HAMILTONIAN MECHANICS AND THE CONSTRUCTION OF NUMERICAL INTEGRATORS

KATHRYN FARRELL

Department of Mathematics, University of California, San Diego
9500 Gilman Drive, Dept. 0112, La Jolla, CA 92093-0112 USA
Email: `mholst@math.ucsd.edu`

Abstract. Introductory courses on differential equations cover integration techniques for integrable differential equations. However, most systems of ordinary differential equations are too complicated to be integrated exactly. Therefore, mathematicians have developed ways through which we can approximate such systems. These *numerical integrators* solve systems of differential equations to within a certain error. The complexity and cost of such integrators grows with their precision. Numerical analysts are always looking for new integration schemes that have low error and low cost. In this paper, we discuss the derivation of numerical integrators as well as their benefits and disadvantages.

Contents

Date: June 1, 2009.

1. Introduction

This paper is concerned with systems of ordinary differential equations (Ordinary Differential Equations). More specifically, it is concerned with systems of ODEs that conserve energy and angular momentum, which are called *conservative systems*. Even more specifically, this paper is concerned with conservative Hamiltonian systems.

Most systems are too complicated to integrate directly and exactly. These are the cases with which this paper is concerned. In these cases, the equations are *discretized*, that is, approximations are made at several points in time over the interval of integration rather than using the continuous, or exact, solution. These methods have numerous applications, from Kepler's laws of motion which describe gravitational dynamics to molecular models which are described by the laws of classical mechanics. The details and formulation of the theory behind mechanical systems will be discussed in the first section of the paper.

After a basic understanding of unconstrained and constrained dynamics is established, we will begin to develop the theory behind numerical integration. As the paper progresses, so does the complexity and accuracy of these integrators. We begin with Taylor's Theorem and how it gives rise to one-step methods of arbitrary order p. We will specifically look at Euler's method to demonstrate basic analysis techniques that can be applied to most numerical methods. Then we develop Runge-Kutta methods, which use Taylor methods to produce high-order methods without the complication of computation and evalutation of derivatives that we see in Taylor methods. Then we begin to discuss integrators that were created specifically for systems of ODEs, rather than just dealing with one differential equation. Partitioned Runge-Kutta methods use different sets of quadrature rules for each set of variables and give rise to direct discretization schemes. Specifically, we will look at SHAKE and RATTLE, two numerical integrators that preserve certain properties of conservative Hamiltonian systems. Finally, we will look at a method that produces high-order integrators by composition of lower-order integrators while preserving their symplecticity.

2. Dynamics

One of the most common applications of numerical analysis is to physical problems. The most common example is the dynamics of a pendulum, but numerical integration is also useful in multi-particle systems, as in molecular dynamics. In this section, we first introduce various theory and notation of Hamiltonian mechanics. Then we discuss constrained mechanics and their Hamiltonian formulation.

2.1. **Hamiltonian Dynamics.**

Hamitonian dynamics is a special formulation of Newtonian mechanics. Certain properties of Hamiltonian systems will be discussed in this section and throughout the paper. First, let us clarify notation. For an

N-body system, the matrix $\boldsymbol{M} \in \mathbb{R}^{3N\times 3N}$ is the diagonal mass matrix of the form

$$\begin{pmatrix} m_1 & 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & m_1 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & m_1 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & m_2 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & m_2 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & m_2 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

such that

$$\mathbf{M}\mathbf{v} = (m_1\mathbf{v}_1, m_2\mathbf{v}_2, \dots, m_N\mathbf{v}_N)^T$$

. We also denote the particle coordinates in vector form:

$$\mathbf{q} := (\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_N), \mathbf{q}_k := (q_k^{(1)}, q_k^{(2)}, q_k^{(3)}).$$

This same notation is used for velocity vectors. Note also that we can express the force $\mathbf{F}$ as the negative gradient of the potential energy function with respect to the particle positions:

$$\mathbf{F}(\mathbf{q}) := -\nabla_{\mathbf{q}} V(\mathbf{q}).$$

From Newtonian mechanics, we have the equations of motion

$$\frac{d}{dt}\mathbf{q}_i = \mathbf{v}_i$$
$$m_i \frac{d}{dt}\mathbf{v}_i = \mathbf{F}_i$$

The Hamiltonian formulation of this system relies on *linear momenta* $\mathbf{p} \in \mathbb{R}^{3N}$, which is defined as

$$\mathbf{p} := \mathbf{M}\dot{\mathbf{q}}.$$

With this definition, we can write the equations of motion in their Hamiltonian formulation:

$$\frac{d}{dt}\mathbf{q} = \mathbf{M}^{-1}\mathbf{p}$$
$$\frac{d}{dt}\mathbf{p} = -\nabla_{\mathbf{q}} V(\mathbf{q}).$$

This Hamiltonian system has *Hamiltonian* (energy)

$$H(\mathbf{q}, \mathbf{p}) := \frac{\mathbf{p}^T\mathbf{M}^{-1}\mathbf{p}}{2} + V(\mathbf{q}). \tag{1}$$

The *phase space* of an N-body problem is the set of all possible positions and velocities of the particles. Given a phase space $\mathbb{R}^d \times \mathbb{R}^d$ of even dimension $2d \geq 2$ and a smooth $H : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, we can write the system of equations in their canonical form:

$$\frac{d}{dt}\mathbf{q} = +\nabla_{\mathbf{p}} H(\mathbf{q}, \mathbf{p})$$
$$\frac{d}{dt}\mathbf{p} = -\nabla_{\mathbf{q}} H(\mathbf{q}, \mathbf{p}).$$

2.2. **Constrained Mechanics.** Some mechanical systems operate in some constrained space. A system is *constrained* when the distance between two bodies in an N-body sytem is fixed. These rigid length constraints are of the form

$$\|\mathbf{q}_1 - \mathbf{q}_2\|^2 = L^2$$

Sometimes a system may have several constraints, thereby making the entire group of particles into a single rigid body.

We can extend Newton's equations of motion to the constrained case. Consider first the motion of a single particle. At any point in time, there are two types of forces acting on a moving particle: the *applied forces* that are defined by the potential energy fuction V, and the *constraint forces* that keep the particle on the constraint surface. The *principle of d'Alembert* states that the constraint force acts along the normal direction to the constraint surface, i.e. it acts along the direction of the gradient to the function g. The constraint force acts at the point of contact. Therefore, if we denote the constraint forces by $\mathbf{F}_g$, we have

$$\mathbf{F}_g = \lambda \nabla_{\mathbf{q}} g(\mathbf{q}),$$

where $\lambda \in \mathbb{R}$. Now we can rewrite the equations of motion using Newton's second law:

$$m\dot{\mathbf{v}} = -\nabla_{\mathbf{q}} V(\mathbf{q}) + \lambda \nabla_{\mathbf{q}} g(\mathbf{q}) \tag{2}$$
$$\dot{\mathbf{q}} = \mathbf{v} \tag{3}$$
$$g(\mathbf{q}) = 0. \tag{4}$$

The parameter λ is a parameter that is determined by the condition that $\mathbf{q}(t)$ satisfies (4) and is unique to each dynamic system.

There are numerous types of constraints, each of which has its own formulation of the equations of motion and produces certain properties of the solution. To narrow the discussion, we will concentrate only on systems that have *holonomic constraints.*

Definition 1. *A constraint that can be described by algebraic relations among the position variables of the system are called* ***holonomic constraints****. These constraints are defined by the equations of the form*

$$g_i(\boldsymbol{q}) = 0, \quad i = 1, 2, \ldots, m$$

for the smooth functions g_i.

Given m holonomic constraints, $g_i(\mathbf{q}) = 0, i = 1, 2, \ldots, m$, on a multiparticle system, each constraint has its own constraint force that acts in the normal direction to the constraint surface. Then we have the following form of the equations of motion:

$$\begin{aligned} \frac{d}{dt}\mathbf{q} &= \mathbf{v} \\ \mathbf{M}\frac{d}{dt}\mathbf{v} &= -\nabla_{\mathbf{q}} V(\mathbf{q}) - \sum_{i=1}^{m} \nabla_{\mathbf{q}} g_i(\mathbf{q})\lambda_i \\ 0 &= g_i(\mathbf{q}). \end{aligned}$$

Typically we assume that the gradients of the constraint functions, $\nabla_{\mathbf{q}} g_i(\mathbf{q})$, form a linearly independent set. Here we can introduce the vector function

$$\mathbf{g}(\mathbf{q}) = (g_i(\mathbf{q}), \ldots, g_m(\mathbf{q}))^T,$$

which has Jacobian matrix

$$\mathbf{G}(\mathbf{q}) = \mathbf{g}_{\mathbf{q}}(\mathbf{q}), \quad \textit{or} \quad \mathbf{G}(\mathbf{g})^T = \nabla_{\mathbf{q}}\mathbf{g}(\mathbf{q}),$$

and the vector of multipliers $\vec{\lambda} = (\lambda_1, \dots, \lambda_m)^T$. Then we can write the above system in a more compact form

$$\frac{d}{dt}\mathbf{q} = \mathbf{v}$$
$$\mathbf{M}\frac{d}{dt}\mathbf{v} = -\nabla_{\mathbf{q}}V(\mathbf{q}) - \mathbf{G}(\mathbf{q})^T \vec{\lambda}$$
$$\mathbf{0} = \mathbf{g}(\mathbf{q})$$

By introducing momentum $\mathbf{p} = \mathbf{M}\dot{\mathbf{q}}$, we can rewrite these equations as

$$\frac{d}{dt}\mathbf{q} = \mathbf{M}^{-1}\mathbf{p} \tag{5}$$
$$\frac{d}{dt}\mathbf{p} = -\nabla_{\mathbf{q}}V(\mathbf{q}) - \mathbf{G}(\mathbf{q})^T \vec{\lambda} \tag{6}$$
$$\mathbf{0} = \mathbf{g}(\mathbf{p}) \tag{7}$$

Note that Hamiltonian 1 can be *augmented* to include the forces of the constraints:

$$\tilde{H} = \frac{\mathbf{p}^T\mathbf{M}\mathbf{p}}{2} + V(\mathbf{q}) + \mathbf{g}(\mathbf{q})^T \vec{\lambda},$$

which can be expressed as

$$\tilde{H}(\mathbf{q}, \mathbf{p}) = H(\mathbf{q}, \mathbf{p}) + \mathbf{g}(\mathbf{q})^T \vec{\lambda}\ .$$

With this formulation, we can see that (5) is the gradient of $\tilde{H}$ with respect to $\mathbf{p}$ and (6) is the negative gradient with respect to $\mathbf{q}$, and $\vec{\lambda}$ is a vector of constants. this means that the canonical Hamiltonian equations of motion for a holonomic system are

$$\frac{d}{dt}\mathbf{q} = \nabla_{\mathbf{p}}H(\mathbf{q}, \mathbf{p})$$
$$\frac{d}{dt}\mathbf{p} = -\nabla_{\mathbf{q}}H(\mathbf{q}, \mathbf{p}) - \mathbf{G}(\mathbf{q})T \vec{\lambda}$$
$$\mathbf{0} = \mathbf{g}(\mathbf{q})$$

It must be noted that the introduction of constraints into a dynamic system also introduces new challenges in numerical discretization. For example, error propagation in numerical algorithms for constrained systems is more complicated than that of unconstrained systems. Therefore, the constraints should be resolved at each timestep. Two integration methods which we will discuss later, SHAKE and RATTLE, make sure that this happens with each iteration. However, before we can derive these two methods, we must derive the theory behind numerical integration. We begin with one-step methods.

3. One-step methods

One-step methods are low-order approximation methods that need only an initial point from which to start. That is, one-step methods approximate $\mathbf{z}_{n+1}$ using only $\mathbf{z}_n$. To begin, we need Taylor's Theorem.

Theorem 1. *Suppose $f \in C^n[a,b]$, that $f^{(p+1)}$ exists on $[a,b]$, and $x_0 \in [a,b]$. For every $x \in [a,b]$, there exists a number $\xi(x)$ between x_0 and x with $f(x) = P_p(x) + R_p(x)$, where*

$$\begin{aligned} P_p(x) &= f(x_0) + f'(x_0)(x-x_0) + \tfrac{f''(x_0)}{2}(x-x_0)^2 + \cdots + \tfrac{f^{(p)}(x_0)}{p!}(x-x_0)^p \\ &= \textstyle\sum_{k=0}^{p} \tfrac{f^{(k)}(x_0)}{k!}(x-x_0)^k \end{aligned}$$

and

$$R_p(x) = \frac{f^{(p+1)}(\xi(x))}{(p+1)!}(x-x_0)^{p+1}.$$

3.1. **Euler's method.** Suppose we are given the initial value problem

$$\dot{\mathbf{y}} = \mathbf{f}(t,\mathbf{y}), \quad a \le t \le b, \quad \mathbf{y}(a) = \alpha. \tag{8}$$

We want to approximate the solution to this initial value problem over the interval $[a,b]$ with $N-1$ equal intervals. The points at which approximations are taken are called *mesh points*, which are defined by

$$t_n = a + n\Delta t, \quad n = 0,1,2,\dots N.$$

Here, Δt is the *stepsize* and is given by $\Delta t = (b\text{-}a)/N = t_{n+1} - t_n$. Now suppose that $\mathbf{y}(t)$ is the exact solution to (8) and that it has two continuous derivatives on the interval $[a,b]$. Then using Taylor's Theorem, for each $n = 0,1,2,\dots,N-1$, we have

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + (t_{n+1}-t_n)\dot{\mathbf{y}}(t_i) + \frac{(t_{n+1}-t_n)^2}{2}\ddot{\mathbf{y}}(\xi_n),$$

where ξ_n is some number in the interval (t_n, t_{n+1}). Using the definition of Δt,

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + \Delta t\dot{\mathbf{y}}(t_n) + \frac{\Delta t^2}{2}\ddot{\mathbf{y}}(\xi_n).$$

Since we know that $\mathbf{y}(t)$ is a solution to (8), we have

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + \Delta t\mathbf{f}(t_n, \mathbf{y}(t_n)) + \frac{\Delta t^2}{2}\ddot{\mathbf{y}}(\xi_n).$$

We can approximate $\mathbf{z}_n \approx \mathbf{y}(t_n)$ for every $n = 1,2,\dots,N$ by deleting the remainder term in the Taylor expansion. Therefore, we are left with Euler's method:

$$\mathbf{z}^0 = \alpha\mathbf{z}^{n+1} = \mathbf{z}^n + \Delta t\mathbf{f}(t_n, \mathbf{z}_n), \quad n = 0,1,2,\dots,N-1.$$

There is one question that remains: does this algorithm approximates the exact solution with a bounded error after n iterations? In order to answer this question, we first need the definition of a Lipschitz condition.

Definition 2. *A function $\boldsymbol{f}(t,\boldsymbol{y})$ satisfies a Lipschitz condition in $\boldsymbol{y}$ on a set $D \subset \mathbb{R}^2$ with Lipschitz constantL if there exists an $L > 0$ such that*

$$\|\boldsymbol{f}(t,\boldsymbol{y}_1) - \boldsymbol{f}(t,\boldsymbol{y}_2)\| \le L\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|,$$

whenever $(t,\boldsymbol{y}_1)$, $(t,\boldsymbol{y}_2) \in D$.

Now we state the boundedness of Euler's method in the following theorem.

Theorem 2. *Suppose f is continuous and satisfies a Lipschitz condition with Lipschitz constant L on $D = \{(t,\boldsymbol{y}) | a \le t \le b, -\infty \le \boldsymbol{y} \le \infty\}$. Suppose also that there exists an M such that $\|\ddot{\boldsymbol{y}}(t)\| \le M$, for all $t \in [a,b]$. Let $\boldsymbol{y}(t)$ be the exact solution* *to the initial value problem, as before, and $\boldsymbol{z}(t)$ denote the approximation given by Euler's method. Then the error for Euler's method has a bound of the form*

$$\|\boldsymbol{y}(t_n) - \boldsymbol{z}_n\|$$

To prove the boundedness of Euler's method, we also need the following lemma

Lemma 1. *For every $x \leq -1$ and any positive m, we have $0 \leq (1+x)^m \leq e^{mx}$.*

Proof. Here we apply Taylor's Theorem to $f(x) = e^x$ with $x_0 = 0$, and $n = 1$ to get

$$e^x = 1 + x + \frac{1}{2}x^2 e^{\xi},$$

where $\xi \in [x, 0]$. We know

$$0 \leq 1 + x \leq 1 + x + \frac{1}{2}x^2 e^{\xi},$$

and since $(1+x) \geq 0$

$$0 \leq (1+x)^m \leq (e^x)^m = e^{mx}$$

□

Theorem 3. *Suppose f is continuous and satisfies a Lipschitz condition with constant L and that the exact solution y is twice continuously differentiable. Then the error for Euler's method admits a bound of the form*

$$\|\boldsymbol{y}(t_n) - \boldsymbol{z}_n\| \leq K(e^{t_n L} - 1)\Delta t \quad n = 1, 2, \ldots, N$$

where K is independent of the stepsize Δt.

Proof. First construct a recurrence relation for numerical error $\mathbf{e} = \mathbf{y}(t_n) - \mathbf{z}_n$. Now we expand both of these terms into their Taylor polynomials:

$$\mathbf{e}_{n+1} = (\mathbf{y}(t_n) + \Delta t\dot{\mathbf{y}}(t_n) + \frac{1}{2}\Delta t^2 \ddot{\mathbf{z}}(\tau)) - (\mathbf{z}_n + \Delta t \mathbf{f}(\mathbf{z}_n))$$

Since $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}(t_n))$,

$$\begin{aligned}\mathbf{e}_{n+1} &= (\mathbf{y}(t_n) + \Delta t f(\mathbf{y}(t_n)) + \frac{1}{2}\Delta t^2 \ddot{\mathbf{z}}(\tau)) - (\mathbf{z}_n + \Delta t \mathbf{f}(\mathbf{z}_n)) \\ &= (\mathbf{y}(t_n) - \mathbf{z}_n) + \Delta t(\mathbf{f}(\mathbf{y}(t_n)) - \mathbf{f}(\mathbf{z}_n)) + \frac{1}{2}\Delta t^2 \ddot{\mathbf{y}}(\tau),\end{aligned}$$

where $\tau \in [t_n, t_{n+1}]$. Using the triangle inequality and the Lipschitz condition,

$$\begin{aligned}\|\mathbf{e}_{n+1}\| &\leq \|\mathbf{y}(t_n) - \mathbf{z}_n\| + \Delta t \|\mathbf{f}(\mathbf{y}(t_n)) - \mathbf{f}(\mathbf{z}_n\| + \tfrac{1}{2}\Delta t^2 \|\ddot{\mathbf{y}}(\tau))\| \\ &\leq (1 + \Delta t L)\|\mathbf{e}_n\| + \tfrac{1}{2}\Delta t^2 \|\ddot{\mathbf{y}}(\tau)\|\end{aligned}$$

Since the solution is twice continuously differentiable, we may $\ddot{\mathbf{y}}$ on $[0, T]$ by a constant M. Note that a linear recurrence relation of the form

$$a_{n+1} \leq C a_n + D$$

satisfies the bound

$$a_n \leq C^n a_0 + \frac{C^n - 1}{C - 1}D$$

By setting $a_n = \|\mathbf{e}_n\|$,

$$\|\mathbf{e}_{n+1}\| \leq (1 + \Delta t L)a_n \leq (1 + \Delta t L)(C^n a_0 + \frac{C^n - 1}{C - 1}D).$$

Furthermore, since $C = 1 + \Delta t L$, $D = \frac{\Delta t^2 M}{2}$ and $a_0 = 0$

$$\|\mathbf{e}_{n+1}\| \leq \frac{(1 + \Delta t L^{n+1}) - (1 + \Delta t L)}{\Delta t L} \frac{\Delta t^2 M}{2}$$

Finally, by setting $K = \frac{M}{2L}$ and from Lemma 3.1, we know that $(1+\Delta t L)^n \leq e^{n\Delta t L}$, we can obtain the result:

$$\|\mathbf{e}_{n+1}\| \leq (e^{t_{n+1}L} - 1)\Delta t K$$

□

This theorem allows us to claim that Euler's method produces a bounded approximation for the exact solution $\mathbf{y}$ after n iterations. This claim can be proved to be true for other numerical methods by using a similar argument.

3.2. **Local Error.** When we compare the accuracy of different numerical methods, we look at their local error and their global error. The local error of a method is the error that is associated with one step from $\mathbf{z}_n$ to $\mathbf{z}_{n+1}$ for each $n = 0, 1, \ldots, N-1$. This error depends on the numerical method, Δt, and n. The global error is taken over the entire interval of integration, $[a, b]$.

Definition 3. *The numerical method*

$$\begin{aligned} &\boldsymbol{z}_0 = \alpha \\ &\boldsymbol{z}_{n+1} = \boldsymbol{z}_n + \Delta t \phi(t_n, \boldsymbol{z}_n), \quad n = 0, 1, \ldots N-1 \end{aligned}$$

has ***local truncation error***

$$\tau_{n+1}(\Delta t) = \frac{\boldsymbol{y}_{n+1} - (\boldsymbol{y}_n + \Delta t \phi(t_n, \boldsymbol{y}_n))}{\Delta t} = \frac{\boldsymbol{y}_{n+1} - \boldsymbol{y}_n}{\Delta t} - \phi(t_n, \boldsymbol{y}_n)$$

for all $n = 0, 1, \ldots, N-1$.

For example, for Euler's method we have

$$\begin{aligned} \tau_{n+1}(\Delta t) &= \tfrac{(\mathbf{y}(t_n) + \Delta t \mathbf{f}(t_n, \mathbf{y}(t_n)) + \frac{\Delta t^2}{2}\ddot{\mathbf{y}}(\xi_n)) - y_n}{\Delta t} - f(t_n, y_n) \\ &= \tfrac{\Delta t}{2}\ddot{\mathbf{y}}(\xi_n). \end{aligned}$$

Since $\ddot{\mathbf{y}}(\xi_n)$ is bounded by M on $[a, b]$,

$$|\tau_{n+1}(\Delta t)| \leq \frac{\Delta t}{2} M.$$

This implies that the local truncation error for Euler's method is $\mathcal{O}(\Delta t)$, i.e. first order.

If the local truncation error for a certain numerical method has order $p + 1$ at all t and $\mathbf{y}$ considered. Then the global error is order p.

3.3. **Taylor methods.** Recall that Euler's method was derived by using Taylor's Theorem with $p = 1$ and that it has order $\mathcal{O}(\Delta t)$. If we let $p > 1$ then we can derive numerical methods of higher order and obtain a better approximation of the exact solution $\mathbf{y}$. Methods that are derived in this way are called *Taylor methods.*

Suppose we have the initial value problem

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}), \quad a \leq t \leq b, \quad \mathbf{y}(a) = \alpha.$$

where $\mathbf{y}(t) \in C^{(p+1)}[a,b]$. Then using Taylor's Theorem, we can expand $\mathbf{y}(t)$ about t_n to its p^{th} Taylor polynomial. Evaluating at t_{n+1}, we have

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + \Delta t\mathbf{y}'(t_n) + \frac{\Delta t^2}{2}\mathbf{y}''(t_n) + \ldots + \frac{\Delta t^p}{p!}\mathbf{y}^{(p)}(t_n) + \frac{\Delta t^{p+1}}{(p+1)!}\mathbf{y}^{(p+1)}(\xi_n),$$

with $\xi_n \in (t_n, t_{n+1})$. We know

$$\mathbf{y}'(t) = \mathbf{f}(t,\mathbf{y}(t)), \quad \mathbf{y}''(t) = \mathbf{f}'(t,\mathbf{y}(t)), \quad \ldots, \quad \mathbf{y}^{(k)}(t) = \mathbf{f}^{(k-1)}(t,\mathbf{y}(t)),$$

so we can write

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + \Delta t\mathbf{f}(t_n,\mathbf{y}(t_n)) + \tfrac{\Delta t^2}{2}\mathbf{f}'(t_n,\mathbf{y}(t_n)) + \ldots + \tfrac{\Delta t^{(p)}}{p!}\mathbf{f}^{(p-1)}(t_n,\mathbf{y}(t_n)) \\ + \tfrac{\Delta t^{(p+1)}}{(p+1)!}\mathbf{f}^{(p)}(\xi_n,\mathbf{y}(\xi_n)).$$

Therefore, by deleting the remainder term, $\frac{\Delta t^{(p+1)}}{(p+1)!}\mathbf{f}^{(p)}(\xi_n,\mathbf{y}(\xi_n))$, we obtain a *Taylor polynomial of order* p:

$$\mathbf{z}_0 = \alpha \\ \mathbf{z}_{n+1} = \mathbf{z}_n + \Delta t T^{(p)}(t_n,\mathbf{z}_n),$$

where

$$T^{(p)}(t_n,\mathbf{z}_n) = \mathbf{f}(t_n,\mathbf{z}_n) + \frac{\Delta t}{2}\mathbf{f}'(t_n,\mathbf{z}_n) + \ldots + \frac{\Delta t^{p-1}}{(p-1)!}\mathbf{f}^{(p-1)}(t_n,\mathbf{z}_n).$$

Note that if the $(p+1)^{th}$ derivative of $\mathbf{y}(t)$ is bounded by M on $[a,b]$, this method has local truncation error

$$|\tau_{n+1}| = \frac{\Delta t^p}{(p+1)!}M.$$

3.4. **Runge-Kutta Methods.** Although Taylor methods have high-order local truncation error, they require computing and evaluating the derivatives of $\mathbf{f}(t,\mathbf{y})$. This process is not only time-consuming, it can be complicated. There is another family of numerical methods, however, that have the advantages of a high-order local truncation error without the disadvantage of the computation and evaluation of the derivatives of $f(t,\mathbf{y})$. These methods are *Runge-Kutta methods.* In order to derive these methods, we need the following theorem.

Theorem 4. *Suppose that $f(t,y)$ and all of its partial derivatives of order less than or equal to $p+1$ are continuous on $D = \{(t,y)|a \le t \le b, c \le y \le d\}$, and let (t_0,y_0) be a point in D. Then for all $(t,y) \in D$, there exists a value $\xi \in (t,t_0)$ and $\mu \in (y,y_0)$ such that*

$$f(t,y) = P_p(t,y) + R_p(t,y)$$

where

$$P_p(t,y) = f(t_0,y_0) + [(t-t_0)\tfrac{\partial f}{\partial t}(t_0,y_0) + (y-y_0)\tfrac{\partial f}{\partial y}(t_0,y_0)] + [\tfrac{(t-t_0)^2}{2}\tfrac{\partial^2 f}{\partial t^2}(t_0,y_0) \\ + (t-t_0)(y-y_0)\tfrac{\partial^2 f}{\partial t\partial y}(t_0,y_0) + \tfrac{(y-y_0)^2}{2}\tfrac{\partial^2 f}{\partial y^2}(t_0,y_0)] + \\ \ldots + \left[\tfrac{1}{p!}\textstyle\sum_{j=0}^{p}\binom{p}{j}(t-t_0)^{p-j}(y-y_0)^j\tfrac{\partial^p f}{\partial t^{p-j}\partial y^j}(t_0,y_0)\right]$$

and

$$R_p(t_p,y_p) = \frac{1}{(p+1)!}\sum_{j=0}^{p+1}\binom{p+1}{j}(t-t_0)^{p+1-j}(y-y_0)^j\frac{\partial^{p+1}f}{\partial t^{p+1-j}\partial y^j}(\xi,\mu).$$

The function $P_p(t, y)$ is called the p^{th} Taylor polynomial in two variables for the function f about (t_0, y_0) and $R_p(t, y)$ is the remainder term associated with $P_p(t, y)$.

Proof. For this proof, we reduce the Taylor polynomial in two variables down to the single variable case. Suppose we have the straight line

$$t = a + tx, \quad y = b + kx, \quad x \in [0, 1]$$

where $h = (t - t_0)$ and $k = (y - y_0)$. Then $f(t, y)$ is a function of a sinle variable, x, whose s^{th} order derivative is given by

$$\left(\frac{d}{dx}\right)^s f(x, y) = \left(\sum_{j=0}^{p} \binom{s}{j} h^{s-j} k^j \frac{\partial^s f}{\partial t^{s-j} \partial y^j}\right) f(x, y).$$

When $x = 0$, this becomes

$$\left(\sum_{j=1}^{s} \binom{s}{j} h^{s-j} k^j \frac{\partial^s f}{\partial t^{s-j} \partial y^j}\right) (t_0, y_0).$$

By theorem (3), the Taylor expansion of $f(x, y)$ in powers of x is

$$f(x, y) = f(t_0, y_0) + \sum_{s=1}^{p-1} \frac{x^s}{s!} \left(\sum_{j=1}^{s} \binom{s}{j} h^{s-j} k^j \frac{\partial^s f}{\partial t^{s-j} \partial y^j}\right) (t_0, y_0) + R_p,$$

where

$$R_p = \sum_{s=1}^{p-1} \frac{x^p}{p!} \left(\sum_{j=1}^{s} \binom{s}{j} h^{s-j} k^j \frac{\partial^s f}{\partial t^{s-j} \partial y^j}\right) (\xi, \mu),$$

with $\xi = t_0 + \Delta t \theta x$, $\mu = y_0 + \Delta t \theta x$, and θ is between 0 and 1. When $x = 1$, we have the desired result. □

Now we will derive a Runge-Kutta method using $T^2(t, y) = f(t, y) + \frac{\Delta t}{2} f'(t, y)$. We need to determine values for a_1, α_1, and β_1 such that $a_1 f(t + \alpha_1, y + \beta_1)$ approximates $T^{(2)}$ with error of order two. We know

$$f'(t, y) = \frac{df}{dt}(t, y) = \frac{\partial f}{\partial t}(t, y) + \frac{\partial f}{\partial y} \cdot y'(t)$$

and

$$y'(t) = f(t, y).$$

This implies

$$T^{(2)}(t, y) = f(t, y) + \frac{\Delta t}{2} \frac{\partial f}{\partial t}(t, y) + \frac{\Delta t}{2} \frac{\partial f}{\partial y}(t, y) \cdot f(t, y) \tag{9}$$

If we expand $a_1 f(t + \alpha_1, y + \beta_1)$ into its Taylor polynomial of degree one about (t, y), we get

(10)

$$a_1 f(t + \alpha_1, y + \beta_1) = a_1 f(t, y) + a_1 \alpha_1 \frac{\partial f}{\partial t}(t, y) + a_1 \beta_1 \frac{\partial f}{\partial y}(t, y) + a_1 \cdot R_1(t + \alpha_1, y + \beta_1),$$

where

$$R_1(t + \alpha_1, y + \beta_1) = \frac{\alpha_1^2}{2} \frac{\partial^2 f}{\partial t^2}(\xi, \mu) + \alpha_1 \beta_1 \frac{\partial^2 f}{\partial y \partial t}(\xi, \mu) + \frac{\beta_1^2}{2} \frac{\partial^2 f}{\partial y^2}(\xi, \mu)$$

with $\xi \in (t, t+\alpha_1)$, $\mu \in (y, y+\beta_1)$. By comparing (9) and (10), we can deduce the following:

$$\begin{aligned} f(t,y) &: a_1 = 1 \\ \frac{\partial f}{\partial t}(t,y) &: a_1\alpha_1 = \frac{\Delta t}{2} \\ \frac{\partial f}{\partial y}(t,y) &: a_1\beta_1 = \frac{\Delta t}{2} f(t,y). \end{aligned}$$

Therefore $a_1 = 1$, $\alpha_1 = \frac{\Delta t}{2}$, and $\beta_1 = \frac{\Delta t}{2} f(t,y)$, which yields the approximation

$$T^{(2)}(t,y) = f(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} f(t,y)) - R_1(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} f(t,y)),$$

where

$$\begin{aligned} R_1(t + \frac{\Delta t}{2}, y + \frac{\Delta t}{2} f(t,y)) = \frac{\Delta t^2}{8} \frac{\partial^2 f}{\partial t^2}(\xi,\mu) + \frac{\Delta t^2}{4} f(t,y) \frac{\partial^2 f}{\partial y \partial t}(\xi,\mu) \\ + \frac{\Delta t^2}{8} (f(t,y))^2 \frac{\partial^2 f}{\partial y^2}(\xi,\mu). \end{aligned}$$

Furthermore, if all of the second partial derivatives of f are bounded, then R_1 is $\mathcal{O}(\Delta t)$. Note that the error that results from the approximation of $T^{(2)}$ increases the error, but it does not increase the order of the error. By approximating $T^{(2)}(t,y)$ with $f(t+\alpha_1, y+\beta_1)$, we have produced the Runge-Kutta method called the *Midpoint Method*:

$$\begin{aligned} &\mathbf{z}_0 = \alpha \\ &\mathbf{z}_{n+1} = \mathbf{z}_n + \Delta t \mathbf{f}(t_n + \tfrac{\Delta t}{2}, \mathbf{z}_n + \mathbf{f}(t_n, \mathbf{z}_n)), \quad n = 0, 1, \ldots, N-1. \end{aligned}$$

We can compare other Taylor methods of order n to an approximation similar to the one in the example given above to produce Runge-Kutta methods of higher order.

In general, the class of s-stage Runge-Kutta methods for $\frac{d}{dt}\mathbf{z} = \mathbf{f}(\mathbf{z})$ is given by:

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \Delta t \sum_{j=1}^{s} b_j \mathbf{f}(\mathbf{Z}_j),$$

where

$$\mathbf{Z}_j = \mathbf{z}_n + \Delta t \sum_{k=1}^{s} a_{jk} \mathbf{f}(\mathbf{Z}_k)$$

for $j = 1, 2, \ldots, s$. Here, the coefficients $\{b_j\}$, $\{a_{jk}\}$ are determined by $T^{(n)}$ and its approximation. As example of a fourth-order Runge-Kutta method (RK4) is as follows:

$$\begin{aligned} \mathbf{Z}_1 &= \mathbf{z}_n \\ \mathbf{Z}_2 &= \mathbf{z}_n + \frac{\Delta t}{2} \mathbf{f}(\mathbf{Z}_1) \\ \mathbf{Z}_3 &= \mathbf{z}_n + \frac{\Delta t}{2} \mathbf{f}(\mathbf{Z}_2) \\ \mathbf{Z}_4 &= \mathbf{z}_n + \frac{\Delta t}{2} \mathbf{f}(\mathbf{Z}_3) \\ \mathbf{z}_{n+1} = \mathbf{z}_n + \frac{\Delta t}{6} &[\mathbf{f}(\mathbf{Z}_1) + 2\mathbf{f}(\mathbf{Z}_2) + 2\mathbf{f}(\mathbf{Z}_3) + \mathbf{f}(\mathbf{Z}_4)] \end{aligned}$$

4. Multistep Methods

One-step methods use information from only one previous timestep and accumulate error with each new $\mathbf{z}_n$. Therefore, using the more accurate data from other previous timesteps would lead to a numerical method that has less error. These types of methods are called *multistep methods.* Note that the majority of the methods discussed from here on out will be multistep methods.

Definition 4. *Given the initial value problem*

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}), \quad a \leq t \leq b, \quad \boldsymbol{y}(a) = \alpha$$

and starting values

$$\boldsymbol{z}_0 = \alpha, \quad \boldsymbol{w}_1 = \alpha_1, \quad \boldsymbol{w}_2 = \alpha_2, \quad \ldots, \quad \boldsymbol{z}_{m-1} = \alpha_{m-1}$$

an approximation $\boldsymbol{z}_{n+1}$ at t_{n+1} can be found using an ***m-step multistep method****, where m is an integer greater than 1, given by*

$$\begin{aligned}\boldsymbol{z}_{n+1} = a_{m-1}\boldsymbol{z}_n + a_{m-2}\boldsymbol{z}_{n-1} + \Delta t[b_m\boldsymbol{f}(t_{n+1}, \boldsymbol{z}_{n+1}) + b_{m-1}\boldsymbol{f}(t_n, \boldsymbol{z}_n) + \\ \ldots + b_0\boldsymbol{f}(t_{n+1-m}, \boldsymbol{z}_{n+1-m})],\end{aligned}$$

where $n = m-1, m, \ldots, N-1$, $\Delta t = \frac{(b-a)}{N}$, and $a_0, \ldots, a_{m-1}$ and $b_0, \ldots, b_m$ are constants.

Note that when $b_m = 0$, the approximation $\mathbf{z}_{n+1}$ is defined strictly in terms of previously determined values. In this case, the method is called *explicit.* When $b_m \neq 0$, $\mathbf{z}_{n+1}$ apears in its own definition and the method is called *implicit.*

Before we can set up a method by which multistep methods are derived, we need the following definition.

Definition 5. *Suppose we are given distinct points $x_0, x_1, \ldots, x_n$ for which we have values of a given function f. Then a unique polynomial of at most degree n, called the* ***interpolating polynomial****, exists such that*

$$f(x_k) = P(x_k), \quad k = 0, 1, \ldots, n.$$

, and is given by

$$P(x) = f(x_0)L_{n,0}(x) + \ldots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x),$$

where

$$\begin{aligned}L_{n,k}(x) &= \frac{(x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)} \\ &= \prod_{i=0,i\neq k}^{n} \frac{(x-x_i)}{(x_k-x_i)} \quad k = 0, 1, \ldots, n.\end{aligned}$$

Now we present a method for the derivation of multistep methods. Consider the initial value problem given in the definition above. If integrated over the interval $[t_n, t_{n+1}]$, then

$$\mathbf{y}(t_{n+1}) - \mathbf{y}(t_n) = \int_{t_n}^{t_{n+1}} \dot{\mathbf{y}}(t)dt = \int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))dt,$$

or

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + \int_{t_n}^{t_{n+1}} \mathbf{f}(t, \mathbf{y}(t))dt.$$

Note that we cannot integrate $\mathbf{f}(t, \mathbf{y}(t))$ directly since we do not know the solution $\mathbf{y}(t)$. Instead, we integrate an interpolating polynomial, $P(t)$, that is determined by some of its previously determined data points $(t_0, \mathbf{z}_0), (t_1, \mathbf{z}_1), \ldots, (t_n, \mathbf{z}_n)$. Because we can assume $\mathbf{y}(t_n) \approx \mathbf{z}_n$, we have

$$\mathbf{y}(t_{n+1}) \approx \mathbf{z}_n + \int_{t_n}^{t_{n+1}} P(t)dt.$$

Any interpolating polynomial can be used to derive a multistep method.

Note: We will not go into as much detail about the derivation of multistep methods as we did for one-step methods. The definition of a multistep method is sufficient for the understanding of this paper; however, a full background in the development of the Runge-Kutta methods is essential for the understanding of the next and following sections.

5. Partitioned Runge-Kutta methods

As we have seen, some canonical Hamiltonian systems admit a natural dichotomoy between positions and momenta. For these types of systems, which can be written in the form $\frac{d}{dt}\mathbf{u} = \mathbf{g}(\mathbf{u}, \mathbf{v})$, $\frac{d}{dt}\mathbf{v} = \mathbf{h}(\mathbf{u}, \mathbf{v})$, we can use different quadrature rules for each set of variables. These methods, called *s-stage Partitioned Runge-Kutta methods*, use two sets of coefficients $(\{\bar{b_j}\}, \{\bar{a_{jk}}\})$ and $(\left\{\tilde{b}_j\right\}, \{\tilde{a}_{jk}\})$:

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \Delta t \sum_{j=1}^{s} \bar{b_j}\mathbf{g}(\mathbf{U}_j, \mathbf{V}_j)$$

$$\mathbf{v}_{n+1} = \mathbf{u}_n + \Delta t \sum_{j=1}^{s} \tilde{b_j}\mathbf{h}(\mathbf{U}_j, \mathbf{V}_j)$$

where

$$\mathbf{U}_j = \mathbf{u}_n + \Delta t \sum_{k=1}^{s} \bar{a}_{jk}\mathbf{g}(\mathbf{U}_k, \mathbf{V}_k)$$

$$\mathbf{V}_j = \mathbf{v}_n + \Delta t \sum_{k=1}^{s} \tilde{a}_{jk}\mathbf{h}(\mathbf{U}_k, \mathbf{V}_k)$$

for $n = 1, 2, \ldots, s$. Note that in Hamiltonian dynamics,

$$\mathbf{u} = \mathbf{q}, \quad \mathbf{v} = \frac{d}{dt}\mathbf{q}$$

Consider a system of second order differential equations

$$\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}),$$

where $\mathbf{f}(\mathbf{q})$ does not depend on $\dot{\mathbf{q}}$. Then if we have Δt and points in time $t_n = t_0 + n\Delta t$ for $n = 0, 1, \ldots, N$ as before, using second order forward differencing, we have

$$\mathbf{f}(\mathbf{q}_n) = \frac{\mathbf{q}_{n+1} - 2\mathbf{q}_n + \mathbf{q}_{n-1}}{\Delta t^2}. \tag{11}$$

This equation can be rewritten and coupled with another equation to produce the Verlet scheme

$$\mathbf{q}_{n+1} = -\mathbf{q}_{n=1} + 2\mathbf{q}_n + \Delta t^2 \mathbf{f}(\mathbf{q}_n) + \mathcal{O}(\Delta t^4) \tag{12}$$

$$\dot{\mathbf{q}}_n = \frac{\mathbf{q}_{n+1} - \mathbf{q}_{n-1}}{2\Delta t} + \mathcal{O}(\Delta t^2) \tag{13}$$

By introducing velocity, $\dot{\mathbf{q}} = \mathbf{v}$, we can have the first order system

$$\dot{\mathbf{q}} = \mathbf{v}, \quad \dot{\mathbf{v}} = \mathbf{f}(\mathbf{q}).$$

We can introduce the discrete approximations:

$$\begin{aligned} \mathbf{v}_n &= \frac{\mathbf{q}_{n+1} - \mathbf{q}_{n-1}}{2\Delta t} \\ \mathbf{v}_{n-1/2} &= \frac{\mathbf{q}_n - \mathbf{q}_{n-1}}{\Delta t} \\ \mathbf{q}_{n-1/2} &= \frac{\mathbf{q}_n + \mathbf{q}_{n-1}}{2}, \end{aligned}$$

where the first two approximations make use of first order central differencing and the third equation is an average of the neighboring whole steps. The evaluation of $\mathbf{v}_{n-1/2}$ on a "staggered grid" to preserve second-order and symmetry. By plugging each of these three equations into (11), we get the following group of equations:

$$\begin{aligned} \mathbf{v}_{n+1/2} &= \mathbf{v}_n + \frac{\Delta t}{2} f(\mathbf{q}_n) \\ \mathbf{v}_{n+1} &= \mathbf{v}_{n+1/2} + \frac{\Delta t}{2} f(\mathbf{q}_{n+1}) \\ \mathbf{q}_{n+1} &= \mathbf{q}_n + \Delta t \mathbf{v}_{n+1/2}. \end{aligned}$$

Then since we have Newton's equations

$$\dot{\mathbf{q}} = \mathbf{v}, \quad \mathbf{M}\dot{\mathbf{v}} = -\nabla_{\mathbf{q}} V(\mathbf{q}),$$

we can reformulate the above equations into what is known as the velocity version of the Verlet method, or the Störmer-Verlet method:

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + \Delta t \mathbf{v}_{n+1/2} \\ \mathbf{M}\mathbf{v}_{n+1/2} &= \mathbf{M}\mathbf{v}_n - \frac{\Delta t}{2} \nabla_{\mathbf{q}} V(\mathbf{q}_n) \\ \mathbf{M}\mathbf{v}_{n+1} &= \mathbf{M}\mathbf{v}_{n+1/2} - \frac{\Delta t}{2} \nabla_{\mathbf{q}} V(\mathbf{q}_{n+1}). \end{aligned}$$

For further discretization, this system can be solved in terms of half-step velocities:

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + \Delta t \mathbf{v}_{n+1/2} \\ \mathbf{M}\mathbf{v}_{n+1/2} &= \mathbf{M}\mathbf{v}_{n-1/2} - \Delta t \nabla_{\mathbf{q}} V(\mathbf{q}_n). \end{aligned}$$

This method is called the Leapfrog method. Both Leapfrog and Störmer-Verlet are $\mathcal{O}(\Delta t^2)$.

6. Direct Discretization

The integration schemes we have discussed until now were developed for unconstrained dynamic systems. However, these methods can be generalized to develop integration schemes for constrained systems. These generalizations lead to a direct discretization of the equations of motion.

6.1. **SHAKE.** In 1976, Ryckaert, *et al.*, created an integration scheme aimed at performing molecular dynamics calculations with the Cartesian equations of motion. In their algorithm, they started with the Verlet algorithm, (12) – (13). They noted that a position, $\mathbf{q}$, can be written as the sum of two contributions:

$$\mathbf{q} = \mathbf{q}' + \delta\mathbf{q},$$

where $\mathbf{q}'$ is independent of $\vec{\lambda}$ and $\delta\mathbf{q}$ is linear in $\vec{\lambda}$. In this case, for each body in the system,

$$\begin{aligned} \mathbf{q}'_{n+1} &= -\mathbf{q}_{n-1} + 2\mathbf{q}_n - \frac{\Delta t^2}{m}\nabla_{\mathbf{q}}V(\mathbf{q}) \\ \delta\mathbf{q}_{n+1} &= \frac{\Delta t^2}{m}\sum_{j=1}^{l}\vec{\lambda}\,\nabla_{\mathbf{q}}\mathbf{g}(\mathbf{q}). \end{aligned}$$

Therefore, Ryckaert, *et al.*, produced the new discretization scheme, called SHAKE:

$$\begin{aligned} \mathbf{q} &= \mathbf{q}' + \delta\mathbf{q} \\ \mathbf{0} &= \mathbf{g}(\mathbf{q}), \end{aligned}$$

which can be rewritten in the form

$$\mathbf{M}\frac{\mathbf{q}_{n+1} - 2\mathbf{q}_n + \mathbf{q}_{n-1}}{\Delta t^2} = -\nabla_{\mathbf{q}}V(\mathbf{q}_n) - \mathbf{G}(\mathbf{q}_n)^T\,\vec{\lambda}^n \tag{14}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{q}_{n+1}). \tag{15}$$

Note that this form can also be derived by adopting the leapfrog method from the previous section to the constrained case.

To implement this scheme, solve (14) for $\mathbf{q}^{n+1}$ and insert it into (15). this will yield a system of equations with m equations and m unknown Lagrangian multipliers

$$\mathbf{0} = \tilde{\mathbf{g}}(\vec{\lambda}_n) := \mathbf{g}(\bar{\mathbf{q}}_{n+1} - \Delta t^2\mathbf{M}^{-1}\mathbf{G}(\mathbf{q}_n)^T\,\vec{\lambda}_n,$$

where

$$\bar{\mathbf{q}}_{n+1} := 2\mathbf{q}_n - \mathbf{q}_{n-1} - \Delta t^2\mathbf{M}^{-1}\nabla_{\mathbf{q}}V(\mathbf{q}_n)$$

is attained using an unconstrained step attained from one step of the leapfrog method.

We can reformulate this discretization scheme into a position-velocity form, despite the fact that the scheme integrates only position over time. To do this, we can use the same method as we did for Störmer-Verlet to set

$$\mathbf{v}_{n+1/2} = \frac{\mathbf{q}_{n+1} - \mathbf{q}_n}{\Delta t}, \quad \mathbf{v}_n = \frac{\mathbf{v}_{n-1/2} + \mathbf{v}_{n+1/2}}{2}.$$

This yields the position-velocity form of SHAKE:

$$\begin{aligned} \mathbf{q}_{n+1} &= \mathbf{q}_n + \Delta t\mathbf{v}_{n+1/2} \\ \mathbf{M}\mathbf{v}_{n+1/2} &= \mathbf{M}\mathbf{v}_{n-1/2} - \Delta t\nabla_{\mathbf{q}}V(\mathbf{q}_n) - \Delta t\mathbf{G}(\mathbf{q}_n)^T\,\vec{\lambda}_n \\ \mathbf{0} &= \mathbf{g}(\mathbf{q}_{n+1}) \\ \mathbf{v}_n &= \frac{1}{2}\mathbf{v}_{n+1/2} + \mathbf{v}_{n-1/2} \end{aligned}$$

The symplecticity and error of this method will be discussed later on.

6.2. **RATTLE.** In order to compare SHAKE with the method to be derived in this section, we will need the following definitions.

Definition 6. *A* ***configuration manifold*** $\mathcal{M}$ *is the space of all positions subject to the position constraints*

$$\mathcal{M} = \left\{ \boldsymbol{q} \in \mathbb{R}^d | g_j(\boldsymbol{q}) = 0, j = 1, \ldots, m \right\}.$$

Suppose $\mathbf{q} \in \mathcal{M}$. Each parameterization curve $\mathbf{q}(t)$ that contains $\bar{\mathbf{q}} = \mathbf{q}(t_0)$ and lies in $\mathcal{M}$ has a velocity vector $\mathbf{v}$ at $t = t_0$, such that $\bar{\mathbf{v}} = \dot{\mathbf{q}}(t_0)$. Note that since

$$g_j(\mathbf{q}(t)) \quad \forall t$$

we can write

$$\frac{d}{dt} g_j(\mathbf{q}(t)) = \nabla_{\mathbf{q}} g_j(\mathbf{q}(t)) \cdot \mathbf{q}(t) = \nabla_{\mathbf{q}} g_j(\mathbf{q}(t)) \cdot \mathbf{v}(t) = 0.$$

Definition 7. *The* ***tangent space*** *of* $\boldsymbol{q}$ *is a linear vector space defined by the set of all possible velocity vectors at the point* $\boldsymbol{q}$*:*

$$T_{\boldsymbol{q}}\mathcal{M} = \left\{ \bar{\boldsymbol{v}} \in \mathbb{R}^d | \nabla_{\boldsymbol{q}} g_j(\bar{\boldsymbol{q}}) \cdot \boldsymbol{v} = 0, j = 1, \ldots, m \right\}$$

Definition 8. *The space of all pairs* $(\boldsymbol{q}, \boldsymbol{v})$*, where* $\boldsymbol{q} \in \mathcal{M}$ *and* $\boldsymbol{v} \in T_{\boldsymbol{q}}\mathcal{M}$ *is the* ***tangent bundle*** *of* $\mathcal{M}$*, and is denoted* $T\mathcal{M}$*.*

Because SHAKE approximates the only the projection of $\mathbf{q}$, it is a mapping of the configuration manifold $\mathcal{M}$. The method discussed in this section, RATTLE, defines a mapping of the tangent bundle, $T\mathcal{M}$. The purpose of RATTLE is to correct SHAKE so that its solution lies on $T\mathcal{M}$ through projection of $\mathbf{v}_{n+1}$ onto the tangency constraint

$$\nabla_{\mathbf{q}} g_j(\mathbf{q}(t)) \cdot \bar{\mathbf{v}} = 0.$$

Hans C. Anderson introduced RATTLE in 1982 as a "velocity version" of SHAKE. The derivation of RATTLE is similar to that of SHAKE. However, instead of using the Verlet algorithm, Anderson uses Störmer-Verlet (recall that this is the velocity version of Verlet). We start by writing Störmer-Verlet in the following form:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \Delta t \mathbf{v}(t) - \frac{\Delta t^2}{2} \nabla_{\mathbf{q}} V(\mathbf{q}_n)$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n - \frac{\Delta t}{2} \left[\nabla_{\mathbf{q}} V(\mathbf{q}_n) + \nabla_{\mathbf{q}} V(\mathbf{q}_{n+1}) \right]$$

This implies that for constrained dynamics, we have

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \Delta t \mathbf{v}(t) + \frac{\Delta t^2}{2} \left[-\nabla_{\mathbf{q}} V(\mathbf{q}_n) + \mathbf{G}(\mathbf{q}_n, \mathbf{v}_n)^T \vec{\lambda} \right] \tag{16}$$

$$\begin{aligned} \mathbf{v}_{n+1} = \mathbf{v}_n &+ \frac{\Delta t}{2} \left[-\nabla_{\mathbf{q}} V(\mathbf{q}_n) + \mathbf{G}(\mathbf{q}_n, \mathbf{v}_n) \right] \\ &+ \frac{\Delta t}{2} [\nabla_{\mathbf{q}} V(\mathbf{q}_{n+1}) + \mathbf{G}(\mathbf{q}_{n+1}, \mathbf{v}_{n+1})] \end{aligned} \tag{17}$$

Note that in (16), we need to know $\mathbf{v}(t)$ before we can calculate $\mathbf{G}_n^T$ and that in (17), we need to know $\mathbf{G}_{n+1}^T$ before we can calculate $\mathbf{v}_{n+1}$. However, there is no need to use the same approximation for $\mathbf{G}^T$ in both the position and velocity equations. Therefore, we can choose an approximation for the $\mathbf{G}^T$ in (16) so that $\mathbf{q}_{n+1}$ satisfies the constraints either exactly or to within a desired precision. We can do the same for (17) by using a different approximation for $\mathbf{G}^T$. This procedure yeilds

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \Delta t \mathbf{v}_n + \frac{\Delta t^2}{2}\left[-\nabla_{\mathbf{q}} V(\mathbf{q}_n) + \mathbf{G}(\mathbf{q}_n)^T \overrightarrow{\lambda}^{n}_{(r)}\right]$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \frac{\Delta t}{2}\left[-\nabla_{\mathbf{q}} V(\mathbf{q}_n) + \mathbf{G}(\mathbf{q}_n)^T \overrightarrow{\lambda}^{n}_{(r)} - \nabla_{\mathbf{q}} V(\mathbf{q}_{n+1}) + \mathbf{G}(\mathbf{q}_{n+1})^T \overrightarrow{\lambda}^{n}_{(v)}\right]$$

These equations define RATTLE. They can also be written in the form

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \Delta t \mathbf{v}_{n+1/2} \tag{18}$$

$$\mathbf{M}\mathbf{v}_{n+1/2} = \mathbf{M}\mathbf{v}_n - \frac{\Delta t}{2}\nabla_{\mathbf{q}} V(\mathbf{q}_n) - \frac{\Delta t}{2}\mathbf{G}(\mathbf{q}_n)^T \overrightarrow{\lambda}^{n}_{(r)} \tag{19}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{q}_{n+1}) \tag{20}$$

$$\mathbf{M}\mathbf{v}_{n+1} = \mathbf{M}\mathbf{v}_{n+1/2} - \frac{\Delta t}{2}\nabla_{\mathbf{q}} V(\mathbf{q}_{n+1}) - \frac{\Delta t}{2}\mathbf{G}(\mathbf{q}_{n+1})^T \overrightarrow{\lambda}\,(v)^n \tag{21}$$

$$\mathbf{0} = \mathbf{G}(\mathbf{q}_{n+1})\mathbf{v}_{n+1}. \tag{22}$$

The multipliers are chosen so that the position and velocity constraints are enforced. Thus RATTLE requires that both position and velocity satisfy the constraints at each timestep.

6.3. **The Multipliers.** To solve for the multipliers $\overrightarrow{\lambda}^n$, one has to solve the system of constraints. If the constraints are linear, the system can be solved using a linear solver. But in most cases, the system of constraints is nonlinear. These nonlinear systems have the form

$$\mathbf{g}(\overline{\mathbf{q}}^{n+1} - \Delta t^2 \mathbf{M}^{-1} \sum_{i=1}^{m} \mathbf{G}(\mathbf{q}^n)^T \lambda^n) = \mathbf{0}$$

where $\overline{\mathbf{q}}^{n+1}$ represents an unconstrained step using leapfrog or Stormer-Verlet.

6.3.1. *SHAKE iteration.* The multipliers of SHAKE can be found by using a coordinate resetting iteration that is basically a Gauss-Seidel iteration. One multiplier is adjusted at each iteration. Here we will switch notation by putting the step of the notation, n, into the superscript and the index of the components, i, in the subscript. Let g_i be the i^{th} component of g. Then $\mathbf{G}_i = \nabla_{\mathbf{q}} g_i(\mathbf{q})$ is the i^{th} row of the constraint Jacobian matrix.

First set

$$\mathbf{Q} := \mathbf{q}^n + \Delta t \mathbf{v}^{n-1/2} - \Delta t^2 \mathbf{M}^{-1} \nabla_{\mathbf{q}} V(\mathbf{q}^n)$$

This is equivalent to taking $\overrightarrow{\lambda}^n$ equal to zero in the system of constraints above.

Next, cycle through the list of constraints and correct each one. Compute *offset* $\Delta\Lambda_i$ to satisfy the i^{th} linearized constraint equation

$$\Delta\Lambda_i := \frac{g_i(\mathbf{Q})}{\mathbf{G}_i(\mathbf{Q})\mathbf{M}^{-1}\mathbf{G}_i(\mathbf{q}^n)}$$

and update $\mathbf{Q}$

$$\mathbf{Q} := \mathbf{Q} - \mathbf{M}^{-1}\mathbf{G}_i(\mathbf{q}^n)^T \Delta\Lambda_i$$

Repeat this cycle until all constraint residuals are within some tolerance, i.e. $g_i(\mathbf{Q}) < tol$. Then set $\mathbf{q}^{n+1} = \mathbf{Q}$

Given a good enough initial guess (or small enough stepsize) this iterative method converges.

Now we talk about the True Newton iteration and quasi-Newton iteration. Both of these methods (as well as others) rely on the vector of offesets $\Delta\Lambda = \{\Delta\Lambda_i\}$

$$\Delta\Lambda := \mathbf{R}^{-1}\mathbf{g}(\mathbf{Q})$$

For the true Newton iteration, $\mathbf{R} = \mathbf{G}(\mathbf{Q})\mathbf{M}^{-1}\mathbf{G}(\mathbf{q}^n)^T$, where $\mathbf{Q}$ is the updated approximation from

$$\mathbf{Q} := \mathbf{Q} - \mathbf{M}^{-1}\mathbf{G}(\mathbf{q}^n)^T\Delta\Lambda$$

For the quasi-Newton iteration, $\mathbf{R} = \mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T$, where $\mathbf{G} = \mathbf{G}(\mathbf{q}^k)$ for $\mathbf{q}^k$ computed at timestep t^k. Here, $\mathbf{G}$ is updated as needed for convergence.

6.3.2. *RATTLE iteration.* RATTLE has two multipliers for each timestep: one for correcting the position approximations and another for correcting the velocity approximationss. One multiplier is adjusted at a time so that the constraints are satisfied within some given tolerance, *tol*.

The Anderson provided an iteration for the solution of the multipliers in the introductory paper of RATTLE. In a general constrained system, let i and j be two particles or point masses such that there is a constraint g_{ij} between them, $i, j = 1, 2, ...N$. First, recall that

$$\mathbf{F}_i^n = -\nabla_{\mathbf{q}_i} V(\mathbf{q}_1^n, \mathbf{q}_2^n, \ldots, \mathbf{q}_N^n)$$

and define

$$h_{ij} = \Delta t \overset{\rightarrow n}{\lambda}_{(r)ij}$$

$$k_{ij} = \Delta t \overset{\rightarrow n+1}{\lambda}_{(v)ij}$$

Then (19) becomes

$$\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^n + \frac{\Delta t}{2m_i}\mathbf{F}_i^n - \frac{h_{ij}}{2m_i}\mathbf{G}(\mathbf{q}_i^n)$$

and (21) becomes

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^{n+1/2} - \frac{\Delta t}{2m_i}\mathbf{F}_i^{n+1} - \frac{k_{ij}}{2m_i}\mathbf{G}(\mathbf{q}_i^{n+1})$$

To start iteration, let

$$\mathbf{v}_i^{n+1/2} = \mathbf{v}^n + \frac{\Delta t}{2m_i}\mathbf{F}_i^n$$

The iterative loop begins here. Since the constraints can be solved in any order, pick one that involves point masses i and j. Define

$$\mathbf{s} = \mathbf{q}_i^n + \Delta t\mathbf{v}_i^{n+1/2} - \mathbf{q}_j^n - \Delta t\mathbf{v}_j^{n+1/2}$$

This is the approximation of the vector displacement of point masses i and j. In most cases, the constraints are distances, so let d_{ij} be the distance between point masses i and j. If

$$|\mathbf{s}|^2 - d_{ij}^2 < tol$$

then the constraint is solved within the acceptable tolerace, so pick a new constraint and continue the process. Otherwise, $\mathbf{v}_i^{n+1/2}$ and $\mathbf{v}_j^{n+1/2}$ need to be corrected by amounts proportional to h. Update the position vectors by

$$\mathbf{q}_i^t = \mathbf{q}_i^n + \Delta t \left[\mathbf{v}_i^{n+1/2} - \frac{h}{m_i}\mathbf{q}_{ij} \right]$$

and

$$\mathbf{q}_j^t = \mathbf{q}_j^n + \Delta t \left[\mathbf{v}_j^{n+1/2} + \frac{h}{m_j}\mathbf{q}_{ij} \right]$$

where $\mathbf{q}_{ij}^n = \mathbf{q}_i^n - \mathbf{q}_j^n$. Let these be the new values for $\mathbf{q}_i^{n+1}$ and $\mathbf{q}_j^{n+1}$, respectively. Note that the goal is to choose h so that $|\mathbf{q}_i^t - \mathbf{q}_j^t|^2 = d_{ij}^2$. That is, we want their difference to be within the given tolerance. Solving for h we have

$$h = \frac{|\mathbf{s}|^2 - \mathbf{d}_{ij}^2}{2\Delta t \left[\mathbf{s} \cdot \mathbf{q}_{ij}^n\right] (m_i^{-1} + m_j^{-1})}$$

where the new values for $\mathbf{q}_i^t$ and $\mathbf{q}_j^t$ are used to calculate $\mathbf{s}$ and the second order terms in h are neglected. Once the correct h is found, update $\mathbf{v}_i^{n+1/2}$ and $\mathbf{v}_j^{n+1/2}$ by

$$\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^n + \frac{\Delta t}{2m_i}\mathbf{F}_i^n - \frac{h}{m_i}\mathbf{q}_{ij}^n$$

and

$$\mathbf{v}_j^{n+1/2} = \mathbf{v}_j^n + \frac{\Delta t}{2m_j}\mathbf{F}_j^n + \frac{h}{m_j}\mathbf{q}_{ij}^n$$

respectively. Then go to the beginning of the iterative loop, pick a new constraint, and repeat the procedure. Note that these are the values to be used in the iterative loop for the new constraint. This ensures that by the end of the iteration, all of the constraints are satisfied, not just the latest one.

Now we have to solve for $\overset{\rightarrow}{\lambda}{}_{(v)}^{n+1}$. The procedure is similar to the previous iteration. Start by letting

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^{n+1/2} + \frac{\Delta t}{2m_i}\mathbf{F}_i^{n+1}$$

Note that the $\mathbf{v}_i^{n+1/2}$ that satisfies all of the position constraints due to the previous iterative procedure for $\overset{\rightarrow}{\lambda}{}_{(r)}^{n}$ is used here. The iterative loop begins. Again, the constraints can be solved in any order so pick one that involves point masses i and j. If

$$\mathbf{q}_{ij}^{n+1} \cdot \mathbf{v}_{ij}^{n+1} < tol$$

where tol is some given tolerance, then this constraint is satisfied. Pick a new constraint and begin the iterative loop. Otherwise, we need to correct $\mathbf{v}_i$ and $\mathbf{v}_j$ by amounts proportional to k. Let

$$\mathbf{v}_i^t = \mathbf{v}_i^{n+1} - \frac{k}{m_i}\mathbf{q}_{ij}^{n+1}$$

and

$$\mathbf{v}_j^t = \mathbf{v}_j^{n+1} + \frac{k}{m_j}\mathbf{q}_{ij}^{n+1}$$

be the new values for $\mathbf{v}_i^{n+1}$ and $\mathbf{v}_j^{n+1}$, respectively. Note that the goal here is to choose k so that $\mathbf{v}_i^t - \mathbf{v}_j^t$ is perpendicular to $\mathbf{q}_{ij}^{n+1}$. That is, their dot product should be zero, or within *tol* of zero. Thus we want

$$k = \frac{\mathbf{q}_{ij}^{n+1} \cdot \left[\mathbf{v}_i^{n+1} - \mathbf{v}_j^{n+1}\right]}{d_{ij}^2 (m_i^{-1} + m_j^{-1})}.$$

Once the correct k is found, replace $\mathbf{v}_i^{n+1}$ by $\mathbf{v}_i^t$ and $\mathbf{v}_j^{n+1}$ by $\mathbf{v}_j^t$. Pick a new constraint and begin the iterative loop with these new values.

6.4. **Symplecticity.**

In this section, we derive the definition of symplecticity and discuss its implications. Then we will show that SHAKE and RATTLE are both symplectic maps. We begin with a couple of definitions.

Definition 9. *A smooth map $\Psi : \mathbb{R}^{2d} \to \mathbb{R}^{2d}$ on the phase space of a system is called a **symplectic map** with respect to the constant and invertible structure matrix $\boldsymbol{J}$ if its Jacobian $\Psi_{\boldsymbol{z}}(\boldsymbol{z})$ satisfies*

$$[\Psi_{\boldsymbol{z}}(\boldsymbol{z})]^T \boldsymbol{J}^{-1} \Psi_{\boldsymbol{z}}(\boldsymbol{z}) = \boldsymbol{J}^{-1}$$

for all $\boldsymbol{z}$ in the domain of definition of Ψ.

Definition 10. *The **wedge product** of two differentials df and dg is an operator defined by*

$$(df \wedge dg)(\xi, \eta) := dg(\xi) df(\eta) - df(\xi) dg(\eta)$$

If we have a location $\mathbf{z}$, then we can apply a coordinate transformation $\Psi : \mathbb{R}^m \to \mathbb{R}^m$

$$\widehat{\mathbf{z}} = \Psi(\mathbf{z}).$$

Note that this implies

$$d\widehat{\mathbf{z}} = \Psi_{\mathbf{z}}(\mathbf{z}) d\mathbf{z}.$$

Let $\mathbf{z} = (\mathbf{q}, \mathbf{p}) \in \mathbb{R}^{2d}$. Then we have the transformation

$$\widehat{\mathbf{q}} = \Psi^1(\mathbf{q}, \mathbf{p}) \tag{23}$$

$$\widehat{\mathbf{p}} = \Psi^2(\mathbf{q}, \mathbf{p}). \tag{24}$$

Then we can write

$$d\widehat{\mathbf{q}} \wedge d\widehat{\mathbf{p}} = d\mathbf{q} \wedge d\mathbf{p}, \tag{25}$$

where

$$d\widehat{\mathbf{q}} = \Psi_{\mathbf{q}}^1(\mathbf{q}, \mathbf{p}) d\mathbf{q} + \Psi_{\mathbf{p}}^1(\mathbf{q}, \mathbf{p}) d\mathbf{p}$$
$$d\widehat{\mathbf{p}} = \Psi_{\mathbf{q}}^2(\mathbf{q}, \mathbf{p}) d\mathbf{q} + \Psi_{\mathbf{p}}^2(\mathbf{q}, \mathbf{p}) d\mathbf{p}$$

Before we come up with a symplecticity condition using the wedge product, we will need to know the properties of the wedge product. If $d\mathbf{a}, d\mathbf{b}, d\mathbf{c}$ are k-vectors of differential one-forms on $\mathbb{R}^m$, then the wedge product has the following properties:

(1) *Skew-symmetry*

$$d\boldsymbol{a} \wedge d\boldsymbol{b} = -d\boldsymbol{b} \wedge d\boldsymbol{a}$$

(2) *Bilinearity*

$$d\boldsymbol{a} \wedge (\alpha d\boldsymbol{b} + \beta d\boldsymbol{c}) = \alpha d\boldsymbol{a} \wedge d\boldsymbol{b} + \beta d\boldsymbol{a} \wedge d\boldsymbol{c}$$

(3) *Rule of matrix multiplication*

$$d\boldsymbol{a} \wedge (\boldsymbol{A} d\boldsymbol{b}) = (\boldsymbol{A}^T d\boldsymbol{a}) \wedge d\boldsymbol{b}$$

is a consequence of property 2 and the definition, and holds for any $k \times k$ matrix $\mathbf{A}$.

Lemma 2. *Let du be any arbitrary differential in $\mathbb{R}^n$ and let A be any $n \times n$ real symmetric matrix. Then $du \wedge (Adu) = 0$.*

Proof. Using the skew-symmetric property and the rule of matrix multiplication, we have

$$\begin{aligned} du \wedge (Adu) &= (A^T du) \wedge du \\ &= -du \wedge (A^T) du \\ &= -(Adu) \wedge du \end{aligned}$$

The only way this is possible is if $du \wedge (Adu) = 0$. □

Now we can prove the following theorem.

Theorem 5. *A transformation Ψ as defined above is symplectic if $d\widehat{\boldsymbol{q}} \wedge d\widehat{\boldsymbol{p}} = d\boldsymbol{q} \wedge d\boldsymbol{p}$.*

Proof. First note that by the definition of the wedge prduct,

$$\begin{aligned} (\mathbf{J}^{-1} d\mathbf{z}) \wedge d\mathbf{z} &= \textstyle\sum_{i-1}^{d} [dz_i \wedge dz_{d+i} - dz_{d+i} \wedge dz_i] \\ &= \textstyle\sum_{i=1}^{d} [dq_i \wedge dp_i - dp_i \wedge dq_i] \\ &= 2d\mathbf{q} \wedge d\mathbf{p} \end{aligned}$$

Then $d\widehat{\mathbf{q}} \wedge d\widehat{\mathbf{p}} - d\mathbf{q} \wedge d\mathbf{p}$ is equivalent to

$$(\mathbf{J}^{-1} d\widehat{\mathbf{z}}) \wedge d\widehat{\mathbf{z}} = (\mathbf{J}^{-1} d\mathbf{z}) \wedge d\mathbf{z}.$$

Since

$$d\widehat{\mathbf{z}} - \Psi_{\mathbf{z}}(\mathbf{z}) d\mathbf{z},$$

we have

$$(\mathbf{J}^{-1} d\widehat{\mathbf{z}}) \wedge d\widehat{\mathbf{z}} = (\mathbf{J}^{-1} \Psi_{\mathbf{z}}(\mathbf{z}) d\mathbf{z}) \wedge (\Psi_{\mathbf{z}}(\mathbf{z}) d\mathbf{z}).$$

By property 3,

$$(\mathbf{J}^{-1} d\widehat{\mathbf{z}}) \wedge d\widehat{\mathbf{z}} = (\Psi_{\mathbf{z}}(\mathbf{z})^T \mathbf{J}^{-1} \Psi_{\mathbf{z}}(\mathbf{z}) d\mathbf{z}) \wedge d\mathbf{z}.$$

Therefore,

$$\mathbf{J}^{-1} = \Psi_{\mathbf{z}}(\mathbf{z})^T \mathbf{J}^{-1} \Psi_{\mathbf{z}}(\mathbf{z}).$$

□

Here we switch back to our original notation, with the step, n, in the subscript. Note that most numerical integrators preserve symplecticity up to a certain error. A numerical method is called a *symplectic integrator* if the symplecticness condition

$$d\boldsymbol{q}_{n+1} \wedge d\boldsymbol{p}_{n+1} = d\boldsymbol{q}_n \wedge d\boldsymbol{p}_n$$

is preserved *exactly*.

Symplectic maps preserve certain properties of the dynamical system which they are integrating. Energy, for example, can be preserved through the use of a symplectic integrator. The symplecticness of SHAKE and RATTLE was analyzed by Leimkuhler and Skeel in 1994. They derive what they call *velocity-level SHAKE* for the one-dimenstional case in the following way.

If the SHAKE algorithm is iterated to convergence, then we have

$$q_{n+1} = 2q_n - q_{n-1} - \Delta t^2 \nabla_q V(q_n) + \Delta t^2 g'(q_n)^T \lambda_n$$
$$g(q_{n+1}) = 0$$

Set $p_{n+1/2} = (q_{n+1} - q_n)/\Delta t$ to obtain the leapfrog form that has error of order three

$$q_{n+1} = q_n + \Delta t p_{n+1/2}$$
$$p_{n+1/2} = p_{n-1/2} - \Delta t \nabla_q V(q_n) + \Delta t g'(q_n)^T \lambda_n$$
$$g(q_{n+1}) = 0$$

Define further $p_n = (q_{n+1} - q_{n-1})/(2\Delta t)$ to get

$$q_{n+1} = q_n + \Delta t p_{n+1/2}$$
$$p_{n+1/2} = p_n - \frac{\Delta t}{2} \nabla_q V(q_n) + \frac{\Delta t}{2} g'(q_n)^T \lambda_n$$
$$g(q_{n+1}) = 0$$
$$p_{n+1} = p_{n+1/2} - \frac{\Delta t}{2} \nabla_q V(q_{n+1}) + \frac{\Delta t}{2} g'(q_{n+1})^T \lambda_{n+1}$$

This system of equations is called velocity-level SHAKE (VS). Note that this cannot be a symplectic method as defined here. Although $g(q_n) = 0$ at every grid point, the hidden constraint $g'(q_n)M^{-1}p_n = 0$ will typically fail to be satisfied. We can therefore only show that VS preserves the wedge product.

VS can be viewed as a one-step mapping, where the differentials obey

$$dq_{n+1} = dq_n + \Delta t dp_{n+1/2} \tag{26}$$
$$dp_{n+1/2} = dp_n - \frac{\Delta t}{2} d\nabla_q V(q_n) + \frac{\Delta t}{2} d(g'(q_n)^T \lambda_{n+1})$$
$$g'(q_{n+1}) dq_{n+1} = 0$$
$$dp_{n+1} = dp_{n+1/2} - \frac{\Delta t}{2} d\nabla_q V(q_{n+1}) + \frac{\Delta t}{2} d(g'(q_{n+1})^T \lambda_{n+1})$$

It remains to be shown that VS preserves the wedge product. To prove this we will need the following lemma.

Lemma 3. $dq_n \wedge d(q'(q_n)^T \lambda_n) = 0$

Proof.

$$dq_n \wedge d(g'(q_n)^T \lambda_n) = dq_n \wedge g'(q_n)^T d\lambda_n + \sum_{i=0}^{m} \lambda_n^i dq_n \wedge \Gamma_n^i dq_n$$

where λ_n has been indexed by a superscript and Γ_n^i is the Hessian of the ith constraint function. We know $g(q_n) = 0$, which implies $g(q_n)dq_n = 0$. So

$$dq_n \wedge g'(q_n)^T d\lambda_n = g'(q_n)dq_n \wedge d\lambda_n = 0$$

. Thus the first term equals zero. All of the terms of the summation can be eliminated by 2 $\square$

Now we can show that VS preserves the wedge product. Let V'' be the Hessian of V. So $d\nabla_q V(q_n) = V''(q_n)dq_n$. Then we have

$$\begin{aligned} &dq_{n+1} \wedge dp_{n+1} \\ &\quad = dq_{n+1} \wedge (dp_{n+1/2} - \tfrac{\Delta t}{2} V''(q_{n+1})dq_{n+1}) \\ &\quad = dq_{n+1} \wedge dp_{n+1/2} - \tfrac{\Delta t}{2} dq_{n+1} \wedge V''(q_{n+1})dq_{n+1} + \\ &\qquad \tfrac{\Delta t}{2} dq_{n+1} \wedge d(g'(q_{n+1}^T \lambda_{n+1}) \end{aligned}$$

The second and third terms in this equation can be eliminated by use of Lemma 2 and Lemma 3, respectively. So we have

$$dq_{n+1} \wedge dp_{n+1} = dq_{n+1} \wedge dp_{n+1/2}$$

From (26) we have

$$\begin{aligned} &dq_{n+1} \wedge dp_{n+1} = (dq_n + \Delta t dp_{n+1/2}) \wedge dp_{n+1/2} \\ &\quad = dq_n \wedge dp_{n+1/2} \\ &\quad = dq_n \wedge (dp_n - \tfrac{\Delta t}{2} d\nabla_q V(q_n) + \Delta t d(g'(q_n)^T \lambda_n)) \end{aligned}$$

Here, again, the second and third terms can be eliminated by Lemma 2 and Lemma 3, respectively. So we have

$$dq_{n+1} \wedge dp_{n+1} = dq_n \wedge dp_n$$

which means VS preserves the wedge product.

We know that SHAKE and RATTLE are equal, but RATTLE satisifies both the position and velocity constraints at discretization points. Since symplecticness is directly related to the wedge product, the proof that RATTLE is symplectic is similar to the proof that SHAKE preserves the wedge product.

The converged RATTLE algorithm can be expressed as

$$q_{n+1} = q_n + \Delta t p_{n+1/2}$$

$$p_{n+1/2} = p_n - \frac{\Delta t}{2} \nabla_q V(q_n) + \frac{\Delta t}{2} g'(q_n)^T \lambda_{(r)}^n$$

$$p_{n+1} = p_{n-1/2} - \frac{\Delta t}{2} \nabla_q V(q_{n+1}) + \frac{\Delta t}{2} g'(q_{n+1})^T \lambda_{(v)}^{n+1}$$

This set of equations is called a VR step that satisfies constraints

$$g(q_{n+1}) = 0$$

$$g'(q_{n+1})p_{n+1} = 0$$

.

If we write

$$q_{n+1} = q_n + \Delta t p_{n+1/2}$$

then

$$p_{n+1/2} = p_{n-1/2} - \Delta t \nabla_q V(q_n) + \frac{\Delta t}{2} g'(q_n)^T (\lambda^n_{(r)} + \lambda^n_{(v)}).$$

So we have

$$\begin{aligned} dq_{n+1} \wedge dp_{n+1} = dq_{n+1} \wedge dp_{n+1/2} - \tfrac{\Delta t}{2} dq_{n+1} \wedge d\nabla_q V(q_{n+1}) \\ + \tfrac{\Delta t}{2} dq_{n+1} \wedge dg'(q_{n+1})^T \lambda^{n+1}_{(v)} \end{aligned}$$

The second and third terms in this equation can be eliminated by Lemma 2 and Lemma 3, respectively. Then we have

$$\begin{aligned} & dq_{n+1} \wedge dp_{n+1} = dq_n \wedge dp_{n-1/2} \\ & \quad = dq_n \wedge dp_{n+1/2} + \Delta t dq_n \wedge d\nabla_q V(q_n) - \tfrac{\Delta t}{2} dq_n \wedge dg'(q_n)^T (\lambda_n^{(r)} + \lambda_n^{(v)}) \end{aligned}$$

The second term is eliminated by Lemma 2 and the two last terms are eliminated by Lemma 3. So we are left with

$$\begin{aligned} & dq_{n+1} \wedge dp_{n+1} = dq_n \wedge dp_{n+1/2} \\ & \quad = dq_n \wedge dp_n - \tfrac{\Delta t}{2} dq_n \wedge d\nabla_q V(q_n) + \tfrac{\Delta t}{2} dq_n \wedge dg'(q_n)^T \lambda_n \end{aligned}$$

Here, again, the second and third terms are eliminated by Lemma 2 and Lemma 3, respectively. Thus we have

$$dq_{n+1} \wedge dp_{n+1} = dq_n \wedge dp_n$$

so VR is a symplectic mapping.

7. Higher Order Methods

High-order methods can be obtained, of course, by using Taylor and Runge-Kutta methods. However, high-order Taylor methods have drawbacks which we have already discussed and high-order Runge-Kutta methods are usually implicit and are therefore more expensive computationally. A cheaper way to obtain higher-order symplectic methods is through composition. There are two methods by which one can form a computation method: first, by splitting the Hamiltonian into two or more subproblems, and second, by using a symmetric second-order symplectic approximation. Another benefit is that these types of higher-order methods preserve certain geometric properties of the problem being analyzed. Before we begin, we need to define what it means for a Hamiltonian system to be separable.

Defintion 11. *A Hamiltonian is **separable** if it can be written as a sum of kinetic and potential energy in the form*

$$H(\boldsymbol{q}, \boldsymbol{p}) = T(\boldsymbol{p}) + V(\boldsymbol{q})$$

The corresponding systems of equations is written

$$\begin{aligned} \frac{d}{dt}\boldsymbol{q} &= \nabla_{\boldsymbol{p}} T(\boldsymbol{p}) \\ \frac{d}{dt}\boldsymbol{p} &= -\nabla_{\boldsymbol{q}} V(\boldsymbol{q}). \end{aligned}$$

Note that this can be written as two Hamiltonian subsystems

$$\frac{d}{dt}\boldsymbol{q} = 0$$
$$\frac{d}{dt}\boldsymbol{p} = -\nabla_{\boldsymbol{q}} V(\boldsymbol{q})$$

and

$$\frac{d}{dt}\boldsymbol{q} = \nabla_{\boldsymbol{p}} T(\boldsymbol{p})$$
$$\frac{d}{dt}\boldsymbol{p} = 0.$$

7.1. **Notation.** The purpose of this section is to clarify the meanin of the notation that will appear in the following sections. We will introduce the notatation for linear systems of differential equations and then extend it to the nonlinear systems to ensure complete understanding.

Given the linear differential equation

$$\frac{d}{dt}\mathbf{z} = \mathbf{A}\mathbf{z}$$

the flow map can be written

$$\Phi_{t,\mathbf{A}}(\mathbf{z}) = e^{t\mathbf{A}}\mathbf{z},$$

where the matrix exponential is defined as

$$e^{t\mathbf{A}} = \mathbf{I}_k + t\mathbf{A} + \frac{t^2}{2}\mathbf{A}^2 + \frac{t^3}{3!}\mathbf{A}^3 + \dots.$$

Note that

$$e^{t\mathbf{A}}e^{t\mathbf{B}} = e^{t(\mathbf{A}+\mathbf{B})}$$

is only true if $\mathbf{A}$ and $\mathbf{B}$ are commutable matrices.

Now consider the nonlinear differential equation

$$\frac{d}{dt}\mathbf{z} = \mathbf{f}(\mathbf{z})$$

if this system is Hamiltonian, where $\mathbf{z} = (\mathbf{q}, \mathbf{p})$, then we can write

$$\dot{\mathbf{z}} = \{\mathbf{z}, H(\mathbf{z})\},$$

where the braces represent the Poisson bracket

$$\{F, G\} = F_{\mathbf{q}}G_{\mathbf{p}} - F_{\mathbf{p}}G_{\mathbf{q}}.$$

We can introduce the differential operator D_G by

$$D_G F := \{F, G\}$$

which implies

$$\dot{\mathbf{z}} = D_H \mathbf{z}.$$

The goal of composition methods is to find coefficients (weightfactors) $\{c_i\}_{i=0,\dots,s}$ and $\{d_i\}_{i=0,\dots,s}$ for a given order p so that

$$exp[\Delta t(A+B)] = \prod_{i=1}^{k} exp(c_i \Delta t A) exp(d_i \Delta t B) + \mathcal{O}(\Delta t^{p+1}) \tag{27}$$

where $A = \nabla_{\mathbf{p}} T(\mathbf{p})$ and $B = \nabla_{\mathbf{q}} V(\mathbf{q})$.

Now if we look at the flow of $\mathbf{z}(t)$ from $t=0$ to $t=\Delta t$, we have

$$\Phi_{t,H} = \mathbf{z}(\Delta t) = [exp(\Delta t D_H)]\mathbf{z}(0).$$

Note that for separable Hamiltonians,

$$D_H = D_T + D_V$$

so

$$\Phi_{t,H} = \mathbf{z}(\Delta t) = [exp(\Delta t(A+B))]\mathbf{z}(0),$$

where $A := D_T$ and $B := D_V$.

Now suppose we have found such $\{c_i\}$ and $\{d_i\}$ for some given p. Then a mapping from $\mathbf{z} = \mathbf{z}(0)$ to $\mathbf{z}' = \mathbf{z}(\Delta t)$ is given by

$$\Phi = (\prod_{i=1}^{k} exp(c_i \Delta t A) exp(d_i \Delta t B))\mathbf{z}.$$

This mapping is a product, i.e. composition, of symplectic mappings and is therefore symplectic. From this map we can define an p^{th} order symplectic integrator

$$\mathbf{q}_i = \mathbf{q}_{i-1} + \Delta t c_i \frac{\partial T}{\partial \mathbf{p}}(\mathbf{p}_{i-1})$$
$$\mathbf{p}_i = \mathbf{p}_{i-1} + \Delta t d_i \frac{\partial V}{\partial \mathbf{q}}(\mathbf{q}_i),$$

where $\mathbf{z} = \mathbf{z}(0) = (\mathbf{q}_0, \mathbf{p}_0)$ and $\mathbf{z}' = \mathbf{z}(\Delta t) = (\mathbf{q}_k, \mathbf{p}_k$.

7.2. **The Coefficients.**

If we expand the left side of equation (27) in powers of Δt, we can compare the coefficients between terms with equal powers of Δt. This ields a nonlinear set of equations for $\{c_i\}$ and $\{d_i\}$. For example, when $p=2$, we get three equations. From the coefficients in front of A, we have

$$c_1 + c_2 + \ldots + c_k = 1.$$

Similarly, from the coefficients in front of B, we have

$$d_1 + d_2 + \ldots + c_k = 1.$$

In addition, we have an equation that comes from the coefficients in front of AB

$$c_1(d_1 + d_2 + \ldots + d_k) + c_2(d_2 + \ldots + d_k) + \ldots + c_k d_k = \frac{1}{2}.$$

In this case, the simplest solution is $k = 2, c_1 = c_2 = \frac{1}{2}, d_1 = 1$.

However, the complexity of finding values for k, $\{c_i\}$, and $\{d_i\}$ increases as n increases. Therefore, we introduce another method by which higher order composition methods can be derived.

7.3. **Composition Methods of Even Order.** First note that (27) is equivalent to

$$S(\Delta t) := \prod_{i=1}^{k} exp(c_i \Delta t A) exp(d_i \Delta t B) = exp[\Delta t(A+B) + \mathcal{O}(\Delta t^{p+1})] \tag{28}$$

Note also that any symmetric operator has even order and time reversibility:

$$S(\Delta t)S(-\Delta t) = S(\Delta t)S(\Delta t) = identity.$$

One advantage to using symmetric methods is the fact that the order conditions simplify due to the fact that the odd power terms in the Taylor expansion of the local error vanish.

Suppose that we have the second order integrator from the brief example in the previous section:

$$S_{2nd}(\Delta t) := exp(\frac{1}{2}\Delta t A) exp(\Delta t B) exp(\frac{1}{2}\Delta t A).$$

Then we can obtain a 4^{th} order symplectic integrator by a symmetric composition

$$S_{4th}(\Delta t) := S_{2nd}(x_1 \Delta t) S_{2nd}(x_0 \Delta t) S_{2nd}(x_1 \Delta t), \tag{29}$$

where x_0 and x_1 are two real numbers that have yet to be determined. In order to continue the analysis of this new integrator, we will need the following formula.

Definition 12. *For any non-commutative operators X and Y, the product of the two exponential functions, $exp(X)$ and $exp(Y)$, can be expressed as a single exponential function:*

$$exp(X)exp(Y) = exp(Z),$$

where

$$\begin{aligned} Z = X + Y + \tfrac{1}{2}[X,Y] + \tfrac{1}{12}([X,X,Y] + [Y,Y,X]) + \tfrac{1}{24}[X,Y,Y,X] - \tfrac{1}{720}([Y,Y,Y,Y,X] \\ + [X,X,X,X,Y]) + \tfrac{1}{360}([Y,X,X,X,Y] + [X,Y,Y,Y,X]) + \tfrac{1}{120}([X,X,Y,Y,X,] \\ + [Y,Y,X,X,Y]) + \ldots . \end{aligned}$$

Here the brackets represent the commutator

$$[X,Y] := XY - YX$$

and higher order commutators

$$[X,X,Y] := [X,[X,Y]].$$

This formula is called the ***Baker-Campbell-Hausdorff (BCH) formula****. Repeated application of the BCH formula yields*

$$exp(X)exp(Y)exp(X) = exp(W),$$

where

$$\begin{aligned} W = 2X + Y + \tfrac{1}{6}[Y,Y,X] - \tfrac{1}{6}[X,X,Y] + \tfrac{7}{360}[X,X,X,X,Y] - \tfrac{1}{360}[Y,Y,Y,Y,X] \\ + \tfrac{1}{90}[X,Y,Y,Y,X] + \tfrac{1}{45}[Y,X,X,X,X,Y] - \tfrac{1}{60}[X,X,Y,Y,X,] + \tfrac{1}{30}[Y,Y,X,X,Y] + \ldots . \end{aligned}$$

This allows us to write

$$S_{2nd}(x_1 \Delta t) = exp(\Delta t x_1 \alpha_1 + \Delta t^3 x_1^3 \alpha_3 + \Delta t^5 x_1^5 \alpha^5 + \ldots)$$

and

$$S_{2nd}(x_0 \Delta t) = exp(\Delta t x_0 \alpha_1 + \Delta t^3 x_0^3 \alpha_3 + \Delta t^5 x_0^5 \alpha^5 + \ldots)$$

where

$$\alpha_1 - A + B, \quad \alpha_3 = \frac{1}{12}[B,B,A] - \frac{1}{24}[A,A,B], \quad \alpha_5 = \frac{7}{5760}[A,A,A,A,B] + \dots.$$

This yields the composition

$$S_{4th}(\Delta t) = exp[\Delta t(x_0 + 2x_1)\alpha_1 + \Delta t^3(x_0^3 + 2x_1^3)\alpha_3 + \Delta t^5(x_0^5 + 2x_1^5)\alpha_5 + \dots].$$

In order for this to be a 4^{th} order integrator, it must be of the form

$$S_{4th}(\Delta t) = exp[\Delta t(A+B) + \mathcal{O}(\Delta t^5)].$$

Therefore, we will need two conditions:

$$x_0 + 2x_1 = 1, \quad and \quad x_0^3 + 2x_1^3 = 0.$$

This yields the unique solution

$$x_0 = \frac{-2^{1/3}}{2-2^{1/3}}, \quad x_1 = \frac{1}{2-2^{1/3}}.$$

By comparing (28) with (29), the coefficients for (28) can be found:

$$d_1 = d_3 = x_1, \quad d_2 = x_0, \quad c_1 = c_4 = \frac{1}{2}x_1, \quad c_2 = c_3 = \frac{1}{2}(x_0 + x_1).$$

This gives us the exact coefficients of a 4^{th} order integrator.

Now we can find a 6^{th} order integrator by the same process, using S_{4th}:

$$S_{6th}(\Delta t) := S_{4th}(y_1\Delta t)S_{4th}(y_0\Delta t)S_{4th}(y_1\Delta t),$$

which can also be written

$$S_{6th} = S_{2nd}(x_1y_1\Delta t)S_{2nd}(x_0y_1\Delta t)S_{2nd}(x_1y_1\Delta t) \times S_{2nd}(x_1y_0\Delta t)S_{2nd}(x_0y_0\Delta t)S_{2nd}(x_1y_0\Delta t) \times S_{2nd}(x_1y_1\Delta t)S_{2nd}(x_0y_1\Delta t)S_{2nd}(x_1y_1\Delta t).$$

Through this process we can see two things

(1) If a symmetric integrator of order $2p$, $S_{2nd}(\Delta t)$, is previously known, we can produce an integrator of order $(2p+2)$ using the composition

$$S_{2p+2}(\Delta t) := S_{2p}(z_1\Delta t)S_{2p}(z_0\Delta t)S_{2p}(z_1\Delta t),$$

where z_0 and z_1 satisfy

$$z_0 = \frac{-2^{1/(2p+1)}}{2-2^{(1/2p+1)}}, \quad z_1 = \frac{1}{2-2^{1/(2p+1)}}.$$

Note that

$$S_{2p+2}(\Delta t) = exp[\Delta t(x_0 + 2x_1) + \mathcal{O}(\Delta t^{2p+2})]$$

because the lower powers of Δt are eliminated when developing the lower-order integrators.

(2) To get a $(2p)^{th}$ order symplectic integrator, we have to compose S_{2nd} with itself 3^{p-1} times. Therefore, the number of steps k is $k = 3^{p-1} + 1$, which obviously grows rapidly as p increases.

8. Conclusion

As numerical techniques developed, certain advantages began to appear with the use of specific schemes. Symplecticity, for example, is highly valued when working with dynamical systems, especially constrained systems. SHAKE preserves the wedge product and its "velocity" verson, RATTLE, is symplectic. These two methods, however, have global error $\mathcal{O}(\Delta t^2)$ since they are derived from globally second-order methods. However, higher order symplectic integrators can be constructed using composition methods. However, it must be noted that the higher the demanded order, the greater the complexity of its derivation. Therefore, it can be seen that accuracy comes at a price. More accurate methods are usually more costly than lower order methods, so the search continues for symplectic methods that have minimal cost and high accuracy.

Acknowledgements

I'd like to thank Mike Holst for taking me on as an Undergraduate Researcher last summer and for guiding me in my research efforts. His interest and support in my academic life has helped me realize not only that I have potential, but also that I can reach it. It is with his support that I have excelled in this my last year of my undergraduate study at the University of California, San Diego and that I will be able to excel next year as I start a Ph.D. program in the applied math program at the University of Texas, Austin.

I'd also like to thank Nick Miller, my research partner and friend, for contributing to my knowledge and for his continuous input, support, and humor.

References

[1] H.C. Anderson. Rattle: a "velocity" version of the Shake algorithm for molecular dynamics calculations. *J. Comput. Phys.*, **52**: 24-34, 1983.

[2] F. Bowman and F.A. Gerard. *Higher Calculus.* London: Cambridge University Press, 1967.

[3] K. Farrell, N. Miller, and M. Holst. Some notes on Hamiltonian mechanics and numerical integrators. Department of Mathematics, University of California, San Diego, 2008.

[4] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Englewood Cliffs, NJ: Prentice-Hall, 1971.

[5] E. Hairer, Ch. Lubich, and G. Wanner. Geometric numerical integration illustrated by the Störmer-Verlet method. *Acta Numerica*, **12**: 399-450, 2003.

[6] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I, Nonstiff Problems.* Berlin Heidelberg: Springer-Verlag, 2nd edition, 1993.

[7] B. Leimkuhler and S. Reich. *Simulating Hamiltonian Dynamics.* New York: Cambridge University Press, 2004.

[8] B. Leimkuhler and R.D. Skeel. Symplectic numerical integrators in constrained Hamiltonian systems. *J. Comput. Phys.*, **112**: 117-125, 1994.

[9] J.P. Ryckaert, G. Ciccotti, and H.J.C. Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *J. Comput. Phys.*, **23**:327-341, 1977.

[10] H. Yoshida. Construction of higher order symplectic integrators. *Phys. Lett. A*, **150**: 262-268, 1990.