MATH1072

Practical 5: Numerical Integration Methods for Solving First-Order ODEs

Motivation: We have already explored one method of numerically solving first-order ODEs: Euler's Method. We established that Euler's method is a first-order integration method which is derived using first-order (linear) approximations through Taylor expansion. Recall that we also remarked on Heun's method which is a second-order integration method from the family know as *Runge-Kutta* methods.

The derivation of each improved order comes from increasing the number of terms in the Taylor approximation at each point. An added benefit to increasing the order of integration is the affect it has on the stability of the method. For those interested in pursuing numerics in further courses, you will have plenty of opportunities to rigorously derive and establish upper-bounds on the local and global errors of these methods. For now, we will explore two methods for numerically solving first-order ODEs.

1 Euler's method

(a) Consider the initial value problem given by,

$$\frac{dy}{dt} = \sin(t + y),\ y(0) = 0. \tag{1}$$

Find the solution to (1) using Euler's method for integration with step-size $h = 0.1$ on the interval $0 \leq t \leq 10$. Hint: Define Euler's method in a for loop and save the value at each step.

```
t = ; %defined vector t as a mesh [0,10] with step size
   0.1
y(1) = ; %initialize the solution with the initial
   condition
dy = ; %define the implicit function representing the ODE
for i =  %complete the for-loop
    y(i) = ;
end
```

(b) Repeat (a) for step-size $h = 10^{-6}$.

(c) Plot your results from (a) and (b) on the same plot. If you're interested, play around with different markers, marker sizes, colors, line widths to make your plots as nice as possible. What do you see?

2 Built-in higher-order solvers

MATLAB has many ODE (initial value problem) solvers. We input details of the initial value problem to be solved, and the solver outputs vectors of data points for t and the numerical solution y. It does not output a function as a solution, just data points. We can then visualise the solution by viewing a table of data or by plotting these data points. Given an initial value problem of the form (1), we can use a MATLAB ODE solver by entering

```
[T, Y] = solver(rhs, [a,b], Y0);
```

where solver can be one of the following commands: `ode45, ode23, ode113, ode78, ode89, ode15s, ode23s, ode23t, ode23tb, ode15i`. Each command represents a different algorithm used to generate the solution data. The solver `ode45` is usually the first one to try, and would be sufficient for most problems in MATH1072. The input `rhs` is a function defined using a function handle (implicit representation), representing the function $f(t, y)$ on the right hand side of the ODE $\frac{dy}{dt} = f(t, y)$ and `Y0` is the initial condition.

(a) Use `ode45` to solve (1). Use the outline code below.

```
rhs = %define the implicit function representing the ODE
[T, Y] = solver(rhs, [a,b], L); %complete the argument
```

(b) Plot the solution from (a). How does this compare to the solutions found via Euler's method? Stability of the method is KEY!

(c) In some cases, we may be interested to know the value of the solution y at a specific value of t in a specified interval. Say we want to know the value of the solution in our example at $t = 1.3$. One way would be to use the `deval` command.

```
sol = ode45(rhs, [a,b], L); %complete the argument
yVal = deval(sol,1.3) %give the value of y at t = 1.3
```

3 Extra Problems

Consider an initial value problem

$$\frac{dy}{dt} = t - y,\ y(0) = y_0.$$

Find the value y_0 so that the solution satisfies $y(1) = 0.5$.