Exercise 5

Part 1

Task 1

1.) $x'' + x' + 4x = 0 \quad \Rightarrow \quad x'' = -x' - 4x$

$x_1(t) = x(t) \qquad x_2 = x'(t)$

$$\Rightarrow \quad \begin{bmatrix} x_1'(t) \\ x_2'(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -x_2 - 4x_1 \end{bmatrix}$$

2.) $x''' - 3x^2 x'' + x' - 7x = 0$

$\Rightarrow \quad x''' = 3x^2 x'' - x' + 7x$

$x_1(t) = x(t) \qquad x_2(t) = x'(t) \qquad x_3 = x''(t)$

$$\begin{bmatrix} x_1'(t) \\ x_2'(t) \\ x_3'(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ x_3(t) \\ 3x_1^2 x_3 - x_2 + 7x_1 \end{bmatrix}$$

Task 2

1)

a) The code from PST5_1.py :

```
#a)

A=np.array([[-1, -3],[-2, 1]])
eigenval, eigenmat = la.eig(A)

print(eigenval)
print(eigenmat)
```

gives :

```
eigenvalues= [-2.64575131  2.64575131]
eigenvectors[[-0.8767397   0.6354064 ]
 [-0.48096517 -0.7721779 ]]
```

we see that $\lambda_i > 0$ for $i = 2$

$\Rightarrow$ not stable

code from same file:

```
xb = np.array([1.0,0.0])
xc = np.array([-1.0,0.0])
def f(x):
    x1 = x[0]
    x2 = x[1]
    return np.array([-0.5*x1**2 + x2**2 , -x2*x1])

jac = jacobian(f)

eigenvalb, eigenmatb = la.eig(jac(xb))
eigenvalc, eigenmatc = la.eig(jac(xc))
```

gives

B - stable

```
eigenvalues= [-1. -1.]
eigenvectors=[[1. 0.]
 [0. 1.]]
eigenvalues c= [1. 1.]
eigenvectorsc=[[1. 0.]
 [0. 1.]]
```

C unstable

d)

the stability of a non linear system is dependent on the point the system is linearize around

Part 2

1) defined in the code "PST5_2.py"

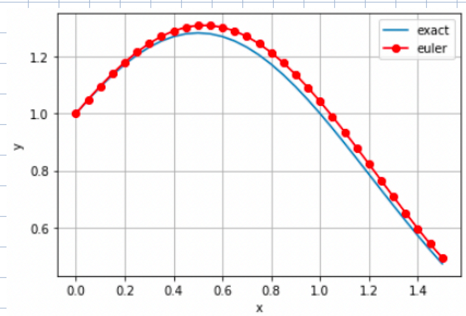
```python
import numpy as np
import matplotlib.pyplot as plt

def forwardeuler(f, h, x0, t0, tf):

    steps = int(np.ceil((tf - t0) / h))

    vt_k = np.empty(steps + 1)
    vx_k = np.empty((steps + 1, len(x0)))

    vt_k[0] = t0
    vx_k[0] = x0


    for k in range(steps):
        tk = t0 + h*k
        xk = vx_k[k]

        vx_k[k+1] = xk + h * f(tk, xk)
        vt_k[k + 1] = tk + h

    vt_k[-1] = tf
    vx_k[-1] = xk + h * f(vt_k[-1], vx_k[-1])


    return (vt_k, vx_k)
```

2)

added test:

```python
def fn_gh(t,x):
    return (1-2*t)*x

def exact(t):
    return np.exp(1/4 - (1/2 - t)**2)

### define stating conditions

h = 0.05
t0 = 0.0
tf = 1.5
x0 = np.array([1.0])

t, x = forwardeuler(fn_gh, h, x0, t0, tf)
plt.plot(t,exact(t), label = "exact")
plt.plot(t,x, "ro-", label= 'euler')
plt.legend()
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
```

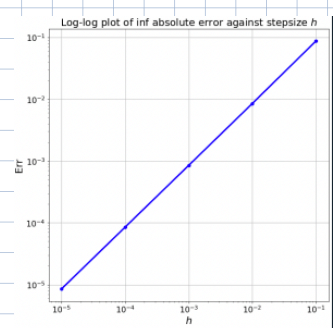
save plot



3) The code PST5_2_2.py

```python
import numpy as np
import matplotlib.pyplot as plt
from PST5_2 import *
import autograd.numpy as np


t0 = np.float64(0.0)
tf = np.float64(1.5)
x0 = np.array([ 1.0 ], dtype=np.float64)

Nh = 5
vh = 10**np.linspace(-1, -5, Nh)
err_h = np.zeros(Nh, dtype=np.float64)

for i in range(0, Nh):
    (vt_k, vx_k) = forwardeuler(fn_gh, vh[i], x0, t0, tf)
    exact_sol = np.exp(1/4 - (1/2 - vt_k)**2)
    err_h[i] = np.linalg.norm(exact_sol - vx_k[:,0], np.inf)


fig, ax = plt.subplots(1, 1, figsize=(10, 10))
ax.loglog(vh, err_h, 'bo-', linewidth=3)
ax.grid()
ax.set_title(r"Log-log plot of inf absolute error against stepsize $h$", fontsize=20)
ax.set_xlabel(r'$h$', fontsize=20)
ax.set_ylabel(r'Err', fontsize=20)
ax.tick_params(axis='x', labelsize=16)
ax.tick_params(axis='y', labelsize=16)
```

save this



This shows that this system the error is proportional with h'

Part 2.2

1) $$\frac{dU}{dt} = h_{in} - h_{out} + Q$$

$$dH - \cancel{P\Delta V} = h_{in} - h_{out} + Q$$

$$\Rightarrow \quad dH = h_{in} - h_{out} + Q \qquad dH = c_p dT$$

$$\Rightarrow \quad c_p dT = h_{in} - h_{out} + Q$$

$$\Rightarrow \quad c_p dT = F c_p (T_{in} - T^\circ) - F c_p (T_{out} - T^\circ) + Q$$

$$\Rightarrow \quad V\rho c_p dT = F c_p (T_{in} - T_{out}) + Q$$

$$T_{in} = u_2 \qquad T_{out} = T = y \qquad u_1 = Q$$

$$\Rightarrow \quad V\rho c_p \, dy = F c_p (u_2 - y) + u_1$$

$$= F c_p u_2 - F c_p y + u_1$$

$$\frac{V\rho c_p \, dy}{F c_p} = -y + \frac{1}{F c_p} u_1 + u_2$$

$$\Rightarrow \quad k_2 = 1$$

$$k_1 = \frac{1}{F c_p} = \frac{1}{10 \frac{kg}{s} \cdot 4.2 \frac{kJ}{kg K}} \approx 0.024 \frac{kJ}{s K}$$

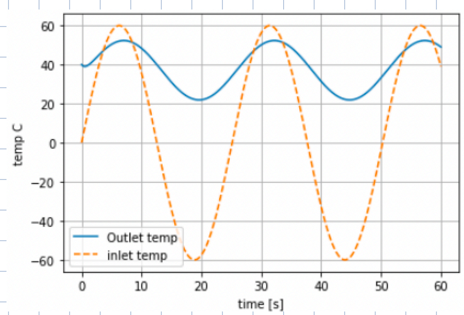
$$\frac{V\rho}{F} = \tau = \frac{30 L \cdot 1 \frac{kg}{L}}{10 \, kg/s} = 3 s$$

2) a) $\tau \frac{dy}{dt} = -y + k_1 \cdot k_3 (T_s - y) + k_2 (30 \cdot 2 \sin(\frac{t}{4}))$

$$\Rightarrow y' = \frac{-y + k_1 \cdot k_3 (T_s - y) + k_2 (30 \cdot 2 \sin(\frac{t}{4}))}{\tau}$$

The code PST5_Simulation.py gives this plot:

```python
import numpy as np
import matplotlib.pyplot as plt
from PST5_2 import *
import autograd.numpy as np


#def starting conditions
t0 = np.float64(0.0)
tf = np.float64(60.0)
x0 = np.array([ 40 ], dtype=np.float64)
h = 0.05

# def konstants
tau = 3
k1 = 0.024
k2 = 1
k3 = 120
Ts = 50


#define function

def f(t,y):
    return (-y+k1*k3*(Ts-y)+k2*(30*2*np.sin(t/4)))/tau


t, x = forwardeuler(f, h, x0, t0, tf)

plt.plot(t,x, label='Outlet temp')
plt.plot(t, 30*2*np.sin(t/4),"--", label="inlet temp")
plt.xlabel("time [s]")
plt.ylabel("temp C")
plt.legend()
plt.grid()
```



b) higher flow rate will decrease τ which will increase the change in temperature per second