We've discussed 3 numerical techniques to approximate sol's to

\[
\begin{aligned}
\frac{dy}{dx} &= f(x,y) \\
y(x_0) &= y_0 \\
x_0 \leq x \leq x_n
\end{aligned}
\]

\[
h := (x_n - x_0) / n
\]

\[
xval[0] := x_0 \\
yval[0] := y_0
\]

\[
\text{for } i \text{ from } 1 \text{ to } n \\
x := xval[i] \\
y := yval[i] \\
k := f(x, y) \quad \# \text{slope} \\
xval[i+1] := x + h \\
yval[i+1] := y + h \times k \\
\text{od}
\]

\[
\text{* Improved Euler} \\
\text{for } i \text{ from } 1 \text{ to } n \\
x := xval[i] \\
y := yval[i] \\
k1 := f(x, y) \quad \# \text{left slope} \\
k2 := f(x + h, y + h \times k1) \\
\quad \# \text{right slope estimate, using Euler} \\
xval[i+1] := x + h \\
yval[i+1] := y + (k1 + k2) / 2 \times h \\
\quad \# \text{use average of } k1, k2 \\
\text{od}
\]
All three methods are based on the identity:

\[ y(x_{i+1}) = \int_{x_i}^{x_{i+1}} f(x, y(x)) \, dx \]

which follows from the Fundamental Theorem of Calculus and the DE:

\[ \frac{dy}{dx} = f(x, y(x)) \]

If we knew \( y(x) \), then we’d know \( f(x, y(x)) \):

Euler = “leftsum” = \( h \cdot f(x_i, y_i) \)

Trapezoidal rule for same integral:

\[ \frac{h}{2} \left[ f(x_i, y_i) + f(x_i+h, y(x_i+h)) \right] \]

Simpson’s rule:

\[ \frac{h}{6} \left[ f(x_i, y_i) + 4f(x_i+h, y(x_i+h)) + f(x_i+2h, y(x_i+2h)) \right] \]

= area (integral) of the parabola which agrees with the graph at \( x = x_i, x_i+h, x_i+2h \).
In case
\[
\begin{aligned}
\frac{dy}{dx} &= f(x) \\
y(0) &= y_0
\end{aligned}
\]
as in 6.1.2, then Runge-Kutta & Imp Euler are exactly Simpson and Trapezoid rule.

approximations of
\[
F(x_n) = y_0 + \int_{x_0}^{x_n} f(x) \, dx = \frac{d}{dx} \left( F(x) \right) = f(x) \quad (\text{sol of } x \frac{dy}{dx} = f(x) \quad y(0) = y_0)
\]
In particular
\[
F(x_n) \approx y_0 + \frac{h}{2} (f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)) \quad (\text{Trap})
\]
\[
\approx y_0 + \frac{h}{6} (f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + f(x_n)) \quad (\text{Simp})
\]

Error estimates
\[
|F(b) - \text{Trapest}(b)| \leq \left[ \max_{x \in [a,b]} |f''(x)| \right] \cdot [b-a] \cdot \frac{1}{12} \cdot h^2
\]
improves quadratically as \( h \) shrinks.

\[
|F(b) - \text{Simpest}(b)| \leq \left[ \max_{x \in [a,b]} |f^{(4)}(x)| \right] \cdot [b-a] \cdot \frac{1}{180} \cdot h^4
\]
improves as \( h^4 \)

\( \sim \) see Maple handout, for
\[
\begin{aligned}
\frac{dy}{dx} &= \cos x \\
y(0) &= 0
\end{aligned}
\]
\( 0 \leq x \leq \pi/2 \) \( (y(x) = \sin x) \)

For \( \frac{dy}{dx} = f(x,y(x)) \)

accuracy is much more subtle to determine!!
(See maple handout)
If the differential equation is of the type we studied in section 1.2,
\[
\frac{dy}{dx} = f(x),
\]
then the improved Euler and Runge-Kutta algorithms for approximating the solution are exactly the Trapezoid rule and Simpson's rule for numerical integration. (CHECK!) You may have studied these numerical methods in 1250 or 1220, and I'll review them in today's class notes.

Here's a good example:
\[
\frac{dy}{dx} = \cos(x) \\
y(0) = 0
\]

Of course, the solution is
\[
y(x) = \sin(x)
\]

And the slope field looks like
```maple
> with(DEtools):
> deqtn:=diff(y(x),x)=cos(x):
> DEplot(deqtn,y(x),x=0..Pi/2,[[y(0)=0],[y(0)=1],[y(0)=-1],
   [y(0)=-2]],y=-2..2,arrows=line,
   color=black,linecolor=black,dirgrid=[30,30],stepsize=.1,
   title='an easy DE to solve');
```

![Slope Field Image]
Let's try to solve this initial value problem with improved Euler and Runge-Kutta. We will only print our approximate value for \( \frac{1}{2} \frac{\pi}{2} \) which should be close to one.

```plaintext
restart:
Digits:=12:
x0:=0;y0:=0.0:
evalf(Pi/2):
> f := (x,y) -> cos(x):
print('improved-euler est ', 'error ', 'error bound '):
for m from 1 to 5 do
n:=2^m:
h:=(xn-x0)/n:
x:=x0:y:=y0:
for i from 1 to n do
k1:=f(x,y):
# left-hand slope
k2:=f(x+h,y+h*k1):
# approximation to right-hand slope
k:=(k1+k2)/2:
# approximation to average slope
x:=x+h:
# improved Euler update
y:=y+h*k:
od:
print(y,1-y,evalf((Pi/2)^(1/(12*n^2))):
```

Halving h quadruples accuracy

<table>
<thead>
<tr>
<th>Improved Euler est</th>
<th>error</th>
<th>error bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.948059448967</td>
<td>0.00194055103</td>
<td>0.0007455121884</td>
</tr>
<tr>
<td>0.98715809973</td>
<td>0.0012884199027</td>
<td>0.0021863780471</td>
</tr>
<tr>
<td>0.986785171887</td>
<td>0.0003214828113</td>
<td>0.00050465945178</td>
</tr>
<tr>
<td>0.999196680490</td>
<td>0.000008033195100</td>
<td>0.00126164862794</td>
</tr>
<tr>
<td>0.999799994335</td>
<td>0.000002008056650</td>
<td>0.000315412156987</td>
</tr>
</tbody>
</table>

> print('simpson's rule est ', 'error ', 'error bound '):
for m from 1 to 5 do
n:=2^m:
h:=(xn-x0)/n:
x:=x0:y:=y0:
for i from 1 to n do
k1:=f(x,y):
# left-hand slope
k2:=f(x+h/2,y+h*k1/2):
# 1st guess at midpoint slope
k3:=f(x+h/2,y+h*k2/2):
# second guess at midpoint slope
k4:=f(x+h,y+h*k3):
# guess at right-hand slope
k:=(k1+k2+k3+k4)/6:
# Simpson's approximation for the integral
x:=x+h:
# update
y:=y+h*k:
# y update
od:
print(y,1-y,evalf((Pi/2)^(5/(180*n^4))):
```

Halving h goes 16 times the accuracy!
\[
\begin{align*}
\text{eqn 1:} \quad & \frac{dy}{dx} = f(x, y) \\
& y(x_0) = y_0
\end{align*}
\]

is much more subtle as far as approximating solutions!

\[
\begin{align*}
\text{eqn 2:} \quad & \frac{dy}{dx} = -5y + 4e^{-x} \\
& y(0) = 1
\end{align*}
\]

\[
\begin{align*}
daq \quad & \frac{dy}{dx} = f(x, y) \\
& \text{if } f_y < 0 \text{ or if } f_y > 0? \text{ Explain.}
\end{align*}
\]

\[
\begin{align*}
\text{eqn 2:} \quad & \frac{dy}{dx} = 5y - 6e^{-x} \\
& y(0) = 1
\end{align*}
\]

\[
\begin{align*}
\text{eqn 2:} \quad & \frac{dy}{dx} = 5y - 6e^{-x} \\
& y(0) = 1
\end{align*}
\]

\[
\begin{align*}
\text{soln:} \quad & y = e^{-x}
\end{align*}
\]

Would you feel more confident approximating a solution to:

Would you feel more confident approximating a solution to:

\[
\begin{align*}
\text{eqn 2:} \quad & \frac{dy}{dx} = f(x, y) \\
& \text{if } f_y < 0 \text{ or if } f_y > 0? \text{ Explain.}
\end{align*}
\]

and, to see if you’re right, let’s try Runge Kutta! (Also, see pages 138-139 and your Maple project.)
"Easy case"
> y1:=x->exp(-x); #solution to first IVP
> y1 := x -> e^(-x)
> x0:=0; xN:=3; y0:=2;
> f:=(x,y)->5*y+4*exp(-x); #RHS for first IVP
> f := (x, y) -> 5*y + 4*exp(-x)
> print('Simpson's rule est.', 'error');
for n from 1 to 5 do
n:=2^n;
    h:=(xN-x0)/n;
x:=(x0+y0)/2;
    for i from 1 to n do
        k1:=f(x,y);
k2:=f(x+h/2,y+h*k1/2);
k3:=f(x+h/2,y+h*k2/2);
k4:=f(x+h,y+h*k3);
k:=k1+2*k2+2*k3+k4;
        #Simpson's approximation for the integral
        x:=x+h;
y:=y+h*k;
    od;
    print(y, y1(x)-y);
od:

"Hard case"
New let's try for the second DE IVP:
> y2:=y1; #solution to first IVP
> y2 := y1
> x0:=0; xN:=3; y0:=1;
> f:=(x,y)->5*y-6*exp(-x); #RHS for first IVP
> f := (x, y) -> 5*y - 6*exp(-x)
> print('Simpson's rule est.', 'error');
for n from 1 to 8 do
n:=2^n;
    h:=(xN-x0)/n;
x:=(x0+y0)/2;
    for i from 1 to n do
        k1:=f(x,y);
k2:=f(x+h/2,y+h*k1/2);
k3:=f(x+h/2,y+h*k2/2);
k4:=f(x+h,y+h*k3);
k:=k1+2*k2+2*k3+k4;
        #Simpson's approximation for the integral
        x:=x+h;
y:=y+h*k;
    od;
    print(y, y2(x)-y);
od: