Ordinary differential equations (ODE's). Initial Value Problem

$$y'(x) = f(x, y) \quad y(x_0) = a; \quad x \geq x_0$$

where $x \in \mathbb{R}$ and $y(x)$ is an N-dimensional function. The N-dimensional vector a is called the *Initial Condition*.

We can write this coordinate wise as

$$\begin{aligned}
[y]'_0(x) &= f_0(x, [y]_0, \ldots, [y]_{N-1}) \quad [y]_0(x_0) = a_0 \\
[y]'_1(x) &= f_1(x, [y]_0, \ldots, [y]_{N-1}) \quad [y]_1(x_0) = a_1 \\
\ldots &\quad \ldots \\
[y]'_{N-1}(x) &= f_{N-1}(x, [y]_0, \ldots, [y]_{N-1}) \quad [y]_{N-1}(x_0) = a_{N-1}
\end{aligned}$$

Notation: $[y]_n$ is the n'th coordinate of vector y, and y_n is the numerical approximation of $y(x_n)$ obtained after n steps of the numerical integration method (NR uses the same notation for these !!!).

What a numerical solution will generate:

In numerical solutions to ordinary differential equations, we first define a stepsize h and

$$\begin{aligned}
x_n &= x_0 + nh \quad n = 0, 1, 2, \ldots \\
y_n &\simeq y(x_n) \quad n = 0, 1, 2, \ldots
\end{aligned}$$

where y_n is the numerical approximation to the true value $y(x_n)$.

$$y'(x) = f(x, y) \quad y(x_0) = a \quad x \geq x_0$$

where $x \in \mathbb{R}$ and $y(x)$ is an N-dimensional function. The N-dimensional vector a is called the *Initial Condition.*

Example:

$$\begin{aligned} u'(x) &= u(x)\cos(v(x)) & u(0) = 1 \\ v'(x) &= -u(x)^3 & v(0) = \frac{\pi}{2} \end{aligned}$$

$$\begin{aligned} N &= 2 \\ x_0 &= 0 \\ a &= \begin{pmatrix} 1 \\ \frac{\pi}{2} \end{pmatrix} \equiv y_0 \\ y(x) &= \begin{pmatrix} [y]_0(x) \\ [y]_1(x) \end{pmatrix} \\ f(x,y) &= \begin{pmatrix} [y]_0(x)\cos([y]_1(x)) \\ -[y]_0(x)^3 \end{pmatrix} \end{aligned}$$

Higher order differential equations can also be written in this way:

Problems involving ordinary differential equations (ODEs) can always be reduced to the study of sets of first-order differential equations. For example the second-order equation

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x) \tag{17.0.1}$$

can be rewritten as two first-order equations,

$$\begin{aligned} \frac{dy}{dx} &= z(x) \\ \frac{dz}{dx} &= r(x) - q(x)z(x) \end{aligned} \tag{17.0.2}$$

In numerical solutions to ordinary differential equations, we first define a stepsize h and

$$y'(x) = f(x, y) \quad y(x_0) = a; \quad x \geq x_0$$

$$\begin{aligned} x_n &= x_0 + nh \quad & n = 0, 1, 2, \ldots \\ y_n &\simeq y(x_n) \quad & n = 0, 1, 2, \ldots \end{aligned}$$

where y_n is the numerical approximation to the true value $y(x_n)$.

Explicit one-step methods (Runge-Kutta methods):

$$y_{n+1} = F(x_n, h, y_n, f)$$

(the function $F(x_n, h, y_n, f)$ is often computed with some set of sequential substeps).

We will often write this as

$$y_{n+1} = F(x_n, h, y_n, f) + \mathcal{O}(h^{k+1})$$

where the last term indicates that if we would have $y_n \equiv y(x_n)$, then we would get $\|y_{n+1} - y(x_{n+1}\| = \mathcal{O}(h^{k+1})$, where k is called the *order* of the numerical method.

1st order Runge-Kutta (Euler):

$$y_{n+1} = y_n + hf(x_n, y_n) + O(h^2)$$

2nd order Runge-Kutta (Midpoint):

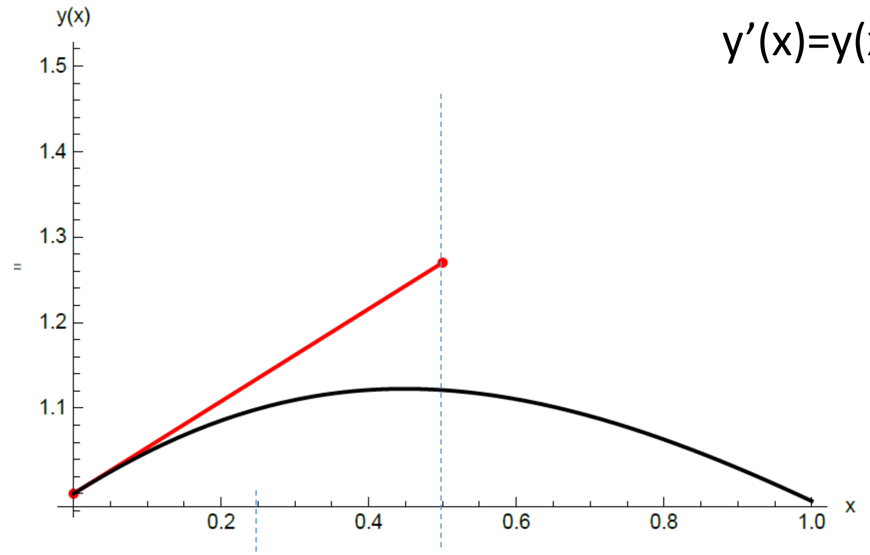
$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned}$$

$$y'(x) = f(x,y) \quad y(x_0) = a; \quad x \geq x_0$$

y'(x)=y(x)Cos(x+y(x)); y(0)=1

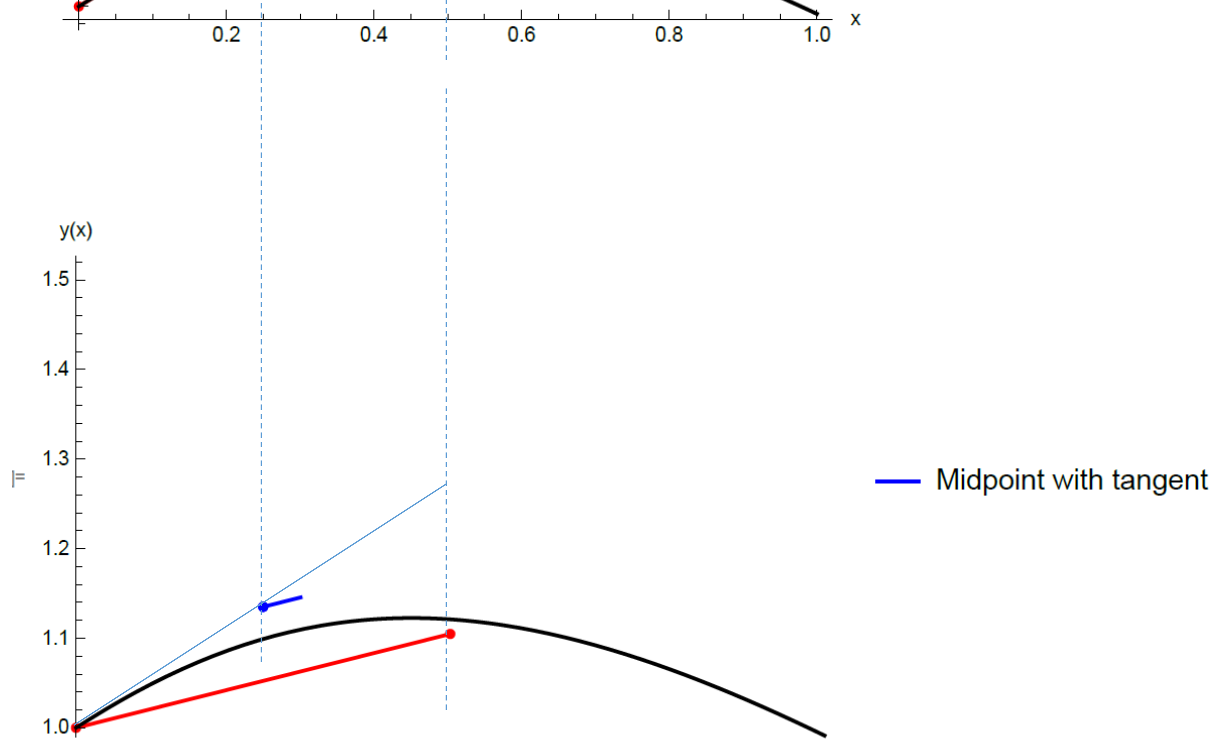
One step with 1st order Runge-Kutta (Euler).
Stepsize: h=0.5

$$y_{n+1} = y_n + h\boxed{f(x_n, y_n)} + O(h^2)$$



One step with 2nd order Runge-Kutta (Midpoint).
Stepsize: h=0.5

$$\begin{aligned} k_1 &= h\boxed{f(x_n, y_n)} \\ k_2 &= h\boxed{f\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right)} \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned}$$



4th order Runge-Kutta:

$$\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1) \\
k_3 &= hf(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_2) \\
k_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + \tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4 + O(h^5)
\end{aligned}$$

Function evaluations:

As usual, we assume that the computationally expensive part is the computation of the function f(x,y). For 1st order Runge-Kutta, we have one function evaluation per step. For 2nd order Runge-Kutta, it is two function evaluations and for 4th order Runge-Kutta, it is four function evaluations. Of course, we also seem to get higher accuracy. Our aim is to find the **most suitable method that gives the proven accuracy we want with as few function evaluations as possible.**

Example

Defines the application.
In your code: Separate the application from the method !!!

$$
\begin{aligned}
N &= 2 \\
x_0 &= 0 \\
a &= \begin{pmatrix} 1 \\ \frac{\pi}{2} \end{pmatrix} \equiv y_0 \\
y(x) &= \begin{pmatrix} [y]_0(x) \\ [y]_1(x) \end{pmatrix} \\
f(x,y) &= \begin{pmatrix} [y]_0(x)\cos([y]_1(x)) \\ -[y]_0(x)^3 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
u'(x) &= u(x)\cos(v(x)) \quad u(0) = 1 \\
v'(x) &= -u(x)^3 \qquad v(0) = \frac{\pi}{2}
\end{aligned}
$$

1st order Runge-Kutta (Euler): $y_{n+1} = y_n + hf(x_n, y_n) + O(h^2)$

We can now perform one step with Euler with stepsize h to obtain

$$
\begin{aligned}
f(x_0, y_0) &= \begin{pmatrix} 0 \\ -1 \end{pmatrix} \\
y_1 &= y_0 + hf(x_0, y_0) = \begin{pmatrix} 1 \\ \frac{\pi}{2} \end{pmatrix} + h\begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{\pi}{2} - h \end{pmatrix}
\end{aligned}
$$

$$\begin{aligned} u'(x) &= u(x)\cos(v(x)) \quad u(0)=1 \\ v'(x) &= -u(x)^3 \qquad v(0)=\frac{\pi}{2} \end{aligned}$$

$$\begin{aligned} N &= 2 \\ x_0 &= 0 \\ a &= \begin{pmatrix} 1 \\ \frac{\pi}{2} \end{pmatrix} \equiv y_0 \\ y(x) &= \begin{pmatrix} [y]_0(x) \\ [y]_1(x) \end{pmatrix} \\ f(x,y) &= \begin{pmatrix} [y]_0(x)\cos([y]_1(x)) \\ -[y]_0(x)^3 \end{pmatrix} \end{aligned}$$

2nd order Runge-Kutta (Midpoint):

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf\left(x_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}k_1\right) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned}$$

If we do one step with the midpoint method (2nd order Runge-Kutta), we get

$$\begin{aligned} f(x_0,y_0) &= \begin{pmatrix} 0 \\ -1 \end{pmatrix} \\ k_1 &= hf(x_0,y_0) = h\begin{pmatrix} 0 \\ -1 \end{pmatrix} \\ y_0 + \frac{1}{2}k_1 &= \begin{pmatrix} 1 \\ \frac{\pi}{2} \end{pmatrix} + \frac{h}{2}\begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{\pi}{2} - \frac{h}{2} \end{pmatrix} \\ k_2 &= hf(x_0 + \frac{h}{2}, y_0 + \frac{1}{2}k_1) = h\begin{pmatrix} \cos(\frac{\pi}{2} - \frac{h}{2}) \\ -1 \end{pmatrix} \\ y_1 &= y_0 + k_2 = \begin{pmatrix} 1 \\ \frac{\pi}{2} \end{pmatrix} + h\begin{pmatrix} \cos(\frac{\pi}{2} - \frac{h}{2}) \\ -1 \end{pmatrix} = \begin{pmatrix} 1 + h\cos(\frac{\pi}{2} - \frac{h}{2}) \\ \frac{\pi}{2} - h \end{pmatrix} \end{aligned}$$

Order of a numerical method (global order):

$$y_{n+1} = F(x_n, h, y_n, f) + \mathcal{O}(h^{k+1})$$

where the last term indicates that if we would have $y_n \equiv y(x_n)$, then we would get $\|y_{n+1} - y(x_{n+1})\| = \mathcal{O}(h^{k+1})$, where k is called the *order* of the numerical method.

In applications, we will not be interested in the error after one step, but the error at some fixed x. If we use a a number of subdivisions

$$x_n = x_0 + nh \quad n = 0, \dots, M$$

so that $x = Mh$, we get a first approximation of the error at x as

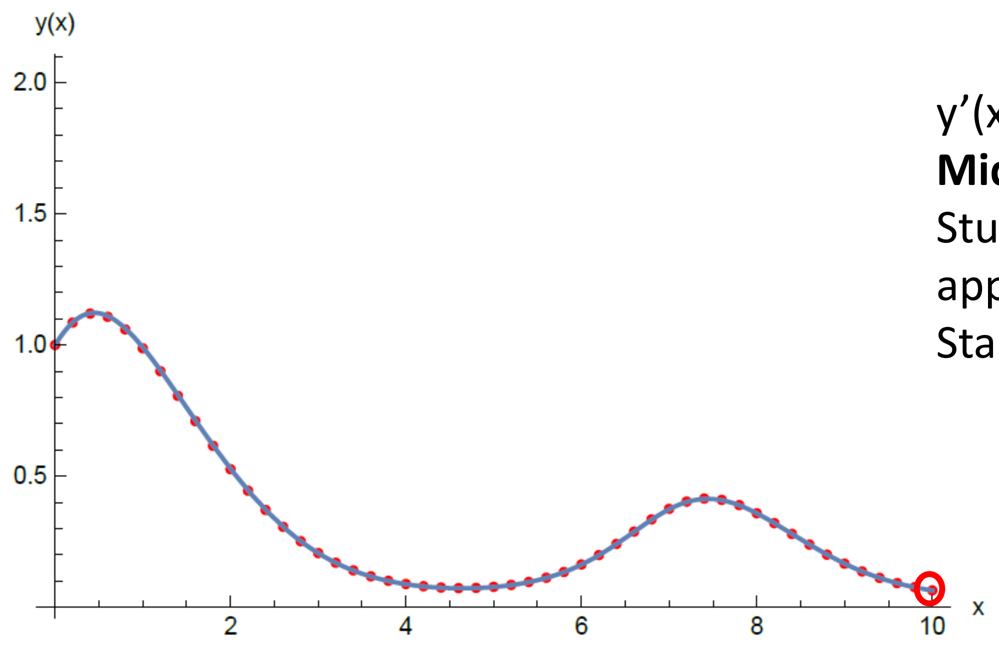
$$\|y_M - y(x)\| \simeq M * \mathcal{O}(h^{k+1})$$

Since $M = \frac{x}{h}$, we expect to get

$$\|y_M - y(x)\| \simeq \mathcal{O}(h^k)$$

which is why k (and not $k+1$) is called the order of the method.

The term for one step $\mathcal{O}(h^{k+1})$ is called the ”local order” of the method



$y'(x)=y(x)\mathrm{Cos}(x+y(x))$; $y(0)=1$

Midpoint method.

Studying the error on the numerical approximation to y(10).

Start with h=1. Then h=0.5; 0.25;…

i	A(hi)	A(hi-1)-A(hi)	Rich-alp^k	A(hi)-A	Rich-error	f-computations
1.	0.0569798	*	*	-0.00736924	*	20.
2.	0.0665769	-0.00959713	*	0.00222789	*	60.
3.	0.0649463	0.0016306	-5.88566	0.000597297	0.000543532	140.
4.	0.0644924	0.000453938	3.59211	0.000143359	0.000151313	300.
5.	0.0643838	0.000108616	4.17929	0.0000347431	0.0000362054	620.
6.	0.0643576	0.0000261993	4.14576	8.54381×10^{-6}	8.73311×10^{-6}	1260.

Implicit one-step methods:

$$y_{n+1} = F(x, h, y_n, \boxed{y_{n+1}}, f)$$

Here only the Trapezoidal method:

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right) + \mathcal{O}(h^3)$$

Implementation:

Perform an Euler step

$$y^*_{n+1} = y_n + hf(x_n, y_n)$$

Then use Newtons method to solve the system of potentially non-linear equations

$$\phi(y) = y - y_n - \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y)\right) = 0$$

using $y = y^*_{n+1}$ as the initial guess and choose then $y_{n+1} = y$. One or at most two iterations will usually do to obtain an error that is negligible compared to the discretization error. For Newtons method, we obtain the Jacobian

$$J(y) = I - \frac{h}{2} J_f(y)$$

where J_f is the Jacobian wrt. y of $f(x_{n+1}, y)$.

$$\begin{aligned} u'(x) &= u(x)\cos(3x+v(x)) \quad u(0)=1 \\ v'(x) &= x^2 - u(x)^3 \qquad v(0) = \frac{\pi}{2} \end{aligned}$$

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n,y_n) + f(x_{n+1},y_{n+1})\right) + \mathcal{O}(h^3)$$

We obtained with Euler's method

$$\begin{aligned} f(x_0,y_0) &= \begin{pmatrix} 0 \\ -1 \end{pmatrix} \\ y_1 &= \begin{pmatrix} 1 \\ \frac{\pi}{2} - h \end{pmatrix} \end{aligned}$$

$$\phi(y) = \begin{pmatrix} [y]_0 \\ [y]_1 \end{pmatrix} - \begin{pmatrix} 1 \\ \frac{\pi}{2} \end{pmatrix} - \frac{h}{2}\left[\begin{pmatrix} 0 \\ -1 \end{pmatrix} + \begin{pmatrix} [y]_0\cos(3h+[y]_1) \\ h^2 - [y]_0^3 \end{pmatrix}\right]$$

$$J(y) = \begin{pmatrix} 1 - \frac{h}{2}\cos(3h+[y]_1) & \frac{h}{2}[y]_0\sin(3h+[y]_1) \\ \frac{h}{2}3[y]_0^2 & 1 \end{pmatrix}$$

Perform an Euler step

$$y^*_{n+1} = y_n + hf(x_n,y_n)$$

Then use Newtons method to solve the system of potentially non-linear equations

$$\phi(y) = y - y_n - \frac{h}{2}\left(f(x_n,y_n) + f(x_{n+1},y)\right) = 0$$

using $y = y^*_{n+1}$ as the initial guess and choose then $y_{n+1} = y$. One or at most two iterations will usually do to obtain an error that is negligible compared to the discretization error. For Newtons method, we obtain the Jacobian

$$J(y) = I - \frac{h}{2}J_f(y)$$

where J_f is the Jacobian wrt. y of $f(x_{n+1},y)$.

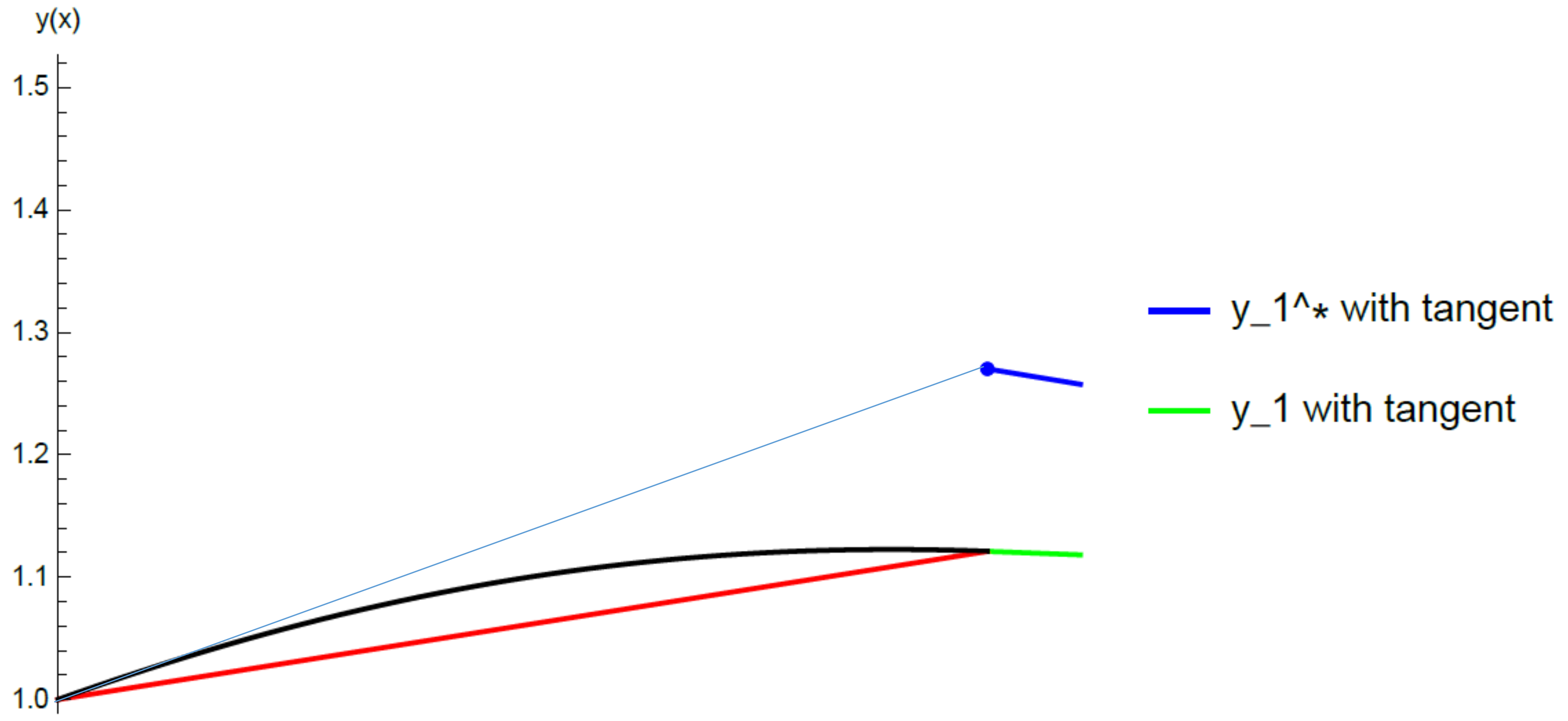
$$\begin{aligned} N &= 2 \\ x_0 &= 0 \\ a &= \begin{pmatrix} 1 \\ \frac{\pi}{2} \end{pmatrix} \equiv y_0 \\ y(x) &= \begin{pmatrix} [y]_0(x) \\ [y]_1(x) \end{pmatrix} \\ f(x,y) &= \begin{pmatrix} [y]_0(x)\cos(3x+[y]_1(x)) \\ x^2 - [y]_0(x)^3 \end{pmatrix} \\ \phi(y) &= y - y_n - \frac{h}{2}\left(f(x_n,y_n) + f(x_{n+1},y)\right) \\ J(y) &= \begin{pmatrix} 1 - \frac{h}{2}\cos(3x+[y]_1) & \frac{h}{2}[y]_0\sin(3x+[y]_1) \\ \frac{h}{2}3[y]_0^2 & 1 \end{pmatrix} \end{aligned}$$

We are then ready to use $y = y_1$ as the initial guess for solving $\phi(y) = 0$ with Newtons method.

y’(x)=y(x)Cos(x+y(x)); y(0)=1

Trapezoidal method.

$$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right) + \mathcal{O}(h^3)$$



Two-step Leap-frog method:

$$y_{n+1} = y_{n-1} + 2hf(x_n, y_n) + \mathcal{O}(h^3)$$

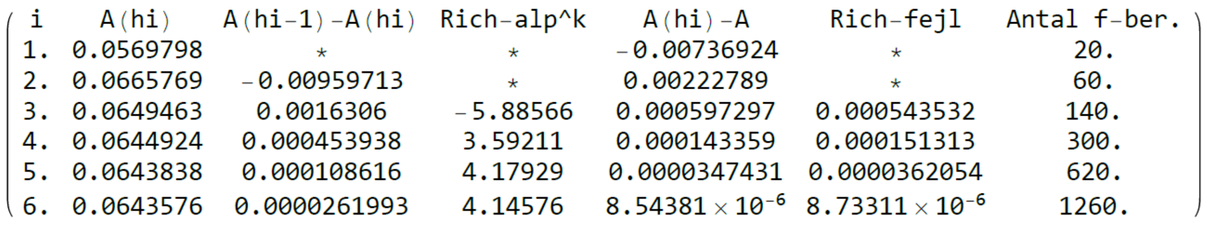
Do an Euler step first to initiate the method with y_0 and y_1.

Exercise:

u'(x)=u(x)v(x)	u(0)=1
v'(x)= -u(x)^2	v(0)=1

Solve with Euler, Midpoint, Trapezoidal, Leap-frog and 4th order Runge-Kutta
Consider x=10 and estimate the order using Richardson
Subdivide h with 2 until you reach an accuracy on u(x) of 10^-6
Establish a table like on slide 9 (of course without the A(h)-A column)

Table from slide 9:



i	A(hi)	A(hi-1)-A(hi)	Rich-alp^k	A(hi)-A	Rich-fejl	Antal f-ber.
1.	0.0569798	*	*	-0.00736924	*	20.
2.	0.0665769	-0.00959713	*	0.00222789	*	60.
3.	0.0649463	0.0016306	-5.88566	0.000597297	0.000543532	140.
4.	0.0644924	0.000453938	3.59211	0.000143359	0.000151313	300.
5.	0.0643838	0.000108616	4.17929	0.0000347431	0.0000362054	620.
6.	0.0643576	0.0000261993	4.14576	8.54381×10^{-6}	8.73311×10^{-6}	1260.