Exercise 5: For this very special initial value problem

$$y'(x) = y = f(x,y), \text{ slope fcn.}$$
$$y(0) = 1$$

which has $y(x) = e^x$ as the solution, set up Euler on the x-interval $[0, 1]$, with n subdivisions, and step size $\boxed{h = \frac{1}{n}}$. Write down the resulting Euler estimate for $\exp(1) = e$. What is the limit of this estimate as $n \to \infty$? You learned this special limit in Calculus!

Skip this (I'll fill it in after class)

$x_0 = 0,\ y_0 = 1;\ f(0,1) = 1$

$x_1 = \frac{1}{n},\ y_1 = y_0 + \underset{\uparrow\, f(x_0,y_0)}{1} \cdot \underset{\nwarrow h}{\frac{1}{n}} = 1 + \frac{1}{n};\quad f(x_1, y_1) = f\left(\frac{1}{n}, 1+\frac{1}{n}\right) = 1 + \frac{1}{n}.$

$x_2 = \frac{2}{n};\ y_2 = y_1 + \frac{1}{n} f(x_1, y_1) = \left(1+\frac{1}{n}\right) + \frac{1}{n}\left(1+\frac{1}{n}\right) = \left(1+\frac{1}{n}\right)\left(1+\frac{1}{n}\right) = \left(1+\frac{1}{n}\right)^2$

$x_3 = \frac{3}{n},\ y_3 = \left(1+\frac{1}{n}\right)^3$ (after some work)

$\vdots$

$x_n = \frac{n}{n} = 1;\ \boxed{y_n = \left(1+\frac{1}{n}\right)^n} \leftarrow$ e.g. with $n = 5$ & $h = .2$, this $= (1.2)^5 \approx 2.488$ that we did by hand

Start here Wednesday

$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e$!!

(L'Hôpital! go look at your Calculus text)

In more complicated differential equations it is a very serious issue to find relatively efficient ways of approximating solutions. An entire field of mathematics, ``numerical analysis" deals with such issues for a variety of mathematical problems. Our text explores improvements to Euler in sections 2.5 and 2.6, in particular it discusses improved Euler, and Runge Kutta. Runge Kutta-type codes are actually used in commerical numerical packages, e.g. in Maple and Matlab.

Let's summarize some highlights from 2.5-2.6.

Suppose we already knew the solution $y(x)$ to the initial value problem

$$\begin{cases} y'(x) = f(x,y) \\ y(x_0) = y_0. \end{cases} \qquad y'(t) = f(t, y(t))$$

If we integrate the DE from x to $x+h$ and apply the Fundamental Theorem of Calculus, we get

$$y(x+h) - y(x) = \int_x^{x+h} f(t, y(t))\,dt, \text{ i.e.} \qquad y(x+h) - y(x) = \int_x^{x+h} y'(t)\,dt$$

$$y(x+h) = y(x) + \left.\int_x^{x+h} f(t, y(t))\,dt.\right\} \approx f(x, y(x))\cdot h$$

One problem with Euler is that we approximate this integral above by $h \cdot f(x, y(x))$, i.e. we use the value at the left-hand endpoint as our approximation of the integrand, on the entire interval from x to $x+h$. This causes errors that are larger than they need to be, and these errors accumulate as we move from subinterval to subinterval and as our approximate solution diverges from the actual solution. The improvements to Euler depend on better approximations to the integral. These are subtle, because we don't yet have an approximation for $y(t)$ when t is greater than x, so also not for the integrand $f(t, y(t))$ on the interval $[x, x+h]$.

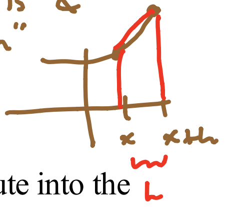
Improved Euler uses an approximation to the Trapezoid Rule to compute the integral in the formula

$$y(x+h) = y(x) + \int_x^{x+h} f(t, y(t))\, \mathrm{d}t\,.$$

Recall, the trapezoid rule to approximate the integral

average of integrand @ two endpts & mult by "h"

$$\int_x^{x+h} f(t, y(t))\, \mathrm{d}t$$



would be

$$\frac{1}{2} h \cdot \left(f(x, y(x)) + f(x+h, y(x+h)) \right).$$

Since we don't know $y(x+h)$ we approximate its value with unimproved Euler, and substitute into the formula above. This leads to the improved Euler "pseudocode" for how to increment approximate solutions.

Improved Euler pseudocode:

```
$k_1 = f(x_j, y_j)$                      #current slope
$k_2 = f(x_j + h, y_j + h k_1)$          #use unimproved Euler guess for $y(x_j + h)$ to estimate slope function when $x = x_j + h$
$k = \frac{1}{2}(k_1 + k_2)$             #average of two slopes
$x_{j+1} = x_j + h$                      #increment $x$
$y_{j+1} = y_j + h k$                    #increment $y$
```

Exercise 6 Contrast Euler and Improved Euler for our IVP

$f(x,y) = y$

$$y'(x) = y$$
$$y(0) = 1$$

$\}$ $y(x) = e^x$

to estimate $y(1) \approx$ e with one step of size $h = 1$. Your computations should mirror the picture below. Note that with improved Euler you actually get a better estimate for e with ONE step of size 1 than you did with 5 steps of size $h = 0.2$ using unimproved Euler. (It's still not a very good estimate.)

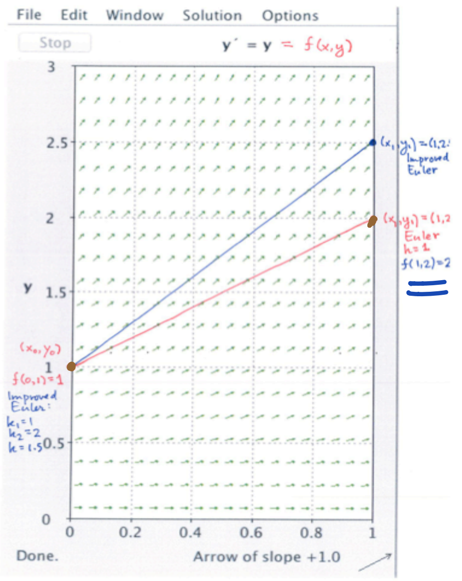
$x_0 = 0,\ y_0 = 1 \qquad k_1 = 1 \quad (= f(0,1) = 1)$

$k_2 = f(0+1, y_0 + 1 \cdot 1)$

$= f(1, 2) = 2$

$k = \frac{1}{2}(k_1 + k_2) = 1.5$

$x_1 = 1,\ y_1 = y_0 + hk = 1 + 1(1.5) = \boxed{2.5}$



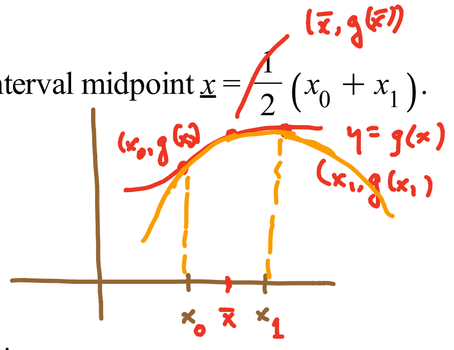
In the same vein as ``improved Euler" we can use the Simpson approximation for the integral for Δy instead of the Trapezoid rule, and this leads to the Runge-Kutta method. You may or may not have talked about Simpson's Parabolic Rule for approximating definite integrals in Calculus. It is based on a quadratic approximation to the integrand g, whereas the Trapezoid rule is based on "linear" approximation.

Simpson's Rule:

Consider two numbers $x_0 < x_1$, with interval width $h = x_1 - x_0$ and interval midpoint $\underline{x} = \frac{1}{2}(x_0 + x_1)$.
If you fit a parabola $p(x)$ to the three points

$$(x_0, g(x_0)),\ (\underline{x}, g(\underline{x})),\ (x_1, g(x_1))$$

then



$$\int_{x_0}^{x_1} p(t)\,dt = \underline{\frac{h}{6}\left(g(x_0) + 4\cdot g(\underline{x}) + g(x_1)\right)}.$$

(You will check this fact in your homework this week!) This formula is the basis for Simpson's rule, which you can also review in your Calculus text or at Wikipedia. (Wikipedia also has a good entry on Runge-Kutta.) Applying the Simpson's rule approximation for our DE, and if we already knew the solution function $y(x)$, we would have

$$y(x+h) = y(x) + \underline{\int_x^{x+h} f(t, y(t))\,dt}$$

$$\approx y(x) + \frac{h}{6}\cdot\left(\underbrace{f(x, y(x))}_{\text{left endpt}} + 4f\left(x + \frac{h}{2}, y\left(x + \frac{h}{2}\right)\right) + f(x+h, y(x+h))\right).$$

However, we have the same issue as in Trapezoid - that we've only approximated up to $y(x)$ so far. Here's the pseudo-code for how Runge-Kutta takes care of this. (See also section 2.6 to the text.)

Runge-Kutta pseudocode:

```
$k_1 = f(x_j, y_j)$   # left endpoint slope
$k_2 = f\left(x_j + \frac{h}{2}, y_j + \frac{h}{2} k_1\right)$ # first midpoint slope estimate, using $k_1$ to increment $y_j$
$k_3 = f\left(x_j + \frac{h}{2}, y_j + \frac{h}{2} k_2\right)$ # second midpoint slope estimate using $k_2$ to increment $y_j$
$k_4 = f(x_j + h, y_j + h\,k_3)$   # right hand slope estimate, using $k_3$ to increment $y_j$
$k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ #weighted average of all four slope estimates, consistent with Simpson's rule
$x_{j+1} = x_j + h$   # increment $x$
$y_{j+1} = y_j + h\,k$   # increment $y$
```

<u>Exercise 7</u> Contrast Euler and Improved Euler to Runge-Kutta for our IVP

$$y'(x) = y \leftarrow f(x,y)=y$$
$$y(0) = 1 \qquad x_0=0,\ y_0=1$$

to estimate $y(1) \approx$ e with one step of size $\underline{h = 1}$. Your computations should mirror the picture below. Note that with Runge Kutta you actually get a better estimate for e with ONE step of size $h = 1$ than you did with 100 steps of size $h = 0.01$ using unimproved Euler!

$x_0 = 0,\ y_0 = 1$

$k_1 = f(x_0, y_0) = f(0,1) = 1$

$k_2 = f(.5, y_0 + .5k_1) = f(.5, 1.5) = 1.5$

$k_3 = f(.5, \underset{1}{y_0} + .5 \cdot \underset{1.5}{k_2}) = f(.5, 1.75)$
$= 1.75$

$k_4 = f(1, \underset{\overset{\shortparallel}{1}}{y_0} + 1 \cdot \underset{\overset{\shortparallel}{1.75}}{k_3}) = f(1, 2.75) = 2.75$

$k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

$= \frac{1}{6}(1 + 3 + 3.5 + 2.75)$

$= \frac{1}{6}(10.25)$ $\qquad \begin{array}{r} 7.5 \\ 2.75 \\ \hline 25 \end{array}$

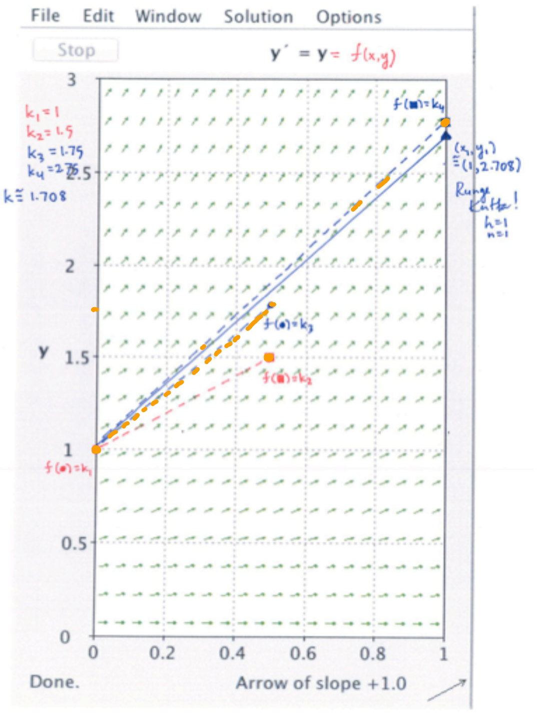
$= 1.708$

$x_1 = 1$

$y_1 = y_0 + hk = 1 + 1 \cdot (1.708)$

$= \underline{\underline{2.708}}$

vs. 100 steps of Euler.



Numerical experiments

I'll illustrate by doing the "famous numbers" problem of estimating e.

Estimate e *by estimating* $y(1)$ for the solution to the IVP

$$y'(x) = y$$
$$y(0) = 1.$$

Apply Runge-Kutta with n = 10, 20, 40 ... subintervals, successively doubling the number of subintervals until you obtain the target number below - rounded to 9 decimal digits - twice in succession.

```
> evalf(e);
                          2.718281828459045                                (1)
```

Initialize

```
> restart : # clear any memory from earlier work
> Digits := 15 : # we need lots of digits for "famous numbers"
>
> unassign(`x`, `y`); # in case you used the letters elsewhere, and came back to this piece of code
> f := (x, y) → y; # slope field function for our DE i.e. for y'(x)=f(x,y)
> x[0] := 0; y[0] := 1; #initial point
  h := 0.1; n := 10;   #step size and number of steps - first attempt for "famous number e"
```

Runge Kutta loop. WARNING: ONLY RUN THIS CODE AFTER INITIALIZING

```
> for i from 0 to n do #this is an iteration loop, with index "i" running from 0 to n
     print(i, x[i], y[i]); #print iteration step, current x,y, values, and exact solution value
     k1 := f(x[i], y[i]); #current slope function value
     k2 := f(x[i] + h/2, y[i] + h/2·k1);
       # first estimate for slope at right endpoint of midpoint of subinterval
     k3 := f(x[i] + h/2, y[i] + h/2·k2); #second estimate for midpoint slope
     k4 := f(x[i] + h, y[i] + h·k3); #estimate for right-hand slope
     k := (k1 + 2·k2 + 2·k3 + k4)/6; # Runge Kutta estimate for rate of change of y
     x[i + 1] := x[i] + h;
     y[i + 1] := y[i] + h·k;
  end do: #how to end a for loop in Maple
>
```

For other problems you may wish to use or modify the following Euler and improved Euler loops.

Euler loop: WARNING: ONLY RUN THIS CODE AFTER INITIALIZING

```
> for i from 0 to n do  #this is an iteration loop, with index "i" running from 0 to n
    print(i, x[i], y[i]);
        #print iteration step, current x,y, values, and exact solution value
    k := f(x[i], y[i]);  #current slope function value
    x[i + 1] := x[i] + h;
    y[i + 1] := y[i] + h·k;
  end do:  #how to end a for loop in Maple
>
```

Improved Euler loop: WARNING: ONLY RUN THIS CODE AFTER INITIALIZING

```
> for i from 0 to n do  #this is an iteration loop, with index "i" running from 0 to n
    print(i, x[i], y[i]);  #print iteration step, current x,y, values, and exact solution value
    k1 := f(x[i], y[i]);  #current slope function value
    k2 := f(x[i] + h, y[i] + h·k1); #estimate for slope at right endpoint of subinterval
    k := (k1 + k2)/2;  # improved Euler estimate
    x[i + 1] := x[i] + h;
    y[i + 1] := y[i] + h·k;
  end do:  #how to end a do loop in Maple
>
```