HW3

November 11, 2022

[2]:
```python
import numpy as np
from scipy.optimize import minimize
import scipy.signal as si
import sympy as sp
import control as ct
from typing import List
from sympy.plotting import plot
import matplotlib.pyplot as plt
from IPython.display import display, Latex, Math, Image
%matplotlib inline

def eq_disp(varstring, expr, unit=""):
    display(Latex(f"${varstring}={sp.latex(expr)} \: {unit}$"))

def reduce_feedback(G_fwd, G_bwd):
    """Assumes feedback is deducted from signal, if not
    change sign of feedback"""
    return sp.simplify(G_fwd/(1+G_fwd*G_bwd))
```

1 Problem 1

Measureing the voltage at X_i without the resitor R_m is given by:
From ohms law the current of the circuit is found:

[7]:
```python
Rm, Ri, Rt, E = sp.symbols('R_m, R_i, R_t, E')
I_nopot = E/Rt
I_nopot
```

[7]: $\frac{E}{R_t}$

Now from kirchovs law we can find the voltage at point Xi in the circuit

[8]:
```python
V1 = I_nopot*(Rt-Ri)
eq_disp('V1', V1)
```

$V1 = \frac{E(-R_i+R_t)}{R_t}$

We now do the same thing but considering the resistor R_m

[5]:
```python
R_pot = (Rt-Ri) + (1/Ri + 1/Rm)**(-1)
I_pot = E/R_pot
V2 = I_pot*(Rt-Ri)
V2
```

[5]:

$$\frac{E\left(-R_i + R_t\right)}{-R_i + R_t + \frac{1}{\frac{1}{R_m} + \frac{1}{R_i}}}$$

Now the difference between the measured voltage at X_i between the two scenarios will be the error in the measurement

[6]:
```python
sp.simplify(V1-V2)
```

[6]:

$$\frac{ER_i^2\left(-R_i + R_t\right)}{R_t\left(R_i^2 - R_iR_t - R_mR_t\right)}$$

2 Problem 2

The mean and standard deviation of the voltage data

[17]:
```python
U = np.array([1.53, 1.57, 1.54, 1.54, 1.50, 1.51, 1.55, 1.54, 1.56, 1.53])
eq_disp('U', U, 'V')
```

$U = [1.53\ 1.57\ 1.54\ 1.54\ 1.5\ 1.51\ 1.55\ 1.54\ 1.56\ 1.53]\ V$

Is

[18]:
```python
U_mean = np.mean(U)
U_std = np.std(U)
eq_disp('\\bar{U}', round(U_mean,4), 'V')
eq_disp('\\sigma', round(U_std,4), 'V')
```

$\bar{U} = 1.537\ V$

$\sigma = 0.02\ V$

The standard error of the mean is

[19]:
```python
alpha = U_std/np.sqrt(U.size)
eq_disp('\\alpha', round(alpha,4), 'V')
```

$\alpha = 0.0063\ V$

If 1000 measurements with the same standard deviation, the standard error of the mean is

[20]:
```python
alpha1000 = U_std/np.sqrt(1000)
eq_disp('\\alpha_{1000}', round(alpha1000,4), 'V')
```

$\alpha_{1000} = 0.0006\ V$

So the imnprovement is

[21]:
```
alpha_impr = alpha - alpha1000
eq_disp('\\alpha - \\alpha_{1000}', round(alpha_impr,4), 'V')
```

$\alpha - \alpha_{1000} = 0.0057\ V$

3 Problem 3

3.1 a)

[22]:
```
t = 10
h1 = 2
h2 = 3
d = 2
Volume = (h2 - h1)*np.pi*(d/2)**2
```

So the volume flow rate is

[23]:
```
Q = Volume/t
eq_disp('Q', round(Q,3), '\\frac{m^3}{min}')
```

$Q = 0.314\ \frac{m^3}{min}$

3.2 b)

If the error of each length measurement is $\pm 1\%$ the maximum error of the volume flow rate is

[24]:
```
h1_max = 2*1.01
h2_max = 3*1.01
d_max = 2*1.01
Volume_max = (h2_max - h1_max)*np.pi*(d_max/2)**2
Q_max = Volume_max/t
Error_max = Q_max - Q
eq_disp('Error_{max}', round(Error_max,4), '\\frac{m^3}{min}')
```

$Error_{max} = 0.0095\ \frac{m^3}{min}$

3.3 Problem 4

3.3.1 a)

We use the cumulative distribution function to find the chance that a sample is within the given range, then multiply by the population size

[25]:
```
import statistics
pop_sz = 10**5
mean = 20
std = 2
nd = statistics.NormalDist(mean, std)
low_g = 19.8
```

```
high_g = 20.2
prob_of_interval = (nd.cdf(high_g)-nd.cdf(low_g))
eq_disp('N', prob_of_interval*pop_sz)
```

$N = 7965.56745540577$

3.3.2 b)

Once again use cummulative distribution function

[29]:
```
eq_disp('N', (1-nd.cdf(17))*pop_sz)
```

$N = 93319.2798731142$