Problem 3 **Numerical methods for ODEs [10 pts]** Consider python code that solves an initial value problem with a Runge-Kutta method and prints the resulting x and y value after N steps:

```
1  def f(x, y):
2    return x * y**2
3
4  def step(x, y, h):
5    k1 = f(x, y)
6    k2 = f(x + h, y + h * k1)
7    y_new = y + (h / 2) * (k1 + k2)
8    x_new=x+h
9    return x_new, y_new
10
11 x = 0
12 y = 0.5
13 h = 0.1
14 N=20
15
16 for n in range(N):
17   x,y=step(x,y,h)
18
19 print(x,y)
```

$f(t,y) = ty^2$

$x = t$ her

a) What is the initial value problem that the code solves?

b) Write the Butcher tableau for the method used. Is the method explicit or implicit? motivate your answer.

a) $y' = f(t,y) = ty^2$, $y(0) = 0,5$

b) RK med s steg

$$u_{n+1} = u_n + h\sum_{i=1}^{s} b_i K_i$$

$$K_i = f\left(t_n + c_i h, u_n + h\sum_{j=1}^{s} a_{ij} K_j\right)$$

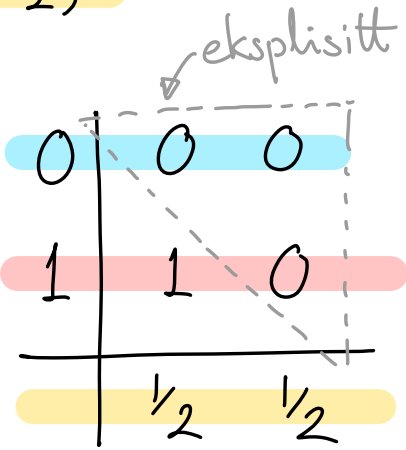
$$\begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b}^T \end{array}$$

2 steg

$$\begin{cases} K_1 = f(t_n, u_n) \\ K_2 = f(t_n + h, u_n + hK_1) \end{cases}$$

$$u_{n+1} = u_n + \frac{h}{2}(K_1 + K_2)$$

Butcher tablå



$a_{ij} = 0$ for $j \geq i$, så metoden er eksplisitt.

Oppgave 8

a) Gitt initialverdiproblemet

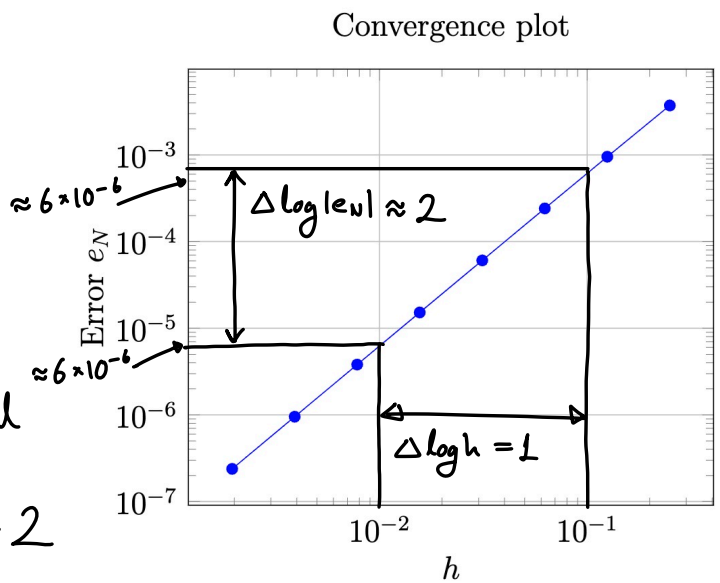
$$y' = \sqrt{y}, \quad y(0) = 1.$$

Skriv ned en *fullstending* algoritme for å finne en tilnærmelse til $y(2)$ ved bruk av implisitt (baklengs) Eulers metode, med steglengde $h = 2/N$.

Utfør et steg med algoritmen med $h = 0.1$, dvs. finn en tilnærmelse til $y(0.1)$.

NB! Algoritmen må gjerne skrives i form av kode i f.eks. MATLAB eller Python. Den skal være tilstrekkelig detaljert til at den kan implementeres.

b) Vi antar nå at ligningen over løses med en ikke oppgitt metode. Feilen $e_N = |y(2) - y_N|$ er målt for ulike skrittlengder $h = 2/N$, og resultatet er presentert i følgende konvergensplott:



stigningstall

$$\frac{\Delta \log|e_N|}{\Delta \log h} \approx 2$$

Hva mener vi med en metodes orden, og hvordan kan ordenen leses av et konvergensplott som dette?

Hva er denne metodens orden?

a) Baklengs Euler: $u_{n+1} = u_n + hf(t_{n+1}, u_{n+1})$

$y' = f(t,y) = \sqrt{y}$, estimer $y(2)$

- Kode

```
import math
N = 100   <- f.eks.
h = 2/N
u = 1
```

Alternativ 1) bruk en rot-finner

$u_{n+1} = u_n + h\sqrt{u_{n+1}} \rightarrow g(x,s) = x - s - h\sqrt{x}$

($x \leftarrow u_{n+1}$, $s \leftarrow u_n$)

```
def g(x, s):
    return x - s - h * math.sqrt(x)

for i in range(N):
    u = "løs g(x,u) = 0 m.h.p. x"
```

(gjetning)

```
(u = scipy.optimize.fsolve(g, u, u)
                           s
```

```
print(u)
```

{Alternativ 2) eksplisitt formel for u_{n+1}

```
for i in range(N):
    u = u +(h**2 + math.sqrt(4*u*h**2 + h**4))/2
print(u)
```

- Utledning av formel for u_{n+1}:

$$u_{n+1} = u_n + h\sqrt{u_{n+1}}$$

$$(h\sqrt{u_{n+1}})^2 = (u_{n+1} - u_n)^2$$

$$\vdots$$

$$u_{n+1}^2 - (2u_n + h^2)u_{n+1} + u_n^2 = 0$$

$$u_{n+1} = \frac{2u_n + h^2 \pm \sqrt{(2u_n + h^2)^2 - 4u_n^2}}{2}$$

$$= u_n + \frac{1}{2}h^2 \pm \frac{1}{2}\underbrace{\sqrt{4u_n h^2 + h^4}}_{\geq \sqrt{h^4} = h^2}$$

$u_{n+1} - u_n < 0$ for $-$, og ≥ 0 for $+$.

Vi vil ha $y' = \sqrt{y} \geq 0$, så vi velger $+$.

- Ett steg med $h = 0{,}1$

$u_0 = 1$ $\qquad u_0 = 1$

$$u_1 = u_0 + \frac{1}{2}\left(h^2 + \sqrt{4u_0h^2 + h^4}\right)$$

$$\approx 1{,}1051$$

b) • Metodens orden er det største heltallet p slik at

$$|e_N| = |y(2) - u_N| \leq C h^p$$

for en konstant C og $h > 0$.

- For små h er typisk $|e_N| \approx C h^p$

$$\Rightarrow \log|e_N| \approx \log(Ch^p) = \log C + p \cdot \log h$$

$\rightarrow$ linear graf i log-log plot

Metodens orden kan dermed leses av som stigningstallet til $\log|e_N|$ som en funksjon av $\log h$.

- Grafen har stigningstall 2, så metodens orden er 2.