Introduction to symplectic integrator methods for solving diverse dynamical equations

Siu A. Chin
Department of Physics and Astronomy
Texas A&M University
College Station, TX, 77840, USA

First Lecture

Introduction, the stability of numerical algorithms, and the four formulations of classical mechanics.

Introduction

An analogy: solving equations and writing calligraphy.

There are two kinds of language in the world: alphabet-based phonetic (English) and non-alphabet-based pictographic (Chinese).

To write well in a phonetic language, such as English, you only need to know the letters of the alphabet, no need to know the meaning of any word.

To write well in a pictographic language, such as Chinese, you must know the meaning of every word. (Otherwise the word “smile” would not smile.)

Most mathematicians think that solving equation numerically is like writing fonts for the letters of an alphabet, basic methods (such as finite-difference) can be applied to all equations.

Physicists (like me) think that some equations are like Chinese, there are further meanings to the equation and you must know them before you can solve it well, like writing calligraphy.

The analogy is not exact, but has some truth. For example, let's consider Newton's equation of motion

$$m\frac{d^2\mathbf{r}}{dt^2} = \mathbf{F}(\mathbf{r}), \qquad (1)$$

where $\mathbf{r} = (x, y, z)$. Let's define

$$\mathbf{v} \equiv \frac{d\mathbf{r}}{dt} \quad \text{and} \quad \mathbf{a}(\mathbf{r}) \equiv \frac{\mathbf{F}(\mathbf{r})}{m},$$

then (1) reads,

$$\frac{d\mathbf{v}}{dt} = \mathbf{a}(\mathbf{r}). \qquad (2)$$

A simple approximation would be

$$\frac{d\mathbf{v}}{dt} \approx \frac{\mathbf{v}(t+\Delta t) - \mathbf{v}(t)}{\Delta t} \quad \text{and} \quad \frac{d\mathbf{r}}{dt} \approx \frac{\mathbf{r}(t+\Delta t) - \mathbf{r}(t)}{\Delta t},$$

When applied to the two equations above, this produces the Euler algorithm

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t\mathbf{a}(\mathbf{r}_n) \quad \text{and} \quad \mathbf{r}_{n+1} = \mathbf{r}_n + \Delta t\mathbf{v}_n$$

where $\mathbf{v}_{n+1} = \mathbf{v}(t + \Delta t)$, $\mathbf{v}_n = \mathbf{v}(t = n\Delta t)$, etc., which can solve any general continuous vector function $\mathbf{a}(\mathbf{r})$. However, Euler is a very poor algorithm and is always unstable no matter how small is Δt. This is because being general, Euler knows nothing about the general function $\mathbf{a}(\mathbf{r})$. But we are not interested in a general $\mathbf{a}(\mathbf{r})$, only certain types of $\mathbf{a}(\mathbf{r})$ are of physical interest.

For example, one is interested mostly in

$$\mathbf{F}(\mathbf{r}) = -\nabla U(\mathbf{r}) \quad \rightarrow \quad \mathbf{a}(\mathbf{r}) = -\frac{1}{m}\nabla U(\mathbf{r})$$

where $U(\mathbf{r})$ is the potential energy function. In this case, the dot product of (2) with $\mathbf{v}$ gives,

$$m\mathbf{v}\cdot\frac{d\mathbf{v}}{dt}=-\nabla U(\mathbf{r})\cdot\mathbf{v}\quad\rightarrow\quad\frac{m}{2}\frac{d(\mathbf{v}\cdot\mathbf{v})}{dt}=-\frac{dU(\mathbf{r})}{dt},$$

and hence

$$\frac{d}{dt}\left(\frac{m}{2}(\mathbf{v}\cdot\mathbf{v})+U(\mathbf{r})\right)=0\quad\rightarrow\quad\frac{m}{2}(\mathbf{v}\cdot\mathbf{v})+U(\mathbf{r})=const=E,$$

a constant of motion, the mechanical energy E of the system.

Furthermore, if the potential function is spherically symmetric, only depends on $r=\sqrt{\mathbf{r}\cdot\mathbf{r}}=\sqrt{x^2+y^2+z^2}$, then the cross product of (2) with $\mathbf{r}$ would give

$$m\mathbf{r}\times\frac{d\mathbf{v}}{dt}=-\mathbf{r}\times\nabla U(r).$$

Since

$$\nabla U(r)=\frac{\partial U}{\partial r}\hat{\mathbf{r}},$$

this then gives

$$\rightarrow \frac{d(\mathbf{r} \times m\mathbf{v})}{dt} = 0 \rightarrow \mathbf{r} \times m\mathbf{v} = const = \mathbf{L}$$

the conservation of angular momentum $\mathbf{L}$.

Since the Euler algorithm knows nothing about the specific form of $\mathbf{F}(\mathbf{r})$, it cannot conserve these two important physical constants of motion and cannot give a good description of the motion.

However, if we just take the Euler algorithm,

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t \mathbf{a}(\mathbf{r}_n) \quad \text{and} \quad \mathbf{r}_{n+1} = \mathbf{r}_n + \Delta t \mathbf{v}_n$$

and just modify it slightly, to become the Cromer algorithm,

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t \mathbf{a}(\mathbf{r}_n) \quad \text{and} \quad \mathbf{r}_{n+1} = \mathbf{r}_n + \Delta t \mathbf{v}_{n+1}$$

then

$$\begin{aligned} \mathbf{r}_{n+1} \times \mathbf{v}_{n+1} &= (\mathbf{r}_n + \Delta t \mathbf{v}_{n+1}) \times \mathbf{v}_{n+1} \\ &= \mathbf{r}_n \times \mathbf{v}_{n+1} = \mathbf{r}_n \times (\mathbf{v}_n + \Delta t \mathbf{a}(\mathbf{r}_n)) \\ &= \mathbf{r}_n \times \mathbf{v}_n! \end{aligned}$$

The Cromer algorithm conserves angular momentum!

(For periodic motion the energy error of the Cromer is periodically bounded, but the energy error of the Euler algorithm grows without bound.)

Why is such a slight change makes such a big difference?

The stability of algorithms

Consider the 1D case for notational simplicity. An algorithm is a mapping of dynamical variables, from the present time $t = n\Delta t$ to the next time $t = (n+1)\Delta t$.

$$x_{n+1} = f(x_n, v_n) \qquad (3)$$

$$v_{n+1} = g(x_n, v_n)$$

Let's see how the errors propagate as we iterate this mapping. Let $\widetilde{x}_n = x(n\Delta t)$, $\widetilde{v}_n = v(n\Delta t)$ be the exact solutions at $t = n\Delta t$, and ϵ_n and δ_n, their respective errors. That is,

$$x_n = \widetilde{x}_n + \epsilon_n$$

$$v_n = \widetilde{v}_n + \delta_n$$

then (3) becomes,

$$\begin{aligned}\widetilde{x}_{n+1} + \epsilon_{n+1} &= f(\widetilde{x}_n + \epsilon_n, \widetilde{v}_n + \delta_n) \\ &= f(\widetilde{x}_n, \widetilde{v}_n) + \epsilon_n \frac{\partial f}{\partial x_n} + \delta_n \frac{\partial f}{\partial v_n}.\end{aligned}$$

If the algorithm is exact, then $f(\widetilde{x}_n, \widetilde{v}_n) = \widetilde{x}_{n+1}$. If the algorithm is not exact, $\widetilde{x}_{n+1} - f(\widetilde{x}_n, \widetilde{v}_n)$ is the systematic, *trancation error* of the algorithm, proportional to some powers of Δt. We ignore this systematic error for now and only concentrate on the propagation errors:

$$\begin{aligned}\epsilon_{n+1} &= \epsilon_n \frac{\partial f}{\partial x_n} + \delta_n \frac{\partial f}{\partial v_n} \\ \delta_{n+1} &= \epsilon_n \frac{\partial g}{\partial x_n} + \delta_n \frac{\partial g}{\partial v_n}\end{aligned}$$

$$\begin{pmatrix} \epsilon_{n+1} \\ \delta_{n+1} \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial x_n} & \frac{\partial f}{\partial v_n} \\ \frac{\partial g}{\partial x_n} & \frac{\partial g}{\partial v_n} \end{pmatrix} \begin{pmatrix} \epsilon_n \\ \delta_n \end{pmatrix} = \begin{pmatrix} \frac{\partial x_{n+1}}{\partial x_n} & \frac{\partial x_{n+1}}{\partial v_n} \\ \frac{\partial v_{n+1}}{\partial x_n} & \frac{\partial v_{n+1}}{\partial v_n} \end{pmatrix} \begin{pmatrix} \epsilon_n \\ \delta_n \end{pmatrix}$$

The errors propagate according to the Jacobian matrix

$$M = \begin{pmatrix} \frac{\partial x_{n+1}}{\partial x_n} & \frac{\partial x_{n+1}}{\partial v_n} \\ \frac{\partial v_{n+1}}{\partial x_n} & \frac{\partial v_{n+1}}{\partial v_n} \end{pmatrix}$$

of the transformation

$$(x_n, v_n) \rightarrow (x_{n+1}, v_{n+1})!$$

In particular, the "volume" element transforms as

$$\det M dx_n dv_n = dx_{n+1} dv_{n+1}$$

Familiar example: $(r, \theta) \to (x, y)$

$$x = r\cos\theta, \qquad y = r\sin\theta$$

$$M = \begin{pmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{pmatrix},$$

Therefore,

$$\det M dr d\theta = r dr d\theta = dx dy,$$

which is the correct transformation of the area in this case.

Consider the case of a simple harmonic oscillator with $a(x) = -(k/m)x = -\omega^2 x$, we will see later that M is just a constant 2×2 matrix, and

$$\binom{\epsilon_{n+1}}{\delta_{n+1}} = M^n \binom{\epsilon_1}{\delta_1}.$$

Let M have eigenvalues (generally complex) λ_1, λ_2 and corresponding eigenvectors $(\epsilon_1^*, \delta_1^*)$ and $(\epsilon_2^*, \delta_2^*)$ such that

$$M \binom{\epsilon_{1,2}^*}{\delta_{1,2}^*} = \lambda_{1,2} \binom{\epsilon_{1,2}^*}{\delta_{1,2}^*}.$$

We can then expand

$$\begin{pmatrix} \epsilon_1 \\ \delta_1 \end{pmatrix} = c_1 \begin{pmatrix} \epsilon_1^* \\ \delta_1^* \end{pmatrix} + c_2 \begin{pmatrix} \epsilon_2^* \\ \delta_2^* \end{pmatrix}.$$

then

$$\begin{pmatrix} \epsilon_{n+1} \\ \delta_{n+1} \end{pmatrix} = M^n \left[c_1 \begin{pmatrix} \epsilon_1^* \\ \delta_1^* \end{pmatrix} + c_2 \begin{pmatrix} \epsilon_2^* \\ \delta_2^* \end{pmatrix} \right]$$

$$= \left[c_1 \lambda_1^n \begin{pmatrix} \epsilon_1^* \\ \delta_1^* \end{pmatrix} + c_2 \lambda_2^n \begin{pmatrix} \epsilon_2^* \\ \delta_2^* \end{pmatrix} \right]$$

The the propagation error will grow exponentially with the modulus of the eigenvalues. Unless the modulus of each eigenvalue is less (can't be) or equal to unity, the algorithm will be unstable.

Let's now find out the condition for which the eigenvalues have unit modulus.

For a 2×2 matrix, the condition for determining the eigenvalues is given by

$$M - \lambda I = \det \begin{pmatrix} a - \lambda & b \\ c & d - \lambda \end{pmatrix} = 0,$$

$$(a - \lambda)(d - \lambda) - cb = \lambda^2 - (a + d)\lambda + (ad - cb) = 0,$$

$$\lambda^2 - T\lambda + D = 0 \quad \rightarrow \quad \lambda_{1,2} = \frac{T}{2} \pm \sqrt{\left(\frac{T}{2}\right)^2 - D} \qquad (4)$$

where T and D are the trace and determinant of M respectively. Note that

$$T = \lambda_1 + \lambda_2 \quad \text{and} \quad D = \lambda_1 \lambda_2.$$

From (4), if the eigenvalues are complex, then they must be complex conjugate of one another, *i.e.*, $\lambda_2 = \lambda_1^*$, with the same modulus $|\lambda_{1,2}| = \sqrt{\lambda_1 \lambda_2}$.

But since $\lambda_1\lambda_2 = \det M$, the modulus will be unity and the algorithm stable if 1) $\det M = 1$ and 2) the eigenvalues are complex.

When $\det M = 1$, (4) can be rewritten as

$$\lambda_{1,2} = \frac{T}{2} \pm i\sqrt{1 - \left(\frac{T}{2}\right)^2}$$

If $|T/2| \leq 1$, can define $\cos\phi = T/2$, then,

$$\lambda_{1,2} = \cos\phi \pm i\sin\phi = \mathrm{e}^{\pm i\phi}.$$

Thus the eigenvalues are indeed complex and unit modulus when $\det M = 1$ and with Δt sufficiently small so that $|T/2| \leq 1$.

Let's examine the Euler and the Cromer algorithm in light of this understanding. Recall that for the simple harmonic oscillator, $a(x) = -\omega^2 x$.

For Euler:

$$x_{n+1} = x_n + \Delta t v_n \qquad v_{n+1} = v_n - \Delta t \omega^2 x_n$$

$$M = \begin{pmatrix} \frac{\partial x_{n+1}}{\partial x_n} & \frac{\partial x_{n+1}}{\partial v_n} \\ \frac{\partial v_{n+1}}{\partial x_n} & \frac{\partial v_{n+1}}{\partial v_n} \end{pmatrix} = \begin{pmatrix} 1 & \Delta t \\ -\Delta t \omega^2 & 1 \end{pmatrix},$$

and therefore

$$|\lambda_{1,2}| = \sqrt{\det M} = \sqrt{1 + \Delta t^2 \omega^2} > 1,$$

always unstable. (Note that we only need to know whether $\det M$ is equal or greater than one. We don't need to evaluate the individual eigenvalues.)

For Cromer:

$$x_{n+1} = x_n + \Delta t v_{n+1} \qquad v_{n+1} = v_n - \Delta t \omega^2 x_n$$

$$M = \begin{pmatrix} \frac{\partial x_{n+1}}{\partial x_n} & \frac{\partial x_{n+1}}{\partial v_n} \\ \frac{\partial v_{n+1}}{\partial x_n} & \frac{\partial v_{n+1}}{\partial v_n} \end{pmatrix} = \begin{pmatrix} 1 - \Delta t^2 \omega^2 & \Delta t \\ -\Delta t \omega^2 & 1 \end{pmatrix},$$

and therefore

$$|\lambda_{1,2}| = \sqrt{\det M} = 1,$$

and will be stable if

$$|T/2| \leq 1 \ \rightarrow \ 1 - \frac{1}{2}\omega^2 \Delta t^2 \geq -1 \ \rightarrow \Delta t \leq \frac{2}{\omega}.$$

How can one derive algorithms with $\det M = 1$?

Hint: $\det M = 1$ means that $dx_n dv_n = dx_{n+1} dv_{n+1}$. This means that the phase-space formed by the dynamical variables x and $v = p/m$ is unaltered, not expanded nor contracted by the algorithm. This is an exact property of Hamiltonian mechanics, known as Liouville's Theorem.

Thus numerical stability $\leftrightarrow$ phase-space preservation.

Therefore, to devise the best algorithm of solving any equation, you must know more deeply, the underlying meaning of your equation. (Of course, not all equations are so menaingful.)

Classical mechanics

The **Four** formulations of mechanics:

1. **The Newtonian formulation:** Primarily just a second order differential equation as we have been using:
$$m\frac{d^2\mathbf{r}}{dt^2} = \mathbf{F}(\mathbf{r}).$$
For comparison with other formulations, we should express this in terms of the *momentum*
$$\mathbf{p} = m\frac{d\mathbf{r}}{dt}, \tag{5}$$
and the potential energy function $U(\mathbf{r})$
$$\mathbf{F}(\mathbf{r}) = -\nabla U(\mathbf{r}),$$
as a pair of equation (5) and below
$$\frac{d\mathbf{p}}{dt} = -\nabla U(\mathbf{r}). \tag{6}$$

2. **The Lagrangian formulation:** Generalize from Cartesian coordinates $r_i = (x, y, z)$, to *generalized coordinates* q_i such as *angles*, to satisfy constraints cumbersome to do so in terms of x, y, z. Corresponding to q_i are *generalize momentum*

$$p_i = \frac{\partial L}{\partial \dot{q}_i}$$

where $\dot{q}_i = dq/dt$ and where the *Lagrangian function* is given by

$$L = \frac{1}{2} m \sum_i \dot{q}_i^2 - U(q_i).$$

The equation of motion is given by

$$\frac{\partial p_i}{\partial t} = \frac{\partial L}{\partial q_i}.$$

First we check that if q_i are just Cartesian coordinates, then we recover Newton's equation of motion. In this case, $q_i = r_i$, $\dot{q}_i = v_i$, then

$$L = \frac{1}{2} m \sum_i v_i^2 - U(r_i).$$

and

$$p_i = \frac{\partial L}{\partial \dot{q}_i} = m v_i.$$

The equation of motion is then

$$\frac{\partial p_i}{\partial t} = \frac{\partial L}{\partial q_i} = -\frac{\partial U}{\partial r_i}.$$

Since we will not use the Lagrangian formulation much in this series of lectures, we will skip examples with constraints.

3. **The Hamiltonian formulation:** The most important key point. The generalized momenta p_i are to be regarded as *independent* fundamental dynamical degrees of freedom in equal footing as q_i. That is, p_i are NOT to be regarded as depending on $\dot{q}_i$, but rather, $\dot{q}_i$ are to be regarded as depending on p_i.

   The generalize momenta p_i are defined via L as before, but after that we eliminate all $\dot{q}_i$ by p_i in defining the Hamiltonian function

$$H(p_i, q_i) = \sum_i p_i \dot{q}_i - L(q_i, \dot{q}_i) = \frac{1}{2m} \sum_i p_i^2 + U(q_i).$$

   The dynamics is then given by the pair Hamilton's eqaution of motion:

$$\dot{q}_i = \frac{\partial H}{\partial p_i} \quad \text{and} \quad \dot{p}_i = -\frac{\partial H}{\partial q_i}.$$

Again, we check that it reduces back to Newton's equation for Cartesian coordinates. The Hamiltonian function is given by

$$H = \frac{1}{2m}\sum_i p_i^2 + U(r_i).$$

The equations of motion are:

$$\dot{r}_i = \frac{\partial H}{\partial p_i} = \frac{p_i}{m}$$

This equation is to be interpreted as the time-evolution of r_i is dependent on the fundamental degree-of-freedom p_i. This is NOT a definition of p_i as in the Newtonian formulation. The equation is the same, but the interpretation is entirely different.

Newton's second law remains the same:

$$\dot{p}_i = -\frac{\partial H}{\partial r_i} = -\frac{\partial U}{\partial r_i}$$

The main advantage of the Hamiltonian formulation is this: Just as some problems are easier to solve by use of spherical coordinates rather than Cartesian coordinates, some dynamical problems are easier to solve by transformations that completely mix up the n generalize momenta p_i *and* the n generalize coordinates q_i. In order for the transformed variables to solve the same dynamical problem, they must obey Hamilton's equation with respect to the Hamiltonian of the transformed variables. Transformations that can do this are called *canonical*. Canonical transformations are the basis for developing the best numerical algorithm for solving classical dyanamics.

4. **Poissonian:** Dynamics is expressed via *Poisson brackets*, which are the same for all canonically transformed variables $\rightarrow$ dynamics is coordinate invariant. For dynamical variable $W(q_i, p_i)$,

$$\begin{aligned}
\frac{\partial W}{\partial t} &= \frac{\partial W}{\partial q_i}\frac{\partial q_i}{\partial t} + \frac{\partial W}{\partial p_i}\frac{\partial p_i}{\partial t} \\
&= \frac{\partial W}{\partial q_i}\frac{\partial H}{\partial p_i} - \frac{\partial W}{\partial p_i}\frac{\partial H}{\partial q_i} \equiv \{W, H\} \\
&= \Big(\frac{\partial H}{\partial p_i}\frac{\partial}{\partial q_i} - \frac{\partial H}{\partial q_i}\frac{\partial}{\partial p_i}\Big)W
\end{aligned}$$

which is of the form $\frac{\partial w}{\partial t} \equiv (T + V)w$. We will discuss this extensively in later lectures.