The Runge - Kutta Method of Numerically Solving Differential Equations

We have spent some time in the last few weeks learning how to discretize equations and use Euler' s Method to find numerical solutions to differential equations. The Euler Method is traditionally the first numerical technique taught, since it is simple to understand and geometrically easy to articulate. However, because it assumes the rate of change is constant between any two evaluation points, the method has limited accuracy for more complicated functions (i.e., those functions whose derivatives can change rapidly).

A more robust and intricate numerical technique is the Runge - Kutta (RK) method. Before comparing these two techniques, let' s recall that the basic idea of the Euler Method is :

$$\text{value of y at (n)} = \text{value of y at (n} - 1) + \text{change in value of y between (n} - 1) \text{ and n.}$$

Whereas the Euler method assumes the derivative is constant between two evaluation points, the Runge - Kutta method computes the derivative at four points in the evaluation interval (this means the distance between x[n - 1] and x[n], and uses a weighted average to determine the change in the value of the function between evaluation points. While the Euler method uses one straight line (i.e., one secant) to connect the beginning and end points of the interval), the RK method uses 4 secants.

We can best illustrate this idea by computing the following code :

```
(* Code for Runge-Kutta Numerical Method *)
Clear[x, y, k1, k2, k3, k4]
h = 0.025; x[0] = 0; y[0] = 0;
f[x_, y_] := Cos[x y] + 1
x[n_] := x[n] = x[n - 1] + h
k1[n_] := h f[x[n - 1], y[n - 1]]
k2[n_] := h (f[x[n - 1] + h / 2, y[n - 1] + k1[n] / 2])
k3[n_] := h (f[x[n - 1] + h / 2, y[n - 1] + k2[n] / 2])
k4[n_] := h f[x[n - 1] + h, y[n - 1] + k3[n]]
y[n_] := y[n] = y[n - 1] + (k1[n] + 2 k2[n] + 2 k3[n] + k4[n]) / 6

g1 = ListPlot[Table[{x[n], y[n]}, {n, 1, 240}]]
```

We are solving the same differential equation you encountered on a recent homework :

$$\frac{dy}{dx} = f(x, y) = \cos(x\,y) + 1$$

The difference in these two numerical techniques is seen clearly in the inclusion of the newly defined functions k1[n], k2[n], k3[n] and k4[n]. If you study these functions carefully, you should notice that k1[n] is simply the value of the derivative at the beginning of the evaluation interval; in fact, k1[n] is equal to the slope used in the Euler method; we could use k1[n] to solve this equation via the Euler method by simply writing :

$$y_{Euler}[n\_] := y_{Euler}[n-1] + h\,k1[n] \;=\; y[n-1] + h\,f[x[n-1], y[n-1]]$$

In the Euler method, the term h f[x[n - 1], y[n - 1]] represents the change in value of y between the points (n - 1) and n. Each of the "k" functions computes the slope of f (x, y) at four different points in the evaluation interval, thereby allowing us a more accurate calculation of how much the function changes between evaluation points. The penultimate line of code computes the new value of y :

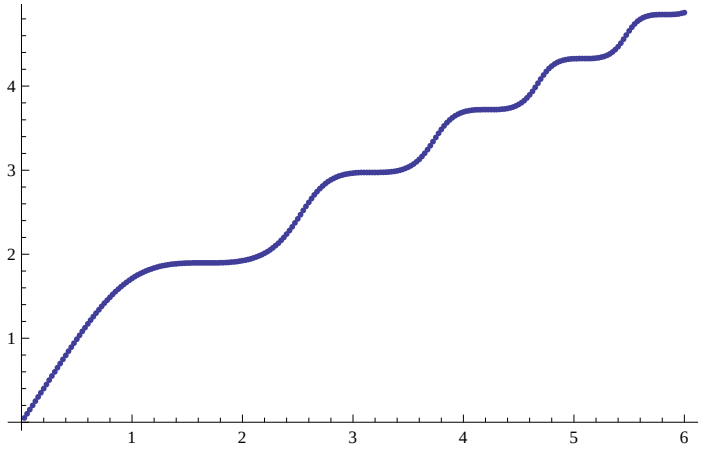
$$y[n] = y[n-1] + (k1[n] + 2\,k2[n] + 2\,k3[n] + k4[n])/6$$

This algorithm gives more weight to the slopes in the middle of the interval (k2 and k3) and less weight to the slopes at the beginning (k1) and end (k4) of the interval.

```
Clear[x, y, k1, k2, k3, k4]
h = 0.025; x[0] = 0; y[0] = 0;
f[x_, y_] := Cos[x y] + 1
x[n_] := x[n] = x[n - 1] + h
k1[n_] := h f[x[n - 1], y[n - 1]]
k2[n_] := h (f[x[n - 1] + h / 2, y[n - 1] + k1[n] / 2])
k3[n_] := h (f[x[n - 1] + h / 2, y[n - 1] + k2[n] / 2])
k4[n_] := h f[x[n - 1] + h, y[n - 1] + k3[n]]
y[n_] := y[n] = y[n - 1] + (k1[n] + 2 k2[n] + 2 k3[n] + k4[n]) / 6
```

```
g1 = ListPlot[Table[{x[n], y[n]}, {n, 1, 240}]]
```

Out[578]=



Find the solution using Euler' s method, and plot the two results on the same set of axes to show their agreemeent.

If you read up on the Runge - Kutta technique, you will find that there are many versions of this general technique. The particular method outlined above is often called the classical Runge - Kutta since it was the first version of this method published. It is also known as RK4 since there are 4 slopes computed for each new value of y. The method is also knows as the fourth order Runge - Kutta; in this context, "order" does not refer to the order of the differential equation, but refers to the 4 slopes computed at each step.