HW_1

September 8, 2022

[1]:
```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import display, Latex, Math
```

Group members
Alexander Stoustrup
Mathias Tyranski
Benjamin Simonsen

1 E2.4

1.1 a)

[2]:
```
def eq_disp(varstring, expr):
    display(Latex(f"${varstring}={sp.latex(expr)}$"))
s, t = sp.symbols('s, t')
R = 1/s
```

The laplace transform of a unit step function is

$$F(s) = \frac{1}{s}$$

The output is given by:

$$Y(s) = G(s)R(s)$$

The transfer function $G(s)$ is given by

[3]:
```
G = sp.factor(4*(s + 50)/(s**2 + 30*s + 200))
p, q = G.as_numer_denom()
G
```

[3]:
$$\frac{4\,(s+50)}{(s+10)\,(s+20)}$$

To find $Y(s)$ we need the partial fraction expansion of $G(s)R(s)$

We can construct the partial fractions according to:

$$G(s)R(s) = \frac{K_{s1}}{s+s_1} + \frac{K_{s2}}{s+s_2} + \cdots + \frac{K_{sn}}{s+s_n}$$

First solve for the n poles

[4]:
```
Y = G*R
p, q = Y.as_numer_denom()
poles = sp.solve(q,s)
for i, pole in enumerate(poles):
    eq_disp(f'-s_{i}', pole)
```

$-s_0 = -20$

$-s_1 = -10$

$-s_2 = 0$

Now we find the numerators K_{s_i}

[5]:
```
K = []
for pole in poles:
    K.append(sp.simplify(p/q*(s-pole)).subs(s, pole))
```

[6]:
```
p_fracs = [K[i]/(s-poles[i]) for i in range(len(K))]
Y = sum(p_fracs)
eq_disp('Y(s)',Y)
```

$Y(s) = \frac{3}{5(s+20)} - \frac{8}{5(s+10)} + \frac{1}{s}$

We can check the validity of the partial fractions by comparing their sum to $G(s)R(s)$

[7]:
```
sp.factor(G*R) == sp.factor(sum(p_fracs))
```

[7]:
```
True
```

We can then transform to time domain to obtain $y(t)$

[8]:
```
p_fracs_t = []
for frac in p_fracs:
    p_fracs_t.append(sp.inverse_laplace_transform(frac,s, t))
y = sum(p_fracs_t)
eq_disp('y(t)', y)
```

$y(t) = \theta(t) - \frac{8e^{-10t}\theta(t)}{5} + \frac{3e^{-20t}\theta(t)}{5}$

where $\theta(t)$ is the heaviside function or unit step function

1.2 b)

To get the final value of $y(t)$ we can set the unitstep function to 1

[9]:
```
eq_disp('y(\infty)', y.subs(sp.Heaviside(t), 1))
```

$y(\infty) = 1 - \frac{8e^{-10t}}{5} + \frac{3e^{-20t}}{5}$

2 2.25

[10]:
```
a, x = sp.symbols('a, x')
x0 = 0.6
y = a*x**3
eq_disp('y', y)
```

$y = ax^3$

The linear approximation is obtained from the first order taylor expansion of the amplifier function at the operating point

[11]:
```
y_lin = y.subs(x, x0) + y.diff(x).subs(x, x0)*(x-x0)
eq_disp('y_{linear}', y_lin)
```

$y_{linear} = 1.08a\,(x - 0.6) + 0.216a$

3 2.31

The transfer function $V(s)$ is given by

[12]:
```
V = sp.factor(400/(s**2 + 8*s + 400))
eq_disp('V(s)',V)
p, q = V.as_numer_denom()
```

$V(s) = \frac{400}{s^2+8s+400}$

The denominator is set to be $q(s)$. $q(s) = 0$ is solved to find the poles

[13]:
```
poles = sp.solve(q,s)
for i, pole in enumerate(poles):
    eq_disp(f'-s_{i}', pole)
```

$-s_0 = -4 - 8\sqrt{6}i$

$-s_1 = -4 + 8\sqrt{6}i$

We can construct the partial fractions according to:

$$V(s) = \frac{K_{s1}}{s+s_1} + \frac{K_{s2}}{s+s_2} + \cdots + \frac{K_{sn}}{s+s_n}$$

And find K_{si} with

$$K_{si} = \left[(s+s_i)\,\frac{p(s)}{q(s)}\right]\Bigg|_{s=-s_i}$$

[14]:
```
K = []
for pole in poles:
    K.append(sp.simplify(p/q*(s-pole)).subs(s, pole))

eq_disp('K_{si}',K)
```

$K_{si} = [0,\ 0]$

[15]:
```
p_fracs = [K[i]/(s-poles[i]) for i in range(len(K))]
V = sum(p_fracs)
eq_disp('V(s)',V)
```

$V(s) = 0$

We can then transform to time domain to obtain $y(t)$

[16]:
```
p_fracs_t = []
for frac in p_fracs:
    p_fracs_t.append(sp.inverse_laplace_transform(frac,s, t))
y = sum(p_fracs_t)
eq_disp('y(t)', y)
```

$y(t) = 0$

4 2.26

Initializing functions:

[31]:
```
k, b, m, M = sp.symbols('k, b, m, M')
y = sp.Function('y')(t)
x = sp.Function('x')(t)
F = sp.Function('F')(t)
Fs = sp.Function('F')(s)
dx = x.diff(t)
dy = y.diff(t)
ddx = dx.diff(t)
ddy = dy.diff(t)
```

The ODE for the mass M

[32]:
```
msd_M = M*ddx + b*(dy-dx) + k*(y-x)
eq_disp('F(t)',msd_M)
```

$F(t) = M\frac{d^2}{dt^2}x(t) + b\left(-\frac{d}{dt}x(t) + \frac{d}{dt}y(t)\right) + k\left(-x(t) + y(t)\right)$

The ODE for the mass m

[33]:
```
msd_m = m*ddy - b*(dy-dx) - k*(y-x)
eq_disp('0',msd_m)
```

$0 = -b\left(-\frac{d}{dt}x(t) + \frac{d}{dt}y(t)\right) - k\left(-x(t) + y(t)\right) + m\frac{d^2}{dt^2}y(t)$

So the two differential equations are

$$M\ddot{x} + b(\dot{y} - \dot{x}) + k(y - x) = F(t) \tag{1}$$

$$m\ddot{y} - b(\dot{y} - \dot{x}) - k(y - x) = 0 \tag{2}$$

The Laplace transform of equation 1 is found, initial conditions are assumed to be 0

$$ms^2Y(s) - bsY(s) + bsX(s) - kY(s) + kX(s) = 0$$

$X(s)$ is isolated

$$bsX(s) + kX(s) = -ms^2Y(s) + bsY(s) + kY(s) \tag{1}$$

$$\Leftrightarrow X(s) = \frac{-ms^2Y(s) + bsY(s) + kY(s)}{bs + k} \tag{2}$$

The Laplace transformed equation 2 is found

$$Ms^2X(s) + bsY(s) - bsX(s) + kY(s) - kX(s) = F(s) \tag{3}$$

$$\Leftrightarrow X(s)(Ms^2 - bs - k) + Y(s)(bs + k) = F(s) \tag{4}$$

Insert $X(s)$ found from equation 1 into the Laplace transform of equation 2.

$$\frac{-ms^2Y(s) + bsY(s) + kY(s)}{bs + k}(Ms^2 - bs - k) + Y(s)(bs + k) = F(s) \tag{5}$$

$$\Leftrightarrow Y(s)\left(\frac{(-ms^2 + bs + k)(Ms^2 - bs - k)}{bs + k} + bs + k\right) = F(s) \tag{6}$$

$$\Leftrightarrow Y(s) = \frac{F(s)}{\frac{(-ms^2+bs+k)(Ms^2-bs-k)}{bs+k} + bs + k} \tag{7}$$

[34]:

```
Y = sp.simplify(Fs/((-m*s**2+b*s+k)*(M*s**2-b*s-k)/(b*s+k)+b*s+k))
eq_disp('Y(s)',Y)
```

$Y(s) = \frac{(bs+k)F(s)}{(bs+k)^2-(-Ms^2+bs+k)(bs+k-ms^2)}$

The transfer function of the robot arm model is found by $G(s) = \frac{Y(s)}{F(s)}$

[35]:

```
G = sp.simplify(Y/Fs)
eq_disp('G(s)', G)
```

$G(s) = \frac{bs+k}{(bs+k)^2-(-Ms^2+bs+k)(bs+k-ms^2)}$