Physics
Numerical Simulation

Finite Difference Methods

Damian R. Sowinski
DRSowinski@gmail.com

Outline

In this primer you will learn how to solve differential equations on a computer using the finite difference method. Why would you want to know how to do this? Almost every differential equation cannot be solved analytically, so if you want to understand the behavior of the vast majority of systems described by differential equations, you're out of luck if all you have is pencil and paper. That's not to say that you can't still learn a lot from a differential equation through analytical means, but it does mean that if you want a fuller, more complete understanding of the solutions contained within them, then you must supplement analysis with computation. There's plenty of precedent where some numerical solutions were a complete, and delightful, surprise to physics, opening up new lines of research — oscillons, pulsons, criticality in gravitational collapse, just to name a few. Let's get started!

Ordinary Differential Equations

An Exploratory Example and the Big Picture

Here we're going to solve a simple ordinary differential equation, explaining how we're doing it, but without generality in mind. Take the concrete example in; we'll discuss the general theory later. Consider then, the differential equation describing the 1-d motion of a mass m experiencing dampening,

$$m\frac{dv}{dt} = -bv \qquad \text{with} \qquad v(t_0) = v_0 \tag{1}$$

where b is the dampening constant. What units does b have? We could solve this analytically using integration by parts, finding

$$v(t) = v_0 e^{-\frac{t-t_0}{\tau}}, \tag{2}$$

(where $\tau = m/b$ is the dampening time constant) so we'll be able to check our numerical result using this. Of course, most equations won't have known analytical solutions. :(

When working with numerical solutions, units are kinda tough — it's easy to store numbers, but how do you store a unit? Rather than tackling that problem, numerical physicists dimensionalize equations before solving them. Since Eq. 1 provides us with a timescale, τ, and a spatial scale $v_0\tau$, we define the dimensionless space and time variables

$$t = \tau\tilde{t} \tag{3}$$

$$x = v_0\tau\tilde{x}. \tag{4}$$

This just means we measure time intervals in τ chunks, and spatial lengths in $v_0\tau$ chonks. It also means the dimensionless velocity is $v(\tau) = v_0\tilde{v}(\tilde{t})$. The dimensionless form of Eq. 1 becomes

$$\frac{d\tilde{v}}{d\tilde{t}} = -\tilde{v} \qquad \text{with} \qquad \tilde{v}(\tilde{t}_0) = 1. \tag{5}$$

Much simpler, eh, and no free parameters! Of course, writing those tildes all the time is a pain in the butt, so we drop them and remember we have to reintroduce τ and v_0 at the end.

A computer has finite memory, so it won't be able to store every possible value of $v(t)$. Knowing this, we choose a finite time difference, Δt, and define $t_n \equiv n\Delta t$ and consider a finite number of times $t_0, t_1, t_2, \ldots, t_N$. Our goal will be to compute $v_n \equiv v(t_n)$ for $n = 1, 2, \ldots, N$, given v_0. To do this we convert the ODE into a finite difference equation (FDE) by replacing the derivative

$$\frac{dv}{dt} = -v \quad \mapsto \quad \frac{v_{n+1} - v_n}{\Delta t} = -v_n. \tag{6}$$

The FDE is *consistent* with the ODE if is the limit $\Delta t \to 0$ the former becomes the latter. Eq.6 is an algebraic equation, which must be manipulated so that all the terms at time t_{n+1} are on the left, and t_n on the right, giving us a recursion relation

$$v_{n+1} = (1 - \Delta t)v_n. \tag{7}$$

We apply the recursion relation n times to $v_0 = 1$ to get the value v_n,

$$v_n = (1 - \Delta t)^n, \tag{8}$$

and we're done.

Let's check our result versus the continuum limit. Take the finite time interval $[t_i, t]$ and subdivide it into N parts, $t - t_i = N\Delta t$, and call $t_n = t_i + n\Delta t$ so that $t = t_N$ We take the $\Delta t \to 0$ limit of Eq. 8:

$$\begin{aligned} v(t) &= \lim_{\Delta t \to 0} (1 - \Delta t)^N \\ &= \lim_{N \to \infty} (1 - \frac{t_f - t_0}{N})^N \\ &= e^{-(t - t_0)}, \end{aligned} \tag{9}$$

where in the last line we used the limit definition of Euler's constant. Remember these are all actually dimensionless quanitites, so let's put those tildes back. Upon restoring units, $\tilde{v}(\tilde{t}) = v(t)/v_0$ and $\tilde{t} = t/\tau$ we see that the finite difference solution Eq. 9 is identical to the analytical solution Eq. 2 in the continuum limit. We say that the FD solution *converges* to the analytical solution.

Next, let's see how errors propagate in our FD solution. Consider our FD solution at step n and introduce an error term, $v_n \to v_n + \varepsilon_n$. Why would we do this? Well, remember how computers have finite memory? Most real numbers are impossible to store in memory, so they're truncated. We call this round-off error, and that's where the error ε_n comes from. Numerical work is A LOT of keeping track of where errors are coming from because, at the end of the day, you want to see solutions to equations which are impossible to solve by hand. Ok, so now we have

$$\begin{aligned} \varepsilon_{n+1} &= (1 - \Delta t)(v_n + \varepsilon_n) - v_{n+1} \\ &= (1 - \Delta t)\varepsilon_n \\ &= g\varepsilon_n. \end{aligned} \tag{10}$$

We call $g = \varepsilon_{n+1}/\varepsilon_n$ the amplification factor; $g = 1 - \Delta t < 1$, so the errors decay and the FD scheme is *stable*.

The key take away from this section is the triplet: *consistency, convergence*, and *stability*. Take a moment and try to define them to yourself. Convergence is what we are striving for: that our numerical solutions converge to analytical ones. Unfortunately, as stated before, we won't have analytical solutions for most problems, so how will we be able to check convergence? As it so happens, that's where consistency and stability come into play. Convergence implies consistency and stability, but that's not all! The converse is also true: Consistency and stability imply convergence. Building consistent FDEs out of differential equations that give stable solutions is what we aspire to.

A Harder Example

Now that you've been introduced to convergence, consistency, and stability, let's look at them for a slightly more complicated problem. We consider the same mass as above, but attach it to a spring with spring constant k, so that the equation of motion is our favorite damped simple harmonic oscillator

$$m\frac{d^2x}{dt^2} = -b\frac{dx}{dt} - kx \qquad \text{with} \qquad \begin{aligned} x(0) &= x_0 \\ v(0) &= v_0 \end{aligned} \tag{11}$$

It's a second order differential equation! Before you start trying to dimensionalize and figure out what the FD is for second derivatives, STOP. We detest temporal derivatives higher than first order. Spatial derivatives, which we'll encounter in PDEs are fine, but this is time we're talking about. We ALWAYS convert our higher order ODEs into several first order ODEs, so our Equations of motion in this case become

$$\frac{dx}{dt} = v \tag{12}$$

$$\frac{dv}{dt} = -\frac{1}{\tau}v - \omega^2 x, \tag{13}$$

where we've brought back our old friends the dampening time, τ, and spring natural frequency, $\omega = \sqrt{k/m}$. Now we dimensionalize!

This problem gives us two timescales, τ and ω^{-1}. We choose τ as our timescale, so there is a dimensionless parameter $c_1 = \omega\tau$ floating around. We also have two spatial scales, x_0 and $v_0\tau$. We choose x_0 as our spatial scale, which means there will be a second free parameter $c_2 = v_0\tau/x_0$ floating around as well. Dimensionalizing space and time and then dropping all those tildes, our equations are

$$\frac{dx}{dt} = v \qquad\qquad x(0) = 1 \tag{14}$$

$$\frac{dv}{dt} = -v - c_1^2 x \qquad\qquad v(0) = c_2. \tag{15}$$

Can you write down the dimensionless equations for different choices of the dimensionless space-time variables? This is one of those things that's more of an art, and the choices you make will give the equations of motion different aesthetics. As with all art you will get better at it as you do it more often and develop your own aesthetic for where the constants should be sitting around. Time to discretize!

We use the definition of the derivative to derive consistent FDEs, and then once again sovle for the recursion relations, keeping future quantities, at t_{n+1}, on the LHS, and present quantities, at t_n, on the RHS:

$$\frac{x_{n+1}-x_n}{\Delta t} = v_n \qquad \Rightarrow \qquad x_{n+1} = x_n + \Delta t\ v_n \tag{16}$$

$$\frac{v_{n+1}-v_n}{\Delta t} = -v_n - c_1^2 x_n \qquad \Rightarrow \qquad v_{n+1} = -c_1^2 \Delta t x_n + (1-\Delta t) v_n \tag{17}$$

The recursion relations form a set of linear equations. If we define the column vector $\boldsymbol{u}_n = [x_n, v_n]^T$, then we have a vector recursion relation

$$\boldsymbol{u}_{n+1} = \boldsymbol{A}\cdot\boldsymbol{u}_n \qquad \text{where} \quad \boldsymbol{A} = \begin{bmatrix} 1 & \Delta t \\ -c_1^2\Delta t & 1-\Delta t \end{bmatrix} \text{ and } \boldsymbol{u}_0 = \begin{bmatrix} 1 \\ c_2 \end{bmatrix} \tag{18}$$

The solution to the FDE is just as before, but now using matrices,

$$\boldsymbol{u}_n = \boldsymbol{A}^n \cdot \boldsymbol{u}_0 \tag{19}$$

Voila! It's no coincidence that we got a solution that looks similar to the last example: the equations are both linear. Remember how we said we detest higher order temporal derivatives? With linear first order systems, you'll always find an equivalent form for the solution.

Notice how c_1 is in $\boldsymbol{A}$ and c_2 is in the initial condition. This is what I meant by aesthetic earlier. The dimensionalization I chose means that I will have to vary both $\boldsymbol{A}$ and $\boldsymbol{u}_0$ if I want to explore the behavior of the solutions. Another dimensionalization choice would have lead to both parameters living inside of $\boldsymbol{A}$. In that case I would have to only vary $\boldsymbol{A}$, but in two different ways, to explore the behavior of solutions. *Which one is better?* is a silly question to ask, both require the same amount of work. *Which one is more elegant?* is the right question. It doesn't have a right or wrong answer, but depends on whose answering it.

Ok, let's tackle stability. We assume our results have rounding error, $\boldsymbol{u}_n + \boldsymbol{\varepsilon}_n$, and find as before $\boldsymbol{\varepsilon}_{n+1} = \boldsymbol{A}\cdot\boldsymbol{\varepsilon}_n$. More similarity, once again a consequence of having a linear first order (in time) system of ODEs. To figure out stability, we need to look at the this equation in a coordinate system that diagonalizes $\boldsymbol{A}$. In that system, the error recursion relations decouple, and we have $\varepsilon_{i,n+1} = g_i \varepsilon_{i,n}$ for $i = 1, 2$: two simple scalar equations. g_1, g_2 are the eigenvalues of $\boldsymbol{A}$, and if they both satisfy $|g_i| < 1$ then we have stability! See if you can diagonalize the system and find these equations. The eigenvalues turn out to be $1 - \frac{\Delta t}{2}(1 \pm \sqrt{1-4c_1^2})$. Uh oh, do you see a problem? When $c_1 \leq 1/2$ both eigenvalues are less than 1, so the FDS is stable there. When $c_1 > 1/2$, the discriminant is negative and the eigenvalues get an imaginary component, $(1-\frac{\Delta t}{2}) \pm i\frac{\Delta t}{2}\sqrt{4c_1^2-1}$. Can show that as long as $c_1 \leq 1$, the FDS is stable?

What if physically $\omega^2\tau^2 = kb^2/m^3 > 1$? It appears we can't solve this simple differential equation numerically because it is unstable. Instability came from the structure of our FDE, so we shouldn't give up hope yet! Can we find ANOTHER FDE that is consistent with our ODE, one that is stable for all $\omega\tau$? Consider the new FDE

$$\frac{x_{n+1}-x_n}{\Delta t} = v_{n+1} \tag{20}$$

$$\frac{v_{n+1}-v_n}{\Delta t} = -v_{n+1} - c_1^2 x_{n+1} \tag{21}$$

This FDE is also consistent with the ODE; just take the limit $\Delta t \to 0$ to see that. Don't move on until you're convinced of this.

Now let's rearrange these equations to find the recurrence relations. Prove to yourself that the system is

$$\begin{bmatrix} 1 & -\Delta t \\ c_1^2\Delta t & 1+\Delta t \end{bmatrix} \cdot \begin{bmatrix} x_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ v_n \end{bmatrix} \tag{22}$$

This FDE has produced a recurrence relation of the form $\boldsymbol{B}\cdot\boldsymbol{u}_{n+1} = \boldsymbol{u}_n$, so we must invert the matrix $\boldsymbol{B}$ to find $\boldsymbol{A}$ as before. Write down the $\boldsymbol{A}$ for this recurrence relation, and diagonalize it. Under what conditions are the solutions stable? You should convince yourself that the solutions are unconditionally stable. The lesson you should take out of this is that you won't always get the discretization of an ODE right on the first try. Even though it is consistent, it may not be stable. Your job is to ensure that BOTH consistency and stability happen.

Consistency TODO

To make consistent FDEs out of ODEs we need to go back to the basics. One of a physicists favorite tools is a Taylor series, and that's especially true for us. We use Taylor series to discover discretization schemes for our differential equations, giving us FDEs, and also to estimate the resulting error. The basic idea is simple - replace functions and their derivatives with values on our computational grid:

$$\left.\frac{d^m f}{dx^m}\right|_{x_n} = \sum_k c_k^{(m)} f(x_{n+k}) + \varepsilon \tag{23}$$

The error, ε, will depend on Δx to some power. In the continuum limit $\Delta x \to 0$, the error vanishes, $\varepsilon \to 0$; the higher the power, the faster the error will vanish. The hard part is figuring out what the c_n are. In this section, we do just that!

Let's start with a simple example, $f'(x) = df/dx$, the first derivative. We could discretize it in a *forward scheme* using the points x_n and x_{n+1},

$$f'(x_n) = \frac{f(x_{n+1}) - f(x_n)}{\Delta x} + \varepsilon_n \tag{24}$$

What is the error associated with this scheme? For that we Taylor expand,

$$\begin{aligned} \frac{f(x_{n+1}) - f(x_n)}{\Delta x} &= \frac{1}{\Delta x}(f(x_m + \Delta x) - f(x_N)) \\ &= \frac{1}{\Delta x}\left(f(x_m) + f'(x_n)\Delta x + \frac{1}{2}f''(x_n)\Delta x^2 + \cdots - f(x_N)\right) \\ &= f'(x_n) + \frac{f''(x_n)}{2}\Delta x + \mathcal{O}(\Delta x^2) \end{aligned} \tag{25}$$

from which we see that $|\epsilon| = |f''(x_n)\Delta x/2| \sim \mathcal{O}(\Delta x^1)$. We say that the order of the error is 1. Similarly, we could discretize using a *backward scheme*,

$$f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{\Delta x} + \varepsilon_n \tag{26}$$

Show that the error in this scheme is the same as teh forward scheme. Thirdly, we could discretize using a *center scheme*,

$$f'(x_n) = \frac{f(x_{n+1}) - f(x_{n-1})}{2\Delta x} + \varepsilon_n \tag{27}$$

Show that for this scheme $\varepsilon \sim \mathcal{O}(\Delta x^2)$. All these schemes are consistent with the first derivative, but the center scheme has the lowest error.

Now let's examine this process more generally. Consider a computational grid x_n, which, for the time being consists of an infinite number of points. We start with some well-behaved function, f, at the point $x = x_n$, and Taylor expand it at all the points on the grid,

$$\begin{aligned} f_{n+j} &= f(x + j\Delta x) \\ &= \sum_{m=0}^{\infty} \frac{1}{m!} \left.\frac{d^m f}{dx^m}\right|_{x_n} j^m \Delta x^m \\ &= \sum_{m=0}^{\infty} \frac{1}{m!} j^m \Delta x^m \sum_{k=-\infty}^{\infty} c_k^{(m)} f_{n+k} \\ &\Downarrow \end{aligned}$$

$$\sum_{m=0}^{\infty} \frac{j^m \Delta x^m}{m!} c_k^{(m)} = \delta_{j,k} \tag{28}$$

So, at least formally, the coefficients of a FD scheme

Stability