**M.Eng. 2.6 Mathematics** **February 2002** **Problem sheet 7**

This sheet can be found on the Web: http://www.ma.ic.ac.uk/~ajm8/MEng26

**1.** Write down Euler's method on the regular grid $x_n = nh$ for the problem

$$y' = -y \qquad \text{in} \quad x > 0 \qquad \text{with} \quad y(0) = 1 \ ,$$

and show that $y_n$, the approximation to $y(x_n)$, is given by

$$y_n = (1-h)^n \ .$$

What is the corresponding result for the Trapezium Method?
Which is the more accurate approximation to the exact solution $y(x_n) = e^{-nh}$?
{*Answer:* $y_n = \left[(1-\frac{1}{2}h)/(1+\frac{1}{2}h)\right]^n$}

**2.** Consider the Runge–Kutta formula on the regular grid $x_n = nh$

$$\begin{aligned} y^*_{n+1} &= y_n + hf(x_n,\, y_n) \\ y_{n+1} &= y_n + \tfrac{1}{2}h\big[f(x_n,\, y_n) + f(x_{n+1},\, y^*_{n+1})\big] \end{aligned}$$

for computing approximate solutions of the equation $y' = f(x,\, y)$. Show that this method has order 2 (so that the local error $E_n \propto h^3$).

**3.** Use the above method to find an approximation to the solution of

$$y' = 1 - y \qquad \text{with} \quad y(0) = 0$$

at the point $x = 0.1$ using a step $h = 0.05$. Compute the error in the approximate solutions at $x = 0.05$ and $x = 0.1$ by finding the exact solution of the differential equation.
{*Answer:* $y_1 = 0.04875$, $y_2 \simeq 0.09512$. Exact solution $y(.1) \simeq 0.0951625$}

**4.** Show that the local truncation error of the Runge–Kutte scheme

$$\begin{aligned} y^*_{n+1} &= y_n + \tfrac{1}{2}hf(x_n,\, y_n) \\ y_{n+1} &= y_n + hf(x_n + \tfrac{1}{2}h,\, y^*_{n+1}) \end{aligned}$$

is proportional to $h^3$.

**5.** Use the method of (4) to find an approximation to the solution of

$$y' = xy \qquad \text{with} \quad y(0) = 1$$

at $x = \frac{1}{2}$, using a step $h = \frac{1}{4}$. Compare the solution obtained with the exact solution.
{*Answer:* $y_1 \simeq 1.03125$, $y_2 \simeq 1.13095$ Exact solution $y = \exp(\frac{1}{2}x^2)$, $y(.5) \simeq 1.13315$.}

While it is possible to perform a few steps by hand with a calculator (and exam questions sometimes ask you to do this), it is just as easy to write a simple program to do it, which can of course perform thousands of steps per second. For example, the following program, written in Pascal (by someone unfamiliar with the language!) probably solves the first part of question 3 overleaf. Simple modifications should solve question 5.

```
program runge

    {Program to implement a second order Runge-Kutte scheme}

var
  x, y, h, f1, f2: real,
  i: integer;

function f(x, y: real): real;
begin
  f := 1 − y
end;

begin
  x := 0.0;
  h := 0.05;
  y := 0.0;
  for i := 1 to 2 do
  begin
    f1 := f(x, y);
    f2 := f(x + h, y + h ∗ f1);
    y := y + 0.5 ∗ h ∗ (f1 + f2);
    x := x + h;
    writeln(x, y)
  end
end.
```