Answers: Exercise Sheet No. 5
ODE Pt. I.

1 Basic theory and manipulations

1. Reduction of order

   (a) Let $y = x',\ y' = x''$

   $$\begin{aligned} y' &= -x' - 4x \\ x' &= y \end{aligned}$$

   (b) Let $z = x'', z' = x''', y = x',\ y' = z,$

   $$\begin{aligned} z' &= 3x^2 z - y + 7x \\ y' &= z \\ x' &= y \end{aligned}$$

2. Linear stability

   (a) The system has a positive eigenvalues so the system is unstable stable.

   (b) The system has positive eigenvalues so the system is locally unstable.

   (c) The system has negative eigenvalues so is locally stable.

   (d) Nonlinear systems can be stable, unstable, etc. at different points in the state-space. Thus, the requirements on the solver may change in at different points.

2 Euler

2.1 Coding Forward Euler

The code for part 1 and 2 is below.

```
# -*- coding: utf-8 -*-
"""
@author: peterma
"""

import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt



def forwardeuler(f, h, x0, t0, tf):
    # Setup arrays
    Nk = np.int_(np.ceil(1+(tf-t0)/h))
    vt_k = np.zeros(Nk, dtype=np.float64)
    vx_k = np.zeros((Nk, x0.size), dtype=np.float64)
    for i in range(0, Nk):
        vt_k[i] = t0 + h*i
        if vt_k[i] > tf: vt_k[i] = tf
```

```

    # Do integration
    vx_k[0][:] = x0[:]
    for i in range(1, Nk):
        h = vt_k[i] - vt_k[i-1]
        vx_k[i] = vx_k[i-1] + h*f(vt_k[i-1], vx_k[i-1])

    return (vt_k, vx_k)



def fn_gh(t, x):
    return np.array([ (1-2*t)*x[0] ])




t0 = np.float64(0.0)
tf = np.float64(1.5)
x0 = np.array([ 1.0 ], dtype=np.float64)
h = 0.05
(vt_k, vx_k) = forwardeuler(fn_gh, h, x0, t0, tf)


fig, ax = plt.subplots(1, 1, figsize=(10, 10))
ax.grid()
exact_sol = np.exp(1/4 - (1/2 - vt_k)**2)
ax.plot(vt_k, exact_sol, 'b-', linewidth=3, label='Exact Sol')
ax.plot(vt_k, vx_k[:,0], 'r--o', linewidth=3, markersize=10, label=
    'Euler')
ax.legend(fontsize=18)

ax.set_title(f'Euler and exact solution for $x'' = (1-2t)x, x(0)=1$
    ', fontsize=20)
ax.set_xlabel(r'$x$', fontsize=20)
ax.set_ylabel(r'$t$', fontsize=20)
ax.tick_params(axis='both', labelsize=16)
```

(3) The global truncation order is first order $O(h)$, which you can argue from the figure (see slides for theoretical justification).

2.2 Heater example

1. $\tau = \frac{\rho V}{F}$, $k_2 = 1$, $k_1 = \frac{1}{F c_P}$

2.
```
# -*- coding: utf-8 -*-
"""
@author: emturan
"""

import numpy as np
import matplotlib.pyplot as plt
rho = 1
V = 30
F = 10
tau = rho*V/F
k1 = 1/(F*4.2)
k3= 120
Ts=50
def heater(t,y):
    return (1/tau)(30+2*np.sin(t/4)-y+(k1)*(k3*(50-y)))
```

```

y0 = [40]
tspan = [0.,60]
sol = solve_ivp(fun, tspan, y0, method='LSODA', atol=1e-6)
plt.plot(sol.t, sol.y[0,:])
plt.show()
```

3. If the flow rate increases than the time constant will decrease , and the gain k_1 will decrease. Material will spend less time in the vessel, and the cyclic steady state offset from the set-point will be larger, etc..