Maths Honours – Numerical Modelling

Topic 2: Numerical Differential Equations

A common method for modelling time-dependent systems is via a **differential equation** of the form

$$\dot{u} = f(u,t)\,, \qquad \text{where} \qquad u = u(t)$$

or more generally, a **system of differential equations**

$$\dot{\vec{u}} = \vec{f}(\vec{u},t) \qquad \text{where} \qquad \vec{u} = \left[u_1(t)u_2(t),\ldots,u_n(t)\right].$$

The vector field $\vec{u}$ is called the **state vector**. It's a time-dependent vector containing a set of scalar evolution variables, $u_1,\ldots,u_n$.

The RHS of this equation gives an instruction for how the $\vec{u}$ on the LHS changes, i.e. it gives the direction that $\vec{u}$ is moving. By following this direction, we can map out a solution to the differential equation.

This suggests a solution method:

1. We measure $\dot{u}$ by evaluating the RHS of $\dot{\vec{u}} = \vec{f}$.
2. We take a step in that direction to update u.
3. Then we use the new $\vec{u}$ value to re-calculated $\dot{\vec{u}}$ and update the direction of motion.

Repeating these steps will trace out a curve along the solutions of the differential equation.

The problem with doing this in a computer is that we can't take infinitely small steps along the curve. Consider the definition of the derivative:

$$\frac{du}{dt} = \lim_{\Delta t\to 0}\frac{u(t+\Delta t)-u(t))}{\Delta t}\,.$$

In a computer, the $\Delta t \to 0$ creates a problem. First of all, there is a limit to the smallest number a digital computer can compute with **floating point** decimal numbers. This is called the **floating point accuracy**. For a **double precision** number, the smallest unit of Δt is around 4.95×10^{-324}, which is quite small but still very distant from actual zero.

But a more practical problem is that if we take arbitrarily small steps, and each step takes a finite amount of time, then it will take an infinite amount of time to get anywhere along the line, and will possibly require an infinite amount of memory if we'd like to store those steps. As a result, when we apply a method like the one listed above, we usually take steps significantly larger than the theoretical minimum Δt supported by the computer.

We'd like to develop a practical way to *approximate* derivatives using a fixed Δt, and also to assess the errors that arise when we make the approximation.

In this and the following exercises, we will focus on the method of **finite differences** as a solution to these questions. In Semester 2, the course on **spectral methods** will present another good solution.

Computations on a grid

As an example, consider the following differential equation for $u = u(t)$:

$$\dot{u} = ru(1-u)\,, \qquad \text{where } r \text{ is some constant.}$$

Since we are not able to use a continuous time variable, we will instead evaluate the function at a set of times t^n seperated by a fixed spacing Δt:

$$t_0 = 0 \qquad t_1 = \Delta t \qquad t_2 = 2 \times \Delta t \qquad \ldots \qquad t_N = N \times \Delta t$$

The coordinate values of t^n make up our **computational grid**. We'll evaluate our approximation to u at each of these times, and store the value in a variable called v^n:

$$v^n \simeq u(t^n) \qquad v^n \simeq u(t^n) \qquad v^n \simeq u(t^n) \qquad \ldots \qquad v^N \simeq u(t^N)$$

If we like, we can store these values in arrays

$$\vec{t} = \left[t^0, t^1, t^2, \ldots, t^N\right] \qquad \vec{v} = \left[v^0, v^1, v^2, \ldots, v^N\right]$$

(This is useful for plotting output, though most of the methods we'll develop only need the last point v^n to calculate v^{n+1}.)

Euler's method

As our first attempt to approximation du/dt, we note that the derivative gives us the **slope** of the function at a point which for a straight line we could write as

$$m = \frac{\Delta u}{\Delta t} = \frac{u(t^n + \Delta t) - u(t^n)}{\Delta t} \simeq \frac{v^{n+1} - v^n}{\Delta t}$$

In the last step, we've replaced $u(t^n)$ with our approximation v^n. We can use this expression to replace du/dt in our differential equation:

$$\dot{u} = ru(1-u) \qquad \Longrightarrow \qquad \frac{v^{n+1} - v^n}{\Delta t} = rv^n(1-v^n)\,.$$

Here we've used the "current" value, v^n, to evaluate the RHS. If we solve for the "future" value, v^{n+1}, we arrive a the following formula:

$$v^{n+1} = v^n + \Delta t\, rv^n(1-v^n)\,. \tag{1}$$

This suggests a useful method to evaluate $\vec{v}$. If we know v^n, we can evaluate the RHS and use this to evaluate v^{n+1}. Then we can use this to re-evaluate the RHS to calculate v^{n+2}, etc. That is, given an initial value $v^0 = u(t^0)$ we can use Eq. (1) to evaluate the rest of the entries in $\vec{v}$ sequentially:

$$v^0 \quad \longrightarrow \quad v^1 \quad \longrightarrow \quad v^2 \quad \longrightarrow \quad \cdots \quad \longrightarrow \quad v^N\,.$$

Note, though, that each of these steps is an approximation, each step introduces a (hopefully small) error. We might expect that the value of v^N has the largest total error since it accumulates all of the error in the previous computations. But at the moment, it's difficult to quantify the error and how it might depend on the step Δt. For this, we proceed to a more rigorous derivation of the Euler method.

Taylor expansion approximations

An alternative way to approach the approximation is to consider a Taylor expansion of the original function:

$$u(t + \Delta t) = u(t) + \Delta t \dot{u}(t) + \frac{1}{2!}\Delta t^2 \ddot{u} + \cdots$$

We solve this equation for $\dot{u}(t)$

$$\dot{u}(t) = \frac{u(t + \Delta t) - u(t)}{\Delta t} - \frac{1}{2!}\Delta t \ddot{u} + \cdots$$

We can approximate the first term on the RHS using v^{n+1} and v^n. The second term involves $\ddot{u}$, which we don't know. But it's proportional to Δt, so for small values of Δt this term should be small. And the other "$\cdots$" terms are proportional to Δt^2, Δt^3, etc., so will be even smaller. We typically write this as

$$\dot{u}(t) = \frac{v^{n+1} - v^n}{\Delta t} + \mathcal{O}(\Delta t)$$

where $\mathcal{O}(\Delta t)$ indicates terms whose leading order are proportional to Δt. Substituting this into the original ODE, we arrive at the formula

$$v^{n+1} = v^n + \Delta\, r v^n (1 - v^n) + \mathcal{O}(\Delta t^2)\,.$$

When we implement this method we neglect the $\mathcal{O}(\Delta t^2)$ term, since it can't be evaluated without knowing the entire Taylor expansion of u. But for small values of Δt, the size of this and the following terms will decrease rapidly, so we hope that neglecting them does not spoil our approximation too badly.

We've arrived at a formula which is identical to the Euler method described above. For the more general problem $\dot{u} = f(u, t)$, the Euler method reads

$$v^{n+1} = v^n + \Delta t\, f(v^n, t^n) \qquad \text{(Euler's method)} \tag{2}$$

where the function f on the RHS is evaluated using v^n and t^n, which introduces an error of size $\mathcal{O}(\Delta t^2)$ each time it is evaluated. We will explore these errors in more detail in the next assignment.

Midpoint method

We can exploit the Taylor expansion to try to come up with alternate, hopefully more accurate, methods to approximate a derivative. Consider the following two formulae that we find by doing Taylor expansions to each side of a point t^n:

$$u(t^n + \Delta t) = u(t^n) + (\Delta t)\,\dot{u}(t^n) + \frac{1}{2!}(\Delta t)^2\ddot{u}(t^n) + \frac{1}{3!}(\Delta t)^3\dddot{u}(t^n) + \cdots$$

$$u(t^n - \Delta t) = u(t^n) - (\Delta t)\,\dot{u}(t^n) + \frac{1}{2!}(\Delta t)^2\ddot{u}(t^n) - \frac{1}{3!}(\Delta t)^3\dddot{u}(t^n) + \cdots$$

When we subtract the second formula from the first, we get

$$u(t^n + \Delta t) - u(t^n - \Delta t) = 2(\Delta t)\dot{u}(t^n) + \frac{1}{3!}(\Delta t)^3\dddot{u}(t^n) + \cdots$$

and rearranging to solve for $\dot{u}$ we find

$$\dot{u}(t^n) = \frac{u(t^n + \Delta t) - u(t^n - \Delta t)}{2\Delta t} + \frac{2}{3!}(\Delta t)^2\dddot{u}(t^n) + \cdots$$

By replacing $u(t^n \pm \Delta t)$ with $v^{n\pm1}$ values, we arrive at the approximation

$$\dot{u}(t^n) = \frac{v^{n+1} - v^{n-1}}{2\Delta t} + \frac{2}{3!}(\Delta t)^2\dddot{u}(t^n) + \cdots$$

Note that the second term in the expansion on the RHS is proportional to $(\Delta t)^2$ and all of the later terms will have higher exponents, so we can re-write this as

$$\dot{u}(t^n) = \frac{v_{j+1} - v_{j-1}}{2\Delta t} + \mathcal{O}(\Delta t^2)$$

Substituting this into the differential equation $\dot{u} = f(u, t)$ gives

$$\frac{v^{n+1} - v^{n-1}}{2\Delta t} = f(v^n, t^n) + \mathcal{O}(\Delta t^2)$$

or, on solving for v^{n+1},

$$v^{n+1} = v^{n-1} + 2\Delta t f(v^n, t^n)\,. \qquad \text{(Midpoint method)} \tag{3}$$

Compare this with Eq. (2) for Euler's method. In this case, we take two steps from v^{n-1} to v^{n+1}, evaluating f at the midpoint. The Δt factor is doubled to take into account the double-sized step.

The error term for this method works out to be $\mathcal{O}\Delta^3$, compared to $\mathcal{O}(\Delta t^2)$ for the Euler method. This suggests that for small values of Δt the midpoint method should have a lower overall error.

Runge-Kutta methods

Finally we develop a third method for evaluating the differential equation $\dot{u} = f(u,t)$ which belongs to a class called **multistep methods**. As an example of this type of method, we will derive the 2nd-order **Runge-Kutta** algorithm (also called **RK2**) which works as follows. Given v^n at some time t^n, we compute two intermediate quantities:

$$k_1 = \Delta t\, f(v^n, t^n) \tag{4a}$$

$$k_2 = \Delta t\, f(v^n + k_1, t^n + \Delta t) \tag{4b}$$

Then we combine these two results to find the next value

$$v^{n+1} = v^n + \frac{1}{2}k_1 + \frac{1}{2}k_2\,. \qquad \text{(RK2)} \tag{4c}$$

This method turns of to have an error of $\mathcal{O}(\Delta t^3)$ with each step, similar to the multistep method. But in this case, we don't need to store two previous timesteps, v^n and v^{n-1}. Given v^n, we can compute k_1, and then we can use this result to compute k_2.

- The factor k_1 can be viewed as a first approximation to the RHS, using v^n. Our first approximation to the new value of v^{n+1} is an Euler step to find $\tilde{v} = v^n + k_1$.
- Then k_2 uses this updated $\tilde{v} = v^n + k_1$ as its input, effectively calculating f at the $n+1$ timelevel.
- In the end, we choose the average of these two approximations for f to determine v^{n+1}.

One disadvantage is that we need to do two function evaluations of f, which might be expensive if the RHS of the ODE is very complicated. But generally, this method is fairly efficient and robust in comparison to the midpoint methods, for reasons that we'll explore in future weeks.

We derive the method as follows: Consider the following calculations:

$$k_1 = \Delta t\, f(u,t) \tag{5a}$$

$$k_2 = \Delta t\, f(u + \beta k_1, t + \alpha \Delta t) \tag{5b}$$

$$u(t + \Delta t) = u(t) + a k_1 + b k_2 \tag{5c}$$

where we've introduced four constants α, β, a and b. We will choose these constants to eliminate as many terms in the Taylor expansion for $u(t+\Delta t)$ as possible.

We have

$$u(t^n + \Delta t) = u(t^n) + (\Delta t)\dot{u}(t^n) + \frac{1}{2!}(\Delta t)^2 \ddot{u}(t^n) + \mathcal{O}(\Delta t^3)$$

But notice that we can replace the $\dot{u}$ and $\ddot{u}$ terms on the RHS using f from our original ODE:

$$\dot{u} = f\,, \qquad \ddot{u} = \frac{d}{dt}\dot{u} = \frac{d}{dt}f = \frac{\partial f}{\partial t} + f\frac{\partial f}{\partial u}\,,$$

where we've made use of the chain rule in the last step. Using these replacements in the Taylor expansion leads to

$$u(t+\Delta t) = u(t) + (\Delta t)f + \frac{1}{2!}(\Delta t)^2\left(\frac{\partial f}{\partial t} + f\frac{\partial f}{\partial u}\right) + \mathcal{O}(\Delta t^3) \tag{6}$$

We can also expand the expression for k_2 using a Taylor expansion:

$$\begin{aligned} k_2 &= \Delta t\, f(u + \beta k_1, t + \alpha\Delta t) \\ &= \Delta t\left(f(u,t) + \alpha\Delta t\frac{\partial f}{\partial t} + \beta k_1\frac{\partial f}{\partial u} + \mathcal{O}(\Delta t^2, u^2)\right) \\ &= k_1 + \Delta t\left(\alpha\Delta t\frac{\partial f}{\partial t} + \beta k_1\frac{\partial f}{\partial u}\right) + \mathcal{O}(\Delta t^3, k_1^2) \end{aligned}$$

Substituting this expression into Eq. (5c), we find

$$u(t+\Delta t) = u(t) + (a+b)(\Delta t)f + b(\Delta t)^2\left(\alpha\Delta t\frac{\partial f}{\partial t} + \beta k_1\frac{\partial f}{\partial u}\right) + \mathcal{O}(\Delta t^3) \tag{7}$$

Comparing the coefficients of similar terms between Eqs. (6) and (7) leads to the following equations for the coefficients:

$$a + b = 1 \qquad \alpha b = \frac{1}{2} \qquad \beta b = \frac{1}{2}$$

One way that we can satisfy these three equations is to set

$$a = \frac{1}{2} \qquad b = \frac{1}{2} \qquad \alpha = 1 \qquad \beta = 1$$

By making these choices, we end up with a formula for $u(t+\Delta t)$ which is $\mathcal{O}(\Delta t^3)$ in time.

Making the replacements $u(t) \to v^n$ as usual, we arrive at the method described in Eqs. (4).

Note that these choices for the parameters are not unique. We have three equations and four parameters, so there is one remaining degree of freedom. As a result, there is a whole family of Runge-Kutta methods which are $\mathcal{O}(\Delta t^3)$ but which make use of different parameter choices.

Exercises — Solving ODEs numerically

1. Consider the following ODE problem for $u = u(t)$, involving the logistic equation:

$$\dot{u} = ru(1-u)\,, \qquad u(0) = u_0$$

   a) Use the forward Euler method to solve this problem for $r = 1$ and $u_0 = 0.05$. Use a timestep of $\Delta t = 0.01$ and solve the problem in the range $0 \le t \le 20$. Plot the resulting approximation for $u(t)$ as a function of time.

   b) Repeat Question (1a) using the midpoint method with the same parameter values. Note that this method has a complication at the first step, since you need *two* previous time values:

$$v^1 = v^{-1} + 2\Delta f(v^0, t^0)$$

   but you only have v^0, not v^{-1}. To get around this, use an Euler step for the first step to get v^1. Once you have v^1, you can use a midpoint method for the following steps, e.g.

$$v^2 = v^0 + 2\Delta f(v^1, t^1)$$

   c) Repeat Question (1a) using the RK2 method with the same parameter values.

2. Repeat Question 1 for initial data ranging from $u_0 = -0.1$ to $u_0 = 2$ in steps of 0.1, i.e.

$$u_0 = -0.1,\ 0.0,\ 0.1,\ 0.2,\ \ldots,\ 1.9,\ 2.0$$

   Use any of the three methods you prefer.

   For each choice of u_0, you'll get a new curve $u(t)$. Plot all of these curves on the same axes. Can you identify any stable or unstable equilibrium points? Compare this with your analytical understanding of the logistic equation (e.g. from MAP312 Dynamical Systems).

3. Repeat Question 1 but for the following ODE:

$$\dot{u} = u - \frac{1}{2}e^{t/2}\sin(5t) + 5e^{t/2}\cos(5t)\,, \qquad u(0) = 0\,.$$

   Solve the problem between $t = 0$ and $t = 5$ using $\Delta t = 0.08$. Repeat the solution with $\Delta t = 0.04$. Plot both curves on the same axes.

   Produce this plot for each of the three methods.