Computational Projects

Lecture 4: Solution of ODEs

Note: this lecture will cover material likely useful for a core IB project (and several other IB and II projects)

Introduction

Computers are often used for solving ordinary differential equations (ODEs) as well as partial differential equations

For this lecture, we consider a simple class of ODEs: consider $x \in \mathbb{R}$ and some (unknown) function $y\colon \mathbb{R} \to \mathbb{R}$.

We are given a function f, an interval $[a, b]$ and an initial condition y_0 such that

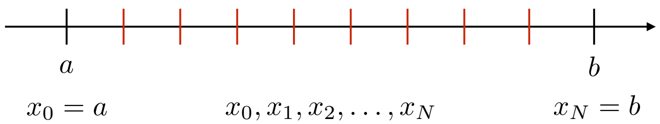
$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y)$$

and $y(a) = y_0$.

We seek a numerical approximation to the function y, for values of x in the interval $[a, b]$

Euler's method

A simple method for doing this is called Euler's method



Choose an increasing sequence of N points, in the interval $[a, b]$

Simplest choice: equally spaced points

$$x_n = a + nh, \qquad h = \frac{b-a}{N}$$

Notation

We will compute a sequence $Y_0, Y_1, \ldots, Y_N$ such that Y_n is our estimate of $y_n = y(x_n)$

Position	Exact solution	Approx solution
$x_0 = a$	$y_0 = y(a)$	$Y_0 = y_0$
$x_1 = a + h$	y_1	Y_1
$\vdots$	$\vdots$	$\vdots$
$x_n = a + nh$	y_n	Y_n
$\vdots$	$\vdots$	$\vdots$
$x_N = a + Nh = b$	$y_N = y(b)$	Y_N

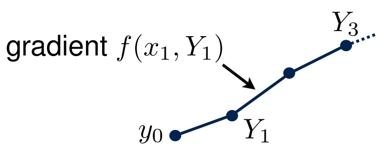
Euler's method

The Euler method takes

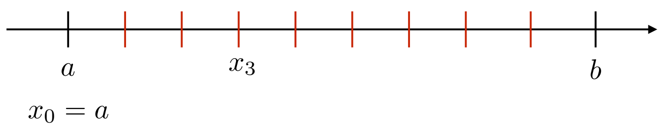
$$Y_{n+1} = Y_n + hf(x_n, Y_n)$$

...think of Taylor's theorem

$$y(x_n + h) = y(x_n) + hf(x_n, y_n) + O(h^2)$$



...we are making a piecewise-linear approximation to y.



Simple ODE example

Consider $\dfrac{dy}{dx} = f(x, y) = xy^2$

Exact solution, for any constant C

$$y(x) = \frac{2}{C - x^2}$$ (e.g. by separation of variables)

Note the 2 asymptotes when $x = \pm\sqrt{C}$

Initial condition:

$$y(0) = 1 \Rightarrow C = 2$$

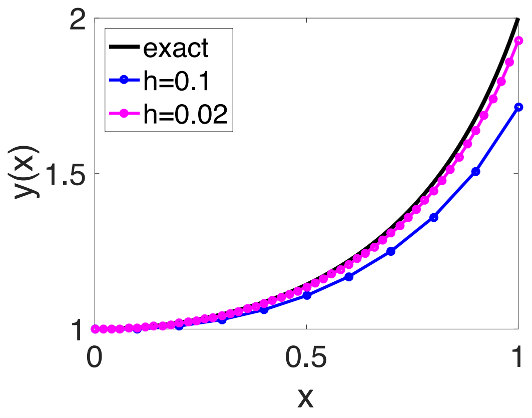
Euler method -- MATLAB Function

```
function [x, y] = eulerSolve(xstart, ystart, xend, h)
% eulerSolve: return data points using Euler's method to solve y' = xy^2
%   xstart, ystart determine the initial condition
%   xend sets the end point and h is the step size
%   returns 2 column vectors, estimates of x and y(x) at n points

    % the "round" function gives the closest integer to some real number
    n = round((xend-xstart+eps)/h);
    % adjust h so that the range is exactly n*h
    % (to ensure that we have exactly x(n+1) = xend
    hTrue = (xend-xstart)/n;

    x(1) = xstart;
    y(1) = ystart;

    for i=1:n
        yprime = x(i)*y(i)^2;
        y(i+1) = y(i) + hTrue*yprime;
        x(i+1) = x(i) + hTrue;
    end
return
```

... might be nice to modify eulerSolve so that it solves (dy/dx) = f(x,y) where f is an input to the function (as in the binarySearch example)

Example: `eulerSolve.m`

Effect of step size



`eulerExample.m`

We plot the exact solution and the numerical solution from Euler's method. As $h \to 0$ we approach the exact solution.

Accuracy

There are several ways to assess the accuracy of our numerical estimates

The simplest quantity to consider is the error at step n,

$$E_n = Y_n - y_n$$

We can also consider the **local error** which is the error that we make in a single step of the algorithm, under the assumption that our previous steps were all exactly correct.

Suppose that we already computed Y_{n-1}. Let $\tilde{y}(x)$ be the solution to our original ODE, with initial condition $\tilde{y}(x_{n-1}) = Y_{n-1}$.

Then define $z_n = \tilde{y}(x_n)$. The local error is

$$e_n = Y_n - z_n$$

Local error: Euler

If $e_n = O(h^{p+1})$ as $h \to 0$ (for fixed x_{n-1}, Y_{n-1}) then we say that we have an "order p method".

Assume that the solution $\tilde{y}$ is "nice enough" (e.g. analytic)

From Taylor's theorem

$$\tilde{y}(x_n) = \tilde{y}(x_{n-1}) + h\tilde{y}'(x_{n-1}) + \tfrac{1}{2}h^2\tilde{y}''(\xi_{n-1})$$

for some $\xi_{n-1} \in [x_{n-1}, x_n]$.

$$\begin{aligned} e_n &= Y_n - \tilde{y}(x_n) \\ &= \underbrace{\boxed{[Y_{n-1} + hf(x_{n-1}, Y_{n-1})]}}_{Y_n} - \underbrace{\boxed{[Y_{n-1} + hf(x_{n-1}, Y_{n-1}) + \tfrac{1}{2}h^2\tilde{y}''(\xi_{n-1})]}}_{\tilde{y}(x_n)} \\ &= -\tfrac{1}{2}h^2\tilde{y}''(\xi_{n-1}) = O(h^2) \end{aligned}$$

Therefore, p=1: the local error of Euler's method is order 1 (for "nice enough" ODEs)

Global error

Let $Y(x,h)$ be our piecewise-linear estimate of $y(x)$, obtained with step size h. Then $E_n = Y(x_n, h) - y(x_n)$ is the global error (from before).

Also let $E(x,h) = Y(x,h) - y(x)$ so we have also $E_n = E(x_n, h)$.

Rough argument:

To estimate E_n, we must consider n steps of the algorithm. It seems reasonable to assume that the error on each step is similar to the local error, hence $O(h^{p+1})$.

Since we need to make x/h steps in order to reach the point x, we guess that

$$E(x,h) = O(x/h \times h^{p+1}) = O(h^p)$$

(taking $h \to 0$ at fixed x)

Global error

We gave a (rough) argument that if the local error is $O(h^{p+1})$ then the global error is $O(h^p)$.

This argument is correct (and can turned into a rigorous proof) if f is bounded, continuous, and satisifes a Lipschitz condition: there exists some finite L such that for all x, y, z

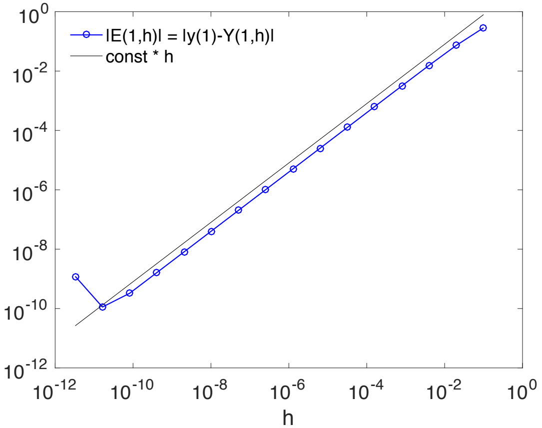
$$|f(x,y) - f(x,z)| < L|y - z|$$

(but note f is not bounded in our example. . .)

Recall an "order p method" has a local error that is $O(h^{p+1})$. In this case it has a global error that is $O(h^p)$. . . this justifies the name. . .

Computing errors

The example program `eulerTest.m` generates the graphs that appear in the next few slides



We solve our original ODE $y'(x) = f(x, y)$ with $f(x, y) = xy^2$ and $y(0) = 1$

We compute the global error at $x = 1$

Data are consistent with $E(1, h) = O(h)$, except for round-off error at very small h

graph is `eulerErr1.pdf`

Round-off in ODEs

In each step of the Euler method, we introduce a round-off error (on Y_n) of the order of the machine epsilon ϵ

In the worst case, all these errors would have the same sign. In computing $y(x)$ we have x/h steps so the global error on $y(x)$ due to round-off is then $(x/h) \times O(\epsilon)$

This error would be small for the values of h where one typically uses the method, but it diverges at small h so it limits the maximal accuracy. (This can be a good reason to use a higher-order method.)

If one assumes that the signs of the round-off errors are completely random, one predicts instead an error of order $|x/h|^{1/2} \times O(\epsilon)$. This is smaller but still divergent at small h.

Testing the order of a method

To see how our method is performing, we should measure $E(x, h)$... but of course we do not usually know the true solution $y(x)$.

If $E(x, h) = O(h^p)$ then

$$Y(x, h) = y(x) + \lambda(x)h^p + \dots$$

This means that

$$Y(x, h) - Y(x, h/2) = \left(1 - 2^{-p}\right) \lambda(x)h^p + \dots$$

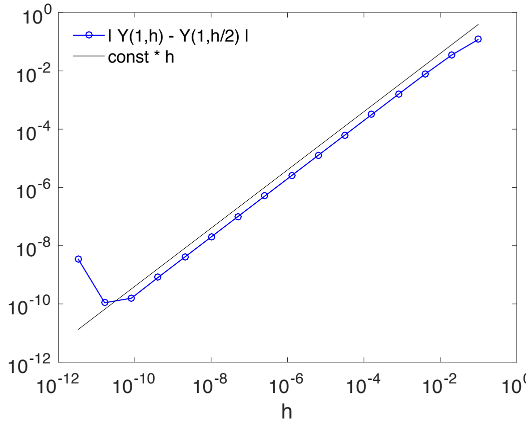
so

$$\log|Y(x, h) - Y(x, h/2)| = p \log h + \log\left[\left(1 - 2^{-p}\right)|\lambda(x)|\right] + \dots$$

A plot of $\log|Y(x, h) - Y(x, \frac{h}{2})|$ against $\log h$ has gradient p (for small h)

Computing errors

If we don't know the exact solution, we can also estimate the order of the method by comparing step sizes h and $h/2$



graph is `eulerErr2.pdf`

Data are consistent with $\log|Y(1, h) - Y(1, h/2)| = \log h + O(h^0)$, that is $p = 1$. (Again, round-off problem at very small h)

... an improved estimate

If we know p and we compute approximate solutions using two different values of h, we can "extrapolate to $h = 0$" in order to get a more accurate answer

Assuming

$$Y(x, h) = y(x) + \lambda(x)h^p + O(h^{p+1})$$

we can define

$$Y_R(x, h) = \frac{2^p Y(x, h/2) - Y(x, h)}{2^p - 1}$$

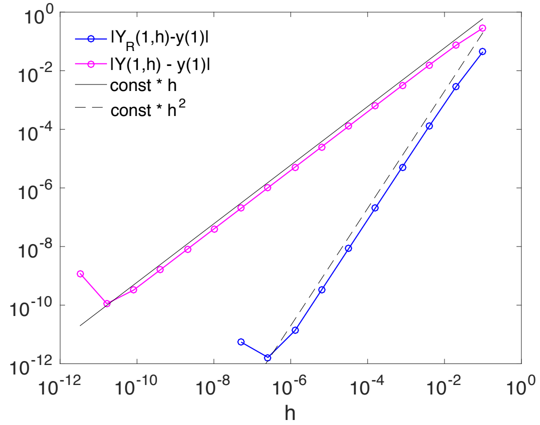
and show that

$$Y_R(x, h) = y(x) + O(h^{p+1})$$

This is called the Richardson method. It is more accurate, by a factor of order h.

Computing errors

Compare the error on the Richardson estimate with the regular estimate. . .



In this case $p = 1$ so
$Y_R(1, h) = 2Y(1, h/2) - Y(1, h)$

graph is `eulerErr3.pdf`

Data are consistent with $\log |Y_R(1, h) - y(1)| = O(h^2)$.

What can go wrong?

If we want accurate solutions, we can try to design or analyse higher-order methods. . .

For this course, a more important question is what can go wrong: there are at least two things to worry about here. . .

(1) ODEs that are "not nice enough"

(for example, if f is not bounded or not Lipshitz then all our arguments above can fail. . .)

see `eulerExample2.m` for some of the effects of the asymptotes that appear at x = C in our simple example

(2) Stability – we can make statements about the limit $h \to 0$ but in practice we work at finite h. . .

Stability

Consider the differential equation

$$\frac{dy}{dx} = -\lambda y$$

Exact solution

$$y = y_0\, e^{-\lambda(x-x_0)}$$

Euler's method

$$Y_n = Y_{n-1} - h\lambda Y_{n-1} = Y_{n-1}(1 - \lambda h)$$

... hence

$$Y_n = Y_0(1 - \lambda h)^n$$

So $Y(x, h) = y_0(1 - \lambda h)^{x/h}$, at least for $x = nh$

Stability

We have

$$Y_n = y_0(1-\lambda h)^n, \qquad Y(x,h) = y_0(1-\lambda h)^{x/h}$$

Can check that $\lim_{h\to 0} Y(x,h) = y(x)$, we get the exact solution . . . seems ok

Clearly $\left|\frac{Y_{n+1}}{Y_n}\right| = |1-\lambda h|$. For $h < (2/\lambda)$, the $|Y_n|$ form a decreasing sequence, consistent with the exact solution.

The problem comes if we take $h > (2/\lambda)$. In this case the $|Y_n|$ increase and the approximate solution $Y(x,h)$ diverges exponentially fast from $y(x)$, as x increases.

This is an example of a 1st order method that becomes *unstable* when h is not small enough.

Stability vs accuracy

If a method is pth order, this is a statement about the limiting behaviour as $h \to 0$, this is related to accuracy of the solution

This says nothing about the behaviour at finite h: the method might be unstable in which case the error diverges

In “real-world applications” there is often a trade-off between stability and accuracy: what is important in that specific application?

... today

a brief overview of ODE solution by a simple (Euler) method, and associated errors...

... more complex methods certainly exist, see later courses and also the computational projects themselves...

... next lecture

a more complicated algorithm, to illustrate how to build up programs from simple starting points...

... matrix inversion by LU decomposition