Workshop 14: Numerical Solvers for Ordinary Differential Equations

Adam G Peddle and Beth Wingate

This workshop will introduce you to the basic ideas for solving ordinary differential equations numerically. This is major area of study in modern mathematics in its own right, and is also used in solving all kinds of practical problems. We'll introduce a few key concepts and workshop a few practical examples.

The Analytical Perspective

When solving ordinary differential equations, we concern ourselves with searching for a solution to a problem of this form,

$$\frac{d\,y}{dt} = f(y,t) \qquad y(0) = y_0 \tag{1}$$

Where the initial condition is given. To illustrate this, let's look at a very simple ODE:

$$\frac{d\,y}{dt} = y(t) \tag{2}$$

Solving this means finding some function of t which is equal to its own derivative. We know this solution from mathematics:

$$y(t) = y_0\, e^t \tag{3}$$

where the constant, y_0, is fixed by the initial condition.[1] Let's check to see if this solution is correct. To do so, take the time derivative of Eq (4) and see if it identically satisfies our ODE, Eq (2).

$$\frac{d}{dt}y(t) = \frac{d}{dt}y_0\, e^t = y_0\frac{d}{dt}\, e^t = y_0\, e^t. \tag{4}$$

Therefore we've shown that the solution to this simple ODE is exactly Eq (4).

[1] This property (2) is what makes e such an important number and why it appears so often in physical problems.

What if the ODE is more complicated?

More complicated ODEs have solutions that cannot be written down analytically. Therefore, we will make a numerical approximation to help us estimate the answer. By writing an approximation for the derivative on the left-hand side we can transform the differential equation into an algebraic equation.

Let's consider our more general ODE, eq. (1), but we'll replace the analytical derivative with the numerical derivative[2]. This gives us:

$$\frac{y_{i+1} - y_i}{\Delta t} = f(y_i, t_i) \qquad y(0) = y_0 \tag{8}$$

where we use a subscript i to keep track of each discrete moment in time. Now we'll rearrange this to put what we want on the left-hand side and what we know on the right-hand side:

$$y_{i+1} = y_i + \Delta t f(y_i, t_i) \tag{9}$$

Let's look at the first few time steps using Eq 9. We were given $y(0)$ as the initial condition. We will call our first point in time $y_0 = y(0)$. Then we can compute y_1

$$y_1 = y_0 + \Delta t f(y_0, t_0). \tag{10}$$

Now that we know y_1, we can compute the next value,

$$y_2 = y_1 + \Delta t f(y_1, t_1) \tag{11}$$

For any known point in time (subscript i) we can advance the solution by our so-called TIMESTEP to find the next one (subscript $i + 1$), like you see in figure 1. Starting with $i = 0$, we find y_1. *Then*, we repeat to find y_2, using what we know at t_1. Rinse and repeat. It's simple but tedious – perfect for a computer. This procedure is called a lot of things, but FORWARD EULER or EXPLICIT EULER are the most common.

[2] Numerical Approximations of derivatives can be thought of as approximations of the slope of a line. Consider what you learned in the Calculus in NSC1002 (Lecture 10-14 notes, and Tutorial Examples) :

$$\frac{dy}{dt} = \lim_{\Delta t \to 0} \frac{y(t + \Delta t) - y(t)}{\Delta t} \tag{5}$$

. We make our first numerical approximation of the derivative by dropping the 'limit'

$$\frac{dy}{dt} \approx \frac{y(t + \Delta t) - y(t)}{\Delta t}. \tag{6}$$

If we use Python notation, where we have discrete points in time (e.g. using np.linspace()), then this formula is,

$$\frac{dy}{dt} \approx \frac{y_{i+1} - y_i}{\Delta t}. \tag{7}$$

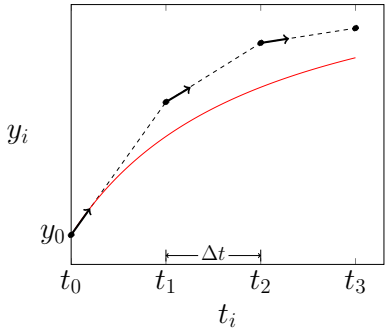

Figure 1: Solving ODEs numerically. The analytical solution is shown in red, and the numerical approximation is dashed. The arrows show the calculated derivatives at the points, which are just the slopes of the 'shots' to the next point.

Workshop Exercises

1. Using the explicit Euler method, solve the following ODE:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = e^{t - t^2} - 2ty \qquad y(0) = -1 \tag{12}$$

on the time domain from 0 to 2 and plot the result. Try taking timesteps of 0.1 and 0.05 and look at what it does to the solution. Compare the results to the analytical solution for this ODE, which you may take to be:

$$y(t) = e^{t - t^2} - 2e^{-t^2} \tag{13}$$

It's a good idea throughout this section to write your solver and trial functions in a way that they can be re-used easily – you'll be reusing them a fair bit today.

You may wish to first write the first few steps, like we did above, before using a loop! You may wish to create a function for solving the next problem!

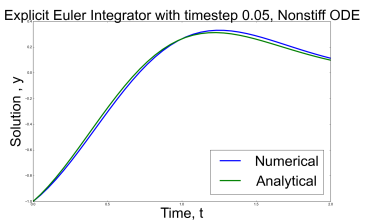

Figure 2: Here is an example comparison between numerical and analytical solutions for the ODE in Question 1.

2. Using your explicit Euler code from the first question, solve:

$$\frac{dy}{dt} = -30y \qquad y(0) = 1 \tag{14}$$

up to $t = 1$. For reference, the analytical solution is:

$$y(t) = e^{-30t} \tag{15}$$

First try the same timesteps as you used in the previous section. You should see some foolishness in the solution. Now try a very tiny timestep, like 0.001. Why is this? The answer of course is that this ODE is a so-called STIFF PROBLEM, which we shall treat in the next section.

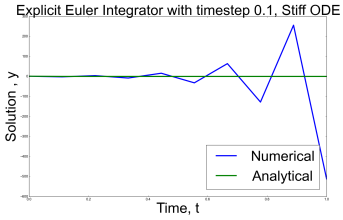

Figure 3: Note the numerical instability as compared to the smooth analytical solution.

Stiff Problems

A STIFF DIFFERENTIAL EQUATION is one for which explicit methods don't work so well.[3] For big problems, this can make them too expensive to solve. We'll then turn our attention to IMPLICIT methods, in particular the IMPLICIT a.k.a. BACKWARD EULER solver.

[3] Believe it or not, that's one of the best definitions available. They're not only hard to solve – they're hard to talk about too.

The basic idea for using an implicit solver is that they are a lot more stable for stiff problems. It's only in the implementation that the dragon begins to rear its ugly head. Backward Euler is almost like forward Euler (9), except that you use the value of f at the unknown point instead of the known one, like this:

$$y_{i+1} = y_i + \Delta t f(y_{i+1}, t_{i+1}) \tag{16}$$

Almost looks same, right? The problem is now, you have a difficult implicit non-linear problem to solve at each timestep. [4] If we rewrite the bit above to put everything on the left-hand side, we have:

[4] Your practices with matrices and vectors will come in handy here!

$$\Delta t f(y_{i+1}, t_{i+1}) - y_{i+1} + y_i = 0 \tag{17}$$

Note that we always know y_i, and we know t_{i+1} because we choose the timestep. All we don't know is y_{i+1} and the value of the function it depends on. So, we pull out our favourite root-finding algorithm and let it give us the answer. Every. Single. Timestep. These methods are a lot more expensive, but sometimes they make up for it by letting us take much longer timesteps.

Workshop Exercises

3. (Optional/Advanced) Code up an implicit Euler solver and use it to solve the problems from Questions 1 and 2 on the same domains and with the same timesteps. You'll need to know the formula for a central difference derivative:

$$\frac{\mathrm{d}f(x,t)}{\mathrm{d}x} \approx \frac{f(x+h/2,t) - f(x-h/2,t)}{h} \tag{18}$$

where h is some suitably small timestep. You can also use a Newton-Raphson type idea, which we used in root-finding earlier in the term, which will find x where $f(x) = 0$. Starting from some initial guess, x_0, just keep iterating:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{19}$$

until after n iterations, $f(x_n) < \epsilon$. 0.0001 is a good value for ϵ today. You should find that the implicit method doesn't go unstable like the explicit one did, although it's not necessarily a whole lot more accurate for the coarse timestep.