MATH 320, WEEK 5:
Numerical Approximation (Runge-Kutta)

Let's revisit the earlier example

$$\frac{dy}{dx} = x^2 + y^2, \qquad y(0) = 0. \tag{1}$$

No explicit solution for this formula can be found by the methods we have used so far, and in fact, no solution involving elementary functions exists for this differential equation. Nevertheless, a solution just exist because $f(x, y) = x^2 + y^2$ is continuous everywhere. In order to determine specific information about this solution, we must use a numerical method such at the forward Euler method or the Runge-Kutta method. Respectively, the formulas are given by

$$\text{Forward Euler:} \qquad y_{n+1} = y_n + f(x_n, y_n)\Delta x. \tag{2}$$

and

$$\begin{aligned} &y_{n+1} = y_n + \frac{\Delta x}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\ &\qquad\qquad \text{where} \\ &k_1 = f(x_n, y_n) \\ \text{Runge-Kutta:} \qquad &k_2 = f(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_1\Delta x) \\ &k_3 = f(x_n + \frac{1}{2}\Delta x, y_n + \frac{1}{2}k_2\Delta x) \\ &k_4 = f(x_n + \Delta x, y_n + k_3\Delta x). \end{aligned} \tag{3}$$

Let's suppose we have want information about the particular solution to (1) at the point $x = 1.5$—that is to say, we want to estimate $y(1.5)$. (For reference, the "true" value is $y(1.5) = 1.517447537$.) We will attempt to use the forward Euler method with the Δx values $\Delta x = 0.5, 0.1, 0.01$, and 0.001. We have our work cut out for us! We will need $(x_{final} - x_{initial})/\Delta x = 1.5/0.001 = 1500$ computations to produce the estimate for $y(1.5)$ using $\Delta x = 0.001$. Fortunately, computers can implement such recursive algorithms as Euler's method (and the Runge-Kutta method) very quickly.

We can carry out the procedure outlined in the lecture notes by hand to get the first few estimates with our calculators, but for small step-sizes we

Δx	$y(1.5)$	error	steps
0.5	0.6328125	0.884635	3
0.1	0.9307268557	0.5867207	15
0.01	1.479113716	0.0383338	150
0.001	1.513502037	0.0039455	1500

will definitely have to use a computer. The output from this gives the result contain in the table above.

We can see that there is a marked improvement by reducing the step size. This makes sense—we have less chance of floating far away from the true trajectory if we take smaller steps before correcting ourselves. We might wonder if there is a better way, however, and of course there is: we can choose a different numerical schemical scheme. The Runge-Kutta method is a popular choice and is known to produce less error per step than the forward Euler, but at the cost of being more computationally intensive during each step. Let's see how it performs for this example taking $\Delta x = 0.5, 0.1$ and 0.01.

The computations are easy enough to perform for Δx that we will do one step by hand. Thereafter, we will have to rely on a computer—or dedicate a significantly greater amount of time to this course than any of us currently have. As always, we have $x_1 = x_0 + \Delta x = (0) + (0.5) = 0.5$. To compute y_1, we need k_1, k_2, k_3, and k_4. We have

$$k_1 = f(x_0, y_0)\Delta x = x_0^2 + y_0^2 = (0)^2 + (0)^2 = 0$$

and

$$\begin{aligned} k_2 &= f(x_0 + \frac{1}{2}\Delta x, y_0 + \frac{1}{2}k_1\Delta x) \\ &= (x_0 + \frac{1}{2}\Delta x)^2 + (y_1 + \frac{1}{2}k_1\Delta x)^2 \\ &= ((0) + \frac{1}{2}(0.5))^2 + ((0) + \frac{1}{2}(0)(0.5))^2 = 0.0625 \end{aligned}$$

and

$$\begin{aligned} k_3 &= f(x_0 + \frac{1}{2}\Delta x, y_0 + \frac{1}{2}k_2\Delta x) \\ &= (x_0 + \frac{1}{2}\Delta x)^2 + (y_0 + \frac{1}{2}k_2\Delta x)^2 \\ &= ((0) + \frac{1}{2}(0.5))^2 + ((0) + \frac{1}{2}(0.0625)(0.5))^2 \\ &= 0.06274414 \end{aligned}$$

and

$$\begin{aligned} k_4 &= f(x_0 + \Delta x, y_0 + k_3 \Delta x) \\ &= (x_0 + \Delta x)^2 + (y_0 + k_3 \Delta x)^2 \\ &= ((0) + (0.5))^2 + ((0) + (0.06274414)(0.5))^2 \\ &= 0.250984206. \end{aligned}$$

That was a lot of work, and we haven't even computed the estimate y_1 yet! We finally have

$$\begin{aligned} y_1 &= y_0 + \frac{\Delta x}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ &= (0) + \frac{0.5}{6}((0) + 2(0.0625) + 2(0.06274414) + (0.250984206)) \\ &= 0.041789373. \end{aligned}$$

At this point, we are probably about to throw our hands up and swear off the Runge-Kutta method once and for all. This was a pile of work just to do one time-step! Before we despair too much, however, we should recognize that all the work have done is easily programmed into a computer, and that is exactly what is done in application. Letting my laptop do the rest of the work, in a fraction of a second we have the following estimates:

Δx	$y(1.5)$	error	steps
0.5	1.521061677	0.00361414	3
0.1	1.517473413	0.000025876	15
0.01	1.517447548	0.000000011	150

The reason we have gone through all of this trouble—or rather, let our computers go through all this trouble—should now be clear. The Runge-Kutta method gives a *significantly* better estimate of the true value per step. No matter how ridiculous we find the amount of computation necessary in each step to be, we cannot escape the *overall* efficiency. We have obtained a better estimate of $y(1.5)$ in three steps of the Runge-Kutta method (error= 0.00361414) than we obtained in 1500 iterations of the forward Euler method (error= 0.0039455). It should come as no surprise, therefore, to learn that the forward Euler method—while illustrative and intuitive—is never, ever, ever use in practice. Even though each step is easy to compute, the overall burden of cumulative errors makes it tremendously inefficient.