410 Tutorial 8: Additional Information

October 2017

In the previous tutorial, I indicated that I would give you some additional information about how to decide whether the solution you are receiving from your numeric solver is actually providing you with a solution to the desired problem, or if it is just giving you utter nonsense. Unfortunately, I don't really have time to go over this in lab (a single hour is not nearly enough), so I decided to type up a few useful guidelines.

1 Type of Integrator

When solving an initial value problem such as,

$$y'' = -y \tag{1}$$

the first thing one typically does is to choose which integrator we will use. For example, you might use one of RK2, RK4 RK45, Crank Nicholson, Implicit Euler, or a Symplectic integrator. Each of these has different advantages and disadvantages and there are no set rules for determining which is best for a given problem. However, these are a few guidelines which will hopefully make your lives easier. Please note that this list is by no means definitive or particularly mathematically robust. Mostly the following is just my observations from working with these methods for a decade or so.

1.1 Explicit Runge-Kutta Methods

Prototype: RK4,

$$y_{i+1} = y_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) + \mathcal{O}(h^5) \tag{2}$$

where,

$$k1 = f(t_i, y_i) \tag{3}$$

$$k2 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \tag{4}$$

$$k3 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2) \tag{5}$$

$$k4 = f(t_i + h, y_i + hk_3) \tag{6}$$

Advantages:

1. Simple to implement
2. Fast and efficient
3. Easy to implement varying step-size methods such as RK45 to add and subtract resolution when needed

Disadvantages:

1. Doesn't preserve conserved quantities very well (errors in energy and momentum can grow quite quickly, but usually only linearly)
2. Has difficulties with stiff equations and shocks; varying step-size methods help but don't always make things better

Use Case:

1. Runge-Kutta methods should be your go-to for a first attempt at solving a problem. Easy to implement and check, RK4 or Rk45 will probably work adequately on "most" problems
2. Great for short integration times
3. Great for non-stiff equations

1.2 Implicit Runge-Kutta Methods, Crank Nicholson

Prototype: Implicit Euler,

$$y_{i+1} = y_i + hf\left(t + h, y_{i+1}\right) + \mathcal{O}(h^2) \tag{7}$$

Prototype: Crank Nicholson,

$$y_{i+1} = y_i + \frac{h}{2}f\left(t + h, y_{i+1}\right) + \frac{h}{2}f\left(t, y_i\right) + \mathcal{O}(h^3) \tag{8}$$

Advantages:

1. Great for stiff equations
2. Very stable
3. Great for implementing solutions to PDEs (tend to be less susceptible to high frequency noise)

Disadvantages:

1. Each time-step requires solving matrix equations
2. Higher order implicit methods are costly to evaluate
3. Tend to blur out shocks, but at least you don't typically get as bad Gibbs phenomena as with RK methods

Use Case:

1. Crank-Nicholson should be your go-to method for solving wave-type equations in multiple dimensions
2. Great for stiff equations when you don't really care about shock thickness
3. Great for solving parabolic PDEs (you can take much larger time-steps with this type of method). For heat-type equations, explicit methods require $\Delta t < \Delta x^2$ for stability whereas Crank-Nicholson and many other implicit methods are unconditionally stable

1.3 Symplectic Integrators

Prototype: Verlet Method,

Assume a Hamiltonian and equations of motion of the form,

$$H(p,q) = T(p) + V(q) \tag{9}$$

$$\frac{\partial p}{\partial t} = -\frac{\partial H}{\partial q} = f(q) \tag{10}$$

$$\frac{\partial q}{\partial t} = \frac{\partial H}{\partial p} = g(p) \tag{11}$$

$$\tag{12}$$

then the Verlet method is:

$$p_{i+1/2} = p_i + \frac{h}{2} f(q_i) \tag{13}$$

$$q_{i+1} = q_i + hg(p_{i+1/2}) \tag{14}$$

$$p_{i+1} = p_i + \frac{h}{2} f(q_{i+1}) \tag{15}$$

Advantages:

1. Symplectic integrators possess, as a conserved quantity, a Hamiltonian which is slightly perturbed from the original one.

2. This means they can be used for long term integration of systems with preserved energy and momenta and do not suffer from non-conservation like other methods.

3. Easy to implement

Disadvantages:

1. Only really useful for systems derived from a Hamiltonian written in its canonical form

2. Generally less useful in PDEs since when you add in numerical dissipation (which is often necessary for stability) it destroys the constraint preserving properties of the method.

Use Case:

1. Long term evolution of interacting particle systems.

2. Certain wave-like PDEs

2 Convergence testing

Once we have a solution, it is important to check that we have actually solved the correct problem rather than come up with some unrelated numerical artifact. I have illustrated this in Figure 1 for some fictional scenario; the blue curve has smaller tolerance (or h, or other fit parameter) than the green curve but if we decrease the tolerance again we should get something closer to the exact solution. If we keep decreasing the tolerance and plotting the results eventually the curves should be indistinguishable (then its probably a good idea to look at the error as a function of time to work out how far apart the curves actually are).

More precisely, we can exploit the convergence properties of our methods. After performing an integration with an $\mathcal{O}(h^n)$ method, we expect our solution to be of the form:

$$y_{num}(t) = y_{true}(t) + h^n \epsilon(t) \tag{16}$$

where $\epsilon(t)$ is an error function reflecting the truncation error of our method. We can verify that we have converged to the true answer by computing various solutions using different time-steps:

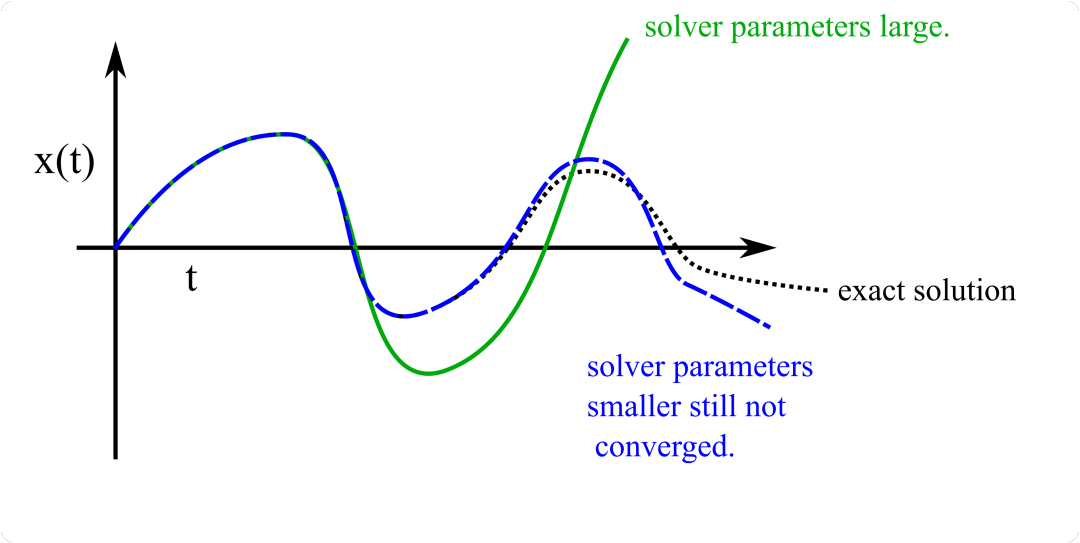

Figure 1: Example of "eyeballing" numerical convergence

$$y_h(t) = y(t) + h^n \epsilon(t) \tag{17}$$

$$y_{h/2}(t) = y(t) + \left(\frac{h}{2}\right)^n \epsilon(t) \tag{18}$$

$$y_{h/4}(t) = y(t) + \left(\frac{h}{4}\right)^n \epsilon(t) \tag{19}$$

then, taking the difference between two successive evaluations,

$$\frac{y_h(t) - y_{h/2}(t)}{y_{h/2}(t) - y_{h/4}(t)} = \frac{h^n \epsilon(t) - \left(\frac{h}{2}\right)^n \epsilon(t)}{\left(\frac{h}{2}\right)^n \epsilon(t) - \left(\frac{h}{4}\right)^n \epsilon(t)} \tag{20}$$

$$= 2^n \tag{21}$$

So one can easily determine if you have converged to the correct solution by verifying that the ratio of successive errors obeys the expected relationship.

3 Independent Residual Evaluators

The above method will tell you whether you are converging to a solution, but it will not give you any information on whether or not you are converging to the correct solution. There are many things that can go wrong when writing a program to solve a given problem, and it is a very real possibility that either

the implementation of the solver or the discretization of the equation or even the derivation of the equation itself is incorrect. We can attempt to protect ourselves against these eventualities through the judicial use of independent residual evaluators.

The basic idea is as follows. Using a completely separate discretization of the differential system in question, derive a new operator for the evolution of the system. After solving the system, apply this new operator to the solution and verify that the operator residuals vanish in the same manner as demonstrated in the Convergence Testing Section.

For example, if you were solving the wave equation in 1D,

$$\frac{\partial y}{\partial t} = v \tag{22}$$

$$\frac{\partial v}{\partial t} = -\frac{\partial^2 y}{\partial x^2} \tag{23}$$

one might use a Crank Nicholson discretization:

$$y_i^{n+1} = y_i^n + \frac{h}{2} v_i^n + \frac{h}{2} v_i^{n+1} + \mathcal{O}(h^3) \tag{24}$$

$$v_i^{n+1} = v_i^n + \frac{h}{2} \frac{y_{i-1}^n - 2y_i^n + y_{i+1}^n}{\Delta^2} + \frac{h}{2} \frac{y_{i-1}^{n+1} - 2y_i^{n+1} + y_{i+1}^{n+1}}{\Delta^2} + \mathcal{O}(h^3) \tag{25}$$

You could then use a backwards Euler discretization as an independent residual evaluator which you would expect to converge as a first order method.

$$L_y = \frac{y_i^{n+1} - yn_i}{h} - v_i^{n+1} + \mathcal{O}(h^2) \tag{26}$$

$$L_v = \frac{v_i^{n+1} - vn_i}{h} - \frac{y_{i-1}^{n+1} - 2y_i^{n+1} + y_{i+1}^{n+1}}{\Delta^2} + \mathcal{O}(h^2) \tag{27}$$