Submit final review Qs via Canvas by 8am Sunday

Goals for Day 28

- Use the improved Euler method to approximate solutions to IVPs.
- Use the Runge-Kutta method for the same purpose.

Improved Euler Method

Remember that Euler's method approximates the solution to the first-order IVP

$$y' = f(t,y), \quad y(t_0) = y_0$$

as follows.

We choose a step size h and then set

$$t_{n+1} = t_n + h$$

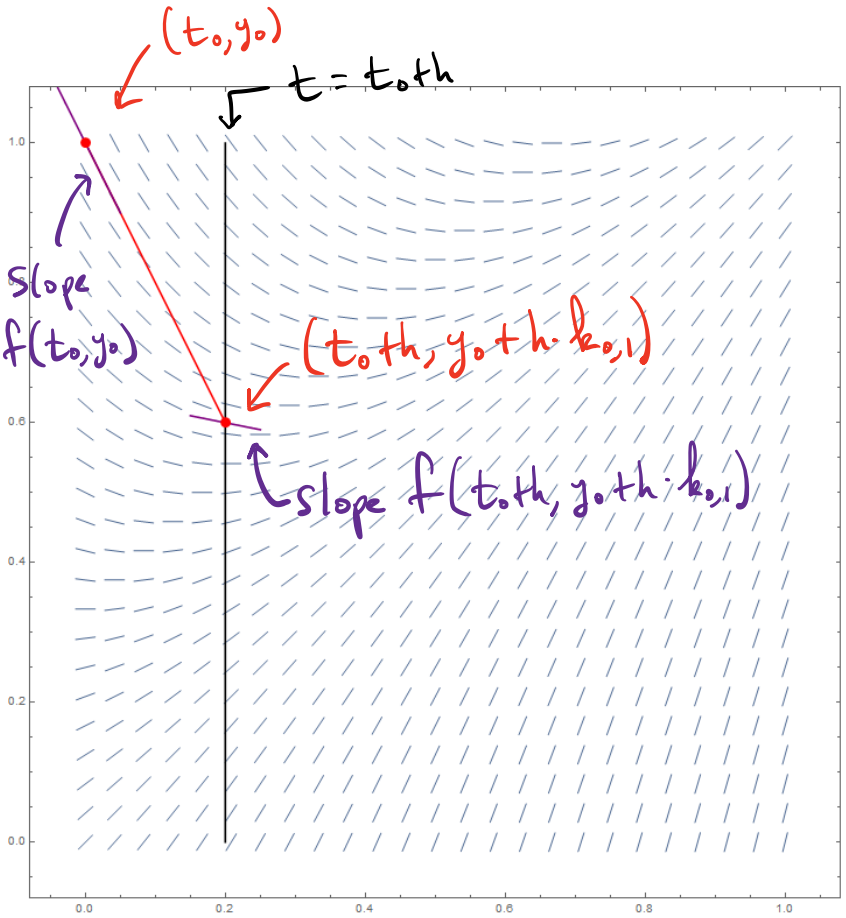
$$y_{n+1} = y_n + h \cdot f(t_n, y_n),$$

for $n \geq 0$.

We then connect $(t_0, y_0) \to (t_1, y_1) \to (t_2, y_2) \to \cdots$ linearly.

This method pretends that the slope $f(t_n, y_n)$ is valid throughout the interval $t_n \le t \le t_n + h$.

We'll improve this method by using the same step size, but more accurately approximating the average slope of $y(t)$ on the interval $t_n \le t \le t_n + h$.



Euler's method wants to use slope $k_{n,1} = f(t_n, y_n)$.

At the other end is slope $f(t_n + h, y_n + h \cdot k_{n,1})$, which is $k_{n,2}$.

We average these to get $\dfrac{k_{n,1} + k_{n,2}}{2}$.

Our next point is then

(t_{n+1}, y_{n+1}),

where $t_{n+1} = t_n + h$,

$y_{n+1} = y_n + h \cdot \frac{k_{n,1}+k_{n,2}}{2}$,

$k_{n,1} = f(t_n, y_n)$,

&

$k_{n,2} = f(t_n + h, y_n + h \cdot k_{n,1})$

The n in $k_{n,1}$ & $k_{n,2}$ tells us the interval of interest.

Very informally,

$k_{n,1}$ = slope @ LHS of interval

$(t_n + h, y_n + h \cdot k_{n,1})$ = fake next point

$k_{n,2}$ = slope @ fake next point

Ex Consider the IVP

$y' + 3y = 3t + 1, \quad y(0) = 1,$

with $h = 0.2$.

* avg. of -2 & -0.2

t_n	y_n	$k_{n,1}$	$(t_n+h,\ y_n+hk_{n,1})$	$k_{n,2}$	k	y_{n+1}
0	1	-2	(0.2, 0.6)	-0.2	-1.1 *	0.78
0.2	0.78	-0.74	(0.4, .632)	0.304	$-.218$	0.7364
0.4	0.7364	-0.0092	(0.6, 0.73456)	0.59632	0.29356	0.795112
0.6	0.795112	0.414664	(0.8, 0.878045)	0.765866	0.590265	0.913165
0.8	0.913165	0.660505	(1, 1.04527)	0.864202	0.762354	1.06564
1	1.06564		↑ fake next point			

$f(t,y) = 3t + 1 - 3y$

$k_{0,1} = f(t_0, y_0) = f(0,1) = 3\cdot 0 + 1 - 3 = -2$

$y_0 + h\cdot k_{0,1} = 1 + 0.2(-2) = 1 - 0.4 = 0.6$

$k_{0,2} = f(0.2, 0.6) = 3(0.2) + 1 - 3(0.6) = -0.2$

$y_1 = y_0 + h\cdot(-1.1) = 1 + (0.2)(-1.1) = .78$

$k_{1,1} = f(t_1, y_1) = f(0.2, 0.78)$

$= 3(0.2) + 1 - 3(.78) = -.74$

$y_1 + h\cdot k_{1,1} = .78 + (.2)(-.74) = .632$

$k_{1,2} = f(0.4, 0.632) = 3(.4) + 1 - 3(0.632)$

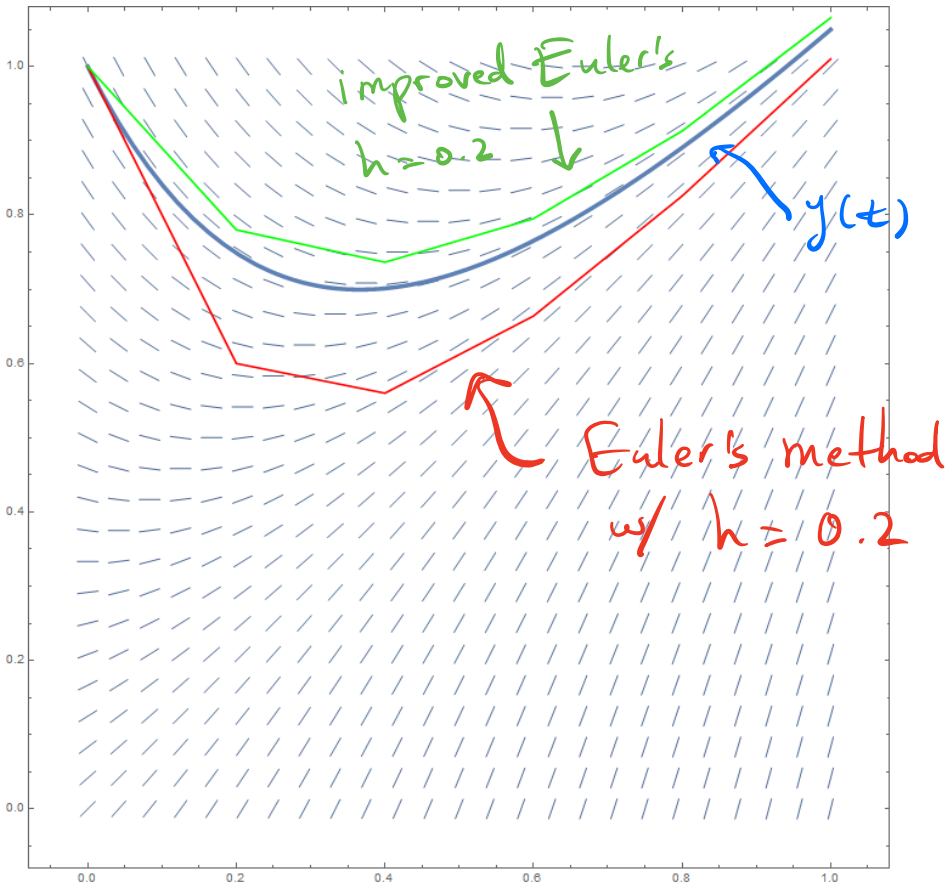
$= .304$

$y_2 = y_1 + h\cdot(-.218) = .78 + (.2)(-.218)$

$= .7364$

$y_5 = y_4 + h\cdot(.762354) = .913165 + (.2)(.762354)$ ≈ 1.06564

We approximated this IVP solution using Euler's method last time. Here's a plot:



One can show that the improved Euler method has local truncation error of order $O(h^3)$ and global truncation error $O(h^2)$.

So cutting h in half will reduce the global truncation error by a factor of 4.

Comparison with Euler's method

For a fixed step size $h < 1$, improved Euler will have a smaller global truncation error bound than Euler's method.

But! Improved Euler requires twice as many evaluations of $f(t,y)$ compared to Euler's method.

So is it really an improvement?

Consider estimating $y(1)$ in the above example.

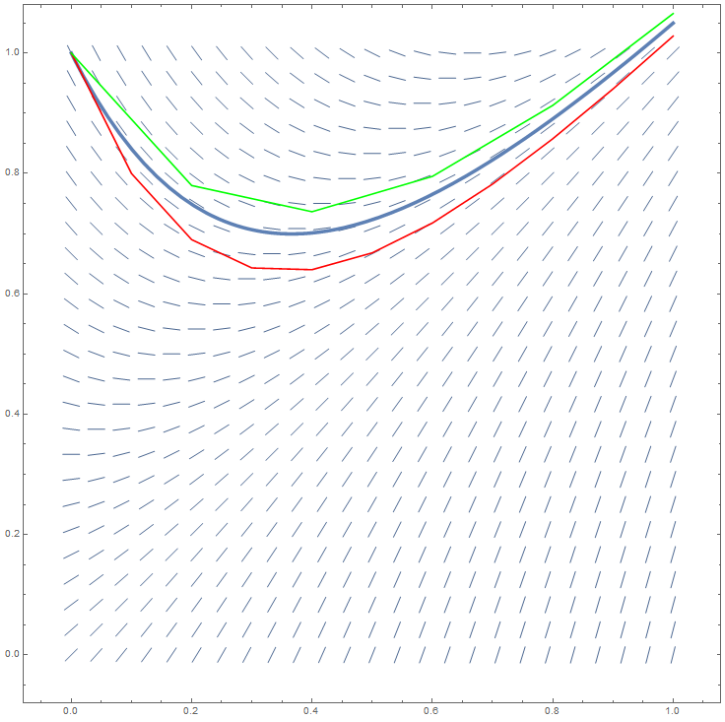
Using $h = 0.2$ with improved Euler will require 10 evaluations of f. (Five intervals with two evaluations each.)

OTOH, using a step size of $\frac{1}{2}h = 0.1$ in Euler's method also requires 10 evaluations of f.

improved Euler error : $O(h^2)$, $h^2 = 0.04$

Euler's method error : $O(\frac{1}{2}h)$, $\frac{1}{2}h = 0.1$

So the improved Euler method allows us to achieve smaller error bounds with the same number of evaluations of f (but a larger step size).



Green: improved Euler w/ $h = 0.2$

Red: Euler's method w/ $h = 0.1$

Runge-Kutta Method

These days, Euler's method and the improved Euler method are considered to be part of the "Runge-Kutta family" of numerical methods.

Our last method will be the classical Runge-Kutta method — we'll just call it the Runge-Kutta method.

The basic structure is the same as before:

given

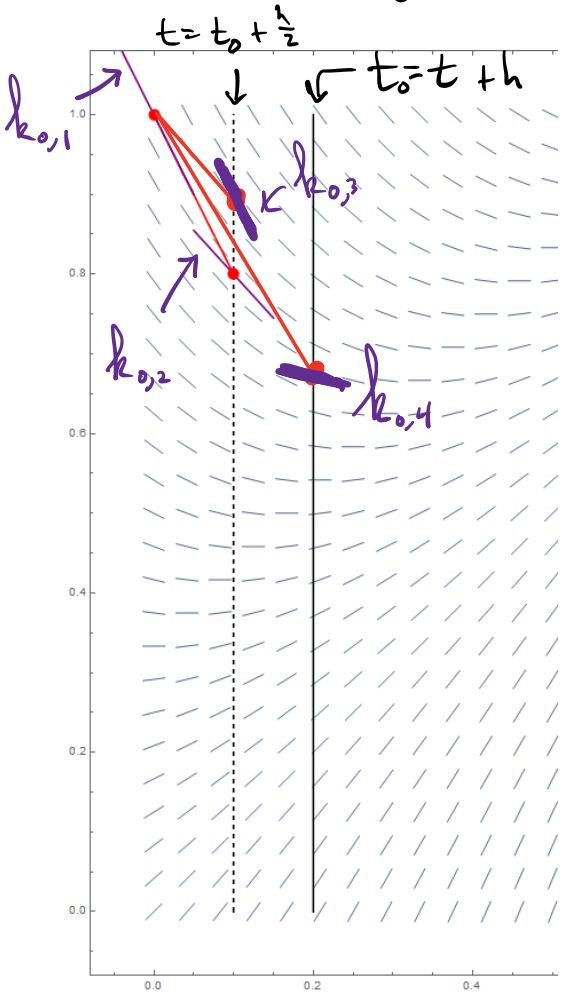
$$y' = f(t,y), \quad y(t_0) = y_0,$$

and a step size h, we'll set

$$t_1 = \underline{t_0 + h}, \quad y_1 = \underline{y_0 + h \cdot k},$$

for some slope k.

But computing k will be a bit more complicated.



The slope k will be a weighted average of four "test slopes":

$$k_{n,1} = \underline{f(t_n, y_n)}$$

$$k_{n,2} = \underline{f\left(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2} \cdot k_{n,1}\right)}$$

$$k_{n,3} = \underline{f\left(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2} \cdot k_{n,2}\right)}$$

$$k_{n,4} = \underline{f(t_n + h, y_n + h \cdot k_{n,3})}$$

Find t_{n+1}, y_{n+1} using slope

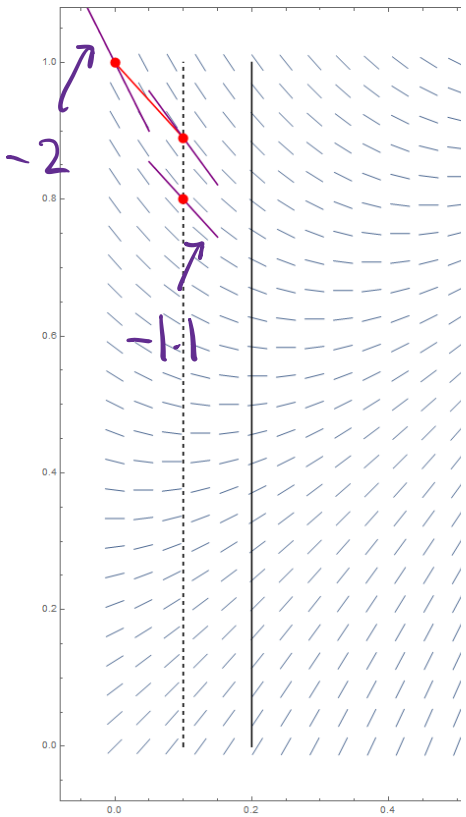
$$\frac{k_{n,1} + 2k_{n,2} + 2k_{n,3} + k_{n,4}}{6}$$

For the example $f(t,y) = 3t + 1 - 3y$,

$t_0 = 0$, $y_0 = 1$, $h = 0.2$,

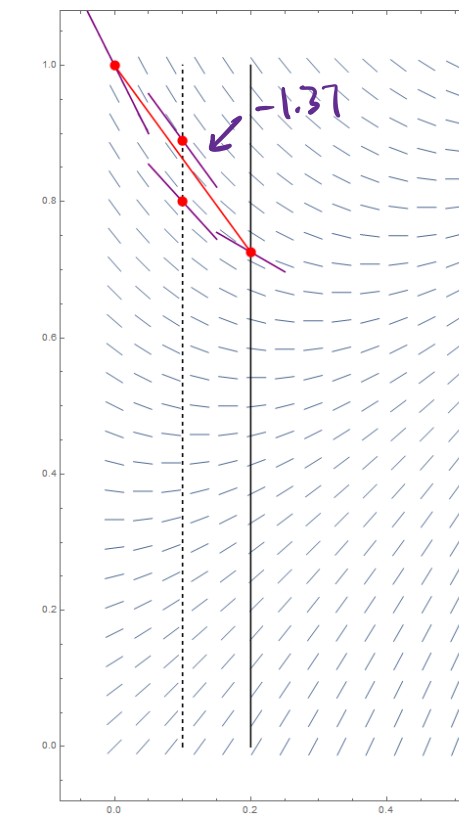
let's find

$k_{0,1}$, $k_{0,2}$, $k_{0,3}$ & $k_{0,4}$.



$$k_{0,1} = f(t_0, y_0)$$
$$= 3 \cdot 0 + 1 - 3 \cdot 1 = -2$$

$$k_{0,2} = f(t_0 + 0.1, y_0 + 0.1(-2))$$
$$= f(0.1, 0.8)$$
$$= 3(0.1) + 1 - 3(0.8) = -1.1$$



$$k_{0,3} = f(t_0 + 0.1, y_0 + 0.1(-1.1))$$
$$= f(0.1, 0.89)$$
$$= 3(0.1) + 1 - 3(.89) = -1.37$$

$$k_{0,4} = f(t_0 + 0.2, y_0 + 0.2(-1.37))$$
$$= f(0.2, 0.726)$$
$$= 3(0.2) + 1 - 3(0.726) = -0.578$$

Finally, we compute the slope k using the formula

$$k = \frac{k_{n,1} + 2k_{n,2} + 2k_{n,3} + k_{n,4}}{6}$$

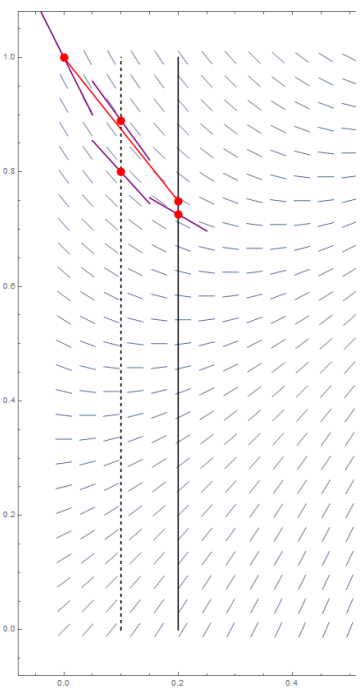
So for the example considered above,

$$k = \frac{-2 + 2(-1.1) + 2(-1.37) + (-0.578)}{6}$$

$$= -1.253$$

At last, we can compute (t_{n+1}, y_{n+1}):

$$t_{n+1} = t_n + h \quad \& \quad y_{n+1} = y_n + h \cdot k.$$



So in our example,

$$t_1 = 0.2$$

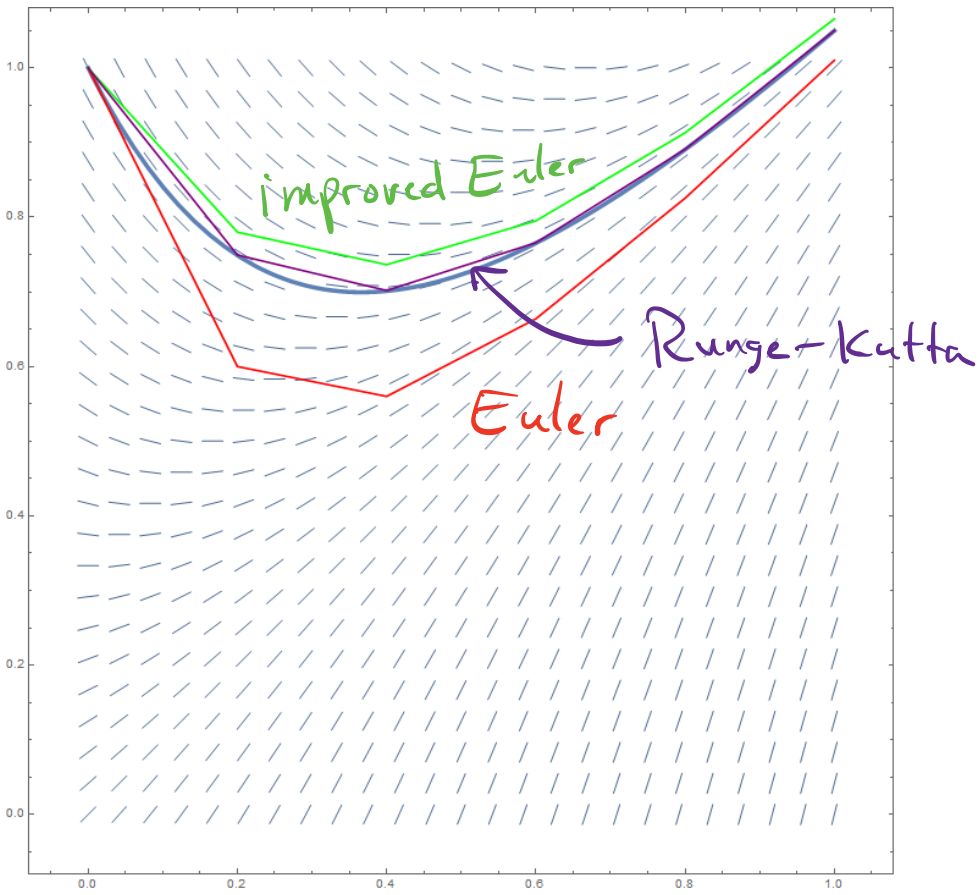
$$\& \quad y_1 = 1 + (0.2)(-1.253)$$

$$= 0.7494$$

We can ask a computer to estimate $y(t)$, $0 \le t \le 1$, using a step size of $h = 0.2$:

t_n	Euler	Improved Euler	Runge-Kutta	$y(t)$
0.	1.	1.	1.	1.
0.2	0.6	0.78	0.7494	0.748812
0.4	0.56	0.7364	0.70184	0.701194
0.6	0.664	0.795112	0.765831	0.765299
0.8	0.8256	0.913165	0.891108	0.890718
1.	1.01024	1.06564	1.05005	1.04979

Note that each of the values given by Runge-Kutta agree with $y(t)$ when rounded to the nearest hundredth.



Of course, each iteration of Runge-Kutta requires 4 evaluations of f, so it's more expensive than either of our previous methods.

But the local truncation error is $O(h^5)$ and the global truncation error is $O(h^4)$.

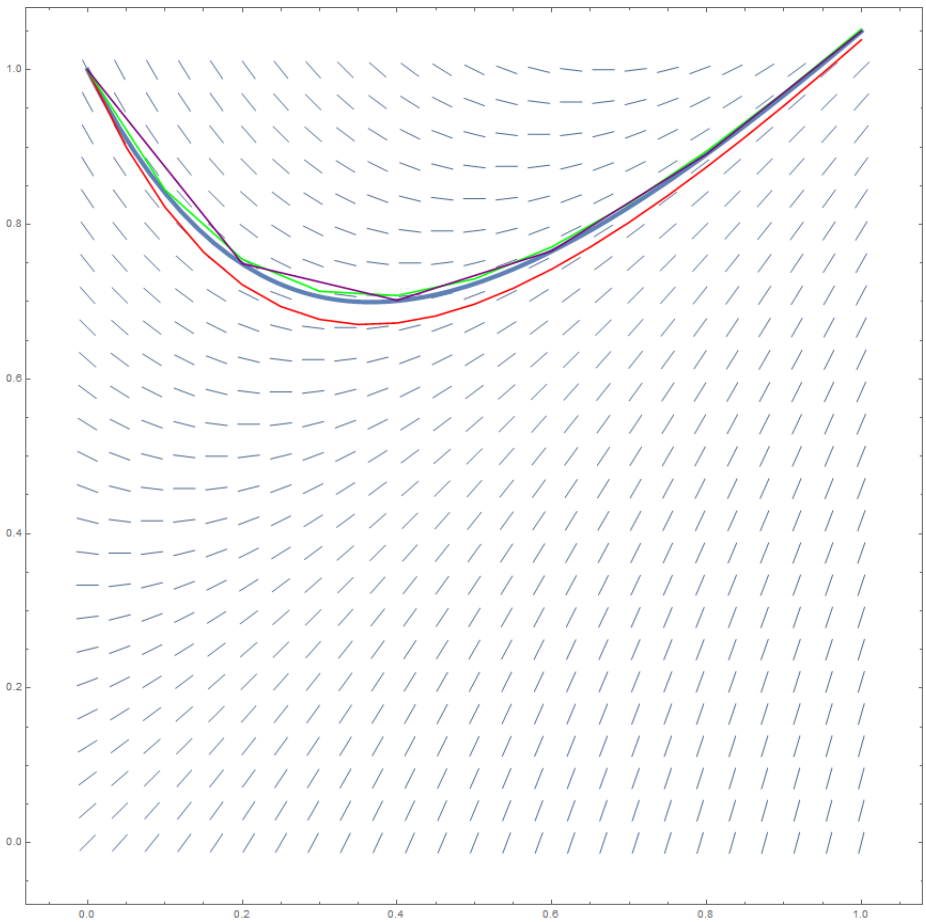
Consider using 20 evaluations of f to estimate $y(1)$ in the previous example.

For Euler's method, this means a step size of 0.05, so the global truncation error is $O(0.05)$.

Improved Euler → step size = 0.1
→ error of order $O((0.1)^2) = O(0.01)$

Runge-Kutta → step size = 0.2
→ error of order $O((0.2)^4) = O(0.0016)$

i.e., with the same number of evaluations, Runge-Kutta has a global truncation error bound which is 84.01% smaller than that of improved Euler, and way smaller than that of Euler's method.



red: Euler's w/ $h = 0.05$

green: improved Euler w/ $h = 0.1$

purple: Runge-Kutta w/ $h = 0.2$