ECE 204 *Numerical methods*

Project 4

In this course, we have described how Euler's method is O(h), Heun's method is O(h^2) and the 4th-order Runge-Kutta method is O(h^4). This is by looking at the error when estimating the value of the solution at a point $y(t_f)$ and when there are n points approximating the solution between $[t_0, t_f]$ ($t_k = t_0 + kh$ where $h = (t_f - t_0)/n$ and k runs from 1 to n and $t_n = t_f$).

Another measure of error is the root-mean-squared error (RMSE). For a sequence of values from 0 to n, if a_k is an approximation to y_k, the root-mean-squared error is defined as:

$$\sqrt{\frac{1}{n+1}\sum_{k=0}^{n}\left(y_k - a_k\right)^2}\ .$$

Note the name: we are taking the square root of the average (mean) of the sum of errors squared, hence the name.

Suppose one real-valued function a approximates a real-valued function y on the interval $[t_0, t_f]$. In this case, the root-mean-squared error is given by:

$$\sqrt{\frac{1}{t_f - t_0}\int_{t_0}^{t_f}\left(y(t) - a(t)\right)^2 dx}\ .$$

You will recognize that $\frac{1}{t_f - t_0}\int_{t_0}^{t_f}\left(y(t) - a(t)\right)^2 dx$ is the formula for the average (mean) value of the square of the error on the given interval, so once again, the name is appropriate.

Your task is to determine if there is a relationship between h and the RMSE for Euler's, Huen's and the 4th-order Runge-Kutta method as h gets small (but not too small).

Required approach:

1. Choose a value of n.
2. Use Euler's, Heun's and 4th-order Runge-Kutta to approximate solutions at t_k for $k = 1, \ldots, n$.
3. You now have to approximate $\int_{t_0}^{t_f} \left(y(t) - a(t)\right)^2 dt$ where $a(t)$ is the approximating function, but we only know that $a(t_k) = y_k$ and at best we can interpolate between these points. Instead, you can choose to use either the trapezoidal rule or Simpson's rule, whichever you determine is more appropriate:

$$\int_{t_0}^{t_f} \left(y(t) - a(t)\right)^2 dt \approx \sum_{k=0}^{n-1} \frac{h}{2}\left[\left(y(t_k) - y_k\right)^2 + \left(y(t_{k+1}) - y_{k+1}\right)^2\right]$$

$$\int_{t_0}^{t_f} \left(y(t) - a(t)\right)^2 dt \approx \sum_{k=0}^{\frac{n}{2}-1} \frac{2h}{6}\left[\left(y(t_{2k}) - y_{2k}\right)^2 + 4\left(y(t_{2k+1}) - y_{2k+1}\right)^2 + \left(y(t_{2k+2}) - y_{2k+2}\right)^2\right]$$

   Of course, if you use Simpson's rule, you must ensure your initial n is even.
4. Repeat this with $2n$ (so halving h). Recalculate the RMSE. Determine if there is a relationship between the h and the RMSE. (Is it $O(h)$, $O(h^2)$, etc.?)

You will use ODEs that are tailored to your uWaterloo Student ID number. If your uWaterloo Student ID number is 20**123456**, then one ODE will be

$$y^{(1)}(t) + 1.\mathbf{456}y(t) = 0$$

and the other will be

$$y^{(1)}(t) + 1.\mathbf{123}y(t) = 1.\mathbf{2345}\cos(t)$$

In all cases, your initial condition will be $y(0) = 1$. You can chose any t_f you wish, but $t_f = 10$ is reasonable.

With Matlab (using `format long`), you are capable of finding a solution to these IVPs. You will then implement solutions to each of these, so in the above case, I would implement, respectively,

```
double y1( double t ) {
    return std::exp(-1.456*t);
}

double y2( double t ) {
    return 0.6131200387063277*std::cos(t)
         + 0.5459661965327940*std::sin(t)
         + 0.3868799612936723*std::exp(-1.123*t);
}
```

These are the exact solutions, and you will then proceed to test the approximations of Euler's, Heun's and the 4th-order Runge-Kutta methods in comparison with the exact solution by calculating the RMSE.

You may use Matlab or C++ or any other programming language you wish. You can, if you wish, start with the code at

https://replit.com/@dwharder/ECE-204-Project-4