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 $h = \mathbf{2}/N$.

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.

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

Approksimer $y(2)$ med steglengde $h = \frac{2}{N}$

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

- Kode:

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

<- oppgaven spør kun om $y(2)$, så vi trenger ikke å lagre u_n for hver n

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

print(u) ← tilnærming til y(2)
```

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

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

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

$$\vdots$$

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

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

$$= u_n + \frac{1}{2}h^2 \pm \frac{1}{2}\sqrt{4u_nh^2 + h^4}$$

$$\geq \sqrt{h^4} = h^2$$

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

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

Alternativ: bruk en rot-finner på

```
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"
```

f.eks.

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

(gjetning ↖)

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

$u_0 = 1$

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

$\approx 1{,}1051$

Problem 3. (Ralston's method, 14 points)

For the ordinary differential equation

$$y'(t) = -6y(t), \quad \text{with} \quad y(0) = 1,$$

consider *Ralston's method* given by the following Butcher tableau:

0	0	0
2/3	2/3	0
	1/4	3/4

Using the tableau and expanding the stage derivatives k_i, we can write the solution y_{n+1} in terms of the previous one, y_n, and of the time-step size $h > 0$. More precisely:

$$y_{n+1} = R(h)\, y_n, \quad \text{so that} \quad y_n = [R(h)]^n\, y(0),$$

in which $R(h)$ is a second-degree polynomial.

a) How many stages does this Runge–Kutta method have?

b) Determine the polynomial $R(h)$.

RK med s steg

$$\begin{cases} u_{n+1} = u_n + h\sum_{i=1}^{s} b_i K_i \\ K_i = f\left(t_n + c_i h,\ u_n + h\sum_{j=1}^{s} a_{ij} K_j\right) \end{cases}$$

$\vec{c}$	A
	$\vec{b}^T$

a) 2 steg (fordi tabellen har 2 rader / kolonner)

b)

0	0	0
$2/3$	$2/3$	0
	$1/4$	$3/4$

$f(t,y) = -6y$

$$K_1 = f(t_n + 0, u_n + 0) = f(t_n, u_n) = -6u_n$$

$$K_2 = f(t_n + \tfrac{2}{3}h, u_n + h\tfrac{2}{3}K_1) = -6(u_n + h\tfrac{2}{3}K_1) = -6(u_n - \tfrac{2}{3}h\,6u_n) = -6u_n(1 - 4h)$$

$$u_{n+1} = u_n + h\tfrac{1}{4}K_1 + h\tfrac{3}{4}K_2$$

$$= \cdots = \underbrace{(1 - 6h + 18h^2)}_{R(h)}\,u_n$$

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:

```
def f(x, y):
  return x * y**2

def step(x, y, h):
  k1 = f(x, y)
  k2 = f(x + h, y + h * k1)
  y_new = y + (h / 2) * (k1 + k2)
  x_new=x+h
  return x_new, y_new

x = 0
y = 0.5
h = 0.1
N=20

for n in range(N):
  x,y=step(x,y,h)

print(x,y)
```

$x = t$ 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) $y' = f(t,y) = ty^2$ med $y(0) = 0{,}5$

b) RK med s steg

$$\begin{cases} u_{n+1} = u_n + h\sum_{i=1}^{s} b_i K_i \\ K_i = f\left(t_n + c_i h,\ u_n + h\sum_{j=1}^{s} a_{ij} K_j\right) \end{cases}$$

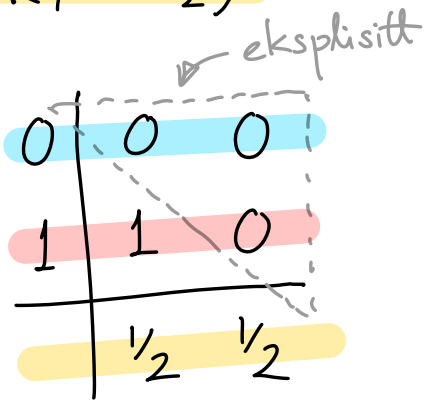
$$\begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b}^T \end{array}$$

2 steg

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

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

Butcher tablå:



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