower than in Figure 8.4.

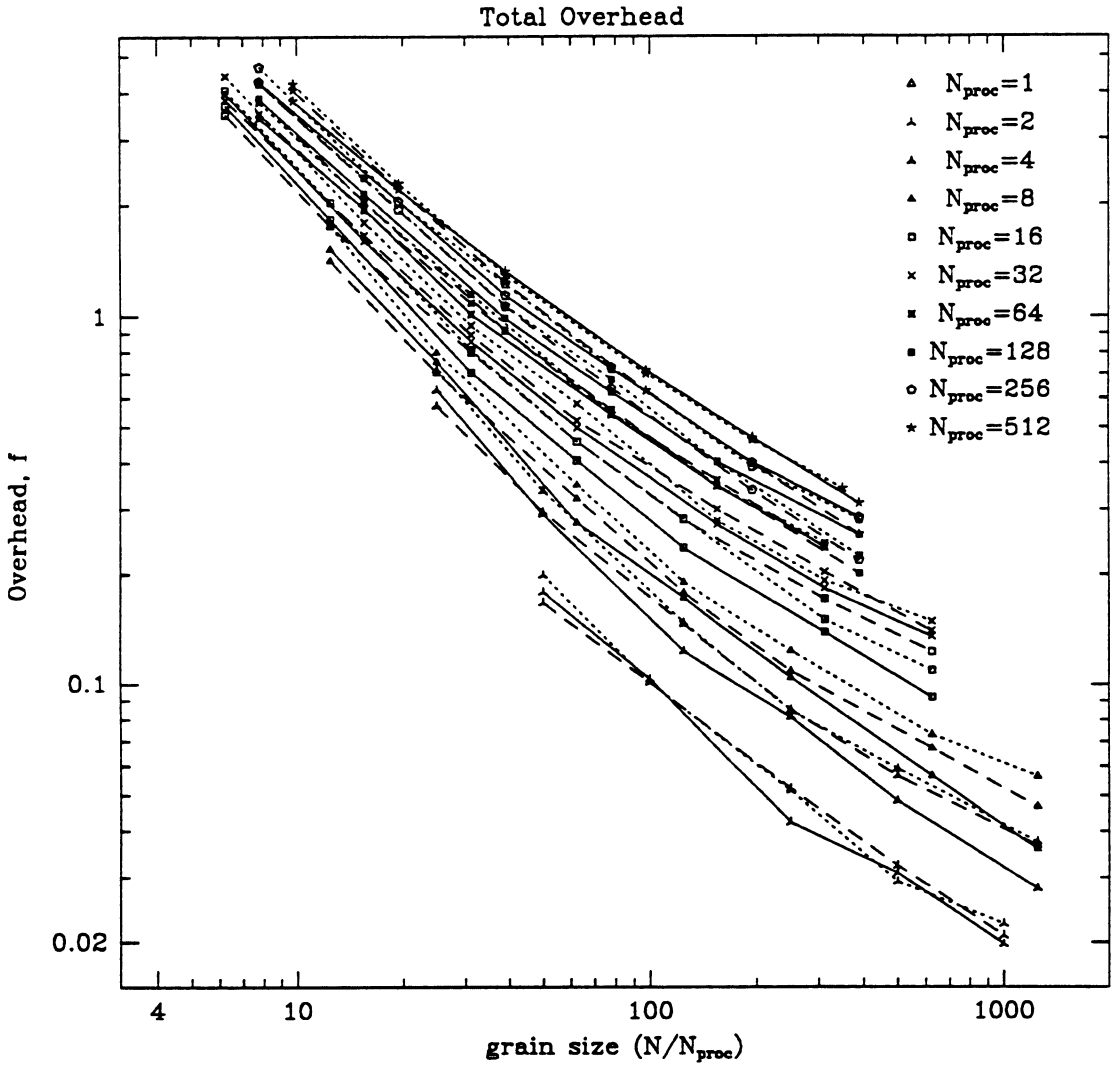


Figure 8.4. Total overhead vs. Grain size.

8.2.4. Single-processor data requirement, $D_{oneproc}$.

Before we turn to the individual components of the overhead, we investigate the memory overhead. As we shall see, the computational overhead is closely related to the memory overhead, and it will be much easier to understand the computational overheads after the memory overheads have been discussed.

The one-processor memory requirement, $D_{oneproc}$, is considerably easier to estimate than $T_{oneproc}$. In the case of memory, we have a very good theoretical as well as empirical basis for asserting that

$$\begin{aligned} D_{oneproc} &= 17.5N \quad \text{words}, \\ D_{intrinsic} &= 17.5N_{grain} \quad \text{words}. \end{aligned} \tag{8.31}$$

This results from the fact that in a single-processor implementation, each `Body` requires 8 words (1 word for the mass, 3 each for position and velocity and one more to identify the object as a `Body`), and each `Cell` requires 19 words (1 word for the mass, 3 for $\vec{r}_\gamma$, 6 for the quadrupole moment, 8 for pointers to children and one more to identify the object as a `Cell` and hold the various boolean flags, `FreeWhenSent`, etc.). We find in Chapter 2 that the number of `Cells` in a BH tree containing N bodies is expected to be

$$N_{cells} = c_1 N \approx 0.5N. \tag{8.32}$$

Thus, the total number of words, for both the `Bodies` and the `Cells` is as in Eqn. 8.31. This relationship is confirmed by the single-processor simulations, which were limited to $N < 5000$ because of memory constraints.

8.2.5. Memory overheads, M_{copy} and M_{imbal}.

With $D_{intrinsic}$ established from Eqn. 8.31, it is a simple matter to compute g_{copy} from recorded values of $avg(d_{step})$. The result is plotted in Figure 8.5. It is also easy to compute g_{imbal} from recorded values of $avg(d_{step})$ and $\max(d_{step})$. The result is shown in Figure 8.6.

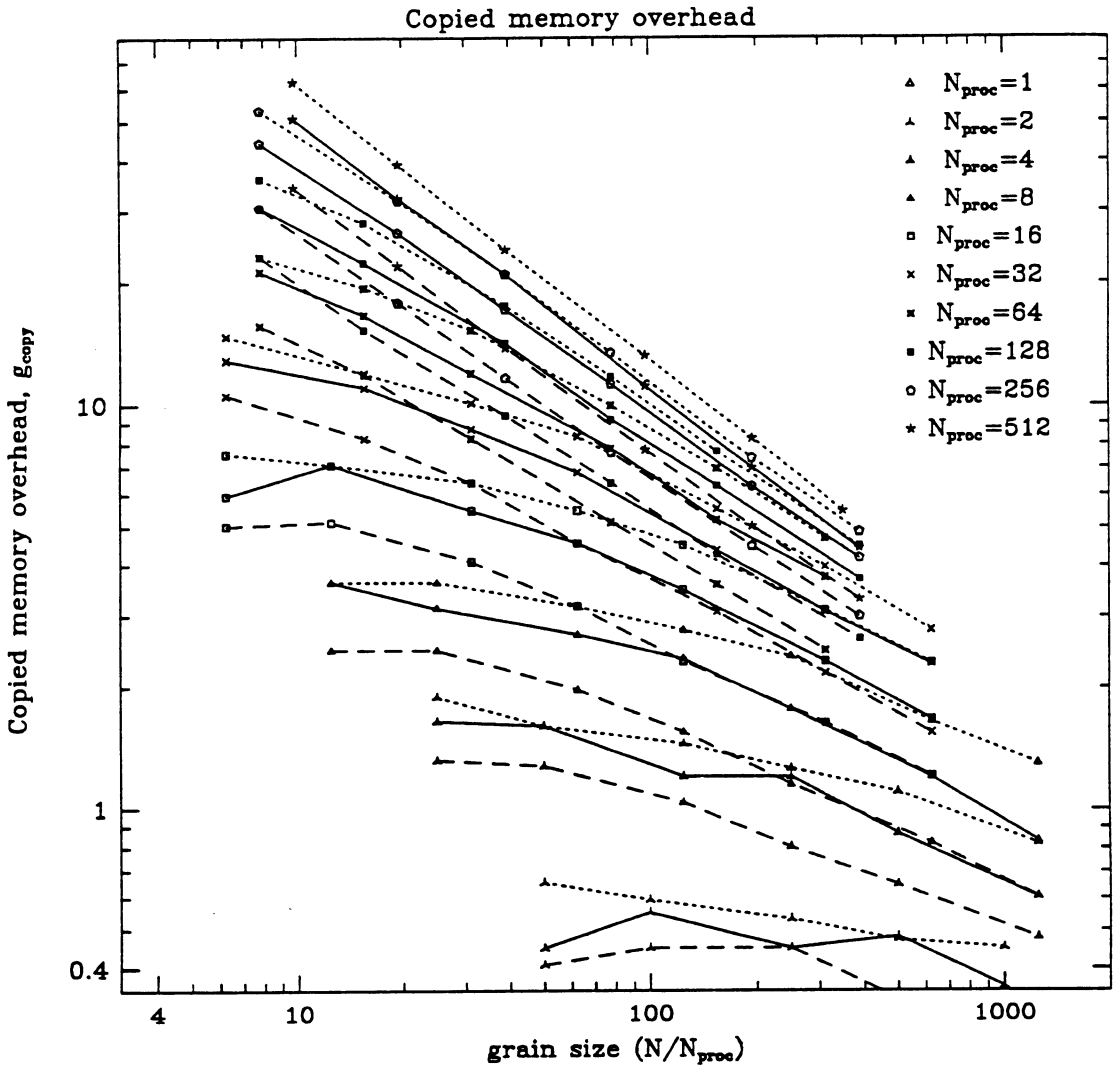


Figure 8.5. Copied memory overhead, g_{copy} vs. N_{grain}

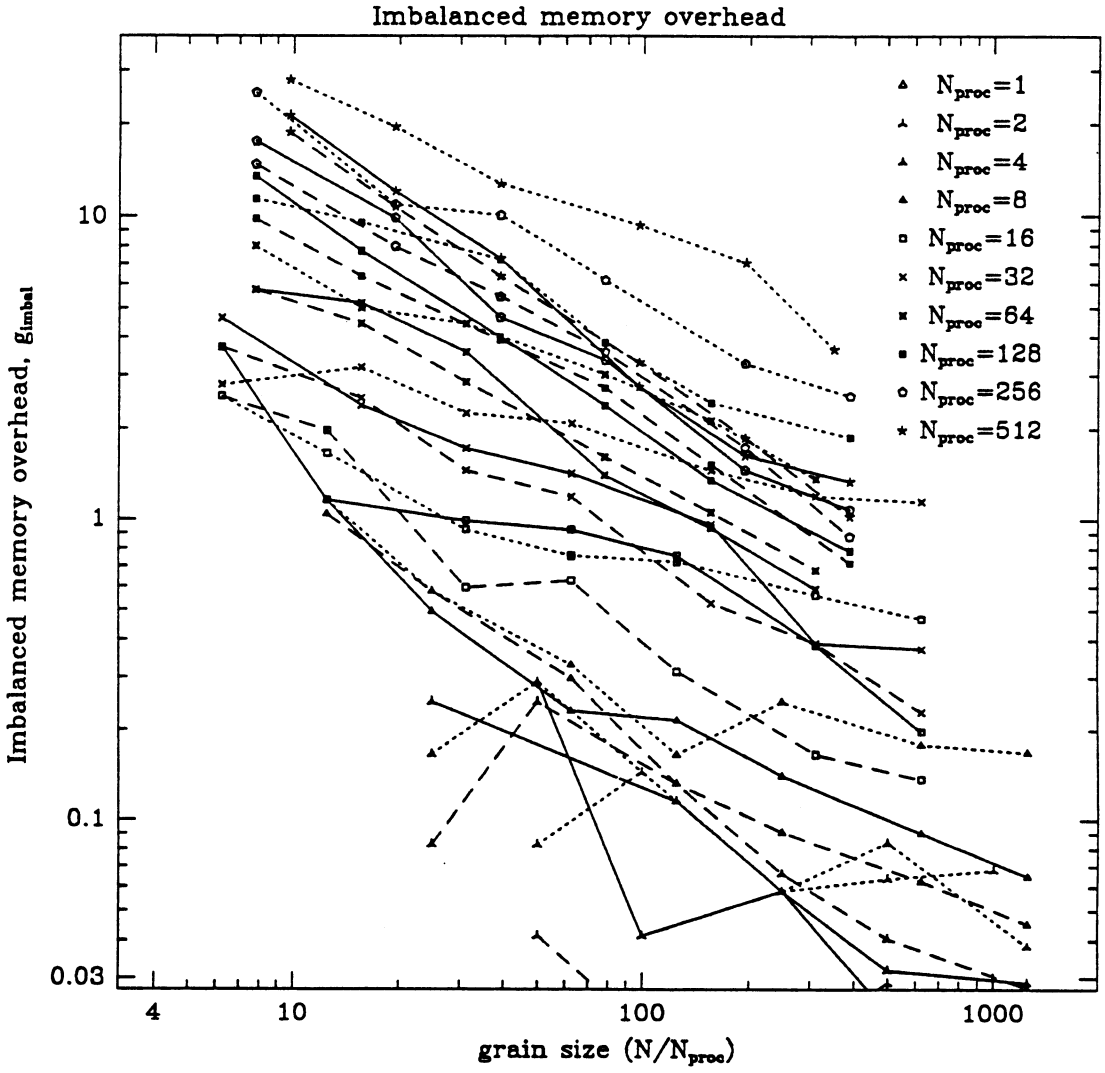


Figure 8.6. Imbalanced memory overhead, g_{imbal} vs. N_{grain}

These plots are only moderately interesting, in and of themselves. As we shall see in the following sections, their importance lies in the fact that much of the computational overhead can be directly related to memory overhead. It is worth noting that the overheads decrease quite rapidly with N_{grain}, indicating that less and less memory is wasted as the grain-size grows. This fact is especially reassuring, as it suggests that the largest runnable simulation grows faster than linearly with increasing processor memory. That is, if processor memory is doubled, we are capable of running a simulation with more than twice as many particles. This is demonstrated by Figure 8.7, which plots the size of the simulation, N_{grain} against $max(d_{step})$. The trend is especially evident for large numbers of processors, for which the maximum allowable grain size increases rapidly with increasing memory.

8.2.6. Natural grain size, q.

The reason for this surprising behavior is that the grain sizes attainable with the Ncube's limited memory are small compared to the "natural" size dictated by the algorithm. In order to provide a gross estimate of the algorithm's "natural" grain size, let's make some drastic assumptions about the nature of the algorithm. During each force calculation, each body interacts with a number of other bodies and cells. Call this number N_{int}. We saw in Chapter 4 that N_{int} is approximately proportional to $\log_8 N$, with a constant of proportionality roughly of order 100, i.e., F_{OC}. In order to estimate the memory that will be required during a calculation, let's assume that the N_{int} interactions are with the nearest N_{int} other bodies in the simulation, which lie in a ball of radius, R_{int}. * Furthermore, let us assume that the domain of a processor is spherical, with radius, R_{grain}. Figure 8.8 schematically shows region from which bodies must be drawn, in order to obtain the nearest N_{int} bodies for each body in a given processor.

If the density of bodies is ρ, and each body obtained by a processor requires

* This assumption is grossly inaccurate, but suffices for the discussion at hand.

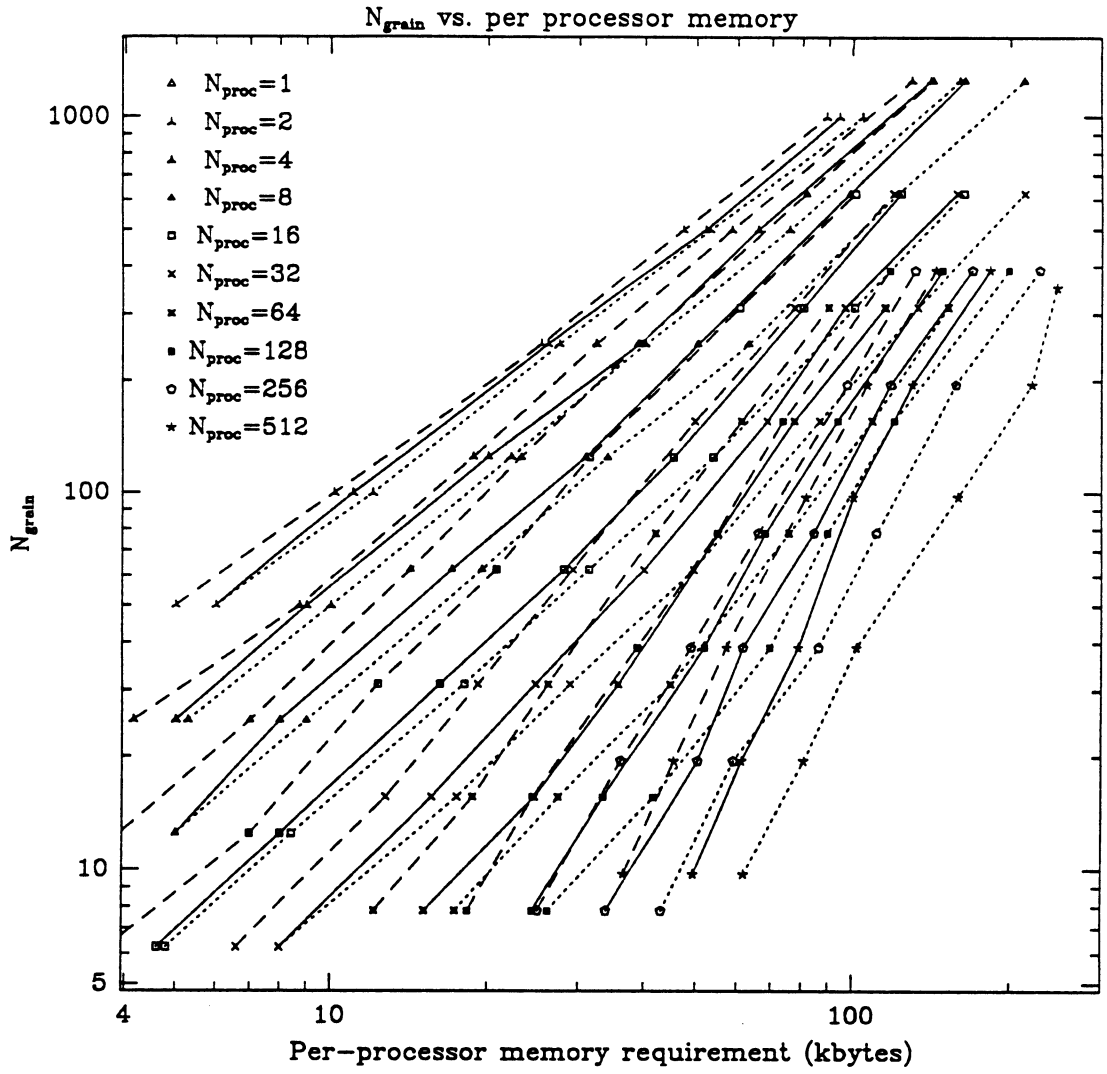


Figure 8.7. N_{grain} vs. $max(d_{step})$.

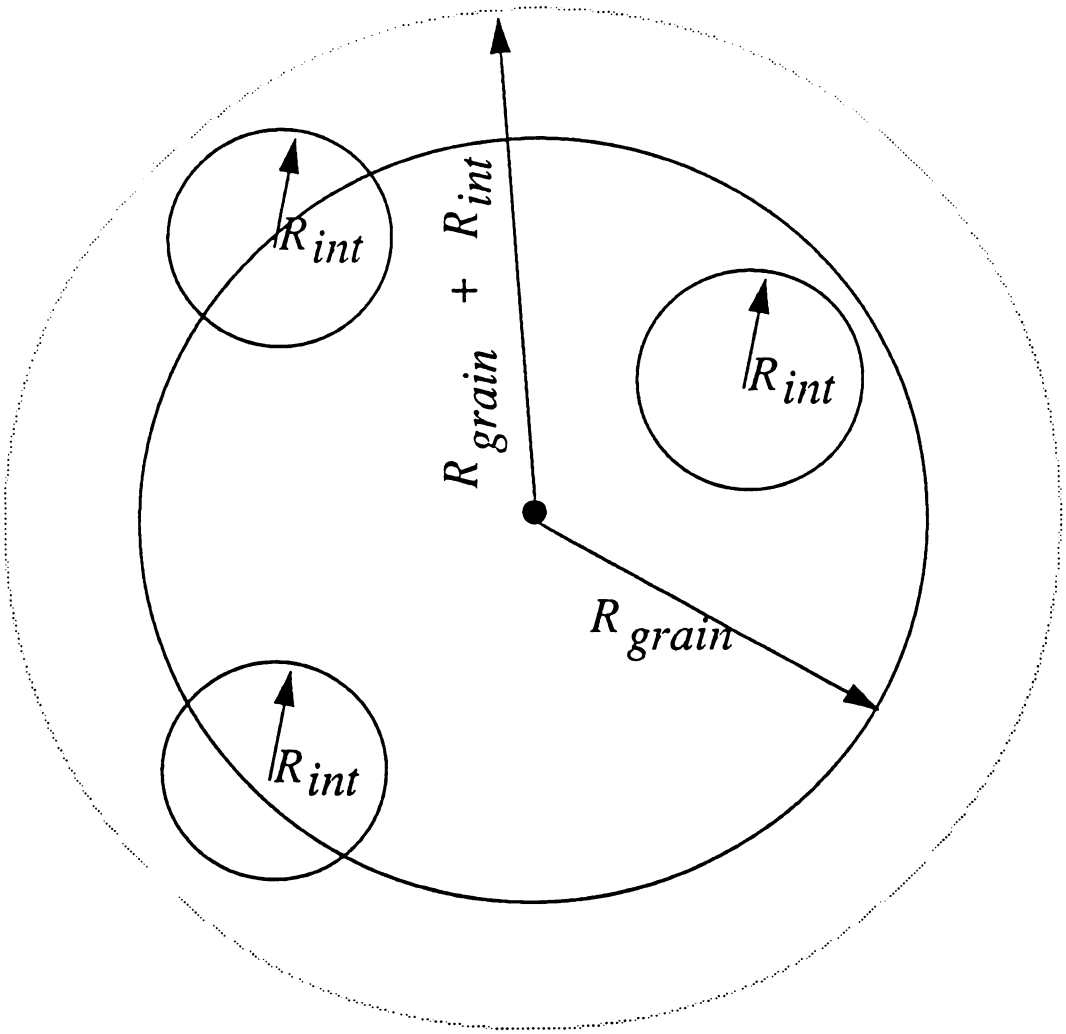


Figure 8.8. Simplified interaction domain of a processor.

m_0 units of memory, then we immediately have, from Figure 8.9,

$$\begin{aligned} N_{grain} &= \frac{4\pi}{3} R_{grain}^3 \rho, \\ N_{int} &= \frac{4\pi}{3} R_{int}^3 \rho, \\ M &= m_0 \frac{4\pi}{3} (R_{grain} + R_{int})^3 \rho, \end{aligned} \tag{8.33}$$

which are easily reduced to

$$M = m_0 N_{grain} \left(1 + \left(\frac{N_{int}}{N_{grain}} \right)^{1/3} \right)^3 \rho, \tag{8.34}$$

$$\frac{d \log N_{grain}}{d \log M} = 1 + \left(\frac{N_{int}}{N_{grain}} \right)^{1/3}. \tag{8.35}$$

Equation 8.35 tells us that when the grain size is significantly greater than N_{int}, we can expect N_{grain} to be proportional to M. On the other hand, for smaller grain sizes, the logarithmic slope of N_{grain} vs. M may be significantly greater than 1. Thus, the "natural" grain size for the algorithm is simply the value of N_{int}. Now we can compare the grain sizes in our simulations to N_{int}. For the largest simulations, N_{grain} and N_{int} are roughly comparable. Thus, we should not be surprised that the maximum attainable N_{grain} increases faster than linearly in Figure 8.7 for most of our simulations.

One might be tempted to try to fit the actual memory requirements to a parameterized version of Eqn. 8.35. Unfortunately, the assumptions required to "derive" Eqn. 8.35 are grossly inaccurate, and Eqn. 8.35 is a very poor model for the true dependence of memory on N_{grain} and N_{proc}. Its sole utility is to motivate the existence of a natural grain-size, approximately equal to N_{int}.

Unfortunately, no simple, well motivated model for the memory usage seems to fit the data. The problem is that the data spans the regime with N_{grain} near to N_{int}, but does not extend to high enough values of N_{grain} to exhibit the asymptotic

behavior at large grain sizes. In the range of grain-sizes and processor numbers for which we have data, the memory usage is determined by complicated tradeoffs in the transition from small N_{grain} to large N_{grain} behavior, as well as the transition of the algorithm from its small N, $O(N^2)$, to large N, $O(NlogN)$, behavior. It is no surprise that the data fails to display a conveniently simple form. Nevertheless, a simple empirical form does fit the available data.

Figure 8.8 shows a plot of M_{copy} vs. the quantity,

$$q = \frac{N_{grain}}{N_{int}\log_2(N_{proc})}. \tag{8.36}$$

The figure suggests that the M_{copy} is fairly insensitive to changes in N, N_{proc}, N_{int} and model type, which leave q unchanged.

We can now attempt to extrapolate from Figure 8.9 to values of q inaccessible in our simulations. Here, we lack a theoretical guide to the expected behavior, and the conclusions must be regarded as conjecture. Figure 8.9 shows a line of logarithmic slope $-2/3$, which fits the data reasonably well, leading us to conjecture

$$\begin{aligned} M_{copy} &= N_{grain} f(q), \\ \lim_{q\to 0} M_{copy} &\propto N_{grain} q^{-\alpha}, \\ \alpha &\approx 2/3. \end{aligned} \tag{8.37}$$

What does Eqn. 8.37 imply about the asymptotic dependence of memory on N, N_{grain} and N_{proc}? First, we notice that for large N, we may use

$$N_{int} \propto \log(N) = \log(N_{grain}) + \log(N_{proc}). \tag{8.38}$$

If we let N go to infinity, holding N_{proc} fixed, then q increases in proportion to N, and Eqn. 8.37 becomes

$$\lim_{\substack{N\to\infty, \\ N_{proc}\,fixed}} M_{copy} \propto N_{grain}^{1-\alpha}\left(1+\left(\frac{\log(N_{grain})}{\log(N_{proc})}\right)\right)^{\alpha}. \tag{8.39}$$

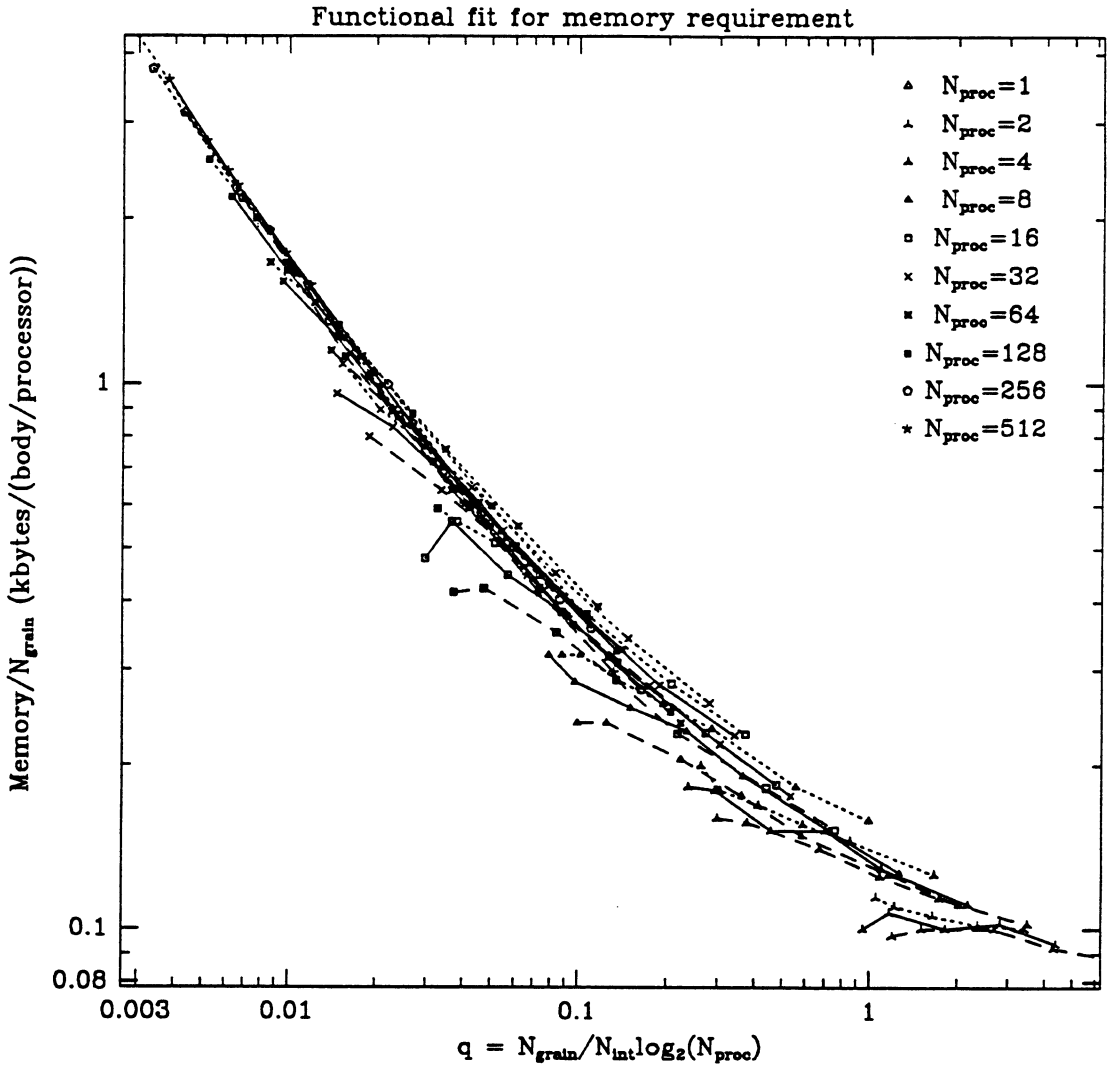


Figure 8.9. Memory/N_{grain} vs. natural grain size, q.

Similarly, if we let N_{proc} go to infinity, holding N_{grain} fixed, then q approaches zero and

$$\lim_{\substack{N_{proc}\to\infty,\\ N_{grain}\ fixed}} M \propto \log^{\alpha}(N_{proc})\left(1+\left(\frac{\log(N_{proc})}{\log(N_{grain})}\right)\right)^{\alpha}. \tag{8.40}$$

The specific value of α in Eqn. 8.37 is highly suspect, although it does fit the limited range of available data. Nevertheless, it is probably safe to draw the following general conclusions about the behavior of M for large numbers of processors:

1. For fixed N_{proc} and large N_{grain}, M_{copy} is proportional to some low power of N_{grain}, near to 1/3.
2. For fixed N_{grain} and large N_{proc}, M_{copy} is proportional to a power the logarithm of N_{proc}, near to 4/3.

8.2.7. Complexity overhead, f_{cplx}.

The significant portion of the total overhead comes from additional complexity introduced into the code in order to parallelize it. The complexity overhead, f_{cplx}, is plotted against grain size in Figure 8.10.

We have already seen in Section 8.2 that the force evaluation in parallel is executed with no complexity overhead (in fact, the overhead associated with force evaluation is very slightly negative, as discussed in Section 8.2). Thus, we must look elsewhere for the complexity overhead.

A comparison of the code fragments in Chapter 7, which describe the construction of the locally essential tree, with those in Chapter 2, which describe the construction of a complete tree, makes it clear that there is considerably more involved in building the tree in parallel than on a single processor. None of the median searches or other computations associated with orthogonal recursive bisection need to be performed on a single processor. Nevertheless, the detailed timing data (not plotted) reveals that only a small part of the complexity overhead can be attributed to the orthogonal recursive bisection. The vast majority of the complexity overhead is associated with building the locally essential tree,

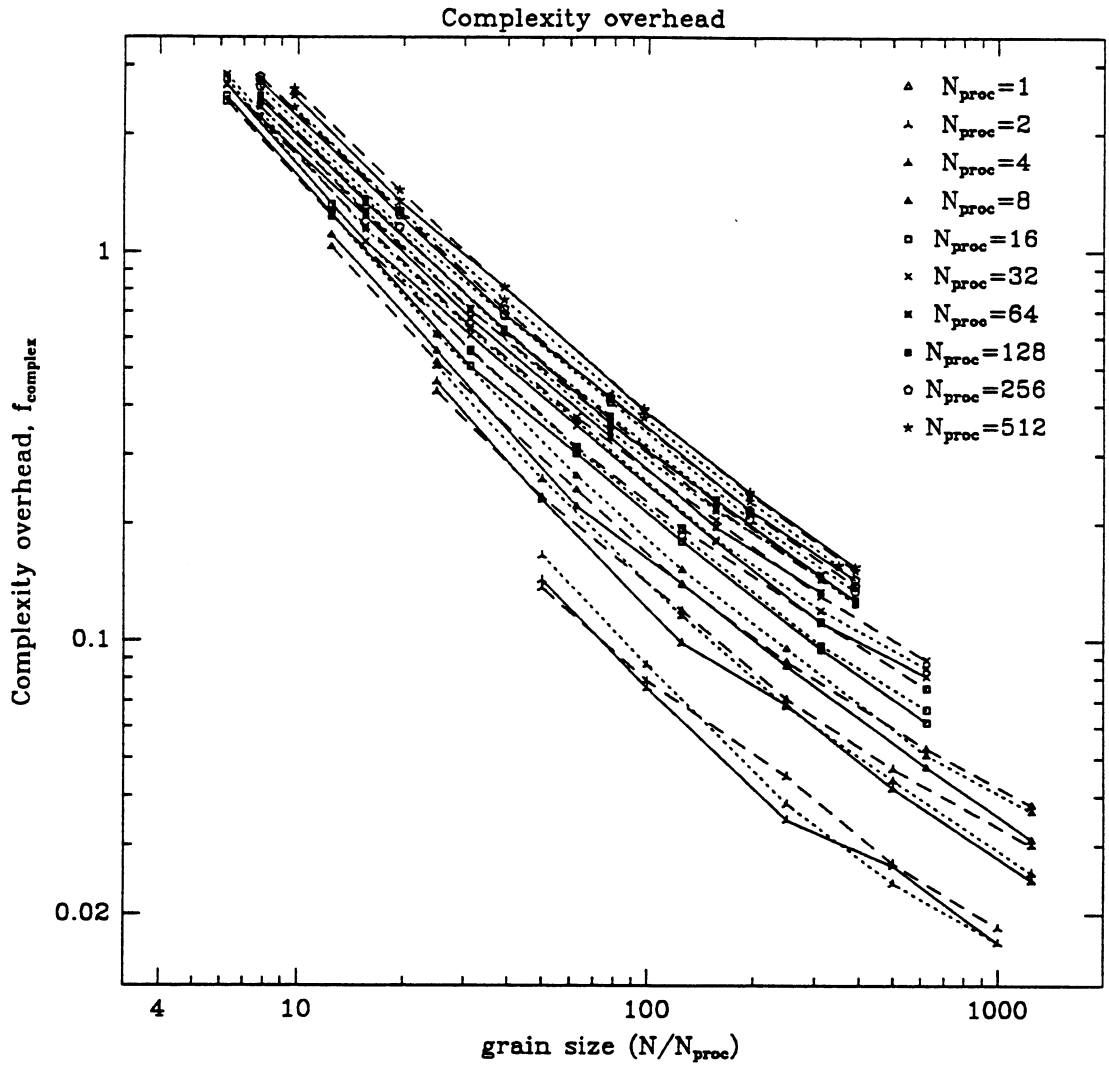


Figure 8.10. Complexity overhead, f_{cplx} vs. grain size, N_{grain}.

and not specifically with determining the domain decomposition.

The steps required by Code 7.4 which specifies two traversals of the tree, along with the procedure for inserting new `FBodies` and `MTCells` into the tree are undeniably more complex than the simple procedure in the sequential procedure in Code 2.1. Nevertheless, only a small amount of the complexity overhead is due to the complexity of the insertion process itself, i.e., the additional comparisons, bit-setting operations, branches, etc. necessary for each inserted item. Instead, almost all of the additional complexity is due to the fact that the number of `Cells` and `Bodies` in a locally essential tree is considerably larger than the N_{proc}^{-1} times the number of `Cells` and `Bodies` in the equivalent complete sequential tree. Thus, the subroutine `InsertBody`, which inserts a `Body` into the tree, and the subroutines `BodyMpole` and `ParAxis`, which compute multipole moments, are executed more times in parallel than on a single processor. Examination of Code 7.14 through Code 7.21 reveals that insertion of `FBodies` and `MTCells` into the tree involves a tree descent to find a terminal cell at which to insert the new object, followed by a substantial calculation to compute the new multipole moments of the terminal cell with the new object included. The tree descent is not costly, but the computation of new multipole moments for each inserted item constitutes the vast majority of T_{cplx}.

We can substantiate this claim by plotting T_{cplx} against M_{copy}. If the claim of previous paragraph is correct, then T_{cplx} will be proportional to the excess memory required by the parallel implementation. It is clear from Figure 8.11 that T_{cplx} is reasonably well accounted for by the relationship,

$$T_{cplx} = cM_{copy} \qquad c \approx 0.2(sec/kbyte). \tag{8.41}$$

It is especially reassuring that the different models types, i.e., Jaffe sphere, uniform sphere, and merger, all coincide in Figure 8.11. Thus, the extra time spent "computing" in parallel, over and above that which is required by the sequential algorithm is proportional to the extra memory used in parallel, over and above

that which is required by the sequential algorithm. The constant of proportionality in Eqn. 8.41 depends, of course, on the particulars of the implementation, but does not depend on the detailed distribution of bodies in space.

We can combine Eqn. 8.37 together with Eqn. 8.41 to obtain an estimate of T_{cplx} purely in terms of N and N_{proc}. If we take

$$T_{intrinsic} \propto N_{grain} N_{int}, \tag{8.42}$$

then

$$f_{cplx} = \frac{T_{cplx}}{T_{intrinsic}} \propto \frac{f(q)}{N_{int}}. \tag{8.43}$$

Figure 8.12 shows a plot of $f_{cplx} N_{int}$ vs. q. The points in Figure 8.12 are reasonably coincident, confirming that Eqn. 8.43 is a good empirical model for the behavior of f_{cplx}. Furthermore, the curves all have logarithmic slope of about $-2/3$, at least for small q, in agreement with Eqn. 8.37.

As we hope to be able to use the algorithm on systems larger than the Ncube, it is important to estimate the behavior of f_{cplx} for large values of N_{proc} and large values of N_{grain}. Simple algebraic rearrangement reveals that

$$\lim_{\substack{N_{proc} \to \infty, \\ N_{grain} \text{ fixed}}} f_{cplx} \propto \log(N_{proc}) \tag{8.44}$$

and

$$\lim_{\substack{N_{grain} \to \infty, \\ N_{proc} \text{ fixed}}} f_{cplx} \propto \frac{\log(N_{grain})^{1/3}}{N_{grain}^2}. \tag{8.45}$$

These results are extremely encouraging, as they tell us that adding more processors of a fixed size will lead to a degradation of performance which grows very slowly (logarithmically) with the number of processors. Furthermore, overhead decreases rapidly (as the $-2/3$ power) as a function of grain size (and hence, processor memory).

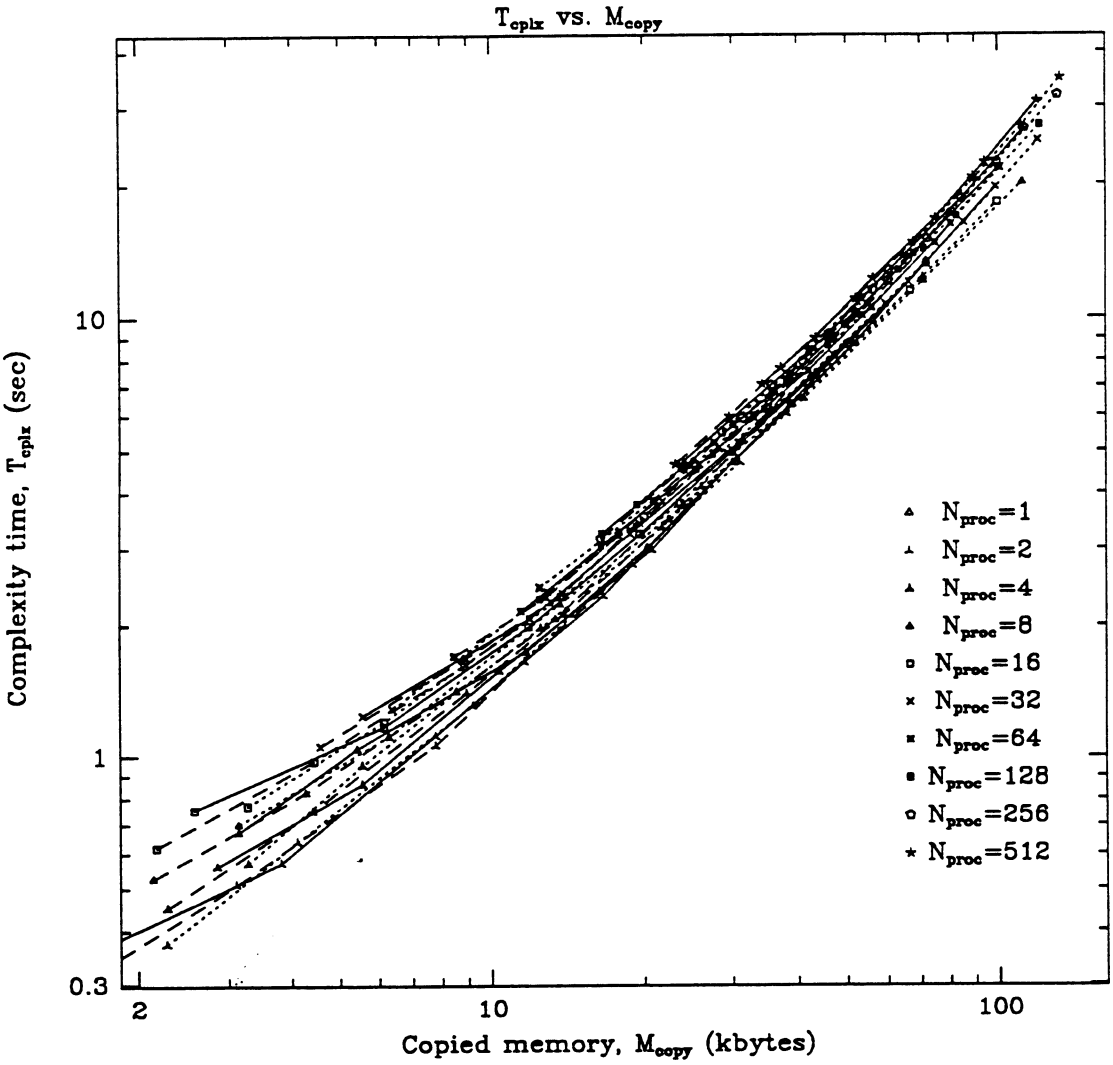


Figure 8.11. Complexity time, T_{cplx} vs. copied memory, M_{copy}.

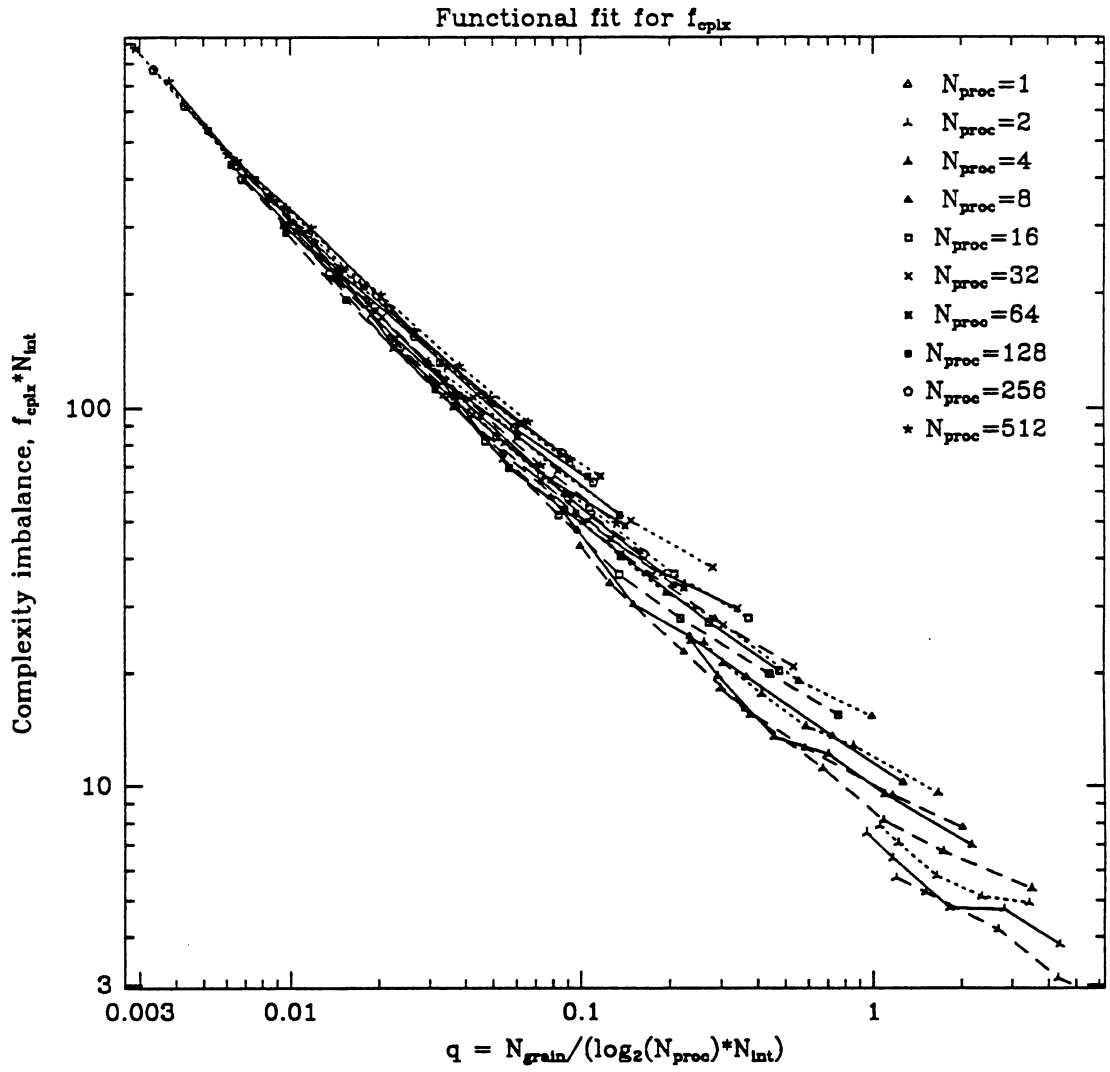


Figure 8.12. Complexity overhead, $N_{int}f_{cplx}$ vs. natural grain size, q.

8.2.8. Waiting overhead, f_{wait}.

The overhead associated with waiting for processor synchronization points is accounted for by f_{wait}. In the BH algorithm, f_{wait} is the largest identifiable fraction of the total overhead in large simulations. Figure 8.13 shows f_{wait} plotted against N_{grain}.

Since the algorithm presented in Chapter 7 is based on a synchronous model of communication, it is a simple matter to locate where waiting overhead might occur. Waiting can occur at any point in the algorithm at which data is exchanged between processors. Since the communication is synchronous, all such points appear explicitly in the program, i.e., as subroutine calls or language constructs.

A review of Chapter 7 reveals two points in the algorithm at which data must be exchanged, during the orthogonal recursive bisection, and during the tree-build phase. In orthogonal recursive bisection we see that each processor carries out an explicit exchange of data with each of $\log_2(N_{proc})$ other processors. Furthermore, each time through the loop of `ORB`, it is necessary to find the median of a distribution of work, which in turn is equivalent to finding the root of a function. Each function evaluation contains an explicit synchronization amongst all processors in a `ProcessorSubset`. Fortunately, the time between synchronizations is both short, and approximately the same on all processors in the `ProcessorSubset`, so the T_{wait} associated with these synchronization points is small. In fact, the total time spent in `ORB`, including waiting, communication and computation is negligible compared to $T_{intrinsic}$. Thus, at least for the purposes of understanding the dominant contribution to f_{wait}, we may disregard `ORB`. (Note, however, that Figure 8.14 shows all contributions to f_{wait}, including that from `ORB`).

The majority of T_{wait} occurs during the tree-build phase of the algorithm. The top-level procedure for constructing the tree is shown in Code 7.4, which consists of an iteration over each of the bisectors ($\log_2(N_{proc})$ of them) identified by `ORB`. Each of the iterations consists of two tree traversals, followed by an exchange of data between two processors, with an implied synchronization, followed by the insertion

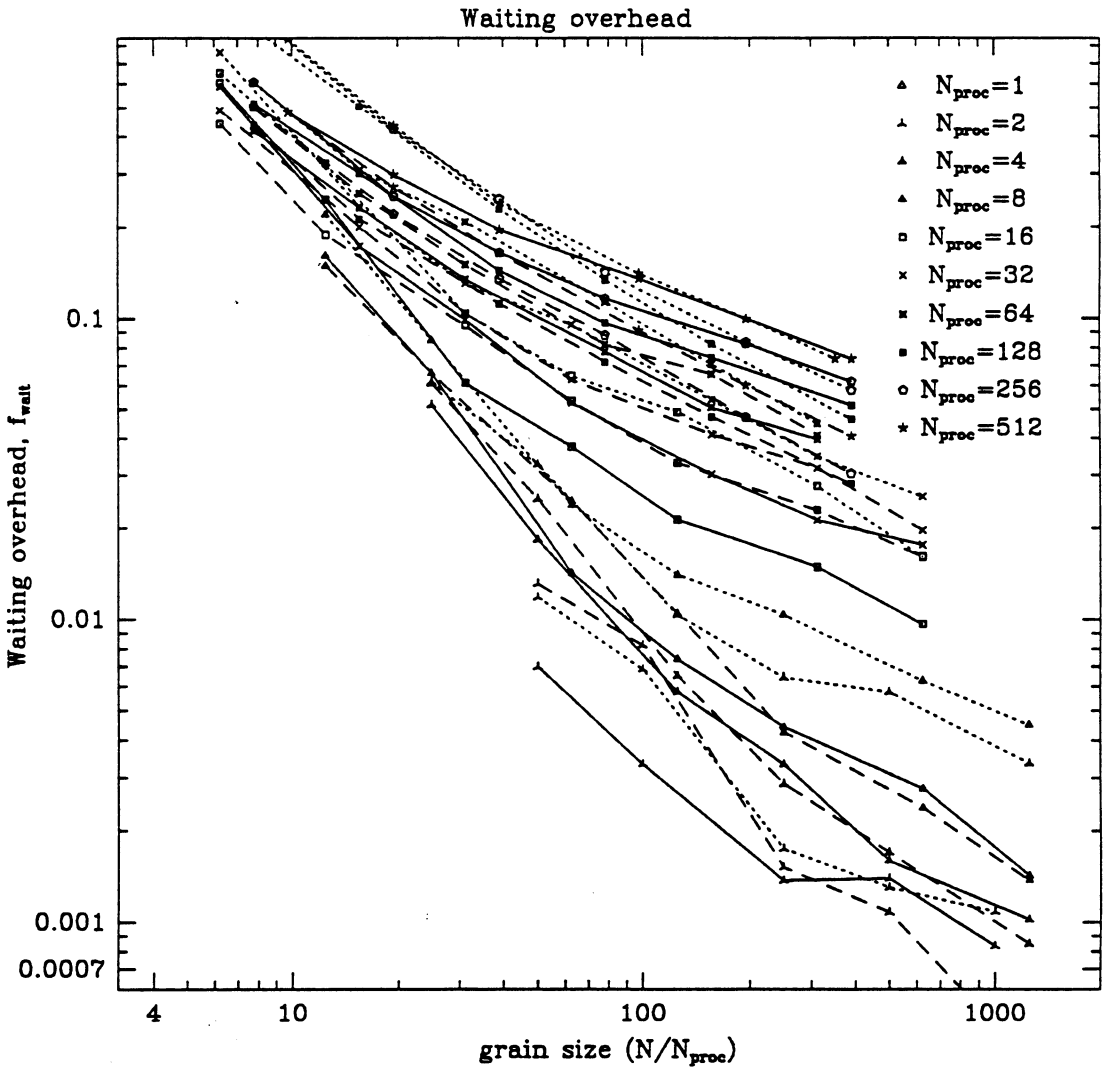


Figure 8.13. Waiting overhead, f_{wait} vs. N_{grain}.

of the newly received data into the tree. The possibility for synchronization delay is present at each of the data exchanges. The amount of time a processor waits at one of the synchronization points depends, of course, on the amount of work it was required to perform since the last synchronization point, and how much work its neighbor was required to perform since the last synchronization point.

We saw in the previous section, that the computational effort expended in building the tree, calculating multipoles, etc., is accounted for by T_{cplx}. Let us assume, pessimistically, that the slowest processor spends some fraction of T_{cplx} waiting at each of the $\log_2(N_{proc})$ synchronization points. Then, we conclude that

$$\begin{aligned} T_{wait} &\propto T_{cplx} \log_2(N_{proc}), \\ f_{wait} &\propto f_{cplx} \log_2(N_{proc}). \end{aligned} \tag{8.46}$$

Using Eqn. 8.43, we can predict that

$$f_{wait} \propto \frac{\log_2(N_{proc})}{N_{int}} f(q). \tag{8.47}$$

Figure 8.14 shows a plot of $f_{wait} N_{int} / \log_2(N_{proc})$ against q. If Eqn. 8.47 is valid, then all the curves in Figure 8.13 should coincide. The coincidence in this case is not as convincing as in, for example, Figure 8.12. Nevertheless, Eqn. 8.47 appears to be accurate to within a small numerical factor for the data available. In fact, the data appear to lie on a line of logarithmic slope -1. If that trend continues to hold for data outside the regime we have studied, then

$$f_{wait} \propto \frac{\log_2(N_{proc})}{q N_{int}} = \frac{(log_2(N_{proc}))^2}{N_{grain}}, \tag{8.48}$$

which means that as the number of processors increases with fixed N_{grain}, f_{wait} will increase as the square of the logarithm of N_{proc}, but if the grain size increases with the number of processors fixed, then f_{wait} will drop as the inverse of N_{grain}. This is a reassuring result, as it tells us we can expect slowly degraded performance with increasing processor number, but rapidly improving performance with increasing grain size.

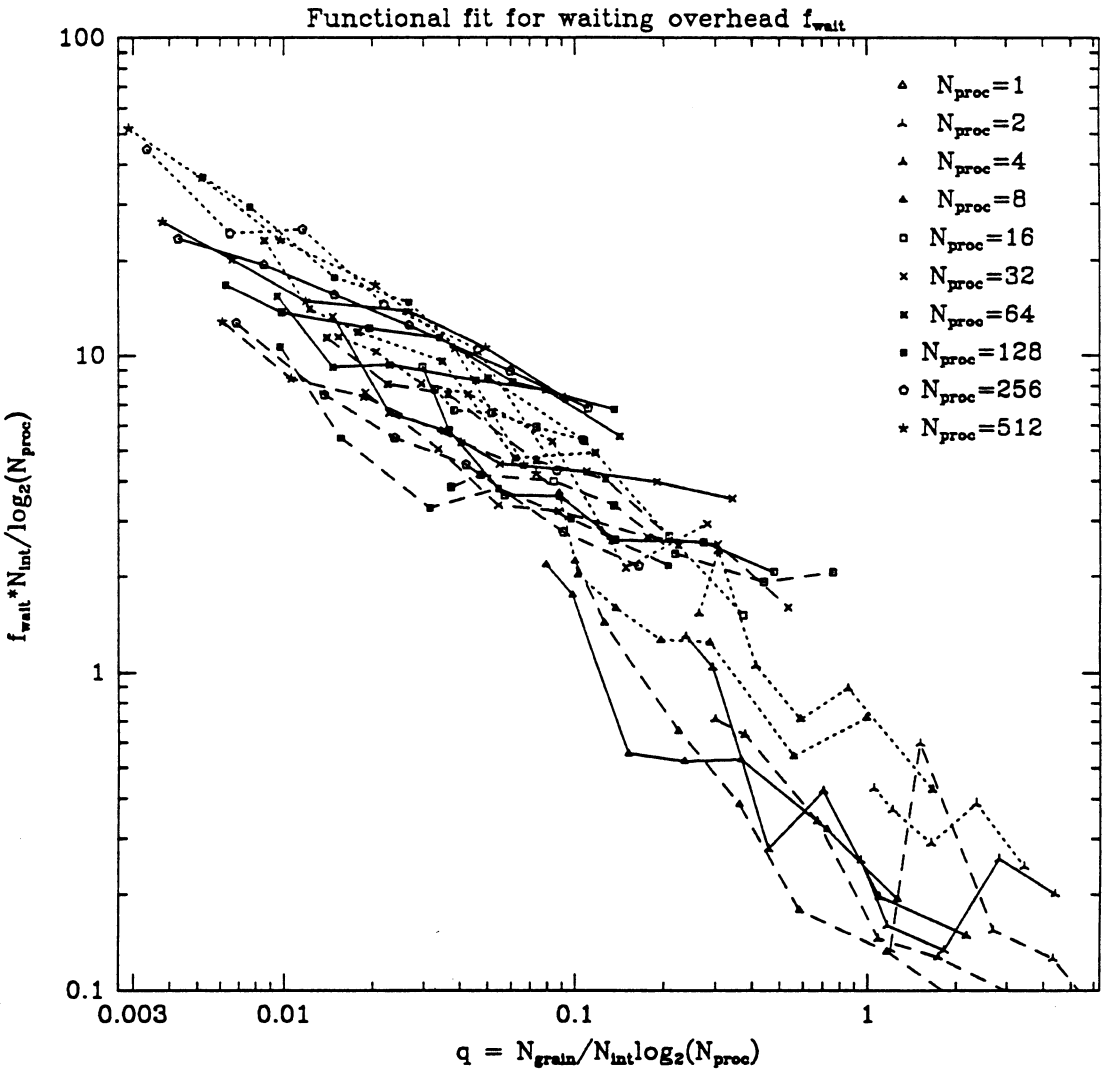


Figure 8.14. Empirical fit for waiting overhead, $\frac{f_{wait}*N_{int}}{\log_2(N_{proc})}$ vs. natural grain size, q.

8.2.9. Communication overhead, f_{comm}.

The time spent in communication is of little practical importance in the algorithm. It is much smaller than any of the other sources of overhead we discuss. Of course, the magnitude of the T_{comm} depends on the performance of the hardware and software used to transmit data from one processor to another. The dimensionless communication overhead is the ratio of T_{comm} to the intrinsic time associated with the algorithm, $T_{intrinsic}$. $T_{intrinsic}$ does not depend in any way on the communication subsystem, but does reflect the computational power of each processor in isolation. Thus, one often characterizes communication overhead in terms of the ratio of "characteristic communication time," τ_{comm}, and "characteristic calculation time," τ_{calc}. Unfortunately, real computers are rarely so simple as to be characterized by one or two characteristic times. Communication, for example, may be characterized by a latency, and a rate of throughput, or perhaps may be highly load dependent. Integer, floating-point, vectorized and non-vectorized calculations may all have widely different characteristic times. Thus, it is difficult to predict with precision how an algorithm will perform on another type of hardware. Nevertheless, the communication overhead on the Ncube is about a factor of 10 smaller than other overheads, which implies that communication will not be a significant problem on any machine with a suitably defined ratio of τ_{comm} and τ_{calc} less than ten times the ratio for the Ncube. Even if the ratio is larger, that only means that f_{comm} is the dominant form of overhead. It does not necessarily imply that f_{comm} is large in absolute magnitude. Figure 8.15 indicates that communication overhead drops rapidly with grain size, so a poor communication system may still be compensated for by a large grain size, i.e., memory.

Figure 8.15 shows communication overhead plotted against grain size. Comparison of Figure 8.15 with Figure 8.10 reveals that communication overhead closely parallels complexity overhead. This result follows from the following considerations:

1. Data ready for transmission is queued, so that it may be sent in large blocks

(whose maximum size is dictated by memory constraints). Thus, there are few individual messages, of large size, so throughput is more significant than latency, and T_{comm} is approximately proportional to the amount of data transmitted.

2. As discussed in Section 8.2, T_{cplx} is the result of building a locally essential tree containing data acquired from other processors, and is roughly proportional to the amount of data copied from other processors. Since the copied data referred to in Section 8.2 must have been transmitted, we conclude that T_{comm} is roughly proportional to T_{cplx}.

We note that all communication takes place between processors which have the logical structure of a hypercube. If the hardware is configured as a hypercube, then communication overheads may be significantly reduced, as the mapping from logical to hardware processors, and the routing of messages is extremely simple. If the underlying hardware reflects a different topology, then some mapping from the logical hypercube structure onto the hardware must be devised, and messages must be routed to their logical neighbors. Whether this entails any significant overhead will depend on the details of the hardware and software available. We emphasize that the logical hypercube structure is the natural one of the algorithm, and it is simply fortuitous that it corresponds to the hardware configuration of the Ncube. Furthermore, communication overhead constitutes a small fraction of the total overhead. Only machines with a very much greater ratio of communication speed to computation speed than the Ncube will be affected by communication overhead at all. In most cases, the overhead that arises from supporting the logical hypercube topology in software (if necessary) will not be significant.

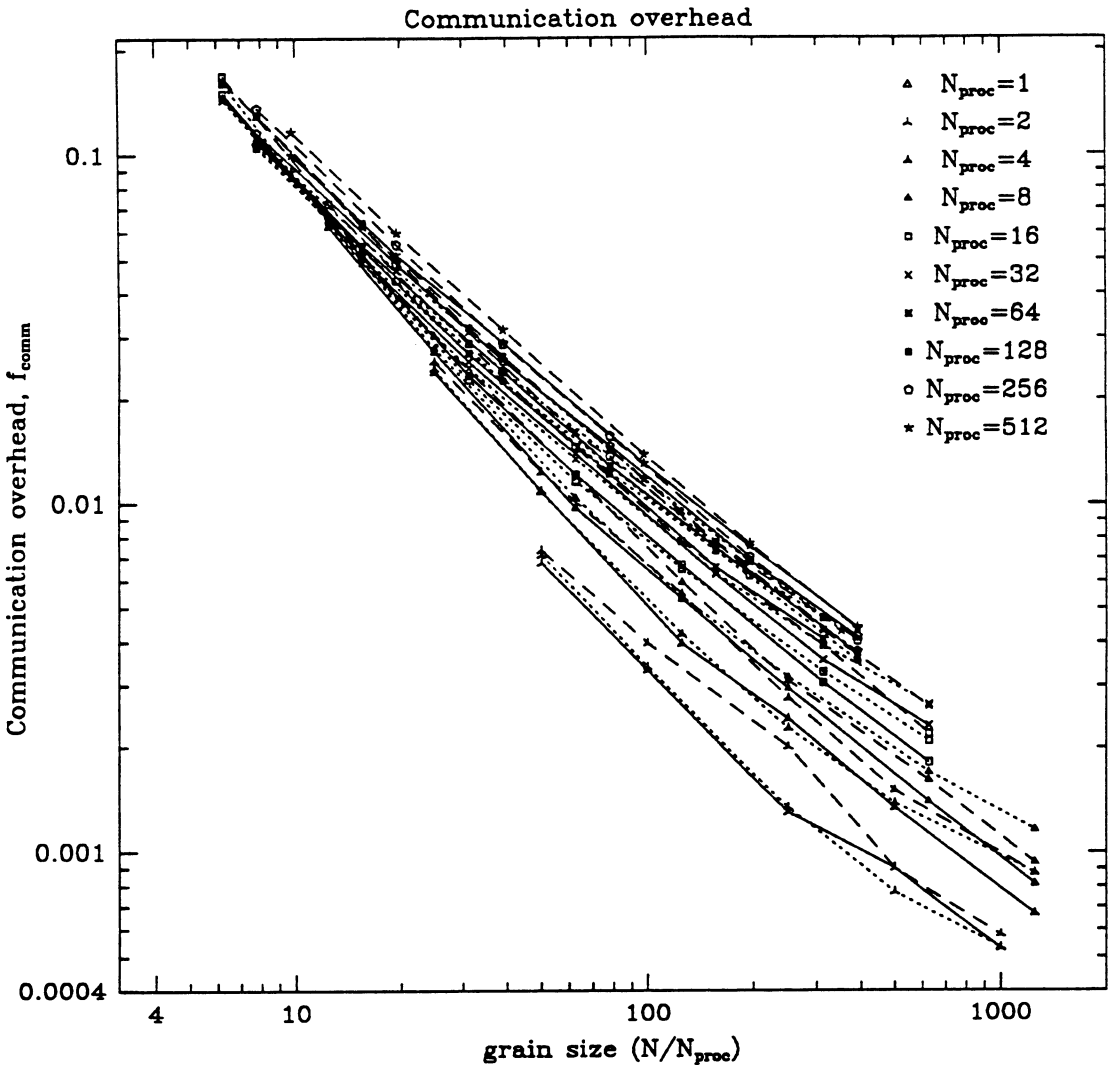


Figure 8.15. Communication overhead, f_{comm} vs. N_{grain}.

8.2.10. Load imbalance overhead, f_{imbal}.

Finally, we turn to the last significant component of the computational overhead, the load imbalance. Figure 8.16 shows the load imbalance overhead plotted against N_{grain}. In most situations with substantial grain size, i.e., $N_{grain} > 100$, the load imbalance is considerably smaller than the other sources of overhead. We can use Eqn. 8.16 to distinguish between computational and communication load imbalance. In this case, the communication load imbalance is at least a factor of ten smaller than the communication overhead, and the data are not plotted separately. In fact, the communication load imbalance, defined as in Eqn. 8.15, is almost as likely to be positive as negative. The slowest processor, overall, may well have a slightly less than average communication load. In any case, the overwhelming majority of the load imbalance is due to computation and not communication.

In Section 7.2, we described the technique used to minimize computational load imbalance. The procedure, which involves dynamically adjusting the target percentile used in the orthogonal recursive bisection algorithm, performs very well in practice. In most cases, the computational load imbalance is within a factor of a few times N_{grain}^{-1}. This level of load imbalance is generally the best that one can hope for. Load balance can only be achieved by transferring discrete bodies, of which each processor has, on average, N_{grain}. The "roundoff error" introduced by this discreteness effect is of order N_{grain}^{-1}, so we expect that under good conditions, i.e., when the balancing algorithm is working effectively, the residual load imbalance will be proportional to N_{grain}^{-1}.

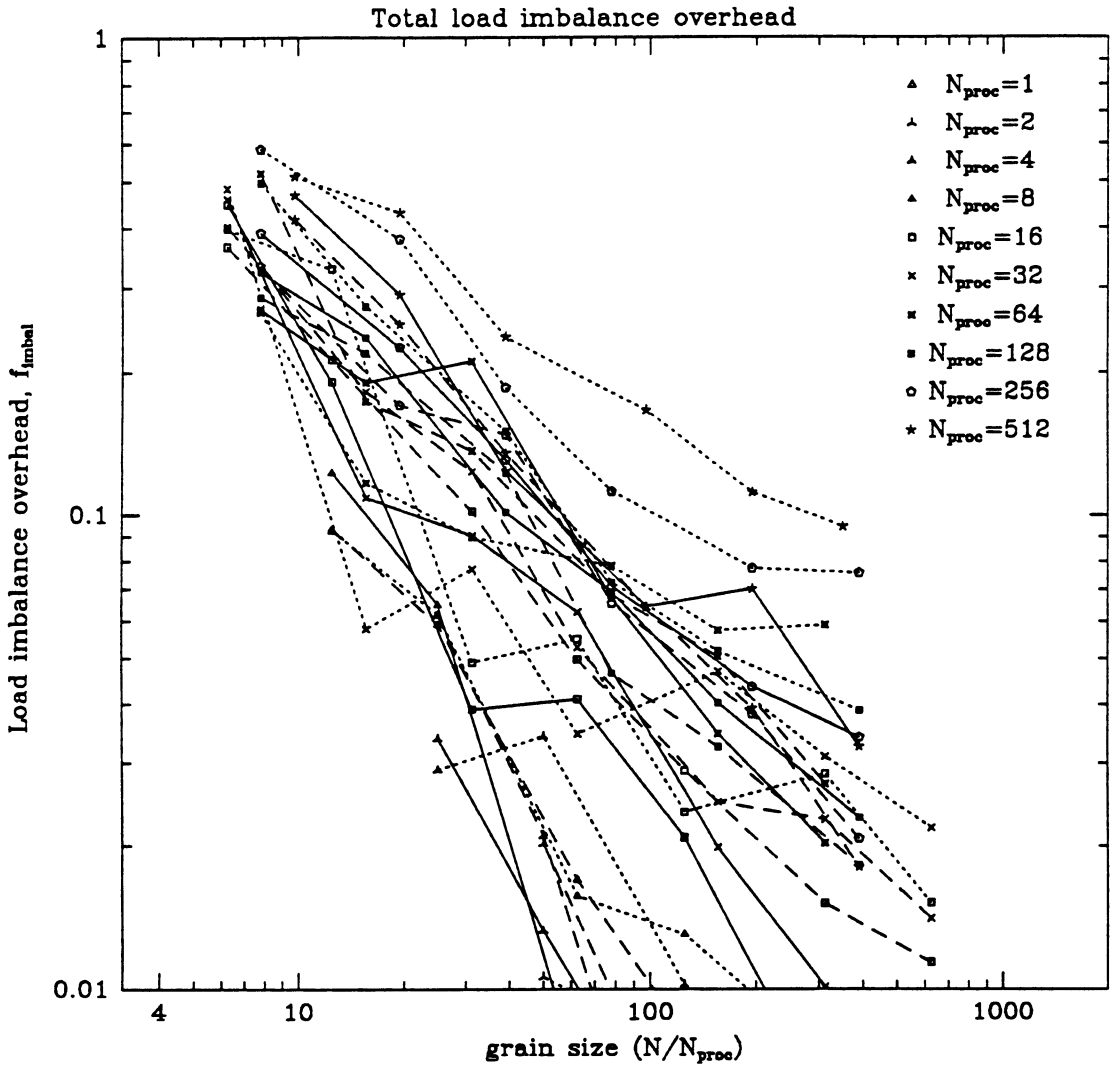


Figure 8.16. Load imbalance overhead, f_{imbal} vs. N_{grain}.

8.3. Summary.

The parallel N-body algorithm performs extremely well. The most significant remaining sources of overhead are processor synchronization and additional parallel complexity. Load imbalance is very low, indicating that the orthogonal recursive bisection with dynamically adjusted splitting, described in Section 7.1, is effectively seeking an equitable distribution of work. Communication overhead, which is often the most significant source of overhead,[6] is completely negligible. It seems unlikely that substantial reduction in the two remaining sources of overhead can be achieved without drastically modifying the algorithm. Warren's[44] approach may succeed in eliminating some of the remaining complexity overhead, by eliminating redundant calculation and storage of multipole moments. Communication overhead will increase, however, and the extent to which gains offset losses remains to be seen. In any event, even if one could "magically" eliminate all sources of overhead, the program would only run about 35% faster (less for larger problems), suggesting that the algorithm of Chapter 7 has probably reached the point of diminishing returns on investment of additional improvements.

A. Proof of bounds on C_{avg} and D_{avg}.

In this appendix we present a detailed proof of the claim that C_{avg} is bounded by Nc_m, and an outline of a proof that D_{avg} is bounded by $\log_8 d_m N$.

A.1. Bounds on C_{avg}.

We wish to show that there exists a constant, c_m, such that

$$C_{avg}(N) < Nc_m \tag{A.1}$$

for all $N > 0$.

In order to complete the proof, we will make use of the following three lemmas:

Lemma 1. For all N, m, and p,

$$\frac{D_{8N,m}(p/8)}{D_{N,m}(p)} < \exp\left(\frac{Np^2}{1-p} + \frac{m(m+1)}{2(N-m)}\right). \tag{A.2}$$

Lemma 2. If $Np(1-p) > 2m$ then $D_{8N,m}(p/8) < D_{N,m}(p)$.

Lemma 3. If there exists constants, $A > 1$, δ_1, δ_2, α_1, $\alpha_2 > 0$, x_0 and a_0, and the function, $f(x)$, has the following properties:

$$f(Ax) < f(x)\exp\left(\frac{\delta_1}{x^{\alpha_1}}\right) + \frac{\delta_2}{x^{\alpha_2}}, \qquad \text{for all } x > x_0, \tag{A.3}$$

$$f(x) < a_0, \qquad \text{for all } x_0 < x < 8x_0, \tag{A.4}$$

then

$$f(x) < \exp(\Delta_1)(a_0 + \Delta_2), \qquad \text{for all } x > x_0, \tag{A.5}$$

where

$$\begin{aligned} \Delta_1 &= \frac{\delta_1}{x_0^{\alpha_1}} \frac{A^{\alpha_2}}{A^{\alpha_2} - 1}, \\ \Delta_2 &= \frac{\delta_2}{x_0^{\alpha_2}} \frac{A^{\alpha_2}}{A^{\alpha_2} - 1}. \end{aligned} \tag{A.6}$$

A.1.1. Proof of Lemma 1.

It is well known [34](Eqn.26.5.24) that the cumulative binomial distribution may be evaluated in terms of the incomplete beta function,

$$D_{N,m}(p) = I_p(m+1, N-m) = \frac{1}{B(m+1, N-m)} \int_0^p t^m (1-t)^{N-m-1} dt. \quad (A.7)$$

Where $B(a,b)$ is the beta function,

$$B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}. \quad (A.8)$$

Thus,

$$D_{8N,m}(p/8) = \frac{1}{B(m+1, 8N-m)} \int_0^{p/8} t^m (1-t)^{8N-m-1} dt. \quad (A.9)$$

A simple change of variables under the integral, to $y = 8t$, and a small amount of algebraic rearrangement leads to

$$D_{8N,m}(p/8) = \left(\frac{B(m+1, N-m)}{8^{m+1} B(m+1, 8N-m)} \right) \quad (A.10)$$
$$\frac{1}{B(m+1, N-m)} \int_0^p y^m (1-y)^{N-m-1} \left(\frac{(1-y/8)^{8N-m-1}}{(1-y)^{N-m-1}} \right) dt.$$

We may place an upper bound on this expression by noting the following relationship, which holds when $f(x)$ and $g(x)$ are everywhere positive,

$$\int_a^b f(x)g(x)dx < \sup_{a<x<b} g(x) \int_a^b f(x)dx. \quad (A.11)$$

Using Eqn. A.11, we transform the integral in Eqn. A.10 into $D_{N,m}(p)$ times an expression N, m, and p,

$$D_{8N,m}(p/8) < D_{N,m}(p) \left(\frac{B(m+1, N-m)}{8^{m+1} B(m+1, 8N-m)} \right) \sup_{0<y<p} \left(\frac{(1-y/8)^{8N-m-1}}{(1-y)^{N-m-1}} \right). \quad (A.12)$$

We note that the final expression in Eqn. A.12 is an increasing function of y, and so we may simply replace it with its value at $y = p$. With some additional algebraic rearrangement, Eqn. A.12 becomes

$$\frac{D_{8N,m}(p/8)}{D_{N,m}(p)} < \left(\frac{B(m+1,N-m)}{8^{m+1}B(m+1,8N-m)}\right)\left(\frac{(1-p/8)^8}{1-p}\right)^N\left(\frac{1-p}{1-p/8}\right)^{m+1}. \tag{A.13}$$

Now, we treat the three terms on the right-hand side of Eqn. A.13 separately.

The last term is less than 1, by inspection

$$\left(\frac{1-p}{1-p/8}\right)^{m+1} < 1. \tag{A.14}$$

The second term may be recast as an exponential,

$$\left(\frac{(1-p/8)^8}{1-p}\right)^N = \exp\left(N\left(8\ln(1-p/8) - \ln(1-p)\right)\right). \tag{A.15}$$

We make use of two properties of the natural logarithm.

$$\ln(1-x) < -x \tag{A.16}$$

$$-\ln(1-x) < \frac{x}{1-x}. \tag{A.17}$$

From these, we have have the following inequality,

$$8\ln(1-p/8) - \ln(1-p) < \frac{p^2}{1-p}, \tag{A.18}$$

and thus, from Eqns. A.15 and A.18,

$$\left(\frac{(1-p/8)^8}{(1-p)}\right)^N < \exp\left(\frac{Np^2}{1-p}\right). \tag{A.19}$$

Now we turn to the first term on the right-hand side of Eqn. A.13,

$$\left(\frac{B(m+1,N-m)}{8^{m+1}B(m+1,8N-m)}\right) = \frac{\Gamma(N-m)\Gamma(8N)}{8^{m+1}\Gamma(N)\Gamma(8N-m)}. \tag{A.20}$$

We use the well known property of gamma functions, for integer m,

$$\frac{\Gamma(z+m)}{\Gamma(z)} = (z+m-1)(z+m-2)\cdots(z). \tag{A.21}$$

Thus, we obtain

$$\begin{aligned}\frac{B(m+1,N-m)}{8^{m+1}B(m+1,8N-m)} &= \frac{(8N-1)(8N-2)\cdots(8N-m)}{(8N-8)(8N-16)\cdots(8N-8m)} \\ &= \frac{(1-1/8N)(1-2/8N)\cdots(1-m/8N)}{(1-1/N)(1-2/N)\cdots(1-m/N)} \\ &< \frac{1}{(1-1/N)(1-2/N)\cdots(1-m/N)}.\end{aligned} \tag{A.22}$$

Now, we use Eqn. A.17 again to obtain

$$\begin{aligned}-\ln(1-1/N)(1-2/N)\cdots(1-m/N) &< \sum_{k=1}^{m}\frac{\frac{k}{N}}{1-\frac{k}{N}} \\ &< \frac{1}{N-m}\sum_{k=1}^{m}k,\end{aligned} \tag{A.23}$$

which implies,

$$-\ln(1-1/N)(1-2/N)\cdots(1-m/N) < \frac{m(m+1)}{2(N-m)}. \tag{A.24}$$

Therefore,

$$\frac{B(m+1,N-m)}{8^{m+1}B(m+1,8N-m)} < \exp\left(\frac{m(m+1)}{2(N-m)}\right). \tag{A.25}$$

We now combine Eqns. A.13, A.14, A.19, and Eqn. A.25 to obtain

$$\frac{D_{8N,m}(p/8)}{D_{N,m}(p)} < \exp\left(\frac{Np^2}{1-p}+\frac{m(m+1)}{2(N-m)}\right). \qquad \text{Q.E.D.} \tag{A.26}$$

A.1.2. Proof of Lemma 2.

For this lemma, we take a different approach, and return to the definition of $D_{N,m}(p)$ in terms of the binomial distribution function,

$$D_{N,m}(p) = 1 - C_{N,m}(p) = 1 - \sum_{i=0}^{m} B_{N,i}(p). \tag{A.27}$$

We define the continuous function,

$$b_{\lambda,i}(n) = B_{n,i}(\lambda/n) = \frac{\Gamma(n+1)}{\Gamma(i+1)\Gamma(n+1-i)} \left(\frac{\lambda}{n}\right)^{i} \left(1 - \frac{\lambda}{n}\right)^{n-i}, \tag{A.28}$$

and observe that with $\lambda = Np$,

$$\begin{aligned} b_{\lambda,i}(N) &= B_{N,i}(p) \\ b_{\lambda,i}(8N) &= B_{8N,i}(p/8). \end{aligned} \tag{A.29}$$

Now we compute the logarithmic derivative of $b_{\lambda,i}(n)$ with respect to n.

$$\frac{d \ln b_{\lambda,i}(n)}{dn} = (\psi(n+1) - \psi(n+1-i)) - \frac{i}{n}\frac{1}{1-x} + (\frac{x}{1-x} + \ln(1-x)). \tag{A.30}$$

Where ψ is the digamma function,

$$\psi(z) = \frac{\Gamma'(z)}{\Gamma(z)}, \tag{A.31}$$

$$x = \frac{\lambda}{n}. \tag{A.32}$$

Properties of the digamma function are well known. In particular, for integer i[34],

$$\psi(n+1) - \psi(n+1-i) = \sum_{j=0}^{j<i} \frac{1}{n-j}. \tag{A.33}$$

Clearly each of the terms in this sum is positive, and there are precisely m terms. Furthermore, the smallest term is $\frac{1}{n}$, so we may place a lower bound.

$$\psi(n+1) - \psi(n+1-i) > \frac{i}{n}. \tag{A.34}$$

The last term in parentheses in Eqn. A.31 may be expanded in a power series,

$$\begin{aligned} \frac{x}{1-x} + \ln(1-x) &= (x + x^2 + x^3 + \cdots) - (x + \frac{x^2}{2} + \frac{x^3}{3} + \cdots) \\ &= \frac{x^2}{2} + \frac{2x^3}{3} + \frac{3x^4}{4} + \cdots \\ &> \frac{x^2}{2}. \end{aligned} \tag{A.35}$$

Thus, from Eqns. A.30, A.34, and A.35 we conclude

$$\frac{d \ln b_{\lambda,i}(n)}{dn} > \frac{i}{n} - \frac{i}{n} \frac{1}{1-x} + \frac{x^2}{2}. \tag{A.36}$$

Simple algebraic manipulation of the terms on the right-hand side of Eqn. A.36 gives

$$\frac{d \ln b_{\lambda,i}(n)}{dn} > \frac{x}{2n(1-x)} (\lambda(1-x) - 2i). \tag{A.37}$$

Finally, we recall that we are concerned with the derivative of $b_{\lambda,i}(n)$ only for $N \leq n \leq 8N$, in which case

$$1 - \frac{\lambda}{n} > 1 - \frac{\lambda}{N} \tag{A.38}$$

or

$$1 - x > 1 - p. \tag{A.39}$$

Thus, we conclude that

$$\frac{d \ln b_{\lambda,i}(n)}{dn} > \frac{x}{2n(1-x)} (Np(1-p) - 2i), \tag{A.40}$$

and, finally,

$$\frac{d \ln b_{\lambda,i}(n)}{dn} > 0 \qquad \text{for all } n > N,\ Np(1-p) > 2i. \tag{A.41}$$

Since the logarithmic derivative of $b_{\lambda,i}(n)$ is positive, $b_{\lambda,i}(n)$ must be a strictly increasing function of n, for $n > N$. Therefore,

$$\begin{aligned} b_{\lambda,i}(8N) &> b_{\lambda,i}(N) \qquad \text{if } Np(1-p) > 2i, \\ B_{8N,i}(p/8) &> B_{N,i}(p) \qquad \text{if } Np(1-p) > 2i. \end{aligned} \tag{A.42}$$

According to the premise of the Lemma, $Np(1-p) > 2m$, so Eqn. A.42 may be applied to each term in the following sum,

$$\begin{aligned} \sum_{i=0}^{m} B_{8N,i}(p/8) &> \sum_{i=0}^{m} B_{N,i}(p) \\ C_{8N,m}(p/8) &> C_{N,m}(p). \end{aligned} \tag{A.43}$$

Thus, we finally have

$$D_{8N,m}(p/8) < D_{N,m}(p). \qquad \text{Q.E.D.} \tag{A.44}$$

A.1.3. Proof of Lemma 3.

First, we show by induction that

$$f(A^k x) < \exp\left(\frac{\delta_1}{x^{\alpha_1}} \sum_{i=0}^{k-1} \frac{1}{A^{i\alpha_2}}\right)\left(f(x) + \frac{\delta_2}{x^{\alpha_2}} \sum_{i=0}^{k-1} \frac{1}{A^{i\alpha_2}}\right). \tag{A.45}$$

From the premise of the Lemma, Eqn. A.3 we have

$$f(A^{k+1} x) < \exp\left(\frac{\delta_1}{x^{\alpha_1}} \frac{1}{A^{k\alpha_2}}\right)\left(f(A^k x) + \frac{\delta_2}{x^{\alpha_2}} \frac{1}{A^{k\alpha_2}}\right). \tag{A.46}$$

We now assume that Eqn. A.45 holds for the induction variable equal to k. Thus, we obtain

$$\begin{aligned} f(A^{k+1} x) < \exp\left(\frac{\delta_1}{x^{\alpha_1}} \frac{1}{A^{k\alpha_2}}\right) \exp\left(\frac{\delta_1}{x^{\alpha_1}} \sum_{i=0}^{k-1} \frac{1}{A^{i\alpha_2}}\right) \times \\ \left(f(x) + \frac{\delta_2}{x^{\alpha_2}}\left(\sum_{i=0}^{k-1} \frac{1}{A^{i\alpha_2}}\right) + \frac{\delta_2}{x^{\alpha_2}} \frac{1}{A^{k\alpha_2}}\right). \end{aligned} \tag{A.47}$$

Simple algebraic rearrangement of Eqn. A.47 results in

$$f(A^{k+1}x) < \exp\left(\frac{\delta_1}{x^{\alpha_1}}\sum_{i=0}^{k}\frac{1}{A^{i\alpha_2}}\right)\left(f(x)+\frac{\delta_2}{x^{\alpha_2}}\sum_{i=0}^{k}\frac{1}{A^{i\alpha_2}}\right), \tag{A.48}$$

which verifies the induction. The first step of the induction, i.e., $k = 0$, is asserted by the premise of the Lemma, Eqn. A.5. Thus, Eqn. A.48 is proved for all $k \geq 0$.

We now note that the summations in Eqn. A.44 are bounded for $A > 1.$ and $k \geq 0$, i.e.,

$$\sum_{i=0}^{k}\frac{1}{A^{i\alpha}} < \frac{A^{\alpha}}{A^{\alpha}-1}. \tag{A.49}$$

Therefore, for all $k \geq 0$ and $x > x_0$, we have

$$f(A^k x) < exp(\Delta_1)(\Delta_2 + f(x)), \tag{A.50}$$

where

$$\begin{aligned}\Delta_1 &= \frac{\delta_1}{x_0^{\alpha_1}}\frac{A^{\alpha_2}}{A^{\alpha_2}-1},\\ \Delta_2 &= \frac{\delta_2}{x_0^{\alpha_1}}\frac{A^{\alpha_2}}{A^{\alpha_2}-1}.\end{aligned} \tag{A.51}$$

Now let

$$x = A^{k_1}x_1, \tag{A.52}$$

$$k_1 = \lfloor \log_A \frac{x}{x_0} \rfloor \geq 0. \tag{A.53}$$

Then we have

$$x_0 \leq x_1 \leq Ax_0, \tag{A.54}$$

and from the premise, Eqn. A.4,

$$f(x_1) < a_0. \tag{A.55}$$

Combining Eqns. A.50, A.52, and A.55, we have

$$f(x = A^{k_1}x_1) < exp(\Delta_1)(\Delta_2 + a_0) \qquad \text{for all } x > x_0. \qquad \text{Q.E.D.} \tag{A.56}$$

A.1.4. Proof of Theorem

We prove the theorem by demonstrating that the the function,

$$c(N) = \frac{C_{avg}}{N}, \tag{A.57}$$

satisfies the conditions on $f(x)$ in Lemma 3. We obtain a tight bound by careful selection of the constants, δ_1, δ_2, and a_0.

First, we refer to Figure 2.7, from which it is clear that the constant a_0 exists, and that Eqn. A.4 is satisfied with $x_0 > 4000$. In the following, we let

$$p_d = 8^{-d}. \tag{A.58}$$

Then

$$\begin{aligned} c(8N) &= \sum_{d=0}^{\infty} \frac{1}{8Np_d} D_{8N,m}(p_d) \\ &= \frac{1}{8N} + \sum_{d=1}^{\infty} \frac{1}{Np_{d-1}} D_{8N,m}(p_d) \\ &= \frac{1}{8N} + \sum_{d=0}^{\infty} \frac{1}{Np_d} D_{8N,m}(p_d/8). \end{aligned} \tag{A.59}$$

Now we choose a depth, d_0, and split the sum into two parts,

$$c(8N) = \frac{1}{8N} + \sum_{d=0}^{d_0-1} \frac{1}{Np_d} D_{8N,m}(p_d/8) + \sum_{d=d_0}^{\infty} \frac{1}{Np_d} D_{8N,m}(p_d/8). \tag{A.60}$$

We select the depth, d_0, so it has the following properties:

$$Np_{d_0-1}(1 - p_{d_0-1}) > 2m, \tag{A.61}$$

$$Np_{d_0}(1 - p_{d_0}) < 2m. \tag{A.62}$$

If $N > 32m$, then we can be sure that

$$p_{d_0} < \frac{1}{8}. \tag{A.63}$$

From Eqn. A.62, we can also show that

$$\begin{aligned} N^2 p_{d_0}^2 (1-p_{d_0})^2 &< 4m^2, \\ \frac{N p_{d_0}^2}{1-p_{d_0}} &< \frac{4m^2}{N(1-p_{d_0})^3}, \\ \frac{N p_{d_0}^2}{1-p_{d_0}} + \frac{m(m+1)}{2(N-m)} &< \frac{4m^2}{N(1-p_{d_0})^3} + \frac{m(m+1)}{2(N-m)}, \\ \frac{N p_{d_0}^2}{1-p_{d_0}} + \frac{m(m+1)}{2(N-m)} &< \frac{1}{N}\left(\frac{4m^2}{(1-p_{d_0})^3} + \frac{m(m+1)}{2(1-\frac{m}{N})}\right), \end{aligned} \tag{A.64}$$

and, using Eqn. A.63,

$$\frac{N p_{d_0}^2}{1-p_{d_0}} + \frac{m(m+1)}{2(N-m)} < \frac{1}{N}\left(m(6m+\frac{1}{2})\right). \tag{A.65}$$

Equation A.63 tells us that the premise of Lemma 2 is satisfied for all the terms in the first summation in Eqn. A.62. Thus, we may write

$$c(8N) < \frac{1}{8N} + \sum_{d=0}^{d_0-1} \frac{1}{Np_d} D_{N,m}(p_d) + \sum_{d=d_0}^{\infty} \frac{1}{Np_d} D_{N,m}(p_d) \left(\exp\left(N^2 p_{d_0}^2 (1-p_{d_0})^2\right)\right). \tag{A.66}$$

We now use Eqn. A.65, to obtain

$$\begin{aligned} c(8N) &< \frac{1}{8N} + \sum_{d=0}^{d_0-1} \frac{1}{Np_d} D_{N,m}(p_d) + \exp\left(\frac{m(6m+\frac{1}{2})}{N}\right) \sum_{d=d_0}^{\infty} \frac{1}{Np_d} D_{N,m}(p_d) \\ &< \frac{1}{8N} + \exp\left(\frac{m(6m+\frac{1}{2})}{N}\right) c(N). \end{aligned} \tag{A.67}$$

Equation A.67 is formally identical to Eqn. A.3 with

$$\alpha_1 = \alpha_2 = 1, \tag{A.68}$$

$$A = 8, \tag{A.69}$$

$$\delta_1 = m(6m+\frac{1}{2}), \tag{A.70}$$

$$and \quad \delta_2 = \frac{1}{8}. \tag{A.71}$$

Thus, we have verified the conditions of Lemma 3, and the conclusion follows. The constants, c_m, in Table 2.1 are obtained simply by evaluating the right-hand side of Eqn. A.5.

A.2. Bounds on D_{avg}

Figure 2.10 suggests that $D_{avg}(N)$ is bounded by $\log_8 \frac{d_m N}{m}$. An argument very similar to the one in the previous section shows that this is, indeed, the case. In this section, we will be slightly less rigorous than in the previous section. The proof will be outlined, but not presented in as much detail. This time, we define a function,

$$d(N) = \frac{m}{N} 8^{D_{avg}(N)}, \tag{A.72}$$

which is plotted in Figure 2.10. Clearly,

$$D_{avg}(N) < \log_8 \frac{d_m N}{m} \tag{A.73}$$

follows from

$$d(N) < d_m, \tag{A.74}$$

so we need to prove that $d(N)$ is bounded by a constant.

We begin by placing a bound on $D_{avg}(8N)$,

$$\begin{aligned} D_{avg}(8N) &= \sum_{d=0}^{\infty} D_{8N-1,m-1}(p_d) \\ &= 1 + \sum_{d=0}^{\infty} D_{8N-1,m-1}(p_d/8) \\ &< 1 + \exp(\frac{c}{N}) D_{avg}(N), \end{aligned} \tag{A.75}$$

where c is a constant, independent of N. Some algebraic manipulations suffice to prove by induction that Eqn. A.75 implies

$$D_{avg}(8^k N) < \sum_{i=0}^{k-1} \exp\left(\frac{c}{N} \sum_{j=1}^{i} \frac{1}{8^{k-j}}\right) + \exp\left(\frac{c}{N} \sum_{j=0}^{k-1} \frac{1}{8^j}\right) D_{avg}(N). \tag{A.76}$$

We now bound the sums inside the exponentials in Eqn. A.76, using Eqn. A.49, and obtain

$$D_{avg}(8^k N) < \sum_{i=0}^{k-1} \exp\left(\frac{8}{7}\frac{c}{N}\frac{1}{8^{k-i}}\right) + \exp\left(\frac{8}{7}\frac{c}{N}\right) D_{avg}(N). \qquad (A.77)$$

Now, we use the fact that the exponential of a small argument is close to unity. More precisely,

$$\exp(x) - 1 < (\exp(x_1) - 1) * \frac{x}{x_1} \qquad x < x_1. \qquad (A.78)$$

This allows us to rewrite Eqn. A.77 as

$$D_{avg}(8^k N) < k + \left(\exp(\left(\frac{8}{7}\frac{c}{N}\right) - 1\right) \sum_{i=0}^{k-1} \frac{1}{8^{k-i}} + \exp\left(\frac{8}{7}\frac{c}{N}\right) D_{avg}(N), \quad (A.79)$$

and, again, using Eqn. A.49, we have

$$D_{avg}(8^k N) < k + \frac{1}{7}\left(\exp(\left(\frac{8}{7}\frac{c}{N}\right) - 1\right) + \exp\left(\frac{8}{7}\frac{c}{N}\right) D_{avg}(N). \qquad (A.80)$$

We define ϵ, as

$$\epsilon = \exp(\left(\frac{8}{7}\frac{c}{N}\right) - 1, \qquad (A.81)$$

and using Eqn. A.81, we have

$$d(8^k N) < 8^{\frac{k}{7}} 8^{(1+\epsilon) D_{avg}(N)} \qquad k > 0. \qquad (A.82)$$

Equation A.82 holds for all $k > 0$. It is clear from Figure 2.10 that the right-hand side of Eqn. A.82 is bounded by a constant, designated d_m, for $4000 < N < 32000$. It follows that

$$d(N_1) < d_m \qquad N_1 > 4000 \qquad (A.83)$$

because for any $N_1 > 4000$, it is possible to find an integer k and a value $4000 < N < 32000$, for which $N_1 = 8^k N$. From Eqn. A.83, we have

$$D_{avg}(N) < \log_8 d_m N \qquad N > 4000. \qquad \text{Q.E.D.} \quad (A.84)$$

B. Shell Formation Using 180k Particles...

Reprinted by permission from "Dynamics and Interactions of Galaxies", ed. R. Wielen, 216–218, Springer-Verlag, (1989). J. Salmon, P.J. Quinn, M. Warren. "Using Parallel Computers for Very Large N-Body Simulations: Shell Formation Using 180K Particles"

Using Parallel Computers for Very Large N-Body Simulations: Shell Formation Using 180 K Particles

J. Salmon[1], *P.J. Quinn*[2], *and M. Warren*[1]

[1]Caltech Concurrent Computing Project

[2]Mt. Stromlo and Siding Spring Observatories, The Australian National University, Woden Post Office, ACT 2606, Canberra, Australia

We have used a parallel version of the Barnes-Hut C treecode running on a parallel supercomputer (a 512 node NCUBE) at the Caltech Concurrent Computing Project to follow the evolution of systems with very large numbers of particles (N>180,000). Many important problems in astrophysics demand large N and in particular we have studied the fully selfconsistent formation of stellar shells around an elliptical parent galaxy via the merger with another elliptical of one tenth its mass. The final system shows a spread in shell radii similar to that in observed shell galaxies. Major changes in the structure and dynamics of the primary have resulted from the merger.

Large N problems : the motivation

Traditionally, N-body simulations of collisionless systems have been run with the largest number of particles possible given the limitations on machine time and memory. This push to large N was motivated by the need to control the collisionality of the simulation over the desired number of dynamical times. The arrival of methods with computational times that scaled like $N Log N$ instead of N^2 (PM, PPPM, Treecodes) meant that for a fixed computing resource,10,000 - 100,000 particles could be evolved with high dynamical resolution and in runs of only a few tens of hours on machines like the CRAY 1. In trying to achieve very large N, most of the NlogN codes run up against serious memory constraints on machines like the CRAY X-MP . In an attempt to address this memory problem we have developed the Barnes-Hut treecode on a parallel architecture in which individual nodes have substantial amounts of memory. The 512 node NCUBE at the Caltech Concurrent Computing Project has 512K of memory per node giving a total machine memory of 256 MgB which is more than ten times the typical maximum process size on the X-MP.

Tree Codes on Parallel Computers

The basic problem in designing a parallel algorithm is the decision of "who gets what data, when?" Given an appropriate organization of the data, termed a "decomposition", one is usually able to carry over the basic sequential algorithm with little or no change. That is, each processor behaves as though it is performing the original sequential algorithm on a subset of the complete data set.

The simplest decomposition to consider is one in which each processor is assigned all the bodies in a region of space. The processor boundaries are chosen to assign approximately equal work loads to the processors. Given a decomposition of the particles which balances the load, it is tempting to think we can construct the tree on each processor, merge them into a collective tree which is copied to each processor and proceed with the force evaluation. Unfortunately, this leads to severe problems with the available memory. One is limited to treating problems whose entire tree fits into the 512 kbytes available to each processor. Thus, this approach provides extremely good utilization of the parallel processor when the total number of bodies is less than a few thousand, but it doesn't work at all for large numbers of bodies.

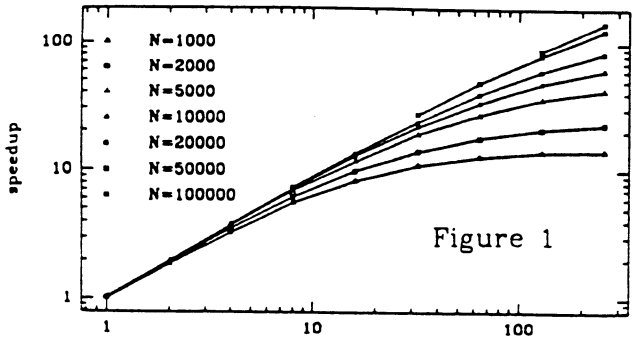
At this point, it is crucial to realize that by the very nature of the Barnes-Hut(1986) algorithm, it is not necessary to store the entire tree in each processor. Since each processor is concerned with computing forces only within a restricted region of space, it will need to access the most detailed levels of the tree only in a restricted region. Distant regions of space may be accounted for with much less data.

Thus, the parallel force calculation algorithm proceeds as follows :

(a) Determine processor boundaries by recursive bisection of the estimated work.
(b) Build the tree for the local particles.
(c) Communicate to obtain only those details of other processors' trees which may be necessary for force computations.
(d) For each local particle descend the tree and compute the force.

The performance of the parallel algorithm may be measured several ways. Of primary interest is the overall performance. Counting all of the overheads, load imbalances, i/o operations, etc., the system computes the force on our 180,000 bodies in approximately 200 seconds (θ = 0.8,monopole). In addition, it is of interest how this figure scales with the number of processors. Obviously, we were not able to evaluate the forces on a 180,000 body

Dynamics and Interactions of Galaxies Editor: R. Wielen



system on a single processor with 512 kbytes of memory. Nevertheless, extrapolation from a series of runs with fewer particles indicates that we obtained a speedup of 300 on 512 processors. Figure 1 shows the speedup resulting from the use of parallel processing as a function of the number of processors, as well as the number of bodies. It is typical of such plots that for a fixed problem size, the speedup begins to level off and the addition of more processors does little to reduce the overall running time. On the other hand, if the problem size is allowed to grow with the number of processors, the speedup continues to increase up to very large numbers of processors. Thus, large parallel computers are generally well matched to large problems, but are a poor match to small problems.

Shell Formation

Since the early 1980s, stellar shells around ellipticals have been recognized as the product of a collision between a large elliptical and a compact and/or dynamically cold companion galaxy. Models of shell formation by Quinn (1984), Dupraz and Combes (1986) and Hernquist and Quinn (1988) have all used rigid potential models for the primary and hence have ignored important selfconsistent effects like dynamical friction and tidal stripping. As a result several important features of shells galaxies, like the small radii of the innermost shells, have remained unexplained. In order to address important dynamical effects such as the orbit decay and progressive tidal distruction of the companion as well as the selfconsistent effects on the primary, we have undertaken a study of shell formation using the treecode on the NCUBE.

Figure 2 shows the distribution of companion particles near the end of the simulation. The core of the companion stayed intact for five pericentre passages and was destroyed in the final pass. The following is a summary of the main points of the simulation with regard to shell production :

- O Shell making particles are removed from the companion after each close encounter with the core of the primary. As a consequence the final shell distribution is complex, consisting of several overlapping shell systems.
- O Since the companion core continues to lose orbital energy over the course of the simulation, the final shells are confined to smaller radii than the shells that resulted from earlier pericentre passages . The innermost shells

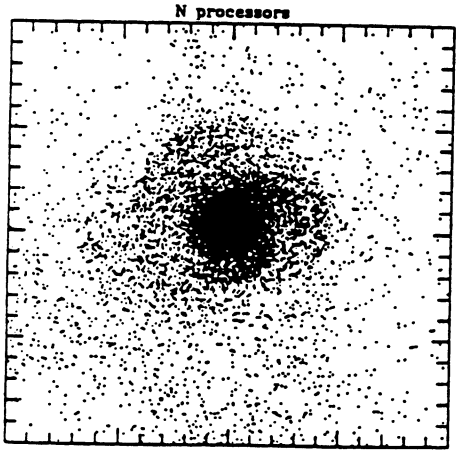


Figure 2 : The distribution of companion particles at the end of the simulation seen from about the original orbit plane. The box is 15 half-light radii of the initial primary on an edge.

are within 0.3 effective radii and the outer shells extend to more than 20 effective radii which gives a shell radius range similar to many observed shell galaxies.

- Because of the complex nature of the multiple shell system, it is very difficult in this case to apply the techniques outlined by Hernquist and Quinn (1987) to probe the underlying potential using the distribution of shell radii.

Conclusion

Many problems in galaxy formation and evolution will require the use of very large numbers of particles in N-body simulations. The model of shell formation we ran has addressed a number of outstanding issues in shell formation; in particular the radius of the inner most shells. The simulation has also given us valuable new insights into the effects shell making mergers have on the primary galaxies. These effects may explain many generic features of ellipticals (photometric and kinematic axis alignment, rotation curve shape, see Quinn and Zurek, this volume). The simulation also teaches us about the complex process of tidal disruption and the dynamics of mergers. By comparing this simulation with others on the formation of counterrotating cores by Balcells and Quinn (1989) and the cosmological simulations of Quinn, Salmon and Zurek(1986), we can start to see common processes emerge that may tell us a great deal about the way mergers work in general and how they may have influenced the properties of galaxies along the Hubble sequence.

Acknowledgements

We would like to thank the Caltech Concurrent Computing Project for providing travel funds and CPU time on the NCUBE. This work was supported in part by Department of Energy, Applied Mathematical Sciences grant DE-FG03-85ER25009.

References

Balcells, M. and Quinn, P.J., 1989, preprint.
Barnes.J. and Hut,P., 1986, *Nature*, **324**, 446.
Dupraz, C. and Combes, F., 1986, *Astr. & Ap.*, **166**, 53.
Hernquist, L. and Quinn, P.J., 1987, *Ap.J.*, **312**, 1.
Hernquist, L. and Quinn, P.J., 1988, *Ap.J.*, **331**, 682.
Quinn, P.J., 1984, *Ap.J.*, **279**, 596.
Quinn, P.J. and Zurek, W.H., 1988, *Ap.J.*, **331**, 1.
Quinn, P.J. and Zurek, W.H., 1989, preprint.
Quinn, P.J., Salmon, J.K. and Zurek, W.H., 1986, *Nature*, **322**, 392.

C. Primordial Density Fluctuations...

Reprinted from Nature, Vol. 322, No. 6077, pp. 329-335, 24 July 1986

Primordial density fluctuations and the structure of galactic haloes

P. J. Quinn*†‡, J. K. Salmon†‡ & W. H. Zurek‡

N-body models used to study the formation of structure in $\Omega = 1$ universes reveal that the mass profiles of the collapsed structures—galactic haloes—are intimately related to the power spectrum of initial, gaussian, density perturbations. In particular, flat rotation curves observed in disk galaxies are obtained only when the exponent of the power law $P(k) \sim k^n$ is, on the megaparsec scale, less than -1, but not as small as -3. These results have important implications for various cosmological models.

THE structures observed in the Universe on the galactic and larger scales are believed to have been seeded by primordial density perturbations[1]. These perturbations were presumably imprinted much earlier by a process such as inflation[2-4]. Later, they were modified in the course of the radiation-dominated era (redshift $z > 2.5 \times 10^4 \Omega h^2$). Both this initial imprinting, and the more recent modifications depend sensitively on the composition of matter in the Universe and on the physics at very high energies. Therefore, an understanding of the formation of structure from initial, seed perturbations is essential for cosmology as well as for particle physics.

We now explore, by means of computer modelling, the formation of collapsed and virialized structures in Einstein–deSitter ($\Omega = 1$) universes. We focus on a class of model universes characterized by gaussian, scale-free density perturbations. Their power spectra:

$$P(k) = Ak^n \tag{1}$$

are completely specified by the normalization constant A and the exponent n. k is the wavenumber. The white noise case corresponds to $n = 0$. Scale-free perturbations in an $\Omega = 1$ universe result in hierarchical clustering[5-10], which is both reasonably easy to analyse and often a good approximation of more complicated and more realistic models. We shall also simulate the formation of haloes in CDM (cold dark matter) universes[11-13]. Indeed, we shall normalize our scale-free power spectra so that on a 'galactic scale' they have similar power to the CDM case. The influence of Ω on the structure of galactic haloes will be treated elsewhere (W.H.Z., P.J.Q. and J.K.S., manuscript in preparation).

Observations of rotational velocities in disk galaxies indicate that the bulk of the mass of these objects is invisible 'dark matter'. Moreover, this dark matter is distributed so that the rotational velocities of stars, H II regions and neutral hydrogen[14-18] are approximately independent of radius out to distances near to and even well beyond the optical radius of the galaxy. This implies a density (ρ) of dark material that, at large radii, falls off like;

$$\rho(r) \sim r^{-2} \tag{2}$$

or alternatively, the mass profile:

$$M(r) \sim r \tag{3}$$

We will demonstrate that mass profiles of this kind can be obtained when the initial density perturbations have at least as much energy on the large scale as they have on the small scale, $n < -1$. This conclusion has been anticipated analytically by the 'secondary infall' models proposed by Gunn and Gott[19]. In particular, using the secondary infall paradigm, Hoffmann and Shaham[20] have conjectured that the average density profile should be related to the power-law exponent n by

$$\rho(r) \sim r^{-\gamma} \tag{4}$$

where

$$\gamma = \frac{3(3+n)}{(4+n)} \tag{5}$$

for $n > -1$, and should relax to $\rho \sim r^{-2}$ for $-1 > n > -3$. Our results are in approximate agreement with these conclusions for $n > -2$, even though the process of formation of collapsed objects does not seem to conform with the secondary infall picture.

Computer simulations

A gaussian field of initial density perturbations can be reproduced by writing:

$$\Delta\rho(\vec{r}) = \rho(\vec{r}) - \langle\rho\rangle = \sum_{\vec{k}} |\delta_{\vec{k}}|\, e^{-i\phi_{\vec{k}}}\, e^{-i\vec{k}\cdot\vec{r}} \tag{6}$$

Here $|\delta_{\vec{k}}|$ are the absolute values of the complex amplitudes of all the modes and $\phi_{\vec{k}}$ are their phases. One can realize power spectrum $P(k)$ by choosing:

$$\delta_{\vec{k}} = \sqrt{2P(k)}\, R^g_{\vec{k}} \tag{7a}$$

$$\phi_{\vec{k}} = 2\pi R^\phi_{\vec{k}} \tag{7b}$$

where $\{R^g_{\vec{k}}, R^\phi_{\vec{k}}\}$ are a pair of random numbers with distributions which are, respectively, gaussian in the interval $(0, \infty)$, and uniform in the interval $(0, 1]$.

We have found it extremely useful to implement different power spectra by using the same set of random number pairs. This allows us to model 'the same' fragment of the universe with different power laws and therefore to bring out systematic changes which would otherwise be overwhelmed by the random nature of the initial perturbations.

The density distribution can be faithfully represented in the form given by equation (6) only when, on the scales of interest, the amplitude $\sigma = \sqrt{\langle(\Delta\rho)^2\rangle}$ is much smaller than the average density;

$$\sigma \ll \langle\rho\rangle \tag{8}$$

This follows not just from the trivial requirement that ρ be positive; beyond the linear regime mode–mode interactions induce correlations between modes which in turn invalidates

* Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, Maryland 21218, USA.
† Theoretical Astrophysics, California Institute of Technology, Pasadena, California 91125, USA
‡ Theoretical Astrophysics, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA.

the assumption of gaussian density perturbations and makes the use of equation (6) impossible (see ref. 1).

We guarantee the validity of equation (6) by adjusting the normalization constant A in equation (1) so that mass fluctuations on the scale-of-interest (Λ) are small:

$$\left[\frac{\delta M}{M}\right]_{\Lambda} = 0.06 \tag{9}$$

We impose the same condition on $\delta M/M$ for the CDM models. It will be useful but will entail no loss of generality, to refer to this chosen scale Λ by analogy with the CDM case—for which A can be estimated by comparison with present-day galaxy correlation functions[12,13,21]—as the 'megaparsec scale'. In other words, we anticipate that 'galactic haloes' will form from Λ-sized fragments of the initial distribution after a time interval corresponding to the Hubble time. With this convention in mind we can now state that our models are initiated with the Hubble constant $H_o = 100\ \mathrm{km\ s^{-1}\ M\ pc^{-1}}$ at $z_{\mathrm{init}} = 24$ and are evolved for ~10,000–14,000 Myr beyond the conventional present which for this set of initial conditions occurs at an unrealistically short time of 6,700 Myr after the big bang. It is essential to keep in mind that for the power law models the above convention is nothing but a convention as the scale-free nature of $P(\vec{k})$ implies. Our results—density profiles of haloes—are independent of the value of H_0. The Hubble constant influences only the total values of the masses of haloes, but not their shapes.

The initial conditions are imprinted on a system of 64^3 particles, each of them with a conventional mass of $m = 1.02 \times 10^9 M_\odot$, $\Omega = 1$. They represent a $(10\mathrm{Mpc})^3$ section of the model universe. The initial conditions are imposed using the Zel'dovich 'growing mode' method[1,20,22] which deforms the cubic lattice of particles in the desired manner. The evolution of the system to $z = 0$ is then simulated by a Fourier (cloud-in-cell) code with a 64^3 mesh.

The Fourier methods are inaccurate on scales smaller than a few mesh spacings. Therefore, we use it only to set up initial conditions for the N-body simulations. These are provided by the output of the Fourier code at $z = 5.25$ and/or $z = 10.1$. (We do not want to generate initial conditions this late in the history of the universe, as by then the inequality (8) is no longer valid and equation (6) cannot be trusted.) The use of the Fourier code allows us to employ sufficiently many particles at very early times to imprint initial conditions without the danger of introducing errors via shot noise or aliasing (ref. 21 and work in preparation). Moreover, using this technique we can obtain the first 'blurred' but, nevertheless, informative pictures of the model at $z = 0$.

Two kinds of N-body runs are used. Low-resolution simulations use particles with a mass $M_{\mathrm{L}} = 27\mathrm{m}$. A sphere with a diameter of 10 Mpc cut out from the Fourier cube provides the initial conditions. Each of the low resolution particles has the location and the velocity of the centre of mass of a $3 \times 3 \times 3$ fragment of the original Fourier lattice. However, particle masses are now close to $3 \times 10^{10} M_\odot$ which means that a typical galactic halo will contain no more than ~100 particles. Therefore, to investigate smaller scale structures we have employed a different mapping of the Fourier particles onto the N-body initial conditions.

High-resolution runs use a one-to-one mapping in a chosen sphere of 2 Mpc radius, and a very low-resolution mapping (64 Fourier particles to one N-body particle) towards the outskirts of the model. Masses in the transition region between 2 and 3 Mpc take on intermediate values. The resulting multi-resolution system is evolved using the N-body code for a time comparable to the corresponding low-resolution run (see Fig. 1).

Typically 5,000–7,000 particles are evolved and a run is completed within 2–4 hrs of Cray-1 CPU time. Usually more than one high-resolution sphere-of-interest is chosen for each low-resolution model. Therefore, the same fragment of the universe is modelled using three distinct resolutions. All of the N-body runs use a smoothing length of 10 kpc. Note that on the scales on which all three resolutions are expected to be accurate, all of them yield comparable results. Low-resolution runs are then used to extract information about the large-scale structure, mergers, and so on. The results of these runs will be discussed elsewhere. Here we shall concentrate on the small-scale information contained in the high-resolution runs. Specifically, we shall consider the density profiles of haloes, that is objects with masses between $10^{11} M_\odot$ and $10^{12} M_\odot$.

Halo density profiles

Figure 2 contains a graphic summary of the key results discussed in this section. Two conclusions are immediately apparent: (1) the high-resolution portions of the particle plots in Fig. 2 contain related structures (the 'same' halo can be usually identified in the figures corresponding to the neighbouring values of n). (2) The realizations with more power on small scales (larger n) have larger numbers of more compact haloes. In particular, while the model with $n = -2.75$ exhibits only two reasonably well developed haloes, there are about 10 such structures of various sizes for $n = 1$. Moreover, models with a lot of power on the large scale exhibit more pronounced 'caustics' and 'filaments'. The CDM model is the closest in appearance to the $n = -1$ and $n = -2$ cases. This is not unexpected: the effective exponent of the cold dark matter power spectrum is close to -1.5 on the megaparsec scale.

Most of the haloes in the high resolution region contain only high-resolution particles. However, some of them are contaminated by heavy particles that have entered from the low-resolution part of the model. The results given below are inferred from haloes that contain $\leqslant 10\%$ of these heavies by mass. (In fact, only 10% of the analysed halos had more than 1 or 2 heavies).

For the densities given by equation (4), the mass profile (the mass within radius r) is given by the relation:

$$M(r) \sim r^{3-\gamma} \tag{10}$$

The corresponding rotational velocity inferred for such $M(r)$ is;

$$v(r) \sim r^{(2-\gamma)/2} \tag{11}$$

The circular rotational velocities of haloes obtained in the computer experiment are for a subset of the haloes shown in Fig. 2. We also indicate there the velocity profile predicted by equation (5).

Models with $n = 1$ and $n = 0$ produce centrally condensed haloes with the mass falling off faster than in the flat rotation case. The typical slope of the density profile is well approximated by the value of γ inferred from equation (5) although a systematic discrepancy can also be noted: densities tend to fall off faster than equation (5) would have it. In particular, rotation curves are not quite flat for $n = -1$ but clearly flatten for CDM as well as for the $n = -2$ case. In all cases when $n > -1$ rotation curves flatten at radii between 10 and 20 kpc. (The resolution of our N-body code is 10 kpc and does not allow us to comment on the structure on the innermost parts of the halo.) It is interesting that virtually all haloes formed in a simulation with a given power law have very similar rotation curves.

The situation is different when $n = -2.75$. Now the rotational velocities are still rising at $r = 50$ kpc and some of them do not flatten even in the 100 kpc range. This seems to imply that the structure of these haloes is inconsistent with the conclusions of Hoffman and Shaham, who conjecture that flat rotation curves will form even for $n = -3$. It is unlikely that such haloes with rising rotation velocities are a temporary, unstable phenomenon: We have checked the slope of the density profile at $z = -0.5$, at a time equal to 14,000 Myr. It is still rising and equation (5) still provides as good an estimate as a flat rotation profile. More relevant in understanding this discrepancy is probably the systematic increase of the size of the soft core with n decreasing to -3. This trend is visible already for $n = -2$. Indeed, rotation profiles between 100 and 200 kpc, not shown in Fig. 2, are quite

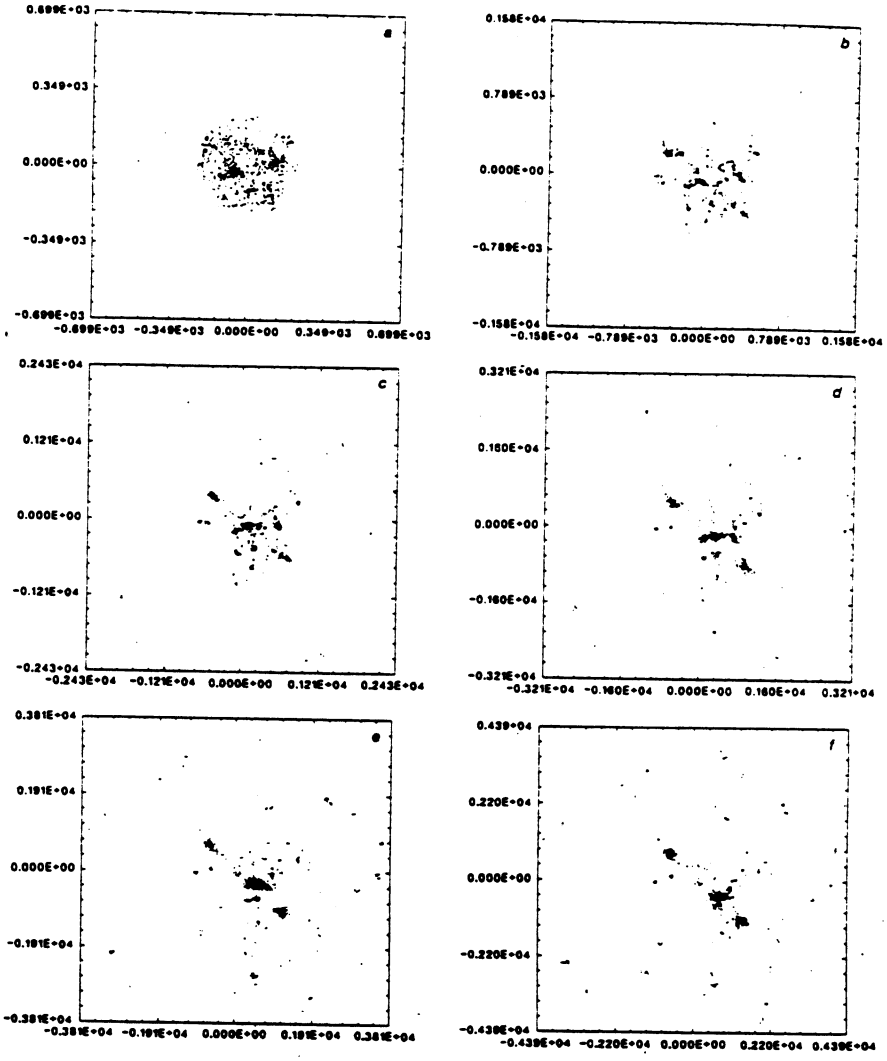


Fig. 1 Six photographs taken in comoving coordinates at equal time intervals during the evolution of the high-resolution model ($P(k) \sim k^{-1}$). The initial conditions ($z = 5.25$) for the N-body run as obtained from the Fourier code (which was started with gaussian density perturbations at $z = 24$), are shown in *a*. Note that there are sizeable regions with large overdensities inside the high-resolution sphere, and a slightly perturbed lattice of massive particles on the outside. *b–e*, The state of the system at intermediate times. *f*, $z = 0$. Only the inner, high-resolution parts of such models were used (see Fig. 2). Units are kiloparsecs.

flat for $n = -2.75$. Therefore, the relevant part of Fig. 2 may be showing just the soft core parts of the haloes for this case. Moreover, the treatment given in ref. 20 breaks down for $n < -3$. Hence, it is not too surprising that its predictions for $n = -2.75$ are inaccurate. Indeed, one should be surprised that the arguments of Hoffman and Shaham give as good an estimate of the resulting density profiles as they do, as the assumptions on which they are based are only approximately valid.

The mass profile of the cold dark matter haloes is reasonably close to the flat rotation curve variety out to 100–150 kpc. Beyond that distance rotation profiles begin to fall off. The rotation parameters (λ) of the collapsed objects are typically in the 0.05–0.1 range ($\lambda = |E|^{1/2} J / GM^{5/2}$, where E is the total energy of the halo, J its angular momentum, M its mass, and G is the gravitational constant.) This is consistent with our earlier findings[24,25] as well as with a number of other cold dark matter simulations[26].

We have concluded that the mass profiles of collapsed, virialized objects in the $\Omega = 1$ universe initiated with an effective power law spectrum on the galactic scale are determined over the range of interest by the value of n. Equations (4) and (5) give a fair estimate of the corresponding density profiles. Although a power spectrum translates unambiguously into a rotation curve, one should keep in mind that the reverse argument is not so straightforward: The fact that a flat rotation curve is observed, often allows, within the errors, a range of halo density profiles. Moreover, baryonic material in the dissipative process of settling in the centre of the core will inevitably steepen the final rotation curve[27].

Comparisons with theory

Gunn and Gott have developed the secondary infall scenario based on the assumption that the formation of collapsed objects in an overdense region begins from the density peak which in due course accretes surrounding shells of material[19] Each of these shells is thought to be much less massive than the already collapsed and virialized core. Assuming spherical symmetry, one can show that material from each consecutive shell will be eventually deposited at a radius:

$$r = \frac{r_m(\beta - 1)}{2} \qquad (12)$$

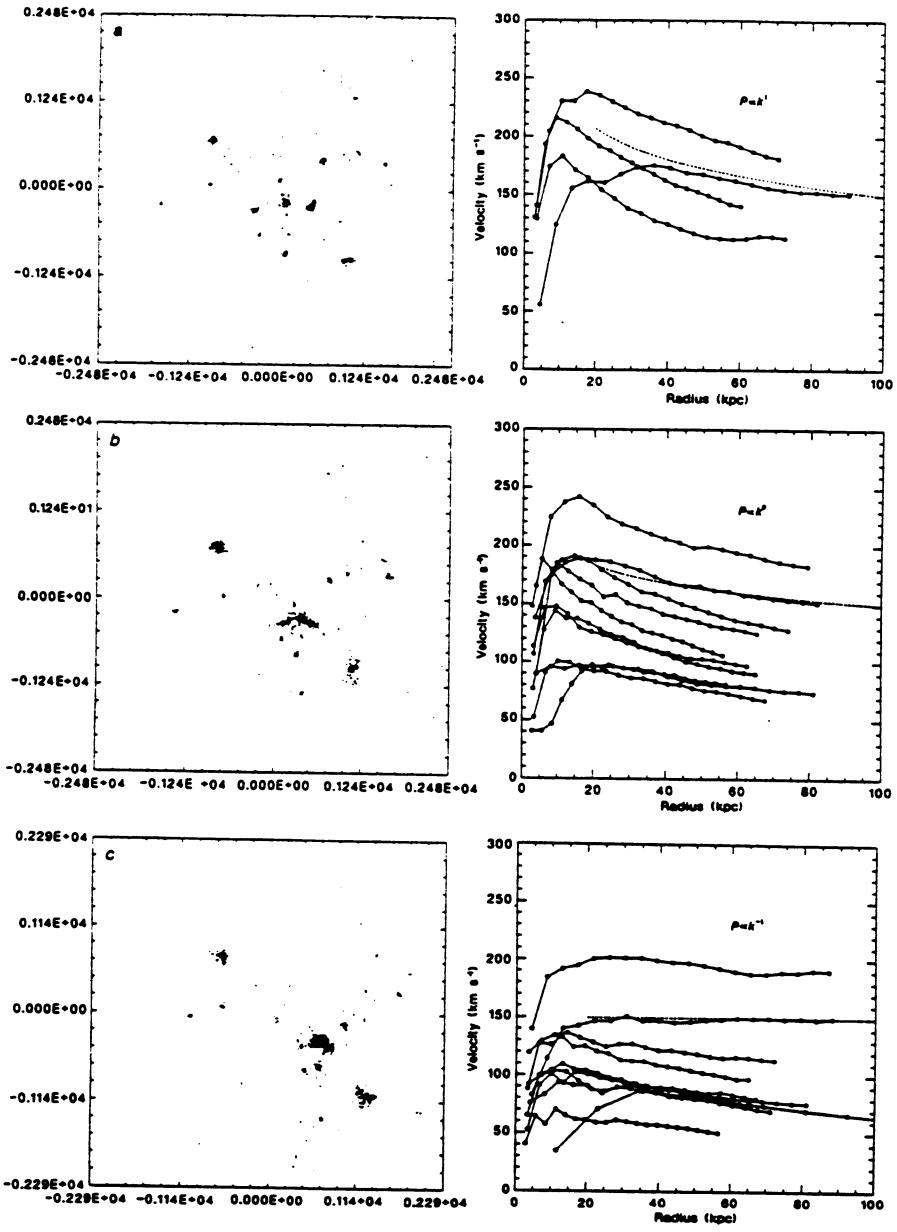


Fig. 2 The formation of haloes in an Einstein-deSitter ($\Omega = 1$) universe with different power spectra. Particle plots show only the inner, high-resolution regions. The exponent of the power law is indicated in the plots of the rotation velocities. The rotation profile predicted by $\rho(r) \sim r^{-3(3+n)/(4+n)}$ is indicated by dashed line. A scaled down average rotation profile of Sc spirals, ref. 15, is plotted along with the CDM haloes.

around the centre, where r_m is the radius of maximum expansion, and β characterizes the density of the already virialized core:

$$\rho(r) \sim r^{-\beta} \tag{13}$$

With a few additional assumptions, this can be used to calculate the density profile. The maximum radius of expansion for a spherically symmetric density perturbation with the initial relative overdensity δ_i is given by:

$$r_m = r_i \frac{1+\delta_i(r_i)}{\delta_i(r_i) - (\Omega_i^{-1} - 1)} \tag{14}$$

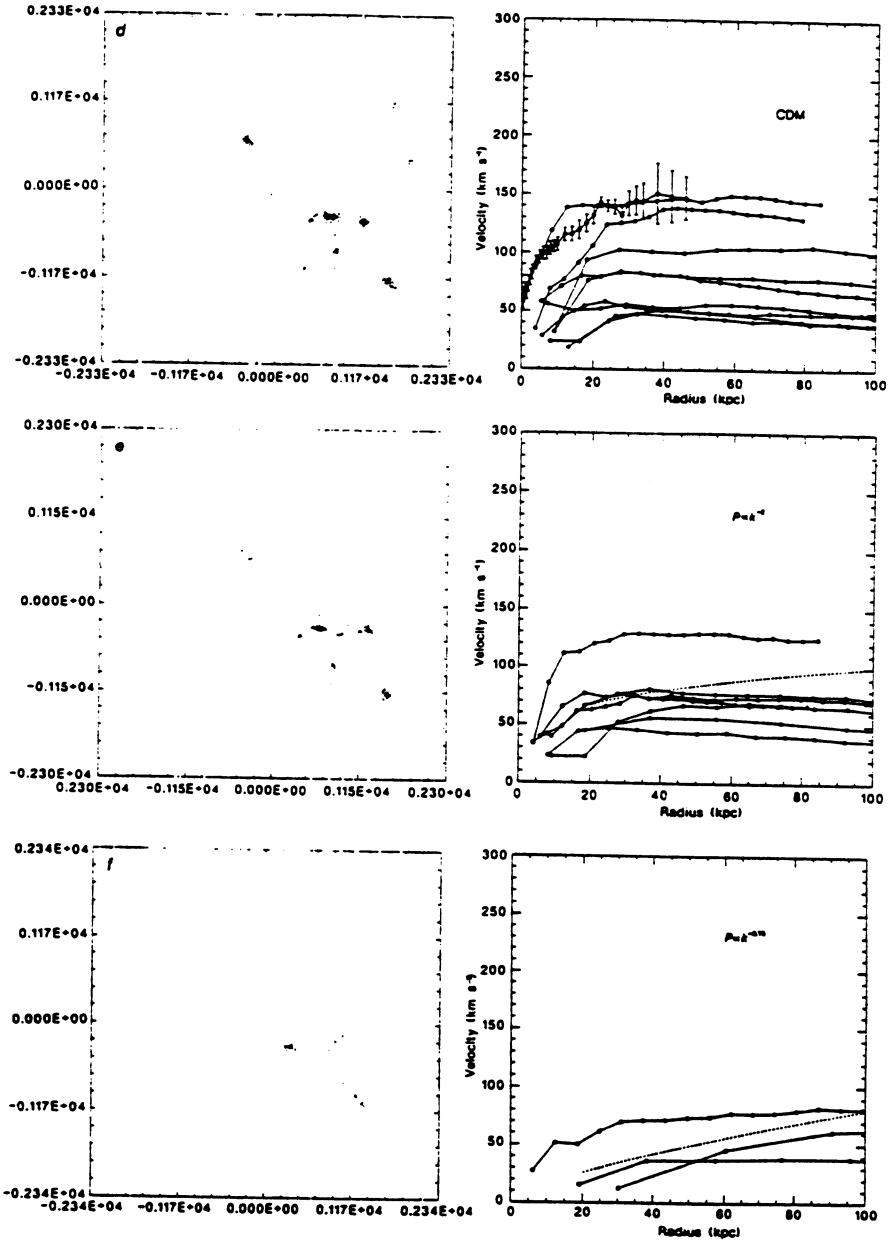
Here δ_i characterizes the initial overdensity inside the sphere of radius r_i:

$$M(r < r_i) = \left(\frac{4\pi}{3}\right) r_i^3 \langle\rho\rangle (1 + \delta_i(r_i)) \tag{15}$$

In the case considered here $\Omega_i = 1$. Therefore, the final density of the halo is:

$$\rho(r) \sim \delta_i(r_i(r))^3 \tag{16}$$

Hoffman and Shaham[20], following Peebles[12], argue that in a universe with a scale-free spectrum of initial density perturba-



tions, given by equation (1), the relative overdensity around a local maximum can be estimated by:

$$\delta_i \sim \left(\frac{r_i}{r_\Lambda}\right)^{-(3+n)} \tag{17}$$

Here r_Λ is the characteristic smoothing distance[12,20,28], related to the distance between the peaks. Consequently, the final density of the collapsed halo is given by:

$$\rho(r) \sim (r_i(r))^{-3(3+n)} \tag{18}$$

To express the right-hand side of this relation in terms of the distance from the halo centre we use equation (14) together with equation (17) to obtain:

$$r = \frac{r_m(\beta-1)}{2}$$

$$\sim \frac{(\beta-1)}{2}\left(\frac{r_i}{r_\Lambda}\right)^{4+n} \tag{19}$$

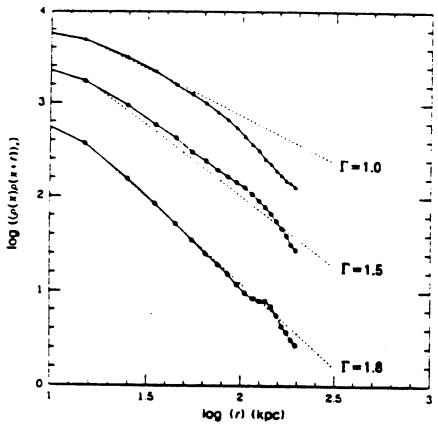


Fig. 3 Density autocorrelation functions, equation (21), for the particles in the high-resolution portions of some of the power law models. The prediction of the hierarchical clustering scenario—power law with the exponent given by equation (22)—is indicated by the dashed line. Results for: ▲, $n=-2$; ●, $n=-1$; ○, $n=0$ respectively. Note that each curve has been offset vertically by an arbitrary amount for clarity.

Therefore,

$$\rho(r)\sim\left(\frac{r}{r_\lambda}\right)^{-3(3+n)/(4+n)} \tag{20}$$

Our derivation of equation (20) paraphrases a more detailed discussion in ref. 20.

Hoffman and Shaham, following Fillmore and Goldreich[29], conjecture that the mass profiles with $M\sim r^{1+\varepsilon}$, $\varepsilon>0$, are unstable. Therefore, they conclude that haloes with $\gamma<2$ will relax into flat rotation curve haloes. This additional conjecture has not been so far confirmed by our computer models. The key difference between the situation analysed by Fillmore and Goldreich and our computer simulations may be due to the absence of angular momentum in the spherically symmetric, self-similar calculations reported in the ref. 29. Large angular momenta may stabilize haloes with rising rotation curves.

The analysis leading to equation (20) hinges on assumptions which are at best only approximately valid: (1) The initial overdensity profile, equation (17), is a reasonable estimate of the actual density profile only for a limited range of initial radii. When $r<r_\lambda$, near the peak, $\delta_i(r_i)$ is flatter than equation (17) would suggest. Moreover, for larger r, fluctuations of overdensity soon begin to exceed the systematic behaviour given by equation (17)[20,28]. Therefore, the derivation is based on an assumption which is approximately accurate only for a rather limited range of radii. (2) A detailed discussion by Bardeen *et al*[28] shows that the criterion implicitely used by Hoffman and Shaham to define density maxima is biased towards unusually broad peaks. (3) The density profile around correctly chosen density maxima falls off more steeply than equation (17) would have it. This difference may account for the discrepancy between the rotation curves predicted on the basis of equation (20) and the steeper rotation profiles obtained in our simulations. (4) The paradigm of the secondary infall onto a peak does not appear to be valid in the course of collapse leading to the formation of the haloes considered here. In particular, merging of smaller, already collapsed objects (see Fig. 1) appears to be a better description of the formation process. This is at odds with the picture of shells deposited spherically symmetrically onto a pre-existing core. In spite of all these problems equations (4) and (5) give a reasonably accurate prediction of the density profiles of the collapsed objects. Discrepancies, which we have already pointed out in the previous section, are worth further study.

Scale-free spectra lead to a self-similar clustering hierarchy[5-10] Peebles has pointed out that the density autocorrelation function in an evolved region of such a hierarchical universe should have a power law form[6]:

$$\langle\rho(x)\rho(x+r)\rangle_x = r^{-\Gamma} \tag{21}$$

with an 'hierarchical' exponent Γ given by:

$$\Gamma=\frac{3(3+n)}{(5+n)} \tag{22}$$

This Γ differs from the γ in the density formula, equation (5). We have verified (see Fig. 3) that this estimate for Γ is indeed correct in the high-resolution portions of our models. Note, perhaps not unexpectedly, that when the density autocorrelation is calculated only for the particles inside the haloes—that is, for x chosen from within the perimeter of the halo, but for an arbitrary r in the notation of equation (21)—it is considerably steeper, with the effective power law exponent close to the one given by equation (5).

Conclusions

Our numerical experiments provide clear evidence of a direct connection between the form of the power spectrum responsible for the gaussian density perturbations and the rotation curves of the resulting haloes. In particular, $P(k)\sim k^n$ with $n\sim-1\to-2$ leads to flat rotation curves in present day galactic haloes. Power law spectra with $n>-1$ give haloes with rotation curves steeper than observed. This puts severe constraints on the shape of the power spectrum in the range corresponding to 1 Mpc. Furthermore, baryon condensation in the central parts of the halo can be expected to further steepen the rotation profile[27]. This indicates that density perturbations with the preponderance of the power on small scales are untenable as models for galaxy formation. On the other hand, baryon condensation can probably flatten rising rotation curves inside broad soft cores, making $n=-1$ or $n=-2$ attractive. The cold dark matter power spectrum falls into the category of spectra producing haloes compatible with observations. Further studies to determine the environment dependence of other halo properties (such as the radii of their soft cores, angular momenta, and eccentricities) are underway.

We thank Stirling Colgate, Richard Epstein, Mike Fall, Yehuda Hoffman, Craig Hogan, Robert Hotchkiss, Nick Kaiser, Jeremy Ostriker and Martin Rees for stimulating discussion and help. Completion of this project would have been impossible without the large amounts of computer time provided by Los Alamos National Laboratory, Caltech and the Space Telescope Science Institute. We thank Kate Quinn for helping to prepare the manuscript.

Received 14 February; accepted 23 May 1986.

1. Peebles, P. J. E. *The Large Scale Structure of the Universe* (Princeton University Press, 1980).
2. Guth, A. *Phys. Rev.* D23, 347–356 (1981).
3. Linde, A. D. *Phys. Lett.* 108B, 389–393 (1982).
4. Albrecht, A. & Steinhardt, P. J. *Phys. Rev. Lett.* 48, 1220–1223 (1982).
5. Press, W. H. & Schechter, P. *Astrophys. J.* 187, 425–438 (1974).
6. Peebles, P. J. E. *Astrophys. J.* 189, L51–L53 (1974).
7. Gott, J. R. & Rees, M. J. *Astr. Astrophys.* 45, 365–376 (1975).
8. Gott, J. R. & Turner, E. L. *Astrophys. J.* 216, 357–371 (1977).
9. White, S. D. M. & Rees, M. J. *Mon. Not. R. astr. Soc.* 183, 341–358 (1978).
10. Fabian, S. M. in *Astrophysical Cosmology* (eds Bruck, H. A., Coyne, G. & Longair, M.) 191–217 (Pontificia Academia Scientiarum, 1982).
11. Peebles, P. J. E. *Astrophys. J.* 263, L1–L5 (1982).
12. Peebles, P. J. E. *Astrophys. J.* 277, 470–477 (1984).
13. Blumenthal, G. R., Faber, S. M., Primack, J. R. & Rees, M. J. *Nature* 311, 517–525 (1984).
14. Rubin, V. C., Ford, W. K. & Thonnard, N. *Astrophys. J.* 238, 471–487 (1980).
15. Rubin, V. C., Burstein, D., Ford, W. K. & Thonnard, N. *Astrophys. J.* 289, 81–104 (1985).
16. Bosma, A. thesis, Univ. of Groningen (1978).
17. Caroenna, C. & Freeman, K. C. *Astrophys. J.* 294, 494–501 (1980).
18. van Albada, T. S., Bahcall, J. N., Begeman, K. & Sancisi, R. Preprint (Princeton Univ., 1985).
19. Gunn, J. E. & Gott, J. R. *Astrophys. J.* 176, 1–19 (1972).
20. Hoffman, Y. & Shaham, J. *Astrophys. J.* 297, 16–22 (1985).
21. Efstathiou, G., Davis, M., Frenk, C. S. & White, S. D. M. *Astrophys. J. Suppl.* 57, 241–260 (1985).
22. Zeldovich, Ya. B. *Astr. Astrophys.* 5, 84–89 (1970).
23. Davis, M., Efstathiou, G., Frenk, C. S. & White, S. D. M. *Astrophys. J.* 292, 371–394 (1985).
24. Quinn, P. J., Salmon, J. K. & Zurek, W. H. *IAU Symp.* 117 (in the press).
25. Zurek, W. H., Quinn, P. J. & Salmon, J. K. *IAU Symp.* 117 (in the press).
26. Frenk, C. S., White, S. D. M., Efstathiou, G. & Davis, M. *Nature* 317, 595–597 (1985).
27. Blumenthal, G. R., Faber, S. M., Flores, R. & Primack, J. R. *Astrophys. J.* 301, 27–34 (1986).
28. Bardeen, J. M., Bond, J. R., Kaiser, N. & Szalay, A. S. *Astrophys. J.* 304, 15–61 (1986).
29. Fillmore, J. A. & Goldreich, P. *Astrophys. J.* 281, 1–8 (1984).

D. Rotation of Halos in Open and Closed Universes...

Reprinted by permission from The Astrophysical Journal, **330**, 519–534, July 15, 1988. W.H. Zurek, P.J. Quinn, J.K. Salmon, "Rotation of Halos in Open and Closed Universes: Differentiated Merging and Natural Selection of Galaxy Types."

THE ASTROPHYSICAL JOURNAL, 330:519–534, 1988 July 15

ROTATION OF HALOS IN OPEN AND CLOSED UNIVERSES: DIFFERENTIATED MERGING AND NATURAL SELECTION OF GALAXY TYPES

W. H. ZUREK
Theoretical Astrophysics, Los Alamos National Laboratory
P. J. QUINN
Space Telescope Science Institute
AND
J. K. SALMON
Theoretical Astrophysics, California Institute of Technology
Received 1987 September 4; accepted 1988 January 5

ABSTRACT

We use computer simulations to study the properties of galactic halos formed from Gaussian initial density perturbations in both open and closed ($0.2 < \Omega_0 < 8$) dark matter universes. The specific angular momenta of the forming halos are consistent with those observed in spirals but are about one order of magnitude larger than those found in the baryonic components of ellipticals. The rotational properties are almost completely uncorrelated with halo densities or masses and with the density of the large-scale environment characterized by Ω_0. However, the central densities of the halos depend sensitively on the density of their environment. This and the observed dependence of the morphological type on the environment suggest that the structure of the luminous, baryonic components in galaxies is closely related to the central halo density. Denser halos, which occur in denser environments, must allow baryons to lose their angular momentum and form compact, diskless objects with the morphology of elliptical galaxies.

Our simulations suggest that the required selective loss of angular momentum by the baryons occurs in mergers of well-differentiated protogalaxies with condensed-out stellar cores. By contrast, mergers of less differentiated progenitors reheat the gaseous component. Hot gas settles into a disk on a cooling time scale with negligible losses of angular momentum. Such mergers result in spirals with bulges that have formed from the amalgamated stellar cores of their partially differentiated progenitors.

Subject headings: cosmology — dark matter — galaxies: formation — galaxies: internal motions — galaxies: structure

I. INTRODUCTION

The origin of the morphological differences between spirals and ellipticals is a long-standing puzzle (Sandage, Freeman, and Stokes 1976; Gott and Thuan 1976; Fall and Efstathiou 1980; Faber 1982; Kashlinsky 1982). One obvious distinction between the luminous, stellar parts of the two types of galaxies is their specific angular momentum which can be inferred directly from observed velocity and luminosity distributions. For a typical spiral, the ratio of baryonic angular momentum to baryonic mass $[J/M]_B$ is ~ 6 times that of an elliptical of comparable luminosity (Fall 1983) where we have chosen to label collectively all the luminous material in galaxies as baryons. Another way to quantify this difference involves the dimensionless spin parameter $\lambda = J|E|^{1/2}/GM^{5/2}$, where E is the total energy and G is Newton's constant. For ellipticals, the typical value of $\lambda_B \sim 0.05$, while for spirals $\lambda_B \sim 0.5$. Therefore, one is tempted to conjecture that an obvious origin for the morphological differences between ellipticals and spirals is the initial angular momentum content of the material out of which the galaxies were formed. This "obvious" hypothesis faces, however, a number of difficulties.

The spin parameter λ predicted theoretically (Peebles 1969; Doroshkievich 1970) and observed in numerical experiments (Efstathiou and Jones 1979; Zurek, Quinn, and Salmon 1987; Barnes and Efstathiou 1987) falls in the range $\lambda \sim 0.02$–0.08 which is quite reasonable for the luminous parts of ellipticals but an order of magnitude too small for the luminous parts of spirals. This fact should be regarded more as evidence that λ_B for the luminous material can change in the course of the collapse to form a galaxy rather than as an inherent difficulty: Dissipation may be able to increase the binding energy enough to change λ_B by an order of magnitude (Efstathiou and Silk 1983). However, dissipation of energy is not enough to account for the observed differences in λ_B: elliptical and spiral galaxies have similar effective radii at a given luminosity (or mass) (Fall 1980). Hence, they have collapsed by similar factors, which in turn implies that their luminous parts must have dissipated comparable amounts of energy. Therefore, the question: "How did λ_B for ellipticals end up with more or less its initial value in spite of a similar loss of energy due to dissipation?" remains open. The only obvious answer is that, somehow, the formation process for ellipticals has affected both J_B and E_B, while in spirals J_B was left largely untouched. Consequently: *baryons lose their angular momentum when forming ellipticals, but conserve it when forming spirals.* At this preliminary stage of our argument, this statement is only a reasonable guess. Below, in § II, we will show that the specific angular momentum of halos is too large by about an order of magnitude to account for ellipticals. This additional evidence will allow us to raise the status of our guess about the dissipation of J_B to a "conjecture." Proof will probably demand more simulations which account for the dissipative baryonic material and a much better knowledge of the star formation process than is currently available (Bodenheimer 1981). At this moment we

can nevertheless already state that the λ_g value observed in the luminous components of galaxies is likely to be quite different from the λ of the dark halo.

The second difficulty one encounters in trying to trace the origins of the distinction between spirals and ellipticals to the initial angular momentum content of the halos is the strong correlation of galaxy type with environment (Oemler 1974; Dressler 1981; Postman and Geller 1984; Giovanelli, Haynes, and Chincarini 1986). The morphology—environment connection is clearly an important clue about the nature of the galaxy formation process, but, in spite of some early hopes, it does not seem to involve angular momentum. Faber (1982) suggested that the observed morphological selection could be due to an anticorrelation between the spin parameter λ and the average density of the environment. A similar possibility was also considered by Blumenthal *et al.* (1984), who have pointed out that if the high-density peaks—which are known to be more correlated than the "average height" ones—were endowed with smaller values of λ, then the overabundance of ellipticals in clusters would follow. However, a simple argument given by Faber in an appendix of her 1982 paper demonstrated that there is no reason to expect any relation between the peak density and λ. We shall confirm this negative conclusion in § III by showing that the anticorrelation between λ and halo density is negligible, and that the dispersion of the values of λ is much too large to allow small systematic effects (Hoffman 1986) to explain why in clusters $\sim 80\%$ of galaxies are early types, while in field $\sim 80\%$ are late types. Similarly, we shall see in § III that halos which "maintain their identity" in simulations with different values of Ω for the same set of initial Gaussian perturbations show no systematic trends of λ with Ω. We will also confirm the findings of Barnes and Efstathiou (1987) that λ is not strongly correlated with halo mass in contradiction of the claims of Thuan and Gott (1977) that such a correlation should exist.

Section IV contains what we regard as the most important evidence in our attempt to determine which halo parameters may be decisive in choosing the morphological type of a galaxy. There we will present N-body results which prove that the mean central densities of halos are correlated with the large-scale densities of their environments. A simple, analytically derived dependence of the final density of the halo on the initial large-scale overdensity (Zurek, Quinn and Salmon, 1987) is confirmed by the numerical simulations. This suggests that halos forming in overdense regions of future clusters in a flat cold dark matter universe are significantly (factors of 3–10) denser than their field counterparts. This strong correlation suggests that: *the baryonic cores of ellipticals are formed in the deep potential wells of dense halos while spirals form in shallow halos which are more populous in the field. The dynamical process that determines the final spin of the baryons must then be intimately related to the halo density.* Such "natural selection" is forced on the evolution of the galaxy by the environment. It works on two levels; it selects dense halos as the sites for forming ellipticals, and it selects denser cluster environments as the sites of formation of dense halos.

In § V we shall suggest and discuss what we believe is the key difference between the formation process leading to spirals and ellipticals: *both ellipticals and spirals grow to their present sizes through mergers. However, the progenitors of early-type galaxies convert their baryons into stars before merging, while in late-type galaxies mergers occur while the bulk of the baryons is still gaseous.* In § V we shall also discuss what we believe is the origin of the apparent coincidence between the initial value of λ acquired through tidal torquing and the final λ_g of ellipticals. Both fall in the range 0.02–0.08. Yet λ of the dark halo is determined purely by dissipationless dynamics, while λ_g becomes what it is in a process involving the dissipation of both J_g and E_g. We shall venture a speculation that the approximate equality of λ and λ_g is not just a coincidence and argue that this value of λ_g is determined by dissipationless dynamics occurring on the level of collapsed stellar cores of merging subhalos, which are, in the model suggested here, the progenitors of a future elliptical.

In § VI we shall give a summary of the main points of the paper and briefly discuss the status of some of the key assumptions and consequences of our model for the creation of the ellipticals and spirals. There we shall also compare and contrast the model of differentiated mergers with the earlier discussions of the relevance of mergers in determining the galactic morphology.

II. THE SPECIFIC ANGULAR MOMENTUM OF GALACTIC HALOS

We have employed a discrete N-body code to model the evolution of $\sim 10^4$ mass points. "Galactic halos" are collapsed, virialized objects (see example in Fig. 1) of the same type we have described previously in the course of the investigation of rotation curves in scale-free universes (Quinn, Salmon, and Zurek 1986, referred to as QSZ hereafter). A detailed discussion of our methodology is given in QSZ; below we give a brief summary of several key points.

We begin our simulations at a redshift $z = 24$ with a Gaussian density perturbation field characterized by a power-law

$$P \propto k^n , \tag{1}$$

where k denote the wavenumbers of the Fourier modes. The normalization is such that on the scale Λ corresponding to 10% of the length of the edge of our sample "box" of the universe, mass fluctuations $\delta M/M|_\Lambda$ are equal to 0.1 at the initial epoch. The value of Λ corresponds to 1 Mpc (comoving) for a conventional normalization of the cold dark matter spectrum in an $\Omega = 1$, $h = 1$ universe (Blumenthal *et al.* 1984). This fixes our "conventional" unit system which we shall use below. The normalization as well as the choice of Ω and h are somewhat arbitrary (QSZ). Therefore, we shall focus our analysis on these results of our simulations which are expected to be independent of such arbitrary choices. The unit system of the simulations is masses in units of $10^{10}\ M_\odot$ solar masses, lengths in kiloparsecs, time in gigayears, and velocities in kilometers per second.

Following the Zel'dovich approximation we impose only the growing mode of the initial perturbations on a lattice of 64^3 particles at $z = 24$. The early stages of the evolution are carried out with a Fourier code. Subsequently (at the redshift $z \sim 10$ or sometimes $z \sim 5$) we switch from the Fourier code to the N-body code which evolves a subset of 5000–8000 of the Fourier particles inside a sphere of radius 2 Mpc with high spatial resolution. The remainder of the Fourier box is mapped onto a low-resolution shell containing ~ 1000–2000 more massive particles which provide a background. In the conventional units the low-resolution shell reaches out to radius of 5 Mpc.

In the course of the discrete N-body stages of the evolution particles interact with a Newtonian force. For small distances this force law is modified (smoothing length of 10 kpc) which sets a fundamental limit on our resolution. Halos forming in

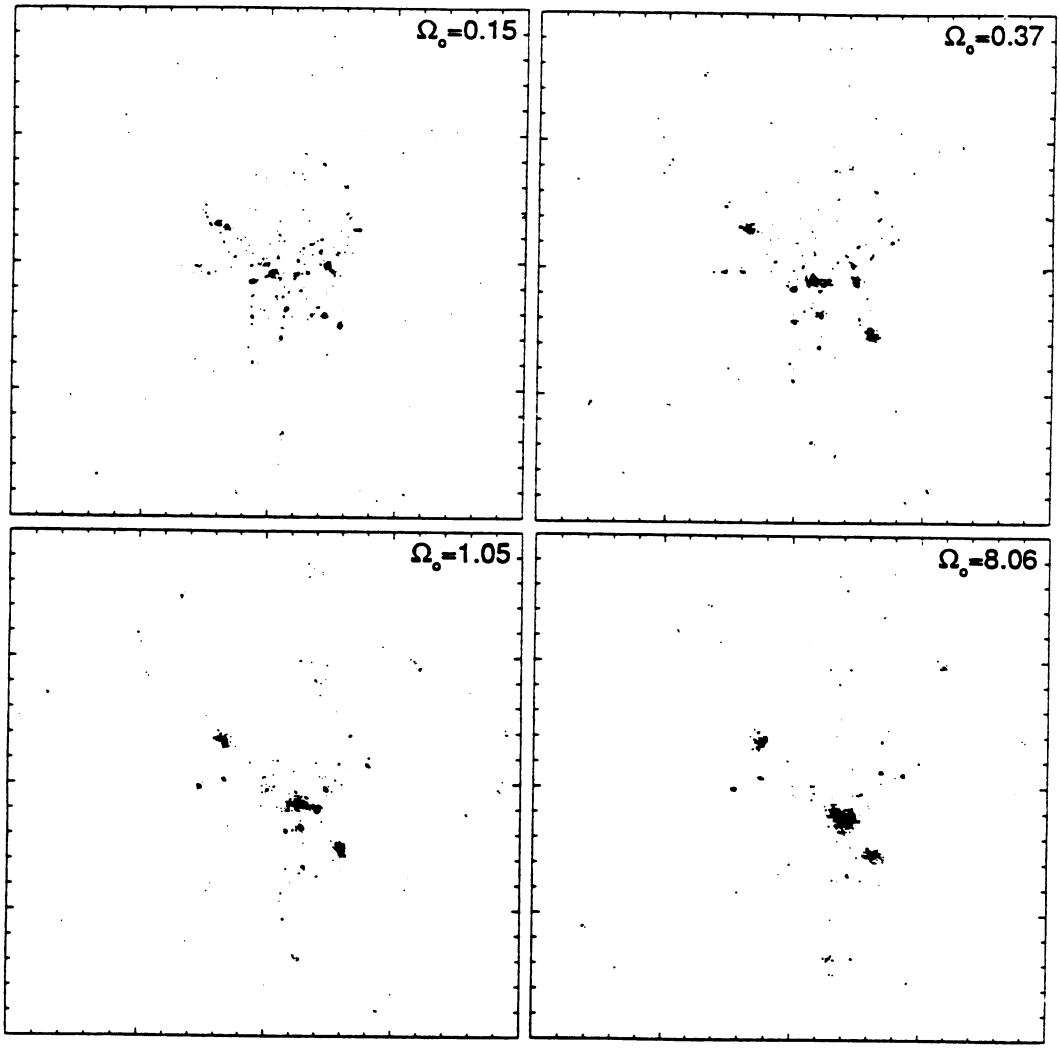


FIG. 1.—Halos formed in universes with different values of Ω. The Gaussian initial conditions were realized using the power spectrum $P(k) \propto k^{-1}$ and Fourier series of density perturbations with the same phases and relative amplitudes of the modes. Note that there are more halos and that they are smaller and denser in comparison with the density of the background for open universes (Ω = 0.15 and 0.37) than in the flat (Ω = 1.05) or closed (Ω = 8.06) universes. The objects which exist individually when Ω < 1 are merged in the Ω > 1 universe at the same epoch. Each panel is ~10 Mpc on an edge and can be thought of as an overdense or underdense box embedded in an Ω = 1, h = 1 universe. In each panel z = 0.

the high-resolution region (Fig. 1) are checked for contamination by low-resolution, massive particles that may get mixed in during collapse. Contaminated halos were found only near the boundaries of the high-resolution sphere and are not considered in this analysis.

Before we focus on specific issues related to the role of the halos in the choice of galaxy type, let us first call attention to the apparent differences between the runs corresponding to Ω_0 = 0.15, 0.37, 1.05, and 8.06. Already "to the naked eye," it is quite obvious that halos are more numerous and more compact in universes with low values of Ω than in similar fragments of universes with Ω > 1. (Here, as in QSZ, we are using the trick of implementing the initial conditions in cases corresponding to different values of Ω with the same set of pseudo random numbers. Therefore the "same" halo is usually formed in universes with different values of Ω. This facilitates a discussion of small trends in halo parameters with Ω, power spectra, etc.).

The halos we shall analyze in the rest of this paper are dense (overdensity in excess of 160) and have no visible substructure. Therefore, they can be thought of as individual galactic halos. This is in contrast to the class of objects investigated recently

by Barnes and Efstathiou (1987), which are typically less overdense ($\delta\rho/\rho \sim 64$ in most instances) and usually have significant substructure.

The angular momentum of each halo is calculated with respect to its center of mass for the region that contains 90% of the total bound mass. Halos typically contain ~100–1000 particles and have conventional masses between $10^{11}\ M_\odot$ and $10^{12}\ M_\odot$. The specific angular momenta of the halos found in all the models are shown in Figure 2, which is the key result of this section. We have made no attempt to differentiate too finely between the points corresponding to different values of Ω and power-law exponent n. No significant systematic trend of J/M with Ω and n is found in the explored range of the investigated parameters ($0.2 \leq \Omega_0 \leq 8.1$ and $-2.75 \leq n \leq 1$).

The results of our simulations are plotted against the estimates of specific angular momentum for spirals and ellipticals given by Fall (1983). It is striking that practically all the halos in Figure 2 have specific angular momenta characteristic of the luminous matter in spiral galaxies (In order to compare with Fig. 1 of Fall where the mass on the horizontal axis is the baryonic mass, we have shifted our masses by a self-consistent factor given by Fall as $\sim(2)^{1/2}/\lambda$ using the customary value of $\lambda = 0.07$. A smaller $\lambda = 0.05$, which is in better accord with our results—see next section—would further increase the gap between the specific angular momenta observed in ellipticals and measured in halos forming in our simulations.) Moreover, few halos which have somewhat smaller—more "elliptical"—J/M do not come from models with large Ω. Rather, there appears to be no definite correlation between the large-scale density and specific angular momentum. Hence, specific angular momentum of the halo cannot account for the prevalence of ellipticals in dense environments. It follows that in order to form ellipticals, the J/M of the baryonic, luminous component must decrease in comparison with the dark halo it inhabits. With Fall (1983), we can consider several possible ways to accomplish this.

Tidal stripping (Fall 1983; Davies *et al.* 1983) of the outermost parts of the forming galaxy is one possibility. It would occur preferentially in clusters and, therefore, if efficient, could account for the high content of ellipticals in dense regions of space. The problem with this suggestion was noted by Fall (1983): stripping of the outer layers of the halo removes angular momentum J and mass M in such proportions that the stripped object "moves" almost parallel to the

$$\left[\frac{J}{M}\right]_B \sim M_B^{3/4} \qquad (2)$$

line (Freeman 1975) on Figure 2. Therefore, unless a halo was to lose ~90% of its initial mass, it could not bridge the gap dividing spirals and ellipticals. This scenario appears rather unlikely.

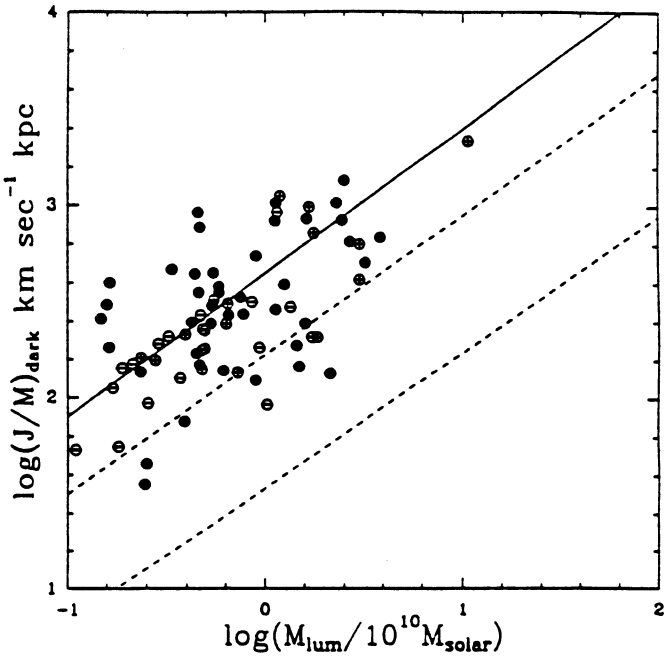


FIG. 2.—Specific angular moments J/M for the objects formed in our simulations with $\Omega = 1$ (●), $\Omega > 1$ (+), and $\Omega < 1$ (−), are plotted as a function of the luminous mass associated with the halo. The ratio of dark to luminous mass is $(2)^{1/2}\lambda^{-1}$ (Fall 1983) where we have taken $\lambda = 0.07$. Solid line has the slope $J/M \propto M^{3/4}$ and is taken from Fall's Figure 1. Dotted lines bracket the region in Fall's diagram occupied by elliptical galaxies. Note that the specific angular momenta of halos formed in our numerical study have J/M values characteristic of spiral galaxies. If, as seems reasonable to assume, tidal torquing imparts the same J/M to both baryonic and dark components of a galaxy, the angular momentum of the luminous parts of the ellipticals must have decreased by about on order of magnitude since the initial collapse. By contrast, the specific angular momentum of the spirals may be close to the initial value.

Mergers (White 1983) are another possible process which could change the angular momentum content or distribution inside a galaxy. We should, however, at the outset carefully differentiate between the two versions of the merging mechanism: mergers of undifferentiated halos without a well-developed central baryonic core and mergers of protogalaxies in which the baryons have condensed out, i.e., with morphologies similar to the present-day galaxies. We shall discuss these two possibilities in detail in § V.

Here let us only note that the halos forming in our simulations have specific angular momenta inconsistent with those seen in ellipticals. Hence, mergers of undifferentiated protogalaxies with mostly gaseous baryons and dark matter distributed in a similar, diffuse fashion do not allow sufficient losses of J to account for the observed properties of ellipticals. Suppose, however, that these progenitors are dense enough to allow baryons to concentrate into "lumps" in their cores and form stars. During the dissipationless merger process, the dynamics of the core material from the progenitors—regardless of whether it is stellar or dark—can be studied using our purely dissipationless N-body techniques. Employing these methods we shall demonstrate in § V that the stellar lumps will sink into the core of the resulting merged object. As the specific angular momentum of the halo material at some radius r is approximately proportional to r (Quinn and Zurek 1988):

$$\left.\frac{dJ}{dM}\right|_r \sim r^{1+\epsilon(n)}, \tag{3}$$

merging of the lumps into the elliptical must be accompanied by the loss of J. (Above ϵ is a small correction which depends on the spectrum of the initial density perturbations. Barnes and Efstathiou 1987 also find—in a study of a somewhat different, not necessarily virialized class of objects—that ϵ is close to 0.) In § V we shall also use numerical simulations to argue more directly that the baryonic material which settles through the dissipationless process that includes relaxation and virialization into the core of the merged object loses angular momentum to the surrounding dark matter halo. Therefore, while neither dissipation nor star formation are incorporated into our present code, N-body results provide us with a very suggestive model for the formation of ellipticals.

This model should be contrasted with the merger of undifferentiated protogalaxies, which is followed by reheating of the gas. Reheated baryons are no longer confined to the core. Therefore, they will settle in to the core of the merged object in a slow, "adiabatic" manner (Blumenthal *et al.* 1986; Ryden and Gunn 1987), with no loss of angular momentum. We shall return to this distinction between mergers of differentiated and undifferentiated progenitors in § V of the paper.

III. IS THE SPIN PARAMETER INFLUENCED BY THE DENSITY OF THE ENVIRONMENT?

On the basis of the numerical simulations our answer to the question posed in the heading would have to be: "perhaps, but not enough to account for the observed dependence of the morphological type of a galaxy on its environment." We shall devote this section to the discussion that justifies such an answer.

Figure 3 shows the spin parameter λ plotted as a function of the halo density and mass at $z = 0$. The striking feature of this figure is the large scatter of the values of λ. The average value of λ over all of the objects that contribute to Figure 3 is

$$\langle\lambda\rangle \sim 0.05 . \tag{4}$$

The scatter of the values of λ around this average is

$$\sqrt{\langle\lambda^2\rangle - \langle\lambda\rangle^2} \sim 0.03 . \tag{5}$$

In spite of the large scatter we have attempted to fit the data corresponding to all the $\Omega = 1$ runs with a power law:

$$\lambda = \rho^s .$$

The resulting s is -1.4 with an error of 0.28 and with χ^2/d.o.f. = 4.42 for the 43 points. The correlation coefficient is, however, only 0.02, and the chance of getting a correlation of this magnitude from 43 uncorrelated points is considerable, $\sim 77\%$.

Points corresponding to halos formed in universes with different values of n and Ω are plotted with different symbols. No obvious systematic trends of λ with any of these parameters can be inferred from the results shown in Figure 3. This is not to claim that such correlations do not exist: a more careful examination of statistical trends may reveal small systematic effects of changes in Ω and n on λ. What is clear, however, is the absence of a strong systematic dependence of the kind seen between n, Ω, and the mass distribution of the halo. (See the next section for a discussion of the dependence of rotation curve shapes and Ω. The trend of rotation curve type and n has already been demonstrated in QSZ.)

Confirmation of this statement can be found in Figure 4. There we have plotted the λ values of several halos which maintain their identity in runs with different values of Ω. Changes in Ω result in a systematic and significant change in halo density. Therefore, if there was a systematic dependence of λ on Ω, one would see it in Figure 4. Yet there is no evidence of such a relation in these plots.

Perhaps more importantly, Figure 4 addresses directly the issue of correlations between the density of the environment and the spin parameter of the halo. A region of larger Ω can be regarded as an overdense region in an Einstein–de Sitter universe (Peebles 1980; Zel'dovich and Novikov 1983). Therefore, if systematic differences in the efficiency of tidal torquing in regions of different large-scale densities existed, one would expect to see an anticorrelation of λ and Ω. There is no sign of such an effect in Figure 4. Hoffman (1986) has argued that there should be an anticorrelation between λ and ρ. However, he also demonstrates (Hoffman 1988*a*) that the scatter of the values of λ is larger than the systematic effects for any reasonable range of halo densities. Indeed, his theoretical predictions of the scatter are consistent with those shown in Figures 3 and 4.

If there is a small systematic anticorrelation between λ and ρ, it appears too small compared to the observed scatter of the values of λ to account for the environmental selection of galaxy types. One should, however, keep in mind that the discussion above provides strong arguments only about the acquisition of angular momentum by halos and not about its subsequent evolution in the environment of large-scale density fluctuations. We have not followed any of the overdense future "clusters" through their collapse and virialization. Therefore, it is still possible that the initially acquired angular momentum of an object that collapsed on the galactic scale may be stripped (Davies *et al.* 1983) or otherwise altered, although we do agree with Fall (1983) that it is hard to imagine that such subsequent evolution could explain the observed degree of morphological dependence of galaxies on their environment, even if stripping somehow specifically singles out baryons.

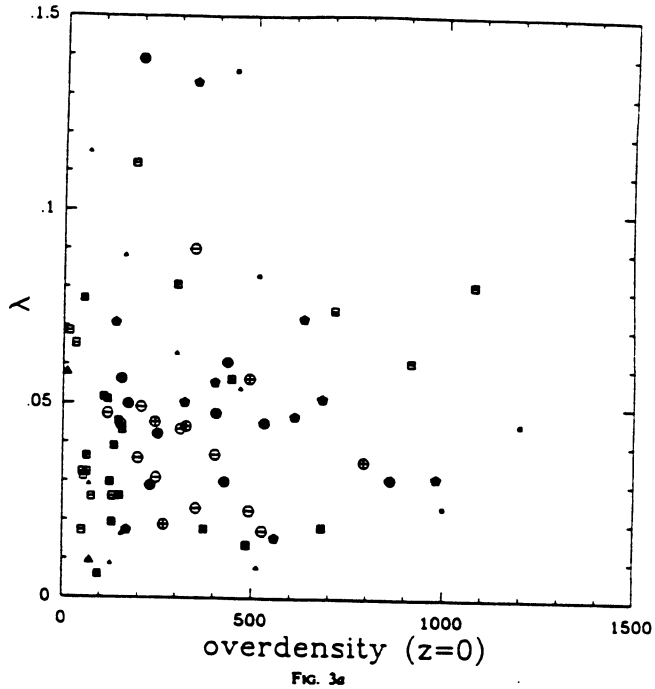


FIG. 3a

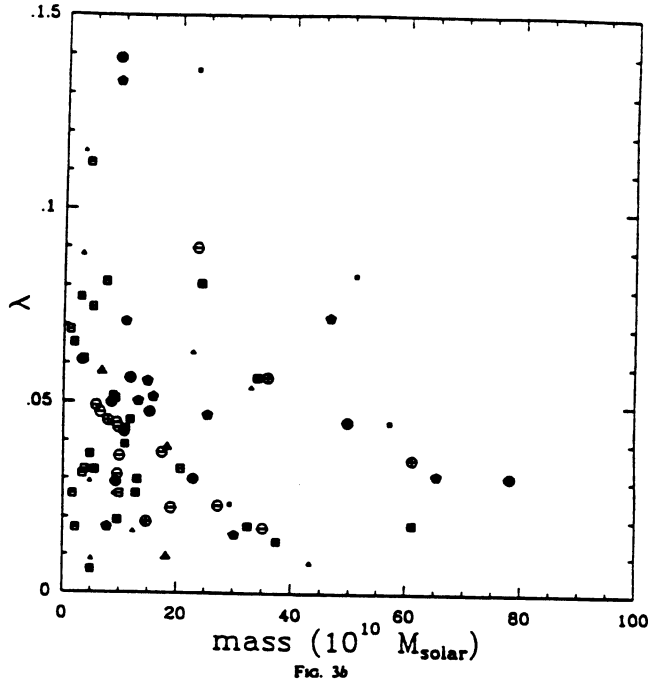


FIG. 3b

FIG. 3.—Rotation parameter λ as a function of (*a*) density and (*b*) total mass of the halos formed in the simulations. Points obtained in the runs with different values of n and different final values of Ω are represented by different symbols. Filled symbols are $\Omega = 1$ runs, symbols containing a minus sign (−) have $\Omega < 1$, and those containing a (+) have $\Omega > 1$. The spectral index of the perturbation spectrum (n) is encoded by circle (−1), large square (−2), large triangle (−2.75), pentagon (0), small square (1), small triangle (CDM). There is no obvious correlation between λ and ρ or λ and mass. The large dispersion of λ vs. ρ makes small systematic decrease of λ as a function of ρ insufficient to explain the observed correlations of morphology with the environment.

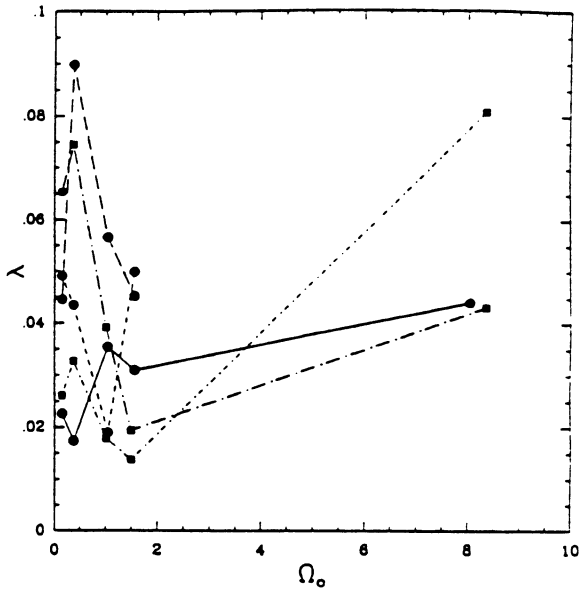


FIG. 4.—Spin parameter λ as a function of Ω for several of the halos which have preserved their identity in runs with different Ω. Squares represent $n = -2$ and circles $n = -1$. Lack of systematic changes of λ with Ω can be also regarded as a lack of sensitivity of λ to the large-scale density of the environment.

IV. THE DEPENDENCE OF MASS PROFILES AND CENTRAL DENSITIES ON Ω

So far we have produced only two "negative" pieces of evidence regarding the connection between angular momentum and the morphology of galaxies: we have shown that the baryonic components of galaxies must have suffered significant changes of specific angular momentum J/M (to account for ellipticals) and of the spin parameter λ (to account for the λ observed in the luminous parts of spirals). We have also shown that the rotational properties of galactic halos have little to do with halo density or with the density of their environment. The specific angular momentum J/M and the spin parameter λ in the baryonic parts of the galaxies must have been altered by about a factor of 6 between the time the halo turned around and the present epoch. Therefore, it is likely that the process which caused changes of J/M and λ is also responsible for the distinction between spirals and ellipticals, and for the environmental selection of the morphological type. The aim of this section is to show that the actual density of the halo as well as its density profile are a sensitive function of the overdensity on some larger scale. To investigate this dependence we shall—as before—simulate the evolution of the "same" universe (same phases in the initial conditions) for different values of Ω.

A sensitive dependence of the final halo density on the large-scale overdensity Δ_i or, alternatively Ω_i, can be best understood using the spherical collapse model. Elementary argument shows that a shell of initial radius r_i with an initial overdensity δ_i inside r_i in a universe which has the average density ρ_{bi} contains a mass

$$M(r < r_i) = \frac{4\pi}{3} r_i^3 \rho_{bi}(1 + \delta_i) \tag{6}$$

and will turn around when its radius reaches the maximal value r_m

$$r_m = r_i\left(\frac{1 + \delta_i}{1 + \delta_i - \Omega_i^{-1}}\right). \tag{7}$$

Above δ_i and Ω_i correspond to the initial instant t_i in the evolution of the perturbation. Arguments leading to equation (7) assume also that at t_i the velocity field is just the Hubble flow; that is, δ_i has not yet induced any peculiar velocities. (In other words, both growing and decaying modes are assumed to be present initially).

We now adopt the basic tenents of the secondary infall model (Gunn and Gott 1972; Fillmore and Goldreich 1984; Bertschinger 1985). That is, we assume that the infalling material will settle after virialization at the radius $r = fr_m$, where f is independent of r_m and is ~ 0.5. Then the final density of the halo is given by

$$\rho_f = \Omega_i \rho_{ci}(1 + \delta_i) f^{-3}\left[\frac{\delta_i - (\Omega_i^{-1} - 1)}{1 + \delta_i}\right]^3. \tag{8}$$

After some simple algebra this can be transformed into

$$\rho_f = \rho_{ci} f^{-3}[(\chi - 1)^3/\chi^2] = \rho_{ci} f^{-3}\eta(\chi), \tag{9}$$

where χ is given by

$$\chi = (1 + \delta_i)\Omega_i, \tag{10}$$

and the critical density $\rho_{ci} = \rho_{bi}/\Omega_i = 1/(6\pi G t_i^2)$ is the average density of a flat ($\Omega_i = 1$) universe.

Figure 5 shows the final density for these halos which have preserved their identity in runs with several values of Ω_i as a function of χ. The linear relation between log (ρ) and log (η)

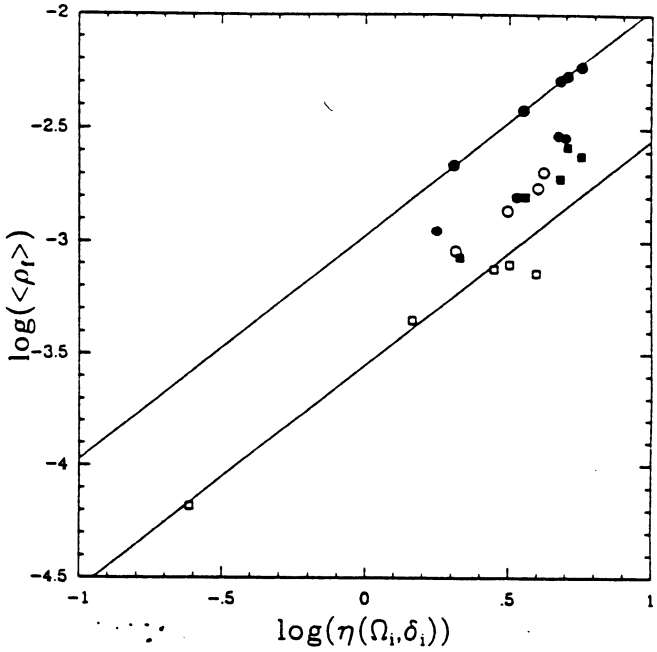


FIG. 5.—Density of the halo as a function of the initial value of the large-scale density parameter Ω_i. Round symbols represent $n = -1$ halos and squares $n = -2$ halos. The quantity plotted on the horizontal axis of the graph, $\eta = (\chi - 1)^3/\chi^2$, where $\chi = (1 + \delta_i)\Omega_i$, increases monotonically with increasing Ω_i for all growing perturbations, $\delta_i > \Omega_i^{-1} - 1$. Lines in the figure have a slope of one, which is the prediction of simple secondary infall. Relative vertical displacement of each halo reflects differences in collapse factors through the term $\rho_{ei} f^{-3}$ in eq. (9a). These graphs are in a good agreement with the simple secondary infall model described in the text which predicts a linear relation with slope 1 in this plane.

holds remarkably well considering the fact that the perturbations which have seeded these objects were the peaks of a Gaussian process and, therefore, were usually significantly aspherical (Bardeen *et al.* 1986). Furthermore, they are subject to tidal interactions with their neighbors and have formed through a process which included mergers of subhalos in a process intermediate between the "violent relaxation," and "secondary infall." In obtaining Figure 5 we have had, however, to recognize that our procedure for imposing initial perturbations is based on the Zel'dovich formalism and, therefore, includes only growing modes. In other words, at the initial instant t_i perturbations in our model have not only a certain initial dimensionless overdensity, but also corresponding velocity perturbation given by

$$\delta v_i \approx \Omega_i^{0.6} H_i r_i \delta_i/3 \,. \tag{11}$$

The above equation holds exactly for $\Omega_i = 1$ and should be understood as relating the peculiar velocity δv_i to the distance r_i from the density peak which will eventually collapse to form a halo. The turnaround radius calculated with the correction implied by equation (11) is therefore:

$$r_m = r_i\left[\frac{1+\delta_i}{1+\delta_i-\Omega_i^{-1}(1-\Omega_i^{0.6}\,\delta_i/3)^2}\right]. \tag{7a}$$

This in turn results in a new version of the formula for the final density of the halo:

$$\rho_f = \Omega_i \rho_{ei}(1+\delta_i) f^{-3}\left[\frac{1+\delta_i-\Omega_i^{-1}(1-\Omega_i^{0.6}\,\delta_i/3)^2}{1+\delta_i}\right]^3 \tag{8a}$$

and

$$\rho_f = \rho_{ei} f^{-3}\{[\chi - (1-\Omega_i^{0.6}\delta_i/3)^2]^3/\chi^2\} = \rho_{ei} f^{-3}\eta(\chi)\,, \tag{9a}$$

where χ is still given by equation (10), but a new η is given by equation (9a).

The agreement between the predictions of the simple "secondary infall model" and the results of our simulations for universes with different values of the density parameter Ω_i can be, in the linear regime ($\delta_i \ll 1$, $\Omega_i^{-1} - 1 \ll 1$), regarded as strong evidence of the dependence of halo parameters on the initial large-scale overdensity $\overline{\Delta}_i$: in the linear regime a sphere used in our simulations can be thought of either as a fragment of a universe with the appropriate value of Ω, or as a fragment of a flat universe ($\Omega = 1$) with the overdensity

$$\overline{\Delta}_i = \Omega_i - 1 \,. \tag{12}$$

Assuming that both δ_i and $\overline{\Delta}_i$ are much smaller than unity, we can rewrite equation (8a) as

$$\rho \sim \rho_{ei} f^{-3}(\delta_i + \overline{\Delta}_i + \delta_i\overline{\Delta}_i)^3/[(1+\delta_i)(1+\overline{\Delta}_i)]^2$$
$$\sim \rho_{ei} f^{-3}(\delta_i + \overline{\Delta}_i)^3/(1+\delta_i+\overline{\Delta}_i)^2 \,. \tag{13}$$

This equation should be compared with equation (8). It has an intuitively obvious meaning: the overdensity of a peak in the flat universe must be measured with respect to the critical density ρ_{ei} and, in the linear regime, is given by the sum of the large-scale and small-scale overdensities.

A comparison of the process of galaxy formation in the field ($\overline{\Delta}_i \le 0$) and in the clusters ($\overline{\Delta}_i > 0$) is an obvious application of

the phenomenon discussed above. Galactic halos collapsing in a larger, overdense region—which, before the present epoch, would have collapsed and formed a cluster—would typically have a larger overdensity than galactic halos forming in the future "field" ($\overline{\Delta}_i \leq 0$).

The cold dark matter universe with canonical $\Omega = 1, h = 0.5$ (Blumenthal *et al.* 1984) provides an excellent example of such an effect. The spectrum of fluctuations, $\delta M/M$, is quite flat on galactic scales and begins to steepen substantially only near cluster scales. In this range of masses, typical overdensities $\langle \delta_i^2 \rangle^{1/2}$ on the galactic scale are ~ 3 times larger than the overdensities $\langle \overline{\Delta}_i^2 \rangle^{1/2}$ on the cluster scale. Moreover, dense clusters are thought to form in regions with significant "3 σ" overdensities on the cluster scale. Therefore, taking $\overline{\Delta}_i = 3\Delta_i^2\rangle^{1/2} \sim \langle \delta_i^2 \rangle^{1/2}$ for the future cluster environment and comparing it with the future field values of large-scale overdensity, $\overline{\Delta}_i = 0$, one arrives at the conclusion that the objects of similar galactic mass will be ~ 8 times denser when formed in the environment of the future cluster than the entities of similar mass forming in the field:

$$\frac{\rho_C}{\rho_F} = \frac{(\delta_i + \overline{\Delta}_i)^3}{\delta_i^3} \sim 8 . \tag{14}$$

This is a very significant difference between the galaxies forming in the field and inside the future clusters. It would be difficult to imagine that a change of density of the dark matter halo by about one order of magnitude would have no effect on the observable properties of the luminous part of the galaxy. An obvious application of this effect would involve an "automatic" bias (Zurek, Quinn, and Salmon 1987) of the kind required to make the cold dark matter scenario compatible with the flat universe (Kaiser 1985; Bardeen *et al.* 1986). We shall return to the discussion of the possible consequences of this effect on the morphology of galaxies in the next section.

One more striking consequence of changes in the large-scale overdensity and/or Ω is shown in Figure 6. There we have plotted the "rotation curves":

$$v(r) = \sqrt{GM(r)/r} \tag{15}$$

of hypothetical test particles moving in the gravitational field of the halo with the cumulative mass profile $M(r)$. We have, of course, chosen halos which have preserved their identity in runs with different values of Ω.

A clear trend emerges from the examples shown in Figure 6: halos forming in denser regions of a flat universe (or in denser universes) have not only larger central densities (as was already established in Fig. 5) but also very flat, or even gently rising rotation curves. By contrast, halos forming in relative "voids" are not only smaller, but also have decreasing rotation curves. This behavior can be reproduced using the secondary infall model (Hoffman 1988*b*).

It is tempting to speculate about the possibility of observing systematic trend in the shape of the rotation curves with the overall density of the environment. One should, however, keep in mind that the typical shape of the rotation curve may also depend on other factors. For example, if one were to choose objects of similar luminosity in two environments of different density, the galaxy existing in the less dense environment would be a product of the initial perturbation with a larger value of δ_i. However, galaxies with larger initial δ_i values will tend to have systematically flatter rotation profiles (QSZ; Hoffman 1988*b*). This selection effect would tend to erase differences in the rotation profile conditioned by the environment. Moreover, only the central parts of the rotation curve of most of the galaxies are readily accessible. Rotation profiles in these regions are, however, only in part due to dark matter, as the baryonic component of the total mass plays a decisive role in establishing the overall mass distribution in the visible part of spiral galaxies. The physics which determines the behavior of these central regions involves both dissipative processes and star formation (Silk 1985*a*, *b*; Blumenthal *et al.* 1986). Therefore, it is difficult to estimate how much of the differences in the rotation profiles conditioned by the environment would be visible. Guhathakurta *et al.* (1987) have recently reported on a study of the H I rotation curves of spirals in the Virgo cluster. They find no systematic differences between the rotation curves of Virgo spirals and those in the field.

Recent simulations of Frenk *et al.* (1988) have also addressed the issue of the influence of density of the environment on the properties of the halos in a cold dark matter (CDM) universe. While Frenk *et al.* (1988) stress different aspects of the influence than us (e.g., they focus on the increase of the total halo mass with Ω rather than on the central density) their results are generally compatible with ours. Small differences—for instance, they seem to see much less trend in the shape of the rotation profiles with varying Ω—can be probably attributed to somewhat different initial conditions, although influence of differences in the method (particle-particle/particle-mesh vs. N-body) and, especially, the epoch in which initial conditions are set up ($z = 6$ in Frenk *et al.* and $z = 24$ in ours) cannot be entirely ruled out.

V. A MECHANISM FOR NATURAL SELECTION: ANGULAR MOMENTUM TRANSPORT IN DIFFERENTIATED MERGERS

Mergers of halos have already taken place in the process of formation of the objects shown in Figure 1 and are contributing to the graph of J/M in Figure 2. Most of the halos formed in such simulations go through several episodes of merging during the course of their formation (QSZ; Frenk *et al.* 1985; Barnes and Efstathiou 1987). Yet the value of J/M of all of these halos still clings to the upper "spiral" branch of the plot. Hence halo merging has not altered the J/M versus M relation enough to let some of the objects enter the "elliptical" region in Figure 2. In other words, the process of merging and virialization in a population of objects with cosmologically induced initial motions and spins produces a rather unique and well-defined specific angular momentum for a system of a given mass when gravity alone is responsible for the evolution. This specific angular momentum is inconsistent with that found in the luminous parts of ellipticals today.

To explain the values of $[J/M]_g$ and λ_g that are characteristic of ellipticals we are forced to search for a mechanism which could alter the specific angular momentum content of their baryonic component alone. In the hierarchical picture of halo formation—which is valid for the scale-free simulations discussed here as well as a good approximation to the CDM scenario on galactic scales—objects existing at present were built up by the merging and accretion of smaller, less massive progenitors. We shall suppose that in the denser regions of the universe these progenitors were dense enough to allow baryons to concentrate in their cores. This process takes place as an inevitable consequence of energy dissipation via radiative cooling.

The dissipation of energy in the gaseous component of an

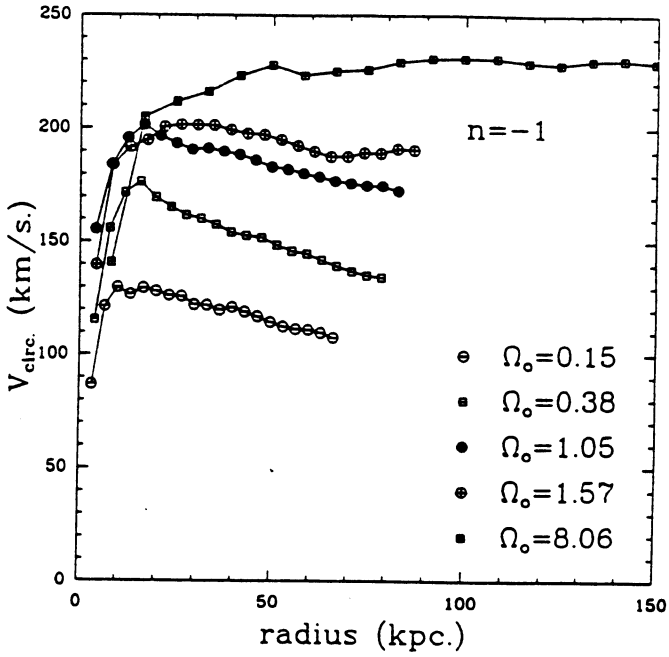


FIG. 6a

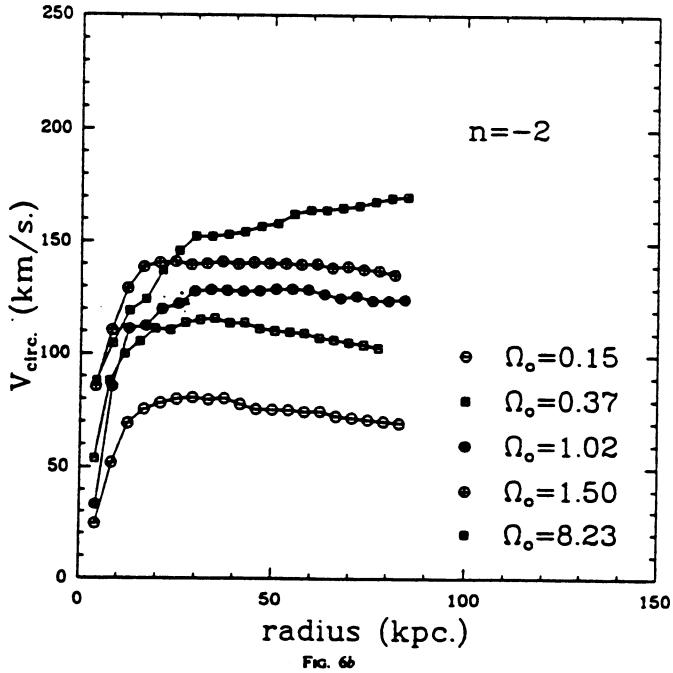


FIG. 6b

FIG. 6.—Rotation curves for some of the halos which have maintained their identity in runs with different values of Ω for (*a*) $n = -1$ and (*b*) $n = -2$. Halos formed when $\Omega > 1$ have flat or even rising rotation curves and are more massive and larger than halos found in open ($\Omega < 1$) universes.

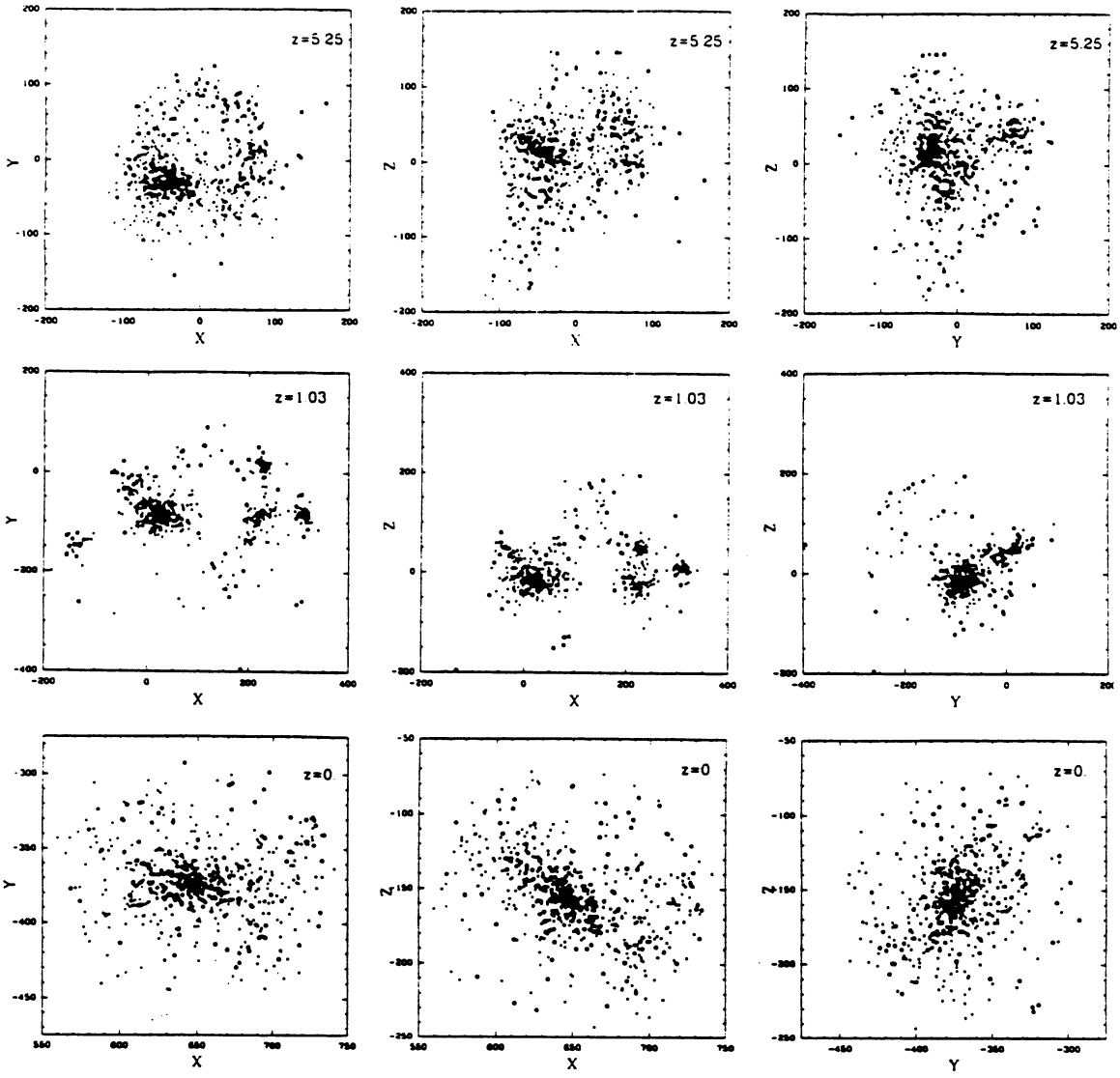


FIG. 7a

FIG. 7.—Evolution of a protohalo from $z = 5.25$ to $z = 0$ for $n = -1$. (*a*) At $z = 0$ all the bound particles within 100 kpc of the core were tagged and divided into three bins of binding energy. The most bound particles are marked as solid dots, the least bound as open circles, and intermediate energies as crosses. The most bound particles at $z = 0$ occupied the cores of subclumps seen at $z = 1.03$ which merged to form the final halo. At $z = 5.25$, the most bound particles also generally live in regions of excess density. Note that each frame at a given z has the same linear scale in both axes (measured in kpc, noncomoving). (*b*) The same as (*a*), except that now the tagging took place at $z = 1.03$. This figure proves that the core material of the progenitors ends up almost exclusively—solely as a result of the dynamical dissipation—in the core of the resultant halo.

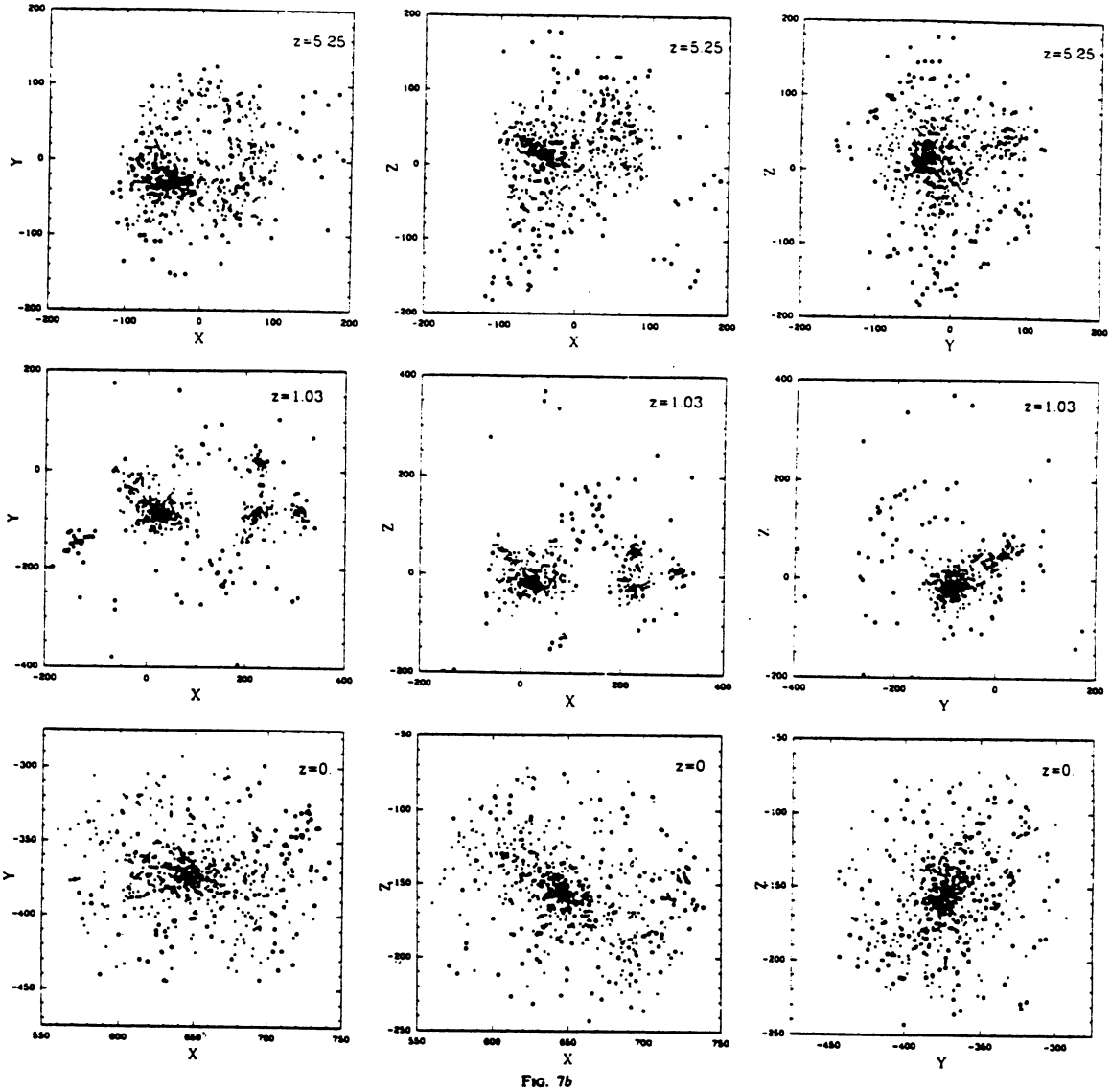


FIG. 7b

object which has recently grown through a merger will typically leave its J_g unaltered: transport of energy occurs through radiation, which does not carry significant amounts of angular momentum. Stellar systems formed from the gas heated in the course of a merger would therefore have $[J/M]_g$, and, presumably, morphologies, similar to those of present-day spiral galaxies. Indeed, it is natural to suppose that the present-day spirals were formed in such a fashion (Fall and Efstathiou 1980; Gunn 1982; Ryden and Gunn 1987).

The increase of density resulting from dissipation is likely to accelerate star formation. Hence, the bulk of the baryons in sufficiently dense progenitors of present-day ellipticals could be converted into stars before the "final"—that is, most recent—merging episodes. These condensed-out, stellar cores in the progenitors of ellipticals make angular momentum transport possible (in fact, sufficiently clumpy gas clouds may work as well if they can avoid reheating). Let us consider a merger of several such progenitors into a larger galaxy. Figure 7 shows the evolution of a halo from our $n = -1$ simulation. At fixed redshifts ($z \sim 0$ and $z \sim 1$), the halo particles were tagged according to their binding energy and traced back to $z = 5.25$. The most tightly bound particles (*solid dots*) show

that the cores of the progenitors sink into the core of the final merged object. In this process, components of the luminous part of the elliptical dispose of some of their angular momentum through "tidal braking." Figure 8 shows the evolution of the halo shown in Figure 7 in the energy—angular momentum (Lindblad) plane. The halo was divided in binding energy bins at $z = 0$, and the particles from these bins were remeasured at $z = 5.25$. The figure shows that angular momentum is transferred from the orbital motion of the tighly bound cores to the loosely bound outer halo particles as the cores sink into the center of the forming halo (see also Quinn and Zurek 1988). This loss of angular momentum by the cores due to dynamical friction will lead to a low specific angular momentum core in the final galaxy. Moreover, the spin of the core could be further reduced if the spins of the progenitors were randomly oriented and therefore would tend to cancel (Fall 1979). Figure 8 also shows that the constituents of the cores not only lose specific angular momentum but, also, become more tightly bound. In our simulations, the cores are composed of particles identical to the rest of the halo. Therefore, when the only interaction between the core components is gravitational—as is the case when the cores are stellar—then the results obtained with our dissipationless simulations apply at least approximately: the equations of motion followed by the stars will be obviously the same as those of cold dark matter particles. However, their initial, premerger orbits inside the progenitors would be presumably more "organized" by rotation—i.e., with λ typical of spirals. Consequently, stars would start with a different set of initial conditions. Nevertheless, given the observed efficiency of angular momentum redistribution in course of a typical merger (Quinn and Zurek 1988), we anticipate that the final state of the baryonic component should be largely determined by their initial binding energy (or alternatively their final location). Consequently, our numerical results show that a merger of protogalaxies with condensed, stellar cores will lead to an object with a relatively low specific angular momentum in the luminous stellar component.

The angular momentum lost by the core is taken up by the rest of the extended halo. In terms of the diagram J/M versus M, Figure 2, this process can be represented—for a single halo—by two arrows. The first one, describing the baryonic core component, would point downward (as the "most bound" particles of Fig. 8 and baryon mass conservation jointly imply) and would push the stellar core component down in the direction of the "elliptical strip" in Figure 2. The other arrow, characterizing the nonbaryonic dark halo, would point upward, almost parallel to the $J \sim M^{7/4}$ line. The vector sum of the two arrows should push the complete merged system (i.e., dark halo plus baryons) up along the "spiral strip" in Figure 2.

In the model of differentiated mergers the dynamics responsible for the formation of ellipticals is purely dissipationless. If the mixing accompanying a merger is efficient, the typical angular momentum of the luminous part of the elliptical can be estimated using its relative size (i.e., as compared with the total size of its dark halo) and a typical total angular

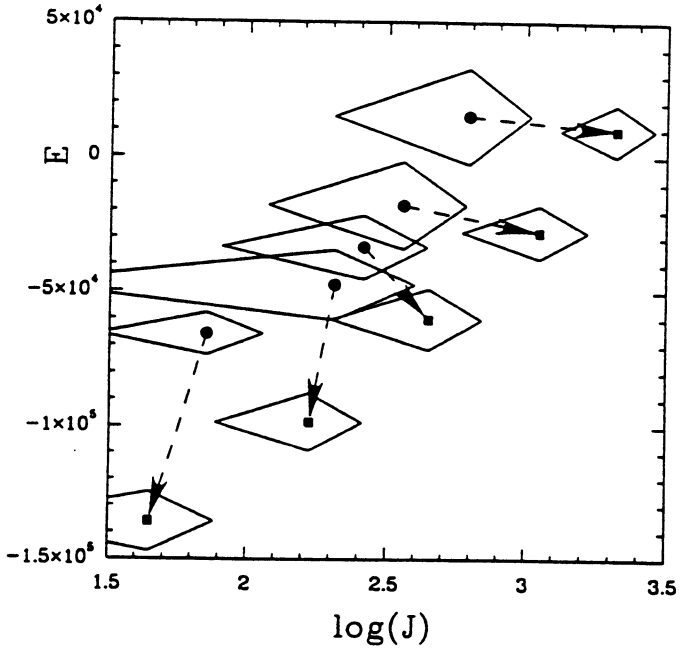


FIG. 8.—Evolution of the protohalo shown in Fig. 7 in the Lindblad (energy-angular momentum) plane. At $z = 0$ the halo was divided up into five binding energy bins. The angular momentum and binding energy of the particles belonging to these bins were then measured at $z = 5.25$ (*filled circles*) and compared to their $z = 0$ (*filled squares*) values. The error-diamond on each point represents the 1 σ dispersion of E and J for each of the binding energy groups at $z = 5.25$ and $z = 0$. In summary: particles that are initially tightly bound become considerably more bound and generally lose angular momentum, while particles that are initially loosely bound gain significant amounts of angular momentum and become slightly more bound.

momentum of a halo. We should be therefore able to test our model by using the results of our simulations to derive the size of the gap between the spiral and elliptical strips in Figure 2. To accomplish this we first assume that the luminous component of a spiral galaxy has approximately preserved its original angular momentum. Hence, its J/M is typical for a halo and defines the spiral strip of Figure 2. On the other hand, the baryons in spirals must have collapsed by a factor of ~ 10 in radius (Efstathiou and Silk 1983). Baryons in ellipticals are presumably "compressed" by a similar ratio in comparison with the sizes of elliptical halos. However, in the course of a merger, stars in ellipticals must have lost enough angular momentum to end up with a value of J/M smaller than the halo as a whole by a factor of ~ 10. Hence, the specific angular momentum of an elliptical should be approximately one order of magnitude smaller than in a spiral of a comparable mass. This factor is clearly consistent with, but probably somewhat larger than the estimate extracted from the somewhat uncertain observational data (see Fig. 1 in Fall 1983). The model of differentiated mergers has therefore passed an important consistency check. It is also reassuring that our estimate of the gap size errs on the high side. The angular momentum of baryons settling in the core of a spiral is probably only approximately conserved, and mixing of stars of the progenitors may not be quite as complete as mixing of dark matter particles. Both of these effects will tend to shrink the gap.

By the same token, relaxation occurring on the scale of the stellar cores should also be responsible for the observed value of the spin parameter λ_g in early-type galaxies. This can be argued as a corollary to the formation model of ellipticals—mergers of differentiated progenitors with stellar, dissipationlessly evolving cores—we have described above: The distribution of the spin parameter is determined in course of the merging process by relaxation and mixing alone and is fairly constant inside a virialized object (Quinn and Zurek 1988). Since the dynamics of merging of differentiated progenitors is governed solely by gravitational interactions, which do not distinguish between stars behaving as gravitationally interacting dark matter, and bona fide dark matter, both stellar and dark components should acquire the same λ. Hence for ellipticals $\lambda_g \sim \lambda$. A typical range of λ, as shown for instance in Figure 3, is similar to the typical values measured in ellipticals.

Each of the two preceding paragraphs describes a test of our model. While they are not completely independent (both of them involve rotation), passing of the first test does not automatically guarantee passing of the second one. Therefore, we regard their outcomes as quite encouraging.

The behavior of baryons which have not condensed into stars or other compact objects and interact only via gravitational attraction is still a major unknown. In the last few paragraphs of this section we shall venture a relatively safe "educated guess." To begin with, in the course of a typical merger gas will reheat and, at least initially, fill in most of the halo. As the distributions of the gas and dark matter are quite similar, their angular momentum content can also be expected to be almost the same. Subsequently the dark halo virializes, and the subclumps in the distribution of dark matter disappear. Gas filling the halo is also relatively uniform. Therefore, it cannot exchange angular momentum with the halo as it begins to cool and collapse to the rotationally supported disk configuration.

This process has been discussed by Gunn (1982), Blumenthal *et al.* (1986), and Ryden and Gunn (1987). Our analysis adds to these treatments the following conjecture: The bulges of spiral galaxies form from the stars which were already born in the progenitors of the present-day spirals, before they underwent the final merger episodes. Hence, they were able to give up angular momentum in the course of these mergers and have settled down into the present-day bulge. By contrast, gas contained in these progenitors settled more slowly into a disk and managed to preserve its original content of angular momentum. Stars formed in the disk should therefore be on the whole younger than those inhabiting the bulge. This is consistent with the studies of stellar populations in spiral galaxies (see, e.g., Freeman 1988, and references therein).

VI. DISCUSSION

The key conclusions from the computer simulations presented in the previous sections can be summed up in four simple statements:

1. If galaxies form through gravitational collapse from initial, Gaussian density perturbations, then their luminous parts do not have the "original" values of either specific angular momentum (ellipticals) or spin parameter λ (spirals).
2. The rotational properties of a halo depend only weakly on either the halo density or the density of its environment.
3. Variations of the large-scale density of the environment have an expected, dramatic effect on the density of individual halos.
4. Dissipationless merging forces the cores of the merging objects into the core of the merger product. In the process cores become more bound and lose angular momentum.

Extrapolation of these results has led us to propose a model for galaxy formation in which both ellipticals and spirals come into being through mergers. The purpose of this section is to review the assumptions which are important for the discussion of the galaxy formation given in § V, but which have not been justified by the computer models. We shall emphasize the role of density—dependent star formation rates in accounting for the correlations between the density of the halo, the environment, and the morphology of the galaxy. We shall also point out similarities and differences between the model of differentiated mergers and some of its forerunners. We shall end with a review of the observational consequences of the proposed galaxy formation model.

Any attempt to explain the morphology of the luminous part of the galaxy must face up to a very serious challenge: the relevant astrophysics involves both dissipative hydrodynamics of a self-gravitating system and star formation. At present, we can claim only a limited understanding of the first, and an almost complete ignorance of the second. We shall therefore adopt the following "working hypothesis": the efficiency of star formation is a steep function of the density of baryonic material (Silk 1985*b*; Shu 1983; Tohline 1985; Terebey, Shu, and Cassen 1985).

With the above assumption in mind, consider now two peaks of the density field, two future "protogalaxies," one of them with the average overdensity, δ_1 and the other with some larger overdensity δ_2. The second peak may be more overdense either because it is inside a future cluster, or because it is just an exceptionally high peak "in the field." In either case, as the universe evolves, the smallest structures, subpeaks of density within each of the two protohalos, will begin to collapse and form individual, if short-lived, objects. To account for the morphological dependence on the environment we will use our assumption about the star formation rates. We assert that only baryons inside the denser peak have a sufficient density to

precipitate efficient star formation. Moreover, stars have formed predominantly near the cores of the fragments. During the next stage of the evolution—merging of protogalactic fragments to form a present-day galaxy—the densest part of the fragments sink into the center of the halo of the new galaxy. In the core of the forming elliptical, torquing of the dark halo material by the quadrupole moments generated by the merging baryonic cores is inevitable and must lead to substantial losses of specific angular momentum by the baryonic component. Such interhalo transport of angular momentum is observed in numerical simulations (Efstathiou and Jones 1979; Frenk *et al.* 1985; Barnes and Efstathiou 1987; Quinn and Zurek 1988) and is necessary to account for the distribution of angular momentum inside individual halos (Quinn and Zurek 1988). Moreover, we may be witnessing it today in ellipticals with multiple cores (Lauer 1987).

Merging would also take place in the course of the evolution of the future spiral. However, now the bulk of the baryonic material remains gaseous and is distributed throughout the fragments of the protohalo. Therefore, during the final merger only the very central parts of the cores of the merging objects—we assert citing the assumed, strong dependence of the star formation rate on density—have managed to undergo some star formation. These central, stellar components will again lose their angular momentum due to tidal braking, and spiral in to form the bulge of the newly formed spiral galaxy. The gas in the outer parts of the halo is, on the other hand, distributed more evenly and will maintain its specific angular momentum J/M. Its spin parameter after reheating will probably be close to the value of λ of the dark matter halo. However, as the gas loses energy during the subsequent evolution, the value of its spin parameter will increase until it is rotationally supported (Fall and Efstathiou 1980). This stage of the evolution is approximately axisymmetric and involves no losses of angular momentum, but large, dissipative losses of energy (Efstathiou and Silk 1983).

Both initial conditions and dynamical evolution play an important role in this model of galaxy formation. Fluctuations of the density field determine the star formation rates. The efficiency with which gaseous baryons are converted into a dissipationless stellar population determines how much interhalo angular momentum transport occurs in course of mergers. Given sufficient understanding of the star formation process one could use this model to discriminate between different spectra and normalizations of the cosmological density fluctuations. Some sort of a "bias" (Kaiser 1984; Bardeen *et al.* 1986) could easily emerge from such considerations and would be naturally and intimately tied to the morphology-environment relation, as it was speculated by Zurek, Quinn, and Salmon (1987). At present, however, any scenario in which tidal torquing supplies enough angular momentum and where merging of smaller objects leads to present-day galaxies could be presumably accommodated with a judicious choice of the star formation rate. In particular, in the version of the cosmic string scenario in which galaxies form by mergers (Zurek 1986) rather than by collapse around individual loops (Turok and Brandenberger 1986) a similar picture could apply.

The role of the mergers in determining galactic morphologies was recognized by Toomre (1977) and explored in a context of the specific "old merger model" by Fall (1979), Efstathiou and Jones (1979), and Aareseth and Fall (1980). There, the formation of ellipticals was a two-stage process. First, came the dissipation of energy which was required to account for the small sizes of galaxies relative to their dark matter halos. Mergers constituted the second stage. The low angular momentum of merger products was attributed to: (1) the selection process—merging required orbits of low impact parameter and (2) the statistical cancellation of the spins in the remnant. While in the model we have suggested the bulk of the angular momentum is transported away from the baryons by tidal braking, selection and cancellation may play a role in determining the outcome of a merger. However, our results appear to suggest that these two effects—which must occur naturally in course of our simulations—alone do not suffice in bringing down the J/M values in typical halos into the elliptical region of Figure 2. The key distinction between the "old merger model" and the model of differentiated merging is therefore the recognition, forced on us by the results of the simulations, that the dark halo can take up excess angular momentum of the stellar cores of merging protogalaxies. The key similarity is, of course, the suggestion that ellipticals form in mergers.

Many of the issues raised by this discussion can be settled only with the dramatic improvements in our understanding of the star formation process. However, some of the predictions may be testable in the less distant future. For example, we may be able to test the dependence of Hubble type on halo density by measuring the density of the dark halos directly. Fall (1986) has concluded that there is now considerable evidence of dark matter in early-type galaxies. The hot X-ray coronae of ellipticals (Forman, Jones, and Tucker 1985), polar ring galaxies (Whitmore, McElory, and Schweizer 1987), and orderly shell systems (Hernquist and Quinn 1987) all indicate the presence of considerable amounts of dark matter. Hernquist and Quinn (1987) have suggested a way to quantify the distribution of the dark matter in galaxies of different morphological types. An isothermal halo [$M(r) \sim r$)] can be conveniently parameterized by

$$\Xi = \frac{M(r)}{r} = \frac{V_A^2}{G}, \tag{16}$$

where V_A^2 is the asymptotic circular velocity. The slope of the mass profile can be estimated from the X-ray data and shell distribution given some assumptions (such as isothermality for the X-ray gas). For spirals and polar ring S0s, V_A can be measured directly if the rotation curve is flat at the outermost points. Figure 9 of Hernquist and Quinn plots Ξ against luminous mass for the available X-ray and shell data as well as the Bahcall and Casertano (1985) sample of spirals. The figure indicates that the elliptical Ξ values may be higher than spirals of the same luminous mass (factor ~ 5). This result is tentative since there is very little overlap in luminous mass between the X-ray elliptical sample and the spirals. A better data set including a large range in luminosity for spirals and more low-luminosity early-type galaxies (such as S0s) is currently being compiled (Fall and Quinn 1987, in preparation). It should be noted that, although halos in clusters were initially more dense than their field counterparts, angular momentum transport from the collapsing baryonic material may have reduced the central halo density.

We would like to thank Mike Fall and Yehuda Hoffman for valuable conversations and comments on the manuscript. We are grateful to Los Alamos National Laboratory and the Space Telescope Science Institute for providing the large amounts of CRAY and VAX-like computer time necessary to run and analyze the N-body models presented in this paper.

REFERENCES

Aarseth, S. J., and Fall, S. M. 1980, *Ap. J.*, **236**, 43.
Bahcall, J. N., and Casertano, S. 1985, *Ap. J. (Letters)*, **293**, L7.
Bardeen, J. M., Bond, J. R., Kaiser, N., and Szalay, A. S. 1986, *Ap. J.*, **304**, 15.
Barnes, J., and Efstathiou, G. 1987, *Ap. J.*, **319**, 575.
Bertschinger, E. 1985, *Ap. J. Suppl.*, **58**, 39.
Blumenthal, G. R., Faber, S. M., Flores, R., and Primack, J. R. 1986, *Ap. J.*, **301**, 27.
Blumenthal, G. R., Faber, S. M., Primack, J. R., and Rees, M. J. 1984, *Nature*, **311**, 517.
Bodenheimer, P. 1981, in *IAU Symposium 93, Fundamental Problems in the Theory of Stellar Evolution*, ed. D. Sugimoto, D. Q. Lamb, and D. N. Schramm (Dordrecht: Reidel), p. 5.
Davies, R. L., Efstathiou, G., Fall, S. M., Illingworth, G., and Schechter, P. L. 1983, *Ap. J.*, **266**, 41.
Doroshkievich, A. G. 1970, *Astrofizika*, **6**, 320.
Dressler, A. 1981, *Ap. J.*, **236**, 351.
Efstathiou, G., and Jones, B. J. T. 1979, *M.N.R.A.S.*, **186**, 133.
Efstathiou, G. and Silk, J. 1983, *Fund. Cosmic Phys.*, **9**, 1.
Faber, S. M. 1982, in *Astrophysical Cosmology: Proceedings of the Vatican Study Week on Cosmology and Fundamental Physics*, ed. H. A. Bruck, G. V. Coyne, and M. S. Longair (Rome: Specola, Vaticana), p. 191.
Fall, S. M. 1979, *Nature*, **281**, 200.
———. 1980, in *The Structure and Evolution of Normal Galaxies*, ed. S. M. Fall and D. Lynden-Bell (Cambridge: Cambridge University Press), p. 1.
———. 1983, in *IAU Symposium 100, Internal Kinematics and Dynamics of Galaxies*, ed. E. Athanassoula (Dordrecht: Reidel), p. 391.
———. 1986, in *Santa Cruz Workshop on Nearly Normal Galaxies*, ed. S. M. Faber (Berlin: Springer), p. 326.
Fall, S. M., and Efstathiou, G. 1980, *M.N.R.A.S.*, **193**, 189.
Fillmore, J. A., and Goldreich, P. 1984, *Ap. J.*, **281**, 1.
Forman, W., Jones, C., and Tucker, W. 1985, *Ap. J.*, **293**, 102.
Freeman, K. C. 1975, *Stars and Stellar Systems*, Vol. 9, *Galaxies and the Universe*, ed. A. Sandage, M. Sandage, and J. Kristian (Chicago: University of Chicago Press), p. 409.
———. 1988, in *IAU Symposium 130, Evolution of Large-Scale Structures in the Universe* (Dordrecht: Reidel), in press.
Frenk, C. S., White, S. D. M., Efstathiou, G., and Davis, M. 1985, *Nature*, **317**, 595.
———. 1988, *Ap. J.*, **327**, 507.
Giovanelli, R., Haynes, M. P., and Chincarini, G. L. 1986, *Ap. J.*, **300**, 77.
Gott, J. R., and Thuan, T. X. 1976, *Ap. J.*, **204**, 649.
Guhathakurta, P., van Gorkom, J. H., Kotanyi, C. G., and Balkowski, C. 1987, preprint.
Gunn, J. E. 1982, in *Astrophysical Cosmology: Proceedings of the Vatican Study Week on Cosmology and Fundamental Physics*, ed. H. A. Bruck, G. V. Coyne, and M. S. Longair (Rome: Specola Vaticana), p. 233.
Gunn, J. E., and Gott, J. R. 1972, *Ap. J.*, **176**, 1.
Hernquist, L., and Quinn, P. J. 1987, *Ap. J.*, **312**, 1.
Hoffman, Y. 1986, *Ap. J.*, **301**, 65.
———. 1988*a*, *Ap. J.*, **328**, 489.
———. 1988*b*, *Ap. J.*, **329**, 8.
Kaiser, N., 1984, *Ap. J. (Letters)*, **289**, L9.
Kashlinsky, A. 1982, *M.N.R.A.S.*, **200**, 585.
Lauer, T. R. 1987, Princeton Univ. preprint.
Oemler, A. 1974, *Ap. J.*, **194**, 1.
Peebles, P. J. E. 1969, *Ap. J.*, **155**, 393.
———. 1980, *The Large Scale Structure of the Universe* (Princeton: Princeton University Press).
Postman, M., and Geller, M. J. 1984, *Ap. J.*, **281**, 95.
Quinn, P. J., Salmon, J. K., and Zurek, W. H. 1986, *Nature*, **322**, 329.
Quinn, P. J., and Zurek, W. H. 1988, *Ap. J.*, in press.
Ryden, B. S., and Gunn, J. E. 1986, *Ap. J.*, **318**, 15.
Sandage, A., Freeman, K., and Stokes, N. R. 1976, *Ap. J.*, **304**, 831.
Shu, F. H. 1983, *Ap. J.*, **214**, 488.
Silk, J. 1985*a*, *Ap. J.*, **297**, 1.
———. 1985*b*, *Ap. J.*, **297**, 9.
Terebey, S., Shu, F. H., and Cassen, P. 1985, *Ap. J.*, **292**, 529.
Thuan, T. X., and Gott, R. J. 1977, *Ap. J.*, **216**, 194.
Tohline, J. E. 1985, *Ap. J.*, **292**, 181.
Toomre, A. 1977, *The Evolution of Galaxies and Stellar Populations*, ed. B. M. Tinsley and R. B. Larson (New Haven: Yale University Press), p. 401.
Turok, N., and Brandenberger, R. H. 1986, *Phys. Rev. D*, **33**, 2182.
White, S. D. M. 1978, *M.N.R.A.S.*, **184**, 185.
———. 1983, in *IAU Symposium 100, Internal Kinematics and Dynamics of Galaxies*, ed. E. Athanassoula (Dordrecht: Reidel), p. 337.
Whitmore, B. C., McElroy, D. B., and Schweizer, F. 1987, *Ap. J.*, **314**, 439.
Zel'dovich, Ya. B., and Novikov, I. D. 1983, *The Structure and Evolution of the Universe* (Chicago: University of Chicago Press).
Zurek, W. H., Quinn, P. J., and Salmon, J. K. 1987, in *IAU Symposium 117, Dark Matter in the Universe*, ed. J. Kormendy and G. R. Knapp (Dordrecht: Reidel), p. 316.
Zurek, W. H. 1986, *Phys. Rev. Letters*, **57**, 2326.

P. J. QUINN: Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218

J. K. SALMON: Theoretical Astrophysics, 130-33 Caltech, Pasadena, CA 91125

W. H. ZUREK: Theoretical Astrophysics, T6, MS B-288, Los Alamos National Laboratory, Los Alamos, NM 87545

E. Correlation of QSO Absorbtion Lines...

Reprinted by permission from the Monthly Notices of the Royal Astronomical Society, **221**, 93–104, 1986. John Salmon, Craig Hogan, "Correlation of QSO Absorption Lines in Universes Dominated by Cold Dark Matter."

Mon. Not. R. astr. Soc. (1986) **221**, 93–104

Correlation of QSO absorption lines in universes dominated by cold dark matter

John Salmon *California Institute of Technology, Pasadena, CA 91125, USA*
Craig Hogan *Steward Observatory, University of Arizona, Tucson, AZ 85721, USA*

Accepted 1986 January 31. Received 1986 January 31; in original form 1985 September 27

Summary. Theoretical predictions for the redshift correlations between QSO absorption-line systems are investigated in the context of 'cold dark matter' cosmological models. Particles in 'particle-mesh' N-body simulations are interpreted as absorbing clouds at epochs corresponding to mean redshifts, z, of 0.0, 1.25 and 3.0. The velocity correlation function for absorbing clouds is found by passing lines-of-sight through the systems and computing velocity differences for those particles which lie close to the lines. It depends strongly on z and Ω but only weakly, if at all, on the number density, diameter or mass of the clouds. The results are compared with observations of Sargent *et al.* Two interpretations are possible (i) the heavy element absorption systems are associated with galaxies which are an unbiased sample of the mass distribution in an $\Omega_0=0.2$ universe or (ii) the Lyman-α absorbers are an unbiased sample of the mass in an $\Omega_0=1$ universe and the heavy-element absorption systems, like galaxies, are more strongly clustered than the mass. The very rapid evolution in the correlations is understood by analogy with a power-law fluctuation spectrum, for which analytical results are obtained using self-similarity.

1 Introduction

Model universes dominated by 'cold dark matter' (Peebles 1984; Blumenthal *et al.* 1984; Davis *et al.* 1985) have become attractive candidates as explanations for large-scale cosmological structure, both because of their general agreement with data and because they are in principle determined by only one parameter, the amplitude of the initial perturbations (we assume that Ω and H_0 will not always be regarded as parameters). In practice, interpretation of these models, and their comparison with traditional statistical measures of the clustering, such as the galaxy autocorrelation function and the distribution of peculiar velocities, is plagued by uncertainties, such as whether or not the galaxy populations used to measure these statistics represent an unbiased sample of the mass distribution predicted by the model. Nevertheless, cold dark matter models are sufficiently precisely defined, and successful enough in agreement with data, that it is worthwhile to investigate other observational predictions.

In this paper, we compute the correlation in redshift space predicted by cold dark matter models for the absorption-line systems observed in distant QSO's. Because the correlation is strongly affected by non-linear peculiar velocities, a numerical simulation is necessary. This preliminary study is designed to investigate the effectiveness of QSO absorption lines as an independent observational test of cosmological models in general. A similar study was made by Dekel (1982), but in the context of 'pancake' scenarios (Zel'dovich 1970). Line studies may remove some of the ambiguity in interpretation alluded to above, since the selection criteria for absorption clouds are very different from those for galaxies in catalogues used to measure traditional statistics. Absorption lines provide in addition a measurement of structure at an earlier epoch, and so give us a direct test of the evolution predicted by the models. For rapidly evolving models, this can provide a powerful constraint. It is clear from the results obtained here that QSO absorption-line correlations are sensitive probes of the evolution of large-scale structure which are in many ways complementary to other types of data. It is possible, however, that the observed velocity correlations are primarily the result of non-gravitationally induced velocities. For example, correlated absorbing systems might arise in high-velocity outflows associated with star formation within single galaxies. Our calculations are intended primarily as benchmarks to provide some idea of the velocity correlation one expects from gravitational clustering alone.

For comparison with our models, we will use the following results of Sargent *et al.* (1980), based on high-resolution spectra of six QSO's:

Nine pairs of heavy-element absorption systems (C IV doublets) in a sample with $1.6 \leq z_{abs} \leq 2.3$ were found with separation between 100–200 km s^{-1}, which is 25 times the expected number for this sample if the systems were uncorrelated in velocity space. We take this to imply that there is probably a real correlation with amplitude between 15 and 35.

(ii) In a sample of Ly-α absorption lines with $1.7 \leq z_{abs} \leq 3.3$ and a mean of 2.44, with an average of 6.5 pairs per 100 km s^{-1} bin, the distribution is consistent with a random Poisson process. We take this to mean that the Ly-α absorbers probably have correlation amplitude less than 2 on all scales larger than 100 km s^{-1}.

These results already indicate a real difference between the cloud populations. It is anticipated that more data will be available in the near future which will allow a more detailed measurement of the correlations.

The aim of the present work is to calculate what correlations are expected under the assumption that the absorbing material is distributed like cold collisionless dark matter. Our cosmological N-body simulations provide an approximation to a continuous density and velocity field which would be produced in a particular volume in a cold dark matter model. We model QSO absorption lines as though they are caused by clouds of gas which constitute an unbiased discrete sampling of this field. We make no other assumptions about the nature of the clouds. [See, for example, Black 1981; Oort 1981; and Fransson & Epstein 1982 for discussions of Ly-α systems, and Young, Sargent & Boksenberg (1982) for discussions of C IV systems.] Both types of systems clearly cannot represent an unbiased sample of the mass so one of the issues is to try to elucidate the constraints imposed by the cold dark matter picture. In particular, we are able to determine whether our picture (clouds following mass) is consistent with the observations for either type of cloud.

Several difficulties with this approach must be addressed. First, there are *modelling uncertainties*: these include the density of sampling, corresponding to the number density of the clouds; the radius assumed for the clouds; small differences in the exact redshift sampling; and variations due to different realizations of the same power spectrum. Here, empirical estimates of the uncertainty introduced into the correlation estimates can be made by varying each of these

separately. Surprisingly, and fortunately, only the latter causes a significant change in the signal. This variation ought to be reflected in the realizations along lines-of-sight towards different QSO's which, although they pass through many volumes the size of our simulation, typically contain few correlated pairs, and therefore also sample limited regions of space. At present, the total number of QSO's observed and the number of realizations simulated are both small, which puts restrictions on the accuracy of the comparison beween the two. There are also many observational difficulties (line blending, etc.), which will not be discussed here.

Secondly, as with any numerical calculation there are *systematic errors*; particle orbits are not treated accurately below a length scale dictated by the mesh used in the 'particle-mesh' algorithm so there is a minimum believable length scale; and initial conditions are not accurately realized below the mass scale corresponding to a few particles. If the true velocity dispersion is dominated by high wavenumber fluctuations, our results will be erroneous. These problems are of course familiar and their magnitude can be estimated using standard techniques (Hockney & Eastwood 1981; Efstathiou *et al.* 1985). The cold dark matter spectrum decreases rapidly enough at high wavenumber, that we do not expect large errors of this kind. This is confirmed by the lack of any discernible trend with either particle number or length scale (see Tables 1–4 and Section 4).

There is also a certain amount of *unbiased sampling error*, which would arise even in an unbiased infinite model from estimating the correlation function with a finite number of pairs. The total sum of modelling uncertainty, systematic error and unbiased sampling noise appears to be less than the intrinsic correlation signal in most models, and also less than the signal found in the data for the heavy-element systems, so we find that the technique provides useful information.

2 Initial conditions

The spectrum of fluctuations in a universe dominated by cold dark matter has been analysed extensively in the linear regime (Peebles 1982b; Blumenthal & Primack 1983; Bond & Szalay

Table 1. Simulation parameters for $\Omega_0=1.0$ models. Dependencies on the Hubble constant are expressed in terms of $h_{22}=H_0/(22\,\mathrm{km\,s^{-1}\,Mpc})$, which must be approximately 1 in order to properly normalize the particle–particle correlation function (Davis *et al.* 1985). The velocity resolution is the Hubble velocity across two cells of the particle-mesh force calculator. It is quoted at $z=3$.

Run	$L\,(h_{22}^{-1}\,\mathrm{Mpc})$	$M_{\mathrm{total}}\,(h_{22}^{-1}\,M_\odot)$	N_{clouds}	$M_{\mathrm{cloud}}\,(h_{22}^{-1}\,M_\odot)$	$v_{\mathrm{res}}\,(h_{22}^{-1}\,\mathrm{km\,s^{-1}})$
EdS1	248	2.0×10^{17}	21952	9.3×10^{12}	128
EdS2	248	2.0×10^{17}	21952	9.3×10^{12}	128
EdS3	248	2.0×10^{17}	21952	9.3×10^{12}	128
EdS4	186	8.6×10^{16}	262144	3.3×10^{11}	96
EdS5	58	2.6×10^{15}	262144	1.0×10^{10}	30

Table 2. Simulation parameters for $\Omega_0=0.2$ models. Dependencies on the Hubble constant are expressed in terms of $h_{100}=H_0/(100\,\mathrm{km\,s^{-1}\,Mpc})$. The velocity resolution is the Hubble velocity across two cells of the particle-mesh force calculator. It is quoted at $z=3$.

Run	$L\,(h_{100}^{-1}\,\mathrm{Mpc})$	$M_{\mathrm{total}}\,(h_{100}^{-1}\,M_\odot)$	N_{clouds}	$M_{\mathrm{cloud}}\,(h_{100}^{-1}\,M_\odot)$	$v_{\mathrm{res}}\,(h_{100}^{-1}\,\mathrm{km\,s^{-1}})$
01	45	1.1×10^{14}	32768	3.4×10^{9}	90
02	45	1.1×10^{14}	21952	5.1×10^{9}	90
03	30	5.0×10^{13}	21952	2.3×10^{9}	60
04	33.75	6.3×10^{13}	262144	2.4×10^{8}	67

Table 3. Representative values of $n_{pairs}/\langle n_{pairs}\rangle$, an estimate of the velocity correlation function, W in $\Omega_0=1.0$ models in the velocity range $100\,h_{22}^{-1}\,\mathrm{km\,s^{-1}} \le \Delta v \le 200\,h_{22}^{-1}\,\mathrm{km\,s^{-1}}$. No entry signifies that we were not able to obtain reliable statistics for that model, cloud diameter and redshift.

Run	z	beam diameter		
		$200\,(h_{22}^{-2}\,\mathrm{kpc})$	$100\,(h_{22}^{-2}\,\mathrm{kpc})$	$50\,(h_{22}^{-2}\,\mathrm{kpc})$
EdS1	3.0	2.0, 1.1	2.4, 1.1	—
	1.25	7.8, 5.5	5.7, 8.2	—
	0.0	13.4, 20.7	—	—
EdS2	3.0	2.1, 1.3	3.3, 1.5	—
	1.25	7.3, 5.8	4.5, 8.2	—
	0.0	13.6, 17.2	—	—
EdS3	3.0	1.6, 1.6	1.3, 2.4	—
	1.25	5.9, 7.4	4.5, 7.0	—
	0.0	16.4, 15.6	—	—
EdS4	3.0	3.2, 2.5	2.3, 2.6	1.8, 2.6
	1.25	7.4, 7.1	6.5, 6.8	6.7, 7.3
	0.0	18.0, 15.8	18.0, 13.9	17.0, 16.8
EdS5	3.0, 3.3	3.3	3.5, 3.0	3.4, 2.6
	1.25	12.0, 13.7	5.0, 5.7	8.4, 7.0
	0.0	—	—	—

Table 4. Same as Table 3, for $\Omega_0=0.2$ models in the velocity range $150\,h_{100}^{-1}\,\mathrm{km\,s^{-1}} \le \Delta v \le 200\,h_{100}^{-1}\,\mathrm{km\,s^{-1}}$.

Run	z	beam diameter		
		$200\,(h_{100}^{-2}\,\mathrm{kpc})$	$100\,(h_{100}^{-2}\,\mathrm{kpc})$	$50\,(h_{100}^{-2}\,\mathrm{kpc})$
01	3.0	—	—	—
	1.25	13.7, 14.3	11.1, 14.7	18.1, 13.9
	0.0	—	—	—
02	3.0	—	6.4, 5.7	3.4, 4.3
	1.25	12.7, 8.6	11.6, 13.7	13.9, 9.3
	0.0	35.7, 27.8	28.5, 31.7	31.1, 26.3
03	3.0	—	15.9, 3.8	7.0, 8.0
	1.25	8.4, 7.0	10.3, 14.0	8.2, 16.4
	0.0	27.5, 31.2	36.2, 25.5	36.5
04	3.0	—	—	—
	1.25	15.3, 7.9	12.1, 13.8	13.3, 11.3
	0.0	21.7, 22.5	34.3, 36.0	31.3, 36.9

1983; Bond & Efstathiou 1984). With scale invariant primordial fluctuations, the power is well approximated by:

$$P(k)=\frac{A^2(t)k}{(1+\alpha k+\beta k^{3/2}+\gamma k^2)^2} \tag{1}$$

where $\alpha=1.7l, \beta=9.0l^{3/2}, \gamma=1.0l^2, l=(\Omega_0 h_{100}^2)^{-1}$ Mpc and the wavenumber, k is in $(\mathrm{Mpc})^{-1}$. The time dependence of A is given by linear theory but its amplitude is arbitrary (Peebles 1980, section 11). Here, we use the value of A^2 found by Davis *et al.* (1985) to give the best agreement of N-body experiments with observational data on the shape of the galaxy autocorrelation function, ξ, at the present epoch. They choose an initial amplitude for numerical reasons and then determine how much evolution is necessary to reach agreement with present observations. Taking the value 1.8

for the required expansion, which they propose for unbiased models, we compute a present epoch value of

$$A^2(t_0)=4.6\times10^3 h_{100}^{-8}\Omega_0^{-4}(\mathrm{Mpc})^4. \tag{2}$$

We use this value of A^2 for all of our initial conditions.* When applied to our models, the statistical measures discussed by Davis *et al.* (1985) should yield the same results, subject to the fact that their experiments follow trajectories accurately on much smaller length scales. Thus, all of our models have the correct shape for $\bar{\xi}$ at the present epoch over an appropriate range of length scales. The Hubble constant is fixed by requiring that the magnitude of $\bar{\xi}$ agree with observation. We scale our results according to the values determined by Davis *et al.* (1985): $H_0=22\,\mathrm{km\,s^{-1}}$ – Mpc for $\Omega_0=1.0$, $H_0=100\,\mathrm{km\,s^{-1}}$ – Mpc for $\Omega_0=0.2$. With these values for H_0, the distribution of peculiar velocities agrees moderately well with data in the $\Omega=0.2$ models, but not in the $\Omega=1$ models, in which case the peculiar velocities are excessive. We allow for the uncertainty in these values by reporting results in terms of h_{22} and h_{100} in the $\Omega_0=1.0$ and $\Omega_0=0.2$ models, respectively.

Given the amplitude of the power spectrum, we generate a set of $(64)^3$ mode amplitudes with independent Gaussian statisics, random phases and variance given by the power spectrum. We convert this to a particle distribution using the Zel'dovich approximation. The particles are initially placed on a cubic lattice (i.e. a distribution with the minimum possible noise). We compute the perturbations, $\Delta\mathbf{x}$, of the particles from their initial positions by requiring that

$$\nabla\cdot\Delta\mathbf{x}=\frac{\delta\varrho}{\varrho}. \tag{3}$$

This is easily solved by reducing it to Poisson's equation and using the Poisson solver that is used for the dynamics:

$$\Delta\mathbf{x}=-\nabla\varphi, \tag{4}$$

$$\nabla^2\varphi=\frac{\delta\varrho}{\varrho}. \tag{5}$$

The particle velocities must also be adjusted to correspond to the growing mode solution of the linear theory. This is achieved by adjusting the peculiar velocities so they are proportional to the perturbations, $\Delta\mathbf{x}$. The proportionality constant is the same for all particles and is given by:

$$\mathbf{v}_{\mathrm{pec}}=\frac{\dot{A}}{A}\Delta\mathbf{x}, \tag{6}$$

where $A(t)$ is the amplitude of density fluctuations in linear perturbation theory.

The particle distribution so constructed reliably realizes the desired power spectrum from length scales $L/N^{1/3}$ up to $L/2$ where L is the comoving size of the computational volume and N is the number of particles. The simulations were begun at an initial redshift of $z=8$, at which time all fluctuations were safely in the linear regime.

3 Time evolution

Evolution in time is carried out with a 'particle-mesh' code which uses the Fast Fourier Transform to solve Poisson's equation on a $(64)^3$ mesh with periodic boundary conditions. A 'cloud-in-cell'

* Peebles (1982a) has shown that the rms multipole moments of the microwave background intensity are proportional to A. Our value of A implies: $(|a_m'|^2)^{1/2}=8.7\times10^{-7}h_{100}^2$, which is smaller than the normalization used by Peebles by about a factor of 3.

scheme is used for mass assignment and force interpolation. The Green's function is the transform of the 7-point finite difference approximation to the three-dimensional Laplacian (Hockney & Eastwood 1981). The time evolution uses a 'leap-frog' scheme with 'time' variable $p=(1+z)^{-1/2}$ and a step size $dp=0.033$. [See Efstathiou *et al.* (1985) for a discussion of time variables, and the correct form of the equations.] The rather large time-step is made possible by the low-resolution particle-mesh algorithm. The force is effectively cut-off at a comoving length scale equal to the mesh size ($L/64$), so that tight binaries cannot occur and a large time-step is numerically stable.

While allowing for a large time-step and large numbers of particles, the particle-mesh algorithm restricts us to length scales larger than the mesh size. Thus, when our runs are compared at $z=0$ with those of Davis *et al.*, we find excellent agreement on large scales, but not on small scales where short-range interactions become important.

Some of the simulations (EdS1, EdS2, EdS3, 01, 02 and 03) were performed on the Caltech Hypercube, a parallel processor. The computational load, as well as the data storage was distributed over 32 'nodes' each of which is an Intel 8086/8087 microprocessor with 256 Kbytes of memory. They are connected by a communication network which is topologically equivalent to a five-dimensional hypercube with the processors on the corners (Fox, Lyzenga & Otto 1985). The speed-up, which is the ratio of the speed on a single processor to the speed on the multi-processor system is a measure of how effectively one is using the system. In our case the speed-up was in excess of 25, so we were effectively using the power of 25 microprocessors, the other seven being wasted in communication overhead, redundant calculation and/or idle time. An improved version of the hypercube is under development. Each of the new nodes will be more than 10 times faster than those of the current prototype, and will have 16 times the memory.

4 Correlation measurements

We 'observe' each simulation at three values of the redshift, z: 3.0, 1.25 and 0.0, by passing lines-of-sight through the computational volume. A particle is 'observed' if it is within a certain minimum distance of the line-of-sight, corresponding to the radius of the absorbing cloud. Proper diameters of 50, 100 and 200 kpc were simulated with no significant differences in the strength of the correlation. Although there is, as yet, no data for $z\lesssim 1.6$ the measurements at smaller redshifts are useful for discussion of evolutionary effects. Also, we expect that such data will become available in the near future, from satellite observations.

The light travel time across the computational volume is much smaller than the time-step, which in turn is much smaller than a dynamical time, so we may make the simplifying assumption that a line-of-sight passes through the simulation instantaneously. Consider a snapshot of the simulation at redshift z. A line-of-sight to a distant QSO is just a straight line through the computational volume. We take the particles of the simulation to be absorbers of finite size and record the positions and velocities of those close enough to the line-of-sight to be observed. If there are two or more objects on any given line-of-sight we compute a velocity splitting for every pair according to:

$$v_{\rm split}=H(z)\,\Delta x+\Delta v_{\parallel} \tag{7}$$

where $\Delta v_{\parallel}$ is the peculiar velocity difference, Δx is the proper separation along the line-of-sight and $H(z)$ is the Hubble constant at redshift z.

$$H(z)=H_0(1+z)\,(1+\Delta_0 z)^{1/2}. \tag{8}$$

A cold, uncorrelated system would have velocity splittings distributed uniformly up to a

maximum of:

$$v_{max} = \frac{L}{2(1+z)} H(z). \tag{9}$$

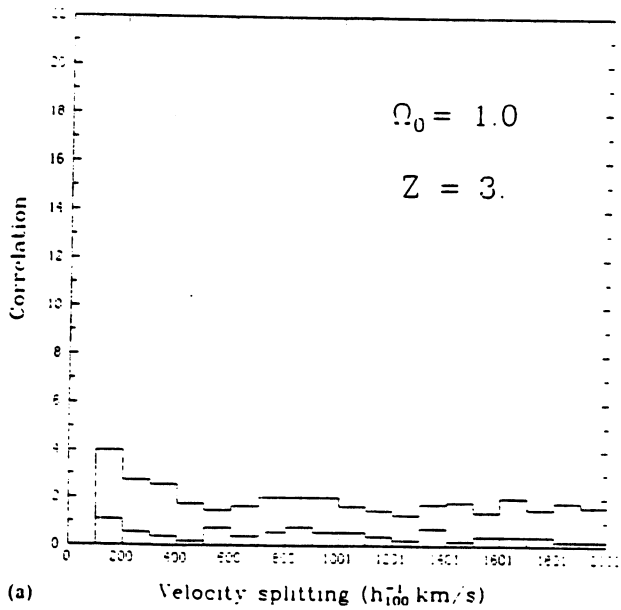
[Due to the periodic boundary conditions in the particle-mesh solver, the maximum separation at redshift z is $L/2(1+z)$.] The number of objects on a line-of-sight is a Poisson random variable with parameter $\lambda = (V_{los}/L^3) N$ where V_{los} is the comoving volume swept out by the line-of-sight. The average number of pairs is then:

$$\langle n_{pairs} \rangle = \sum_{n=0}^{\infty} \exp\left(-\lambda \frac{\lambda^n}{n!}\right) \frac{n(n-1)}{2} = \frac{\lambda^2}{2} \tag{10}$$

and the average number of pairs in a velocity bin of width Δv is $(\lambda^2/2)(\Delta v/v_{max})$. We take this as the background level and compute the velocity two-point correlation function as the fractional excess over this level:

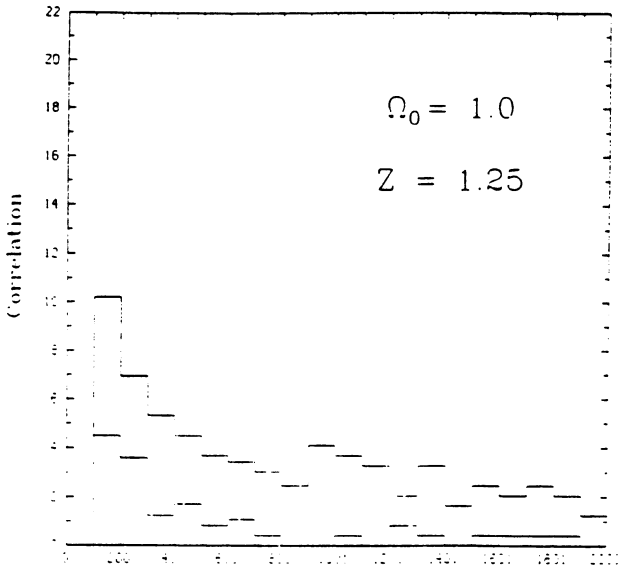
$$W(v) \equiv \frac{n_{pairs}}{\langle n_{pairs} \rangle}. \tag{11}$$

This is binned and plotted in Figs 1 and 2, which shows the largest and smallest values obtained in all runs with the listed parameters, varying the realization, particle radius, simulation scale, and particle number. These plots therefore give an idea of the total scatter in the calculations. We omit $v_{split} < 100\,km\,s^{-1}$ because we expect these to be unreliable due to Fourier mesh effects. Tables 1 and 2 list the parameters for each simulation. Despite variations in mass per cloud, M_{cloud}, by a factor of 30 and in cross-sectional area by a factor of 16, the scatter in Figs 1 and 2 is usually less than a factor of 2.



(a)

Figure 1. Line-of-sight velocity correlations for $\Omega_0 = 1$ models (a) at $z = 3$, (b) at $z = 1.25$ and (c) at $z = 0$. The upper and lower histograms are the largest and smallest estimates obtained from any single set of lines-of-sight, varying the cloud size, computational volume, power-spectrum realization and particle number. The true value is expected to lie beween these curves.



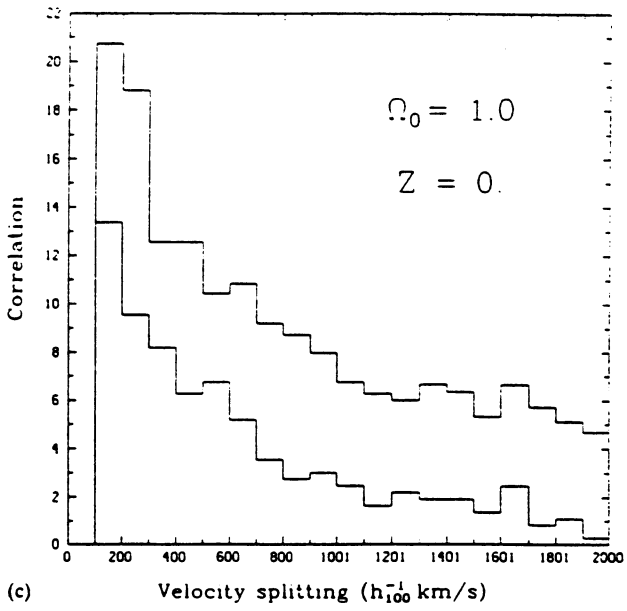
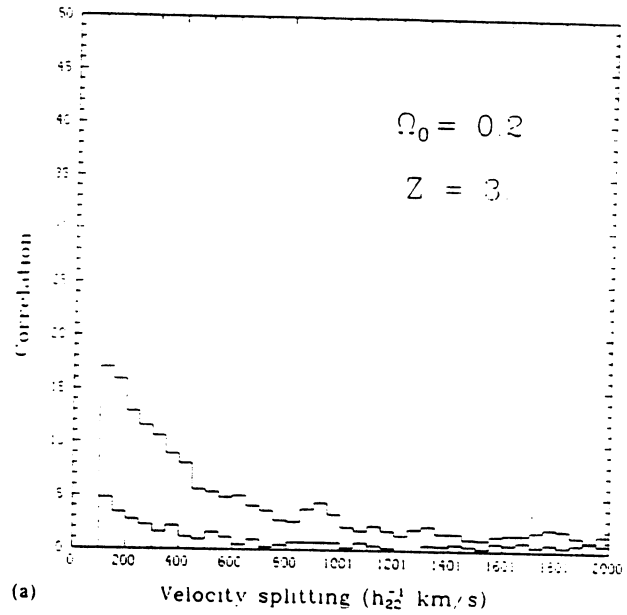


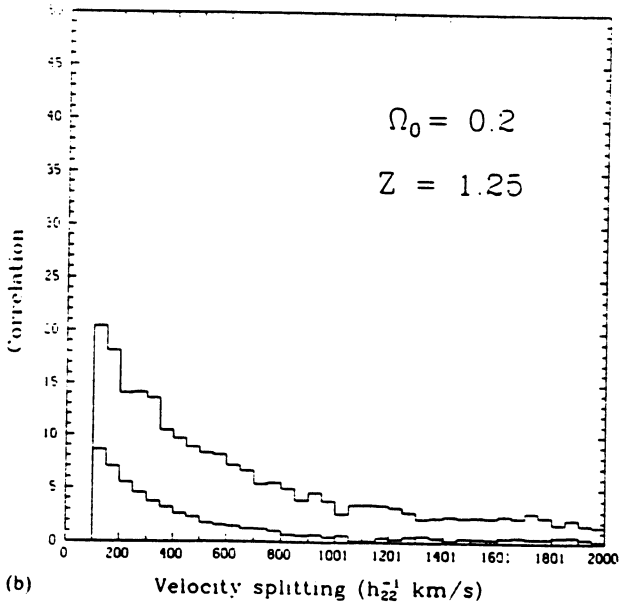
Figure 1 – *continued*

We have also searched for possible systematic trends which could indicate gross inaccuracies in the calculations. Tables 3 and 4 list W in a single velocity bin for each model at the three redshifts. Multiple entries are the results of different 'surveys' of the same model, i.e. independent sets of lines-of-sight yielded the different estimates of the correlation. Reading across Tables 3 or 4, one sees the dependence of W on hole diameter. W displays no significant trend with hole diameter

over a range of at least a factor of 2 and up to a factor of 4. Comparing entries in the same column (at the same redshift) reveals the dependence of W on particle number and box size. For example, models 04 and 05 have 21 952 and 262 144 particles respectively, but are identical in other respects. The differences in W beween them are comparable to the differences within each model. Similarly, models EdS5 and EdS6 have the same number of particles (262 144), but realize the



(a)



(b)

Figure 2. Same as Fig. 1, for Ω_0=0.2 models.

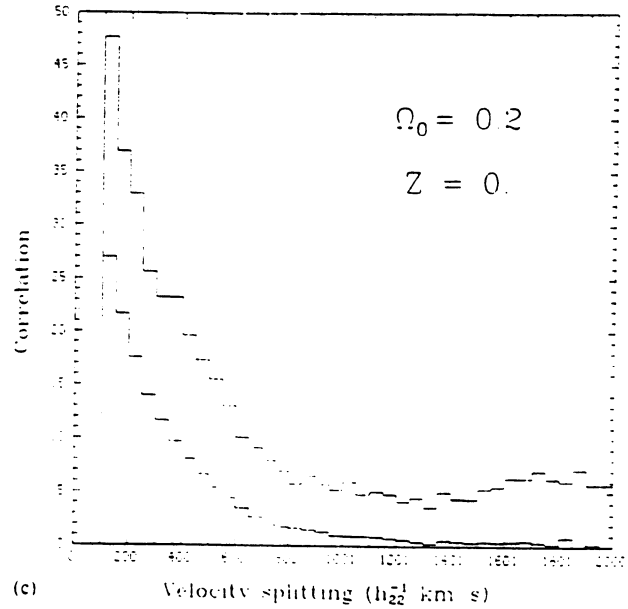


Figure 2 – *continued*

power spectrum on different length scales (L equal to 186 Mpc and 58 Mpc, respectively). Again, the variations between models are comparable to the variations within an individual model. We conclude that the true correlation for a large ensemble of realizations would probably lie within the range indicated by Figs 1 and 2. These figures also represent the range of correlations one expects in real data if the number of pairs per QSO is not large compared to unity, so that correlations are sampled from only a few clusters along the line-of-sight, as they are in our simulations.

The basic trends in Figs 1 and 2 are:

(i) In the $\Omega_0=0.2$ models the 'clouds' are strongly correlated in velocity space. We find, for example, $W=10\pm5$ at $z=3$ for velocities between 100–150 km s^{-1}. At $z=1.25$ the W has grown to 15 ± 5 for the same velocities. These values are in reasonable agreement with the data for C IV absorption systems (Sargent *et al.* 1980) and we may identify the clouds (particles) in these models with such systems. Recall that the initial conditions were arranged so that the particle distribution would resemble the current galaxy distribution at $z=0$, so the heavy-element systems may simply be galaxies in this scenario. The above interpretation sheds no light on the identity of the Ly-α clouds. In fact some mechanism must be invented to prevent them from following the mass and the galaxies.

(ii) In the $\Omega=1.0$ models the correlations are much weaker. In fact, we cannot reliably measure any correlation at $z=3$ on any velocity scale. We identify the particles in these simulations with Ly-α systems which similarly show little or no correlation in the data (Sargent *et al.* 1980). Thus, in an $\Omega=1.0$ CDM universe, the Ly-α systems trace the mass and the heavy-element systems are more clustered than the mass. The various proposals for 'biasing' which have been suggested for the galaxy distribution might well apply to the heavy-element systems as well. We discuss, qualitatively, such a scheme below.

(iii) Evolution in time is extremely rapid in the $\Omega_0=1$ models, and less dramatic but still significant in the $\Omega_0=0.2$ models. This behaviour can be understood by comparison with the self-similar evolution of correlations in a universe with power-law fluctuations.

5 Similarity solution

Let us leave the cold matter spectrum for the moment, and focus on power-law density fluctuation power spectra $|\delta_k^2| \propto k^n$, or $\delta\varrho/\varrho \propto M^{-1/2-n/6}$. For $\Omega=1$, gravitational clustering is then scale-free, and must evolve self-similarly in time (Peebles 1974; Davis & Peebles 1977; Efstathiou, Fall & Hogan 1979; Peebles 1980). This permits us to write the following scaling relation for W.

$$W(z_1, v)=W\left[z_2, \left(\frac{1+z_2}{1+z_1}\right)^{\beta} v\right], \tag{12}$$

where

$$\beta=\frac{n-1}{2n+6}. \tag{13}$$

This relation may be derived by requiring $W(v')$ to be constant where v' is defined by $\xi(v'/H)$=constant, or equivalently by requiring W to be always be the same on the velocity scale which currently has $\delta\varrho/\varrho=1$.

For a 'constant-curvature' spectrum $n=1$, the density contrast is constant in time at any fixed velocity, so as expected W is constant in time. This is the asymptotic form for the cold dark matter spectrum at late times or large scales. For $n<1$, W increases with time, and for $n\rightarrow -3$, which is the asymptotic form of the CDM spectrum at small scales, $\beta\rightarrow -\infty$. This accounts qualitatively for the rapid evolution of W observed in our models.

6 Discussion

Studies of this type will probably be most interesting as data bearing on the absorption cloud and galaxy distributions relative to the mass distribution, and on the nature of the absorbing clouds and the structure of the intergalactic medium. The scenarios, (i) and (ii) above, are already constrained by the QSO data. In the $\Omega=0.2$ scenario, a way must be found to prevent the Ly-α clouds from following the gravitational potential of the galaxies and dark matter. In the $\Omega=1$ scenario the Ly-α clouds may trace the mass distribution, but the fact that heavy-element clouds are already strongly clustered at high redshift seems to imply directly that at least some galaxy-like systems are laid down with substantially stronger correlations than the mass. We arrive at this conclusion entirely independently of any consideration of the present-day velocity distribution; it is a consequence of the weak correlations in the $\Omega=1$ model, which would, without biasing, imply smaller correlations than are observed in the absorption systems. Thus, QSO absorption studies provide a surprisingly effective discriminant between different cold dark matter models

At the present level of precision, the amount of biasing required to explain the QSO data in the $\Omega=1$ case appears to be similar to that required by Davis *et al.* (1985) to obtain agreement with the galaxy velocities. In their biasing prescription, galaxy correlations are little affected by gravitational clustering, and are essentially laid down as initial conditions. We may use this idea to compare clustering at two epochs, using the following illustrative (and unphysical) model for the biasing.

Suppose galaxy positions are fixed in comoving coordinates from $1+z=4$ to the present, and peculiar velocities may be altogether ignored. Then one would expect, in the absence of any dynamical clustering, a similarity relation of the form:

$$W(z_1, v)=W\left[z_2, \left(\frac{1+z_2}{1+z_1}\right)^{1/2} v\right], \tag{14}$$

so, for example, the clustering amplitude W at $z=3$ at $200\,\mathrm{km\,s^{-1}}$ should be that today at $100\,\mathrm{km\,s^{-1}}$. Within the observational errors, this prescription works for the QSO systems, since $W(z=0,\ 100\,\mathrm{km\,s^{-1}})\simeq\xi(1h_{100}^{-1}\mathrm{Mpc})\simeq 25$, which agrees with $W(z=2,\ v=100\text{–}200\,\mathrm{km\,s^{-1}})$ observed for the C IV lines. It is noteworthy that at a fixed velocity scale, the correlation amplitude for this non-dynamical clustering grows weaker with time, whereas that for gravitational clustering with $n<1$ (as with CDM) grows stronger.

The calculations presented here are up to the quality of presently published data, but better data are forthcoming and will require more thorough exploration of parameter space and elimination of several residual sources of error. If correlations are found in the Ly-α systems, they will be weak enough that significantly better calculations will be required to test them against model predictions.

Acknowledgments

We thank Peter Quinn for many useful discussions. This work was supported in part by a Bantrell Fellowship and NSF grant AST82-14126 at Caltech and NASA grant NAGW-763 at the University of Arizona (CJH) and by DOE grant DE-AS03-83ER13118 and a Shell Foundation Fellowship (JS). We are also grateful for the computer time provided by the Caltech Concurrent Computing Project.

References

Black, J. H., 1981. *Mon. Not. R. str. Soc.*, **197**, 553.

Blumenthal, G & Primack, J., 1983. In: *Fourth Workshop on Grand Unification*, p. 256, eds Weldon H. A., Langacker, P. & Steinhardt, P. J., Birkhauser, Boston.

Blumenthal, G., Faber, S. M., Primack, J. R. & Rees, M. J., 1984. *Nature*, **311**, 517.

Bond, J. R. & Szalay, A. S., 1983. *Astrophys. J.*, **274**, 443.

Bond, J. R. & Efstathiou, G., 1984. *Astrophys. J.*, **285**, L45.

Davis, M., Efstathiou, G., Frenk, C. & White, S. D. M., 1985. *Astrophys. J.*, **292**, 371.

Davis, M. & Peebles, P. J. E., 1977. *Astrophys. J. Suppl.*, **34**, 425.

Dekel, A., 1982. *Astrophys. J.*, **261**, L13.

Efstathiou, G., Davis, M., Frenk, C. & White, S. D. M., 1985. *Astrophys. J. Suppl.*, **57**, 241.

Efstathiou, G., Fall, S. M. & Hogan, C. J., 1979. *Mon. Not. R. astr. Soc.*, **189**, 203.

Fox, G., Lyzenga, G. & Otto, S., 1985. *Proceedings of the Computers in Engineering Conference*, Boston, August 1985, in press.

Fransson, C. & Epstein, R., 1982. *Mon. R. astr. Soc.*, **198**, 1127.

Hockney, R. W. & Eastwood, J. W., 1981. *Computer Simulation Using Particles*, McGraw-Hill International, New York.

Oort, J. H., 1981. *Astr. Astrophys.*, **94**, 359.

Peebles, P. J. E., 1974. *Astrophys. J.*, **189**, L51.

Peebles, P. J. E., 1980. *The Large Scale Structure of The Universe*, Princeton Univesity Press, Princeton.

Peebles, P. J. E., 1982a. *Astrophys. J.*, **263**, L1.

Peebles, P. J. E., 1982b. *Astrophys. J.*, **258**, 415.

Peebles, P. J. E., 1984. *Astrophys. J.*, **277**, 470.

Sargent, W. L. W., Young, P. J., Boksenberg, A. & Tytler, D., 1980. *Astrophys. J. Suppl.*, **42**, 41.

Young, P. J., Sargent, W. L. W. & Boksenberg, A., 1982. *Astrophys. J. Suppl.*, **48**, 455.

Zel'dovich, Ya. B., 1970. *Astr. Astrophys.*, **5**, 84.

F. Cubix

CUBIX: Programming Hypercubes without Programming Hosts

JOHN SALMON*

Abstract. Typically, application programs for hypercubes consist of two parts, a master process running on the host and a server running in the nodes of the hypercube.

CUBIX adopts a different viewpoint. Once a program is loaded into the hypercube, that program assumes control of the machine. The host process only serves requests for operating system services. Since it is no more than a file server, the host program is universal; it is unchanged from one application to the next.

This programming model has some important advantages.

- Program development is easier because it is not necessary to write a separate program for the host.
- Hypercube programs are easier to develop and debug because they can use standard I/O routines rather than machine-dependent system calls.
- Hypercube programs can often be run on sequential machines with minimal modification.

Versions of CUBIX exist for both crystalline and amorphous applications. In crystalline applications operating system requests occur synchronously. Requests are either "singular" or "multiple" according to whether all nodes request the same or distinct actions. In amorphous applications requests occur completely asynchronously. The host process serves each request independently. Care is taken so that many processors may simultaneously access the same file.

1. Introduction

CUBIX was created to make programming hypercubes easier. It's goal is to eliminate significant duplication of effort on the part of hypercube programmers, and to make the hypercube environment appear much more familiar to application programmers. It is also intended to make hypercube programs more easily portable to sequential machines as

* Physics Department, California Institute of Technology, Pasadena, CA 91125

well as between different brands of hypercubes.

The motivation for CUBIX can probably best be understood by sitting down with one's favorite hypercube and trying to get each of the nodes to perform a trivial task involving input and output to the console. For example, have each processor identify itself, and multiply its processor number by a number entered on the console, printing an informative message like:

"I am processor 17 and 3 times 17 is 51."

in response to the number 3 being entered. This is an extraordinarily difficult exercise because the nodes of the hypercube do not have direct access to the operating system facilities available on the host. One can not, for instance, execute a "scanf" in the nodes to obtain data from the console. Instead, the host (intermediate host, cube manager, control processor, etc.) must allocate a buffer, read data from the console into it, pass the contents of the buffer to the nodes, read a message for each node containing the results of that node's calculation, format those messages and print the results. Programming this exercise requires two programs, one for the host and one for the nodes of the cube, often compiled with different compilers and different compiler options.

This example is obviously frivolous, but it illustrates an important shortcoming in hypercube programming environments. Maintaining and debugging "real" programs is unnecessarily difficult for exactly the same reason as in the exercise: it is too hard to use the host's operating system. Debugging is extremely difficult because programs cannot be easily modified to produce output tracing the flow of control. Additionally, when a program is modified, it often requires separate but coordinated changes to both the node program and the host program. The necessary coordination is a rich source of minor bugs.

A further deficiency in the hypercube environments is the duplication of effort involved in this programming style. Each programmer is forced to reinvent a host-cube protocol which resembles, functionally at least, the protocols that have been written hundreds, if not thousands, of times already. After writing a few protocols, each programmer tends to develop a characteristic signature. Programmers quickly learn to reuse their 'main' routines, but by then, their time has already been wasted.

Finally, after expending the effort to develop an application on the hypercube, the programmer finds that the program will not run on a sequential machine. The I/O protocol designed for the cube is completely foreign to the sequential machine. Even though the bulk of the application would operate correctly by linking with a very simple library of dummy communication routines, the host program and node program must be "glued" back together. Maintaining an evolving code intended to run on both sequential machines and hypercubes is quite difficult for this reason. (Note that the program, once glued, no longer runs on the hypercube.)

All these deficiencies can be traced to a single source. Hypercubes are viewed as high-speed peripherals attached to a host computer which controls their operation. As peripherals go, they are extremely flexible and programmable, but control, nevertheless, resides in the host. The host loads programs and data into the cube, which then computes and eventually returns results which are expected, in number and length, by the host. In more sophisticated applications, the cube analyzes various tokens passed by the host and may perform different computations based on their values. This general organizational style is familiar to most hypercube programmers.

2. A different perspective

The basic idea behind CUBIX is that the program running in the cube should control the operation of the associated program running on the host. This is exactly opposite to the common style of programming discussed above. In CUBIX, tokens are passed from the cube to the host requesting activities like opening and closing files, reading the time-of-day clock, reading and writing the file system, etc. The host program does nothing more than read requests from the cube, act on them and return appropriate responses. All such requests are generated by subroutine calls in the cube. The host program which serves the requests is universal; it is unchanged from one application to the next, and the programmer need not be concerned with its internal operation.

It is convenient to give the cube subroutines the same names and calling conventions as the system calls they generate on the host. This relieves the programmer of the task of learning a new lexicon of system calls. Any operation he would have performed in a host program can be encoded in a syntactically identical way in the cube. It is of no consequence that the subroutine called in the cube will actually collect its arguments into a message, add a token identifying the request, and send the message to the host for action. All the programmer sees is a call to, e.g., *write(fd, ptr, cnt)*.

High-level utilities are often written in terms of a set of standard system calls. Since the CUBIX system calls have the usual names and calling sequences, system utilities designed for the sequential host computer can be readily ported to the hypercube. For example, the C Standard I/O Library can be compiled and linked with CUBIX allowing various forms of formatted and unformatted buffered I/O. Under CUBIX, the exercise of Section 1 would be programmed as:

```
#include <stdio.h>

main()
{
   int entry, pnum;

   pnum = /* machine dependent specification of local processor number */;
   scanf("%d", &entry);
   fmulti(stdout);                  /* see section 4. */
   printf("I am processor %d, and %d times %d is %d\n",
      pnum, entry, pnum, pnum*entry);
   exit(0);
}
```

3. The catch

It is highly optimistic to think that a set of system calls designed for a sequential computer can be sufficient for use in a parallel environment without modifications or additions. In fact, the requirements of the parallel environment do force one to restrict the use of some routines and also to add a few additional ones. The details differ markedly between crystalline and amorphous environments. The two cases will be taken up in the next two sub-sections. In both cases, the issue addressed is the same:

> How does one resolve the problem that different processors may need to do different things?

3.1 The crystalline case

Crystalline programs are characterized by uniformity from processor to processor and a computation that proceeds in loose lock-step. Synchronization is maintained by enforcing a rendez-vous whenever data is communicated between processors. Since loose synchronization is the norm in crystalline programs, it is not unreasonable to demand that system calls be made loosely synchronously. That is, it is permissible to call system subroutines whenever all communication channels are free. Furthermore, when a system call is made in one processor, it must be made in all processors at the same time, and with identical arguments. (Exceptions will be discussed shortly.) This neatly resolves the problem of how to deal with disparate requests from different nodes; such an event is declared to be in error.

Of course, there are times when different nodes need to request different actions from the host. The short program in Section 2 contains an example in which each processor attempts to print a different string. CUBIX adds two system calls, *mread* and *mwrite*, to the usual set to allow for distinct I/O operations to be performed by different processors. Both must be called loosely synchronously, but they may have different arguments in each node. Their effect is as follows:

mread(fd, ptr, cnt) causes cnt_0 bytes to be read from the file referred to by file descriptor *fd*, into the memory of processor 0 starting at ptr_0. The next cnt_1 bytes are read from the file into the memory of processor 1 starting at ptr_1, etc. Subscripts refer to the value of the argument in the corresponding processor.

mwrite(fd, ptr, cnt) behaves like *mread*, except that data is copied from the memory of the various processors to the file.

In C programs, it is much more common to use the the Standard I/O Library rather than to use system calls like *read*, *write*, *open* and *close* directly. Thus, it is crucial to enhance the Standard I/O Library so users can take advantage of *mread* and *mwrite* along with the usual system calls. In the Standard I/O Library, I/O is directed to *streams*, declared as pointers to type FILE. In CUBIX, streams have a new attribute called *multiplicity*. That is, streams can be in either the *singular* or *multiple* state. The functions, *fmulti(stream)* and *fsingl(stream)* are provided, which change the multiplicity of their argument to multiple and singular, respectively. Singular streams behave in the usual way, and are bound by the usual rules of loose synchronization and identical arguments. Multiple streams form the standard I/O interface to *mread* and *mwrite*. They allow the programmer to read and write data which is distinct in each node of the hypercube. Since output is buffered, queueing data for output to multiple streams need not be synchronous.

On the other hand, flushing the buffer must be done explicitly, and it must be synchronous. Flushing a multiple stream causes the data stored in each processor's buffer to appear in order of increasing processor number. The buffer associated with a stream may be flushed by calling one of *fflush*, *fclose* or *exit*, simultaneously in all the nodes of the hypercube. Since the programmer has control over when buffers are flushed, he can control,in detail, the appearance of his program's output. For example, the code fragment

```
fmulti(stdout);
printf("hello\n");
fflush(stdout);
printf("goodbye\n");
fflush(stdout);
```

```
printf("CUBIX ");
printf("is flexible\n");
fflush(stdout);
```

produces the following output when executed in all processors loosely synchronously:

```
hello
hello
-
hello
goodbye
goodbye
-
goodbye
CUBIX is flexible
CUBIX is flexible
-
CUBIX is flexible
```

Multiple input streams are not quite as flexible as output streams because the data must be available to the program when the input routine returns. This is in contrast to output routines which do not guarantee that the data has appeared on the output device upon return from the function. Thus, when input functions like *scanf* and *getc* are applied to multiple streams, each node reads as much of the input stream as necessary and then passes control on to the next node in sequence. The function, *ungetc*, when applied to multiple input streams replaces the last character read by the last processor.

3.2 The amorphous case

Amorphous (i.e. non-crystalline) programs are naturally asynchronous. It would be extremely inconvenient for the programmer to synchronize his calculation every time he wished to produce output or interact with the operating system. The processors in an amorphous CUBIX program are treated as though they are executing separate and independent processes. There is no notion of singular I/O, and there are no requirements of loose synchronization or identical arguments. Most system calls behave in a completely straightforward way when used in an amorphous CUBIX program, but the programmer must beware of asking for system resources too frequently. With currently available hosts, it would be easy to swamp the host's operating system if every node were to simultaneously request the same resource.

There is some difficulty, however, in maintaining numerous open files. If the host's operating system allowed CUBIX to allocate several hundred file descriptors, CUBIX could simply return a distinct file descriptor to every process that requested one. Unfortunately, there is a limit of about twenty simultaneously open files, so the CUBIX host program must remember what files are already open and avoid reopening them. There is still a limit of about twenty simultaneously open file *names*, which means that the programmer usually cannot open a different file for each processor in the cube.

When a file is opened by a processor, that processor's pointer into the file is unchanged by the activity of other processors. Each processor maintains some information about the files it has opened, including the current offset at which to begin the next read or write operation. When a read or write request is sent to the host, this information is sent as well, so the host can "seek" to the correct place before reading or writing the

data. Thus, each processor has complete control over the location of each byte it writes into the file. Using this system requires considerable care on the part of the programmer to keep processors using the same file from destroying one another's data. Nevertheless, such care often results in programs whose output is repeatable, so that the order of the data in output files does not depend on tiny variations in processor speed, etc. Aside from difficulty of use, there is another important disadvantage. In order for several processors to share a file, it must make sense for that file to have multiple pointers into it. This is simply not true of devices like terminals, to which data may only be appended.

The UNIX operating system provides for file output in *append* mode, in which each datum is placed at the end of the file, regardless of the offset of the process' current file pointer. CUBIX supports the same idea. Placing output files in append mode is a simple way of guaranteeing that data will not be lost because of several processors writing to the same offset. Output to files in append mode may also be directed to a terminal or other serial device. Append mode has the disadvantage that each record in the file must usually be tagged to indicate its originator. A system to automatically tag each record and record a "table-of-contents" at the end of the file upon closure is under development.

4. Experience with CUBIX

A crystalline version of CUBIX has been running at Caltech since early 1986. A version for amorphous applications was implemented about six months later. Since its introduction, CUBIX has become quite popular, and systems are now operating on the Caltech Mark II and Mark III machines as well as the Intel IpSC and the NCUBE. The prevailing attitude among users is that use of CUBIX is vastly simpler than the old host-cube protocols (even among persons not in the author's immediate family). Several programs have been written for which the same code can be compiled and run on a sequential machine, as well as a hypercube running CUBIX.

CUBIX's most significant drawback seems to be the increased code size in node programs. All computation that would have been done on the host is now done in the nodes of the hypercube. Although it is not any slower to perform inherently sequential tasks simultaneously in many processors, a copy of the code must reside in each processor. It is important to realize that both Standard I/O routines like *printf*, which usually does not appear in non-CUBIX programs, and application-dependent sequential code, which would have appeared in the host program, must now be included in the code that runs in every node. The size of this code can be significant, and reduces the amount of space available for data. The code and data linked by a call to *printf*, for example, requires about 6 kbytes on each node in our implementation. Measures can be taken to reduce the size of application-dependent sequential code. For example, filters can be used with UNIX pipes to massage the data prior to sending it into the cube, or after getting it back. So far, we have not needed more generality than that provided by simple input and output filters. Nevertheless, the possibility remains that in subsequent versions of CUBIX, application programs in the cube could call application-dependent subroutines linked into the host program.

5. Conclusion

Adopting the viewpoint that the program running in the nodes of the hypercube should control the behavior of the host program has some extremely desirable consequences.

- It is possible to write a universal host program which accepts commands generated by subroutine calls in the nodes of the hypercube.
- Given a universal host program, programmers only write one program (the one for the nodes) for any application, eliminating considerable labor and an annoying source of bugs.
- All details of the host-cube interface are hidden from the application programmer. Operating system services are obtained by system calls identical to those used on the host.
- Since applications require only one program to operate on the hypercube, it is usually a simple matter to run them on a sequential machine as a special case.
- Since operating system interaction is, for the most part, the same as in sequential programs, there is considerably less to learn before one can begin writing significant hypercube programs.

References

[1] Roger W. Hockney and James W. Eastwood. *Computer Simulation Using Particles*, Mcgraw-Hill International, New York, 1981.

[2] A. Leonard, "Computing Three-Dimensional Incompressible Flows with Vortex Elements," *Annual Rev. Fluid Mech.*, **17**, 523, 1985.

[3] J.J. Monaghan, "An Introduction to SPH," *Computer Physics Communications*, **48**, 89-96, 1988.

[4] G.A. Bird, *Molecular Gas Dynamics*, Clarendon Press, Oxford, 1976.

[5] S. T. O'Donnel and V. Rokhlin, "A Fast Algorithm for the Numerical Evaluation of Conformal Mappings," *SIAM J. Sci. Stat. Comp.*, **10**, 475, 1989.

[6] G.C. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, *Solving Problems on Concurrent Processors*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[7] W. Hackbusch and U. Trottenberg, *Multigrid Methods*, Springer-Verlag, Lecture Notes in Mathematics 960, Berlin, 1981.

[8] W. Briggs, *A Multi-grid Tutorial*, SIAM, Philadelphia, 1987.

[9] Andrew W. Appel, "An efficient Program for Many-Body Simulation," *SIAM J. Sci Stat. Comput.*, **6**, 85, 1985.

[10] I. Newton, *Mathematical Principles of Natural Philosophy and System of the World*, University of California press, Berkeley, 1962 (1st ed., 1687).

[11] Klaas Esselink, "About the order of Appel's Algorithm," Dept. of Computing Science, University of Groningen, Netherlands, Report CS 8910, June 1989.

[12] P. J. E. Peebles, *The Large-Scale Structure of the Universe*, Princeton University Press, Princeton, New Jersey, 1980.

[13] Leslie Greengard, "The Rapid Evaluation of Potential Fields in Particle Systems," PhD Thesis, Yale University, Computer Science Research Report YALEU/DCS/RR-533, 1987.

[14] Feng Zhao, "An O(N) Algorithm for Three-dimensional N-body Simulations," MIT Artificial Intelligence Laboratory, 995, 1987.

[15] F. Pépin, *Simulation of Flow Past an Impulsively Started Cylinder using a Discrete Vortex Method*, PhD. Thesis, California Institute of Technology, 1990.

[16] J. Barnes and P. Hut, "A Hierarchical O(NlogN) Force-Calculation Algorithm," *Nature*, **324**, 446-449, 1986.

[17] Joshua E. Barnes and Piet Hut, "Error Analysis of a Tree Code," *Astrophysical Journal (Suppl.)*, **70**, 389-417, June 1989.

[18] Lars Hernquist, "Performance Characteristics of Tree Codes," *Astrophysical Journal (Suppl.)*, **64**, 64, 1987.

[19] J. Makino and P. Hut, "Gravitational N-Body Algorithms: A Comparison Between Supercomputers and a Highly Parallel Computer," *Computer Physics Reports*, **9**, 199, 1989.

[20] J. Makino, "Comparison of Two Different Tree Algorithms," *Journal of Computational Physics*, **88**, 393, 1990.

[21] Lars Hernquist, "Vectorization of Tree Traversals," *Journal of Computational Physics*, **87**, 137, 1990.

[22] J. Barnes, "A Modified Tree Code: Don't Laugh; It Runs," *Journal of Computational Physics*, **87**, 161, 1990.

[23] L. Greengard and V. I. Rokhlin, "A Fast Algorithm for Particle Simulations," *J. Comput. Phys.*, **73**, 325, 1987.

[24] L. Greengard, "Potential Flow in Channels," *SIAM J. Sci. Stat. Comput.*, **11**, 603, 1990.

[25] J. G. Jernigan, in *IAU Symposium 113, Dynamics of Star Cluseters*, ed. P.

Hut, 275, Reidel, Dordrecht, 1985.

[26] David H. Porter, *A Study of Hierarchical Clustering of Galaxies in an Expanding Universe*, PhD. Thesis, University of California, Berkeley, 1985.

[27] J.G. Jernigan and D.H. Porter, "A Tree Code with Logarithmic Reduction of Force Terms, Hierarchical Regularization of All Variables and Explicit Accuracy Controls," *Astrophysical Journal (Suppl.)*, , 871, 1989.

[28] K. Chua, A. Leonard, F. Pépin, and G. Winckelmans, "Robust Vortex Methods for Three-Dimensional Incompressible Flows," in *Proceedings of Symposium on Recent Advances in Computational Fluid Dynamics*, ASME Winter Meeting, Chicago, 1988.

[29] Feng Zhao and S. Lennart Johnsson, "The Parallel Multipole Method on the Connection Machine," Thinking Machines Corp, CS89-6, 1989.

[30] Jacob Katzenelson, "Computational Structure of the N-Body Problem," *SIAM J. Sci. Stat. Comput.*, **10**, 787-815, July, 1989.

[31] W. Benz, "Applications of SPH to Astrophysical Problems," *Computer Physics Communications*, **48**, 97, 1988.

[32] W. Benz, R. L. Bowers, A. G. W. Cameron, and W. H. Press, "Dynamic Mass Exchange in Doubly Degenerate Binaries I. 0.9 and 1.2 Msolar Stars," *Astrophysical Journal*, **348**, 647, 1990.

[33] W. H. Press, "Techniques and Tricks for N-Body Computation," in *The Use of Supercomputers in Stellar Dynamics*, ed. S. McMillan, 184, Springer Verlag, 1986.

[34] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*, Dover Publications, New York, 1972.

[35] W. Jaffe, "A Simple Model for the Distribution of Light in Spherical Galaxies," *Mon. Not. Roy. Astron. Soc.*, **202**, 995, 1983.

[36] D.E. Knuth, *The Art of Computer Programming vol. 1 Fundamental Algorithms*, Addison Wesley, Reading, Mass., 1973.

[37] L. D. Landau and E. M. Lifshitz, *Mechanics*, Pergamon Press, Oxford, 1960.

[38] L. Hernquist and N. Katz, "TREESPH - A Unification of SPH with the

Hierarchical Tree Method," *Astrohysical Journal (Suppl.)*, 70, 419, 1989.

[39] G. B. Thomas, *Calculus and Analytic Geometry*, Addison-Wesley, Reading, Mass., 1968.

[40] G. C. Fox, "A Graphical Approach to Load Balancing and Sparse Matrix Vector Multiplication on the Hypercube," in *Numerical Algorithms for Modern Parallel Computer Architectures*, ed. M. Schultz, 37-62, Springer-Verlag, 1988.

[41] John Salmon, "Binary Gray Codes and the Mapping of a Physical Lattice into a Hypercube," California Concurrent Computation Project, Report 051, 1984.

[42] W. Furmanski and G. C. Fox, *Hypercube Algorithms for Neural Network Simulation: the Crystal Accumulator and the Crystal Router*, ACM Press, New York, 1988.

[43] ParaSoft Corp., *ExPress Programmer's Reference Manual*, ParaSoft Corp., Pasadena, 1990.

[44] Mike Warren, private communication, 1990.

[45] M.T. Heath, *Hypercube Multi-Processors 1987*, SIAM, Philadelphia, 1987.

[46] *Proceedings of the Third Conference on Hypercube Computers and Applications*, ACM Press, New York, 1988.