INTERPOLATED MODIFIED HAMILTONIANS

BY

ROBERT DANIEL ENGLE

B.A., St. Olaf College, 2002

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

To Betsy.

Acknowledgments

The author is thankful for the help of Matt Drees who performed preliminary work related to finding expressions for low order truncated modified Hamiltonians, as well as calculating coefficients for interpolated modified Hamiltonian formulas.

Table of Contents

List of Figures

List of Figures

This page is included solely for purposes of getting a “List of Figures” entry into the Table of Contents, as per the style guidelines laid out by the UIUC Graduate College Thesis Office. It is not meant to be printed or included in the final thesis.

Chapter 1

Introduction

Hamiltonian systems of ordinary differential equations arise in areas of scientific research, such as molecular dynamics, whose equations are governed by conservative forces. These systems have the form $\dot{x}(t) = JH_x(x(t))$ for some Hamiltonian $H(x)$ with $x = [q^\mathsf{T}, p^\mathsf{T}]^\mathsf{T}$ and $J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$, with the subscript x denoting the gradient operator. An analytical solution to these Hamiltonian systems of equations is not possible, in general, due to their complexity. Solving numerically yields an approximate solution at discrete points in time by stepping forward in time by an amount h, with an integrator Φ_h evolving new approximate values at each timestep as $x^{n+1} = \Phi_h(x^n)$. It is desirable that this integrator be symplectic, because key properties of Hamiltonian systems are conserved by symplectic integrators, resulting in more accurate long time integrations [6, 2]. An integrator is symplectic if $\Phi_{h,x}(x)^\mathsf{T} J \Phi_{h,x}(x) = J$. An example of a symplectic integrator is the leapfrog method which, for a separable Hamiltonian $H(q,p) = \frac{1}{2}p^\mathsf{T} M^{-1} p + U(q)$ with M a diagonal mass matrix and $U(q)$ the scalar potential energy function, defines the values $q^{n+1}, p^{n+1}, F^{n+1}$ in terms of the values q^n, p^n, F^n. The leapfrog method with timestep h is given by

$$p^{n+1/2} = p^n + \frac{h}{2}F^n, \quad q^{n+1} = q^n + hM^{-1}p^{n+1/2}, \quad F^{n+1} = F(q^{n+1}), \quad p^{n+1} = p^{n+1/2} + \frac{h}{2}F^{n+1},$$

where the force function $F(q)$ equals the negative gradient of $U(q)$. At a given step n, the numerical solution x^n approximates the analytical solution after n steps $x(nh)$. This relationship is only approximate because of the use of a discrete timestep, resulting in truncation errors in the integration. The numerical solution is actually the solution not of the original Hamiltonian system, but of some slightly different, modified equations. Typically these modified equations are defined by an asymptotic expansion in powers of the timestep, which, in general, does not converge. The modified equations are Hamiltonian, with modified (or "shadow" [1]) Hamiltonian H_h, if and only if the integrator is symplectic. The modified Hamiltonians are important because, for instance, they better indicate energy drift (due to instability) in a numerical simulation than does the original Hamiltonian. Also, calculating modified Hamiltonians and monitoring how well they are conserved provides evidence for the existence of a Hamiltonian for a particular problem and supports the validity of statistical estimates calculated from long time integration of chaotic Hamiltonian systems [5]. Truncated modified Hamiltonians are obtained by suitably truncating an asymptotic expansion in powers of the timestep, $H_h(x) = H(x) + h^2\eta_2(x) + h^4\eta_4(x) + \cdots$, but they are both difficult and expensive to compute and highly dependent on the particulars of the potentials and integrator used. A scheme is presented in [7] for computing highly accurate estimates of the modified Hamiltonians using only quantities which are already being calculated by the numerical simulation, resulting in interpolated modified Hamiltonians. The construction of these interpolated modified Hamiltonians assumes that the integrator is obtained by splitting the Hamiltonian, but otherwise is nearly independent of the details of the Hamiltonian system or the integrator, enabling their calculation even for systems for which the truncated modified Hamiltonians are impractical or impossible to compute. The degree k of polynomial interpolant used in this construction determines the order of accuracy $2k$ of the resulting interpolated modified Hamiltonian $H_{[2k]}$, with $H_{[2k]}(x) = H_h(x) + \mathcal{O}(h^{2k})$. This thesis presents a more thorough empirical study of the accuracy of these interpolated modified Hamiltonians as well as a more efficient implementation. Because of this less costly construction largely independent of the details of the Hamiltonian system of equations to be solved, interpolated modified Hamiltonians of high order can be efficiently obtained, even for molecular dynamics simulations.

Numerical experiments presented in chapter 2 elucidate the properties of these interpolated modified Hamiltonians. Test problems include one dimensional double well and piecewise-defined flat-bottomed well potentials, the two dimensional Hénon-Heiles Hamiltonian, and a molecular dynamics system consisting of 125 water molecules. Results are as follows:

1. Range of fluctuation of the interpolated modified Hamiltonians decreases as a function of order $2k$, asymptotically approaching some positive lower bound.

2. The magnitude of this lower bound depends on the simulation timestep, being smaller for simulations with smaller timesteps.

3. The magnitude of the lower bound also depends on the smoothness of the potential, being smaller for simulations with smoother potentials. In the case of a less smooth one dimensional C^1 potential the magnitude of the lower bound on conservation of the interpolated modified Hamiltonian is quite large relative to the conservation of the energy.

4. For simulations with sufficiently large timesteps, the conservation of interpolated modified Hamiltonians $H_{[2k]}$ for small values of k is better for even values k_{even} than for subsequent odd values $k_{\text{even}} + 1$; however the conservation of $H_{[2k+4]}$ is always superior to that of $H_{[2k]}$.

5. High order interpolated modified Hamiltonians in molecular dynamics simulations are very well conserved.

6. On timescales tested, fluctuations of the interpolated modified Hamiltonians for the case of molecular dynamics are occasionally interrupted by sudden jumps many orders of magnitude greater than the typical fluctuations, yielding the appearance of large scale piecewise constant behavior onto which the small scale fluctuations are superimposed. These large scale jumps apparently reveal simulation events not visible in plots of the energy alone.

7. Comparisons between truncated and interpolated modified Hamiltonians for simple one and two dimensional potentials show that conservation of interpolated modified Hamiltonians is somewhat inferior to that of truncated modified Hamiltonians of the same order for sufficiently high orders.

8. In the case of the Hénon-Heiles Hamiltonian, trajectories associated with the truncated modified Hamiltonians are superior to those associated with interpolated modified Hamiltonians of the same order.

Chapter 3 outlines the derivation of interpolated modified Hamiltonians as presented in [7]. This involves the definition of a homogeneous extension of the Hamiltonian, which in turn is used to define an augmented method for evolving an approximate numerical solution in time. Construction of interpolated modified Hamiltonians as presented in [7] begins by constructing polynomial interpolants to interpolate values computed by the augmented method, with the degree k of the interpolant determining the order $2k$ of the corresponding interpolated modified Hamiltonian $H_{[2k]}$. Interpolated modified Hamiltonians are then defined in terms of these interpolants by forming combinations of computed quantities, which can be proven to be equal to the modified Hamiltonian of the original unaugmented Hamiltonian plus some error terms of order h^{2k}.

The scheme presented in [7] uses centered differences to form the aforementioned polynomial interpolants. Chapter 4 presents an improved implementation for calculating interpolated modified Hamiltonians which uses backward differences to form the interpolants, resulting in a number of changes to the construction and implementation of interpolated modified Hamiltonians, with a consequent significant reduction in storage requirements. In either case, polynomial interpolants are constructed in terms of differences of values produced by the simulation, which must be stored. Higher degree interpolants require higher order differences. Necessary differences can be efficiently updated at each step as part of the integrator. The use of centered differences results in storage requirements that are quadratic in the order. However, these storage requirements are reduced to a linear function of the order through the use of backward differences, as only the leading edge of the difference table need be stored. The savings are thus significant, especially for high order interpolated modified Hamiltonians and for systems with large numbers of degrees of freedom. In addition to storage savings, the direct calculation from simulation parameters of first and second order backward differences reduces the rounding error inherent in forming the interpolated modified Hamiltonians, resulting in a more accurate approximation.

Chapter 2

Numerical results

2.1 Test problems

Interpolated modified Hamiltonians $H_{[2k]}(x)$, $k = 1, 2, \dots, 12$, are computed for three simple potentials and molecular dynamics.

The first is a one dimensional double well potential $U(q) = \frac{1}{4}(q^2 - 1)^2$ with initial conditions $q = 0,\ \dot{q} = p = 0.2$.

The second is a one dimensional C^1 continuous piecewise potential,

$$U(q) \quad = \quad \begin{cases} \frac{1}{2}q^2, & q \leq 0, \\ 0, & 0 \leq q \leq 6, \\ \frac{1}{2}(q-6)^2, & q \geq 6, \end{cases}$$

with initial conditions $q = 0,\ \dot{q} = p = \sqrt{8}$ corresponding to an initial energy of 4.

The third is the two dimensional Hénon-Heiles Hamiltonian [3],

$$H(q,p) \quad = \quad \frac{1}{2}(p_1^2 + p_2^2) + \frac{1}{2}(q_1^2 + q_2^2 + 2q_1^2 q_2 - \frac{2}{3}q_2^3),$$

in which the first term represents kinetic energy $K(p)$ and the second term represents potential energy $U(q)$, with $q = [q_1, q_2]^\mathsf{T}$ and $p = [p_1, p_2]^\mathsf{T}$. This system exhibits chaotic behavior for energies higher than $\frac{1}{12}$, which is the case for the initial conditions of $q_1 = \frac{1}{2},\ q_2 = p_1 = p_2 = 0$ used in this study.

The molecular dynamics system studied consists of 125 water molecules whose atoms interact via two bonded forces, acting between and among atoms within the same molecule, and two non-bonded forces, acting between all intermolecular atom pairs, and constrained by spherical boundary conditions. A molecular dynamics simulation computes the time evolution of a system of N atoms bound by Newton's equations of motion,

$$\begin{aligned}\dot{q}(t) &= M^{-1}p(t), \\ \dot{p}(t) &= F(q(t)),\end{aligned}$$

subject to some initial positions q^0 and initial momenta p^0 at time $t = 0$. The force is the negative gradient of a scalar potential function, $F(q) = -U_q(q)$. The potential can be expressed as the sum of bonded terms, nonbonded terms, and boundary conditions, where in this study the bonded potentials are bond stretching and angle bending and the nonbonded potentials are the Lennard-Jones potential and the electrostatic potential, so that

$$U(q) = U_{\mathrm{BC}}(q) + U_{\mathrm{bond}}(q) + U_{\mathrm{angle}}(q) + U_{\mathrm{LJ}}(q) + U_{\mathrm{elec}}(q).$$

The atoms are constrained by spherical harmonic boundary conditions centered about the origin with the spherically symmetric potential U_{BC} equal to the sum over all i of terms

$$k_{\mathrm{BC}}(r_i - r_{\mathrm{BC}})^p,$$

with r_i the distance from the origin to the i^{th} atom. This potential takes effect only outside a radius of r_{BC}, and is zero for all atoms inside this radius. In this study the values $r_{\mathrm{BC}} = 21\,\text{Å}$, $k_{\mathrm{BC}} = 1\,\mathrm{kcal/mol/Å}^2$, and $p = 2$ are used.

The first bonded potential, the spring bond, is a 2-body interaction between an (i, j)-pair of covalently bonded atoms, with U_{bond} equal to the sum, over all such (i, j)-pairs, of terms

$$k_{\mathrm{b}}(r_{ij} - r_0)^2,$$

with r_{ij} the separation distance of the two atoms. For the case of hydrogen–oxygen interactions, the coefficient $k_{\mathrm{s}} = 450\,\mathrm{kcal/mol/Å}^2$ and the equilibrium distance $r_0 = 0.957\,\text{Å}$.

The angle potential is a 3-body interaction between a consecutively bonded (i, j, k)-triple of atoms, with U_{angle} equal to the sum, over all such (i, j, k)-triples, of terms

$$k_\theta(\theta - \theta_0)^2 + k_{\text{ub}}(r_{ik} - r_{\text{ub}})^2,$$

with θ the angle in radians between the vector pointing from atom j to atom i and the vector pointing from atom j to atom k. For the case of water molecules, the coefficient $k_\theta = 55\,\text{kcal/mol/rad}^2$ and the equilibrium angle $\theta_0 = 1.8242$ radians. The second term, the Urey-Bradley term, plays no role in this study, as the Urey-Bradley coefficient used was $k_{\text{ub}} = 0\,\text{kcal/mol/Å}^2$.

The Lennard-Jones potential exists between intermolecular (i, j)-pairs of atoms, with U_{LJ} equal to the sum, over all such intermolecular (i, j)-pairs, of terms

$$E_{\min}\left[\left(\frac{R_{\min}}{r_{ij}}\right)^{12} - 2\left(\frac{R_{\min}}{r_{ij}}\right)^{6}\right].$$

The minimum value of the potential $-E_{\min} = U_{\text{LJ}}(R_{\min})$ depends on the atom types involved in the interaction. For hydrogen–hydrogen interactions, $E_{\min} = 0.046\,\text{kcal/mol}$, for hydrogen–oxygen interactions $E_{\min} = 0.0836\,\text{kcal/mol}$, and for oxygen–oxygen interactions $E_{\min} = 0.1521\,\text{kcal/mol}$. The equilibrium separation distance $R_{\min}$ also depends on the types of atoms involved in the interaction. For hydrogen–hydrogen interactions, $R_{\min} = 0.449\,\text{Å}$, for hydrogen–oxygen interactions $R_{\min} = 1.9927\,\text{Å}$, and for oxygen–oxygen interactions $R_{\min} = 3.5364\,\text{Å}$. Commonly in molecular dynamics simulations the Lennard-Jones potential is taken to be zero outside a given cutoff distance c. In order to achieve a smooth transition a switching function is applied to the potential before the cutoff distance but outside some switching distance s, so that in this range the contribution of each intermolecular (i, j)-pair towards the total U_{LJ} becomes

$$E_{\min}\left[\left(\frac{R_{\min}}{r_{ij}}\right)^{12} - 2\left(\frac{R_{\min}}{r_{ij}}\right)^{6}\right]\left[\frac{(c^2 - r_{ij}^2)^2(c^2 + 2r_{ij}^2 - 3s^2)}{(c^2 - s^2)^3}\right] \qquad \text{for } s \leq r_{ij} \leq c.$$

The use of this switching function results in a C^1 continuous Lennard-Jones potential. However, in this study the switching distance was chosen large enough that no switching would ever take place, due to the imposed spherical boundary conditions, except where otherwise noted. The Lennard-

Jones potential, therefore, remains infinitely differentiable. The one exception occurs in section 2.5, in which results from molecular dynamics with $s = 4\,\text{Å}$, $c = 100\,\text{Å}$ are discussed.

The electrostatic potential U_{elec} equals the sum, over all intermolecular atom (i,j)-pairs, of terms

$$\frac{C a_i a_j}{\epsilon_0 r_{ij}},$$

and is attractive for atoms with oppositely signed charges and repulsive for atoms with identically signed charges. The quantities a_i and a_j are the charges of atom i and atom j, respectively. Atom charges, in units of electron charge, are $0.417\,\text{e}$ for hydrogen and $-0.834\,\text{e}$ for oxygen bound in water molecules. The coefficient $C = 332.0636\,\text{kcal}\cdot\text{Å}/\text{mol}/\text{e}^2$ is Coulomb's constant. The dielectric constant used is $\epsilon_0 = 1$. As with the Lennard-Jones potential, the electrostatic potential can be taken to be zero beyond the cutoff distance c with a switching function applied to the potential inside this distance, so that the contribution to U_{elec} from each intermolecular (i,j)-pair of atoms becomes

$$\frac{C a_i a_j}{\epsilon_0 r_{ij}} \left(1 - \frac{r_{ij}^2}{c^2}\right)^2 \qquad \text{for } 0 < r_{ij} \le c.$$

This results in a C^1 continuous electrostatic potential. However, in this study the cutoff was chosen such that it would never be reached, thus maintaining an infinitely differentiable electrostatic potential.

Initial conditions are randomly generated by NAMD [4] for a system at 300 K. Molecular dynamics results are obtained using a modified version of a program[1] compatible with NAMD but limited in scope to enable simpler modification. Interpolated modified Hamiltonian code was added to this program by the author. All other results are obtained using Hamiltonian solver code provided in appendix B. Integration is by the leapfrog method.

[1]credit: David J. Hardy, Department of Computer Science and Beckman Institute, University of Illinois, 1304 West Springfield Avenue, Urbana, Illinois 61801-2987 (dhardy@ks.uiuc.edu)

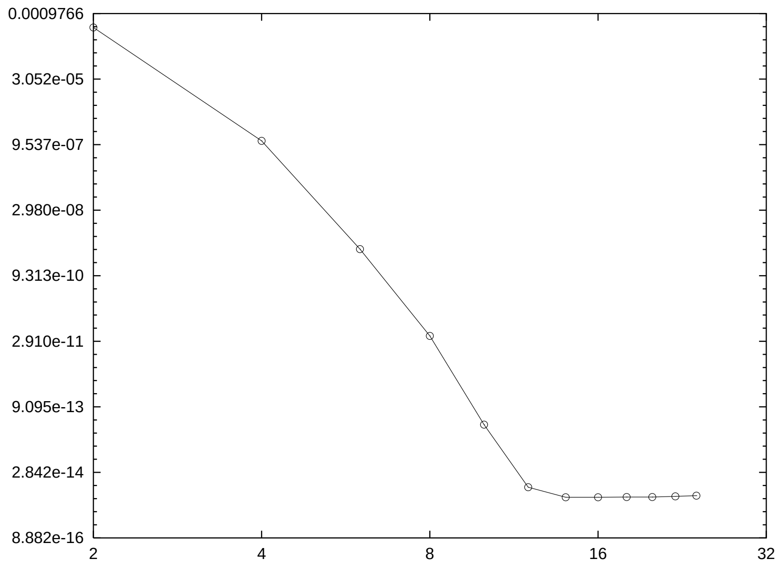

Figure 2.1: Conservation of IMH vs. order $2k$ for double well, $h = 0.05$

2.2 Conservation of interpolated modified Hamiltonians

Define conservation as $\max_n H_h(x^n) - \min_n H_h(x^n)$, where the $H_h(x^n)$ are the values of the given modified Hamiltonian at timesteps $\lceil k/2 \rceil, \lceil k/2 \rceil + 1, \ldots, T - \lceil k/2 \rceil$, with T the total number of timesteps in the simulation.

The conservation of interpolated modified Hamiltonians of higher order is superior to the conservation of those of lower order. Figure 2.1 plots the conservation of interpolated modified Hamiltonians (IMH) as a function of order for the double well potential with timestep 0.05 on a log-log plot. The labels on the x-axis correspond to the $2k$ in $H_{[2k]}$, with $2k$ the order of the corresponding interpolated modified Hamiltonian. Conservation improves towards some non-zero lower bound as the order increases, displaying diminishing returns, in the sense that the use of higher and higher orders of interpolated modified Hamiltonians results in progressively smaller improvements to the conservation on a logarithmic scale.

Similar conservation behavior is observed for truncated modified Hamiltonians as a function of order. A leveling off as conservation approaches some positive lower bound is apparent in plots of the conservation of truncated modified Hamiltonians versus order $2k$, $k = 1, 2, \ldots, 5$, for the Hénon-Heiles Hamiltonian for timesteps of $h = 0.9$ and $h = 0.9300004$. The beginnings of a leveling off

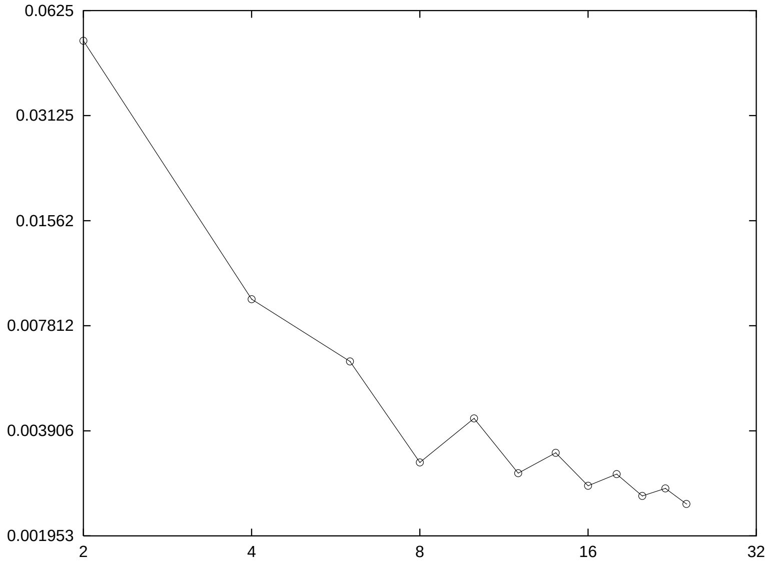

Figure 2.2: Conservation of IMH vs. order $2k$ for double well, $h = 0.5$

are also apparent in the same type of plot for the double well potential with $h = 0.5$.

The magnitude of this lower bound is a function of the simulation timestep. Compare, for instance, figures 2.1 and 2.2, which plot conservation versus order for the double well potential, the former with a timestep of 0.05 and the latter with timestep 0.5. The lower bound is smaller when a smaller simulation timestep is used.

Also, figure 2.2 illustrates how the conservation of the interpolated modified Hamiltonians $H_{[2k]}$ for small values of k is better for even values k_{even} than for subsequent odd values $k_{\text{even}}+1$ when the simulation timestep is made large. The emergence of this phenomenon as the timestep increases is shown by figures C.1 through C.8, two figures for each of the four potentials used in this study. In these cases, convergence is not monotonic as a function of k, but does approach the same lower bound whether k is even or odd. This dependence on k being even or odd arises because the construction of interpolated modified Hamiltonians is somewhat different in these two cases. However, the conservation of $H_{[2k+4]}$ is always superior to that of $H_{[2k]}$. This behavior was observed for the double well potential with a timestep of 0.2 or greater (figures C.1 and C.2), for the Hénon-Heiles Hamiltonian with a timestep of 0.3 or more (figures C.3 and C.4), for the piecewise-defined potential with a timestep of 0.05 or higher (figures C.5 and C.6), and for molecular dynamics with a timestep of 0.5 fs or higher (figures C.7 and C.8).

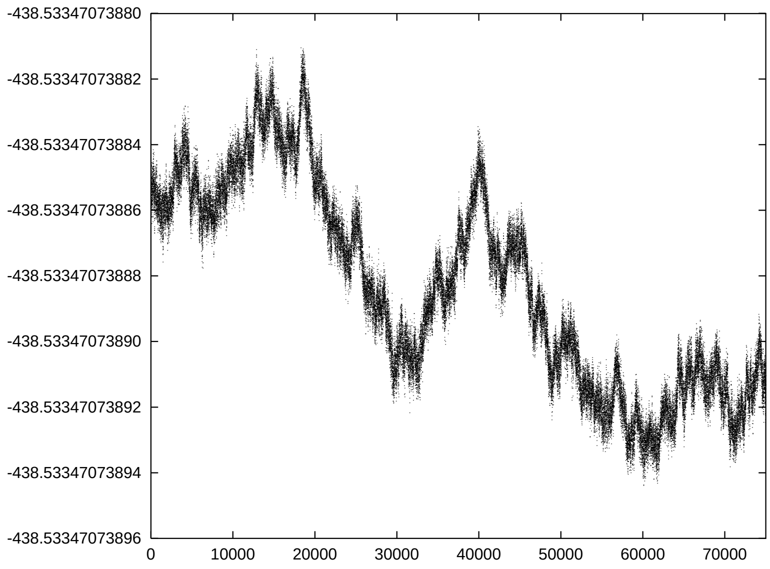

Figure 2.3: 24^{th} order IMH for molecular dynamics, $h = 0.25\,\mathrm{fs}$, 18.75 ps simulation

Additionally, the lower bound is higher when the potential is less smooth. Compare, for example, figure C.5 with figures 2.1 and 2.2. The lower bound is a much larger fraction of the conservation of the true energy for a less smooth one dimensional C^1 potential (figure C.5) than for an infinitely differentiable double well potential (figures 2.1 and 2.2).

2.3 Molecular dynamics

A system of 125 water molecules was simulated with a variety of timesteps and interpolated modified Hamiltonians up to 24^{th} order were calculated. The best conservation is exhibited by the highest order interpolated modified Hamiltonians. Figure 2.3 plots the 24^{th} order interpolated modified Hamiltonian versus time for a molecular dynamics simulation with timestep $h = 0.25\,\mathrm{fs}$. This order of interpolated modified Hamiltonian varies only in the 13^{th} decimal digit over the length of an 18.75 ps simulation. This variation is probably due to roundoff error, as suggested both by the very small magnitude of the fluctuations and by the fact that the conservation of interpolated modified Hamiltonians over the length of 2 ps simulations actually becomes worse when the timestep is reduced from 0.5 fs to 0.25 fs and from 0.25 fs to 0.125 fs.

During the course of longer simulations a different, larger scale behavior appears in plots of the

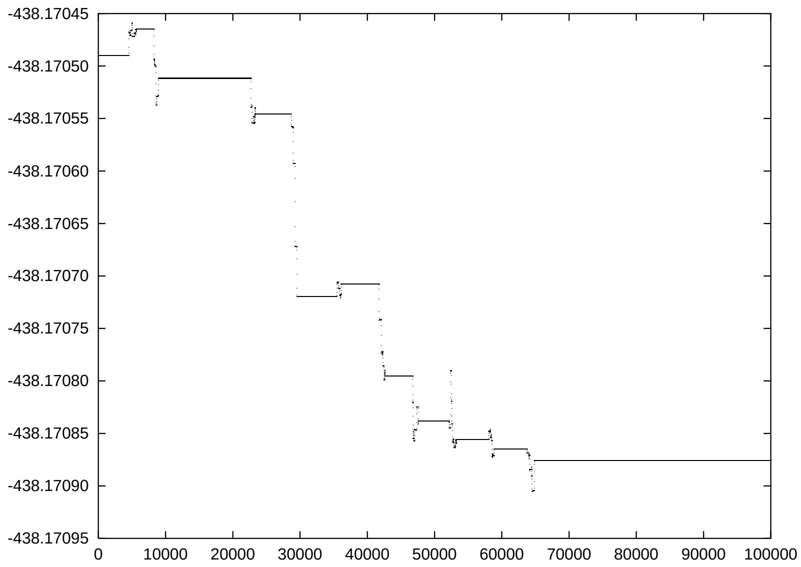

Figure 2.4: 24^{th} order IMH for molecular dynamics, $h = 0.5\,\text{fs}$, 50 ps simulation

high order interpolated modified Hamiltonians. This is illustrated in figure 2.4, which plots the 24^{th} order interpolated modified Hamiltonian for a molecular dynamics simulation at each step of a 50 ps simulation with timestep 0.5 fs. Over the length of the simulation this interpolated modified Hamiltonian is well conserved, with the exception of occasional sudden and clearly delineated jumps, instances in which the conservation of the interpolated modified Hamiltonians is orders of magnitude worse for the span of a handful of timesteps before settling back down into another well conserved regime. These jumps seem to indicate some sort of event not apparent from plots of the energy or lower order interpolated modified Hamiltonians.

2.4 Comparison: interpolated versus truncated

For simpler potentials such as the double well or Hénon-Heiles it is possible to obtain expressions for the first few orders of truncated modified Hamiltonians. Truncated modified Hamiltonians are obtained by forming interpolated modified Hamiltonians $H_{[2k]}$ and gathering terms in powers of the timestep h. These expressions are then truncated at the appropriate power of h to yield the desired truncated modified Hamiltonian. Correctness is checked by comparing these results to modified equations obtained symbolically from first principles. This allows for comparisons to be drawn

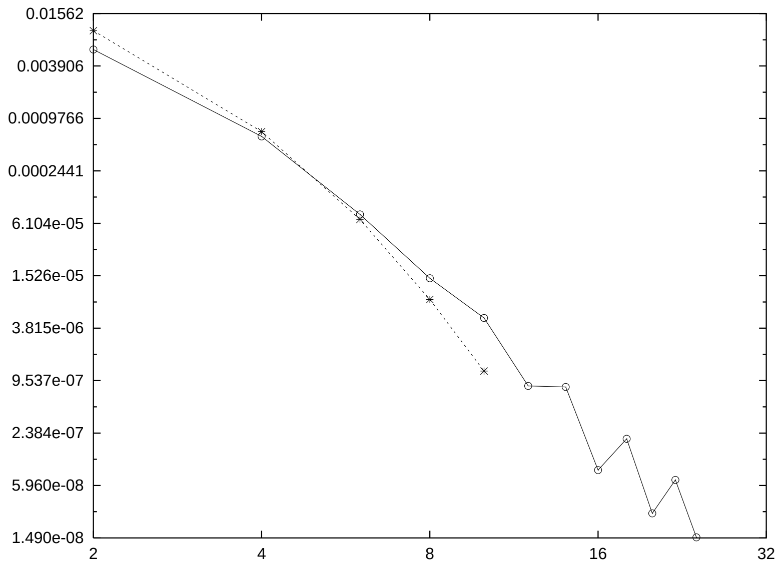

Figure 2.5: Conservation of IMH and TMH vs. order $2k$ for Hénon-Heiles, $h = 0.5$

between these values and the interpolated modified Hamiltonians of corresponding orders.

Figure 2.5 and figures C.1 through C.4 support the observation that for sufficiently high order the truncated modified Hamiltonians are always better conserved than interpolated modified Hamiltonians of the same order. Figure 2.5 plots the conservation of the truncated modified Hamiltonian (TMH) versus order on the same log-log plot as the conservation of the interpolated modified Hamiltonian versus order, for the Hénon-Heiles Hamiltonian with timestep 0.5. Truncated modified Hamiltonian conservation data are plotted as asterisks connected by dotted lines, and interpolated modified Hamiltonian conservation data are plotted as open circles connected by solid lines. The question of what constitutes “sufficiently high order” depends on the timestep of the numerical simulation. For moderate sized timesteps and lower orders, interpolated modified Hamiltonians are better conserved than truncated modified Hamiltonians of the same order. The number of low orders for which the interpolated modified Hamiltonians are better conserved than the truncated ones decreases as the timestep decreases.

Not only does conservation of truncated modified Hamiltonians tend to be better than that of interpolated modified Hamiltonians, but the rate of improvement in conservation as a function of increasing order is also faster for the truncated modified Hamiltonians. Let $P = C_{2k+2}/C_{2k}$ where C_i is the conservation of the i^{th} order modified Hamiltonian. In almost all cases P is smaller for the truncated, not interpolated, modified Hamiltonians for the double well and Hénon-Heiles potentials for $1 \leq k \leq 4$.

2.5 Smoothness of the potential

The cases of the C^1 continuous, piecewise defined potential given previously and molecular dynamics with a switching distance of 4 Å stand alone among the potentials used in this study in that they are not infinitely differentiable. Both highlight one of the benefits of the construction of interpolated modified Hamiltonians, namely the independence of that construction from many of the details of the Hamiltonian system to be solved.

As revealed by figures C.5 and C.6, the lack of smoothness in the piecewise potential has a clearly deleterious effect on the conservation of interpolated modified Hamiltonians. The lower bound on the conservation of these modified Hamiltonians is much larger relative to the conservation of the actual energy than for other, smoother potentials (figures C.7 and C.8). In fact, the conservation of interpolated modified Hamiltonians is only somewhat better than the conservation of the energy, and this improves only very slightly as the order is increased.

In the case of molecular dynamics, setting the switching distance to be just 4 Å, small enough that the Lennard-Jones potential is only C^1 continuous, also has a harmful effect on the conservation of interpolated modified Hamiltonians. In fact, for a 10 ps simulation with timestep 0.25 fs the conservation of the 24^{th} order interpolated modified Hamiltonian is approximately 3.5 times worse than when the Lennard-Jones potential is infinitely differentiable.

2.6 Trajectory accuracies

Another point of comparison between the truncated and interpolated modified Hamiltonians is which better tracks the numerical trajectory. The truncated and interpolated types of the modified equations were solved to high accuracy[2] to produce trajectories associated with the two different types of modified equations for the Hénon-Heiles Hamiltonian. These trajectories were compared to the numerical trajectory calculated by the leapfrog method and evaluated according to the 2-norm

[2]Integration was by the 4^{th} order Runge-Kutta method.

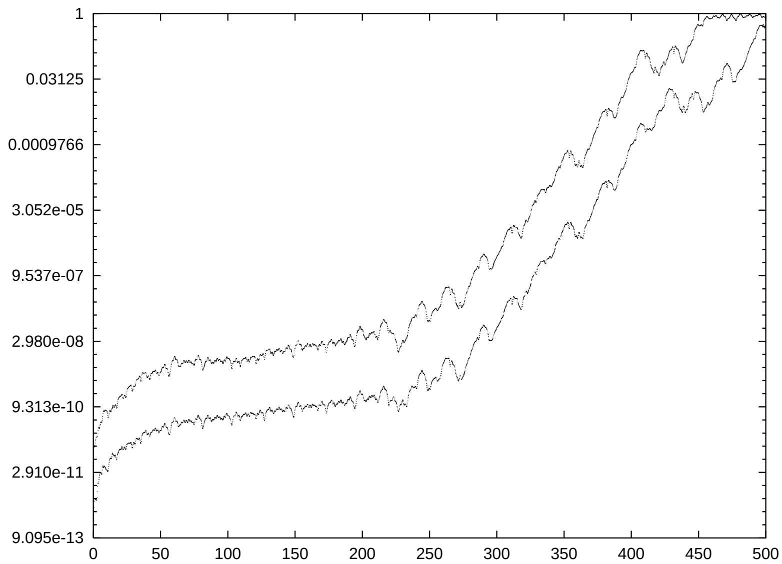

Figure 2.6: 12^{th} order trajectory discrepancies for Hénon-Heiles

of the discrepancy between the numerical trajectory and the modified trajectory at each timestep, where the discrepancy was calculated as the difference between these two trajectories.[3]

The numerical trajectories separate from the trajectories associated with the modified equations, both interpolated and truncated, after some number of steps, with separation in the sense that root mean squared distance and maximum error at any one step are both significant fractions of the total range of the numerical trajectory. As expected, trajectories associated with modified equations of higher order track the numerical solution for a great number of steps than do trajectories associated with modified equations of lower orders, at least up to 12^{th} order (the limit to which this study was carried out).

Figure 2.6 plots the trajectory discrepancies for the 12^{th} order truncated and interpolated modified Hamiltonians. The discrepancy associated with the interpolated modified Hamiltonian is the higher of the two lines. For formulas of orders $2, 4, \ldots, 12$ the trajectory discrepancy[4] is always less for the truncated modified Hamiltonians rather than for the interpolated modified Hamiltonians. As expected from theory, the discrepancy between the modified trajectories and the

[3]Trajectories for modified equations were calculated with an error of less than 5%. An estimate of the error in the modified trajectories was found at each timestep by taking the 2-norm of $1/15^{th}$ the difference between the trajectory calculated with a timestep of h_{RK} and the trajectory calculated with a timestep of $h_{\mathrm{RK}}/2$.

[4]With a simulation timestep of 0.2, simulation length of 500 time units, and Runge-Kutta timestep of 0.00015625 the estimated error in the modified trajectories is less than 5% for both the truncated and interpolated equations over the entire length of the simulation.

numerical trajectory, plotted in figure C.9, decreases exponentially as a function of interpolation degree k used to create the modified trajectory.

Chapter 3

Interpolated modified Hamiltonians

3.1 Augmented integrator

Construction of interpolated modified Hamiltonians proceeds along the same lines as presented in [7]. For some given Hamiltonian H, let Φ_h be an integrator where one step of size h of Φ_h is the composition of exact h-flows for Hamiltonian systems $H = H_1 + H_2 + \cdots + H_L$. Assume that each Hamiltonian $H_l(x)$ is sufficiently smooth on some domain containing the infinite time trajectory. For instance, for separable Hamiltonian systems with $H(q,p) = K(p) + U(q)$ the leapfrog method has $L = 3$ and $H_1(x) = \frac{1}{2}U(q)$, $H_2(x) = K(p)$, and $H_3(x) = \frac{1}{2}U(q)$. Define the homogeneous extension of the Hamiltonian by

$$\bar{H}(q,\alpha,p,\beta) \stackrel{\text{def}}{=} \alpha^2 H(\alpha^{-1}q, \alpha^{-1}p),$$

where $y \stackrel{\text{def}}{=} [q^\mathsf{T}, \alpha, p^\mathsf{T}, \beta]^\mathsf{T}$. Then, as shown in [7],

$$H(x(t)) \equiv \frac{1}{2}\dot{y}(t)^\mathsf{T}\bar{J}y(t),$$

where $y(t)$ is the solution of the homogeneous extension $\bar{H}(y)$ of $H(x)$ with α initially 1 and with $\bar{J}$ equal to J but of augmented dimension. Define the augmented method $y^{n+1} = \Psi_h(y^n)$ for $\bar{H}$ to be the composition of exact flows for systems with Hamiltonians $\bar{H}_1, \bar{H}_2, \ldots, \bar{H}_L$ where

$$\bar{H}_l(q,\alpha,p,\beta) \stackrel{\text{def}}{=} \alpha^2 H_l(\alpha^{-1}q, \alpha^{-1}p).$$

For example, the augmented leapfrog method with $\alpha^0 = 1$ is given by

$$\begin{aligned}
p^{n+1/2} &= p^n + \frac{h}{2} F^n, \\
\beta^{n+1/2} &= \beta^n + \frac{h}{2}(-(q^n)^\top F^n - 2U^n), \\
q^{n+1} &= q^n + hM^{-1} p^{n+1/2}, \\
F^{n+1} &= F(q^{n+1}), \\
p^{n+1} &= p^{n+1/2} + \frac{h}{2} F^{n+1}, \\
\beta^{n+1} &= \beta^{n+1/2} + \frac{h}{2}(-(q^{n+1})^\top F^{n+1} - 2U^{n+1}).
\end{aligned}$$

Let $H_h(q,p)$ be the modified Hamiltonian of the original method Φ_h. Then it can be shown [7] that the augmented method Ψ_h has a modified Hamiltonian

$$\bar{H}_h(q,\alpha,p,\beta) = \alpha^2 H_h(\alpha^{-1}q, \alpha^{-1}p).$$

Furthermore, let the modified Hamiltonians H_h and $\bar{H}_h$ have solutions $x_h(t)$ and $y_h(t)$ respectively. Then it is shown in [7] that

$$H_h(x_h(t)) \equiv \frac{1}{2}\dot{y}_h(t)^\top \bar{J} y_h(t).$$

3.2 Construction of $H_{[2k]}$ for even k

The construction of $H_{[2k]}(q,p)$ for *even values of* k follows. Given a modified extended Hamiltonian system with initial condition $y_h(0) = [q^\top, 1, p^\top, 0]^\top$, let $y_h(t)$ be its solution with values $y_h(jh) = \Psi_h^j(y_h(0))$, $j = 0, \pm 1, \ldots, \pm k/2$. Let $\pi_k(t)$ be the degree k polynomial interpolant of these $k+1$ values. Following the method of [7], let

$$H_{k,j} \stackrel{\text{def}}{=} \frac{1}{jh}\int_{-jh/2}^{jh/2} \frac{1}{2}\dot{\pi}_k(t)^\top J \pi_k(t)\mathrm{d}t, \qquad j = 2, 4, \ldots, k.$$

As shown in [7], this can be expressed as an expansion

$$H_{k,j} = \bar{H}_h + c_{j1}h^{k+2}\dot{\gamma}(0) + c_{j3}h^{k+4}\dddot{\gamma}(0) + \cdots + \mathcal{O}(h^{2k+2}),$$

where

$$\gamma(t) \quad \stackrel{\text{def}}{=} \quad \frac{1}{2}\dot{y}_h(t)^\mathsf{T}\bar{J}y_h^{[k+1]}(t)$$

with the brackets denoting a $(k+1)^{th}$ divided difference, so that $y_h^{[k+1]}(t) \stackrel{\text{def}}{=} y_h[-\frac{1}{2}kh, \ldots, \frac{1}{2}kh, t]$ As in [7], the first $k/2 - 1$ leading error terms can be eliminated by forming a suitable linear combination of the $H_{k,j},\ j = 1, 2, \ldots, k/2$, to yield

$$H_{[2k]} \quad \stackrel{\text{def}}{=} \quad \text{some linear combination of the } H_{k,j} \quad = \quad \bar{H}_h + \mathcal{O}(h^{2k}).$$

The coefficients of these linear combinations are given in appendix A. A concrete example for the case of $k = 4$ is

$$H_{[8]}(x) \quad = \quad \frac{16}{21}H_{4,2} + \frac{5}{21}H_{4,4} \quad = \quad \bar{H}_h(y) + \mathcal{O}(h^8).$$

3.3 Construction of $H_{[2k]}$ for odd k

The construction of $H_{[2k]}(q,p)$ for *odd values of* k differs somewhat from the above. Sensible midstep values

$$y^{n+1/2}, y^{n+3/2}, \ldots, y^{n+k/2}$$

can be defined for most numerical integrators. These midstep values can then be used to get an estimate for the modified Hamiltonian accurate up to $\mathcal{O}(h^{2k})$. Assume that $\Psi_h = \hat{\Psi}^{-1}_{-h/2} \circ \hat{\Psi}_{h/2}$, with $\hat{\Psi}_{h/2}$ the composition of exact flows of homogeneously extended Hamiltonians. For example, using leapfrog integration, $\hat{\Psi}_{h/2}$ equals

$$\begin{aligned}
p^{n+1/2} &= p^n + \frac{h}{2}F^n, \\
\beta^{n+1/2} &= \beta^n + \frac{h}{2}(-(q^n)^\mathsf{T}F^n - 2U^n), \\
q^{n+1/2} &= q^n + \frac{h}{2}M^{-1}p^{n+1/2}.
\end{aligned}$$

Assume that $z_h(t) = \hat{\Psi}_{h/2}(y_h(t-h/2))$, so that the midpoint values of $z_h(t)$ are the midstep values of the integrator. Then this can be used to construct the approximation to the modified Hamiltonian with, as shown in [7],

$$H_h(x_h(t)) \quad \equiv \quad \frac{1}{2}\dot{z}_h(t)^{\mathsf{T}}\bar{J}z_h(t).$$

Let $\pi_k(t)$ be the degree k polynomial interpolant of the $k+1$ values $z_h(jh) = \hat{\Psi}_{h/2} \circ \Psi_h^{j-1/2}(y)$, $j = \pm\frac{1}{2}, \pm\frac{3}{2}, \ldots, \pm\frac{k}{2}$. Again, as in [7], let

$$H_{k,j} \quad = \quad \frac{1}{jh}\int_{-jh/2}^{jh/2} \frac{1}{2}\dot{\pi}_k(t)^{\mathsf{T}} J\pi_k(t)\mathrm{d}t, \qquad j = 1, 3, \ldots, k.$$

It can be shown [7] that this can be expressed as an expansion

$$H_{k,j} \quad = \quad \bar{H}_h + c_{j0}h^{k+1}\gamma(0) + c_{j2}h^{k+3}\ddot{\gamma}(0) + \cdots + \mathcal{O}(h^{2k+2}),$$

where $\gamma(t) \stackrel{\text{def}}{=} \frac{1}{2}\dot{z}_h(t)^{\mathsf{T}}\bar{J}z_h^{[k+1]}(t)$, with, as before, the brackets denoting a $(k+1)^{th}$ divided difference. As proved in [7], the first $(k+1)/2-1$ leading error terms can be eliminated by forming a suitable linear combination of the $(k+1)/2$ different $H_{k,j}$ to yield

$$H_{[2k]} \quad \stackrel{\text{def}}{=} \quad \text{some linear combination of the } H_{k,j} \quad = \quad \bar{H}_h + \mathcal{O}(h^{2k}).$$

The coefficients of these linear combinations can be found in appendix A. To take a single example, for the case of $k = 3$,

$$H_{[6]}(x) \quad = \quad \frac{9}{20}H_{3,1} + \frac{11}{20}H_{3,3} \;\; = \;\; \bar{H}_h(y) + \mathcal{O}(h^6).$$

Chapter 4

Backward difference implementation

4.1 Backward difference derivation for even k

This section presents an implementation of $H_{[2k]}$ for even values of k. Let a_j be the j^{th} backward difference of $y_h(t)$ at $t = (k/2)h$,

$$a_j = \nabla^j y_h((k/2)h), \quad j = 0, 1, 2, \ldots, k,$$

where the backward difference is defined by $\nabla^k w(t) = \nabla^{k-1} w(t) - \nabla^{k-1} w(t-h)$, $\nabla^0 w(t) = w(t)$. Taking the case $k = 4$ as an example, the fourth degree Newton interpolant using backward differences is

$$\begin{aligned} \pi_4(t) &= y_h(2h) + (t-2h)\frac{\nabla y_h(2h)}{h} + (t-2h)(t-h)\frac{\nabla^2 y_h(2h)}{2h^2} \\ &\quad + t(t-2h)(t-h)\frac{\nabla^3 y_h(2h)}{6h^3} + t(t-2h)(t^2-h^2)\frac{\nabla^4 y_h(2h)}{24h^4}. \end{aligned}$$

Thus

$$\pi_4(sh) = a_0 + (s-2)a_1 + \frac{1}{2}(s-2)(s-1)a_2 + \frac{1}{6}s(s-2)(s-1)a_3 + \frac{1}{24}s(s-2)(s^2-1)a_4$$

and

$$h\dot{\pi}_4(sh) = a_1 + (s-\frac{3}{2})a_2 + (\frac{1}{2}s^2 - s + \frac{1}{3})a_3 + \frac{1}{2}(\frac{1}{3}s^3 - \frac{1}{2}s^2 - \frac{1}{6}s + \frac{1}{6})a_4.$$

Define $A_{ij} = a_i^\mathsf{T} \bar{J} a_j/(2h)$. Using the facts that $A_{ii} = 0$ and $A_{ij} = -A_{ji}$, this yields

$$\begin{aligned}
\frac{1}{2}\dot{\pi}_4(sh)^\mathsf{T}\bar{J}\pi_4(sh) &= A_{10} - \frac{3}{2}A_{20} + (2 + \frac{1}{2}s^2)A_{21} \\
&\quad + (\frac{1}{3} + \frac{1}{2}s^2)A_{30} - (\frac{2}{3} + \frac{3}{2}s^2)A_{31} + (\frac{1}{3} + \frac{13}{12}s^2 + \frac{1}{12}s^4)A_{32} \\
&\quad + (\frac{1}{12} - \frac{1}{4}s^2)A_{40} + (-\frac{1}{6} + \frac{11}{24}s^2 + \frac{1}{8}s^4)A_{41} + (\frac{1}{12} - \frac{11}{48}s^2 - \frac{11}{48}s^4)A_{42} \\
&\quad + (\frac{1}{36}s^2 + \frac{13}{144}s^4 + \frac{1}{144}s^6)A_{43} + \text{odd powers of } s
\end{aligned}$$

Average over $-1 \le s \le 1$ to obtain $H_{4,2}$ and over $-2 \le s \le 2$ to obtain $H_{4,4}$, as

$$H_{4,2} = \frac{1}{2}\int_{-1}^{1} \frac{1}{2}\dot{\pi}_4(sh)^\mathsf{T}\bar{J}\pi_4(sh)\mathrm{d}t \quad \text{and} \quad H_{4,4} = \frac{1}{4}\int_{-2}^{2} \frac{1}{2}\dot{\pi}_4(sh)^\mathsf{T}\bar{J}\pi_4(sh)\mathrm{d}t.$$

From this one finds

$$\begin{aligned}
H_{[8]} &= \frac{16}{21}H_{4,2} + \frac{5}{21}H_{4,4} \\
&= A_{10} - \frac{3}{2}A_{20} + \frac{16}{7}A_{21} + \frac{13}{21}A_{30} - \frac{32}{21}A_{31} + \frac{36}{35}A_{32} \\
&\quad - \frac{5}{84}A_{40} + \frac{22}{105}A_{41} - \frac{9}{35}A_{42} + \frac{4}{35}A_{43}.
\end{aligned}$$

This and other formulas in terms of the A_{ij} for $H_{[2k]}$ with k even can be found in appendix A.

4.2 Backward difference derivation for odd k

This section presents an implementation of $H_{[2k]}$ for odd values of k. Let b_j be the j^{th} backward difference of $z_h(t)$ at $t = (k/2)h$,

$$b_j = \nabla^j z_h((k/2)h), \quad j = 0, 1, 2, \ldots, k.$$

The case of $k = 3$ is used as a representative example. The third degree Newton interpolant using backward differences is

$$
\begin{aligned}
\pi_3(t) &= z_h(3h/2) + (t - 3h/2)\frac{\nabla z_h(3h/2)}{h} + (t - 3h/2)(t - \frac{1}{2}h)\frac{\nabla^2 z_h(3h/2)}{2h^2} \\
&\quad +(t - 3h/2)(t^2 - \frac{1}{4}h^2)\frac{\nabla^3 z_h(3h/2)}{6h^3}
\end{aligned}
$$

Thus

$$
\pi_3(sh) = b_0 + (s - \frac{3}{2})b_1 + \frac{1}{2}(s - \frac{3}{2})(s - \frac{1}{2})b_2 + \frac{1}{6}(s - \frac{3}{2})(s^2 - \frac{1}{4})b_3
$$

and

$$
h\dot{\pi}_3(sh) = b_1 + (s - 1)b_2 + (\frac{1}{2}s^2 - \frac{1}{2}s - \frac{1}{24})b_3
$$

Define $B_{ij} = b_i^{\mathsf{T}} \bar{J} b_j/(2h)$. This yields

$$
\begin{aligned}
\frac{1}{2}\dot{\pi}_3(sh)^{\mathsf{T}} \bar{J} \pi_3(sh) &= B_{10} - B_{20} + (\frac{1}{2}s^2 + \frac{9}{8})B_{21} \\
&\quad + (\frac{1}{2}s^2 \ \ \frac{1}{24})B_{30} - s^2 B_{31} + (\frac{3}{64} + \frac{11}{24}s^2 + \frac{1}{12}s^4)B_{32} \\
&\quad + \text{odd powers of } s,
\end{aligned}
$$

using the facts that $B_{ii} = 0$ and $B_{ij} = -B_{ji}$. Average over $-\frac{1}{2} \le s \le \frac{1}{2}$ to obtain $H_{3,1}$ and over $-\frac{3}{2} \le s \le \frac{3}{2}$ to obtain $H_{3,3}$. From this one finds

$$
H_{[6]} = B_{10} - B_{20} + \frac{27}{20}B_{21} + \frac{11}{60}B_{30} - \frac{9}{20}B_{31} + \frac{3}{10}B_{32}.
$$

This and other formulas in terms of the B_{ij} for $H_{[2k]}$ with k odd can be found in appendix A.

4.3 Storage considerations

The efficiency of the construction of interpolated modified Hamiltonians as presented in [7] and reviewed in chapter 3 derives from the fact that the interpolated modified Hamiltonians $H_{[2k]}$ are built from consecutive solution values $y_h^{n-k/2}, y_h^{n-k/2+1}, \ldots, y_h^{n+k/2-1}, y_h^{n+k/2}$, values which are already being calculated by the simulation. Differences of these values are formed, which represent the interpolants, and these differences are used to construct the interpolated modified Hamiltonians. The introduction of these differences of solution values for construction of interpolated modified Hamiltonians requires additional storage beyond what is necessary for the simulation alone, as difference tables must be stored. The scheme given in [7] employs centered differences. However, the use of backward differences, as presented in this thesis, results in significant savings in storage costs. Consider the difference tables up to fourth order differences, which are required for calculating $H_{[8]}$, for both centered and backward differences. The centered difference table is

$$\begin{array}{lllll}
\delta^0 y_h(t-2h) & & & & \\
 & \delta^1 y_h(t-\frac{3h}{2}) & & & \\
\delta^0 y_h(t-h) & & \delta^2 y_h(t-h) & & \\
 & \underline{\underline{\delta^1 y_h(t-\frac{h}{2})}} & & \underline{\underline{\delta^3 y_h(t-\frac{h}{2})}} & \\
\underline{\underline{\delta^0 y_h(t)}} & & \underline{\underline{\delta^2 y_h(t)}} & & \underline{\underline{\delta^4 y_h(t)}} \\
 & \underline{\underline{\delta^1 y_h(t+\frac{h}{2})}} & & \underline{\underline{\delta^3 y_h(t+\frac{h}{2})}} & \\
\underline{\delta^0 y_h(t+h)} & & \underline{\delta^2 y_h(t+h)} & & \\
 & \underline{\delta^1 y_h(t+\frac{3h}{2})} & & & \\
\underline{\delta^0 y_h(t+2h)} & & & &
\end{array}$$

The backward difference table is

$$\begin{array}{lllll}
\nabla^0 y_h(t-2h) & & & & \\
 & \nabla^1 y_h(t-h) & & & \\
\nabla^0 y_h(t-h) & & \nabla^2 y_h(t) & & \\
 & \nabla^1 y_h(t) & & \nabla^3 y_h(t+h) & \\
\nabla^0 y_h(t) & & \nabla^2 y_h(t+h) & & \underline{\underline{\nabla^4 y_h(t+2h)}} \\
 & \nabla^1 y_h(t+h) & & \underline{\underline{\nabla^3 y_h(t+2h)}} & \\
\nabla^0 y_h(t+h) & & \underline{\underline{\nabla^2 y_h(t+2h)}} & & \\
 & \underline{\underline{\nabla^1 y_h(t+2h)}} & & & \\
\underline{\underline{\nabla^0 y_h(t+2h)}} & & & &
\end{array}$$

Formulas for the interpolated modified Hamiltonians are in terms of the doubly underlined values. However, for purposes of calculating these doubly underlined values, both these and the singly underlined values must be stored. Thus, to compute $H_{[8]}$ using centered differences, eleven sets of values must be stored in addition to the storage already required for the simulation, while to compute $H_{[8]}$ using backward differences only 4 additional sets of values must be stored. The value $\nabla^0 y_h(t+2h)$ is "free," since this can be obtained from the values of q, p, and β at the most recent timestep.

The centered difference table requires storage that is quadratic in k, while the backward difference table requires storage that is only linear in k. Let $\widetilde{N}$ be the number of degrees of freedom of the system being simulated, for example $\widetilde{N} = 6N$ for a system of N atoms with 3 position and 3 momentum coordinates per atom. Then the storage requirements for the two schemes are

$$\begin{array}{lll} \text{centered differences:} & (\widetilde{N}+1)\sum_{i=1}^{k+1}(1+\lfloor i/2 \rfloor) & \text{is } \mathcal{O}(k^2) \\ \text{backward differences:} & (\widetilde{N}+1)k & \text{is } \mathcal{O}(k) \end{array}$$

where k determines the maximum order of shadow Hamiltonian $H_{[2k]}$ to be computed. The $+1$ in the $(\widetilde{N}+1)$ terms arises because divided differences of the scalar extended coordinate β must also be stored. Thus the savings is especially significant for systems with large $\widetilde{N}$ and for high order shadow Hamiltonians.

4.4 Rounding error considerations

In practice, implementation of interpolated modified Hamiltonians $H_{[2k]}$ for odd k is unnecessary when the goal is to obtain the formulas with highest order accuracy possible, which, as this thesis shows, exhibit the best conservation behavior. Implementation of only the even k case is sufficient for this purpose, and has the advantage that extra work needed to compute differences of midstep quantities is avoided. For this reason the following discussion is limited to the case of even k.

Using leapfrog integration, the key backward differences of fullstep quantities from which higher order differences are calculated are as follows: $\nabla q^{n+1} = hM^{-1}p^{n+1/2}$, $\nabla^2 q^{n+1} = h^2 M^{-1} F^n$, calculated before the leapfrog force update; $\nabla^i \alpha = 1$ for $i = 0$, $\nabla^i \alpha = 0$ for $i \neq 0$ which need not be stored nor computed; $\nabla p^{n+1} = \frac{1}{2}h(F^{n+1} + F^n)$, which can be calculated during the leapfrog force update so that new and old force values are available; and $\nabla\beta^{n+1} = \frac{1}{2}h(-(q^n)^\mathsf{T}F^n - 2U^n - (q^{n+1})^\mathsf{T}F^{n+1} - 2U^{n+1})$, the first part of which may be calculated before the leapfrog position update using the values q^n, F^n, U^n and the second part of which may be calculated after the leapfrog force update using the values $q^{n+1}, F^{n+1}, U^{n+1}$.

Use of the preceding expressions reduces rounding error since the first and second order backward differences of key fullstep simulation quantities are found analytically, rather than numerically. Directly calculating low order differences avoids harmful cancellation, thus avoiding unnecessary rounding error.

As before, let k determine the maximum order of shadow Hamiltonian $H_{[2k]}$ to be computed. A new row of the backward difference table consists of the values

$$\nabla^i y^{n+1} = [(\nabla^i q^{n+1})^\mathsf{T}, \nabla^i \alpha^{n+1}, (\nabla^i p^{n+1})^\mathsf{T}, \nabla^i \beta^{n+1}]^\mathsf{T}, \quad i = 0, 1, \ldots, k.$$

Presented in the context of a single step of the leapfrog method these updates take place as follows:

$$\begin{aligned}
p^{n+1/2} &= p^n + \frac{h}{2}F^n, \\
\nabla q^{n+1} &= hM^{-1}p^{n+1/2}, \\
\nabla^2 q^{n+1} &= h^2 M^{-1} F^n \quad \text{and} \quad \nabla^i q^{n+1} \;=\; \nabla^{i-1} q^{n+1} - \nabla^{i-1} q^n, \quad i = 3, 4, \ldots, k, \\
\nabla \beta^n_{\text{half}} &= \frac{h}{2}(-(q^n)^\mathsf{T} F^n - 2U^n), \\
q^{n+1} &= q^n + hM^{-1}p^{n+1/2}, \\
\nabla p^{n+1}_{\text{half}} &= \frac{1}{2}hF^n, \\
F^{n+1} &= F(q^{n+1}), \\
\nabla p^{n+1} &= \nabla p^{n+1}_{\text{half}} + \frac{1}{2}hF^{n+1} \quad \text{and} \quad \nabla^i p^{n+1} \;=\; \nabla^{i-1} p^{n+1} - \nabla^{i-1} p^n, \quad i = 2, 3, \ldots, k, \\
p^{n+1} &= p^{n+1/2} + \frac{h}{2}F^{n+1}, \\
\nabla \beta^{n+1} &= \nabla \beta^n_{\text{half}} + \frac{h}{2}(-(q^{n+1})^\mathsf{T} F^{n+1} - 2U^{n+1}) \quad \text{and} \quad \nabla^i \beta^{n+1} \;=\; \nabla^{i-1}\beta^{n+1} - \nabla^{i-1}\beta^n, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 2, 3, \ldots, k.
\end{aligned}$$

Only differences of β, not β itself, need to be computed, since only these are necessary to calculate interpolated modified Hamiltonian formulas. This is the case since all differences of α of first order and above are zero, so terms involving 0^{th} order differences of β drop out.

As constructed, the interpolated modified Hamiltonians are defined by linear combinations of A_{ij} and B_{ij} terms. As the order of interpolated modified Hamiltonians increases these formulas quickly become very long and impractical to code by hand. Instead, the coefficients of the A_{ij} and B_{ij} terms in the various $H_{[2k]}$ equations can be stored in arrays initialized before integration begins and then referenced as needed, simplifying the expansion to higher values of k. See, for example, the code in appendix B.

References

[1] J. Gans and D. Shalloway. Shadow mass and the relationship between velocity and momentum in symplectic numerical integration. *Phys. Rev. E*, 61(4):4587–4592, Apr. 2000.

[2] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*, volume 31 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 2002.

[3] M. Hénon and C. Heiles. The applicability of the third integral of motion: Some numerical experiments. *Astron. J.*, 69(1):73–79, 1964.

[4] L. Kalé, R. Skeel, R. Brunner, M. Bhandarkar, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten. NAMD2: Greater scalability for parallel molecular dynamics. *J. Comput. Phys*, 151(1):283–312, May 1, 1999.

[5] S. Reich. Backward error analysis for numerical integrators. *SIAM J. Numer. Anal.*, 36(5):1549–1570, 1999.

[6] J. Sanz-Serna and M. Calvo. *Numerical Hamiltonian Problems.* Chapman and Hall, London, 1994.

[7] R. D. Skeel and D. J. Hardy. Practical construction of modified Hamiltonians. *SIAM J. Sci. Comput.*, 23(4):1172–1188, Nov. 9, 2001.

Appendix A

Interpolated modified Hamiltonians up to order 24

Below are given the coefficients for construction of interpolated modified Hamiltonians $H_{[2k]}$ for $k = 1, 2, \ldots, 12$ using backward differences.

$$H_{[2] } = B_{1,0},$$

$$H_{[4]} = A_{1,0} - \frac{1}{2}\, A_{2,0} + \frac{2}{3}\, A_{2,1},$$

$$H_{[6]} = B_{1,0} - B_{2,0} + \frac{27}{20}\, B_{2,1} + \frac{11}{60}\, B_{3,0} - \frac{9}{20}\, B_{3,1} + \frac{3}{10}\, B_{3,2},$$

$$\begin{aligned} H_{[8]} = {} & A_{1,0} - \frac{3}{2}\, A_{2,0} + \frac{16}{7}\, A_{2,1} + \frac{13}{21}\, A_{3,0} - \frac{32}{21}\, A_{3,1} + \frac{36}{35}\, A_{3,2} - \frac{5}{84}\, A_{4,0} + \frac{22}{105}\, A_{4,1} \\ & - \frac{9}{35}\, A_{4,2} + \frac{4}{35}\, A_{4,3}, \end{aligned}$$

$$\begin{aligned} H_{[10]} = {} & B_{1,0} - 2\ B_{2,0} + \frac{125}{36}\, B_{2,1} + \frac{47}{36}\, B_{3,0} - \frac{125}{36}\, B_{3,1} + \frac{500}{189}\, B_{3,2} - \frac{11}{36}\, B_{4,0} + \frac{1625}{1512}\, B_{4,1} \\ & - \frac{250}{189}\, B_{4,2} + \frac{25}{42}\, B_{4,3} + \frac{137}{7560}\, B_{5,0} - \frac{125}{1512}\, B_{5,1} + \frac{55}{378}\, B_{5,2} - \frac{5}{42}\, B_{5,3} + \frac{5}{126}\, B_{5,4}, \end{aligned}$$

$$\begin{aligned} H_{[12]} = {} & A_{1,0} - \frac{5}{2}\, A_{2,0} + \frac{54}{11}\, A_{2,1} + \frac{74}{33}\, A_{3,0} - \frac{72}{11}\, A_{3,1} + \frac{125}{22}\, A_{3,2} - \frac{19}{22}\, A_{4,0} + \frac{141}{44}\, A_{4,1} - \frac{375}{88}\, A_{4,2} \\ & + \frac{500}{231}\, A_{4,3} + \frac{29}{220}\, A_{5,0} - \frac{3}{5}\, A_{5,1} + \frac{325}{308}\, A_{5,2} - \frac{200}{231}\, A_{5,3} + \frac{45}{154}\, A_{5,4} - \frac{7}{1320}\, A_{6,0} + \frac{137}{4620}\, A_{6,1} \\ & - \frac{125}{1848}\, A_{6,2} + \frac{5}{63}\, A_{6,3} - \frac{15}{308}\, A_{6,4} + \frac{1}{77}\, A_{6,5}, \end{aligned}$$

$$
\begin{aligned}
H_{[14]} &= B_{1,0} - 3\ B_{2,0} + \frac{343}{52}B_{2,1} + \frac{535}{156}B_{3,0} - \frac{1715}{156}B_{3,1} + \frac{3087}{286}B_{3,2} - \frac{145}{78}B_{4,0} + \frac{12691}{1716}B_{4,1} \\
&\quad - \frac{3087}{286}B_{4,2} + \frac{42875}{6864}B_{4,3} + \frac{1377}{2860}B_{5,0} - \frac{6517}{2860}B_{5,1} + \frac{48363}{11440}B_{5,2} - \frac{8575}{2288}B_{5,3} + \frac{1225}{858}B_{5,4} \\
&\quad - \frac{223}{4290}B_{6,0} + \frac{9947}{34320}B_{6,1} - \frac{343}{520}B_{6,2} + \frac{1225}{1584}B_{6,3} - \frac{1225}{2574}B_{6,4} + \frac{147}{1144}B_{6,5} + \frac{11}{7280}B_{7,0} \\
&\quad - \frac{343}{34320}B_{7,1} + \frac{959}{34320}B_{7,2} - \frac{875}{20592}B_{7,3} + \frac{35}{936}B_{7,4} - \frac{21}{1144}B_{7,5} + \frac{7}{1716}B_{7,6},
\end{aligned}
$$

$$
\begin{aligned}
H_{[16]} &= A_{1,0} - \frac{7}{2}A_{2,0} + \frac{128}{15}A_{2,1} + \frac{73}{15}A_{3,0} - \frac{256}{15}A_{3,1} + \frac{10976}{585}A_{3,2} - \frac{41}{12}A_{4,0} + \frac{1712}{117}A_{4,1} \\
&\quad - \frac{2744}{117}A_{4,2} + \frac{10976}{715}A_{4,3} + \frac{743}{585}A_{5,0} - \frac{3712}{585}A_{5,1} + \frac{406112}{32175}A_{5,2} - \frac{43904}{3575}A_{5,3} + \frac{6860}{1287}A_{5,4} \\
&\quad - \frac{31}{130}A_{6,0} + \frac{4896}{3575}A_{6,1} - \frac{104272}{32175}A_{6,2} + \frac{128968}{32175}A_{6,3} - \frac{3430}{1287}A_{6,4} + \frac{3136}{3861}A_{6,5} + \frac{37}{1925}A_{7,0} \\
&\quad - \frac{28544}{225225}A_{7,1} + \frac{11368}{32175}A_{7,2} - \frac{1568}{2925}A_{7,3} + \frac{140}{297}A_{7,4} - \frac{896}{3861}A_{7,5} + \frac{112}{2145}A_{7,6} - \frac{761}{1801800}A_{8,0} \\
&\quad + \frac{22}{6825}A_{8,1} - \frac{343}{32175}A_{8,2} + \frac{1918}{96525}A_{8,3} - \frac{175}{7722}A_{8,4} + \frac{28}{1755}A_{8,5} - \frac{14}{2145}A_{8,6} + \frac{8}{6435}A_{8,7},
\end{aligned}
$$

$$
\begin{aligned}
H_{[18]} &= B_{1,0} - 4\ B_{2,0} + \frac{729}{68}B_{2,1} + \frac{1337}{204}B_{3,0} - \frac{1701}{68}B_{3,1} + \frac{2592}{85}B_{3,2} - \frac{385}{68}B_{4,0} + \frac{17739}{680}B_{4,1} \\
&\quad - \frac{3888}{85}B_{4,2} + \frac{37044}{1105}B_{4,3} + \frac{1879}{680}B_{5,0} - \frac{9963}{680}B_{5,1} + \frac{34668}{1105}B_{5,2} - \frac{37044}{1105}B_{5,3} \\
&\quad + \frac{1000188}{60775}B_{5,4} - \frac{193}{255}B_{6,0} + \frac{20061}{4420}B_{6,1} - \frac{12528}{1105}B_{6,2} + \frac{913752}{60775}B_{6,3} - \frac{666792}{60775}B_{6,4} \\
&\quad + \frac{9261}{2431}B_{6,5} + \frac{197}{1820}B_{7,0} - \frac{22599}{30940}B_{7,1} + \frac{52488}{25025}B_{7,2} - \frac{201096}{60775}B_{7,3} + \frac{186543}{60775}B_{7,4} \\
&\quad - \frac{3969}{2431}B_{7,5} + \frac{1008}{2431}B_{7,6} - \frac{419}{61880}B_{8,0} + \frac{26973}{523600}B_{8,1} - \frac{72252}{425425}B_{8,2} + \frac{38367}{121550}B_{8,3} \\
&\quad - \frac{3969}{11050}B_{8,4} + \frac{189}{748}B_{8,5} - \frac{252}{2431}B_{8,6} + \frac{243}{12155}B_{8,7} + \frac{7129}{61261200}B_{9,0} - \frac{6849}{6806800}B_{9,1} \\
&\quad + \frac{297}{77350}B_{9,2} - \frac{1029}{121550}B_{9,3} + \frac{2877}{243100}B_{9,4} - \frac{105}{9724}B_{9,5} + \frac{7}{1105}B_{9,6} - \frac{27}{12155}B_{9,7} \\
&\quad + \frac{9}{24310}B_{9,8},
\end{aligned}
$$

$$
\begin{aligned}
H_{[20]} &= A_{1,0} - \frac{9}{2}A_{2,0} + \frac{250}{19}A_{2,1} + \frac{484}{57}A_{3,0} - \frac{2000}{57}A_{3,1} + \frac{30375}{646}A_{3,2} - \frac{497}{57}A_{4,0} + \frac{167125}{3876}A_{4,1} \\
&\quad - \frac{212625}{2584}A_{4,2} + \frac{21600}{323}A_{4,3} + \frac{34167}{6460}A_{5,0} - \frac{9625}{323}A_{5,1} + \frac{88695}{1292}A_{5,2} - \frac{25920}{323}A_{5,3} \\
&\quad + \frac{185220}{4199}A_{5,4} - \frac{14917}{7752}A_{6,0} + \frac{46975}{3876}A_{6,1} - \frac{83025}{2584}A_{6,2} + \frac{192600}{4199}A_{6,3} - \frac{154350}{4199}A_{6,4} \\
&\quad + \frac{666792}{46189}A_{6,5} + \frac{2761}{6783}A_{7,0} - \frac{19300}{6783}A_{7,1} + \frac{501525}{58786}A_{7,2} - \frac{417600}{29393}A_{7,3} + \frac{652680}{46189}A_{7,4} \\
&\quad - \frac{381024}{46189}A_{7,5} + \frac{110250}{46189}A_{7,6} - \frac{831}{18088}A_{8,0} + \frac{4925}{13832}A_{8,1} - \frac{564975}{470288}A_{8,2} + \frac{43740}{19019}A_{8,3}
\end{aligned}
$$

$$
\begin{aligned}
&-\frac{6615}{2431}\,A_{8,4}+\frac{186543}{92378}\,A_{8,5}-\frac{165375}{184756}\,A_{8,6}+\frac{9000}{46189}\,A_{8,7}+\frac{4861}{2116296}\,A_{9,0}-\frac{10475}{529074}\,A_{9,1}\\
&+\frac{14985}{198968}\,A_{9,2}-\frac{53520}{323323}\,A_{9,3}+\frac{21315}{92378}\,A_{9,4}-\frac{882}{4199}\,A_{9,5}+\frac{875}{7106}\,A_{9,6}-\frac{2000}{46189}\,A_{9,7}\\
&+\frac{675}{92378}\,A_{9,8}-\frac{671}{21162960}\,A_{10,0}+\frac{7129}{23279256}\,A_{10,1}-\frac{6849}{5173168}\,A_{10,2}+\frac{99}{29393}\,A_{10,3}\\
&-\frac{1029}{184756}\,A_{10,4}+\frac{2877}{461890}\,A_{10,5}-\frac{875}{184756}\,A_{10,6}+\frac{10}{4199}\,A_{10,7}-\frac{135}{184756}\,A_{10,8}+\frac{5}{46189}\,A_{10,9},
\end{aligned}
$$

$$
\begin{aligned}
H_{[22]} \;=\;& B_{1,0}-5\;B_{2,0}+\frac{1331}{84}\,B_{2,1}+\frac{299}{28}\,B_{3,0}-\frac{1331}{28}\,B_{3,1}+\frac{166375}{2394}\,B_{3,2}-\frac{89}{7}\,B_{4,0}+\frac{161051}{2394}\,B_{4,1}\\
&-\frac{166375}{1197}\,B_{4,2}+\frac{4492125}{36176}\,B_{4,3}+\frac{15797}{1710}\,B_{5,0}-\frac{94501}{1710}\,B_{5,1}+\frac{6355525}{46512}\,B_{5,2}-\frac{898425}{5168}\,B_{5,3}\\
&+\frac{239580}{2261}\,B_{5,4}-\frac{1201}{285}\,B_{6,0}+\frac{2165537}{77520}\,B_{6,1}-\frac{1830125}{23256}\,B_{6,2}+\frac{4372335}{36176}\,B_{6,3}-\frac{239580}{2261}\,B_{6,4}\\
&+\frac{195657}{4199}\,B_{6,5}+\frac{129805}{108528}\,B_{7,0}-\frac{2836361}{325584}\,B_{7,1}+\frac{62523725}{2279088}\,B_{7,2}-\frac{12278475}{253232}\,B_{7,3}\\
&+\frac{10681275}{205751}\,B_{7,4}-\frac{139755}{4199}\,B_{7,5}+\frac{45738}{4199}\,B_{7,6}-\frac{32891}{162792}\,B_{8,0}+\frac{3674891}{2279088}\,B_{8,1}-\frac{6422075}{1139544}\,B_{8,2}\\
&+\frac{74169975}{6584032}\,B_{8,3}-\frac{2894925}{205751}\,B_{8,4}+\frac{94017}{8398}\,B_{8,5}-\frac{22869}{4199}\,B_{8,6}+\frac{45375}{33592}\,B_{8,7}+\frac{42131}{2279088}\,B_{9,0}\\
&-\frac{368687}{2279088}\,B_{9,1}+\frac{6555175}{10456992}\,B_{9,2}-\frac{9283725}{6584032}\,B_{9,3}+\frac{49005}{24206}\,B_{9,4}-\frac{847}{442}\,B_{9,5}+\frac{39809}{33592}\,B_{9,6}\\
&-\frac{15125}{33592}\,B_{9,7}+\frac{15125}{176358}\,B_{9,8}-\frac{4307}{5697720}\,B_{10,0}+\frac{6469991}{888844320}\,B_{10,1}-\frac{2788445}{88884432}\,B_{10,2}\\
&+\frac{40293}{506464}\,B_{10,3}-\frac{26983}{205751}\,B_{10,4}+\frac{24563}{167960}\,B_{10,5}-\frac{9317}{83980}\,B_{10,6}+\frac{3025}{54264}\,B_{10,7}-\frac{3025}{176358}\,B_{10,8}\\
&+\frac{605}{235144}\,B_{10,9}+\frac{83711}{9777287520}\,B_{11,0}-\frac{81191}{888844320}\,B_{11,1}+\frac{78419}{177768864}\,B_{11,2}-\frac{8371}{6584032}\,B_{11,3}\\
&+\frac{3993}{1646008}\,B_{11,4}-\frac{539}{167960}\,B_{11,5}+\frac{1507}{503880}\,B_{11,6}-\frac{1375}{705432}\,B_{11,7}+\frac{605}{705432}\,B_{11,8}\\
&-\frac{55}{235144}\,B_{11,9}+\frac{11}{352716}\,B_{11,10},
\end{aligned}
$$

$$
\begin{aligned}
H_{[24]} \;=\;& A_{1,0}-\frac{11}{2}\,A_{2,0}+\frac{432}{23}\,A_{2,1}+\frac{905}{69}\,A_{3,0}-\frac{1440}{23}\,A_{3,1}+\frac{15972}{161}\,A_{3,2}-\frac{1635}{92}\,A_{4,0}+\frac{702}{7}\,A_{4,1}\\
&-\frac{35937}{161}\,A_{4,2}+\frac{665500}{3059}\,A_{4,3}+\frac{12126}{805}\,A_{5,0}-\frac{76896}{805}\,A_{5,1}+\frac{3865224}{15295}\,A_{5,2}-\frac{1064800}{3059}\,A_{5,3}\\
&+\frac{24257475}{104006}\,A_{5,4}-\frac{953}{115}\,A_{6,0}+\frac{126376}{2185}\,A_{6,1}-\frac{378004}{2185}\,A_{6,2}+\frac{6355525}{22287}\,A_{6,3}-\frac{8085825}{29716}\,A_{6,4}\\
&+\frac{6899904}{52003}\,A_{6,5}+\frac{45454}{15295}\,A_{7,0}-\frac{345888}{15295}\,A_{7,1}+\frac{19489833}{260015}\,A_{7,2}-\frac{7320500}{52003}\,A_{7,3}\\
&+\frac{118053045}{728042}\,A_{7,4}-\frac{41399424}{364021}\,A_{7,5}+\frac{4024944}{96577}\,A_{7,6}-\frac{8315}{12236}\,A_{8,0}+\frac{1168245}{208012}\,A_{8,1}\\
&-\frac{8509083}{416024}\,A_{8,2}+\frac{62523725}{1456084}\,A_{8,3}-\frac{331518825}{5824336}\,A_{8,4}+\frac{230715540}{4732273}\,A_{8,5}-\frac{2515590}{96577}\,A_{8,6}\\
&+\frac{705672}{96577}\,A_{8,7}+\frac{176005}{1872108}\,A_{9,0}-\frac{131564}{156009}\,A_{9,1}+\frac{3674891}{1092063}\,A_{9,2}-\frac{25688300}{3276189}\,A_{9,3}\\
&+\frac{222509925}{18929092}\,A_{9,4}-\frac{55582560}{4732273}\,A_{9,5}+\frac{752136}{96577}\,A_{9,6}-\frac{313632}{96577}\,A_{9,7}+\frac{136125}{193154}\,A_{9,8}
\end{aligned}
$$

$$
\begin{aligned}
&-\frac{2339}{328440}\,A_{10,0}+\frac{126393}{1820105}\,A_{10,1}-\frac{1106061}{3640210}\,A_{10,2}+\frac{1311035}{1670214}\,A_{10,3}-\frac{50132115}{37858184}\,A_{10,4}\\
&+\frac{2117016}{1391845}\,A_{10,5}-\frac{30492}{25415}\,A_{10,6}+\frac{307098}{482885}\,A_{10,7}-\frac{81675}{386308}\,A_{10,8}+\frac{24200}{676039}\,A_{10,9}\\
&+\frac{58301}{240253860}\,A_{11,0}-\frac{51684}{20021155}\,A_{11,1}+\frac{588181}{47322730}\,A_{11,2}-\frac{506990}{14196819}\,A_{11,3}+\frac{98901}{1456084}\,A_{11,4}\\
&-\frac{2119392}{23661365}\,A_{11,5}+\frac{40194}{482885}\,A_{11,6}-\frac{26136}{482885}\,A_{11,7}+\frac{2475}{104006}\,A_{11,8}-\frac{4400}{676039}\,A_{11,9}\\
&+\frac{594}{676039}\,A_{11,10}-\frac{6617}{2883046320}\,A_{12,0}+\frac{83711}{3123300180}\,A_{12,1}-\frac{81191}{567872760}\,A_{12,2}\\
&+\frac{78419}{170361828}\,A_{12,3}-\frac{75339}{75716368}\,A_{12,4}+\frac{35937}{23661365}\,A_{12,5}-\frac{1617}{965770}\,A_{12,6}+\frac{4521}{3380195}\,A_{12,7}\\
&-\frac{4125}{5408312}\,A_{12,8}+\frac{605}{2028117}\,A_{12,9}-\frac{99}{1352078}\,A_{12,10}+\frac{6}{676039}\,A_{12,11},
\end{aligned}
$$

Appendix B

Hamiltonian solver program code

B.1 main.h

```
#include <stdlib.h>

#include <stdio.h>

#include <string.h>


#ifndef _MAIN_H_

#define _MAIN_H_


#include "difftable.h"


enum {

  S2 = 0, S4, S6, S8, S10, S12,

  S14, S16, S18, S20, S22, S24,

  NUMSHADOWS, NUMDIFS

};


typedef struct Data {


  // SIMULATION PARAMETERS

  double dt;         // timestep
```

```
  int natoms;        // number of atoms in the system
  int n, nsteps;     // n is the current iteration, nsteps the total number
  double pe, energy; // pe is potential energy, energy is total energy

  // PER ATOMS PARAMETERS
  double *pos;       // atom positions
  double *midstep;   // midstep positions
  double *vel;       // atom velocities
  double *f;         // forces on each atom
  double *mass;      // atom masses

  // SHADOW HAMILTONIAN PARAMETERS
  DifTab *diftab;    // structure for backward differences
  double *sham;      // sham stores interpolated shadow hamiltonians
  double *trunc;     // trunc stores truncated shadow hamiltonians

  // SHADOW HAMILTONIAN FORUMULA COEFFICIENTS
  double **coef;     // coefficients for shadow hamiltonian formulas
                     // (formulas using backward differences)

  // OUTPUT FILE PARAMETERS - 3 files are created by this program
  FILE *sim_out; // energy, position, & velocity: <filename>.sim
  FILE *in_out;  // interpolated shadow Hamiltonians: <filename>.interp
  FILE *tr_out;  // truncated shadow Hamiltonians: <filename>.trunc

} Data;

int initialize(int argc, char **argv, Data *data);
int initial_conditions(Data *data);
```

```
#endif // _MAIN_H_
```

B.2 main.c

```
#include "main.h"
#include "force.h"
#include "integrator.h"
#include "output.h"
#include "difftable.h"



/* main() - start of program
 *  -allocates memory & initializes data
 *  -calls integration routine (leapfrog)
 *  -frees memory
 */
int main(int argc, char **argv)
{
  int i;
  Data *data = (Data *)malloc( sizeof(Data) );

  if( initialize(argc, argv, data) ) return 1;
  show_some(data);

  leapfrog(data);

  cleanup(data);
  free(data);
```

```
  return 0;
}



/* initialize() - perform necessary variable initialization
 *  -defines arrays of interpolated shadow Hamiltonian formula coefficients
 *  -allocates memory for backward difference table entries
 *  -opens file streams to output files
 */
int initialize(int argc, char **argv, Data *data)
{
  int i, j;
  DifTab *diftab;
  char buf[64];

  // lengths of the 1D arrays of shadow Hamiltonian coefficients
  int sizes[12] = {1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78};

  // The nonzero coefficients for interpolated shadow Hamiltonian formulas
  //  using backward differences, one 1D array per order 2, 4, ..., 24
  // The A_i,j term corresponding to each array element is given in the
  //  comment immediately preceding the array initialization for the first
  //  4 coefficient arrays. The same pattern is continuted for higher order.
  //  (note that A_i,i = 0 and A_i,j = - A_j,i)

  // H2: { A_1,0 }
  double H2[1] = {1};
  // H4: { A_1,0 , A_2,0 , A_2,1 }
  double H4[3] = {1, -1.0/2.0, 2.0/3.0};
```

```
// H6: { A_1,0 , A_2,0 , A_2,1 , A_3,0 , A_3,1 , A_3,2 }
double H6[6] = {1, -1.0, 27.0/20.0, 11.0/60.0, -9.0/20.0, 3.0/10.0};
// H8: { A_1,0 , A_2,0 , A_2,1 , A_3,0 , A_3,1 , A_3,2 , A_4,0 , ... }
double H8[10] = {1, -3.0/2.0, 16.0/7.0, 13.0/21.0, -32.0/21.0, 36.0/35.0,
                 -5.0/84.0, 22.0/105.0, -9.0/35.0, 4.0/35.0};
// Subsiquent arrays follow the same pattern.
double H10[15] = {1, -2, 125.0/36.0, 47.0/36.0, -125.0/36.0, 500.0/189.0,
                  -11.0/36.0, 1625.0/1512.0, -250.0/189.0, 25.0/42.0,
                  137.0/7560.0, -125.0/1512.0, 55.0/378.0, -5.0/42.0,
                  5.0/126.0};
double H12[21] = {1, -5.0/2.0, 54.0/11.0, 74.0/33.0, -72.0/11.0, 125.0/22.0,
                  -19.0/22.0, 141.0/44.0, -375.0/88.0, 500.0/231.0,
                  29.0/220.0, -3.0/5.0, 325.0/308.0, -200.0/231.0,
                  45.0/154.0, -7.0/1320.0, 137.0/4620.0, -125.0/1848.0,
                  5.0/63.0, -15.0/308.0, 1.0/77.0};
double H14[28] = {1, -3, 343.0/52.0, 535.0/156.0, -1715.0/156.0,
                  3087.0/286.0, -145.0/78.0, 12691.0/1716.0, -3087.0/286.0,
                  42875.0/6864.0, 1377.0/2860.0, -6517.0/2860.0,
                  48363.0/11440.0, -8575.0/2288.0, 1225.0/858.0,
                  -223.0/4290.0, 9947.0/34320.0, -343.0/520.0,
                  1225.0/1584.0, -1225.0/2574.0, 147.0/1144.0, 11.0/7280.0,
                  -343.0/34320.0, 959.0/34320.0, -875.0/20592.0, 35.0/936.0,
                  -21.0/1144.0, 7.0/1716.0};
double H16[36] = {1, -7.0/2.0, 128.0/15.0, 73.0/15.0, -256.0/15.0,
                  10976.0/585.0, -41.0/12.0, 1712.0/117.0, -2744.0/117.0,
                  10976.0/715.0, 743.0/585.0, -3712.0/585.0,
                  406112.0/32175.0, -43904.0/3575.0, 6860.0/1287.0,
                  -31.0/130.0, 4896.0/3575.0, -104272.0/32175.0,
                  128968.0/32175.0, -3430.0/1287.0, 3136.0/3861.0,
```

```
                37.0/1925.0, -28544.0/225225.0, 11368.0/32175.0,
                -1568.0/2925.0, 140.0/297.0, -896.0/3861.0, 112.0/2145.0,
                -761.0/1801800.0, 22.0/6825.0, -343.0/32175.0,
                1918.0/96525.0, -175.0/7722.0, 28.0/1755.0, -14.0/2145.0,
                8.0/6435.0};
double H18[45] = {1, -4, 729.0/68.0, 1337.0/204.0, -1701.0/68.0,
                2592.0/85.0, -385.0/68.0, 17739.0/680.0, -3888.0/85.0,
                37044.0/1105.0, 1879.0/680.0, -9963.0/680.0,
                34668.0/1105.0, -37044.0/1105.0, 1000188.0/60775.0,
                -193.0/255.0, 20061.0/4420.0, -12528.0/1105.0,
                913752.0/60775.0, -666792.0/60775.0, 9261.0/2431.0,
                197.0/1820.0, -22599.0/30940.0, 52488.0/25025.0,
                -201096.0/60775.0, 186543.0/60775.0, -3969.0/2431.0,
                1008.0/2431.0, -419.0/61880.0, 26973.0/523600.0,
                -72252.0/425425.0, 38367.0/121550.0, -3969.0/11050.0,
                189.0/748.0, -252.0/2431.0, 243.0/12155.0,
                7129.0/61261200.0, -6849.0/6806800.0, 297.0/77350.0,
                -1029.0/121550.0, 2877.0/243100.0, -105.0/9724.0,
                7.0/1105.0, -27.0/12155.0, 9.0/24310.0};
double H20[55] = {1, -9.0/2.0, 250.0/19.0, 484.0/57.0, -2000.0/57.0,
                30375.0/646.0, -497.0/57.0, 167125.0/3876.0,
                -212625.0/2584.0, 21600.0/323.0, 34167.0/6460.0,
                -9625.0/323.0, 88695.0/1292.0, -25920.0/323.0,
                185220.0/4199.0, -14917.0/7752.0, 46975.0/3876.0,
                -83025.0/2584.0, 192600.0/4199.0, -154350.0/4199.0,
                666792.0/46189.0, 2761.0/6783.0, -19300.0/6783.0,
                501525.0/58786.0, -417600.0/29393.0, 652680.0/46189.0,
                -381024.0/46189.0, 110250.0/46189.0, -831.0/18088.0,
                4925.0/13832.0, -564975.0/470288.0, 43740.0/19019.0,
```

```
                 -6615.0/2431.0, 186543.0/92378.0, -165375.0/184756.0,
                 9000.0/46189.0, 4861.0/2116296.0, -10475.0/529074.0,
                 14985.0/198968.0, -53520.0/323323.0, 21315.0/92378.0,
                 -882.0/4199.0, 875.0/7106.0, -2000.0/46189.0,
                 675.0/92378.0, -671.0/21162960.0, 7129.0/23279256.0,
                 -6849.0/5173168.0, 99.0/29393.0, -1029.0/184756.0,
                 2877.0/461890.0, -875.0/184756.0, 10.0/4199.0,
                 -135.0/184756.0, 5.0/46189.0};
double H22[66] = {1, -5, 1331.0/84.0, 299.0/28.0, -1331.0/28.0,
                 166375.0/2394.0, -89.0/7.0, 161051.0/2394.0,
                 -166375.0/1197.0, 4492125.0/36176.0, 15797.0/1710.0,
                 -94501.0/1710.0, 6355525.0/46512.0, -898425.0/5168.0,
                 239580.0/2261.0, -1201.0/285.0, 2165537.0/77520.0,
                 -1830125.0/23256.0, 4372335.0/36176.0, -239580.0/2261.0,
                 195657.0/4199.0, 129805.0/108528.0, -2836361.0/325584.0,
                 62523725.0/2279088.0, -12278475.0/253232.0,
                 10681275.0/205751.0, -139755.0/4199.0, 45738.0/4199.0,
                 -32891.0/162792.0, 3674891.0/2279088.0,
                 -6422075.0/1139544.0, 74169975.0/6584032.0,
                 -2894925.0/205751.0, 94017.0/8398.0, -22869.0/4199.0,
                 45375.0/33592.0, 42131.0/2279088.0, -368687.0/2279088.0,
                 6555175.0/10456992.0, -9283725.0/6584032.0,
                 49005.0/24206.0, -847.0/442.0, 39809.0/33592.0,
                 -15125.0/33592.0, 15125.0/176358.0, -4307.0/5697720.0,
                 6469991.0/888844320.0, -2788445.0/88884432.0,
                 40293.0/506464.0, -26983.0/205751.0, 24563.0/167960.0,
                 -9317.0/83980.0, 3025.0/54264.0, -3025.0/176358.0,
                 605.0/235144.0, 83711.0/9777287520.0,
                 -81191.0/888844320.0, 78419.0/177768864.0,
```

```
                 -8371.0/6584032.0, 3993.0/1646008.0, -539.0/167960.0,
                 1507.0/503880.0, -1375.0/705432.0, 605.0/705432.0,
                 -55.0/235144.0, 11.0/352716.0};
double H24[78] = {1, -11.0/2.0, 432.0/23.0, 905.0/69.0, -1440.0/23.0,
                 15972.0/161.0, -1635.0/92.0, 702.0/7.0, -35937.0/161.0,
                 665500.0/3059.0, 12126.0/805.0, -76896.0/805.0,
                 3865224.0/15295.0, -1064800.0/3059.0, 24257475.0/104006.0,
                 -953.0/115.0, 126376.0/2185.0, -378004.0/2185.0,
                 6355525.0/22287.0, -8085825.0/29716.0, 6899904.0/52003.0,
                 45454.0/15295.0, -345888.0/15295.0, 19489833.0/260015.0,
                 -7320500.0/52003.0, 118053045.0/728042.0,
                 -41399424.0/364021.0, 4024944.0/96577.0, -8315.0/12236.0,
                 1168245.0/208012.0, -8509083.0/416024.0,
                 62523725.0/1456084.0, -331518825.0/5824336.0,
                 230715540.0/4732273.0, -2515590.0/96577.0,
                 705672.0/96577.0, 176005.0/1872108.0, -131564.0/156009.0,
                 3674891.0/1092063.0, -25688300.0/3276189.0,
                 222509925.0/18929092.0, -55582560.0/4732273.0,
                 752136.0/96577.0, -313632.0/96577.0, 136125.0/193154.0,
                 -2339.0/328440.0, 126393.0/1820105.0, -1106061.0/3640210.0,
                 1311035.0/1670214.0, -50132115.0/37858184.0,
                 2117016.0/1391845.0, -30492.0/25415.0, 307098.0/482885.0,
                 -81675.0/386308.0, 24200.0/676039.0, 58301.0/240253860.0,
                 -51684.0/20021155.0, 588181.0/47322730.0,
                 -506990.0/14196819.0, 98901.0/1456084.0,
                 -2119392.0/23661365.0, 40194.0/482885.0,
                 -26136.0/482885.0, 2475.0/104006.0, -4400.0/676039.0,
                 594.0/676039.0, -6617.0/2883046320.0,
                 83711.0/3123300180.0, -81191.0/567872760.0,
```

```
                  78419.0/170361828.0, -75339.0/75716368.0,
                  35937.0/23661365.0, -1617.0/965770.0, 4521.0/3380195.0,
                  -4125.0/5408312.0, 605.0/2028117.0, -99.0/1352078.0,
                  6.0/676039.0};

// pack all of the 1D arrays into a single 2D array
double *coef[12] = { H2, H4, H6, H8, H10, H12,
                     H14, H16, H18, H20, H22, H24 };

if(argc!=4)
  {
    fprintf(stderr,"Usage: %s timestep numsteps filename\n", argv[0]);
    fprintf(stderr,
            "** Warning: filename.sim, filename.interp, & filename.trunc\n");
    fprintf(stderr,
            "** will be created (or overwritten, if they exist)\n");
    return 1;
  }

data->dt = atof(argv[1]);
data->nsteps = atoi(argv[2]);

data->n = 0;
initial_conditions(data);

// SHADOW HAMILTONIAN PARAMETERS
data->diftab = (DifTab *)malloc( sizeof(DifTab) );
diftab = data->diftab;
// use calloc to initialize these to 0's
```

```
data->sham = (double *)calloc( NUMDIFS, sizeof(double) );
data->trunc = (double *)calloc( NUMDIFS, sizeof(double) );

// BACKWARD DIFFERENCE TABLES
// (FOR INTERPOLATED SHADOW HAMILTONIAN CALCULATION)
diftab->dif_q = (double **)calloc( NUMDIFS, sizeof(double *) );
diftab->dif_p = (double **)calloc( NUMDIFS, sizeof(double *) );
diftab->dif_beta = (double *)calloc( NUMDIFS, sizeof(double) );
for(i=0; i<NUMDIFS; i++)
  { diftab->dif_q[i] = (double *)calloc( data->natoms, sizeof(double) );
    diftab->dif_p[i] = (double *)calloc( data->natoms, sizeof(double) ); }

diftab->A = (double **)calloc( NUMDIFS, sizeof(double *) );
for(i=0; i<NUMDIFS; i++)
  diftab->A[i] = (double *)calloc( NUMDIFS, sizeof(double) );

diftab->mdf_q = (double **)calloc( NUMDIFS, sizeof(double *) );
diftab->mdf_p = (double **)calloc( NUMDIFS, sizeof(double *) );
diftab->mdf_beta = (double *)calloc( NUMDIFS, sizeof(double) );
for(i=0; i<NUMDIFS; i++)
  { diftab->mdf_q[i] = (double *)calloc( data->natoms, sizeof(double) );
    diftab->mdf_p[i] = (double *)calloc( data->natoms, sizeof(double) ); }

diftab->B = (double **)calloc( NUMDIFS, sizeof(double *) );
for(i=0; i<NUMDIFS; i++)
  diftab->B[i] = (double *)calloc( NUMDIFS, sizeof(double) );

// shadow Hamiltonian formula coefficients
data->coef = (double **)calloc( NUMSHADOWS, sizeof(double *) );
```

```
  for(i=0; i<NUMSHADOWS; i++)
    data->coef[i] = (double *)calloc( sizes[i], sizeof(double) );
  for(i=0; i<NUMSHADOWS; i++)
    for(j=0; j<sizes[i]; j++)
      data->coef[i][j] = coef[i][j];

  // output file streams
  strcpy(buf, argv[3]);
  strcat(buf, ".sim");
  data->sim_out = fopen(buf,"w");

  strcpy(buf, argv[3]);
  strcat(buf, ".interp");
  data->in_out = fopen(buf,"w");

  strcpy(buf, argv[3]);
  strcat(buf, ".trunc");
  data->tr_out = fopen(buf,"w");

  for(i=0; i<data->natoms; i++)
    force_eval(data,i); // calculate initial forces
  energy_eval(data);    // calculate initial energy

  return 0;
}

/*
 * cleanup() - deallocate memory
 */
```

```
int cleanup(Data *data)
{
  int i;
  free(data->f); free(data->mass);

  for(i=0; i<NUMDIFS; i++)
    { // catches pos, midstep, & vel (in 0th index)
      free(data->diftab->dif_q[i]); free(data->diftab->mdf_q[i]);
      free(data->diftab->dif_p[i]); free(data->diftab->mdf_p[i]);
      free(data->diftab->A[i]);     free(data->diftab->B[i]);
    }
  free(data->diftab->dif_q);    free(data->diftab->mdf_q);
  free(data->diftab->dif_p);    free(data->diftab->mdf_p);
  free(data->diftab->dif_beta); free(data->diftab->mdf_beta);
  free(data->diftab->A);        free(data->diftab->B);

  free(data->diftab);
  free(data->sham); free(data->trunc);

  fclose(data->sim_out); fclose(data->in_out); fclose(data->tr_out);
  return 0;
}
```

B.3 integrator.h

```
#include "main.h"

#ifndef _INTEGRATOR_H_
#define _INTEGRATOR_H_
```

```
int leapfrog(Data *data);    // leapfrog integration
int energy_eval(Data *data); // calculate the energy
int shadow_hamiltonians_eval(Data *data); // calculate the shadow energies

#endif // _INTEGRATOR_H_
```

B.4 integrator.c

```
#include "integrator.h"


int leapfrog(Data *data)
{
  int i, j, k, nsteps = data->nsteps, natoms = data->natoms;
  double *pos = data->pos, *vel = data->vel, *f = data->f, *m = data->mass;
  double *midstep = data->midstep;
  double beta, midbeta=0;
  double dot, dt = data->dt, half_dt = dt/2.0;

  for(k=0; k<nsteps; k++)
    {
      // HALF KICK : vel n+1/2
      for(j=0; j<natoms; j++) vel[j] += (half_dt/m[j])*f[j];

      // BACKWARD DIFFERENCE TABLE UPDATES (Q PORTIONS ONLY)
      update_q_middiffs(data); // uses F(n) & 1st diff. of q(n)
      update_q_diffs(data);    // uses F(n) & p(n+1/2)

      // Beta n+1/2
      for(dot=i=0; i<natoms; i++) dot += pos[i]*f[i];
      midbeta += beta = half_dt*(-dot - 2.0*data->pe);
```

```
// BACKWARD DIFFERENCE TABLE UPDATE (MIDSTEP BETA PORTION ONLY)
update_beta_middiffs(data, midbeta);

// HALF-DRIFT : pos n+1/2
for(j=0; j<natoms; j++) midstep[j] = pos[j] + half_dt*vel[j];

// calculate Bij's
calc_B_table(data); // 0th differences are midstep values

// DRIFT : pos n+1
for(j=0; j<natoms; j++) pos[j] += dt*vel[j];

// FORCE : f n+1 & BACKWARD DIFFERENCE TABLE UPDATE (P PORTIONS ONLY)
for(j=0; j<natoms; j++) force_eval(data,j);

// HALF KICK : vel n+1
for(j=0; j<natoms; j++) vel[j] += (half_dt/m[j])*f[j];

energy_eval(data); // potential energy n+1 (for use in Beta n+1)

// Beta n+1
for(dot=i=0; i<natoms; i++) dot += pos[i]*f[i];
beta += midbeta = half_dt*(-dot - 2.0*data->pe);

// BACKWARD DIFFERENCE TABLE UPDATE (FULLSTEP BETA PORTION ONLY)
update_beta_diffs(data, beta);

// backward difference table update completed (all portions)
```

```
      // calculate Aij's
      calc_A_table(data);

      // use Aij's & Bij's to calculate shadow hamiltonians
      shadow_hamiltonians_eval(data);
      truncated_eval(data);

      data->n++;
      if(k<NUMDIFS) show_some(data);
      else show_all(data);
    }

  return 0;
}


int energy_eval(Data *data)
{
  int i, natoms = data->natoms;
  double sum = 0;
  double *vel = data->vel, *m = data->mass;

  for(i=0; i<natoms; i++) sum += 0.5*m[i]*vel[i]*vel[i]; // kinetic energy

  data->energy = data->pe = 0;
  potential_eval(data); // potential energy
  data->energy = sum + data->pe; // total = kinetic + potential
```

```
  return 0;
}



/* shadow_hamiltonians_eval() - calculate shadow energies
 *  -evaluates the interpolated shadow Hamiltonian formulas
 *   using the Aij & Bij terms (already computed) & the coefficients
 *   (initialized before start of leapfrog).
 */
int shadow_hamiltonians_eval(Data *data)
{
  int i, k, m, coefnum;
  double **A=data->diftab->A;

  // highest order differences are smallest
  // thus start summing with these, to minimize floating point error
  for(i=0; i<NUMSHADOWS; i++)
    {
      if(i%2==1) A=data->diftab->A;
      else A=data->diftab->B;
      data->sham[i] = 0;
      coefnum = -1;
      for(k=1; k<=i+1; k++)
        for(m=0; m<k; m++)
          coefnum++;
      for(k=i+1; k>0; k--)
        for(m=k-1; m>=0; m--)
          data->sham[i] += data->coef[i][coefnum--] * A[k][m];
    }
```

```
  return 0;
}
```

B.5 difftable.h

```
#include "main.h"


#ifndef _DIFFTABLE_H_
#define _DIFFTABLE_H_


typedef struct DifTab {
  double **dif_q;   // backward differences of positions (one per atom)
  double **dif_p;   // backward differences of velocities (one per atom)
  double *dif_beta; // backward differences of beta
  double **A;       // Aij terms for interpolated shadow Hamiltonian formulas

  double **mdf_q;   // backward differences of midpoint positions
  double **mdf_p;   // backward differences of midpoint velocities
  double *mdf_beta; // backward differences of midpoint beta
  double **B;       // Bij terms for interpolated shadow Hamiltonians
} DifTab;

#endif // _DIFFTABLE_H_
```

B.6 difftable.c

```
#include "difftable.h"


/* update_q_diffs() - calculate the next wavefront of backward differences of
```

```
 *                    fullstep values of q
 *  -2nd backward difference of q is calculated explicitly
 *  -all other backward differences are calculted with respect to this new
 *   value and all of the old values
 */
int update_q_diffs(Data *data)
{
  int i, j, k, natoms = data->natoms;
  double sum, u, dt = data->dt;
  double *m = data->mass, *f = data->f, *vel = data->vel;
  double **dif_q = data->diftab->dif_q;

  for(j=0; j<natoms; j++)
    {
      u = (dt*dt/m[j])*f[j]; // new 2nd backward difference of q (leapfrog)
      for(i=NUMSHADOWS; i>1; i--)
        { for(sum = 0, k=2; k<i; k++) sum += dif_q[k][j];
          dif_q[i][j] = u - sum; }
      dif_q[1][j] = dt*vel[j]; // new 1st backward difference of q (leapfrog)
    }
  return 0;
}


/* update_q_middiffs() - calculate the next wavefront of backward
 *                       differences of midstep q values
 *  -1st backward difference of midstep q is calculated explicitly
 *  -all other backward differences are calculted with respect to this new
 *   value and all of the old values
```

```
 */
int update_q_middiffs(Data *data)
{
  int i, j, k, natoms = data->natoms;
  double sum, u, dt = data->dt;
  double *m = data->mass, *f = data->f, *vel = data->vel;
  double **dif_q = data->diftab->dif_q, **mdf_q = data->diftab->mdf_q;

  for(j=0; j<natoms; j++)
    {
      // new 1st backward difference of midstep values of q (leapfrog)
      u = dif_q[1][j] + (dt*dt/(2.0*m[j]))*f[j];
      for(i=NUMSHADOWS; i>0; i--)
        { for(sum = 0, k=1; k<i; k++) sum += mdf_q[k][j];
          mdf_q[i][j] = u - sum; }
    }

  return 0;
}



/* update_beta_diffs() - calculate the next wavefront of backward differences
                         of fullstep values of beta
 *  -1st backward difference of beta is calculated explicitly
 *  -all other backward differences are calculted with respect to this new
 *   value and all of the old values
 */
int update_beta_diffs(Data *data, double beta)
{
```

```
  int i, k;
  double sum;
  double *dif_beta = data->diftab->dif_beta;

  for(i=NUMSHADOWS; i>0; i--)
    { for(sum = 0, k=1; k<i; k++) sum += dif_beta[k];
      dif_beta[i] = beta - sum; }

  return 0;
}



/* update_beta_middiffs() - calculate the next wavefront of backward
 *                          differences of midstep beta values
 *  -1st backward difference of midstep beta is calculated explicitly
 *  -all other backward differences are calculted with respect to this new
 *   value and all of the old values
 */
int update_beta_middiffs(Data *data, double midbeta)
{
  int i, k;
  double sum;
  double *mdf_beta = data->diftab->mdf_beta;

  for(i=NUMSHADOWS; i>0; i--)
    { for(sum = 0, k=1; k<i; k++) sum += mdf_beta[k];
      mdf_beta[i] = midbeta - sum; }

  return 0;
```

```
}



/* calc_A_table() - calculate the values of the Aij used in the
 *                  formulas for interpolated shadow Hamiltonians
 *                  of order 2k with k even
 */
int calc_A_table(Data *data)
{
  int i, j, k, natoms = data->natoms;
  double dot, *m = data->mass, **A = data->diftab->A;
  double *dif_beta = data->diftab->dif_beta;
  double **dif_q = data->diftab->dif_q, **dif_p = data->diftab->dif_p;

  dif_q[0] = data->pos;
  dif_p[0] = data->vel;
  dif_beta[0] = 0; // shouldn't ever be used

  for(i=1; i<NUMDIFS; i++)
    {
      for(j=0; j<i; j++)
        {
          // difference table stores velocity differences; hence
          // mass must be multiplied in to get momentum.
          for(dot=k=0; k<natoms; k++) dot += dif_q[i][k]*dif_p[j][k]*m[k];
          A[i][j] = dot;
          for(dot=k=0; k<natoms; k++) dot += dif_p[i][k]*dif_q[j][k]*m[k];
          A[i][j] -= dot;
        }
```

```
      A[i][0] -= dif_beta[i];
    }

  for(i=1; i<NUMDIFS; i++)
    for(j=0; j<i; j++)
      A[i][j] /= 2.0*data->dt;

  return 0;
}


/* calc_B_table() - calculate the values of the Bij used in the
 *                  formulas for the interpolated shadow Hamiltonians
 *                  of order 2k with k odd
 */
int calc_B_table(Data *data)
{
  int i, j, k, natoms = data->natoms;
  double dot, *m = data->mass, **B = data->diftab->B;
  double **mdf_q = data->diftab->mdf_q;
  double **mdf_p = data->diftab->mdf_p;
  double *mdf_beta = data->diftab->mdf_beta;

  mdf_q[0] = data->midstep;
  mdf_p[0] = data->vel;
  mdf_beta[0] = 0; // shouldn't ever be used

  for(i=1; i<NUMDIFS; i++)
    {
```

```
      for(j=0; j<i; j++)
        {
          // difference table stores velocity differences; hence
          // mass must be multiplied in to get momentum.
          for(dot=k=0; k<natoms; k++) dot += mdf_q[i][k]*mdf_p[j][k]*m[k];
          B[i][j] = dot;
          for(dot=k=0; k<natoms; k++) dot += mdf_p[i][k]*mdf_q[j][k]*m[k];
          B[i][j] -= dot;
        }
      B[i][0] -= mdf_beta[i];
    }

  for(i=1; i<NUMDIFS; i++)
    for(j=0; j<i; j++)
      B[i][j] /= 2.0*data->dt;

  return 0;
}
```

B.7 force.h

```
#include "main.h"

#ifndef _FORCE_H_
#define _FORCE_H_

/*
 * force_eval() & potential_eval()
 * the definitions of these 2 functions can be changed
 * to use different potentials
```

```
 */
int force_eval(Data *data, int i);
int potential_eval(Data *data);

#endif // _FORCE_H_
```

B.8 output.h

```
#include "main.h"

#ifndef _OUTPUT_H_
#define _OUTPUT_H_

void show_header(Data *data); // output column headings only
void show_some(Data *data);   // output all excluding shadow Hamiltonians
void show_all(Data *data);    // output all including shadow Hamiltonians

#endif // _OUTPUT_H_
```

B.9 output.c

```
#include "output.h"

void show_header(Data *data)
{
  int i, natoms = data->natoms;

  fprintf(data->sim_out,"# %7s %7s %16s ", "n", "time", "energy");
  for(i=0; i<natoms; i++) fprintf(data->sim_out, "%16s ", "pos");
  for(i=0; i<natoms; i++) fprintf(data->sim_out, "%16s ", "vel");
```

```
  fprintf(data->sim_out,"\n");

  fprintf(data->in_out, "# %7s %16s %16s %16s %16s %16s ",
          "time",
          "interp2", "interp4", "interp6", "interp8", "interp10");
  fprintf(data->in_out, "%16s %16s %16s %16s %16s %16s %16s\n",
          "interp12", "interp14", "interp16", "interp18",
          "interp20", "interp22", "interp24");
  fprintf(data->tr_out,"# %7s %16s %16s %16s %16s %16s\n",
          "time",
          "trunc2", "trunc4", "trunc6", "trunc8", "trunc10");
}

void show_some(Data *data)
{
  int i, natoms = data->natoms, n = data->n;
  double energy = data->energy;
  double *pos = data->pos, *vel = data->vel, *sham = data->sham;

  fprintf(data->sim_out, "%8d %8g %20.16g ", n, n*data->dt, energy);
  for(i=0; i<natoms; i++) fprintf(data->sim_out, "%20.16g ", pos[i]);
  for(i=0; i<natoms; i++) fprintf(data->sim_out, "%20.16g ", vel[i]);
  fprintf(data->sim_out,"\n");

  return;
}


void show_all(Data *data)
```

```
{
  int i, natoms = data->natoms, n = data->n;
  double energy = data->energy;
  double *pos = data->pos, *vel = data->vel;
  double *sham = data->sham, *trunc = data->trunc;

  fprintf(data->sim_out, "%8d %8g %20.16g ", n, n*data->dt, energy);
  for(i=0; i<natoms; i++) fprintf(data->sim_out, "%20.16g ", pos[i]);
  for(i=0; i<natoms; i++) fprintf(data->sim_out, "%20.16g ", vel[i]);
  fprintf(data->sim_out,"\n");

  fprintf(data->in_out, "%20.16g %20.16g %20.16g %20.16g %20.16g %20.16g ",
          n*data->dt, sham[S2], sham[S4], sham[S6],  sham[S8], sham[S10]);
  fprintf(data->in_out,
          "%20.16g %20.16g %20.16g %20.16g %20.16g %20.16g %20.16g\n",
          sham[S12], sham[S14], sham[S16], sham[S18],
          sham[S20], sham[S22], sham[S24]);
  fprintf(data->tr_out,
          "%20.16g %20.16g %20.16g %20.16g %20.16g %20.16g\n",
          n*data->dt, trunc[S2], trunc[S4], trunc[S6], trunc[S8], trunc[S10]);

  return;
}
```

B.10 truncated.h

```
#include "main.h"

#ifndef _FORCE_H_
#define _FORCE_H_
```

```
/*
 * force_eval() & potential_eval()
 * the definitions of these 2 functions can be changed
 * to use different potentials
 */
int force_eval(Data *data, int i);
int potential_eval(Data *data);

#endif // _FORCE_H_
```

B.11 henon.c

See http://bionum.cs.uiuc.edu/modHamiltonians.html

B.12 doublewell.c

See http://bionum.cs.uiuc.edu/modHamiltonians.html

Appendix C

Supplementary figures

In the following eight figures, the conservation of the interpolated modified Hamiltonians $H_{[2k]}$ versus order $2k$ is plotted as a solid line with open circles at the data points. Additionally, some figures also plot the conservation of the truncated modified Hamiltonians versus order $2k$ as a dotted line with asterisks at the data points. Figures C.5 through C.8 include a horizontal dashed line indicating the energy conservation, for comparison to the conservation of the interpolated modified Hamiltonians. All are plotted on log-log axes.

The final figure, number C.9, is slightly different. This log-linear plot shows the exponential decrease as a function of interpolation degree in the discrepancy of numerical trajectories associated with the interpolated modified Hamiltonians. Proceeding from the topmost line downward, each line plots the trajectory discrepancy associated with each next higher order of interpolated modified Hamiltonian versus time, starting with 2^{nd} order and culminating with 12^{th} order.

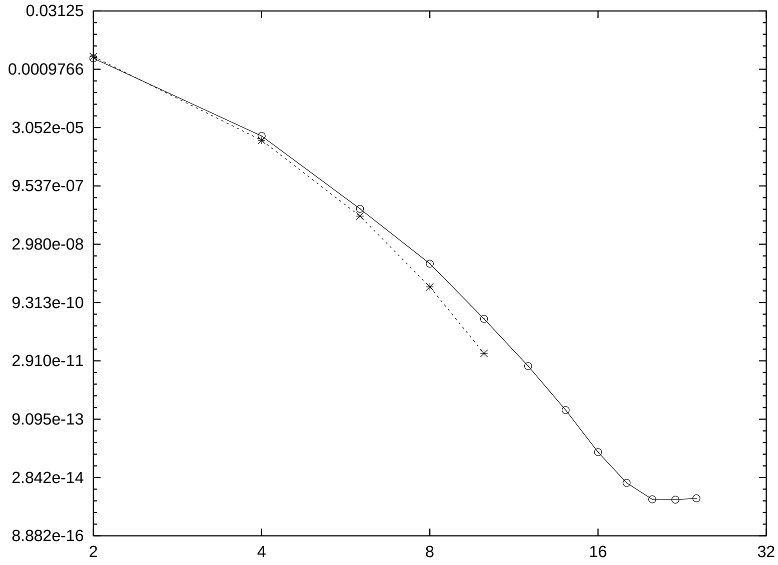

Figure C.1: Conservation of IMH and TMH vs. order $2k$ for double well, $h = 0.1$

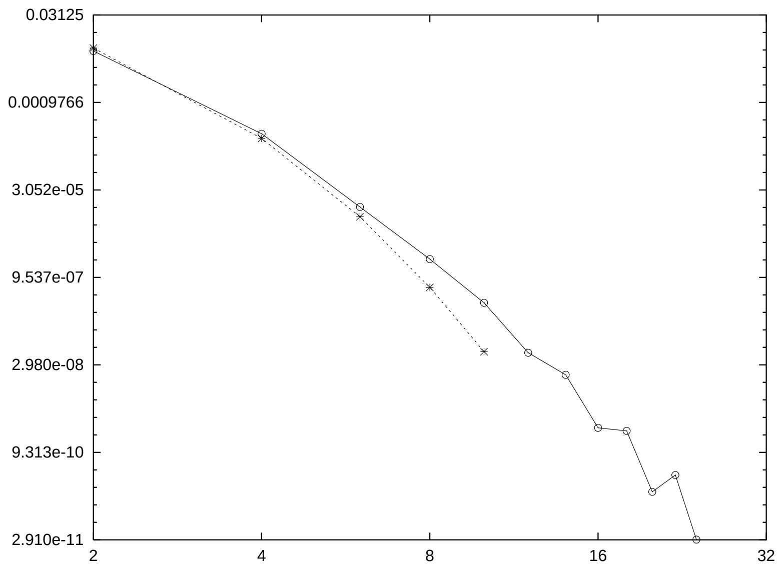

Figure C.2: Conservation of IMH and TMH vs. order $2k$ for double well, $h = 0.2$

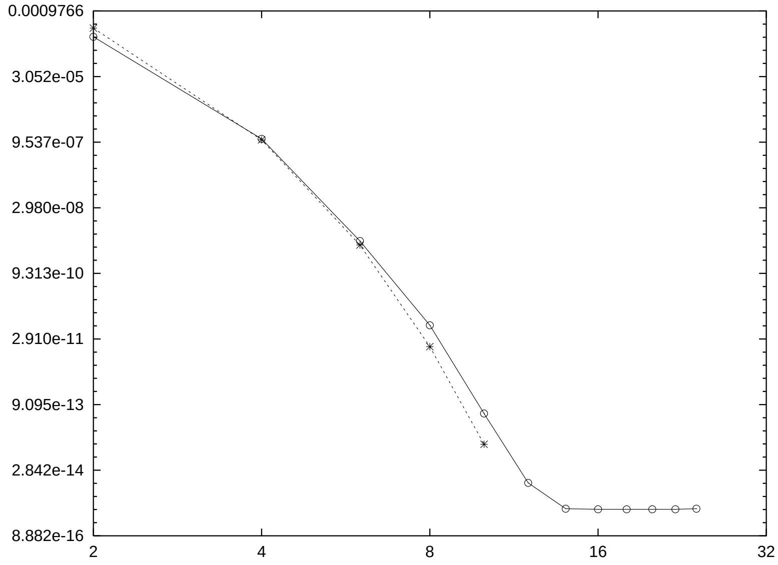

Figure C.3: Conservation of IMH and TMH vs. order $2k$ for Hénon-Heiles, $h = 0.1$

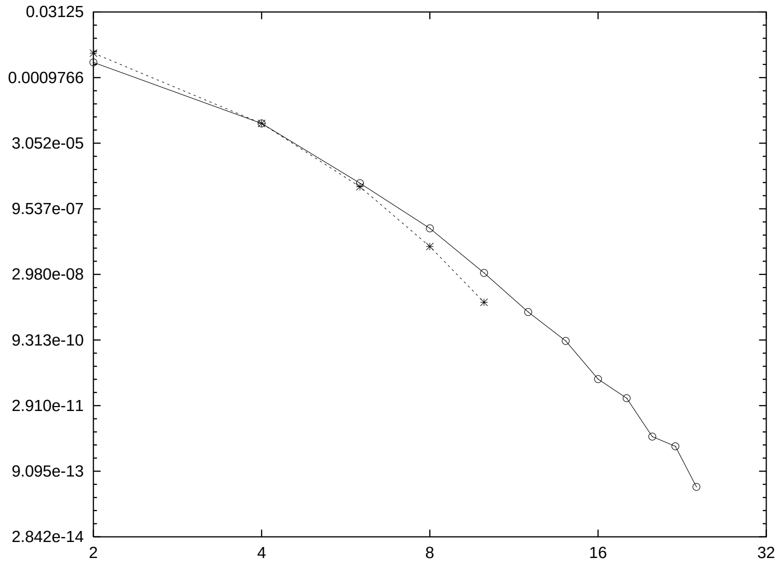

Figure C.4: Conservation of IMH and TMH vs. order $2k$ for Hénon-Heiles, $h = 0.3$

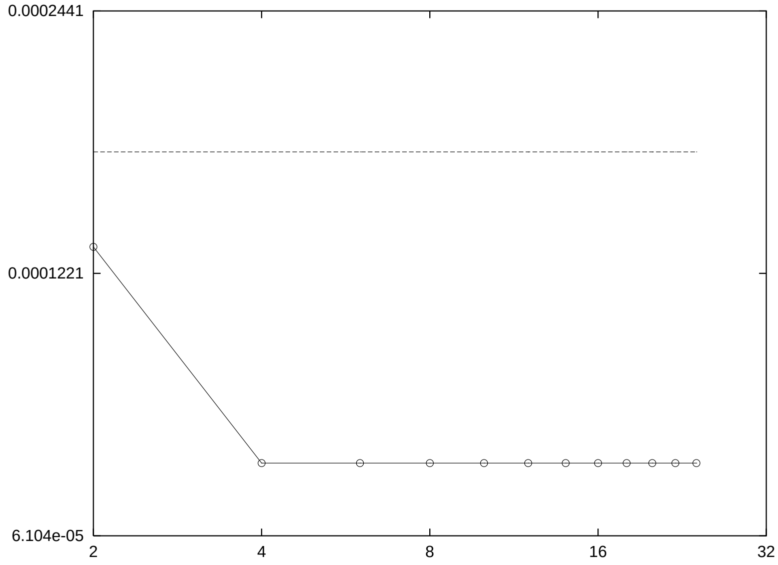

Figure C.5: Conservation of IMH vs. order $2k$ for piecewise, $h = 0.01$

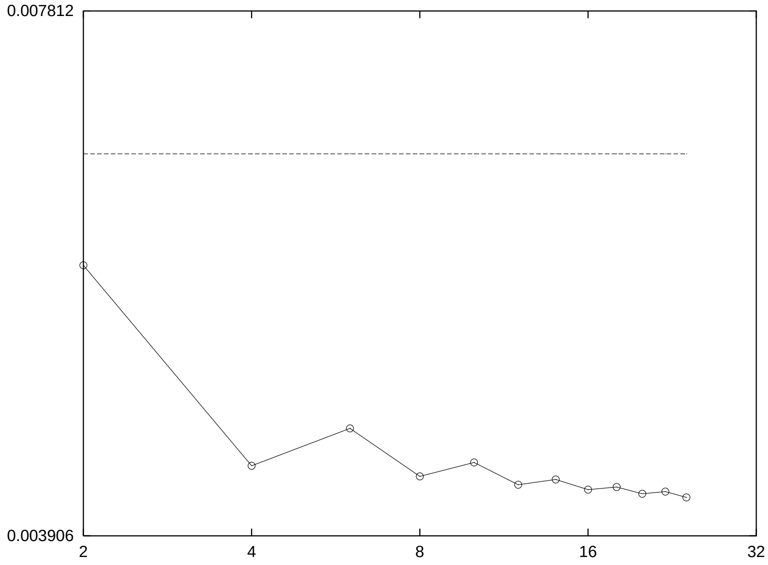

Figure C.6: Conservation of IMH vs. order $2k$ for piecewise, $h = 0.05$

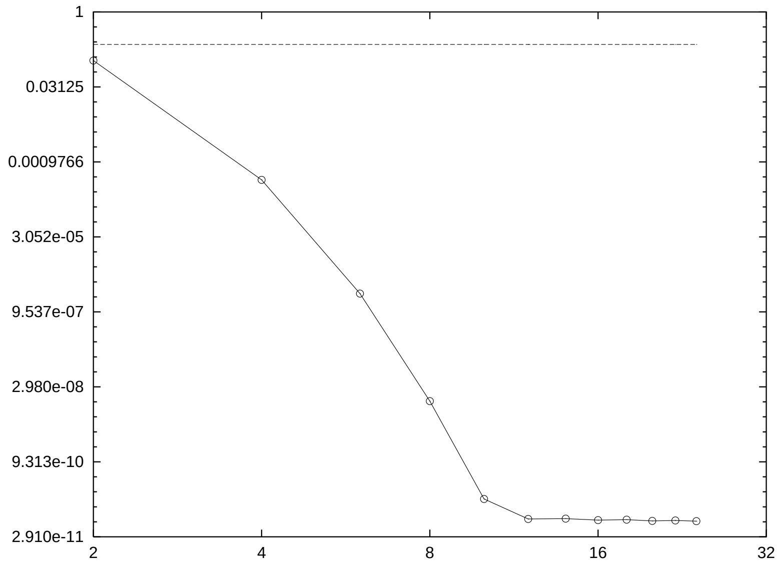

Figure C.7: Conservation of IMH vs. order $2k$ for molecular dynamics, $h = 0.25$ fs

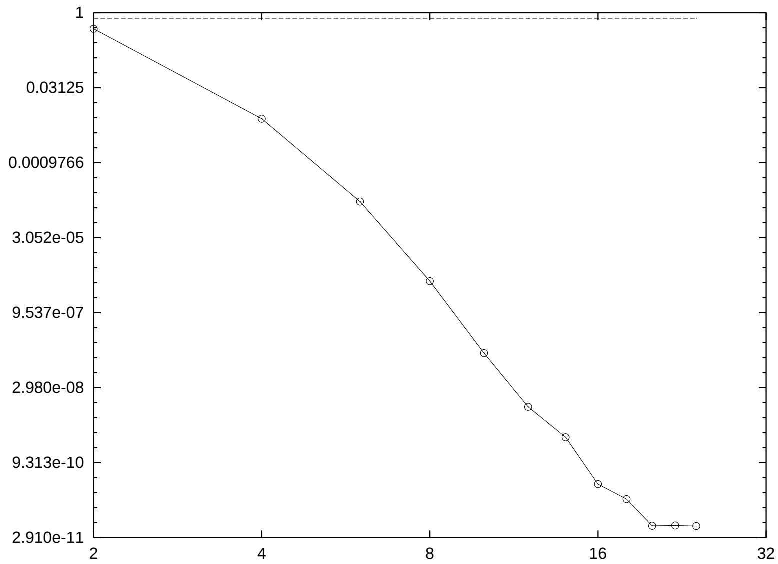

Figure C.8: Conservation of IMH vs. order $2k$ for molecular dynamics, $h = 0.5$ fs

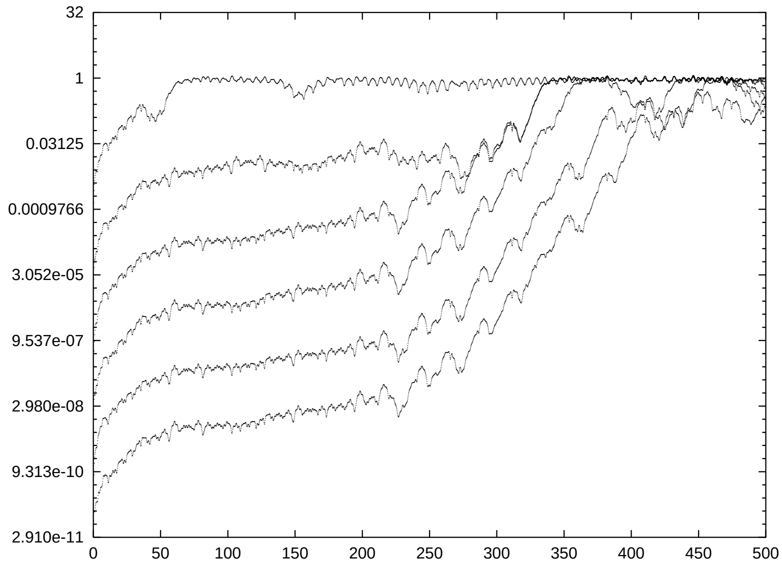

Figure C.9: Trajectory discrepancy vs. time for IMH