A Fast Shadowing Algorithm for High Dimensional ODE Systems

WAYNE HAYES
wayne@cs.utoronto.ca

KENNETH R. JACKSON
krj@cs.utoronto.ca

Computer Science Department, University of Toronto
Toronto, Ontario, Canada M5S 3G4.
20 Mar 1996

Submitted to a special volume of *Annals of Numerical Mathematics* devoted to the *Dynamical Numerical Analysis* conference in Atlanta Georgia, 15-17 Dec 1995.

Abstract

A true trajectory of a chaotic dynamical system lying near an approximate trajectory is called a *shadow*. Finding shadows of numerical trajectories of ODE systems is very compute intensive, and until recently it has been infeasible to study shadows of higher-dimensional systems. We study the shadowing algorithm introduced by Grebogi, Hammel, Yorke and Sauer in 1990 and extended to arbitrary Hamiltonian systems by Quinlan and Tremaine in 1992, and introduce several major optimizations resulting in speedups of over 10^2. This algorithm is used to shadow gravitational N-body systems with up to 150 phase space dimensions.

1 Introduction

Computer simulation is a popular tool in the modern physical scientist's study of complex dynamical systems. Such systems often display sensitive dependence on initial conditions: small changes in initial conditions produce solutions that exponentially diverge from each other. Since a numerical solution introduces small perturbations like roundoff and truncation error, it will generally diverge exponentially away from the true solution having the same initial conditions.

Although there exist other methods of global error analysis, shadowing is unique in that it demands that the equation governing the ODE system remain fixed, while allowing perturbations to the initial conditions in an effort to find a nearby true solution that remains close to the computed solution. The measures of error are the distance between the numerical trajectory and the shadow, and the length of time the two remain near to each other. This is different from the more common form of backward error analysis for ODEs, in which it is shown that the numerical solution lies along the exact solution of an ODE system whose governing equation has been slightly perturbed. Although the latter approach is useful in some contexts, it is not clear that changing the governing equation is always the best way to measure error, especially if the governing equation satisfies special conditions that, if violated, could produce qualitatively different solutions. For example, a Hamiltonian system may not remain Hamiltonian if small, arbitrary, time-varying perturbations are allowed in its governing equation.

We start with some definitions. The terms *orbit*, *trajectory*, and *solution* are used interchangeably. For numerical discussions, we assume a machine precision of approximately 16 decimal digits, and a well-scaled problem where all macroscopic quantities of interest are of order unity. Throughout this paper, $|\cdot|$ denotes Euclidian norm, and $\|\cdot\|$ denotes the max norm, although any other norm should give similar results.

Definition. A *true trajectory* $\{\mathbf{x}_i\}_{i=a}^{b}$ of f satisfies $\mathbf{x}_{i+1} = f(\mathbf{x}_i)$ for $a \le i < b$.

We are interested in the case that 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 passing through $\mathbf{x}$, one timestep after $\mathbf{x}$.

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 $a \le i < b$, where δ_f is the noise amplitude.

For a map, δ_f is often the machine epsilon; for an ODE system that has position $\mathbf{p}_i$ at time t_i, it is the error per step.

Definition of shadowing. The true trajectory $\{\mathbf{x}_i\}_{i=a}^{b}$ $\delta_{\mathbf{x}}$-*shadows* $\{\mathbf{p}_i\}_{i=a}^{b}$ if $\|\mathbf{x}_i - \mathbf{p}_i\| < \delta_{\mathbf{x}}$ for $a \le i \le b$.

Definition. For $a \le i < b$, 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 difference $\mathbf{e}_{i+1} = \mathbf{p}_{i+1} - f(\mathbf{p}_i)$.

Thus, a true trajectory is one whose 1-step errors are identically zero, and a δ_f-pseudo-trajectory is one whose 1-step errors satisfy $\|\mathbf{e}_i\| < \delta_f$ for $a < i \le b$.

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 $\{\mathbf{p}_i\}_{i=a}^{G_0}$, but no true trajectory that $\delta_{\mathbf{x}}$-shadows $\{\mathbf{p}_i\}_{i=a}^{G}$, for $G > G_0$.

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. Refinement is a process, similar to Newton's Method, that iteratively takes a noisy orbit and tries to produce a nearby orbit with less noise, *i.e.,* one with smaller 1-step errors. A refinement iteration is *successful* if before the iteration the trajectory has noise δ_f^0, after the iteration it has noise δ_f^1, and

$$\delta_f^1 < \mu \delta_f^0 \text{ for some reasonable } \mu \in (0, 1). \tag{1}$$

Otherwise the refinement iteration is *unsuccessful.*

Shadowing was first discussed in relation to *hyperbolic* systems by Anosov [1] and Bowen [3]. In a 2-dimensional hyperbolic system, there are two special directions called the *unstable* (or *expanding*) and the *stable* (or *contracting*) directions, which may vary with time and generally are 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 rôles in backward time. In addition, the angle between the stable and unstable directions is uniformly bounded away from 0. In such a system, Anosov and Bowen proved that, for sufficiently small noise amplitude, shadows can exist for arbitrarily long times.

For non-hyperbolic systems, it seems that we must be satisfied with finite-length shadows. The first studies of shadows for non-hyperbolic systems appear to be by Beyn [2] and Hammel *et al.* [12]. More recently, there has been a flurry of activity in the study of numerical methods for finding shadows [5, 6, 7, 8, 17, 18, 9, 19, 16, 13]. In this paper, we concern ourselves with the refinement procedure of Quinlan and Tremaine [16], which is an extension of the two dimensional procedure of Grebogi *et al.* [10] to arbitrary-dimensional Hamiltonian systems. Hereinafter, these two papers will be referred to as QT and GHYS, respectively.

2 The refinement procedure of GHYS and QT

2.1 Outline

The refinement procedure of GHYS and QT is analagous 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, to our knowledge, this has not yet been shown. For pedagogical purposes, we 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 all 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 that used to compute $\mathbf{P}$. This describes a function, which we call g, which takes 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 in which the 1-step errors are identically zero, *i.e.,* a true orbit. This is an ideal situation in which to run Newton's method. If the method converges, then a true orbit has been found.

2.2 The GHYS refinement algorithm

Assume we have a noisy orbit $\mathbf{P} = \{\mathbf{p}_i\}_{i=0}^{S}$, and it has a shadow $\{\mathbf{x}_i\}_{i=0}^{S}$. Then

$$\mathbf{x}_{i+1} = f(\mathbf{x}_i) \text{ and } \mathbf{p}_{i+1} = \tilde{f}(\mathbf{p}_i),$$

where $\tilde{f}$ is an approximation to f with a typical accuracy or *noise* of δ_f. Here, δ_f should be the accuracy typically used by practitioners of the problem being studied. Now suppose we compute the 1-step errors

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

although numerically we use $\hat{f}$, a better approximation to f than $\tilde{f}$. Then $\hat{\mathbf{c}}_i \equiv \mathbf{x}_i - \mathbf{p}_i$ represents a correction term that perturbs point $\mathbf{p}_i$ towards the shadow. So,

$$\hat{\mathbf{c}}_{i+1} \equiv \mathbf{x}_{i+1} - \mathbf{p}_{i+1} = f(\mathbf{x}_i) - f(\mathbf{p}_i) - \mathbf{e}_{i+1} = \frac{\partial f(\mathbf{p}_i)}{\partial \mathbf{p}_i}\hat{\mathbf{c}}_i - \mathbf{e}_{i+1} + O(|\hat{\mathbf{c}}_i|^2) \tag{2}$$

In the spirit of Newton's method, we ignore the $O(|\hat{\mathbf{c}}_i|^2)$ in (2), and so one refinement iteration defines the corrections along the entire orbit:

$$\mathbf{c}_{i+1} = L_i\mathbf{c}_i - \mathbf{e}_{i+1}, \tag{3}$$

where $L_i \equiv \frac{\partial f(\mathbf{p}_i)}{\partial \mathbf{p}_i}$ is the linearized perturbation map. For a discrete 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 $i+1$.[1] It is the computation of the linear maps L_i, called *resolvents* for an ODE system,

[1] In other words, let

$$\dot{\mathbf{p}} = \mathbf{h}(t, \mathbf{p}) \tag{4}$$

that takes most of the CPU time during a refinement. If we let D be the dimension of the problem, then the resolvent has $O(D^2)$ terms in it, and it takes $O(D^3)$ time to compute. 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 variational equation integration is needed.

For simplicity of explanation, we assume $D = 2$ for the remainder of this subsection, deferring discussion of the higher D case to the next subsection. If the problem were not chaotic, the correction terms $\mathbf{c}_i$ could be computed directly from (3). But since L_i will amplify any errors in $\mathbf{c}_i$ not lying in the stable direction, computing the $\mathbf{c}_i$'s by iterating (3) forward will amplify errors and typically produce nothing but noise. Therefore, we split the error and correction terms into components in the stable ($\mathbf{s}_i$) and unstable ($\mathbf{u}_i$) directions at each timestep:

$$\mathbf{e}_i = e_{u_i}\mathbf{u}_i + e_{s_i}\mathbf{s}_i, \qquad \mathbf{c}_i = c_{u_i}\mathbf{u}_i + c_{s_i}\mathbf{s}_i. \tag{5}$$

Since it is not known *a priori* which direction is unstable at each timestep, the unstable vector $\mathbf{u}_0$ at time t_0 is initialized to an arbitrary unit vector. The linearized map is then iterated forward with

$$\bar{\mathbf{u}}_{i+1} = L_i\mathbf{u}_i, \qquad \mathbf{u}_{i+1} = \bar{\mathbf{u}}_{i+1}/|\bar{\mathbf{u}}_{i+1}|. \tag{6}$$

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 too close to the stable direction, then after a few e-folding times $\mathbf{u}_i$ will point roughly in the unstable direction at t_i. Similarly, the stable unit direction vectors $\mathbf{s}_i$ are computed by initializing $\mathbf{s}_S$ to an arbitrary unit vector and iterating backward,

$$\bar{\mathbf{s}}_i = L_i^{-1}\mathbf{s}_{i+1}, \qquad \mathbf{s}_i = \bar{\mathbf{s}}_i/|\bar{\mathbf{s}}_i|. \tag{7}$$

Substituting (5) into (3) yields

$$c_{u_{i+1}}\mathbf{u}_{i+1} + c_{s_{i+1}}\mathbf{s}_{i+1} = L_i(c_{u_i}\mathbf{u}_i + c_{s_i}\mathbf{s}_i) - (e_{u_{i+1}}\mathbf{u}_{i+1} + e_{s_{i+1}}\mathbf{s}_{i+1}). \tag{8}$$

While L_i magnifies errors in the unstable direction, it damps them in the stable direction. Likewise, L_i^{-1} damps errors in the unstable direction and magnifies errors in the stable direction. Thus the c_u terms should be computed backward, and the c_s terms forward. Taking components of (8) in the unstable direction at step $i+1$, iterate backward on

$$c_{u_i} = (c_{u_{i+1}} + e_{u_{i+1}})/|\bar{\mathbf{u}}_{i+1}|, \tag{9}$$

and taking components in the stable direction, iterate forward on

$$c_{s_{i+1}} = |L_i\mathbf{s}_i|c_{s_i} - e_{s_{i+1}}. \tag{10}$$

---

be the first-order ODE. Note that $\mathbf{p}_{i+1} = f(\mathbf{p}_i)$ is the solution of (4) 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 $\mathbf{h}(t,\mathbf{p})$. 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 describes 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 time t_{i+1}. That is, $R(t_{i+1}, t_i)$ is the solution of the *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 $R_2\delta\mathbf{p} = R_1R_0\delta\mathbf{p}$ [11]. Finally, the linear map in the GHYS refinement procedure is $L_i = R(t_{i+1}, t_i)$.

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 (10) damps initial conditions and (9) damps final conditions. GHYS and QT choose them both as 0, as do we for simplicity. 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.[2] 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 end point, and the backward-growing components at the initial point, to be small. That is, *boundary conditions* are being forced on the problem so that the exponential divergence is forcibly masked, if possible, making the solution of (3) numerically stable.

Note that, counter-intuitively, these boundary conditions demand that the initial condition for the shadow and noisy orbits 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 were 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 orbit.

2.3 QT's generalization to arbitrary Hamiltonian systems

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 B of stable and unstable directions is equal, although $B < D$ is possible.[3] At time t_i, let $\{\mathbf{u}_i^j\}_{j=1}^D$ represent D unstable unit vectors, and let $\{\mathbf{s}_i^j\}_{j=1}^D$ represent D stable unit vectors. For any particular timestep, it will be convenient if the unstable vectors are orthonormal to each other, and the stable vectors are orthonormal to each other. However, the stable and unstable vectors together will not in general form an orthogonal system.

The vectors are evolved as in the two dimensional case, except using Gram-Schmidt orthonormalization to produce two sets of D-orthonormal vectors at each timestep. Arbitrary orthonormal bases are chosen, $\mathbf{u}_0^j$ at time t_0, and $\mathbf{s}_S^j$ at time t_S, and then evolved according to

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

At each step i, two Gram-Schmidt orthonormalizations are done: 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, $\mathbf{u}_i^1$ points in the most unstable direction at step 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.*

[2] For any system, *even a chaotic one*, given any true orbit of fixed length, a small enough perturbation in the initial condition in any direction produces a small change in the final condition, although for chaotic systems this perturbation must be exponentially small in the length of the orbit. (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, there exist infinitely many true orbits nearby that also shadow it. However, it may be that all the true orbits are packed into a space unresolved by the machine precision.

[3] It is not hard to show that the Jacobian of the N-body problem has the form

$$J = \begin{pmatrix} \mathbf{0} & I \\ J_a & \mathbf{0} \end{pmatrix},$$

where $J_a = \frac{\partial \mathbf{a}}{\partial \mathbf{r}}$ is symmetrical, and has real but not necessarily positive eigenvalues. If μ is an eigenvalue of J_a, then $\pm\sqrt{\mu}$ are eigenvalues of J. If μ is positive, this gives one expanding and one contracting direction; if μ is negative, it gives directions that neither expand nor contract, but "rotate" in some sense.

The multidimensional generalization of (5) is the obvious

$$\mathbf{e}_i = \sum_{j=1}^{D}(e_{u_i^j}\mathbf{u}_i^j + e_{s_i^j}\mathbf{s}_i^j), \qquad \mathbf{c}_i = \sum_{j=1}^{D}(c_{u_i^j}\mathbf{u}_i^j + c_{s_i^j}\mathbf{s}_i^j). \tag{11}$$

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

From (3) and (11) the equation for the correction co-efficients in the unstable subspace at step $i+1$ 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 are projected 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 $j < k$. As stated previously, the the boundary condition requires the unstable component of the corrections to be small at timestep S, and their stable components to be small at timestep 0. For simplicity we take $\{c_{u_S^j} = 0\}_{j=1}^{D}$ as did QT, and the co-efficients are computed backward using

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

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

Again from (3) and (11), 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 are projected 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 step 0 is $\{c_{s_0^j} = 0\}_{j=1}^{D}$, and the co-efficients are computed forward using

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

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.*

2.4 Discussion of the GHYS/QT 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 with 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 orbit and the more accurate solution 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 the integration tolerance of the "accurate" integrator was accidentally set to only 10^{-8}. The shadowing routine proceeded to refine 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 only 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 if 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 [4]) 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 believe that convergence to a numerical shadow implies, with high probability, the existence of a true shadow of comparable length.

3 Optimizations to the GHYS/QT refinement algorithm

The model problem used in this research is the large-N gravitational N-body problem of astronomy. Since shadowing of excessively large N systems (say, $N \gg 100$) is still infeasible, it was necessary to devise a model that simulates large N systems without actually having high dimensionality. To this end, QT devised a model in which there were N "stars" in a cluster, but only M of them moved, while $N-M$ of them remained fixed. In 3-dimensional space, this gives a system with $6M$ degrees of freedom. Their shadowing experiments were primarily done with $N=100,\ M=1$, because the refinement procedure was too compute intensive to do extensive work for larger M.

3.1 Asymptotic run times

Most of the expense of refinement of an N-body orbit is in computing the resolvents L_i. See footnote 1 in section 2.2 for an explanation of L_i. Manually counting floating-point operations in the N-body 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 $N \gg M$. 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 $k \times S \times (108M^3 + 85MN)$. We found empirically that, using an Adams's integrator [15], k seems to be independent of M, and roughly equal to 500 for the shadow steps used in our N-body simulations. Figure 1 shows some experiments comparing the time to compute one resolvent to M, demonstrating the $O(M^3)$ relationship.

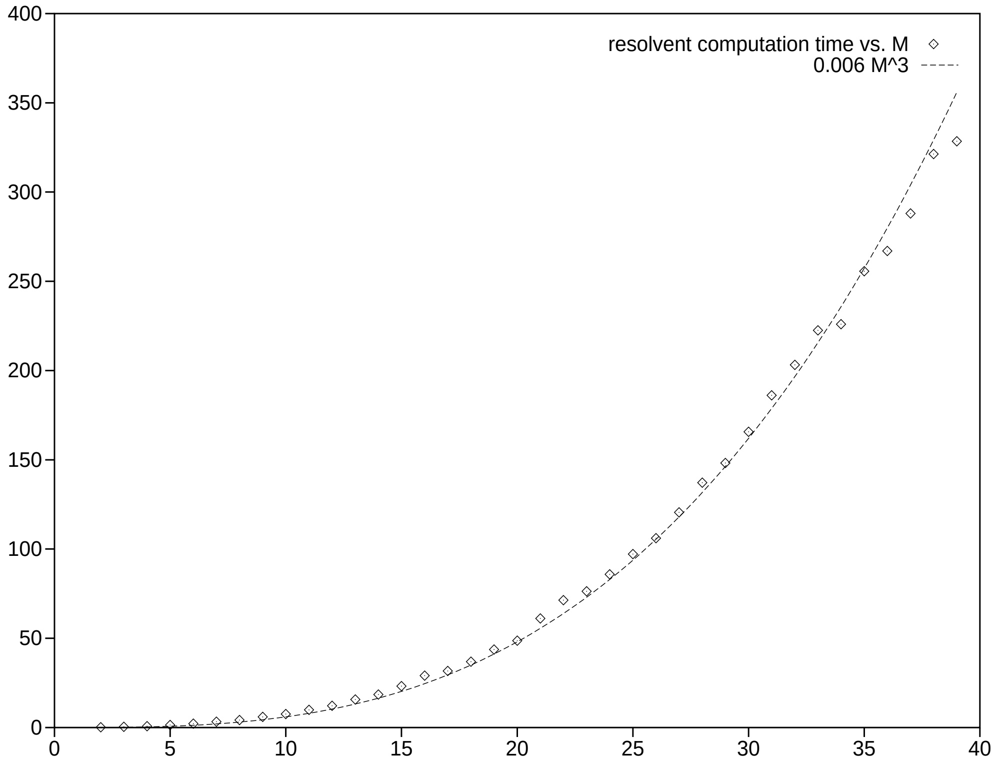

Figure 1: Time in seconds to compute one resolvent as a function of the number of moving particles M. For comparison, $0.006M^3$ is also plotted.

As will be seen in Table I below, a naive, un-optimized implementation of the high-dimensional QT algorithm with $M = 1$ (6 phase space dimensions), can take up to several hours to find a shadow lasting just 1 crossing time. For an $M = 1$ system a shadow can actually exist for several crossing times, which can take on the order of CPU days to find on current hardware. Since this time scales as $O(M^3)$, it is exceedingly expensive to attempt shadowing on higher dimensional systems. However, a vital scientific question is how shadow lengths scale with the dimensionality of the system. So, faster shadowing algorithms are needed.

3.2 Brief overview of the optimizations

This paper introduces 8 major optimizations to the GHYS/QT refinement procedure. Each provides a constant speedup roughly between 1.5 and 8, so the computational complexity is not reduced below $O(M^3)$. However, the cumulative speedup is about 10^2, which we believe is significant. Future work towards an $O(M)$ speedup is discussed in the final section.

All 8 optimizations are briefly introduced in this subsection. The first three are trivial and are not discussed further, while the remaining 5 are discussed at length in the following subsections. Most of the effort is focussed on reducing both the number of resolvents to recompute, and the time spent recomputing each. This necessitates handling the ensuing complications of when a resolvent *must* be recomputed, and when it would be worthless to do so. Furthermore, each of the optimizations is independent of all the others. It is conceivable that some optimizations may significantly decrease the reliability of refinement for some systems, although this wasn't observed for our N-body systems. In that case, some of these optimizations may simply be turned off.

Each optimization (exept the first) has a name, and a single-letter identification. We 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.

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 to be computed: one for evolving perturbations forward in time, and another to evolve them backward. Since most of the time is spent computing the resolvents, fixing this oversight provides an immediate speedup of a factor of about $3M$. The effect of fixing this oversight is *not* included in Table I below, but it partly explains why QT thought that shadowing is wholly impractical for large N systems.

Compute Backward Resolvent by Inverting Forward Resolvent (*I*). The above optimization can be extended further by noting that the resolvent matrix L_i that maps perturbations from time t_i to t_{i+1} is precisely the inverse matrix of the resolvent that maps small perturbations from t_{i+1} to t_i. Thus, rather than doing an integration from t_{i+1} backward to t_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 a speedup of about 2.

Fixed Tolerance Resolvent (*F*). For smooth problems like the N-body problem, it appears that when a resolvent is recomputed at all, it does not need to be computed very accurately, even in late refinements where the 1-step errors approach the machine precision. In these N-body experiments, when this optimization was on, resolvents were always computed to a fixed tolerance of 10^{-6}. It seems that it is the movement of the orbit during refinement that dictates recomputation of a resolvent, not the numerical error in its construction.

Cheaper "Accurate" Integrator (*C*). 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}. Of course, if the *Fixed Tolerance Resolvent* optimization is used, this only effects the 1-step error tolerances.

Constant Resolvent, Unstable, and Stable Directions (*R*). If a shadow exists, then by definition it cannot be too far away from the original noisy orbit. It is reasonable to assume that sometimes 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 accurate integrator, it means that the RUS often needs only to be computed once to a lax tolerance. This is analogous to a Newton's method that does not recompute the Jacobian at every iteration.

Re-use RUS From a Previous Successful Shadow (P). 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 steps of B. Thus only k new RUS_i's need to be computed.

Recompute Specific RUS_i's (i). If it can be shown that a particular RUS_i, or a small set of them, is causing Constant RUS refinements to fail, then perhaps it is sufficient only to recompute the "bad" RUS_i's, rather than all of them along the entire orbit. It is not trivial to decide which ones are bad, though.

Large shadow steps (L). 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. This optimization alone produced speedups of more than an order of magnitude in some cases.

3.3 Constant resolvent, unstable and stable directions

For a smooth, continuous, well-behaved function, the Jacobian is also smooth and continuous. Most Hamiltonian systems (that we have seen) are at least piecewise well-behaved in this sense. The N-body problem, in particular, is of this class, assuming no 0-distance collisions occur, or if the forces are artificially bounded through "softening". 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 computed solely from the resolvents, they will also change only slightly when the path is perturbed. Thus, the RUS computed on one refinement iteration may be re-usable on some subsequent iterations. Computing the new 1-step errors is then the only significant work to be performed.

We have observed 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 10^{-15}. If refinement fails using Constant RUS, then perhaps there is a spot in the orbit where some RUS_i's change quickly with small perturbations in the orbit. In that case, the RUS can be re-computed for a refinement or two, after which Constant RUS may be tried again. However, when Constant RUS fails it often suggests that there is a trouble spot somewhere in the orbit that may cause a glitch when attempting to shadow the orbit over longer periods.

3.3.1 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 norm of the maximum 1-step error over the previous few refinements. Originally, we used a simple algorithm (like QT's) that would signal failure when a certain number K of successive refinements had failed, for $\mu = 0.9$ (see (1)) and $K = 3$. 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 (approximately) $\{8, 4, 2, 1, 8, 4, 2, 1, 8, \ldots\} \times 10^{-13}$. Cases were seen in which the number of refinements in these loops was 3, 4, and even up to 8. Clearly a simple "die when K successive failures are seen" is not general enough to catch this kind of loop.

The *arithmetic running average*

$$A_{j+1} = \alpha A_j + (1-\alpha)\nu, \qquad \alpha \in (0,1) \tag{12}$$

is useful here, 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 each new element. A rule of thumb is that A is roughly an average of the most recent $2/(1-\alpha)$ elements. However, (12) 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. A *geometric* equivalent of (12) is more appropriate, to allow values differing by orders of magnitude to average meaningfully. We define the *geometric running average* G as

$$G_{j+1} = \sqrt[n+1]{(G_j)^n \nu} \tag{13}$$

where ν is the new element, and $\frac{n}{n+1}$ is analogous to α in (12), so higher n implies longer memory. Both (12) and (13) require initialization to some reasonable starting value A_0 and G_0, respectively. n should not be too large; we 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: (a) 1-step errors have reached the machine precision, or (b) 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 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 $I < \mu$ (μ from (1)) then the current refinement is successful.

3.3.2 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 we have built over many months of trial and error. Let μ_R be the success value (from (1)) when the RUS is being recomputed, and μ_C the success value when using Constant RUS. We used $\mu_R = 0.1$, $\mu_C = 0.5$, because we expect a Constant RUS iteration not to be as successful as one that recomputes the RUS, but they are much cheaper so we can afford a lower expected improvement per Constant RUS refinement.

- When using recomputed RUS, it is safe to switch to Constant RUS when the geometric running average improvement becomes $< \mu_R$. 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 $> \mu_C$ for 3 successive refinements. We then discard all progress made by Constant RUS and revert to the previous orbit that used recomputed RUS. We believe it is advantageous 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 usually 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 than is necessary. (Unless the *Fixed Tolerance Resolvent* optimization is used.)
- Finally, after having switched back to recomputed RUS, it is time to admit failure if the geometric running average improvement stays $> \mu_R$ for 3 successive refinements; and it is safe to switch again to Constant RUS when the geometric running average improvement becomes less than μ_R.

We found that there are usually no half-measures when using recomputed RUS — refinement either succeeds geometrically or fails miserably. Finally, there may be a place for other noise-reduction procedures in the set of expensive and cheap refinement algorithms.

3.3.3 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 an orbit B being near to the RUS for a nearby orbit A. 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 which of A's RUS's to use, if A's RUS was ever recomputed. 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 to be computed.

3.3.4 Recompute specific RUS_i's

When Constant RUS refinements fail, it may be because some of the RUS_i's are invalid due to the current trajectory drifting too far from the trajectory for which the RUS_i was computed. If the set of RUS_i's that are invalid is small, then it would be advantageous to recompute only those RUS_i's that are causing the failure.

A difficulty arises, however, when trying to pinpoint which RUS_i's are causing the problem. Note that refinement using the GHYS/QT algorithm is a *global* method: the correction $\mathbf{c}_j$ is a function of the resolvents and 1-step errors of *all* the timesteps, not just L_j and $\mathbf{e}_j$. Thus, it is not trivial to decide which RUS_i's are causing a particular $\mathbf{c}_j$ to be invalid.

What we would like is some measure M_i of how bad a RUS_i is. Then we can insert RUS_i's into a priority queue based on M_i, and extract the RUS_i's that are worse than some criteria, and recompute them. Some obvious measures for M_i are:

- the 1-step error at step i.
- the distance $\mathbf{p}_i$ has moved since the last time its RUS_i was computed.
- the norm $\|L_i\|$, which is a measure of how sensitive that area of the orbit is to small changes.

Once the M_i's are ordered by some measure, a criterion is needed to decide where to put the dividing line between those RUS_i's that need recomputing and those that don't. Some obvious criteria are:

- recompute all RUS_i's for which $M_i > M'$ for some constant M'.
- recompute some constant number of the RUS_i's.
- recompute some constant fraction of the RUS_i's.
- recompute all whose $M_i > w(\bar{M})$ for some function w of the geometric mean measure $\bar{M}$.

None of these measures taken alone with any of the above criteria taken alone seem sufficient, since there were cases in which recomputing the entire RUS resulted in a successful refinement, but a single-measure-single-criterion method for recomputing particular RUS_i's resulted in a long sequence of unsuccessful refinements.

After much trial and error, a measure was found which gives a speedup no better than a factor of 2, which is much less than we expected. Let us focus on a particular timstep i. Let k be the current refinement iteration, and k' be the refinement at which the RUS_i was last computed. Let $L_i^{k'}$ be the resolvent for step i computed during refinement k'. Let $\mathbf{p}_i^{k'}$ be the orbit point at step i from refinement k', and let $\mathbf{p}_i^k$ be the orbit point at step i for the current refinement k. When trying to decide if we need to compute L_i^k, the measure we devised is

$$M_i = \|L_i^{k'}(\mathbf{p}_i^{k'} - \mathbf{p}_i^k)\|.$$

The intent is to combine the effects of the norm of L_i, and how much the orbit has moved at that point since L_i was last computed. L_i^k is recomputed if it satisfies *any* of the following criteria:

- always recompute the worst K RUS$_i$'s, for some integer K;
- always recompute the worst F RUS$_i$'s, for some fraction F of all of them;
- always recompute RUS$_i$ if $M_i > M'$ for some constant M'.
- for each RUS$_i$ satisfying one of the above, also recompute RUS$_{i-1}$, since L_i starts at $\mathbf{p}_i$, while L_{i-1} ends at $\mathbf{p}_i$.

We used $K = 1$, $F = \frac{1}{8}$, and $M' = 10^{-3}$.

Unfortunately, this set of measures and criteria eventually recomputes most of the RUS$_i$'s except in the final few refinements of a failing search. In other words, this set of criteria gives the best speedups only during shadow searches that will eventually fail to find a shadow. This is a useful speedup, because a large fraction of the time spent trying to find the longest shadow is spent in the failing search for a shadow longer than the longest eventually found. However, it would be better if some criteria could be found that also aids in searches that are eventually successful. It seems plausible that some such measure and criterion should exist. Since the benefits are potentially enormous for orbits with many timesteps, more study seems appropriate. There is also the possibility that no significantly better measure exists.

3.4 Large Shadow Steps

3.4.1 Rationale

A numerical solution of an ODE is represented by an ordered, discrete sequence of points. Say we were to construct a continuous solution $\mathbf{p}(t)$ from these points, for example by using a suitably smooth and accurate interpolant. Then we could extend the definition of $\delta_{\mathbf{x}}$-shadowing to continuous systems as follows: if $\mathbf{x}(t)$ is a true solution then $\mathbf{x}(t)$ $\delta_{\mathbf{x}}$-shadows $\mathbf{p}(t)$ on $t_0 \leq t \leq t_1$ if $\forall t \in [t_0, t_1]$, $\|\mathbf{x}(t) - \mathbf{p}(t)\| < \delta_{\mathbf{x}}$.

Now, it should be clear that we can choose *any* representative sequence of points along $\mathbf{p}(t)$ to use in the refinement algorithm; we need not choose the points from which $\mathbf{p}(t)$ 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 $\mathbf{p}(t)$. 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. Each RUS$_i$ requires a matrix (the resolvent), and two sets of basis vectors.[4] Second, if the "accurate" integrator has a large startup cost, then we wish to minimize the number of startups. For example, the Adams's method we used takes extremely small internal timesteps at the beginning of each shadow step.

[4]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).

3.4.2 Estimating the largest shadow steps that can be used

For the N-body simulations reported in this paper, a set of "standardized" units [14] was used such that the scale of the system in all macroscopic units of interest was of order 1 — *i.e.,* the system had a diameter of order 1, the crossing time and total energy were 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. We 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 L_i. Recall that the linear perturbation map or *resolvent* $R(t_1, t_0)$ is a first-order approximation that maps small perturbations of the orbit $\mathbf{p}(t)$ from time t_0 to t_1, so that a system starting at $\mathbf{p}(t_0) + \mathbf{e}_0$ evolves, to first order in $\mathbf{e}_0$, to $\mathbf{p}(t_1) + R(t_1, t_0)\mathbf{e}_0$. For this equation to be valid, $\mathbf{e}_0$ must be small.[5] 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 1-step error. Although the allowable perturbation size is problem-dependent, we 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 $\sim 10^{-3}$ 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 $\mathbf{e}_i$ is the 1-step error at $\mathbf{p}_i$, $\mathbf{c}_i$ is the correction, $L_i = R(t_{i+1}, t_i)$ is the linear map and ϕ is the exact perturbation map such that $f(\mathbf{p}_i + \mathbf{c}_i) = \mathbf{p}_{i+1} + \phi(\mathbf{p}_i, \mathbf{c}_i)$, then $\mathbf{p}_{i+1} + L_i\mathbf{c}_i$ must be close to $\mathbf{p}_{i+1} + \phi(\mathbf{p}_i, \mathbf{c}_i)$ in comparison to the size of the 1-step errors:

$$\|L_i\mathbf{c}_i - \phi(\mathbf{p}_i, \mathbf{c}_i)\| \ll \max_i \|\mathbf{e}_i\| \tag{14}$$

In practice, (14) is impractical to use. Another criterion related to (2) is how well $L_i\mathbf{c}_i$ approximates $f(\mathbf{x}_i) - f(\mathbf{p}_i)$, recalling that the error is $O(|\mathbf{c}_i|^2)$. It may be possible to apply the Newton-Kantorovich theorem for the convergence of Newton's method, but we have not tried this. It may simply 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. This is what we did.

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 ΔT, each with error δ. Then after S timesteps and time $T = S\Delta T$, the first error will be magnified to $e^{\lambda T}\delta$, and the sum of all errors will grow as a random walk with average length $\delta\sqrt{S}$. If E is the maximum allowable 1-step error for which the linear map is valid, then a simplistic upper bound on the size of shadow steps is

$$\min\left(\frac{\ln(E/\delta)}{\lambda}, \quad \Delta T\left(\frac{E}{\delta}\right)^2\right). \tag{15}$$

For our N-body systems, this equation gives an upper bound of about 1 to 10 time units, which is

[5] 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 $\mathbf{e}_0$.

larger than the 0.1 sized shadow steps we used. Thus, (15) seems to be 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 $\{\mathbf{p}_i\}_{i=0}^{Q}$ 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 $\mathbf{p}(t)$ is initialized to the noisy point $\mathbf{p}(t_i) = \mathbf{p}_i$. Accurate integration of $\mathbf{p}(t)$ proceeds successively to times t_j, $j > i$ until $\|\mathbf{p}_{j+1} - \mathbf{p}(t_{j+1})\| > E$, when t_j marks the end of the shadow step. The accurate solution is then re-initialized to $\mathbf{p}(t_j) = \mathbf{p}_j$ to begin the next shadow step.

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 about the "cheaper accurate integrator").

Originally we thought that, as refinement proceeds to decrease the 1-step errors, it would be advantageous to construct larger-and-larger shadow steps. However, we no longer believe this will give much of a speedup, 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] We did not do this.

Finally, implementing dynamic shadow step sizes may help in another area. As described above, we currently discard all progress made by Constant RUS when 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.

3.5 Cheaper accurate integrator

When computing the 1-step errors of a noisy orbit, an integrator must be used 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 do not need to be computed to machine precision. In practice, we found that it is sufficient to compute the 1-step errors to an accuracy of 10^{-3} times 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} times more accurately sometimes results

[6] It is interesting to note in passing that the larger the shadow steps, the more obvious it is that refinement converges to a solution of the accurate integrator, instead of 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 large shadow step.

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 sometimes result in the integrator returning *exactly* the same result. 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. This is less likely to happen with a difference of 10^{-3}. 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.

3.5.1 Choosing the accurate integrator

The only methods we have tried as our accurate integrator are Adams 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 or a Bulirsch-Stoer method is probably apt, because even though they both have high startup cost, they can eventually move quickly along the solution if they are not restarted 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 shadowing with an integration routine that uses individual timesteps for each star, because currently the timestep of the entire system is restricted to the smallest required timestep.

4 Numerical results of the optimizations

To quantify the speedups 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 $(0, 1)$.[8] 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 *Unix* 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 [15] with pure relative error control of

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

[8] 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 our results with those of QT, even though they used a more realistic "Plummer" distribution.

10^{-6}. This figure agreed well with the magnitude of the initial 1-step errors computed during the first refinement. Although one crossing time sounds short, it is long enough that 5 of the orbits contained trouble spots.[9] 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. This is what has been observed with the longer orbits we have shadowed, although we 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, 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.

Table I 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 (after correction of the $3M$ oversight) are comparable to the run times of our 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 (column L) 8 times longer gives an average speedup of about 5.5. Large shadow steps are trivial to implement, so we would recommend that, even if no other optimizations are adopted, large shadow steps be used. 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. The P, F and i optimizations were not tested alone.

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

[9] If the orbit has a trouble spot, then *Constant Resolvent* and *Re-use RUS from previous successful shadow* will have little effect, because the RUS will be re-computed in an attempt to find a shadow. The other optimizations — *Cheaper Accurate Integrator, Large Shadow Steps, Fixed Tolerace Resolvent*, and *Backwards Resolvent by Inverting Forward Resolvent* — still offer significant performance improvement.

Table I — Speedups of the optimizations

seed	time(s)	QT	L	I	C	R	CR	CPR	CLR	KR	KRI	KRIF	KRIFi
1	1442	1.06	4.54	2.01	1.96	3.00	8.87	14.2	30.8	37.2	47.4	53.53	57.37
2	9379	0.70	6.08	1.82	3.09	3.48	11.2	15.7	52.3	68.3	75.7	79.83	79.67
3	7847	0.72	6.30	1.88	2.36	2.56	9.29	13.0	44.0	59.7	65.5	87.61	86.78
4	7011	0.86	11.1	1.98	2.74	2.84	7.88	12.3	40.5	55.8	74.3	79.19	78.52
5	12204	0.93	12.2	2.13	2.42	1.08	1.66	1.54	5.31	4.96	8.0	22.14	37.55
6	9443	1.14	7.24	2.01	3.18	3.98	10.8	13.5	5.36	56.8	29.8	80.13	137.64
7	6277	1.13	5.51	1.85	2.63	2.71	8.50	11.6	46.9	63.7	82.9	91.60	90.58
8	6582	1.13	4.79	2.04	1.91	2.79	7.09	10.6	33.8	45.4	56.9	72.76	67.95
9	7154	1.14	4.33	1.92	1.96	2.70	7.74	11.5	30.5	39.6	51.9	54.12	53.78
10	5832	1.13	4.32	1.87	1.54	2.72	6.46	10.7	29.1	45.5	57.2	63.89	63.59
11	6659	1.13	5.10	1.93	2.34	2.65	7.40	10.7	37.1	50.7	62.9	67.15	62.76
12	8623	0.86	5.00	2.44	2.57	3.37	9.64	13.7	44.8	52.4	75.4	79.77	79.19
13	14566	0.45	4.34	1.00	4.52	0.78	0.93	0.93	5.67	5.15	7.6	23.63	35.86
14	6976	0.74	4.54	2.00	2.17	2.85	7.23	11.1	32.9	45.2	49.4	58.08	57.66
15	5664	0.72	4.79	2.02	1.89	2.71	6.83	10.2	33.5	45.4	62.9	67.03	66.43
16	6634	1.15	4.89	1.97	1.82	2.64	7.35	10.6	33.8	44.8	58.7	62.65	62.08
17	6854	0.94	4.96	2.06	2.16	2.88	8.97	12.7	47.2	60.8	83.2	89.45	88.57
18	4932	0.62	2.88	1.74	1.69	2.17	5.38	7.84	25.7	34.1	45.7	49.70	49.27
19	8470	1.06	5.16	2.98	2.96	1.32	4.37	4.02	31.1	27.5	53.5	64.21	151.73
20	4566	1.12	5.59	1.91	2.04	2.78	6.21	9.50	33.7	48.6	64.3	62.95	66.32
21	7409	1.07	6.01	2.00	2.92	2.86	9.59	13.4	46.6	61.3	82.2	84.44	86.69
22	6988	1.22	4.89	2.17	2.00	3.03	7.34	11.4	36.1	44.0	57.2	61.00	60.52
23	7699	0.75	4.66	1.93	1.91	3.20	8.27	10.4	35.2	46.3	51.1	62.82	62.32
24	8851	1.18	2.53	1.79	2.78	2.89	10.2	15.4	52.9	71.1	93.1	95.29	100.39
25	7287	1.13	3.58	1.96	2.67	2.53	7.58	11.5	34.1	47.4	59.3	62.81	62.34
26	10736	0.85	7.74	2.84	2.79	4.33	12.4	16.9	58.2	76.5	81.1	63.96	63.50
27	7001	1.13	5.07	2.01	2.24	2.79	8.35	12.1	38.9	54.5	70.1	76.20	75.58
28	7324	1.13	4.49	2.06	1.93	2.72	8.01	11.2	33.2	49.3	55.0	58.98	55.21
29	13779	0.80	4.01	1.89	2.01	0.89	1.16	1.15	4.03	3.99	6.5	17.00	29.02
30	7932	1.14	10.9	2.01	2.17	2.81	8.82	13.0	43.3	67.5	86.7	91.38	90.70
31	6989	1.20	5.57	2.26	2.37	2.81	8.44	12.3	47.4	57.2	77.6	84.06	83.29
32	6720	1.08	4.80	1.89	1.90	2.78	7.61	11.5	34.0	50.5	65.5	70.33	69.67
avg	7682	1.07	5.56	2.01	2.36	2.68	7.55	10.8	34.6	47.5	59.3	66.8	72.3

	LEGEND
time(s):	time in seconds for unoptimized version.
QT:	original code of Quinlan & Tremaine [16]
L:	*L*arge shadow steps
I:	backward resolvent by *I*nverting forward resolvent
C:	*C*heaper accurate integrator
R:	Constant *R*US
P:	re-use RUS from *P*revious successful shadow (appears only in combinations)
K:	CPL together.
F:	*F*ixed Tolerance Resolvent.
i:	Recompute specific RUS_i's.

(the final one) to high accuracy. Second, re-using the RUS of a previous successful shadow (P) makes sense only when R is also used. It gives an average speedup of 43% over CR. The next three columns show other combinations. An interesting point to note is that, except for CR, the combinations give a speedup less than the product of appropriate 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 can be seen that the optimizations in the last two columns, F and i, contribute little to the average. This is because most of the systems in this table were shadowable. In fact, F slightly slows down refinement in one case. However, orbits 5, 6, 13, 19, and 29 obtained speedups significantly less than the average in the KRI column, but gain was made in the final two columns of up to a factor of 5. These were precisely the orbits that were most difficult to shadow (as the time column shows), and in fact some of them were not shadowable for their entire lengths. For these orbits, significant time was spent in the failed search to find a shadow lasting 1.28 time units. This indicates that the F and i optimizations are most effective in speeding up failing searches. A detailed discussion about these orbits, and various problems encountered shadowing them, can be found in Chapter 2 of the first author's Master's Thesis [13].

5 Discussion and conclusions

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, we very frequently see cases in which a shadow is not found for shadow steps $0..S$, but a shadow is found for the *super*set $0..2S$. In other words, the algorithm failed to find a shadow of length S even though one exists. This also occurs if all the optimizations were off, so it is not merely that the optimizations reduce reliability. Thus, in the search for the longest shadow, if our algorithm finds a shadow of length S but none of length $2S$, we also try $4S$ before giving up. This is expensive, but necessary to lessen the chance of failing to find a shadow that exists.

The GHYS/QT refinement algorithm is trivially parallelizable, since the computation of each RUS_i is completely independent of all the others; for the same reason, it also has excellent locality of reference in a serial implementation, so virtual memory paging is minimized.

There may be further optimizations possible. For example, in the N-body problem, the phase-space dimensions are split into sets of 6, one set for each particle in the system. Instead of computing the $6M \times 6M$ resolvent in $O((6M)^3)$ time, perhaps it may be possible to refine the trajectories of each particle separately, requiring M resolvents each of dimension 6×6, and $O(6^3 M)$ time to compute [13].

This optimized algorithm has been tested on a smooth, almost hyperbolic Hamiltonian system with 180 dimensions. Shadows were successfully found to last about 50 e-folding times, at which point the noisy trajectory's Hamiltonian had changed by almost one part in 10^{-2}, so it was unrealistic to expect a true shadow to last much longer. This shows that the algorithm is at least capable of finding long, high-dimensional shadows.

The algorithm was then used to attempt shadowing an N-body system with $M = 25$ moving particles amongst $N - M = 75$ fixed ones, *i.e.,* a system with 150 phase space dimensions. Preliminary results [13] seem to indicate that shadow lengths are proportional to $1/M$, *i.e.,* , they decrease rapidly in length as the number of dimensions of the system is increased. If further work confirms this result, it indicates that shadowing may be too stringent an error criterion for simulations of "real" high-dimensional Hamiltonian systems.

Source code is available from the first author.

Acknowledgements

We thank Gerry Quinlan and Scott Tremaine for helpful discussions throughout this research, and for providing their original source code. The University of Toronto undergradute Computing Disciplines Facility generously provided workstation clusters on which we consumed several CPU years during this research. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, and the Information Technology Research Centre of Ontario.

References

[1] D. V. Anosov. Geodesic Flows and Closed Riemannian Manifolds with Negative Curvature. *Proc. Steklov Inst. Math*, 90:1, 1967.

[2] W.-J. Beyn. On Invariant Closed Curves for One-Step Methods. *Numer. Math.*, 51:103–122, 1987.

[3] R. Bowen. ω-Limit Sets for Axiom A Diffeomorphisms. *Journal of Differential Equations*, 18:333, 1975.

[4] Bruce W. Char. *Maple user's guide.* WATCOM, Waterloo, Ontario, Canada, 1985.

[5] S. N. Chow and K. J. Palmer. On the Numerical Computation of Orbits of Dynamical Systems: the One-Dimensional Case. *Dynamics and Diff. Eq.*, 3:361–380, 1991.

[6] S. N. Chow and K. J. Palmer. On the Numerical Computation of Orbits of Dynamical Systems: the Higher-Dimensional Case. *J. Complexity*, 8:398–423, 1992.

[7] S. N. Chow and E. S. Van Vleck. Shadowing of Lattice Maps. In Peter E. Kloeden and Kenneth J. Palmer, editors, *Chaotic Numerics*, pages 97–113. American Mathematical Society, 1993.

[8] S. N. Chow and E. S. Van Vleck. A Shadowing Lemma Approach to Global Error Analysis for Initial Value ODEs. *SIAM J. Sci. Comp.*, 15:959–976, 1994.

[9] 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.

[10] 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.

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

[12] Stephen M. Hammel, James A. Yorke, and Celso Grebogi. Do Numerical Orbits of Chaotic Dynamical Processes Represent True Orbits? *Journal of Complexity*, 3:136–145, 1987.

[13] Wayne Hayes. Efficient Shadowing of High Dimensional Chaotic Systems with the Large Astrophysical N-body Problem as an Example. Master's thesis, Dept. of Computer Science, University of Toronto, January 1995. Available via anonymous ftp from `ftp.cs.utoronto.ca/pub/reports/na/hayes-95-msc`. Source code available from the first author, `wayne@cs.utoronto.ca`.

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

[16] 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.

[17] J. M. Sanz-Serna and S. Larsson. Shadows, Chaos, and Saddles. *Appld. Numer. Math.*, 13:181–190, 1993.

[18] T. Sauer and J. A. Yorke. Rigorous Verification of Trajectories for the Computer Simulation of Dynamical Systems. *Nonlinearity*, 4:961–979, 1991.

[19] Erik S. Van Vleck. Numerical Shadowing Near Hyperbolic Trajectories. *SIAM J. Sci. Comp.*, 16, 1995.