Efficient Shadowing of High Dimensional Chaotic Systems with the Large Astrophysical N-body Problem as an Example

by

Wayne Hayes

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Efficient Shadowing of High Dimensional Chaotic Systems with the Large Astrophysical N-body Problem as an Example

Wayne Hayes
Master of Science
Department of Computer Science
University of Toronto
January 1995

Abstract

N-body systems are chaotic. This means that numerical errors in their solution are magnified exponentially with time, perhaps making the solutions useless. Shadowing tries to show that numerical solutions of chaotic systems still have some validity. A previously published shadowing refinement algorithm is optimized to give speedups of order 60 for small problems and asymptotic speedups of $O(N)$ on large problems. This optimized algorithm is used to shadow N-body systems with up to 25 moving particles. Preliminary results suggest that average shadow length scales roughly as $1/N$, *i.e.,* shadow lengths decrease rapidly as the number of phase-space dimensions of the system is increased. Some measures of simulation error for N-body systems are discussed that are less stringent than shadowing. Many areas of further research are discussed both for high-dimensional shadowing, and for N-body measures of error.

Acknowledgements

I would like to thank my supervisor, Prof. Ken Jackson, for many insightful conversations during the development of the ideas in this thesis, and for helpful comments on the text. My second reader, Dr. Min Hu, also made many helpful suggestions.

I would like to thank Gerry Quinlan and Scott Tremaine for the paper that inspired this thesis, as well as for the many discussions carried on long-distance via e-mail. Gerry also provided me with his original shadowing refinement code, and a postscript version of his figure 6, my figure 3.1. All this was only possible because of the Internet, so I'd like to thank all the organizations worldwide responsible for its upkeep.

I would like to thank Prof. Mart Molle, my supervisor during my foray into Computer Networks, for his wisdom, friendship, understanding, and for teaching me how to be a (successful?) researcher. Thanks, "Dad".

I'd like to thank Nick Trefethen for being a good devil's advocate, and Wayne Enright for offering his 7/8 pair explicit Runge-Kutta integrator, although I haven't used it yet.

For financial support, I'd like to thank the Department of Computer Science and its professors at the University of Toronto. Computer time for the simulations and shadowing runs performed in this thesis were provided in part by Fujitsu Canada, the High Performance Computing Centre of Calgary, and the University of Calgary. The University of Toronto's Computing Disciplines Facility (*CDF*) and Department of Computer Science Labs provided local computer facilities.

Finally, I'd like to thank Robert M. Pirsig, author of the book *Lila*, for his "slips of paper" method of organization. The method consists of transcribing each individual idea onto one slip of paper, and then organizing all the slips of paper into a good order of presentation. This method allowed me to organize the ideas herein with fine-grained detail in only 8 days; writing the thesis was then an $O(n)$ process, with n being the number of slips. Transcribing the slips into LaTeX took just over 2 weeks, followed by 1 week of editing and brushing up. I'm told this is quite a short time to write up a Master's thesis. I would highly recommend this method (and indeed the entire book) to anybody.

Contents

Chapter 0

Introduction and Motivation

"Prediction can be difficult, particularly about the future."
— Mark Twain

Astrophysical N-body systems are chaotic. In other words, they have sensitive dependence on initial conditions. This means that the phase-space distance between two solutions whose initial conditions differ by an arbitrarily small amount will increase exponentially with time. Since computers constantly make small errors in the computation of such solutions, it is guaranteed (with probability 1) that a computed solution will diverge exponentially from the true solution with the same initial conditions. Thus, it is possible that numerical solutions for chaotic systems are overwhelmed by exponential magnification of small errors, which might mean computed solutions are worthless. This could be the case even if quantities such as energy or momentum are conserved to arbitrary accuracy, because there are infinitely many solutions whose energy is exactly the same, but have vastly different phase space trajectories.

0.0 Introduction

Numerical simulation is a standard tool in the repertoire of the modern physical scientist's study of complex systems. The astronomical literature is brimming with results of large N-body simulations. The proliferation of titles such as "Simulations of Sinking Galaxies" [33], "Dissipationless Collapse of Galaxies and Initial Conditions" [23], "A Numerical Study of the Stability of Spherical Galaxies" [24], "The Global Stability of Our Galaxy" [30], and "Dynamical Instabilities in Spherical Stellar Systems" [4] shows that much trust is placed in the results of these simulations.

Can these simulation results really be trusted? What conditions must a simulation meet for its accuracy to be assured? Is there a limit on the length of simulated time a system can be followed accurately? What measures can be used to ascertain the accuracy of a simulation? More fundamentally, what do we mean by "accuracy" and "error" in these simulations?

All these questions have been addressed in the past with varying degrees of success. There is still some controversy about whether simulations of N-body systems can be trusted. The main reason for this concern is that N-body systems are *chaotic* — small perturbations in the phase-space co-ordinates at any point result in a vastly different phase-space solution a short time later. Given that numerical computations are constantly introducing small errors to the computed solution, we must naturally ask what effect these errors have on computed solutions.[0]

0.1 History of exponential divergence in N-body systems

Miller [26] was the first to show that small changes in the initial conditions of an N-body system result in exponentially diverging solutions. Lecar [22] co-ordinated a study between many researchers, each of whom independently computed the solution to an N-body problem with identical initial conditions. They found that different algorithms and computers gave results in which some measures differed by as much as 100%. More recent work on the growth of errors includes Kandrup and Smith [19], who showed that under a large range of parameters, the time scale over which small perturbations grow by a factor of e (Euler's constant, $2.7182\ldots$), called the e-folding time[1], is comparable to the *crossing time* (the average time it takes a particle to cross the system once). Goodman, Heggie, and Hut [10] developed a detailed theory of the growth of small perturbations, and verified it with simulation to show that the exponential growth of small errors results mostly from close encounters, which occur infrequently. This is an interesting result because it says that, even though the full phase-space solutions may experience exponential error growth, this growth is due mainly to a few particles that undergo close encounters. Perhaps the results of collisionless systems can be trusted for longer than collisional ones, since close encounters have a much smaller effect in the former. Kandrup and Smith [20] showed that as softening is increased (*i.e.,* the collisionality is decreased), the e-folding time grows slightly faster than linearly. (*i.e.,* the Lyapunov exponent decreases, so errors are magnified more slowly.) They agree with Goodman *et al.* that the error magnification is due more to the rare individual particles whose errors grow much more quickly than the average, although they claim the global potential also plays a role. Presumably the particles, whose errors grow more quickly than average, are ones that have suffered close encounters.

Since the time-scale for the growth of errors is so short (the errors can be magnified by a factor of ~ 10 each crossing time), the results of all N-body simulations may be suspect. If the relative error per crossing time for a simulation is 10^{-p}, then after about p crossing times, a particle's position will have an error comparable to the size of the system — in other words, all information will have been lost about the particle's position. Typical simulations today have a p between 4 and 8.

I am not aware of *any* convincing justification for the belief that statistical measures taken from these simulations are valid, although I tend to share the same "warm fuzzy" feelings that most astronomers have — namely that, in some sense, large N-body simulations give valid statistical results. The problem is, in *what* sense are they valid, if any; *how* valid are they; and finally, *how*

[0] There is a subspace of measure zero, called the *stable* subspace, in which small perturbations do not result in vastly different solutions. But since it has measure zero, the probability of random numerical errors being restricted to this subspace is zero.

[1] The e-folding time is $1/\lambda$, where λ is the Lyapunov exponent

badly are we allowed to integrate before validity is lost?

0.2 The kinds of errors made in N-body simulations

0.2.0 Input and output Errors

I first distinguish between two general types of errors.

Input errors are errors that can be controlled directly while devising and implementing models of N-body systems. Input errors in N-body systems may also be called *approximations*, and may be divided into modelling approximations and implementation approximations. *Modelling* approximations simplify the system being simulated, and include:

- finite-N sampling, also called *discreteness noise*, because the N used is generally several orders of magnitude less than the N of the real system being modelled. The general consensus seems to be that this is the limiting source of error in current large N-body simulations. For example, Hernquist, Hut and Makino [15], Barnes and Hut [5], and Sellwood [31] explicitly say this; Singh, Hennessy and Gupta [32] have a "master error equation" in which clearly discreteness noise is dominant; and Pfenniger and Friedl [27], Jernigan and Porter [17], and Barnes and Hut [3] all imply that using the largest possible N is a desirable characteristic.
- force softening, *i.e.,* replacing r^2 with $(r^2 + \varepsilon_{soft}^2)$ in the denominator of the gravitational force computation for some small constant ε_{soft}, usually chosen to approximate the average inter-particle separation. This is done because it allows a smaller N to approximate a larger N, and also to eliminate the singularity at $r = 0$ [6].
- reducing the dimensionality of the problem from 3 to 2, if applicable. This is not done as often as in the past.

Implementation approximations measure how well the implementation simulates the model, and include such things as

- numerical integration truncation error.
- machine roundoff error.
- Using approximate force computation algorithms like the Barnes-Hut tree code [3] or the Fast Multipole Method [12]. Hernquist, Hut, and Makino [15] try to show that the effect of this error is negligible, by showing that the energy of each individual particle is conserved to a high degree regardless of whether the Barnes-Hut or the direct $O(N^2)$ algorithm is used. However, they used the leapfrog integrator with a constant timestep, which guarantees that the energy error for the entire system is bounded. I'm not sure if this integrator guarantees that the energy of individual particles is conserved; if it does, the results may not be as conclusive as they appear. Furthermore, as discussed below, energy conservation may not be a stringent enough error criterion.

Output error measures the difference between the output and a real system, and results from the cumulative effect of all the input errors. A simulation with small output errors is said to have high *accuracy.* Given that N-body systems are chaotic, and that their simulation introduces the above input errors, we must now ask precisely what we *mean* by the "accuracy" of a simulation. Amazing as it may seem, there is currently *no clear definition* of simulation accuracy [29]. Obviously, attempting to follow the individual paths of all N particles is infeasible; Goodman, Heggie and Hut [10] show that this would require $O(N)$ digits of precision. On the other hand, most astronomical publications quote energy conservation as their only measure of output error, even though there are infinitely many solutions with equal energy but vastly different phase-space trajectories. Some of these simulations even use an energy-conserving integrator like leapfrog, in which case quoting energy conservation is of dubious merit, because the integrator conserves energy no matter how big other errors become!

0.2.1 Macroscopic statistics *vs.* microscopic details

In large N-body simulations, one is not usually concerned with the precise evolution of individual particles, but instead with the evolution of the distribution of particles.[2] Most practitioners know that the exponential magnification of errors means they cannot possibly trust the microscopic details, but they believe that the *statistical* results are independent of the microscopic errors, although little work has been done to test this belief [10]. Barnes and Hut [5] claim that astrophysical N-body simulations require only "modest" accuracy levels, but also concede that quoting energy conservation isn't enough, and that more stringent tests are needed.

An example of conservation of macroscopic properties is given by Kandrup and Smith [19]. They show that a histogram of the e-folding times of individual particles stays constant within statistical uncertainties, even though the phase-space distribution of those particles is vastly different for different initial conditions.

However, until more stringent tests are applied to N-body simulations, we'll never know, for example, if our simulations of spiral galaxies produce spirals for the same reason that real spiral galaxies do.

0.2.2 Suggestions for measures of output error

We now must distinguish between the *desired* properties of simulations, and deviations that simulations make from those properties, *i.e.,* the output errors they make.

0. We could demand that a simulation precisely follow the exact evolution of the true solution.

   This would be possible in principle for chaotic maps, but not for ODEs. For maps, an arbitrary precision arithmetic package could be employed, but this is infeasible because it requires keeping all the digits of every operation, and each multiplication operation typically doubles the number of digits.

[2] This is in stark contrast to N-body simulations of our solar system, in which case we are interested in the precise evolution of the *particular* solar system we live in, not a hypothetical one very similar to it.

For problems in which the map is really an ODE integration, like N-body systems, this is not possible even in principle, because no numerical integration routine is known that can give exact solutions to arbitrary nonlinear ODEs.

1. Next, we could demand that a simulation have the property that the phase-space distance between the computed solution and the true solution with the same initial conditions be bounded by a small constant.

   If this property could be realized, all our troubles would be over, for it is a *sufficient* condition under any reasonable definition of simulation validity. Unfortunately, ODE integrations have truncation errors, so the magnitude of these errors will be magnified exponentially on a short time scale. Goodman, Heggie and Hut [10] offer some solace in that the exact evolution could be closely followed for a long time if $O(N)$ digits are kept, but this is currently infeasible. There seems little hope of obtaining valid simulations by this criterion.

2. At the least stringent extreme we can demand that such physically conserved quantities as energy, and linear and angular momentum, are conserved to some small error.

   Certainly global conservation of these quantities is *necessary* for any reasonable definition of simulation validity, but it is unclear what other properties are implied by such conservations.

There are several possibilities between these extremes.

3. A little less stringent than (1), we can demand that a simulation follow, with some small error, the exact phase-space evolution of a set of initial conditions that is close to the actual set of initial conditions used in the simulation.

   Since, with large simulations, we are only interested in the evolution of the distribution of particles, and since the initial conditions are usually generated from some random distribution anyway, this is almost as good as option (1). The study of shadowing relates precisely to this property.

What if shadowing turns out also to be an unattainable goal? We will need to demand less stringent properties of simulations, such as:

4. Even if a shadow solution cannot be found for a particular simulation, it is still possible that the global distribution of particles is close to the exact global distribution of a true solution with similar initial conditions. In other words, a low-resolution, "smoothed" animation of the real system and the computed solution would be indistinguishable to the unaided eye.

   This would be almost as good as shadowing, at least for collisionless systems, but it is unclear how one would go about proving the existence of this property for a simulation.

5. If the global distribution cannot be followed closely, then perhaps at least some statistical properties of the distribution could be reproduced by a simulation. A reasonable statistical property may be something like the time-evolution of the Fourier spectrum of the density distribution in spherical or cylindrical co-ordinates, so that the wavelengths of the distribution (*i.e.,* the relative abundance of "clumpiness" of various sizes) are similar to those of the real system.

This is the least stringent property, that I can think of, that a large N-body simulation would need to be considered valid; *i.e.,* it is the weakest *necessary* condition I can think of. Note it is more stringent than energy conservation. However, it is again unclear how one would go about proving this property exists for a simulation.

It should be clear that the ordering in stringency of the above properties, from most to least stringent, allows the following logical deductions: $0 \Rightarrow 1 \Rightarrow 3 \Rightarrow 4 \Rightarrow 5 \Rightarrow 2$.

0.3 Summary of this thesis

Chapter 1 will more formally introduce and explain shadowing. Chapter 2 is the main body of the work, which introduces the optimizations to the shadowing algorithm of Quinlan and Tremaine [29], and talks about high-dimensional shadowing in general. Chapter 3 contains some preliminary results on high-dimensional shadowing that will be of interest both to practicing N-body astronomers, and to those interested in shadowing in general. Chapter 4 contains descriptions of the many open questions and further avenues of research in this area. The appendix contains a review of the basic N-body equations and the derivation of the Jacobian of an N-body system in which M particles move amongst $N - M$ fixed ones, and the Jacobian of such a system.

Chapter 1

Shadowing

"It is in vaine to goe about to make the shadowe straite, if the bodie whiche giveth the shadowe bee crooked."
— Stefano Guazzo, *Civile Conversation.* (1574)

Shadowing is a branch of chaotic systems theory that tries to show that, even in the face of the exponential magnification of small errors, numerical solutions have some validity. It does this by trying to show that, for any particular computed solution (the "noisy" solution), there exists a true *solution with slightly different initial conditions that stays uniformly close to the computed solution. If such a solution exists, it is called a true shadow of the computed solution. An approximation to true shadowing is numerical shadowing, whereby an iterative refinement algorithm is applied to a noisy solution to produce a nearby solution with less noise. If this iterative process converges to a solution with noise close to the machine precision, the resulting solution is called a numerical shadow. Numerical shadowing is very compute intensive, because it requires the storage and manipulation of the full phase-space trajectory of the system, at much higher precision than the original computation.*

1.0 Introduction

1.0.0 Definitions

Throughout this thesis, when referring to mathematical variables, **boldface** will refer to vectors, and *italic* will refer to scalars, matrices and functions. Scalars are written in *small* letters and matrices in *CAPITALS*.

Some of the following definitions are taken, with minor modifications, from Grebogi, Hammel, Yorke, and Sauer [11], hereinafter referred to as GHYS. The terms *trajectory*, *orbit*, and *solution* are used interchangably throughout this thesis.

Definition. A *true trajectory* $\{\mathbf{x}_i\}_{i=a}^{b}$ of f satisfies $\mathbf{x}_{i+1} = f(\mathbf{x}_i)$ for $a \leq i < b$. We are interested in the case where a and b are finite integers. For a chaotic map, f may be a simple equation, such as the logistic equation $f(x) = 1 - 2x^2$, which always maps the interval $[-1, 1]$ onto itself. For an ODE system like the N-body problem, $f(\mathbf{x})$ represents the true solution of integrating the phase-space co-ordinates $\mathbf{x}$ for one timestep.

Definition. $\{\mathbf{p}_i\}_{i=a}^{b}$ is a *δ_f-pseudo-trajectory*, also called a *noisy orbit*, for f if $|\mathbf{p}_{i+1} - f(\mathbf{p}_i)| < \delta_f$ for $a \leq i < b$, where δ_f is the noise amplitude.

Definition. For $a \leq i < b$, the *1-step error* made between step i and step $i+1$ of the pseudo-trajectory $\{\mathbf{p}_i\}_{i=a}^{b}$ is the vector $\mathbf{e}_{i+1} = \mathbf{p}_{i+1} - f(\mathbf{p}_i)$. Thus, a true trajectory is one whose 1-step errors are identically zero.

Definition of shadowing. The true trajectory $\{\mathbf{x}_i\}_{i=a}^{b}$ *$\delta_{\mathbf{x}}$-shadows* $\{\mathbf{p}_i\}_{i=a}^{b}$ on $a \leq i \leq b$ if $|\mathbf{x}_i - \mathbf{p}_i| < \delta_{\mathbf{x}}$ for $a \leq i \leq b$. The two stages, of proving that a shadow exists, are *refinement* and *containment*, defined below.

Definition. A *shadow step* is an interval, that can be larger than the internal timestep of a numerical integrator, across which a 1-step error is computed.

Definition. The pseudo-trajectory $\{\mathbf{p}_i\}_{i=a}^{b}$ has a *glitch* at point $i = G_0 < b$ if for some relevant $\delta_{\mathbf{x}}$ there exists a true trajectory that $\delta_{\mathbf{x}}$-shadows $\{\mathbf{p}_i\}_{i=a}^{b}$ for $a \leq i \leq G_0$, but no true trajectory that $\delta_{\mathbf{x}}$-shadows it for $a \leq i \leq G$, for $G > G_0$.

The first group of chaotic systems for which it was proven that shadow orbits exist was *hyperbolic* systems [2, 7]. In a 2-dimensional hyperbolic system, there are two special directions called the *unstable* (or *expanding*) and the *stable* (or *contracting*) directions, which are generally not orthogonal. Small perturbations along the stable direction decay exponentially in forward time, while small perturbations in the unstable direction grow exponentially in forward time. The two directions reverse roles in backwards time. A "trajectory" for such a system can be imagined as a point moving in a 2D plane, evolving through a third dimension, representing time.

For such a system it was shown that, if the angle between the stable and unstable directions is uniformly bounded away from 0, then a noisy trajectory can be shadowed for all time. For non-hyperbolic systems, it appears that shadows may exist only for finite time. The most important question in this regard is, how *long* can a noisy orbit be shadowed? If the time is at least as long as most typical simulations of chaotic non-hyperbolic systems, then simulations have great validity; if the shadowing time turns out to be too short, then a less stringent error measure must be resorted to, such as one listed in Chapter 0.

Definition. *Refinement* is an iterative process that perturbs each point of a noisy orbit in an attempt to produce a nearby orbit with less noise. A refinement iteration is *successful* if the trajectory before the iteration has noise δ_f^0 and the trajectory after the iteration has noise δ_f^1, and $\delta_f^1 < \mu\delta_f^0$, for some reasonable $\mu \in (0, 1)$. Otherwise the iteration is *unsuccessful*.

Definition. *Containment* is a rigorous method to prove the existence of a shadow orbit. See GHYS for details. Although this is a good area for further work, containment is beyond the scope of this thesis.

The refinement algorithm that concerns us in this thesis is the one first presented in two dimensions by GHYS, and generalized to handle arbitrary Hamiltonian systems by Quinlan and Tremaine [29], hereinafter referred to as QT.

QT make the distinction between *dynamical noise* and *observational noise.* Observational noise does not effect the future evolution of the system. Laboratory measurements of a macroscopic system are usually of this type; another example is computer output that prints fewer digits than are represented internally. In contrast, dynamical noise does effect the future evolution of the system. The noise introduced by numerical solution of a system of ODE's is dynamical. QT studied some existing noise-reduction algorithms in an attempt to refine noisy trajectories of chaotic systems, but none worked as well as that presented by GHYS. QT postulates that this may be because the other noise reduction algorithms were designed to reduce observational noise, whereas the GHYS procedure was designed to reduce dynamical noise in a chaotic system.

1.1 The refinement procedure of GHYS

1.1.0 Introduction

The refinement procedure of GHYS and QT can be likened to Newton's method for finding a zero of a function. Indeed, it may be possible to formulate it *exactly* as a Newton's method with boundary conditions, although this has not yet been shown. For pedagogical purposes, I will assume in the following paragraph that the GHYS refinement procedure can be formulated as a Newton's method.

The basic idea is as follows. Let $\mathbf{P} = \{\mathbf{p}_i\}_{i=0}^{S}$ be a trajectory with S steps that has noise $\delta > \eta\varepsilon_{mach}$, where ε_{mach} is the machine precision, and η is some constant significantly greater than 1 that allows room for improvement towards the machine precision. Let $\mathbf{e}_{i+1} = \mathbf{p}_{i+1} - f(\mathbf{p}_i)$ be the 1-step error at step $i+1$, where $|\mathbf{e}_{i+1}| < \delta$ for all i. The set of 1-step errors is represented by $\mathbf{E} = \{\mathbf{e}_i\}_{i=1}^{S}$, and is estimated by a numerical integration technique that has higher accuracy than used to compute $\mathbf{P}$. This describes a function, call it g, taking as input the entire orbit $\mathbf{P}$ and whose output is the set of 1-step errors $\mathbf{E}$, *i.e.,* $g(\mathbf{P}) = \mathbf{E}$. Since the 1-step errors are assumed to be small, $|\mathbf{E}|$ is small. That is, $\mathbf{P}$ is close to a zero of g, if one exists. A zero of the function would represent an orbit with zero 1-step error, *i.e.,* a true orbit. This is an ideal situation in which to run Newton's method. If Newton's method converges, then a true orbit has been found.

There are exactly two criteria for a trajectory $\mathbf{P}'$ to be called a *numerical* shadow of $\mathbf{P}$:

0. $\mathbf{P}'$ must be substantially less noisy than $\mathbf{P}$. Until further work can show a less stringent condition, we will assume that $\mathbf{P}'$ requires noise comparable to the machine precision. In other words, $\mathbf{P}'$ must have small 1-step errors.

1. The distance between $\mathbf{P}'$ and $\mathbf{P}$ must be bounded by some appropriately chosen constant $\delta_{\mathbf{P}'}$ that is small in comparison to the typical scale of the system. For example, for an N-body system roughly confined to the unit box, a reasonable maximum shadow distance is comparable to the inter-particle separation, or about 10^{-2}, whichever is smaller. In essence,

this says that if a human observer can't see the difference between two paths viewed on a figure with the unaided eye, then the distance is probably small enough.

Any trajectory satisfying these criteria, no matter what algorithm or heuristic "magic" is used to produce it, can be called a numerical shadow of **P**. Later in this chapter a seemingly extremely naïve refinement procedure will be introduced, called *Stable Local Error Subtraction* or SLES, which is completely different from the GHYS procedure, whose speed is comparable to a highly optimized GHYS procedure, but whose reliability has not yet been rigorously tested.

1.1.1 The GHYS refinement algorithm

This section presents the GHYS refinement procedure for a two-dimensional Hamiltonian system. Assume we have a noisy orbit $\mathbf{P} = \{\mathbf{p}_i\}_{i=0}^{S}$, and we want to find a less noisy orbit $\{\tilde{\mathbf{p}}_i\}_{i=0}^{S}$. The 1-step errors are estimated by

$$\mathbf{e}_{i+1} = \mathbf{p}_{i+1} - \tilde{f}(\mathbf{p}_i)$$

where $\tilde{f}$ is an integrator with higher accuracy than the one used to compute **P**. The refined orbit will be constructed by setting

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + \mathbf{c}_i,$$

where $\mathbf{c}_i$ is a correction term for step i. Now,

$$\begin{aligned} \mathbf{c}_{i+1} &= \tilde{\mathbf{p}}_{i+1} - \mathbf{p}_{i+1} \\ &\approx \tilde{f}(\tilde{\mathbf{p}}_i) - (\mathbf{e}_{i+1} + \tilde{f}(\mathbf{p}_i)), \quad \text{if } \tilde{\mathbf{p}}_{i+1} \approx \tilde{f}(\tilde{\mathbf{p}}_i) \end{aligned}$$

Assuming the correction terms $\mathbf{c}_i$ to be small, then $\tilde{f}(\tilde{\mathbf{p}}_i)$ can be expanded in a Taylor series about $\tilde{f}(\mathbf{p}_i)$:

$$\begin{aligned} \tilde{f}(\tilde{\mathbf{p}}_i) &\approx \tilde{f}(\mathbf{p}_i) + \frac{\partial \tilde{f}(\mathbf{p}_i)}{\partial \mathbf{p}_i}\mathbf{c}_i \\ &= \tilde{f}(\mathbf{p}_i) + L_i \mathbf{c}_i, \end{aligned}$$

where $L_i = \frac{\partial \tilde{f}(\mathbf{p}_i)}{\partial \mathbf{p}_i}$ is the linearized map. For a discrete chaotic map, L_i is just the Jacobian of the map at step i. For a system of ODEs, L_i is the Jacobian of the integral of the ODE from step i to step $i+1$.[0] The final equation for the corrections is

$$\mathbf{c}_{i+1} = L_i \mathbf{c}_i - \mathbf{e}_{i+1}. \tag{1.1}$$

[0] In other words, let

$$\dot{\mathbf{p}} = \mathbf{h}(t, \mathbf{p}) \tag{1.0}$$

be the first-order ODE. (Often written as $y' = f(t, y)$ in ODE texts.) Note that $\mathbf{p}_{i+1} = f(\mathbf{p}_i)$ is the solution of equation 1.0 using $\mathbf{p}_i$ as the initial condition and integrating $\mathbf{h}$ to time $i+1$. Then $J = \frac{\partial \mathbf{h}}{\partial \mathbf{p}}$ is the Jacobian of equation 1.0. The Jacobian measures how $\dot{\mathbf{p}}$ changes if $\mathbf{p}$ is changed by a small amount. The *resolvent* $R(t_{i+1}, t_i)$ is the integral of $J(t)$ along the path $\mathbf{p}(t)$, and tells us how a small perturbation $\delta\mathbf{p}$ from $\mathbf{p}_i$ at time t_i gets mapped to a perturbation from $\mathbf{p}_{i+1}$ at timestep t_{i+1}. That is, $R(t_{i+1}, t_i)$ is the solution of the so-called *variational equation*

$$\frac{\partial R}{\partial t} = J(t)R(t, t_i), \quad R(t_i, t_i) = I,$$

where I is the identity matrix. (The reason the arguments to R seem reversed is for notational convenience: they satisfy the identity $R(t_2, t_0) = R(t_2, t_1)R(t_1, t_0)$, and so a perturbation $\delta\mathbf{p}$ at time t_0 gets mapped to a perturbation at time t_2 by the matrix-matrix and matrix-vector multiplication $R_2\delta\mathbf{p} = R_1 R_0 \delta\mathbf{p}$.) Finally, the linear map in the GHYS refinement procedure is $L_i = R(t_{i+1}, t_i)$ [13].

As will be seen later, it is the computation of the linear maps L_i, called *resolvents*, that takes most of the CPU time during a refinement, because the resolvent has $O(N^2)$ terms in it, and it needs to be computed to high accuracy. Presumably, if one is interested in studying simpler high-dimensional systems, a chaotic map would be a better choice than an ODE system, because no ODE integration is needed.

If the problem were not chaotic, the correction terms $\mathbf{c}_i$ could be computed directly from equation 1.1. But since L_i will amplify any errors in $\mathbf{c}_i$ that occur near the unstable direction, computing the $\mathbf{c}_i$'s by iterating equation 1.1 quickly produces nothing but noise. Therefore, GHYS suggest splitting the error and correction terms into components in the stable ($\mathbf{s}_i$) and unstable ($\mathbf{u}_i$) directions at each timestep:

$$\mathbf{e}_i = e_{u_i}\mathbf{u}_i + e_{s_i}\mathbf{s}_i \tag{1.2}$$

$$\mathbf{c}_i = c_{u_i}\mathbf{u}_i + c_{s_i}\mathbf{s}_i. \tag{1.3}$$

Computing the unstable unit direction vectors is currently done by initializing the unstable vector at time 0, $\mathbf{u}_0$, to an arbitrary unit vector and iterating the linearized map forward with

$$\bar{\mathbf{u}}_{i+1} = L_i\mathbf{u}_i, \;\; \mathbf{u}_{i+1} = \bar{\mathbf{u}}_{i+1}/|\bar{\mathbf{u}}_{i+1}|. \tag{1.4}$$

Since L_i magnifies any component that lies in the unstable direction, and assuming we are not so unlucky to choose a $\mathbf{u}_0$ that lies precisely along the stable direction, then after a few e-folding times $\mathbf{u}_i$ will point roughly in the actual unstable direction. Similarly, the stable unit direction vectors $\mathbf{s}_i$ are computed by initializing $\mathbf{s}_S$ to an arbitrary unit vector and iterating backwards,

$$\bar{\mathbf{s}}_i = L_i^{-1}\mathbf{s}_{i+1}, \;\; \mathbf{s}_i = \bar{\mathbf{s}}_i/|\bar{\mathbf{s}}_i|, \tag{1.5}$$

where L_i^{-1} can be computed either by inverting L_i, or by integrating the variational equations backwards from step $i+1$ to step i. The latter is far more expensive, but may be more reliable in rare instances. In any case, it is intended that $L_iL_i^{-1} = L_i^{-1}L_i = I$, the identity matrix. There may be more efficient ways to compute the stable and unstable vectors, possibly having to do with eigenvector decomposition of the Jacobian of the map (not the Jacobian of the *integral* of the map), but I have not looked into this.

Substituting equations 1.2 and 1.3 into equation 1.1 yields

$$c_{u_{i+1}}\mathbf{u}_{i+1} + c_{s_{i+1}}\mathbf{s}_{i+1} = L_i(c_{u_i}\mathbf{u}_i + c_{s_i}\mathbf{s}_i) - (e_{u_{i+1}}\mathbf{u}_{i+1} + e_{s_{i+1}}\mathbf{s}_{i+1}) \tag{1.6}$$

For the same reason that L_i magnifies errors in the unstable direction, it diminishes errors in the stable direction. Likewise, L_i^{-1} diminishes errors in the unstable direction and magnifies errors in the stable direction. Thus the c_u terms should be computed in the backwards direction, and c_s terms in the forward direction. Taking components of equation 1.6 in the unstable direction at step $i+1$ (recall that $L_i\mathbf{u}_i = \bar{\mathbf{u}}_{i+1}$ lies in the same direction as $\mathbf{u}_{i+1}$), iterate backwards on

$$c_{u_i} = (c_{u_{i+1}} + e_{u_{i+1}})/|\bar{\mathbf{u}}_{i+1}|, \tag{1.7}$$

and taking components in the stable direction, iterate forwards on

$$c_{s_{i+1}} = |L_i\mathbf{s}_i|c_{s_i} - e_{s_{i+1}}. \tag{1.8}$$

The initial choices for c_{s_0} and c_{u_S} are arbitrary as long as they are small — smaller than the maximum shadowing distance — because equation 1.8 damps initial conditions, and equation 1.7 damps final conditions. QT and GHYS choose them both as 0. This choice is probably as good as any, but it can be seen here that if one shadow exists, there are infinitely many of them. (Another justification of this is offered below.) Another way of looking at these initial choices for c_{s_0} and c_{u_S} is that they "pinch" the growing components at the final end point, and the backwards-growing components at the initial point, to be small, so that $(\tilde{\mathbf{p}}_0 - \mathbf{p}_0) \cdot \mathbf{s}_0 = 0$ and $(\tilde{\mathbf{p}}_S - \mathbf{p}_S) \cdot \mathbf{u}_S = 0$. That is, *boundary conditions* are being forced on the problem so that the exponential divergence is forcibly masked, if possible.

Note that these boundary conditions allow the initial conditions for the shadow and noisy orbits to differ along the unstable direction. In fact, this *must* occur if the change in initial conditions is to have any effect. That is, when looking for a shadow, if perturbations are only allowed in the stable direction, those perturbations would die out, leading the "shadow" to follow the true orbit that passes through our initial conditions — the one that is already known to diverge exponentially from the computed (noisy) orbit.

Generalizing to arbitrary Hamiltonian systems

This section is derived from QT's Appendix B, although it is presented slightly differently.

If the configuration space is D dimensional, then there are $2D$ dimensions in the phase space. It can be shown that in a Hamiltonian system, the number of stable and unstable directions is each equal to D. At timestep i, let $\{\mathbf{u}_i^j\}_{j=1}^D$ be the D unstable unit vectors, and let $\{\mathbf{s}_i^j\}_{j=1}^D$ be the D stable unit vectors. For any particular timestep, it will be convenient if the unstable vectors are orthogonal to each other, and the stable vectors are orthogonal to each other. However, the stable and unstable vectors together will not in general form an orthogonal system.

The vectors are evolved exactly as before, except using Gram-Schmidt orthonormalization to produce two sets of D-orthonormal vectors at each timestep. Since we do not know *a priori* what directions are stable and unstable at each timestep, we choose an arbitrary orthonormal basis $\mathbf{u}_0^j$ at time zero, and an arbitrary orthonormal basis $\mathbf{s}_S^j$ at time S, and evolve them as:

$$\bar{\mathbf{u}}_{i+1}^j = L_i \mathbf{u}_i^j, \quad \bar{\mathbf{s}}_i^j = L_i^{-1} \mathbf{s}_{i+1}^j.$$

Then, at each timestep i, do two Gram-Schmidt orthonormalizations: one on $\{\bar{\mathbf{u}}_{i+1}^j\}_{j=1}^D$ to produce $\{\mathbf{u}_{i+1}^j\}_{j=1}^D$, and another on $\{\bar{\mathbf{s}}_i^j\}_{j=1}^D$ to produce $\{\mathbf{s}_i^j\}_{j=1}^D$. After a few e-folding times, we find that $\mathbf{u}_i^1$ points in the most unstable direction at timestep i, $\mathbf{u}_i^2$ points in the second most unstable direction, *etc.*Likewise, $\mathbf{s}_i^1$ points in the most stable direction at time i, $\mathbf{s}_i^2$ points in the second most stable direction, *etc.*

The multidimensional generalizations of the error and correction vectors is the obvious

$$\mathbf{e}_i = \sum_{j=1}^{D} (e_{u_i^j} \mathbf{u}_i^j + e_{s_i^j} \mathbf{s}_i^j), \quad \mathbf{c}_i = \sum_{j=1}^{D} (c_{u_i^j} \mathbf{u}_i^j + c_{s_i^j} \mathbf{s}_i^j).$$

To convert the 1-step error at timestep i from phase-space co-ordinates $\mathbf{e}_i' = \{e_i^k\}_{k=1}^{2D}$ to the stable and unstable basis $\mathbf{e}_i = \left\{ \{e_{u_i^j}\}_{j=1}^D, \ \{e_{s_i^j}\}_{j=1}^D \right\}$, one constructs the matrix V_i whose columns are the unstable and stable unit vectors, $V_i = (\mathbf{u}_i^1 \mathbf{u}_i^2 \ldots \mathbf{u}_i^D \mathbf{s}_i^1 \mathbf{s}_i^2 \ldots \mathbf{s}_i^D)$, and solves the system $V_i \mathbf{e}_i = \mathbf{e}'_i$.

The equation for the correction co-efficients in the unstable subspace at timestep $i+1$ is

$$\sum_{j=1}^{D} (c_{u_{i+1}^j} + e_{u_{i+1}^j}) \mathbf{u}_{i+1}^j = \sum_{j=1}^{D} c_{u_i^j} L_i \mathbf{u}_i^j$$

which we project out along $\mathbf{u}_{i+1}^k$ producing

$$c_{u_{i+1}^k} + e_{u_{i+1}^k} = \sum_{j=k}^{D} c_{u_i^j} U_i^{kj}$$

where the scalar $U_i^{kj} = \mathbf{u}_{i+1}^k \cdot L_i \mathbf{u}_i^j = \mathbf{u}_{i+1}^k \cdot \bar{\mathbf{u}}_{i+1}^j$, and the Gram-Schmidt process ensures $U_i^{kj} = 0$ if $j < k$. The boundary condition at timestep S is $\forall j \, \{c_{u_S^j} = 0\}$, and we compute the co-efficients backwards using

$$c_{u_i^k} = \frac{1}{U_i^{kk}} \left(c_{u_{i+1}^k} + e_{u_{i+1}^k} - \sum_{j>k} c_{u_i^j} U_i^{kj} \right)$$

We first solve for $\{c_{u_i^D}\}_{i=0}^S$, which doesn't require knowledge of the other c_u co-efficients, then we solve for $\{c_{u_i^{D-1}}\}_{i=0}^S$, *etc.*

The equation for the correction co-efficients in the stable subspace at timestep i is

$$\sum_{j=1}^{D} (c_{s_{i+1}^j} + e_{s_{i+1}^j}) L_i^{-1} \mathbf{s}_{i+1}^j = \sum_{j=1}^{D} c_{s_i^j} \mathbf{s}_i^j$$

which we project out along $\mathbf{s}_i^k$ producing

$$c_{s_i^k} = \sum_{j=k}^{D} (c_{s_{i+1}^j} + e_{s_{i+1}^j}) S_i^{kj}$$

where $S_i^{kj} = \mathbf{s}_i^k \cdot L_i^{-1} \mathbf{s}_{i+1}^j = \mathbf{s}_i^k \cdot \bar{\mathbf{s}}_i^j$, and the Gram-Schmidt process ensures $S_i^{kj} = 0$ if $j < k$. The boundary condition at time 0 is $\forall j \, \{c_{s_0^j} = 0\}$, and we compute the co-efficients forwards using

$$c_{s_{i+1}^k} = \frac{1}{S_i^{kk}} \left(c_{s_i^k} - \sum_{j>k} (c_{s_{i+1}^j} + e_{s_{i+1}^j}) S_i^{kj} \right) - e_{s_{i+1}^k}$$

As with the unstable corrections, we first compute $\{c_{s_i^D}\}_{i=0}^S$ which does not require knowledge of the other c_s co-efficients, then we compute $\{c_{s_i^{D-1}}\}$, *etc.*

Discussion of the GHYS algorithm

There is no guarantee that refinement converges towards a true orbit; if there was, then all noisy orbits would be shadowable. In fact, even if some refinements are successful, numerical refinement alone does not prove rigorously that a *true* shadow exists; it only proves the existence of a numerical shadow, *i.e.,* a trajectory that has less noise than the original. Furthermore, the 1-step error $\mathbf{e}_{i+1}$ computed by any numerical technique measures the difference between the noisy and more accurate solutions at timestep $i+1$, where both start from the same position at timestep i. Even if this distance is about 10^{-15}, it does not imply that the difference between the noisy solution at timestep $i+1$ and the *true* solution passing through step i is equally small. This was dramatically demonstrated one day when I accidentally set the precision of the "accurate" integrator to only 10^{-8}, and the shadowing routine happily refined the orbit until the 1-step errors all had magnitudes less than 10^{-15}. Considering that the "accurate" trajectories were only being computed to a tolerance of 10^{-8}, it seems that refinement had converged to within 10^{-15} of a specific numerical orbit that had true 1-step errors of about 10^{-8}.

Furthermore, a numerical integration routine that is requested to integrate to a tolerance close to the machine precision might not achieve it, because it might undetectably lose a few digits near the machine precision. Thus, even when a numerical shadow is found with 1-step errors of 10^{-15}, the true 1-step errors are probably closer to 10^{-12}. GHYS provide a method called *containment* that can prove rigorously when a true shadow exists, but we have not implemented containment. As a surrogate to containment, QT did experiments on simple chaotic maps with 100-digit accuracy (using the *Maple* symbolic manipulation package) showing that if the GHYS refinement procedure refined the trajectory to 1-step errors of about 10^{-15}, then successful refinements could be continued down to 10^{-100}. It is reasonable to assume that refinement would continue to decrease the noise, converging in the limit to a noiseless (true) trajectory.

For the above reasons, we are confident that convergence to a numerical shadow implies, with high probability, the existence of a true shadow. However, to prove it rigorously requires implementing a scheme such as the GHYS containment procedure. This is one possible avenue for further research.

There is also no guarantee that, even if the refinement procedure *does* converge, that it converges to a reasonable shadow of $\mathbf{P}$; in principle it could converge to a true orbit that is far from $\mathbf{P}$, in which case the term "shadow" would be inappropriate. However, I have found in practice that refinement always fails due to the 1-step errors "exploding" (becoming large). I have never seen a case in which the refined orbit diverged from the original orbit while retaining small 1-step errors.

The error explosion occurs when 1-step errors are so large that the linearized map becomes invalid for computing corrections. Since the method is global (*i.e.,* each correction depends on all the others), inaccuracies in the computation of the corrections can quickly amplify the noise rather than decreasing it. Thus, within 1 or 2 refinement iterations, the 1-step errors can grow by many orders of magnitude, resulting in failed refinements. It is unclear if local methods like SLES (introduced below) will suffer the same consequences; probably they do, but errors probably grow much more slowly, and only locally in the orbit where 1-step errors are large.

Dependence of shadowing on the choice of accurate integrator

I have tried two integrators as my "accurate" integrator, although both were variable-order, variable-timestep Adams's methods called SDRIV2 and LSODE [18, 16]. QT used a Bulirsch-Stoer integrator by Press *et al.* [28]. It would be interesting to study whether the choice of accurate integrator influences the numerical shadow found, because if one true shadow exists, then infinitely many (closely packed) true shadows exist.[1] However, if the boundary conditions are the same, then the solutions should be the same.

Dependence of shadowing on choice of $\mathbf{u}_0$ and $\mathbf{s}_S$

Since $\mathbf{u}_0$ is arbitrary, and the displacement along the *actual* unstable subspace at time 0 that determines the evolution of the orbit, is it possible that we may fail to find a shadow because the initial perturbations allowed by our choice of $\mathbf{u}_0$ don't include the perturbations necessary to find a shadow?

I have not pursued this question at all, but it seems an interesting one. There are examples introduced below where the GHYS refinement procedure failed to find a shadow for a noisy orbit $\mathbf{Q}$, when it can be shown that one exists; the above scenario may be a cause. One way to test it is to try to find a shadow for $\mathbf{Q}$ using a different initial guess for $\mathbf{u}_0$, although I have not tried this. Similar comments apply to $\mathbf{s}_S$.

However, there are at least 3 arguments against this dependence being a problem. First, since the stable subspace has measure zero in the total space, it seems unlikely that an arbitrary choice for $\mathbf{u}_0$ could choose a subspace that doesn't contain "enough" of the unstable subspace. Secondly, the correction co-efficients for the unstable subspace are computed starting at the *end* of the trajectory, where $\mathbf{u}_S$ is presumably correct; similarly for the correction co-efficients in the stable subspace and $\mathbf{s}_0$. Thirdly, assuming the shadowing distance is much larger than the 1-step errors, it is likely that the intervals near the endpoints where the stable/unstable vectors are inaccurate are too small to allow instabilities in the computations of the corrections to build from 1-step error sizes to shadow-distance sizes. Further study of this possible problem may be helpful.

If it does turn out to be a problem, there are many possible fixes. First, we could attempt to shadow only between points that are known to have accurate approximations to the stable and unstable directions. One measure of this accuracy is to start with two *different* arbitrary unit vectors at time 0 and evolve them forward using equation 1.4. When they converge within some tolerance to the same vector at timestep a, we can assume they are correct after timestep a. The same could be done to the stable vectors, starting with two different guesses at time S and evolving backwards using equation 1.5 until they converged at timestep b. Then, shadowing would be attempted only between a and b.

If we want to shadow all the way to the endpoints, we could attempt to get reliable stable/unstable vectors everywhere in the following manner: compute the times a and b as above. Then compute $\{\mathbf{u}_i\}_{i=a}^{0}$ *backwards*, attempting to evolve the correct $\mathbf{u}_a$ backwards, while hoping that instabilities from the stable subspace don't overwhelm the computation for the few steps that

[1] "Proof": For any system, *even a chaotic one*, given any true orbit of fixed length in time, a small enough perturbation in the initial conditions in any direction produces a small change in the final conditions, although this perturbation must be exponentially smaller for chaotic systems than for non-chaotic systems. (If the perturbation is restricted to the stable subspace, then obviously a similar solution will be obtained.) Thus given any true orbit that δ-shadows a noisy orbit, we can find infinitely many other true orbits nearby that also δ-shadow the noisy orbit. However, it may be that all the true orbits are packed into a space unresolved by the machine precision.

iteration proceeds backwards. Do the same for $\mathbf{s}_b$, except evolving forward. This seems to me the most promising method, although I have not yet tested it. To test how much of the unstable subspace has encroached on $\mathbf{s}_{b+j}$, perhaps we can dot $\bar{\mathbf{s}}_{b+j}$ with $\mathbf{u}_{b+j}$ to ensure it stays small.

Another (untested) method to compute the stable and unstable vectors out to the endpoints is: compute $\{\mathbf{u}_i\}_{i=0}^{S}$ in the normal way. Then choose $\mathbf{s}_S$ to be orthogonal to $\mathbf{u}_S$. Even though these vectors are not generally orthogonal, choosing $\mathbf{s}_S$ to be orthogonal to $\mathbf{u}_S$ seems better than a random choice. Then, compute $\{\mathbf{s}_i\}_{i=S-1}^{0}$ normally. Then recompute $\{\mathbf{u}_i\}_{i=0}^{S}$ using a $\mathbf{u}_0$ orthogonal to $\mathbf{s}_0$.

If a method can be invented that computes the stable and unstable vectors using eigenvector decomposition, then it probably would be better than all the above methods, because it would not require any resolvents and not be restricted to being away from the endpoints.

1.2 A new shadowing procedure: SLES

1.2.0 Neural network backpropagation and error subtraction

Although the following refinement algorithm has absolutely nothing to do with neural networks, the idea for it was inspired by how the neural net backpropagation learning algorithm works. In backpropagation, a function $\mathbf{E}(\mathbf{w}, a)$ measures the error in the output of a neural network, given a certain input a. The neural network "learns" by adjusting its internal variables $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ to reduce its output error; learning stops when the output error is sufficiently small. The way it reduces the error is by taking partial derivatives of the error function with respect to all the internal variables w_i, producing the gradient of the error with respect to $\mathbf{w}$. Moving $\mathbf{w}$ a small amount in the direction opposite the gradient should then reduce the error. The amount by which $\mathbf{w}$ is moved along the negative gradient is called the *learning rate*. In one sentence, backpropagation says, "Find a direction to move that reduces the error, and then move a little bit in that direction."

The inspiration for shadowing is this: once we have computed the 1-step error vectors $\mathbf{e}_i$ along the noisy trajectory $\mathbf{p}_i$, it seems reasonable that the 1-step error at step i would be reduced if we simply *subtracted* a fraction ε of $\mathbf{e}_i$ from $\mathbf{p}_i$, for some $\varepsilon \in (0, 1)$. This is the basis for what I call a *Local Error Subtraction* method.

1.2.1 SLES: Stable Local Error Subtraction

A naïve implementation of the above idea may be simply to set $\tilde{\mathbf{p}}_i = \mathbf{p}_i - \varepsilon\mathbf{e}_i$ for i from 1 to S. However, this does not define how to change the initial point $\mathbf{p}_0$, because there is no $\mathbf{e}_0$. Since any new true trajectory *must* have a different initial condition, we must find a way to correct $\mathbf{p}_0$.

To allow corrections to $\mathbf{p}_0$, I introduce the *backward error*,

$$\mathbf{b}_i = \mathbf{p}_i - \tilde{\mathbf{f}}_{-}(\mathbf{p}_{i+1}),$$

where $\tilde{\mathbf{f}}_{-}(\mathbf{p}_{i+1})$ integrates the solution backwards from time $i+1$ to time i, and $\mathbf{b}_i$ is defined on $0 \leq i < S$, where S is the number of shadow steps. Thus each timestep from 1 to $S-1$ has both a forward error $\mathbf{e}_i$ and backward error $\mathbf{b}_i$; timestep 0 has only a backward error $\mathbf{b}_0$, and timestep S has only a forward error $\mathbf{e}_S$.

Subtracting some linear combination $(\varepsilon \mathbf{e}_i + \beta \mathbf{b}_i)$ from $\mathbf{p}_i$ (setting $\mathbf{e}_0 = \mathbf{b}_S = \mathbf{0}$) does not work for any reasonable values of ε, β that I have tried (between 0.1 and 0.9). I believe this is because 1-step errors computed in the forward direction will be slightly unstable in their components that lie in the unstable subspace, and backwards 1-step errors will be unstable in their components that lie in the stable subspace. The method works, however, if we rewrite $\mathbf{e}_i$ using equation 1.2 (which was $\mathbf{e}_i = e_{u_i}\mathbf{u}_i + e_{s_i}\mathbf{s}_i$), and $\mathbf{b}_i = b_{u_i}\mathbf{u}_i + b_{s_i}\mathbf{s}_i$, and update $\mathbf{p}_i$ as

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i - (\varepsilon e_{s_i}\mathbf{s}_i + \beta b_{u_i}\mathbf{u}_i). \tag{1.9}$$

In other words, we use only the components of the 1-step error that are computationally stable.

The SLES[2] method seems to work about as fast as the optimized GHYS method introduced in the next chapter. For example, using $\varepsilon = \beta = 0.9$, 1-step errors are generally reduced by about a factor of 10 per refinement iteration, as one would expect when subtracting away 90% of the error. However, I have not extensively tested it, and since it seems to have a weaker theoretical basis than the GHYS method, I have not used it in any of the results mentioned in later chapters. However, I have no reason to believe that the method is unreliable, so further testing in the future seems warranted.

Finally, note that SLES makes no explicit use of the linear map L_i. Currently, the only use of the resolvents in SLES is implicitly through the construction of the stable and unstable directions. If a method of computing the stable and unstable directions can be found that doesn't use the linear maps (*eg.,* eigenvector decomposition as mentioned above), then the SLES method will not use the linear maps at all, and they will not need to be computed. Since computing the linear maps (for an ODE system) is currently by far the most computationally intensive section of the program, getting rid of them entirely is an exciting prospect.

Local *vs.* global methods

It is perhaps important to note that there is a fundamental difference between the GHYS refinement procedure, which I call a *global* method, and SLES, which I call a *local* method. Let the initial noisy orbit be $\mathbf{P}^0 = \{\mathbf{p}_i^0\}_{i=0}^{S}$, with the superscript referring to the refinement iteration. Assuming the GHYS refinement procedure can be written as a Newton's method, it can be summarized as

$$\mathbf{P}^{j+1} = \mathbf{P}^j - [g'(\mathbf{P}^j)]^{-1} g(\mathbf{P}^j) \tag{1.10}$$

where g was defined near the beginning of this chapter as the function computing the forward 1-step errors on the entire orbit,[3] and g' is the Jacobian of g. Thus the entire orbit is updated using a

[2] SLES is pronounced "sleeze", because this seems like such a sleezy, naïve method.

[3] This formulation is not complete, because we do not yet know how to include the boundary conditions in g. See section 4.1.0 for more details.

single (large) equation. SLES, on the other hand, applies the local correction formula of equation 1.9 to each point independently; namely

$$\left\{ \mathbf{p}_i^{j+1} = \mathbf{p}_i^j - (\varepsilon e_{s_i}^j \mathbf{s}_i + \beta b_{u_i}^j \mathbf{u}_i) \right\}_{i=0}^{S}.$$

One consequence of this difference occurs when the 1-step errors in the GHYS method "explode" and cause the iteration to fail. The explosion seems to occur much faster using GHYS than using SLES. I believe this is because in GHYS, all the 1-step errors along the entire trajectory contribute to each other's computation of the correction, so that if a single 1-step error is too big for its linear map to be valid, then the corrections for *all* the points will be invalid. However, in SLES, there is no such dependence between points at arbitrary distances along the trajectory, so a point with large 1-step error can propagate its instability to neighbors j shadow steps away only after j refinement iterations.

Currently, the only known way to isolate glitches using the GHYS method is the way QT did it, namely to try shadowing for longer and longer times until the orbit explodes, and then to do a binary search for shorter and longer shadows until the shadow step at which the glitch occurs is pinpointed. This is alot of work. Because of the slow way glitches propagate their effects in SLES, it may be possible to isolate glitches simply by their large errors and that of their neighbors. It may even be possible to pinpoint multiple glitches along a single trajectory. For example, points G_1 and G_2 would be glitches if we could shadow the trajectory from 0 to $G_1 - 1$ and from $G_1 + 1$ to $G_2 - 1$ and from $G_2 + 1$ to S, but not shadow any part of the trajectory containing G_1 or G_2. This introduces the concepts of *shadowable segments*, although it is not clear what use such a concept is.

1.3 A wider perspective: Why shadowing?

With an introductory level of understanding of shadowing behind us, it may be instructive to devote some time to "devil's advocate" questions and answers.

0. **Q**: Science is usually interested in generally applicable results. Why should we care about finding an exact solution to a specific initial condition?

   A: We don't. What shadowing tries to do, empirically for now, is show that given *any* particular initial condition and cheaply integrated solution, that a solution of a more expensive integration exists nearby. Ultimately, using GHYS's containment algorithm, we aim to show that any particular noisy solution has a true solution nearby. If we cannot find such solutions, then we may have good reason to suspect the reliability of our simulations.

1. **Q**: OK, but if the *system* is chaotic, then it is *supposed* to have sensitive dependence on initial conditions. Shouldn't it be fine that there is noise in our orbits? *i.e.,* isn't our theory robust enough to withstand small perturbations? Isn't computational noise roughly equivalent to chaos anyway?

   A: We (or at least I) do not know enough about chaotic systems to say, so for now we should play it safe. In fact, one way of looking at shadowing is that we are actually trying to *discover* if the theory is robust enough!

2. **Q**: There is a large body of theory behind numerical integration of ODE's. In particular, the theory of defect-based backwards error analysis (DBBEA) shows that, for any particular ODE $y' = f(t, y)$, a numerical method will give the *exact* solution to a slightly perturbed problem $\tilde{y}' = f(t, \tilde{y}) + \delta(t)$, where $\delta(t)$ is a function whose magnitude is small, roughly the size of the tolerance requested of the numerical ODE method. Furthermore, there exists a theorem of ODEs that states equivalence between the problem of how solutions change with changing initial conditions, and how they change with changing parameters. Isn't this good enough when we realize that $\delta(t)$ can be considered a parameter with $\delta(t) \equiv 0$ giving the true solution?

   A: Again, we don't know enough about chaotic systems to say. If I want to integrate the gravitational N-body problem, DBBEA tells me that I will get the exact solution to the problem

$$\mathbf{F} = \frac{G m_1 m_2}{r^2} + \delta(t). \tag{1.11}$$

   But this is *not* the N-body problem. Generally in science, one tries to change as few things as possible between the real world and the model of the world. It seems to me that asking for *an exact solution to the* **same** *ODE with sligthly different initial conditions* is changing less than asking for *an exact solution to a slightly different ODE.* It is certainly not clear that the latter is good enough, *especially* considering that we are dealing specifically with problems that we *know* have sensitive dependence on small changes. Furthermore, Hamiltonian systems have many special properties that will be preserved if we change the initial conditions, but may not be preserved by equation 1.11.

3. **Q** : What is the ultimate goal of shadowing?

   A: That's a tough question. For one, we are answering question 0 above. A possible ultimate goal of shadowing could be to identify, learn about, recognize, and finally eliminate glitches from our simulations, but it is unclear if all of these are feasible, especially the last. I suspect the GHYS containment procedure may allow us to eliminate glitches when we detect them, but this line of reasoning requires more thought.

Chapter 2

High Dimensional Shadowing & Optimizations to the GHYS Refinement Algorithm

"What shadows we are, and what shadows we pursue."
— Edmund Burke, *Speech.* (1780)

The main thrust of this thesis is to extend a previously published numerical shadowing refinement procedure to make it more efficient, thus allowing larger and more realistic systems to be shadowed. The astrophysical N-body problem is used as an example, although the refinement procedure could just as easily be used on any chaotic system. With various numerical tricks and physical insights, our algorithm runs, depending on the problem, between 5 and 100 times faster than the original algorithm.

2.0 Introduction

2.0.0 Background

As described in the previous chapter, Grebogi, Hammel, Yorke, and Sauer (GHYS) [11], invented a two-dimensional shadowing procedure consisting of two parts:

0. *containment*, which, if successful, rigourously proves the existence of a true shadow of a noisy trajectory. It is not, however, guaranteed to find a shadow if one exists. Thus it cannot be used to *dis*prove the existence of a shadow.
1. *refinement*, which GHYS intended simply as a method to reduce the noise of a pseudo-trajectory to increase the chances of success of their containment procedure.

Quinlan and Tremaine (QT) [29], wanted to test for the existence of shadows in large N-body systems. They generalized the refinement portion of the GHYS algorithm to work on arbitrary-dimensional Hamiltonian systems. QT showed that, using 100 decimal digits of precision, if the noise level of a four-dimensional orbit could be reduced to about 10^{-15}, then the noise could also be reduced all the way down to 10^{-100}, with the number of correct digits in the solution increasing geometrically. They conclude that it is reasonable to assume that further refinement would continue to reduce the noise indefinitely, leading in the limit to a true orbit. This seems less rigourous that containment, but in practice it is probably almost as reliable.

QT then applied their refinement procedure to a simplified N-body system in which a single particle moves, unsoftened, among 100 fixed particles that are distributed in a standard Plummer distribution. For their "cheap" integrator, they used a 5th-order predictor-corrector routine that is commonly used by astronomers [Appendix B of [6]], arranged to have 1-step errors of about 10^{-5}. For their expensive integrator, they used a Bulirsch-Stoer method [28] with tolerance 10^{-13}. They found they could shadow the particle for several crossing times. Unfortunately, their refinement procedure was too inefficient to work on significantly higher dimensional systems. They tried two and three moving particles, but found they were pushing the bounds of feasibility that their shadowing procedure could handle.

It is perhaps important to note before continuing that it is not clear that refinement alone, even if optimized as introduced in this chapter, and even if it is as reliable as containment, is more efficient than using containment. But if refinement is to be used at all, it needs to be made more efficient than the direct, naïve method of GHYS and QT.

2.0.1 Why High Dimensional Shadowing?

It is not at all clear that being able to shadow one particle moving "pinball" fashion amongst 100 fixed ones implies that shadows of large N-body systems exist. To establish the shadowability of large N-body simulations requires shadowing tests to be applied to more realistic systems. At present it is far beyond feasibility to attempt shadowing systems of 10^5 or 10^6 particles that are commonly being used today by astronomers. Nevertheless, we can learn much about how shadowing results change as the dimension increases, using systems with just tens of moving particles. In such systems it may be reasonable to eliminate the fixed particles entirely, but to extend QT's results, most of the experiments performed in this thesis (and reported in the next chapter) were done on a system with a total of 100 particles, M of which move, for M ranging from 1 to 25.

It is important to emphasize the main scientific question that we will be studying once we have a reasonably fast shadowing implementation. The question is not only a question concerning N-body systems, but a general question for chaotic systems: *how do shadowing results change as the number of dimensions increases?*

2.0.2 Notation

For the remainder of this thesis, the following definitions will be in effect:

N is the total number of particles in the N-body system.

M is the number of particles which move. Thus, $N - M$ of them are fixed. The phase-space has $6M$ dimensions.

S is the number of shadow steps in the orbit. Thus, there are $S + 1$ points along the orbit, from $\mathbf{p}_0$ (the initial condition) to $\mathbf{p}_S$.

See the Appendix for a quick review of the N-body problem and its equations.

2.0.3 Asymptotic run times

As has already been mentioned, most of the expense of refinement of an N-body orbit is in computing the linear maps L_i. See the long footnote on page 9 for an explanation of L_i, and the Appendix for the derivation of the Jacobian of an N-body system. Manually counting floating-point operations in the Jacobian gives an operation count of roughly $60MN + 50M^2$. The total operation count to compute the right hand side of the variational equation, including the matrix-matrix multiply and the computation of the Jacobian, is about $108M^3 + 68M^2 + 85MN$. The M^2 term is probably negligible, and the MN term is only relevant if $N >> M$. Thus, if there are S shadow steps, and the function computing the right hand side is invoked an average of k times per shadow step, the total cost of computing all the resolvents is about $k \times S \times (108M^3 + 85MN)$. I have found empirically that, using an Adams's integrator, k seems to be independent of M, and roughly equal to about 500 for the shadow steps I use in my N-body simulations. Figure 2.0 shows some experiments comparing the time to compute one resolvent to M, demonstrating the $O(M^3)$ relationship.

2.1 The Optimizations

2.1.0 Brief Overview

There are 6 major optimizations that I have applied to the GHYS/QT refinement procedure. The first two are trivial, and no further details will be given about them. The final four will be explained in more detail in the remaining sections of this chapter.

Definition If program A has a *speedup* of α over program B, then the execution time of A is a fraction $1/\alpha$ of B's time; in other words A runs α times faster than B.

The $3M$ oversight. QT re-computed a new resolvent for each stable and unstable vector, *i.e.,* at each timestep they computed $6M$ resolvents. However, a resolvent $R(t_1, t_0)$ is a matrix operator that applies to *all* small perturbations of the orbit from time t_0 to t_1. Thus only 2 resolvents per timestep need be computed: one for evolving perturbations forward in time, and another to evolve

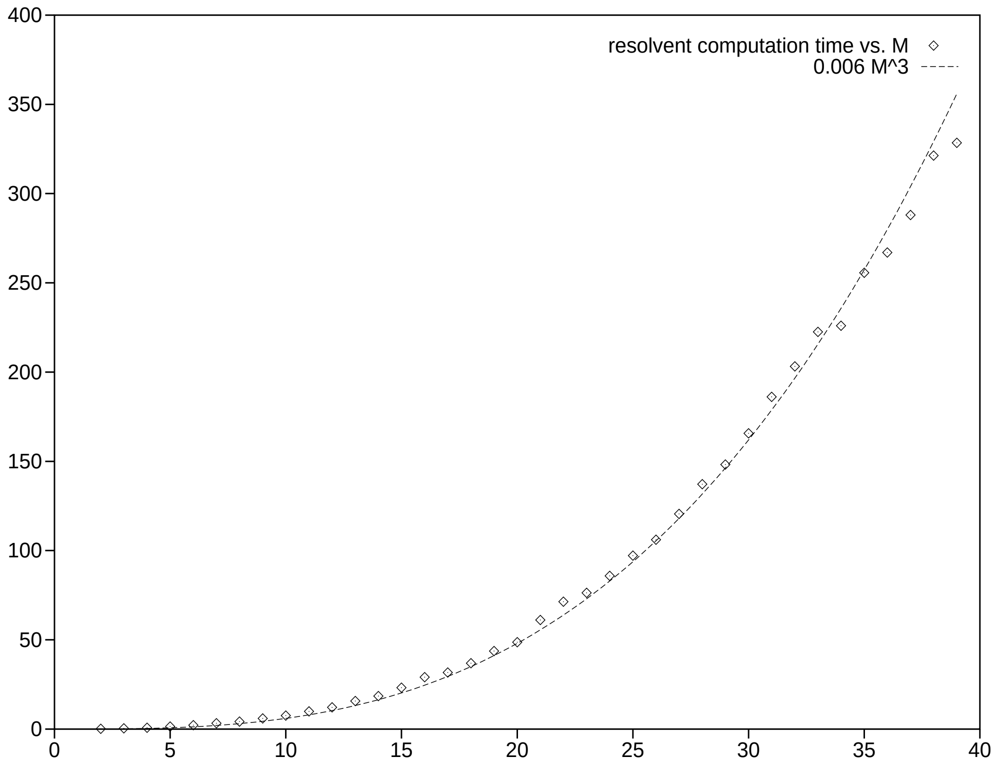

Figure 2.0: Time in seconds to compute one resolvent as a function of the number of moving particles M. For comparison, $0.006M^3$ is also plotted.

them backwards.[0] Since most of the time is spent computing the resolvents, fixing this oversight provides an immediate speedup of a factor of about $3M$.

Compute backwards resolvent by inverting forward resolvent. The above optimization can be extended even further by noting that the resolvent matrix L_i that maps perturbations from timestep i to $i+1$ is precisely the inverse matrix of the resolvent that maps small perturbations from timestep $i+1$ to i. Thus, rather than doing an integration from timestep $i+1$ backwards to time i to compute L_i^{-1}, L_i can be inverted using matrix operations, or the system $L_i\mathbf{x} = \mathbf{b}$ can be solved, whenever necessary, using standard elimination schemes. This gives an additional speedup of about 2, since only one resolvent per timestep need explicitly be computed using integration of the variational equations.

Large shadow steps. Define a *shadow step* as a time interval across which the 1-step error and correction co-efficient are computed. QT used every internal timestep of their integrator's noisy orbit as a shadow step, but this is not necessary. It is reasonable to skip timesteps in the original orbit to build larger shadow steps. This means that fewer resolvents and stable/unstable directions need to be computed and stored.

[0]This oversight is not obvious from the description in their appendix B, it appears only in their code.

Cheaper "accurate" integrator. When computing the resolvent and 1-step errors during early refinements, it is not necessary to compute them to extremely high accuracy. Since the initial 1-step errors may have magnitudes of about 10^{-4}, the resolvents and 1-step errors need only be computed to a tolerance of, say, 10^{-7}, *i.e.,* 3 extra digits. QT always computed the 1-step errors and resolvents to a tolerance of 10^{-13}.

Constant resolvent/unstable/stable directions. If a shadow exists, then by definition it cannot be too far away from the original noisy orbit. It is reasonable to assume that the resolvents, unstable and stable directions (abbreviated RUS) will change little as the orbit is perturbed towards the shadow. Thus, it may not be necessary to recompute the RUS for every refinement iteration. This is probably the biggest savings, because in combination with the cheaper integrator, it means that the RUS usually need only be computed once to a tolerance of about 10^{-7}, and this RUS will suffice for all future refinements. This is analogous to a simplified Newton's method that does not recompute the Jacobian at ever iteration.

Re-use RUS from a previous successful shadow. Recall that to find the longest possible shadow of a noisy orbit, we attempt to shadow for longer and longer times until shadowing fails. Assume shadowing for S shadow steps produces a shadow A. Then attempt shadowing for $S+k$ timesteps for some integer k, and assume a successful shadow B will be found. Since A and the first S steps of B both shadow the same noisy orbit, they must be close to one another. By the same argument as the previous paragraph, the RUS that was computed for A can probably be re-used for the first S shadow steps of B. Thus only k new RUS_i's need be computed.

2.1.1 Constant resolvent, unstable and stable directions

I use the acronym *RUS* to mean the set of all *R*esolvents, *U*nstable, and *S*table vectors for the entire trajectory. The RUS at a particular timestep i is referred to as RUS_i.

For a smooth, continuous, well-behaved function, the Jacobian (*i.e.,* the derivative) is also smooth and continuous. Most Hamiltonian systems (that I have seen) studied by scientists are at least piecewise of this sort. The N-body problem, in particular, is of this sort, as long as no particles pass within 0 distance of each other.[1] This implies that the resolvent along a particular path will not change much if the path is perturbed slightly. Since the unstable and stable unit vectors are derived solely from the resolvents, they will also change only slightly when the path is perturbed. Thus, the RUS computed on the first refinement iteration should be re-usable on later iterations, and computing the new 1-step errors is the only significant work to be performed on each new refinement iteration. (Computing the corrections from the errors is much less CPU intensive than computing the errors themselves.)

I have empirically found that computing the RUS to a tolerance of between 10^{-6} and 10^{-9} usually suffices to refine the orbit down to 1-step errors near the machine precision. If refinement fails using constant RUS, then perhaps there is a spot in the orbit where some RUS_i's change

[1] If the force is "softened", then this problem goes away.

quickly with small perturbations in the path. In that case, the RUS can be re-computed for a few refinements, after which constant RUS may be tried again. However, a case in which constant RUS fails often suggests that there as a mild trouble spot somewhere on the orbit that may cause a glitch in longer shadows.

Formulae for switching between constant and recomputed RUS

The algorithm used to switch between constant RUS and recomputed RUS uses a running-average "memory" scheme to keep track of the progress of the magnitude of the maximum 1-step error over the previous few refinements. Originally, I used a simple algorithm (like QT's) that would signal failure when a certain number K of successive refinements had failed, for $\mu = 0.9$ and $K = 3$. (See page 7 for the definition of a successful refinement.) However, refinement would sometimes get into a loop in which there would be a few successful refinements, followed by one or more that would "undo" the improvements of the previous ones. For example, the largest 1-step error may cycle as $\{8, 4, 2, 1, 8, 4, 2, 1, 8 \ldots\} \times 10^{-13}$. In general, cases were seen in which the number of refinements in these loops was 3, 4, and up to about 8. Clearly a simple "die when K successive failures are seen" is not general enough to catch this kind of loop.

Queuing theory provides a formula that is useful here. It defines an *arithmetic running average* A to be

$$A_{j+1} = \alpha A_j + (1 - \alpha)\nu \tag{2.0}$$

where ν is the newest element added to the set, and α is the *memory* constant. The higher the memory constant, the longer the memory — *i.e.,* the less the effect of an individual new element. A rule of thumb is that A is roughly an average of the most recent $2/(1 - \alpha)$ elements. Equation 2.0, however, is not suited for measuring errors such as those in refinement that can change by many orders of magnitude, and is especially unsuited when smaller means better. For example, a string of errors of 10^{-4} followed by an error of 10^{-7} would average to about 10^{-4}, whereas a change from 10^{-4} to 10^{-7} is an indication that refinement is succeeding. We thus need a *geometric* equivalent of equation 2.0, to allow values differing by orders of magnitude to average meaningfully.[2] I define the *geometric running average* G as

$$G_{j+1} = \sqrt[n+1]{G_j^n \nu} \tag{2.1}$$

where ν is the new element, and n is an integer, $n \propto \frac{1}{1-\alpha}$, so higher n implies longer memory. Both equations 2.0 and 2.1 require initialization to some reasonable starting value A_0 and G_0, respectively. n should not be too large; I found 2 or 3 worked best.

Finally, we want an equation that measures the *improvement* that is being made by successive refinements, rather than one that measures absolute error, because it is the *change* in 1-step errors that indicates whether current refinements are succeeding, not their absolute size. Note that the 1-step errors may stop decreasing for two reasons: (1) 1-step errors have reached the machine precision, or (2) refinement is failing to find a shadow. Using the improvement rather than the absolute value allows us to use the same algorithm to halt refinement in either case. Thus define

[2] It is trivial to show that the geometric mean of a set of numbers is the arithmetic mean of their logarithms, to a suitable base.

the improvement I to be

$$I = \left| \frac{\text{the maximum 1-step error of this refinement}}{\text{maximum 1-step error of previous refinement}} \right| .$$

If I is close to or greater than 1, then the current refinement did not improve on the previous one; if $I << 1$ then the current refinement is successful.

Heuristic for switching between constant and recomputed RUS

We have two refinement methods. In order of decreasing cost, they are recomputed RUS, and constant RUS. The general idea is: if refinement is working well with an expensive method, then switch to a cheaper one; if a cheaper one seems to be failing, switch to a more expensive one. If the most expensive one is failing, then give up refinement entirely, and return failure. Here is the heuristic I have built over many months of trial and error.

- When using recomputed RUS, it is safe to switch to constant RUS when the geometric running average improvement becomes < 0.1. At this time the current orbit and all its statistics must be saved in case constant RUS refinement fails.
- When using constant RUS, it is necessary to switch back to recomputed RUS when the geometric running average improvement stays > 0.5 for 3 successive refinements. At this time I discard all progress made by constant RUS and revert to the previous orbit that used recomputed RUS. I think it is reasonable to discard all progress made by constant RUS, even if significant progress was made, for the following reasons:
  - A refinement that computes the RUS is asymptotically more expensive than one that does not, so discarding all progress made by constant RUS refinements makes little difference, percentage-wise, in the final run-time.
  - Future constant RUS refinements, that will be performed after the RUS is recomputed, will converge faster than the current ones that just failed, because the RUS will be more accurate.
  - Recomputing the RUS with small 1-step errors would mean (using the simple "3 extra digits of accuracy" heuristic) recomputing it at a much higher accuracy, at much more expense, than is necessary.

  This argument is not always applicable; I have seen cases in which discarding constant RUS progress appeared to waste good progress. Perhaps there exists a better heuristic.
- Finally, after having switched back to recomputed RUS, it is time to exit if the geometric running average improvement stays > 0.1 for 3 successive refinements; and it is safe to switch again to constant RUS when the geometric running average improvement becomes less than 0.1. I have found that there are usually no half-measures when using recomputed RUS — refinement either succeeds geometrically or fails miserably.

Further research in recomputed RUS could focus on whether there exists a criterion to decide that a particular RUS_i needs recomputing, rather than all of them. If this is the case, it could be an $O(S)$ speed savings. Also, there may be a place for SLES or other noise-reduction procedures in the loop of expensive and cheap refinement algorithms (which currently contains only the two above: recomputed RUS and constant RUS).

Finally, I note in passing that Newton's method is sometimes used with dampening when it is known that the correction factors may not be accurate, but are assumed to point roughly in the correct direction. In other words, equation 1.10 (page 16) is modified to

$$\mathbf{P}^{j+1} = \mathbf{P}^j - \theta[g'(\mathbf{P}^j)]^{-1} g(\mathbf{P}^j)$$

for some constant $\theta \in (0,1)$. I briefly tried this with the correction co-efficients c_u, c_s, but it did not seem to work. However, this result should not be taken as conclusive, as I did not spend much time testing damped correction or trying to make it work.

Re-use RUS from a previous successful shadow

This idea is based on the same observation as constant RUS, and is only useful if constant RUS is employed. It takes advantage of the RUS for a noisy orbit B being near to the RUS for a nearby noisy orbit A. In particular, the shadows for two overlapping segments of a noisy orbit will be near to each other along the segment that the noisy segments overlap, and thus the RUS's for the overlapping segment will also be near each other. So, when trying to find a shadow for a segment B which is an extension to segment A, the RUS of A can be re-used on the segment of B that overlaps A.

One interesting question is whether to use the first or last RUS that was computed for A, if A ever had to recompute the RUS. An argument in favour of using the first is that the noisy orbit is exactly the same, since B is just an extension of A. However, if A had to recompute the RUS, then probably so will B if A's first RUS is used. Recall that the RUS needed for the early refinements need not be as accurate as the ones used later. Furthermore, we assume that the segment of B's shadow that overlaps A's shadow will be *closer* to A's shadow than the first few iterations of B is to the first few iterations of A. Thus, if we use the last RUS that A used, we are less likely to recompute the RUS for B. If B is a *subset* of A rather than an extension, then obviously no new RUS_i's need be computed. All shadowing runs in this thesis use the last RUS.

Finally, re-using the RUS allows a more fine-tuned search for where a glitch occurs, because it is much less expensive to attempt shadowing particular subsets of a noisy orbit once the entire RUS is computed.

2.1.2 Large Shadow Steps

Rationale

A numerical solution of an ODE is represented by an ordered, discrete sequence of points. Say we were to construct a continuous solution $p(t)$ from these points, for example by using a suitably smooth and accurate interpolation. Then we could extend the definition of δ_x shadowing for some true solution $x(t)$ by saying: $x(t)$ δ_x-shadows $p(t)$ on $t_0 \le t \le t_1$ if $\forall t \in [t_0, t_1]$, $|x(t) - p(t)| < \delta_x$.

Now, it should be clear that we can choose *any* representative sequence of points along $p(t)$ to use in the refinement algorithm introduced in the previous chapter; we need not choose the points from which $p(t)$ was originally constructed. In particular, we can choose a subset of the original set of points, as long as enough points are chosen to be "representative" of $p(t)$. The steps that are finally chosen are called *shadow steps*.

There are at least two reasons to desire as few shadow steps as possible: First, if constant RUS is being used, then the RUS needs to be stored in memory. Each RUS_i requires a matrix (the resolvent), and two sets of basis vectors. Clearly the fewer shadow steps, the less memory is required to store the RUS.[3] Second, if the "accurate" integrator has a large startup cost, then the fewer startups that are required, the better. For example, the Adams's methods I used take extremely small internal timesteps early in the integration. Thus, for each shadow step, the Adams's method must restart with extremely small stepsizes.

Estimating the largest shadow steps that can be used

For the N-body simulations reported in this thesis, a set of "standardized" units was used [14] such that the scale of the system in all units of interest was of order 1 — *i.e.,* the system had a diameter of order 1, the crossing time was of order 1, and the N particles each had mass $1/N$. In such a system, the timesteps of QT's integrator averaged about 0.01, and they used each timestep as a shadow step. I have found empirically that, in this system, using shadow steps of size 0.1 works well. Smaller shadow steps use more time and memory but could not find shadows any better. Shadow steps of 0.2 were slightly less reliable, and steps of size 0.5 were unreliable.

The important criterion for choosing shadow step sizes seems to be the validity of the linear map. Recall that the linear map or *resolvent* $R(t_1, t_0)$ is a first-order approximation that maps small perturbations of the orbit $p(t)$ from time t_0 to t_1, so that a system starting at $p(t_0) + e_0$ evolves, to first order in e_0, to $p(t_1) + R(t_1, t_0)e_0$. For this equation to be valid, e_0 must be small.[4] Since the computation of the correction co-efficients in the GHYS refinement procedure depends on this linear perturbation approximation, the 1-step errors of our shadow steps must be sufficiently small for the linear map to be valid. The larger the shadow steps, the further the accurate solution will diverge from the noisy solution, and thus the larger the correction needed per shadow step. Although the allowable perturbation size is problem-dependent, I have found empirically that, for the unit-sized N-body system used here, perturbations of size 10^{-5} are always small enough, while perturbations of size about 10^{-3} or greater start to show significant difficulties. In general, however, a shadow step must be small enough that the 1-step errors are small enough that the linearized map is accurate enough that, when the correction is applied at step i, the linearized map correctly maps the error towards being smaller. In other words, if e_i is the 1-step error at p_i, c_i is the correction,

[3] For example, with $M = 25$ moving particles, a RUS of length $S = 128$ requires about 100 megabytes of memory, and this space scales as $O(SM^2)$. ((25 particles $\times$ 6 dimensions per particle)^2 $\times$ 8 bytes per double precision number $\times$ 4 (2 resolvents, 2 sets of basis vectors each covering half the space) $\times$ 128 steps).

[4] This neglects to mention the error of the resolvent itself, which depends on $|t_1 - t_0|$. However, the resolvent error doesn't seem to matter much in comparison to the error introduced by the size of the perturbation e_0.

$L_i = R(t_{i+1}, t_i)$ is the linear map and ϕ is the exact perturbation map, then $L_i(p_i + c_i)$ must be close to $\phi(p_i + c_i)$ in comparison to the size of the 1-step errors[5]:

$$|L_i(p_i + c_i) - \phi(p_i + c_i)| << \max_i |e_i| \tag{2.2}$$

In practice, it is probably too difficult to use equation 2.2, and it may be necessary to experiment to find the largest allowable 1-step error, which then must be used to constrain the size of the shadow steps.

If the Lyapunov exponent λ of the system is known *a priori*, it may be possible to choose shadow step sizes *a priori*. Note that the divergence of the noisy and accurate orbits will be dominated by the errors in the noisy orbit. Say the noisy orbit has stepsizes of negligible length, each with error δ. Then each error will be magnified after a time T to $e^{\lambda T}\delta$. If E is the maximum allowable 1-step error for which the linear map is valid, then shadow steps should be no larger than

$$\frac{\ln(E/\delta)}{\lambda}.$$

For our N-body systems, this equation gives an upper bound of about 1 to 10 time units, which is larger than the 0.1 sized shadow steps I used. Thus, the above equation seems a weak upper bound.

Even a system with a particular Lyapunov exponent may have short intervals during which the local divergence rate is faster or slower than the Lyapunov exponent would suggest. Thus, it may be helpful to devise an algorithm that dynamically builds shadow steps. One such algorithm, as yet untested, is as follows. Assume that the timesteps in the noisy orbit are small in comparison to the shadow steps that will be built. Let the noisy orbit be $\{p_i\}_{i=0}^{Q}$ at times $\{t_i\}_{i=0}^{Q}$. Let E be the largest allowable 1-step error for which the linear map is valid. To construct a shadow step starting at time t_i, the accurate solution $p(t)$ is initialized to the noisy point $p(t_i) = p_i$. Accurate integration of $p(t)$ proceeds successively to times t_j, $j > i$ until $|p_{j+1} - p(t_{j+1})| > E$, when t_j marks the end of the shadow step. The accurate solution is then re-initialized to $p(t_j) = p_j$ to begin the next shadow step.

Now, the first refinement is likely to have the largest 1-step errors; thus, the first refinement will need the smallest shadow steps, and thus the largest RUS set of any refinement. This means that a large RUS set needs to be computed at least once, but only to low accuracy (see the section on "cheaper accurate integrator").

Originally I thought that, as refinement proceeds to decrease the 1-step errors, it would be advantageous to construct larger-and-larger shadow steps. However, I no longer believe this will give much of a speed savings, because (1) If refinement is working well, then constant RUS will be invoked, in which case there is no need to recompute the RUS at all; (2) Conversely, if refinement is failing, then there is no justification to increase the size of shadow steps — in fact, it may be advantageous to decrease them. Note that if refinement *is* succeeding, it may be advantageous to use larger-and-larger shadow steps in the computation of the 1-step errors, even though the RUS is not being recomputed.[6]

[5] The error in the error, so to speak.

[6] It is interesting to note in passing that the larger the shadow steps, the more obvious it is that refinement

Finally, implementing dynamic shadow step sizes may help in another area. As described above, I currently discard all progress made by constant RUS if it begins to fail, reverting to the orbit and RUS of the most recent refinement that computed the RUS. However, if constant RUS works for several refinements but then starts to fail, perhaps it would be advantageous to use the orbit of the most recent successful refinement to build new shadow steps using the above dynamic algorithm, regardless of whether the most recent successful refinement computed the RUS or not.

2.1.3 Cheaper accurate integrator

When computing the 1-step errors of a noisy orbit, we must use an integrator that has smaller 1-step errors than the integrator used to compute the noisy orbit. The question is, how much more accurate does it need to be? During early refinements when the 1-step errors are large, the errors in the correction factors may be dominated by the first-order approximation of the resolvents, so the 1-step errors need not be computed to machine precision. In practice, I have found that it is sufficient to compute the 1-step errors to an accuracy 10^{-3} that of the size of the maximum 1-step error of the previous refinement; computing them any more accurately does not speed refinement, although computing them only 10^{-2} more accurately sometimes results in more iterations of the refinement procedure.[7] In addition, a factor of only 10^{-2} is not enough because some integrators have sufficiently gross control of their errors that requesting only 2 extra digits will result in the integrator returning *exactly* the same result.[8] Note it is necessary to loosen this criterion when the 1-step errors are within 10^{-3} of the machine precision, in order not to request more accuracy than the machine precision allows.

The accuracy required to compute the resolvents is less clear. So far, when reverting to recomputed RUS, I have been computing them to the same accuracy as the 1-step errors, described in the previous paragraph. This may be more precision than is necessary, if, for example, the resolvents change more quickly owing to movement of the orbit than to errors in their construction. In other words, it may be sufficient to always compute a resolvent to some constant precision, and only recompute it if the numerical shadow drifts sufficiently far from the original noisy orbit that the resolvent no longer correctly maps small perturbations from one shadow step to the next.

Choosing the accurate integrator

The only methods I have tried as my accurate integrator are Adams's methods. It may be wise to try others. In particular, if the shadow steps are small, it may pay to use a less sophisticated routine, such as a high-order explicit Runge-Kutta method. If the shadow steps are large, an Adams's or a Bulirsch-Stoer method is probably apt, because even though they both have high

converges to a solution of the accurate integrator, not to a true solution. For, if the shadow step errors were small enough, and the machine precision allowed it, we would be able to "shadow" an entire noisy orbit using a single huge shadow step.

[7] More iterations, but each iteration takes less time. There is clearly a trade-off here that is probably problem dependent.

[8] This is probably because it did too much work in the less accurate case, so that the solution it computed had at least 2 more accurate digits than were requested. Then, in the more accurate case, it managed somehow to more accurately judge its errors, and avoided extra work. A difference of 10^{-3} is less likely to cause this to happen.

startup cost, they can eventually move quickly along the solution if they are not forced to restart too often. It may also be interesting to attempt using extended precision integration to test the accuracy of the standard (double precision) routines.

Finally, an important factor in problems, like the N-body problem, that have a large variance of scales all occuring at once (*i.e.,* some pairs of stars are several orders of magnitude closer to each other than other pairs) is a concept called *individual time steps*. The modern hand-coded N-body integrator gives each star a personalized integration timestep; stars that have high accelerations are integrated with smaller timesteps than those with lower accelerations. Perhaps it would be fruitful to attempt to write a general-purpose integration routine that uses individual timesteps for each dimension. However, such an integrator is beyond the scope of this thesis.

2.1.4 Numerical results of the optimizations

To quantify the speedup of the optimizations, 32 noisy orbits were chosen, and each was shadowed using several different combinations of the optimizations. In each case, the system consisted of 99 fixed particles and one moving particle (*i.e.,* $N = 100, M = 1$), identical to the shadowing experiments of QT. Forces were not softened. The 32 orbits were chosen by generating random 3-dimensional positions for all particles from the uniform distribution on $(0, 1)$; a 3-dimensional random initial velocity for the moving particle was also chosen uniformly on $(0, 1)$.[9] The standardized units of Heggie and Mathieu [14] were used, in which each particle has mass $1/N$. The pseudo-random number generator was the popular 48-bit *drand48()*, with seeds 1 through 32. The seed 1 case was run on a Sun SPARCstation 10/514 with a 50 MHz clock; seed cases 2 through 32 were run on 25 MHz Sun SPARCstation IPC's (each about 1/8 the speed of the SS10/514).

Once the initial conditions were set, each noisy orbit was generated by integrating for 1.28 standard time units (about 1 crossing time) using LSODE [16] with pure relative error control of 10^{-6}. This figure agreed well with the magnitude of the initial 1-step errors computed during refinement. Although one crossing time sounds short, it is long enough that 5 of the orbits contained trouble spots.[10] Furthermore, the number of unoptimized QT refinements required to find a shadow that has no trouble spots seems independent of the length of the orbit — each refinement takes much longer, but the convergence is still geometric per refinement. Thus, the results below should be independent of the length of the orbit, as long as the length is non-trivial. This is what has been observed with the longer orbits I have shadowed, although I do not document them here.

For short shadow steps, a constant shadow step size of 0.01 was used, which approximates the average sized shadow step in QT. This results in 128 shadow steps. For large shadow steps, 16 shadow steps of size 0.08 were used. As in a usual shadowing attempt, longer and longer segments are shadowed in an attempt to find the longest shadow. Here, each successive segment had twice as many shadow steps as the previous one, up to 16 and 128, for long and short shadow steps,

[9] Astronomers will note that this does not correspond to any realistic astronomical system; however, it seems unlikely that the precise particle distribution will effect shadowing results. This is supported by the close correspondence of my results with those of QT, even though they used a more realistic "Plummer" distribution.

[10] If the orbit has a trouble spot, then *Constant Resolvent* and *Re-use RUS from previous successful shadow* will have no effect, because the RUS will be re-computed in an attempt to find a shadow. The other optimizations — *Cheaper Accurate Integrator, Large Shadow Steps,* and *Backwards Resolvent by Inverting Forward Resolvent* — still offer significant performance improvement.

respectively. No attempt was made to isolate glitches more accurately than this factor of two.

The highest tolerance requested of the accurate integrator was 10^{-15}. A successful numerical shadow was defined to be one whose 1-step errors were all less than 2×10^{-14}. This number was chosen simply because it was the smallest number that geometrically converging refinement sequences could consistently achieve; 10^{-14} was too small, because many refinement sequences would proceed geometrically until their maximum 1-step error was about 1.5×10^{-14}, and then they would "bounce around" for many refinements until, apparently by chance, the maximum 1-step error would fall below 10^{-14}.

The maximum allowed shadow distance was 0.1, although none over 0.0066 were observed with successful shadows.

The table on page 32 displays the speedups of the various optimizations. All ratios are speedup factors with respect to the time column. There are several interesting observations to make about this table. First, note that the run times of the original QT algorithm are comparable to the run times of my unoptimized version (differing on average by a factor of 1.07), as would be expected. Now looking at each optimization acting alone, we see that using Large Shadow Steps 8 times longer than QT (column L) gives an average speedup of about 5.5. The large shadow steps implemented here (every 8 small shadow steps) was trivial to implement, so I would recommend that, even if no other optimizations are used, large shadow steps are worthwhile. Third, inverting the forward resolvent to produce the inverse resolvent (column I) produces the expected average speedup of 2.0. Fourth, the cheap accurate integrator (column C) gives an average speedup of 2.36. This is about what is expected, because both the QT and C algorithms on average require just over 3 refinements to converge, and all C refinements are cheaper than a QT refinement except the last one, which is about equal in expense. Finally, using constant RUS gives a speedup of almost 3. Again this is about what is expected because QT requires about 3 refinements to converge while R has one RUS computation followed by several cheap constant RUS refinements.

Next we look at combinations of optimizations. First, it is interesting to note that combining the cheap accurate integrator C with constant RUS R results in an average speedup that is greater than the product of the two individual speedups ($2.36 \times 2.68 = 6.32 < 7.55$). This is because, when using CR, only *one* RUS is computed, and it is computed cheaply. Using R alone requires computing the one RUS to high accuracy; using C alone requires computing at least one RUS (the final one) to high accuracy. Second, re-Using the RUS of a previous successful shadow (U or re-Use RUS, seen in column CUR) makes sense only when R is also used. It gives a speedup of about 43% over CR. This is less than a factor of 2 because, with C, the resolvents are computed sufficiently cheaply that the time to do a constant RUS iteration is becoming non-negligible. Presumably if a UR column existed, it would show a speedup over the R column closer to 2. The last three columns show other combinations. The most important point to note is that, except for CR, the combinations give a speedup less than the product of any appropriate choice of previous columns. Perhaps this is at least partially owing to an affect similar to Ahmdal's Law: there still exist parts of the algorithm that have not been sped up, and as the remainder of the program that *is* being optimized speeds up, these unoptimized sections take a greater proportion of the time.

Finally, it will be noticed that orbits 5, 6, 13, 19, and 29 obtained speedups significantly smaller

seed	time(s)	QT	L	I	C	R	CR	CUR	CLR	CULR	CULRI
1	1442	1.06	4.54	2.01	1.96	3.00	8.87	14.2	30.8	37.2	47.4
2	9379	0.707	6.08	1.82	3.09	3.48	11.2	15.7	52.3	68.3	75.7
3	7847	0.728	6.30	1.88	2.36	2.56	9.29	13.0	44.0	59.7	65.5
4	7011	0.866	11.1	1.98	2.74	2.84	7.88	12.3	40.5	55.8	74.3
5	12204	0.933	12.2	2.13	2.42	1.08	1.66	1.54	5.31	4.96	8.0
6	9443	1.14	7.24	2.01	3.18	3.98	10.8	13.5	5.36	56.8	29.8
7	6277	1.13	5.51	1.85	2.63	2.71	8.50	11.6	46.9	63.7	82.9
8	6582	1.13	4.79	2.04	1.91	2.79	7.09	10.6	33.8	45.4	56.9
9	7154	1.14	4.33	1.92	1.96	2.70	7.74	11.5	30.5	39.6	51.9
10	5832	1.13	4.32	1.87	1.54	2.72	6.46	10.7	29.1	45.5	57.2
11	6659	1.13	5.10	1.93	2.34	2.65	7.40	10.7	37.1	50.7	62.9
12	8623	0.861	5.00	2.44	2.57	3.37	9.64	13.7	44.8	52.4	75.4
13	14566	0.452	4.34	1.00	4.52	0.784	0.931	0.939	5.67	5.15	7.6
14	6976	0.749	4.54	2.00	2.17	2.85	7.23	11.1	32.9	45.2	49.4
15	5664	0.729	4.79	2.02	1.89	2.71	6.83	10.2	33.5	45.4	62.9
16	6634	1.15	4.89	1.97	1.82	2.64	7.35	10.6	33.8	44.8	58.7
17	6854	0.946	4.96	2.06	2.16	2.88	8.97	12.7	47.2	60.8	83.2
18	4932	0.629	2.88	1.74	1.69	2.17	5.38	7.84	25.7	34.1	45.7
19	8470	1.06	5.16	2.98	2.96	1.32	4.37	4.02	31.1	27.5	53.5
20	4566	1.12	5.59	1.91	2.04	2.78	6.21	9.50	33.7	48.6	64.3
21	7409	1.07	6.01	2.00	2.92	2.86	9.59	13.4	46.6	61.3	82.2
22	6988	1.22	4.89	2.17	2.00	3.03	7.34	11.4	36.1	44.0	57.2
23	7699	0.750	4.66	1.93	1.91	3.20	8.27	10.4	35.2	46.3	51.1
24	8851	1.18	2.53	1.79	2.78	2.89	10.2	15.4	52.9	71.1	93.1
25	7287	1.13	3.58	1.96	2.67	2.53	7.58	11.5	34.1	47.4	59.3
26	10736	0.854	7.74	2.84	2.79	4.33	12.4	16.9	58.2	76.5	81.1
27	7001	1.13	5.07	2.01	2.24	2.79	8.35	12.1	38.9	54.5	70.1
28	7324	1.13	4.49	2.06	1.93	2.72	8.01	11.2	33.2	49.3	55.0
29	13779	0.802	4.01	1.89	2.01	0.898	1.16	1.15	4.03	3.99	6.5
30	7932	1.14	10.9	2.01	2.17	2.81	8.82	13.0	43.3	67.5	86.7
31	6989	1.20	5.57	2.26	2.37	2.81	8.44	12.3	47.4	57.2	77.6
32	6720	1.08	4.80	1.89	1.90	2.78	7.61	11.5	34.0	50.5	65.5
avg	7682	1.07	5.56	2.01	2.36	2.68	7.55	10.8	34.6	47.5	59.3

	LEGEND
time(s):	time in seconds for unoptimized version.
QT:	original code of Quinlan & Tremaine [29]
L:	*L*arge shadow steps
I:	backward resolvent by *I*nverting forward resolvent
C:	*C*heaper accurate integrator
R:	constant *R*US
U:	re-*U*se RUS from previous successful shadow (appears only in combinations)

than average in some columns. These are the orbits that either contained glitches or trouble spots.[11] I will deal with these on a case-by-case basis, since there are several interesting points to be made.

orbit 5 : As can be seen in Figure 2.1, this orbit suffered a close encounter at timestep 83 (out of 128). Figure 2.2 shows that the energy also took a large dip at step 83. The orbit could be

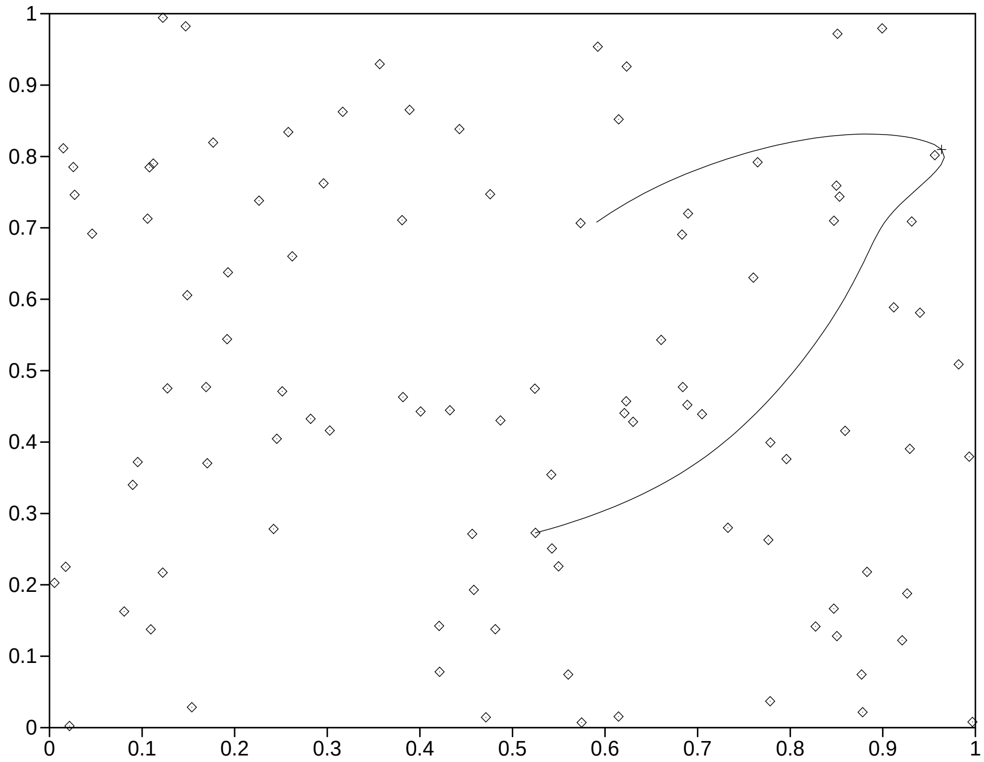

Figure 2.1: Orbit 5 projected onto the xy plane. The close encounter is near the right-most extent of the orbit. The "+" marks the point in the orbit where the 1-step errors could not be decreased below 10^{-13}, and corresponds to the timestep of closest approach. The particle attached to the bottom end of the orbit near $(0.52, 0.27)$ is the moving particle at its initial position.

shadowed for its first half, but not over its entire length, because it contained a trouble spot at the close encounter, where the 1-step error could not be decreased below about 10^{-13}.

This orbit obtained the greatest speedup when the only active optimization was L — the final 3 columns show that the algorithm was slower when other optimizations were added. The reason seems to be constant RUS: since the 1-step errors could be reduced reliably down to 10^{-13}, but no lower, the algorithm switched many times to constant RUS (at which point the 1-step errors would be decreased to 10^{-13}, but no lower), and then back to recomputed RUS, which would run for 1 or 2 refinements, producing refinements better than constant RUS but not geometric; then the algorithm would switch back to constant RUS, again decreasing the 1-step errors to 10^{-13}, then switch back to recomputed RUS, and so forth. Since all progress of a sequence of constant RUS refinements is discarded, progress was slow, until, finally,

[11] Although I have never seen a clear definition of "trouble spot", it seems that other practitioners of shadowing, *eg.,* GHYS and QT, use the term to refer to any trajectory on which refinement encounters a point on the trajectory where the 1-step errors do not decrease with refinement, although they do remain small and bounded.

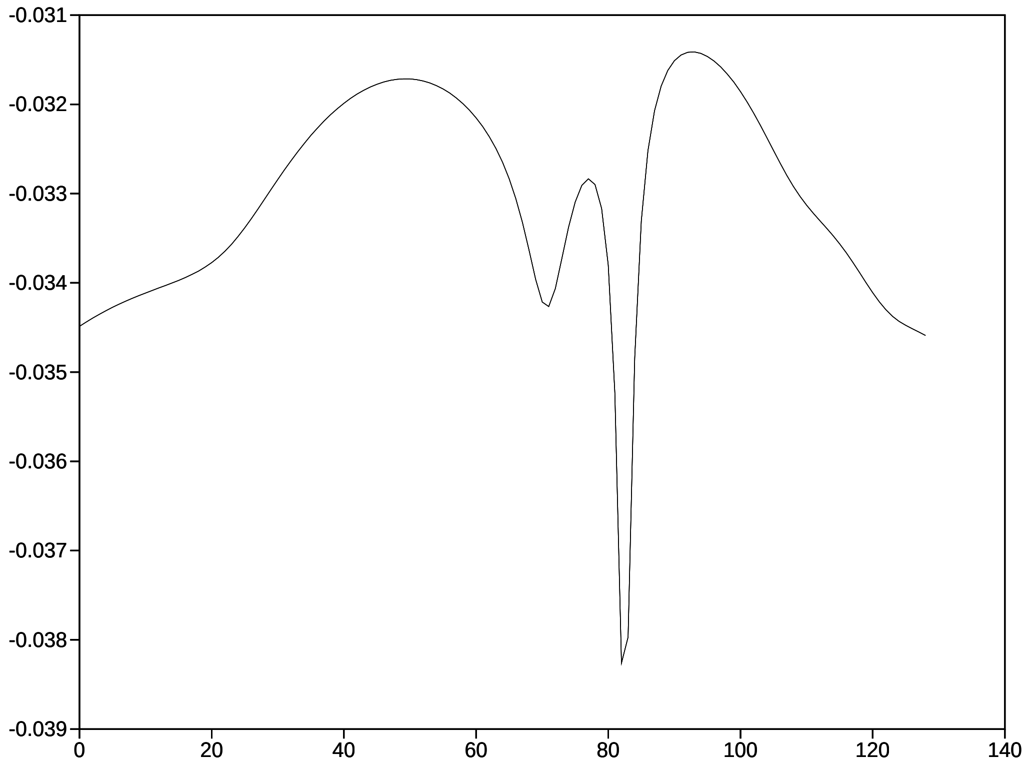

Figure 2.2: The total energy (kinetic + gravitational potential) of orbit 5. The deep trough at timestep 83 marks the close approach that caused the trouble spot.

recomputed RUS refinements got down to 10^{-13}, at which point the algorithm conceded defeat. However, other columns still make sense: for example, comparing column $CULR$ to $CULRI$ shows a speedup of almost 2, which is expected with the I optimization.

orbit 6 : This orbit also suffered a close encounter, near timestep 87. The total energy of the system has a sharp dip at that time, similar to that of orbit 5.

The refinement algorithm came excruciatingly close to finding a numerical shadow for this orbit, at one point achieving a maximum 1-step error of 2.36×10^{-14} at timestep 87. Columns L, C, R, and CR show significant speedups, but the combination CLR has an large drop in speedup, suffering the same alternation between constant RUS and recomputed RUS as did orbit 5. Perhaps the Large Shadow Step at that point was just slightly too large, thus having a linear map that was invalid. A similar fate befell the column $CULRI$, which is surprising considering that only half as many resolvents as column $CULR$ were computed per recomputed RUS refinement.

orbit 13 : This particle suffered two close encounters, both in the latter half of the orbit. The energy had a large dip at each of the close encounters.

Shadowing was easily successful for all attempts that only included the first half of the orbit. However, the attempt to shadow the entire orbit was fraught with problems trying to satisfy the extreme sensitivity of the orbit to two close encounters. The second encounter was sufficiently close that the LSODE integrator in the non-optimized version suffered a fatal convergence error on the 4th refinement, trying to integrate the backwards resolvent past the encounter. This explains why the I column shows no speedup: it did not integrate any resolvents backwards, and by chance the forward resolvents could be integrated well enough that it tried more than twice as many refinement iterations than the unoptimized version before giving up.[12] The R version took longer because of time spent alternating between constant and recomputed RUS.

orbit 19 : This orbit was shadowable for the first half, but not when the second half was included. However, as can be seen in Figure 2.3, this particle suffered no close encounters. In addition,

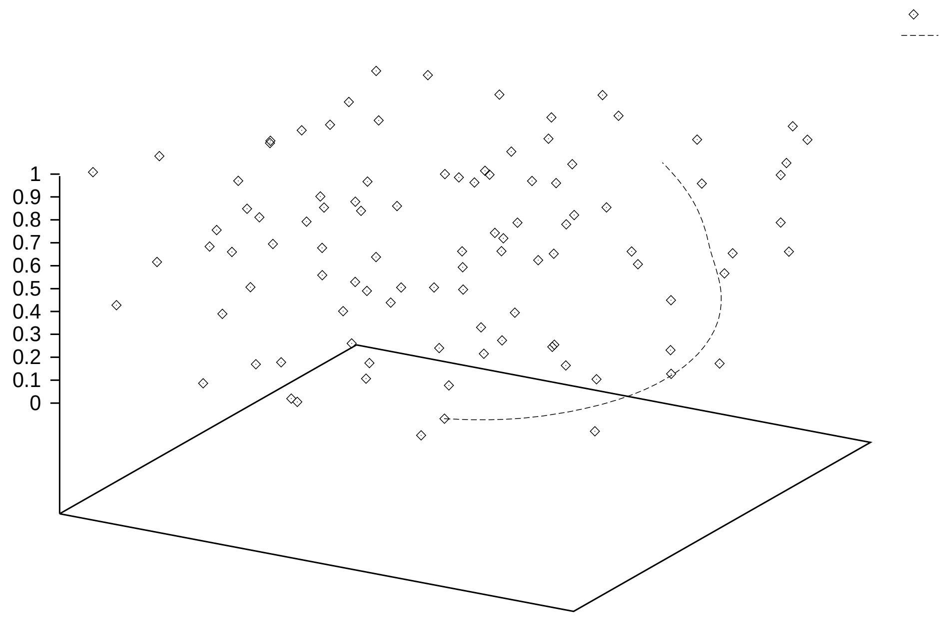

Figure 2.3: Orbit 19. Note there are no close encounters. It is not clear why this orbit was not shadowable.

the energy (Figure 2.4) has no sharp dips like the other non-shadowable orbits did. The observed energy fluctuation alone does not seem to provide an adequate explanation of the non-shadowability of this orbit either, because other orbits, such as orbit 22 (Figure 2.5), had even larger energy fluctuations, but were still shadowable over their entire length. The largest 1-step errors were consistently at or near time 0. Since the entire orbit appeared extremely smooth, it is not understood why no shadow could be found over the entire length of the noisy orbit.

Finally, referring to the speedup table, most of the speedups appear reasonable.

orbit 29 : This particle suffered a close encounter at timestep 53 (out of 128). The first 32 shadow steps were shadowable, but the first 64 were not, and neither was the orbit as a whole. There are no surprises in the speedup table; this orbit simply has a glitch at timestep 53.

[12] Although it is completely by chance that it took exactly the same time to 3 digits.

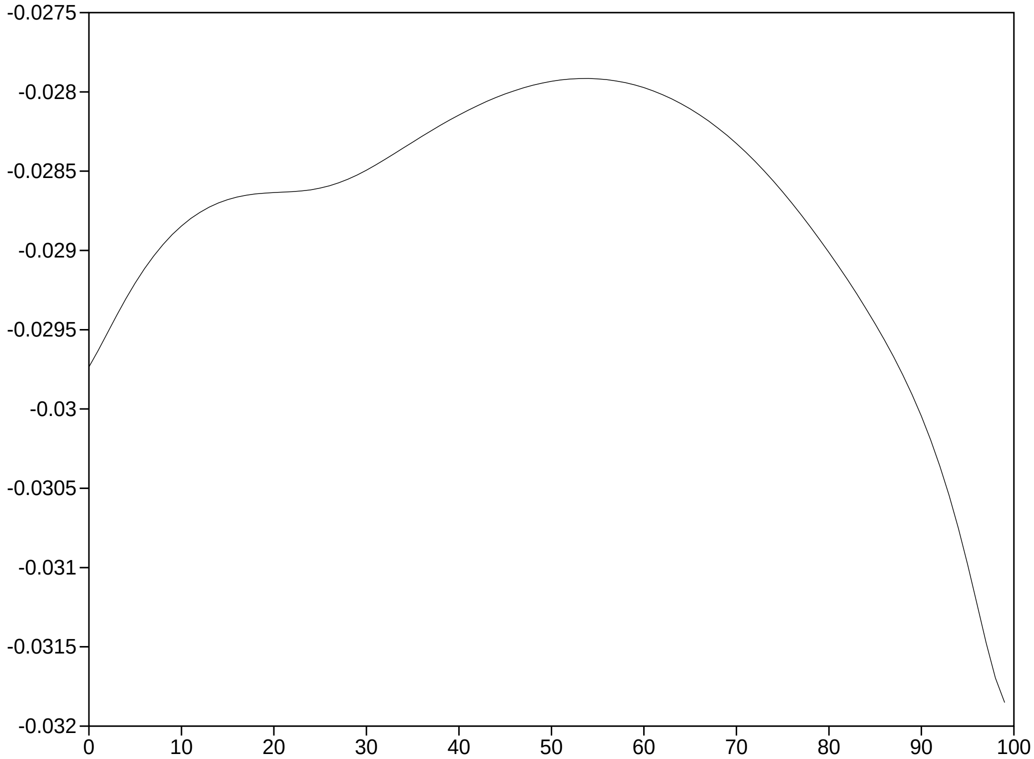

Figure 2.4: Energy of orbit 19. There are no sharp dips. It is not clear why this orbit was not shadowable.

There are many other observations about the shadowing attempts that cannot be seen in the table.

- For any given seed, the various optimized versions generally produced shadows agreeing to 12 or 13 decimal places. (For large shadow steps, of course, only the points that appeared on the large shadow steps — every 8 small steps — could be verified.) However, when computing shadows of short orbits, the number of agreeing decimal places was much smaller, only 5 or 6 in some cases. I do not believe this is a problem, because with a short orbit, the neighborhood of points that stay near to each other under evolution of the system is large. Furthermore, the stable and unstable subspaces (recall, computed with arbitrary — but in these cases equal — subspaces at the endpoints) will be slightly different, because of the resolvents having different numerical errors. As the orbit gets longer, the neighborhood of initial conditions that stay close together gets much smaller, and so any algorithm that computes a shadow must compute ones that agree to many decimal places.
- When using constant RUS, the number of iterations required to find a shadow is always greater by a small factor (usually 2 or 3) than not using constant RUS. However, this is far offset by the speed of a constant RUS iteration, which took an average time of 11.4 seconds, while a recompute RUS iteration averaged 157 seconds — 14 times longer. Since computing a resolvent takes $O(M^3)$ time, and computing a 1-step error takes only $O(M^2)$ time, this speedup is asymptotically $O(M)$.
- The various optimizations seem to decrease reliability slightly. Although there was never a

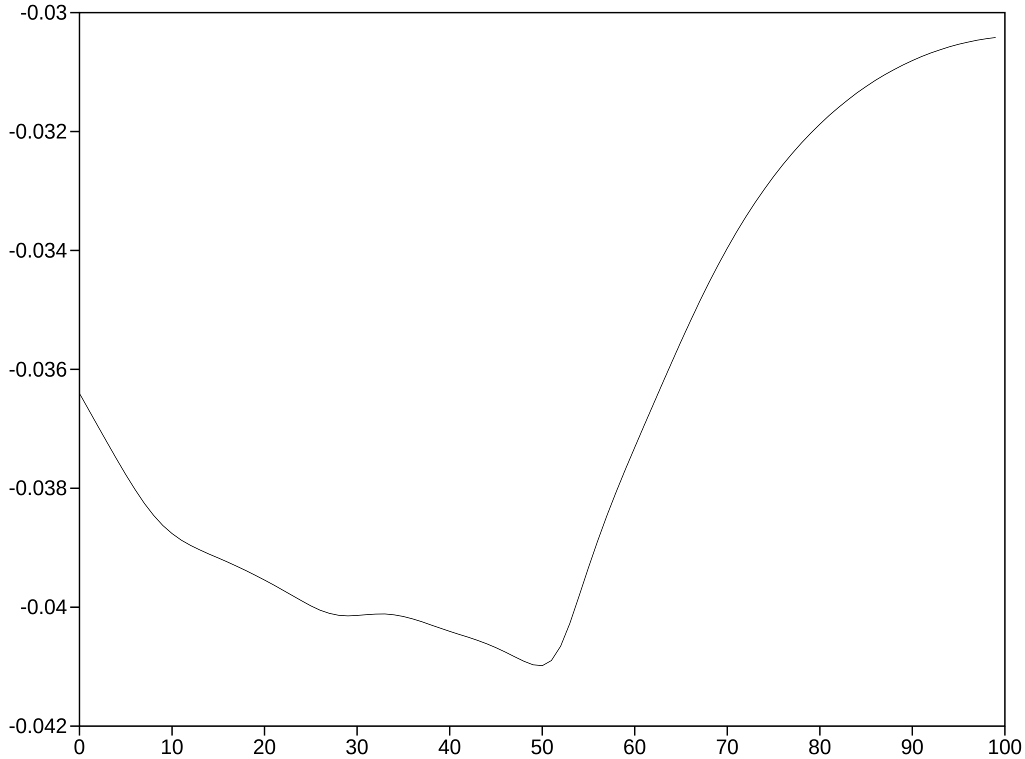

Figure 2.5: Energy of orbit 22. The total energy variation is more than twice as large as that of orbit 19, but this orbit was still shadowable.

case (in this table) where an optimized version found a shadow that the non-optimized version failed to find, there were a few cases vise versa. No repeatable pattern was observed in these failures, so the reason is still a mystery.

2.2 Reliability of refinement

2.2.0 Refinement is not infallible

QT argue that if the refinement algorithm fails, there is good reason to believe that no shadow exists. They apply two arguments. First, from the more rigourous study of simpler systems, glitches are known to exist and are not just a failure of any particular refinement algorithm. Secondly, QT's results are consistent with a conjecture by GHYS on the frequency of glitches.

However, during the many shadowing runs done for this thesis, I came across several cases in which a shadow failed to be found for shadow steps $0..S$[13], but a shadow *was* found for the superset $0..2S$. This occured for various values of S from 1 up to 32, where shadow steps were of size 0.1, and tended to occur more often with larger M. Thus, there were cases in which a shadow was not found for a noisy segment 3.2 time units long, but one was found for an extension of the same

[13] $0..S$ means the orbit from the beginning of shadow step 0 to the end of shadow step S.

segment out to 6.4 time units. Clearly, if a shadow exists for 6.4 time units, the first 3.2 time units of it must also be a valid shadow of the first 3.2 units of the original noisy orbit. This was verified by using the first half of the 6.4 unit shadow as an initial guess for the 3.2-length shadow; the refinement procedure then quickly succeeded in constructing a shadow with the standard boundary conditions for a 3.2-length shadow. Thus, the refinement procedure initially failed to find a shadow for the 3.2-length noisy orbit, even though one exists.

One of the smaller noisy orbits for which this occurred was also re-tested with all the above optimizations turned off, with the same result. Thus this phenomenon does not seem to be related to the optimizations.

It is difficult to say why this happens, but it certainly seems significant. Such a phenomenon also occurs with Newton's method: whether it converges often depends critically on the initial guess; a certain guess may converge to a zero, while a nearby guess may not. To lessen the chance of this failure, it is recommended that if the refinement procedure fails to find a shadow for a length S, that the noisy orbit's length be increased (*eg.,* doubled), and then shadowing tried again on the longer orbit. This is expensive, but necessary to lessen the chance of failing to find a shadow that exists.

2.2.1 Other reliability issues

There are several other interesting issues concerning reliability.

Is it necessary to refine an orbit until the 1-step errors are close to the machine precision? QT noted that most refinement sequences usually either showed geometric convergence within the first few refinements, or else didn't converge at all. Most of the experiments in this thesis showed similar behaviour. Thus it may seem that if geometric convergence is observed, it is safe to assume that geometric convergence would continue, and stop refinement before reaching the machine precision. However, I have also observed several cases in which refinement progressed quickly at first, but then the 1-step errors stopped decreasing before reaching the machine precision. I think these cases raise some interesting issues. It is arguable that these cases are the most interesting ones: *why* did refinement work at first but then fail? What characteristics of the orbit make it hard to shadow? If refinement works at first but fails before reaching the machine precision, does this mean there exist cases in which hypothetical refinement *beyond* machine precision would have failed? If this were the case, then clearly refinement to machine precision does not *always* imply the existence of a true shadow orbit. Sometimes this may simply be a word length problem: perhaps convergence would have continued if more floating-point digits were available. I do not believe this is always the case, however, because sometimes convergence stopped 6 decimal digits above machine precision (*i.e.,* at about 10^{-10}).

One problem with the current standard machine precision of about 10^{-16} is that there is not much room between the scale at which geometric convergence starts (about 10^{-5} or so), and the machine precision. Thus it may be difficult to recognize geometric convergence when it occurs. Possibly, more research into higher-precision refinement should be conducted. Conversely, I have seen cases in which refinement seemed not to be progressing well for several steps, but eventually refinement to machine precision was successful. This phenomenon sometimes also is observed using

Newton's method on standard zero-finding problems. It may or may not be an interesting issue specifically concerning shadowing.

Finally, when deciding if a refinement is successful, how close to machine precision is "close"? I have found empirically that as the number of dimensions increases, the minimum achievable maximum 1-step error increases slightly with the number of moving particles. Given a machine precision of $\varepsilon_{mach} \approx 10^{-16}$, I have found that maximum 1-step errors reach a plateau somewhat below about $10^4 \ dt_{out} \ M \ \varepsilon_{mach}$, where dt_{out} is the size of the shadow step. Although I have not tried to precisely quantify this effect, there are reasonable arguments to explain it. First, when large shadow steps are used, it should be no surprise that even the accurate integrator's output will vary based on slight perturbations of the input, and that the output variations will be larger if the shadow step is longer. Since the errors may be assumed to be random at each internal step taken by the accurate integrator, we may expect a variation-of-sum effect that grows as $O(\sqrt{dt_{out}})$; this is probably a better estimate than the linear growth assumed in the above formula. Furthermore, the minimum achievable maximum 1-step error may be expected to grow as $O(\sqrt{M})$, rather than M, because the Euclidean norm used to measure the 1-step error has $O(M)$ terms inside the radical. Further study of a reasonable limit may prove fruitful.

Another problem is bounding the maximum number of refinements allowed before the algorithm admits defeat, even if no other "fail" criterion has been met. I have found cases in which refinement had to switch back and forth several times between constant RUS and recomputed RUS, resulting in 40 or more total iterations (over 3/4 of which used constant RUS), eventually finding a successful shadow. It is hard to know how any particular upper limit on the number of refinements would affect reliability. I have arbitrarily chosen 100, although some may consider this high. Usually, however, if the number of iterations is more than about 10, then shadowing the orbit for twice as long will fail. In other words, if the number of refinement iterations is high, then there is a mild trouble spot somewhere in the orbit. This trouble spot will become even more of a problem as we attempt to build a longer shadow, because the volume of phase space around the trouble spot that can contain shadows shrinks as the attempted shadowing distance increases (see the discussion on "brittleness" of orbits in [8]). The total number of refinements, when refinement is working well, seems independent of the dimension of the problem, and is typically about 5.

Chapter 3

Some preliminary results of high-dimensional shadowing

"Think not thy own shadow longer than that of others."
— Sir Thomas Browne, *Christian Morals.* (1682)

Using this optimized algorithm, shadowing experiments were performed on N-body systems in which M bodies move amongst $N - M$ fixed ones. For systems of $\{N = 100, M\}_{M=1}^{25}$ with a variable-timestep integrator and no softening, our results show that the length of time an orbit is shadowable decreases with increasing M. However, it is unclear whether this is owing to collective effects of interacting moving particles, or whether each particle individually has a "glitch rate", causing the global glitch rate to increase linearly with the number of particles. However, for a system of $N = 65536, M = 1$ with softening and integrating using constant timestep leapfrog, we were able to shadow the moving particle for two dozen crossing times, which is encouraging.

3.0 High-dimensional N-body shadows encounter difficulty

When employing any particular method to study the reliability of large N-body simulations, it is essential that the method follow the N-body system for a typical duration that an astronomer is likely to simulate the system. Thus the short simulations in the previous chapter tabulating speedups for simulations lasting one crossing time, though good enough for measuring speedups, are not appropriate for studying the reliability of long running large N-body simulations. Although the number of large N-body systems that I have studied is still small, some interesting trends have already been seen.

The shadowing attempts documented in this chapter were produced in the same fashion as the previous chapter, except that longer shadowing times were attempted, up to a maximum of 256 shadow steps (25.6 standard Heggie and Mathieu time units). Shadowing was attempted first on 1 shadow-step, then doubling the number of steps until two successive failures occured on noisy orbits $2S$ and $4S$ steps long, where S was the longest shadow that was successfully constructed. In all the simulations, the total number of particles was held at $N = 100$, while the number of moving particles M was varied from 1 to 25; $N - M$ particles remained fixed.

Figure 3.0 plots the longest shadows found in the above systems as a function of M. The number samples per value of M was 10 for all except M equal to 1,3, and 5, which had 50, 22, and 13 samples, respectively. Although the sample sizes are small and there is much noise in this graph, it clearly shows that the length of the longest shadow that could be found, using the algorithms in this thesis, decreases with increasing M. These shadowing attempts find the glitch position to within a

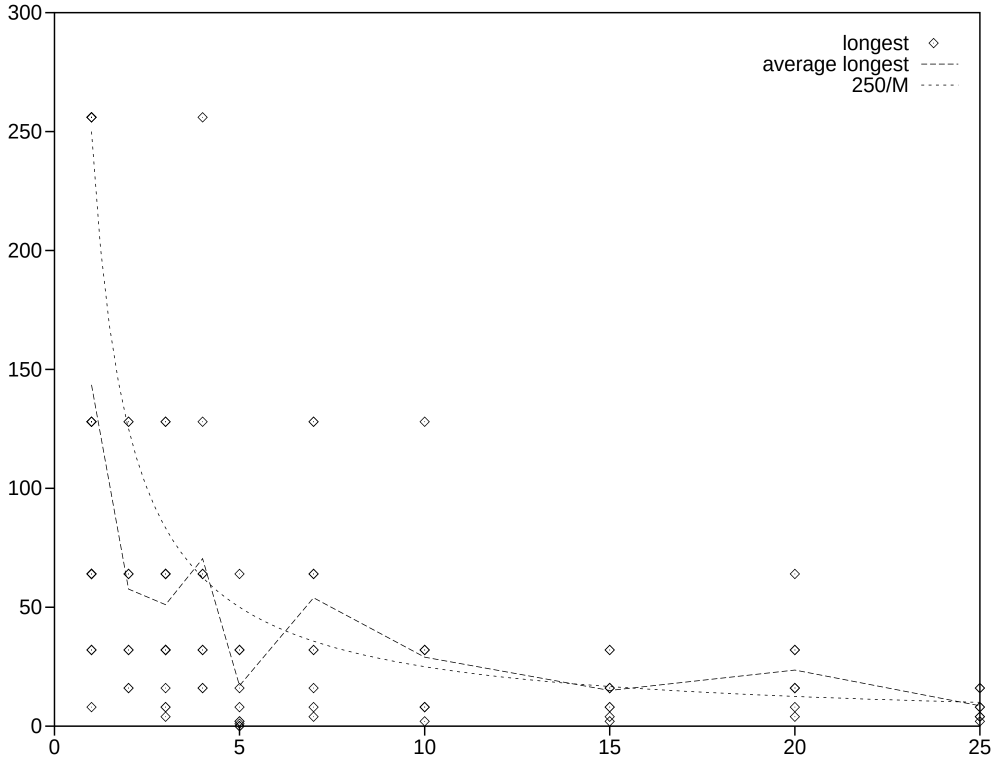

Figure 3.0: Longest shadows found in unsoftened systems of $N = 100$ particles, as a function of the number of moving particles M, while $N - M$ of the particles remain fixed. The vertical axis measures shadow steps, which were each 0.1 standard time units. Each diamond represents the longest shadow found for a particular orbit. The piecewise-linear line links the averages of these systems for each M. For comparison, the curve $250/M$ is also plotted.

factor of 2. The amount of noise in this graph could be decreased significantly by attempting more accurately to pinpoint the timestep of a glitch.

Although this result does not look promising for N-body simulations, there are a number of things to keep in mind. First, it is not clear that allowing $M > 1$ particles to move amongst $N > M$ particles is significantly more realistic, in comparison to real systems, than having only 1 particle moving, unless $M >> 1$. This is because as long as $M << N$, each moving particle acts independently of the other moving particles. Until the number of moving particles is comparable to the number of fixed ones, the moving particles will still encounter fixed ones more often than their moving counterparts. It is not clear that 25 out of 100 is many enough moving particles.

Second, perhaps the method used to scale this problem is not realistic. Perhaps a more realistic scaling of the problem, at least from the astronomer's point of view, would be to have M particles move amongst $100M$ fixed ones. This would smoothen the gravitational potential, which we know from other studies [20, 10] decreases the Lyapunov exponent, and thus may lengthen the average shadow length.

Despite the above caveats, there is reason to believe that high-dimensional shadowing may be difficult. Dawsen *et al.* [8] show that shadowing becomes extremely difficult in systems where a Lyapunov exponent fluctuates about zero. They claim that such fluctuating Lyapunov exponents occur frequently in high-dimensional systems. I do not believe the Dawson result applies to Hamiltonian systems, because it can be shown that the number of positive eigenvalues in a Hamiltonian system is always equal to the number of negative ones.

However, I think there is another reason that high-dimensional shadowing of large N-body systems may be difficult. Assume that, for a fixed noise amplitude, there exists a mean shadow length L for a QT-like system of 1 particle moving amongst $N >> 1$ fixed ones. (It is possible that no such mean exists, if the scatter in shadow lengths is great enough.) Then, in a system in which $M > 1$, $M << N$ particles move, each moving particle will encounter fixed particles far more often than it encounters other moving particles. Thus each particle, if followed individually, will have a mean shadow length comparable to L. Since work in this thesis and previous work has shown that glitches seem to occur most often near close encounters, and since close encounters occur as a stochastic process[0], a shadow length of L is equivalent to a mean *glitch rate* of $1/L$ — *i.e.,* a particle encounters glitches at a rate of $1/L$ per unit time. Thus, the system of M moving particles, as a whole, encounters glitches at a rate M/L per unit time, thus resulting in shadow lengths proportional to L/M. As M becomes large enough to become comparable to N, the rate that moving particles encounter other moving particles increases, perhaps offsetting the fact that each encounter lasts a shorter period of time. This leads to the following conjecture:

Conjecture 1 *If a chaotic system with D dimensions has an average shadow length of T time units, then the equivalent system scaled appropriately to MD dimensions will have an average shadow length of T/M time units, if everything else is held constant. (Especially the integration accuracy.) In other words, shadow length is inversely proportional to dimensionality.*

The graph in figure 3.0 seems consistent with this conjecture, as the curve $250/M$ indicates.

3.1 One QT-like experiment with $N = 65,536$, $M = 1$

One experiment of a QT-like system was performed with 65,535 fixed particles and 1 moving particle. This shadowing experiment took 20 hours on a Fujitsu VPX240/10 vector supercomputer. A vectorization percentage of about 90% was achieved. A reasonable estimate for the time for this simulation would take on a Sun SPARCstation IPC would be about 100 times longer — about 10 weeks. (If 99% vectorization could be achieved, it would be about 1,000 times longer — about 2 years.)

[0] See the discussion on stellar kinematics in [25], particularly pages 431–438.

As with all the simulations in this thesis, the particles each had mass $1/N$, were distributed uniformly in the unit cube, and the initial velocity of the moving particle was also generated in the uniform unit cube in velocity space. However, the intent for this simulation was to determine how the particle would react to a smooth potential, so softening was set to 0.01; the average inter-particle distance is 0.024. The noisy integrator used was time-centred leapfrog with a constant timestep of 0.001. Leapfrog is a second-order, time-symmetric, symplectic integration method used commonly by astronomers doing large N-body simulations. Shadow steps were of size 0.1, and shadowing was attempted over shadow step sequences of length of 1,2,4,8,.... The longest successful shadow was 512 shadow steps, or 51.2 standard time units. This is quite a long shadow, and is encouraging for simulations of softened systems. Considering that the unsoftened $M = 1$ systems rarely had shadows longer than 256, the one sample taken here with a shadow of length 512 would seem to suggest a longer average shadow — although only by a factor of 4.

In hindsight, this experiment may not be significantly more realistic than one with, say, $N = 10^3$, $M = 1$ and appropriate softening. However, it does seem to show that in a smooth potential, shadowing times are significantly longer than in more collisional systems.

3.2 An explanation of one of QT's figures

Figure 3.1 contains the data of QT's figure 6. In their text, they mention that part (b) of their figure was included simply because the scatter was less when they plotted shadow length in timesteps rather than absolute time. (Recall that they used every internal timestep of their integrator as a shadow step; thus the shadow steps would be shorter near close encounters.) They could not provide an adequate explanation of why the scatter was less in such a graph. What follows is a tentative explanation.

First, I assume that there are two distinct error magnification processes effective in N-body systems: one is magnification due to the global potential, and the other, a much higher magnification that acts much less frequently, is due to collisional encounters.[1] Thus, if we were to plot successive times at which a doubling (or e-folding) occurs, the inter-doubling times would be shorter near close encounters than far from them. Second, different solutions obviously have their close encounters at different times in their evolutions. Third, define *pseudo-time* to progress in units such that the changing doubling times occur at constant intervals in pseudo-time. The result is that pseudo-time speeds up near close encounters, in comparison to real time. Since glitches occur more frequently near close encounters in real time, glitches will occur at a more uniform rate in pseudo-time. Finally, the dynamic timesteps used by QT's integrator will more uniformly follow pseudo-time than does real time, since it uses small timesteps near close encounters. In other words, the scatter of glitch occurrence (*i.e.,* end of a shadow that follows the noisy orbit) is less in pseudo-time (and therefore in integration timesteps) than in real time. This may explain why QT's graph 6(b) has less scatter than their graph 6(a).

[1] There is some disagreement that these two factors exist. There seems to be an ongoing debate between Kandrup and Smith [19, 21, 20], who argue that the growth of errors is due both to the global potential and to collisional effects, while Goodman, Heggie, and Hut [10] argue that collisional encounters are the *only* process for the magnification of errors.

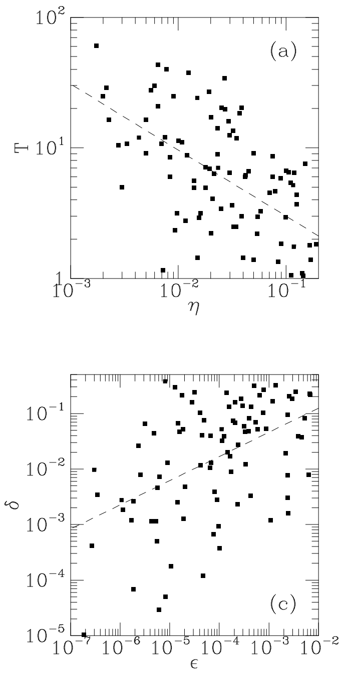

Figure 3.1: Data from QT's figure 6, used with permission. Their caption was: "Results from attempts to shadow noisy orbits in the unsoftened Plummer model of Fig. 1 [their figure 1, not included in this thesis] (each point represents a different orbit): (a) maximum time T for which the orbit could be shadowed versus the accuracy parameter η used to generate the orbit; (b) maximum number of time-steps N for which orbit could be shadowed; (c) shadow distance δ versus the maximum one-step phase-space error ε in the noisy orbit; (d) ε versus η. The dashed lines are least-squares fits to the data." See text for a possible explanation of why (b) has less scatter than (a).

Chapter 4

Further work

"On voit courir après l'ombre Tant de fous qu'on n'en sait pas."
("One sees chasing after shadows more fools than one can count.")
— La Fontaine, *Fables.* (1668)

Finally, there is much further work that should be done both in general high dimensional shadowing, and in the N-body shadowing in particular. We point out some possible directions for further research.

There are two distinct questions addressed in this thesis. One deals with shadowing of high-dimensional systems in general, and the other deals with shadowing of large N-body systems in particular. It is not clear that these questions are close enough to each other for generalities to be drawn from conclusions for the N-body problem. For example, the previous chapter noted that scaling the problem by increasing the number of moving particles, while holding the total number of particles constant, may not be the most realistic scaling method. From the perspective of shadowing in general, it seems that this method changes as little as possible in the system while the dimensionality of the problem is increased, thus arriving at a more "pure" result about how shadowing behaves as the dimensionality increases. But astronomers may be more interested in how the problem scales as the collisionality decreases with increasing N; thus having M moving particles amongst $100M$ fixed ones seems a more apt model for studying this question, because the gravitational potential becomes smoother as the total number of particles is increased (assuming the total mass is kept constant, so each particle has mass $1/N$).

As QT point out, these are two separate questions, even for the N-body problem: even if, in general, shadowing becomes more difficult as the number of dimensions increases, the N-body problem becomes less chaotic (*i.e.,* a smaller Lyapunov exponent) as the potential becomes more smooth with increasing N. The question of how these two processes interact to affect shadowing of large N-body systems is still open. Perhaps, even if Conjecture 1 is correct, the collisionality of large N-body systems decreases enough with increasing N to offset the decreasing average shadow length with increasing dimensionality.

In the first section of this chapter, I look at possible future work for shadowing of N-body systems; in the second section, I look at future work for shadowing in general.

4.0 Future work in N-body shadowing

4.0.0 A better understanding of stable and unstable directions

We should develop a better understanding of precisely what makes a direction stable or unstable. When a moving particle passes close to a fixed particle, do any of the 3 stable and 3 unstable directions lie along any of the obvious directions defined by the encounter? For example, along the path? Perpendicular to the path in the plane of the orbit? Perpendicular to the plane of the orbit? Figure 4.0 (page 47) shows some possibilities. It would be relatively easy to compute and output the dot product of the stable and unstable vectors with these special directions during a collision, but an even better understanding could be had by graphical visualization. What I have in mind is a "roller coaster" ride on the moving particle of an $M = 1$ system as it moves throughout the system. Along with the view from the particle, the vectors of the stable and unstable subspaces could be rendered on screen, changing dynamically as the particle moves throughout the system.

4.0.1 Spherical *vs.* disk systems; collisional *vs.* not

To my knowledge, all large astrophysical N-body shadowing experiments and analyses have focused on collisional spherical systems; none have focused on disk-like or collisionless systems, such as a cold disk. Cold disks may be an ideal testing ground for shadowing of N-body systems where collisionality effects are minimized.

4.0.2 Local glitch propagation

Definitions: A *global orbit* is a trajectory comprising the entire set of phase space co-ordinates of the system as a whole. A *local orbit* is a subset of the phase space dimensions of the global orbit that follows one individual particle. The terms *global noisy orbit, local noisy orbit, global shadow orbit*, and *local shadow orbit* are the obvious extensions of these terms. A *local glitch* occurs when one particle diverges from all possible shadows.

What is the effect on the system, as a whole, if two particles suffer a close encounter between each other that causes them to diverge from all possible shadows? Do the dimensions act mostly alone, suffering local glitches in small sets, or is there significant interaction between dimensions? Are other particles affected almost immediately, resulting in a "cascade of local glitches", or does each particle's orbit diverge independently, largely unaffected by other local glitches?

In a realistic model in which all particles move under their mutual gravitational influence, a useful measure of error may be the number of particles that are still locally shadowable. In other words, the high-dimensional phase-space shadow consists of individual paths for each particle; when one particle undergoes a local glitch and starts to diverge from all possible shadows, the remainder

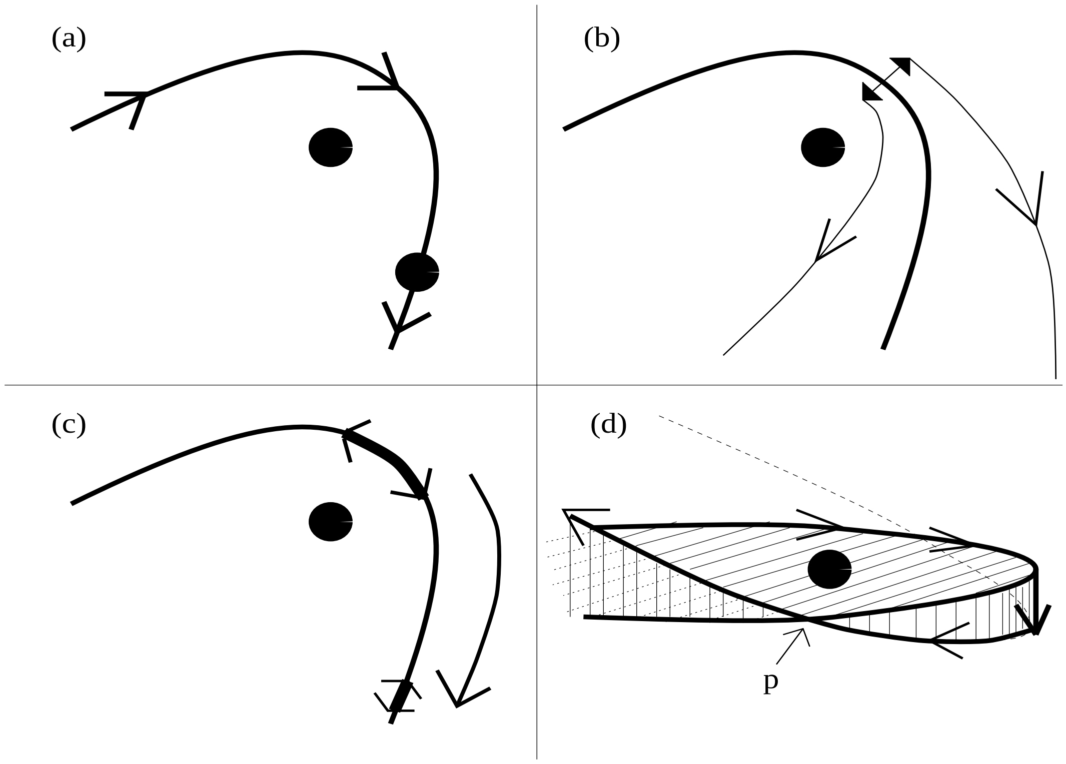

Figure 4.0: Schematic diagrams of which directions may be stable and unstable in a close encounter of a moving particle and a fixed particle. All perturbations happen at the point of closest approach. (a) Schematic diagram of a close encounter, with the plane of the orbit in the plane of the paper. (b) If the particle's position is perturbed off the orbit, but still in the plane of the orbit, the particle will diverge from it's original orbit, although not exponentially. Thus an unstable direction at the point of closest approach may be along the line joining the particles. (c) If the particle's position is perturbed along the orbit, but not off it, the perturbation is damped as the particle's speed slows down on the outbound path. Thus, at the point of closest approach, a stable direction may lie along the orbit. (d) A perturbation out of the plane of the orbit, as seen nearly edge on; the plane of the orbit becomes tilted. We see that initially the orbits converge, then cross at point **p**, then diverge from one another. We thus see that whether a direction is "stable" or "unstable" may depend on the duration that the variational equations are integrated. A resolvent from the point of closest approach to **p** may label as "stable" this perturbation direction; a resolvent that goes beyond **p** may label this direction as "unstable".

of the particles may not be affected by this for a long time. If the number of particles that undergo local glitches increases only slowly, then the simulation still has a reasonable amount of validity until some appreciable fraction of the particles have diverged from their local shadows. The fact that the high-dimensional phase-space orbit, as a whole, diverges from the global shadow at the point of the first local glitch may be too stringent an error criterion for large N-body shadowing. This also leads to the question of what I call *local glitch propagation*: how does the local glitch of one particle effect the others? Clearly, a particle that diverges from its shadow will start to effect nearby particles, eventually participating in different collisions than would occur in any shadow. How fast do these effects propagate throughout the system? Does the number of new local glitches introduced during a small time interval depend on the number of local glitches already present (leading to an exponential growth via the familiar first-order ODE $y' = \lambda y$), or do most local glitches arise independently of one another (giving a linear growth rate $y' = c$)? Can the two be combined as simply as $y' = c_0 y + c_1$? If the system is softened, what does the divergence of one particle's noisy orbit from all possible shadows imply about the validity of the distribution of particles? If the distribution is still valid, then the simulation is still valid under error measure 4 at the end of chapter 0, even if no shadow exists.

Practically speaking, if one is using a refinement algorithm to shadow the system, the question arises of *how* a high-dimensional phase space trajectory should be shadowed when some of its dimensions have undergone local glitches. Those dimensions are now invalid in some sense. Should they simply be ignored for the purposes of the computation of 1-step errors? They cannot be ignored completely, or else the shadowing model will not model glitch propagation, if present. This leads naturally to the idea of what I call *Fixed Motion Shadowing*.

4.0.3 Fixed Motion Shadowing

Shadowing of particular particles in large fixed-motion N-body systems, or simply *Fixed Motion Shadowing*, is more general than the form of shadowing done by Quinlan and Tremaine (QT) [29]. In their system, one particle moves amongst many particles whose *positions* are fixed. A more realistic system may be to run a standard large N-body simulation with all N particles moving for some large N, and then fix the *motion* of all particles in the simulation except one. This one particle's orbit is then refined in an attempt to decrease its 1-step errors under the influence of the gravitational forces of all the other particles, whose motions remain fixed to their orbits in the original noisy simulation. These particles are said to have *fixed motion*, while the individual particle that is shadowed is said to have *moving motion*, and a shadow so constructed for this particle is called a *fixed motion shadow*. Note that we would need to use interpolation to produce the positions of the fixed motion particles between their noisy timesteps.

The meaning of this shadowing technique is as follows. Let the "moving motion" particle be called m. Let the computed global noisy orbit be C. Observe that, if m's 1-step errors can be reduced to zero in a fixed-motion system, then m's local noisy orbit C_m is close to an exact orbit under the influence of the *noisy* positions of the other particles. In other words, C_m is close to a *true* trajectory F that feels its forces from the *noisy* positions of all the fixed-motion particles — their motions in C. F is the fixed motion shadow.[0]

[0]Think of it like this: F is produced by perturbing m's orbit to have small 1-step errors under the influence of some arbitrary, mysterious, external forces produced from a potential that is a function only of time. This potential

This scheme certainly seems more realistic than a QT-like system. The existence of a fixed motion shadow seems consistent with the existence of a global shadow, for if a global shadow exists then each particle's local noisy orbit, and in particular that of the moving motion one, must closely follow its local shadow, and the fixed motion shadow will probably follow the local shadow more closely than does the local noisy orbit.

Does the existence of a fixed motion shadow F for m imply the existence of a global shadow G of comparable length? Probably not in general, because any local glitch of a particle other than m will go undetected, since that particle's motion is fixed. A fixed motion shadow probably provides a statistic similar to the "average shadow length for an individual particle" in conjecture 1 from chapter 3 (p. 42). This in turn may provide an estimate of the "how many particles are still shadowable" error measure discussed in the previous section.

On the other hand, if a global shadow does exist, and if a fixed motion shadow follows the local shadow more closely than does the local noisy orbit, then fixed motion shadowing may provide an efficient method to refine the orbit of a high-dimensional trajectory, by allowing refinement of each local orbit individually in time $O(D^3)$ where D is the number of phase space dimensions per particle (six for N-body systems). Thus numerical shadowing of the entire phase space would take time $O(D^3M)$ per refinement rather than $O((DM)^3)$. The question now is, how many refinements will be needed? This method is analogous to a Gauss-Seidel method for the solution of the matrix equation $A\mathbf{x} = \mathbf{b}$. There may be interplay between local orbits so that refinement of orbit j changes orbit $i < j$ in a way inconsistent with the local correction that was applied to orbit i, even though at the time orbit i was refined, the equation used was valid.[1] Of course, this will not help if the number of refinements required increases, for example, as $O(M^2)$.

If there is no glitch propagation, then each particle encounters local glitches independent of all the others, and so the existence of a fixed motion shadow implies nothing about the existence of a global shadow. I expect that glitch propagation has significant influence in collisional systems, although it probably has little influence in collisionless systems. A collisionless medium controls the motion of its constituent particles via global potential effects, not local collisional effects. So in a collisionless system, perhaps it is possible for other particles to have different motions, but in the same general distribution, and still result in the chosen particle m having the same motion. For collisionless systems, we are led to error measure 4 from Chapter 0.

Measuring Glitch propagation

To attempt the measurement of glitch propagation effects, we could allow M individual particles to have moving motion amongst N fixed-motion particles. Then, when a moving motion particle's trajectory encounters a local glitch, we would note the time, and then transfer that particle to the fixed motion group, giving it its original noisy motion, leaving $M-1$ particles with moving motion. This would continue until all particles have been transferred to the fixed motion group. We could

comes from the other particles, each of whom moves as if it was sliding along a rigid, massless wire winding through space. The motion of each particle along its wire — its motion in C — is a function of time, and only time.

[1] Note that the measure of 1-step error, and thus the measure of a quality of any refinement procedure, is still the full phase-space 1-step error, which is cheap to compute. In this case, the savings per refinement is that we need to compute M $D \times D$ resolvents per timestep, rather than one $MD \times MD$ resolvent.

then try the same experiment, except rather than transfering glitched particles to the fixed motion group with their original noisy motion, we could transfer them to the fixed motion group with the new motion computed from the more accurate trajectory at the time of the local glitch. This may help us compare glitch propagation effects between the noisy trajectory, and a trajectory with the glitched particles "corrected", which we assume is closer to a global shadow of the system.

If we can discover more rigourously what kinds of particle motion cause local glitches (one criterion seems to be close encounters, or large, abrupt changes in energy), then we have a criterion for choosing which particles should belong to the moving motion group. For, if we can successfully fixed-motion shadow the particles that we suspect have the highest probability of undergoing local glitches, then we can be even more confident that a global shadow exists. This leads to the following conjecture:

Conjecture 2 *If 1 particle (or M particles) can be fixed-motion shadowed for time T in a collisional system, and no other particles satisfy glitch conditions, then a global shadow exists whose length is comparable to T.*

More generally, if we can show that for some subset G of all phase-space co-ordinates P, the existence of a shadow of $G \Rightarrow$ the existence of a shadow of P, then we need only attempt shadowing of G. This is an extremely interesting area for further work in the general area of shadowing.

Fixed-motion shadowing could be tested against full-trajectory shadowing for at least models like those in the previous chapter in which 25 particles move. We would attempt fixed motion shadowing on 1 of the 25 particles, and see how the length of the local shadow compares with the computed global shadow.

Finally, the entire issue of fixed-motion shadowing may be moot if figure 3.0 is a correct indication of how shadowing scales with increasing dimensions: if shadowing is as difficult in high dimensional systems as figure 3.0 suggests, then no amount of fiddling with high dimensional kluges, like fixed motion shadowing, will be of any use.

4.0.4 Robustness of shadowing to small perturbations

If shadows can be shown to exist in large collisional N-body systems, it may allow them to be simulated with simpler algorithms that ignore small perturbations that until now were thought to be important to include in models of N-body systems.

For example, Aarseth's [1] N-body integrator is a popular one for collisional systems. It includes regularization, in which close encounters between 2 particles are solved using the analytical 2-body solution with perturbations from the other particles. If a shadow can be shown to exist for such a numerical solution, it may also be possible to show that a shadow would exist if outside perturbations were ignored during 2-body close encounters. However, it's possible this may not be more efficient in the end, because the interval over which the two-body solution is valid may be smaller. Aarseth's integrator also uses individual timesteps for each particle, and interpolation of positions for particles with large timesteps when computing their forces on particles with smaller timesteps. Perhaps discrete positions could be used for particles with large timesteps, rather than their interpolated positions, and still produce shadowable solutions.

Another example of a system with perturbations is the Barnes and Hut [3] $O(N \log N)$ force computation algorithm. This algorithm produces a force function that is discontinuous in both space and time, thus introducing artificial, relatively large perturbations; often the force function for each particle is computed only to about one part in 10^3 or 10^4 [5]. Furthermore these kicks have been shown *not* to be random, but instead have a high correlation [5]. To test whether such a system is shadowable, we would need many moving particles (preferably thousands), and then we should try shadowing it using the $O(N^2)$ force computation algorithm. There has already been work to show that energy distributions are preserved between the Barnes-Hut and $O(N^2)$ algorithms [15], but no shadowing was attempted.

More generally, we would like to study the shadowability of solution methods that introduce arbitrary "kicks" of various magnitudes. Roundoff and truncation error are the only kinds of kicks that have been studied in shadowing of N-body systems thus far. As seen above, there are many other types, and it is not clear which, if any, affect the robustness of shadowing results.

4.0.5 Noisy integrator issues

Thus far, not much has been said about the noisy integrator, other than that it has less accuracy than the "accurate" integrator. Eventually, shadowing should be attempted while using the same noisy integrator that astronomers commonly use — often leapfrog for collisionless systems, and Aarseth's [1] for collisional systems, and usually with individual timesteps for each particle. If it turns out that long shadows do not exist for the integrators already in common use, but shadows do exist for other noisy integrators, it may be necessary for shadowing researchers to dictate to simulation researchers what integrator should be used, and at what accuracy. Comparisons should be made between symplectic *vs.* not and time-symmetric *vs.* not. In the end, the question that shadowing addresses is, *How badly are we allowed to integrate?*

Also, it should be noted that the leapfrog integrator should not be expected to work on non-softened collisional problems. It is an integrator that is more suited to smooth potentials.

4.0.6 Other systems to shadow

Since time-centred leapfrog with constant timestep is a symplectic integrator, then when run with a suitably small timestep, it gives the exact solution to *some* Hamiltonian system, although we do not know exactly what system it is.[2] Therefore, if we were to choose some timestep h, then the solution we get can be considered to be a noisy solution to the exact Hamiltonian system that is solved with the same equations integrated with a timestep of $h/2$. Thus, we may be able to learn something about Hamiltonian shadowing in general by studying a simple system with the leapfrog integrator, or any symplectic integrator.

[2]This idea, and the next, on related areas, are based on ideas from Scott Tremaine

4.0.7 Related areas "inspired" by shadowing

Refinement is an iterative process that attempts to find *a* true solution close to a computed solution; it is not known in advance if such a true solution exists or what its initial conditions will be.

What if we have a specific set of initial conditions for which we want to compute *the* true solution, assuming we don't mind that the solution method may take much more computational effort than traditional solution methods, and assuming we know a solution exists and satisfies certain conditions? Is it possible to develop an iterative method, analogous to refinement in a shadowing context, that can reduce the 1-step errors towards a *particular* solution? Waveform relaxation and defect correction methods both generate better and better approximations to a solution of an initial value problem, but can they be applied to chaotic systems? Could such iterative methods take advantage of knowledge of the stable and unstable subspaces by focusing effort onto the unstable subspace in forward time and the stable subspace in reverse time? Or possibly by using the unstable and stable vectors as *basis* vectors for all computation, rather than the standard Euclidean orthonormal system? If so, such a solution method would be valuable, for example, to long-term solar system researchers.[3]

4.0.8 Reliability issues less stringent than shadowing

If it turns out that shadowing is too stringent an error measure for large N-body systems, we may need to resort to error measure 4 from Chapter 0.

To show that the distribution of particles evolves independent of the paths of individual particles, and evolves similarly for equivalent input distributions, we could attempt many simulations whose initial conditions are all drawn randomly from the same initial distribution. For example, we could start with a distribution that "looks" like a specific spiral galaxy, and see if a low-resolution, smoothed animation looks the same regardless of the initial positions of the particles, as long as the initial distribution looks the same at low resolution.

If this fails, then we can try the same experiment using error measure 5 from Chapter 0. If this fails, then I don't know how to measure the reliability of N-body simulations.

[3]Note, however, that the practicality of attempting to integrate the solar system back several billion years seems dubious. For, if it is already known that the system is chaotic, then there are an almost unlimited number of small perturbations that we cannot know about — small undiscovered comets, or comets that passed through our system once from deep space billions of years ago, which we cannot possibly account for; or stars in our galaxy that perturbed the planets over the past few billion years.

4.1 Further work on shadowing in general

4.1.0 Further work on the GHYS/QT algorithm

Here are some further ideas that I have not yet thought about in depth but should be considered. Some have already been mentioned briefly in previous chapters, but are collected here for convenience.

- An obvious item that has been overlooked is the course grained parallelism inherent in the computation of the resolvents: each resolvent is computed independently of all the others. Thus, each could be computed on a separate processer, for example using the PVM (*P*arallel *V*irtual *M*achinne) software package [9].
- When using constant RUS, and iterations are failing, is there some way to decide if the problem is being caused by one (or a small number) of RUS_i's? If so, then only these RUS_i's need be recomputed, rather than the RUS for the entire trajectory. This is potentially a large savings. An obvious possibility is that the shadow step with the greatest 1-step error (or better, the one that has the least improvement) is one that needs its RUS_i recomputed. Or, perhaps a shadow step with a 1-step error that refuses to decrease is too long, making the resolvent invalid. Here, the shadow step could be split into smaller segments.
- There has been much work done in numerical analysis in the area of two-point boundary value problems. Shadowing trajectories have two ends, and each end has a boundary condition: at the last point, the growth of the expanding direction is pinched; the growth of the contracting direction (expanding in negative time) is pinched at the starting point. Thus there are two points, and boundary conditions at each end. The work on two-point boundary value problems may have bearing on shadowing.
- Thus far, perturbations are allowed only in the phase-space co-ordinates of a trajectory. It seems reasonable to also allow perturbations in *time*. Would it help if perturbations in time were allowed as well? In other words, if the time t_i at which the phase space point $\mathbf{p}_i$ is measured is allowed to be perturbed by small amounts, will shadowing be any more successful?
- It has been repeatedly stated that the GHYS refinement procedure is similar to a Newton's method. It has many similarities: the resolvent is a Jacobian; the method is iterative; when the Jacobian is recomputed at each step, there is geometric converge; and finally, it shows some of the same weaknesses as Newton's method. Is it possible that the GHYS procedure *is* a Newton's method in disguise, with boundary conditions? If it can be arranged to look exactly like a Newton's method, then the massive amount of literature and code devoted to Newton's method may be brought to bear on the problem.

The Newton's method item can be formalized a bit more. In general, a one-dimensional Newton's method trying to find a zero of the equation $y = f(x)$, given an initial guess x_0, is usually written as $x_{k+1} = x_k - f(x_k)/f'(x_k)$. In the refinement problem, the function being computed and for which we are trying to find a zero is the function that computes the 1-step errors along the entire trajectory. Let the entire trajectory be

$$\mathbf{P} = \begin{pmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_S \end{pmatrix},$$

Recall the equation for the 1-step errors is

$$\mathbf{e}_{i+1} = \mathbf{p}_{i+1} - f_+(\mathbf{p}_i)$$

where f_+ is the function that maps point $\mathbf{p}_i$ at time t_i to the true solution at time t_{i+1} by integrating the solution of the ODE forwards in time. In the formulation I have in mind, we also need the backward errors as defined in the SLES refinement algorithm:

$$\mathbf{b}_{i-1} = \mathbf{p}_{i-1} - f_-(\mathbf{p}_i)$$

where f_- integrates the ODE backwards in time. Then, I define a new type of 1-step error, called the *total 1-step error* $\mathbf{a}_i$ as the sum of the forward and backwards 1-step errors:

$$\mathbf{a}_i = \mathbf{e}_i + \mathbf{b}_i = 2\mathbf{p}_i - f_-(\mathbf{p}_{i+1}) - f_+(\mathbf{p}_{i-1}),$$

with $\mathbf{e}_0 = \mathbf{b}_S = 0$. Let the function that computes all the 1-step errors be

$$\mathbf{E}(\mathbf{P}) = \begin{pmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_S \end{pmatrix}.$$

Let $\mathbf{P}^0$ be the initial noisy orbit; $\mathbf{P}^k$ will represent the k iteration of the Newton's method. Then the Newton's method for the GHYS refinement procedure may be written as

$$\mathbf{P}^{k+1} = \mathbf{P}^k - J_{\mathbf{E}}^{-1}(\mathbf{P}^k)\mathbf{E}(\mathbf{P}^k)$$

where $J_{\mathbf{E}} = \mathbf{E}'$ is the Jacobian of the 1-step error function $\mathbf{E}$, *i.e.*, the derivative of the 1-step errors with respect to the phase-space orbit. Writing out $J_{\mathbf{E}}$ explicitly,

$$J_{\mathbf{E}}(\mathbf{P}^k) = \begin{pmatrix} \frac{\partial \mathbf{a}_0}{\partial \mathbf{p}_0} & \frac{\partial \mathbf{a}_0}{\partial \mathbf{p}_1} & \cdots & \frac{\partial \mathbf{a}_0}{\partial \mathbf{p}_S} \\ \frac{\partial \mathbf{a}_1}{\partial \mathbf{p}_0} & \frac{\partial \mathbf{a}_1}{\partial \mathbf{p}_1} & \cdots & \frac{\partial \mathbf{a}_1}{\partial \mathbf{p}_S} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{a}_S}{\partial \mathbf{p}_0} & \frac{\partial \mathbf{a}_S}{\partial \mathbf{p}_1} & \cdots & \frac{\partial \mathbf{a}_S}{\partial \mathbf{p}_S} \end{pmatrix},$$

Taking partial derivatives, we see that

$$\frac{\partial \mathbf{a}_i}{\partial \mathbf{p}_j} = \begin{cases} 2I, & (I \text{ is the identity matrix}), \text{ along the diagonal where } i = j. \\ -f'_+(\mathbf{p}_{i-1}) & \text{along the lower diagonal where } j = i-1 \\ -f'_-(\mathbf{p}_{i+1}) & \text{along the upper diagonal where } j = i+1 \\ 0, & \text{otherwise} \end{cases}$$

Finally, note that $f'_+(\mathbf{p}_i) = L_i$, and $f'_-(\mathbf{p}_i) = L_{i-1}^{-1}$, the resolvents that we already compute.

Somewhere here is where the boundary conditions will need to come into play: if the number of phase space dimensions is $2D$ (each of the $\mathbf{p}_i$ and $\mathbf{a}_i$ vectors has $2D$ dimensions), then there are D boundary conditions at each end of the trajectory, limiting the growth of the stable and unstable components at their respective endpoints. The corrections (computed from $J^{-1}\mathbf{E}$) will probably also need to be computed in a special order, as they are in the GHYS procedure. I do not currently know how to include these boundary conditions in the problem, but assuming they can be, the Newton iteration may look like the scheme described above.

4.1.1 More rigourous algorithms

Containment

Recall that GHYS intended refinement simply as a method to reduce noise in a trajectory, to give the more rigourous process of *containment* a better chance to establish rigourous proof of the existence of a true shadow. Clearly, one avenue of research is to generalize containment to work on arbitrary Hamiltonian systems. It may even prove to be about as efficient, practically speaking, as refining to machine epsilon; and most important, it is *rigourous.*

Uniqueness of the shadow

One interesting question, although it is not crucial to the proof of existence of shadows, is the question of uniqueness of shadows. It is already known that if one shadow exists, then infinitely many of them exist, all packed into a small volume of phase space. However, does choosing boundary conditions for c_{s_0} and c_{u_S} give a unique true shadow from among the infinite number of true shadows that exist, if one exists at all? It seems to me that fixing the conditions probably produces a unique shadow, because the number of boundary conditions is exactly equal to the number of degrees of freedom. For example, fixing the position and velocity at any given time produces a unique true solution, so it seems reasonable to expect that fixing half the co-ordinates at one time a and half at another time b would produce a unique solution (if one exists at all), as long as the image at time b of the initial-condition-subspace at time a is linearly independent from the final-condition-subspace at time b. It could also be tested numerically by trying slightly different first guesses for the shadow before refinement starts, or possibly by using a different "accurate" integrator to find the shadow. However, there may exist pathological boundary cases for which the solution is not unique.

Appendix A

Review of the astrophysical N-body problem

A.0 Basic N-body Equations

The general astrophysical N-body system consists of N particles moving according to Newton's three laws of motion, with Newton's familiar gravitational law $F = Gm_1m_2/r^2$ being the only source of force.

Let $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$ be the unit vectors of the standard Cartesian (x, y, z) system. Let $\mathbf{r}_i = x_i\hat{\mathbf{i}} + y_i\hat{\mathbf{j}} + z_i\hat{\mathbf{k}}$ be the position vector of particle i. Let $\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i$ be the vector pointing from particle i to particle j, *i.e.*, the position of particle j with respect to particle i. Thus

$$\mathbf{r}_{ij} = (x_j - x_i)\hat{\mathbf{i}} + (y_j - y_i)\hat{\mathbf{j}} + (z_j - z_i)\hat{\mathbf{k}}$$

The force that particle j exerts on particle i is

$$\begin{aligned} \mathbf{F}_{ij} &= \frac{Gm_im_j}{r_{ij}^2}\bar{\mathbf{r}}_{ij} \\ &= \frac{Gm_im_j}{|\mathbf{r}_{ij}|^3}\mathbf{r}_{ij} \\ &= -\mathbf{F}_{ji} \end{aligned}$$

where G is the gravitational constant, m_i, m_j are the masses of the particles, $\mathbf{r}_{ij}$ is the vector pointing from particle i to particle j, r_{ij} is the magnitude of this vector, and $\bar{\mathbf{r}}_{ij}$ is the unit vector pointing in the direction of $\mathbf{r}_{ij}$. Let the total force on particle i be $\mathbf{F}_i$. It is the sum of all the forces from all the other particles. Thus the total force $\mathbf{F}_i$ on particle i is

$$\begin{aligned} \mathbf{F}_i &= \sum_{j \neq i} \mathbf{F}_{ij} = \sum_{j \neq i} Gm_im_j\frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^3} \\ &= \sum_{j \neq i} Gm_im_j\frac{(x_j - x_i)\hat{\mathbf{i}} + (y_j - y_i)\hat{\mathbf{j}} + (z_j - z_i)\hat{\mathbf{k}}}{[(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2]^{\frac{3}{2}}}. \end{aligned}$$

For a 3-dimensional space, this is a set of $3N$ second-order ordinary differential equations (ODEs), which we translate into a set of $6N$ first-order ODEs by letting the velocity $\mathbf{v} = \frac{\partial}{\partial t}\mathbf{r} \equiv \dot{\mathbf{r}}$ and then building the phase-space vector of the entire system as $\mathbf{p} \equiv \begin{pmatrix} \mathbf{r} \\ \mathbf{v} \end{pmatrix}$. Thus the first-order ODE system is

$$\dot{\mathbf{p}} \equiv \frac{\partial}{\partial t}\mathbf{p} = \frac{\partial}{\partial t}\begin{pmatrix} \mathbf{r} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ \mathbf{a} \end{pmatrix}$$

where $\mathbf{a}$ is the vector representing the accelerations of all the particles.

At any given time, the position $\mathbf{r}$ and velocity $\mathbf{v}$ of every particle is known. Thus $\dot{\mathbf{r}}$ is simply $\mathbf{v}$, and $\dot{\mathbf{v}}$ is the set of time-derivatives of the velocity $\mathbf{v}_i$ of each particle, where

$$\dot{\mathbf{v}}_i = \mathbf{a}_i = \frac{\mathbf{F}_i}{m_i} = \sum_{j \neq i} G m_j \frac{(x_j - x_i)\hat{\mathbf{i}} + (y_j - y_i)\hat{\mathbf{j}} + (z_j - z_i)\hat{\mathbf{k}}}{[(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2]^{\frac{3}{2}}}. \tag{A.0}$$

If force softening is used, the denominator instead becomes $[(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2 + \varepsilon_s^2]^{\frac{3}{2}}$ where ε_s is the softening parameter.

A.1 Jacobian of an $N - M$ fixed-particle system

Let the function computing the derivative of $\mathbf{p} = \begin{pmatrix} \mathbf{r} \\ \mathbf{v} \end{pmatrix}$ be $\dot{\mathbf{p}} = \begin{pmatrix} \mathbf{v} \\ \mathbf{a} \end{pmatrix} = f(\mathbf{p})$. f has no dependence on time. The Jacobian of f, $\frac{\partial f}{\partial \mathbf{p}}$ is

$$\frac{\partial f}{\partial \mathbf{p}} = \begin{pmatrix} \frac{\partial \mathbf{v}}{\partial \mathbf{r}} & \frac{\partial \mathbf{v}}{\partial \mathbf{v}} \\ \frac{\partial \mathbf{a}}{\partial \mathbf{r}} & \frac{\partial \mathbf{a}}{\partial \mathbf{v}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & I \\ \frac{\partial \mathbf{a}}{\partial \mathbf{r}} & \mathbf{0} \end{pmatrix}$$

where $\mathbf{0}$ is the $3N$-square zero matrix, and I is the $3N$ identity matrix. If all particles move, then the entire Jacobian is $6N$ square. If only M particles move, then there is no meaning to the entries in the Jacobian owned by particles that do not move, and so the Jacobian is only $6M$ square. The only difficult part of this Jacobian is $\frac{\partial \mathbf{a}}{\partial \mathbf{r}}$, which is a $3N$-square matrix, and $\mathbf{a} = (\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_N)^T, \mathbf{r} = (\mathbf{r}_1\ \mathbf{r}_2\ \ldots\ \mathbf{r}_N)^T$ where $\mathbf{a}_i = (a_{ix}\ a_{iy}\ a_{iz})^T$ is the acceleration of particle i, given by equation A.0, and $\mathbf{r}_i = (r_{ix}\ r_{iy}\ r_{iz})^T \equiv (x_i\ y_i\ z_i)^T$ is its position in 3-space. Thus,

$$\frac{\partial \mathbf{a}}{\partial \mathbf{r}} = \begin{pmatrix} \frac{\partial \mathbf{a}_1}{\partial \mathbf{r}_1} & \frac{\partial \mathbf{a}_1}{\partial \mathbf{r}_2} & \cdots & \frac{\partial \mathbf{a}_1}{\partial \mathbf{r}_N} \\ \frac{\partial \mathbf{a}_2}{\partial \mathbf{r}_1} & \frac{\partial \mathbf{a}_2}{\partial \mathbf{r}_2} & \cdots & \frac{\partial \mathbf{a}_2}{\partial \mathbf{r}_N} \\ \vdots & \vdots & \ddots & \cdots \\ \frac{\partial \mathbf{a}_N}{\partial \mathbf{r}_1} & \frac{\partial \mathbf{a}_N}{\partial \mathbf{r}_2} & \cdots & \frac{\partial \mathbf{a}_N}{\partial \mathbf{r}_N} \end{pmatrix}, \text{ where } \frac{\partial \mathbf{a}_i}{\partial \mathbf{r}_j} = \begin{pmatrix} \frac{\partial a_{ix}}{\partial x_j} & \frac{\partial a_{ix}}{\partial y_j} & \frac{\partial a_{ix}}{\partial z_j} \\ \frac{\partial a_{iy}}{\partial x_j} & \frac{\partial a_{iy}}{\partial y_j} & \frac{\partial a_{iy}}{\partial z_j} \\ \frac{\partial a_{iz}}{\partial x_j} & \frac{\partial a_{iz}}{\partial y_j} & \frac{\partial a_{iz}}{\partial z_j} \end{pmatrix}$$

Finally, if we let α, β, γ represent one of $\{x, y, z\}$, it can be shown that

$$\frac{\partial a_{i\alpha}}{\partial \gamma_j} = \begin{cases} Gm_j(3\alpha_{ji}\gamma_{ij})/r_{ij}^5, & \text{if } \alpha \neq \gamma \text{ and } j \neq i \\ -\sum_{k\neq j} Gm_k(3\alpha_{ik}\gamma_{ki})/r_{ik}^5, & \text{if } \alpha \neq \gamma \text{ and } j = i \\ Gm_j(r_{ij}^2 - 3\alpha_{ij}^2)/r_{ij}^5, & \text{if } \alpha = \gamma \text{ and } j \neq i \\ \sum_{k\neq i} Gm_k(-r_{ik}^2 + 3\alpha_{ki}^2)/r_{ik}^5, & \text{if } \alpha = \gamma \text{ and } j = i \end{cases}$$

where, for example, α_{ij} is the α component of $\mathbf{r}_{ij}$. If softening is employed, substitute $(r_{ij}^2 + \varepsilon^2)$ for r_{ij}^2 and $(r_{ij}^2 + \varepsilon^2)^{5/2}$ for r_{ij}^5.

Bibliography

[1] Sverre J. Aarseth. Direct Methods for N-Body Simulations. In *Multiple Time Scales*, pages 377–418. Academic Press, Inc., 1985.

[2] D. V. Anosov. *Proc. Steklov Inst. Math*, 90:1, 1967.

[3] Josh Barnes and Piet Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324:446–449, 4 December 1986.

[4] Joshua Barnes, Jeremy Goodman, and Piet Hut. Dynamical instabilities in spherical stellar systems. *The Astrophysical Journal*, 300:112–131, 1986.

[5] Joshua E. Barnes and Piet Hut. Error analysis of a tree code. *Astrophysical Journal Supplement Series*, 70:389–417, 1989.

[6] James Binney and Scott Tremaine. *Galactic Dynamics*. Princeton Series in Astrophysics. Princeton University Press, 1987.

[7] R. Bowen. *J. Differential Equations*, 18:333, 1975.

[8] Silvina Dawson, Celso Grebogi, Tim Sauer, and James A. Yorke. Obstructions to Shadowing When a Lyapunov Exponent Fluctuates about Zero. *Physical Review Letters*, 73(14):1927–1930, 3 Oct 1994.

[9] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, and Vaidy Sunderam. PVM: Parallel Virtual Machine:A Users' Guide and Tutorial for Network Parallel Computing. Available via anonymous ftp from netlib2.cs.utk.edu:/pvm3/book/pvm-book.ps.

[10] Jeremy Goodman, Douglas C. Heggie, and Piet Hut. On the exponential instability of N-body systems. *The Astrophysical Journal*, 415:715–733, 1993.

[11] Celso Grebogi, Stephen M. Hammel, James A. Yorke, and Tim Sauer. Shadowing of Physical Trajectories in Chaotic Dynamics: Containment and Refinement. *Physical Review Letters*, 65(13):1527–1530, 24 September 1990.

[12] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulation. *Journal of Computational Physics*, 73:325, 1987.

[13] Ernst Hairer, Syvert Paul Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I – Nonstiff Problems*. Springer-Verlag, 1980.

[14] D. C. Heggie and R. D. Mathieu. Standardized units and time scales. In Piet Hut and S. L. W. McMillan, editors, *The Use of Supercomputers in Stellar Dynamics*, pages 233–235. Springer-Verlag, 1986.

[15] Lars Hernquist, Piet Hut, and Jun Makino. Discreteness noise versus force errors in N-body simulations. *Astrophysical Journal Letters*, 402:L85–L88, 1993.

[16] Alan C. Hindmarsh. LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM-SIGNUM Newsletter*, 15(4):10–11, 1980.

[17] J. Garrett Jernigan and David H. Porter. A tree code with logarithmic reduction of force terms, hierarchical regularization of all variables, and explicit accuracy controls. *Astrophysical Journal Supplement Series*, 71:871–893, 1989.

[18] David Kahaner, Cleve Moler, and Stephen Nash. *Numerical Methods and Software*. Prentice-Hall series in Computational Mathematics. Prentice-Hall, 1989.

[19] Henry E. Kandrup and Haywood Smith. On the sensitivity of the N-body problem to small changes in initial conditions. *The Astrophysical Journal*, 374:255–265, 1991.

[20] Henry E. Kandrup and Haywood Smith. On the sensitivity of the N-body problem to small changes in initial conditions. III. *The Astrophysical Journal*, 399:627–633, 1992.

[21] Henry E. Kandrup and Haywood Smith. On the sensitivity of the N-body problem to small changes in initial conditions. II. *The Astrophysical Journal*, 386:635–645, 1992.

[22] M. Lecar. *Bull. Astron.*, 3:91, 1968.

[23] Thomas A. McGlynn. Dissipational collapse of galaxies and initial conditions. *The Astrophysical Journal*, 281:13–30, 1984.

[24] David Merritt and Luis A. Aguilar. A numerical study of the stability of spherical galaxies. *Monthly Notices of the Royal Astronomical Society*, 217:787–804, 1985.

[25] Dimitri Mihalas and James Binney. *Galactic Astronomy — Structure and Kinematics*. Princeton Series in Astrophysics. Freeman, 1981.

[26] R. H. Miller. *The Astrophysical Journal*, 140:250, 1964.

[27] D. Pfenniger and D. Friedli. Computational issues connect with 3D N-body simulations. *Astronomy and Astrophysics*, 270:561–572, 1993.

[28] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[29] Gerald D. Quinlan and Scott Tremaine. On the reliability of gravitational N-body integrations. *Monthly Notices of the Royal Astronomical Society*, 259:505–518, 1992.

[30] J. A. Sellwood. The global stability of our Galaxy. *Monthly Notices of the Royal Astronomical Society*, 217:127–148, 1985.

[31] J. A. Sellwood. The art of N-body building. *Annual Review of Astronomy and Astrophysics*, 25:151–86, 1987.

[32] Jaswinder Pal Singh, John L. Hennessy, and Anoop Gupta. Implications of Hierarchical N-body Methods for Multiprocessor Architecture. Technical Report CSL-TR-92-506, Computer Systems Lab, Stanford University, Stanford, CA 94305, 1992.

[33] Simon D. White. Simulations of sinking galaxies. *The Astrophysical Journal*, 274:53–61, 1983.