NSDE 1: LECTURE 3

TYRONE REES*

Recap

$$u' = f(t,u), \qquad u(t_0) = u_0.$$

Consider a grid of $m+1$ equally spaced points, $t_{n+1} = t_n + h$. Approximate the solution $u(t_0)$, $u(t_1), \ldots, u(t_m)$ by U_0, $U_1, \ldots, U_m$, which are calculated by

$$\begin{aligned} U_0 &= u(t_0) \\ U_{n+1} &= U_n + hf(t_n, U_n)) \end{aligned}$$

We can think of Euler's method in terms of integrals:

$$\begin{aligned} \int_{t_n}^{t_{n+1}} u'\,dt &= \int_{t_n}^{t_{n+1}} f(x,u)\,dt \\ u(t_{n+1}) - u(t_n) &= \int_{t_n}^{t_{n+1}} f(t,u)\,dt \\ u(t_{n+1}) &= u(t_n) + \int_{t_n}^{t_{n+1}} f(t,u)\,dt \\ u(t_{n+1}) &\approx u(t_n) + h\,f(t_n, u(t_n)) \end{aligned}$$

However, we can also make a different approximation:

$$\begin{aligned} u(t_{n+1}) &= u(t_n) + \int_{t_n}^{t_{n+1}} f(t,u)\,dt \\ u(t_{n+1}) &\approx u(t_n) + h\,f(t_{n+1}, u(t_{n+1})) \end{aligned}$$

Which leads to the algorithm:

Euler's implicit method

$$\begin{aligned} U_0 &= u(t_0) \\ U_{n+1} &= U_n + h\,f(t_{n+1}, U_{n+1}) \end{aligned}$$

This has an obvious problem: U_{n+1} appears on the right hand side of the equation. Sometimes (e.g., on problem sheets) such problems can be solved straightforwardly, but usually, in practice, we'll need to use a nonlinear equation solver (e.g., Newton's method).

*Rutherford Appleton Laboratory, Chilton, Didcot, UK, tyrone.rees@stfc.ac.uk

This is why this method is called **implicit**.

We could also do a more sensible (from the point of view of approximating the integral) approximation of

$$\int_{t_n}^{t_{n+1}} f(t,u)\,dt \approx \frac{h}{2}(f(t_n,u(t_n)) + f(t_{n+1},u(t_{n+1})))$$

This motivates what's known as the **Trapezium rule method**:

$$\begin{aligned} U_0 &= u(t_0) \\ U_{n+1} &= U_n + \frac{h}{2}(f(t_n,U_n) + f(t_{n+1},U_{n+1})) \end{aligned}$$

And these are all specific instances of a **θ-method**:

$$\begin{aligned} U_0 &= u(t_0) \\ U_{n+1} &= U_n + h\left((1-\theta)f(t_n,U_n) + \theta f(t_{n+1},U_{n+1})\right) \end{aligned}$$

which takes a **weighted** average of the end points to approximate the integral.

EXAMPLE 0.1.

$$y' = x - y^2, \qquad y(0) = 0$$

Solve using a θ-method for $\theta = 0, 1/2,$ and 1.

Accuracy

How can we know beforehand how accurate the method is going to be? Before we can answer this, we need a concept of error.

The **global error** of any method is defined as

$$e_n = u(t_n) - U_n.$$

We can also define the **truncation error**. Note that the methods we've looked at so far can be written in the form

$$U_{n+1} = U_n + h\Phi(t_n, t_{n+1}, U_n, U_{n+1}; h)$$

We can re-write this in a form that explicitly models the derivatives:

$$\frac{U_{n+1} - U_n}{h} = \Phi(t_n, t_{n+1}, U_n, U_{n+1}; h).$$

The trunction error is the difference between the left and right hand sides *if we plug in the exact solution:*

$$T_n = \frac{u(t_{n+1}) - u(t_n)}{h} - \Phi(t_n, t_{n+1}, u(t_n), u(t_{n+1}); h)$$

If $T_n = O(h^p)$, where p is the largest such integer: the method is said to be p**th order accurate**.

EXAMPLE 0.2. **Euler's method: truncation error**

$$\begin{aligned} T_n &= \frac{u(t_{n+1}) - u(t_n)}{h} - f(t_n, u(t_n)) \\ &= \frac{u(t_{n+1}) - u(t_n)}{h} - u'(t_n) \end{aligned}$$

We can expand $u(t_{n+1})$ via a Taylor series to give

$$u(t_{n+1}) = u(t_n + h) = u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(\xi_n), \qquad \xi_n \in [t_n, t_{n+1}]$$

Subsituting this in we get

$$T_n = \frac{u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(\xi_n) - u(t_n)}{h} - u'(t_n) = \frac{1}{2}hu''(\xi_n)$$

Euler's method is first order accurate.

The truncation error is usually 'easy' to calculate, but it tells us little, by itself, about the quality of the solution. What we really want to know is the size of the global error. Now,

$$\begin{aligned} u(t_{n+1}) &= u(t_n) + hf(t_n, u(t_n)) + hT_n \\ U_{n+1} &= U_n + hf(t_n, U_n) \\ \hline e_{n+1} &= e_n + h[f(t_n, u(t_n)) - f(t_n, U_n)] + hT_n \end{aligned}$$

and, taking absolute values of both sides:

$$|e_{n+1}| \leq |e_n| + h|f(t_n, u(t_n)) - f(t_n, U_n)| + h|T_n|$$

If $f(\cdot,\cdot)$ satisfies a Lipschitz condition, then $|f(t_n, u(t_n)) - f(t_n, U_n)| \leq L|u(t_n) - U_n|$, and so we have

$$|e_{n+1}| \leq (1 + Lh)|e_n| + h|T_n| \qquad \text{for all } n$$

Now if we let $T = \max |T_n|$, we have

$$|e_{n+1}| \leq (1 + Lh)|e_n| + hT \qquad \text{for all } n$$

By induction, we get that

$$|e_n| \leq (1 + Lh)^n|e_0| + \frac{T}{L}[(1 + Lh)^n - 1] \qquad \text{for all } n$$

We know that $e_0 = 0$, and also that

$$(1 + Lh)^n \leq [e^{Lh}]^n = e^{Lhn} = e^{L(t_n - t_0)},$$

where we have used the facts that

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \geq 1 + x \qquad \text{for } x > 0$$

and that

$$t_n = t_0 + n\,h.$$

Therefore we get

$$|e_n| \leq \frac{T}{L}\left[e^{L(t_n - t_0)} - 1\right]$$

Finally, note that

$$T = \max_n |T_n| = \max_n \frac{h}{2}|u''(\xi_n)| \leq \frac{1}{2} h M_2$$

where we have set $M_2 = \frac{1}{2}\max_{t\in[t_n,t_{n+1}]}|u''(t)|$. We therefore have that

$$|e_n| \leq \frac{M_2}{2L}(e^{L(t_n - t_0)} - 1)h \leq \text{Const.}h$$

Hence, if you half the step size, you (at least) half the error.