hw4_2

November 20, 2024

---

1 Question 2

[982]:
```
import pandas as pd

# Initializing the data as a pandas DataFrame
manu_data = {
    'Step': ['a', 'b', 'c', 'd', 'e'],
    'Station': [1, 2, 3, 2, 1],
    'te': [0.16, 0.13, 0.13, 0.18, 0.14],
    'C_e^2': [0.5, 1.2, 0.8, 1.1, 0.6]
}

process_metrics = pd.DataFrame(manu_data)

# Display the DataFrame
print(process_metrics)
```

```
  Step  Station    te  C_e^2
0    a        1  0.16    0.5
1    b        2  0.13    1.2
2    c        3  0.13    0.8
3    d        2  0.18    1.1
4    e        1  0.14    0.6
```

[983]:
```
process_metrics['E[S]^2'] = process_metrics['te'] ** 2
process_metrics['var[S]'] = process_metrics['C_e^2'] * process_metrics['te'] **␣
 ↪2
process_metrics['E[S^2]'] = process_metrics['te'] ** 2 +␣
 ↪process_metrics['var[S]']
# Display the DataFrame
print(process_metrics)
```

```
  Step  Station    te  C_e^2  E[S]^2   var[S]   E[S^2]
0    a        1  0.16    0.5  0.0256  0.01280  0.03840
1    b        2  0.13    1.2  0.0169  0.02028  0.03718
2    c        3  0.13    0.8  0.0169  0.01352  0.03042
```

```
3    d    2  0.18   1.1  0.0324  0.03564  0.06804
4    e    1  0.14   0.6  0.0196  0.01176  0.03136
```

Notice in this question now we will play with multiple machines in the system

[984]:
```python
import numpy as np

# Define the coefficient matrix A and the constants vector b
A = np.array([
    #A      B       C       D       E
    [1,     0,      -1/10,  0,      0],
    [-1,    1,      0,      0,      0],
    [0,     -1,     1,      0,      0],
    [0,     0,      -9/10,  1,      0],
    [0,     0,      0,      -1,     1]
])

# Right-hand side constants vector (all zero in this case)
b = np.array([5, 0, 0, 0, 0])

# Solve the system using np.linalg.solve if it's square and has a unique␣
↪solution
# If it does not have a unique solution, we can use np.linalg.lstsq to get a␣
↪least-squares solution
try:
    lam = np.linalg.solve(A, b)
except np.linalg.LinAlgError:
    # If matrix A is singular, use least-squares solution
    lam, residuals, rank, s = np.linalg.lstsq(A, b, rcond=None)

# The flow rates are the following
lam
```

[984]:
```
array([5.55555556, 5.55555556, 5.55555556, 5.        , 5.        ])
```

[985]:
```python
# Define the lambda inflow rates for each step
lambda_inflow = [lam[0], lam[1], lam[2], lam[3], lam[4]]

# Add the lambda inflow rates to the station_metrics dataframe
process_metrics['lambda_inflow'] = lambda_inflow

# Display the updated DataFrame
print(process_metrics)
```

```
  Step  Station   te  C_e^2  E[S]^2   var[S]   E[S^2]  lambda_inflow
0    a        1  0.16    0.5  0.0256  0.01280  0.03840       5.555556
1    b        2  0.13    1.2  0.0169  0.02028  0.03718       5.555556
2    c        3  0.13    0.8  0.0169  0.01352  0.03042       5.555556
3    d        2  0.18    1.1  0.0324  0.03564  0.06804       5.000000
```

```
4    e        1  0.14    0.6  0.0196  0.01176  0.03136       5.000000
```

Let's articulate this as the individual workstation flow rates

[986]:
```
lam1=lam[1-1]+lam[5-1]
lam1
```

[986]:
```
10.555555555555555
```

[987]:
```
lam2=lam[2-1]+lam[4-1]
lam2
```

[987]:
```
10.555555555555555
```

[988]:
```
lam3=lam[3-1]
lam3
```

[988]:
```
5.555555555555555
```

[989]:
```
# Create a new DataFrame with 'Station' and 'lambda_inflow' columns using lam1,␣
↪lam2, and lam3
station_metrics = pd.DataFrame({
    'Station': [1, 2, 3],
    'lambda_inflow': [lam1, lam2, lam3]
})

# Display the new DataFrame
print(station_metrics)
```

```
   Station  lambda_inflow
0        1      10.555556
1        2      10.555556
2        3       5.555556
```

Now let's get the utilization

[ ]:
```
station_metrics['te'] = [lam[0]/(lam[0]+lam[4])*process_metrics['te'][0]+
                         lam[4]/(lam[0]+lam[4])*process_metrics['te'][4],
                         lam[1]/(lam[1]+lam[3])*process_metrics['te'][1]+
                         lam[3]/(lam[1]+lam[3])*process_metrics['te'][3],
                         process_metrics['te'][2]]

# Display the updated DataFrame
print(station_metrics)
```

```
   Station  lambda_inflow        te
0        1      10.555556  0.150526
1        2      10.555556  0.153684
2        3       5.555556  0.130000
```

[991]:
```python
station_metrics['E[S]^2'] = station_metrics['te'] ** 2

# Display the updated DataFrame
print(station_metrics)
```

```
   Station  lambda_inflow        te    E[S]^2
0        1      10.555556  0.150526  0.022658
1        2      10.555556  0.153684  0.023619
2        3       5.555556  0.130000  0.016900
```

[ ]:
```python
station_metrics['E[S^2]'] = [lam[0]/
 ↪(lam[0]+lam[4])*process_metrics['E[S^2]'][0]+
                             lam[4]/
 ↪(lam[0]+lam[4])*process_metrics['E[S^2]'][4],
                             lam[1]/
 ↪(lam[1]+lam[3])*process_metrics['E[S^2]'][1]+
                             lam[3]/
 ↪(lam[1]+lam[3])*process_metrics['E[S^2]'][3],
                             process_metrics['E[S^2]'][2]]
# Display the updated DataFrame
print(station_metrics)
```

```
   Station  lambda_inflow        te    E[S]^2    E[S^2]
0        1      10.555556  0.150526  0.022658  0.035065
1        2      10.555556  0.153684  0.023619  0.051798
2        3       5.555556  0.130000  0.016900  0.030420
```

[993]:
```python
station_metrics['var(S)'] = station_metrics['E[S^2]'] -␣
 ↪station_metrics['E[S]^2']
# Display the updated DataFrame
print(station_metrics)
```

```
   Station  lambda_inflow        te    E[S]^2    E[S^2]    var(S)
0        1      10.555556  0.150526  0.022658  0.035065  0.012407
1        2      10.555556  0.153684  0.023619  0.051798  0.028179
2        3       5.555556  0.130000  0.016900  0.030420  0.013520
```

[994]:
```python
station_metrics['Ce^2'] = station_metrics['var(S)'] / station_metrics['te'] ** 2
# Display the updated DataFrame
print(station_metrics)
```

```
   Station  lambda_inflow        te    E[S]^2    E[S^2]    var(S)      Ce^2
0        1      10.555556  0.150526  0.022658  0.035065  0.012407  0.547577
1        2      10.555556  0.153684  0.023619  0.051798  0.028179  1.193076
2        3       5.555556  0.130000  0.016900  0.030420  0.013520  0.800000
```

[995]:
```python
station_metrics['m'] = [2, 2, 1]
# Display the updated DataFrame
print(station_metrics)
```

```
   Station  lambda_inflow        te    E[S]^2    E[S^2]    var(S)      Ce^2  m
0        1      10.555556  0.150526  0.022658  0.035065  0.012407  0.547577  2
1        2      10.555556  0.153684  0.023619  0.051798  0.028179  1.193076  2
2        3       5.555556  0.130000  0.016900  0.030420  0.013520  0.800000  1
```

[996]:
```
# Calculate te^2 for each station
station_metrics['u'] = station_metrics['lambda_inflow']*station_metrics['te'] /
 ↪station_metrics['m']
# Display the updated DataFrame
print(station_metrics)
```

```
   Station  lambda_inflow        te    E[S]^2    E[S^2]    var(S)      Ce^2  \
0        1      10.555556  0.150526  0.022658  0.035065  0.012407  0.547577
1        2      10.555556  0.153684  0.023619  0.051798  0.028179  1.193076
2        3       5.555556  0.130000  0.016900  0.030420  0.013520  0.800000

   m         u
0  2  0.794444
1  2  0.811111
2  1  0.722222
```

Now let's get the Variance of the combined distributions for the purpose of calculating the SCV for each station's service time.

Now let's find the CSV of the arrivals

[997]:
```
p12=lam[0]/(lam[0]+lam[4])
p12
```

[997]: 0.5263157894736842

[998]:
```
p21=lam[4]/(lam[4]+lam[1])
p21
```

[998]: 0.4736842105263158

[999]:
```
p31=.1
p32=.9
```

[1000]:
```
p23=lam[1]/(lam[1]+lam[4])
p23
```

[1000]: 0.5263157894736842

[1001]:
```
from sympy import symbols, Eq, solve
```

```python
# Define symbols for the variables
scv_a_1, scv_a_2, scv_a_3 = symbols('scv_a_1, scv_a_2, scv_a_3')
```

$$C_a^2(1)$$

```python
p21=p21
scv_e_2=station_metrics['Ce^2'][2-1]

p31=p31
scv_e_3=station_metrics['Ce^2'][3-1]

m2=m[2-1]
m3=m[3-1]

eq1 = Eq(scv_a_1, 5/lam1
         +lam2*p21/lam1*(p21*(1+(1-station_metrics['u'][2-1]**2)*(scv_a_2-1)+
                              (station_metrics['u'][2-1]**2)*(scv_e_2-1)/
 ↪(m2**(1/2)))+1-p21)
         +lam3*p31/lam1*(p31*(1+(1-station_metrics['u'][3-1]**2)*(scv_a_3-1)+
                              (station_metrics['u'][3-1]**2)*(scv_e_3-1)/
 ↪(m3**(1/2)))+1-p31)
         )
```

$$C_a^2(2)$$

```python
p12=p12
scv_e_2=station_metrics['Ce^2'][2-1]

p32=p32
scv_e_3=station_metrics['Ce^2'][3-1]

m2=m[2-1]
m3=m[3-1]

eq2 = Eq(scv_a_2,
         lam1*p12/lam2*(p12*(1+(1-station_metrics['u'][2-1]**2)*(scv_a_2-1)+
                             (station_metrics['u'][2-1]**2)*(scv_e_2-1)/(m2**(1/
 ↪2)))+1-p12)
         +lam3*p32/lam2*(p32*(1+(1-station_metrics['u'][3-1]**2)*(scv_a_3-1)+
                              (station_metrics['u'][3-1]**2)*(scv_e_3-1)/
 ↪(m3**(1/2)))+1-p32)
)
```

$$C_a^2(3)$$

[ ]:
```
p23=p23
scv_e_3=station_metrics['Ce^2'][3-1]

eq3 = Eq(scv_a_3,
       lam2*p23/lam3*(p23*((1-station_metrics['u'][3-1]**2)*(scv_a_3)+
                           (station_metrics['u'][3-1]**2)*(scv_e_3)))+1-p23)
```

[1005]:
```
# Solve the system of equations
solution = solve((eq1, eq2, eq3), (scv_a_1, scv_a_2, scv_a_3))

print("Solution:", solution)
scv_a_1 = solution[scv_a_1]
scv_a_2 = solution[scv_a_2]
scv_a_3 = solution[scv_a_3]
```

```
Solution: {scv_a_1: 1.01648928724381, scv_a_2: 0.961823241034239, scv_a_3:
0.926617455492835}
```

WIP, CT, and TH of System

[ ]:
```
scv_e_1=0.1505213
m1=2
CT1 = (scv_a_1+scv_e_1)/2*(station_metrics['u'][1-1]**
                         ((2*(m1+1))**(1/2)-1))/
 ↪((1-station_metrics['u'][1-1])*m1)*station_metrics['te'][1-1]
+station_metrics['te'][1-1]
CT1
```

[ ]: 0.303579064338254

[ ]:
```
CT2=(scv_a_2+scv_e_2)/2*(station_metrics['u'][2-1]
                         **((2*(m2+1))
                            **(1/2)-1))/
 ↪((1-station_metrics['u'][2-1])*m2)*station_metrics['te'][2-1]
+station_metrics['te'][2-1]
CT2
```

[ ]: 0.47727983492323

[1008]:
```
CT3=(scv_a_3+scv_e_3)/2*(station_metrics['u'][3-1]**((2*(m3+1))**(1/2)-1))/
 ↪((1-station_metrics['u'][3-1])*m3)*station_metrics['te'][3-1]+station_metrics['te'][3-1]
CT3
```

[1008]: 0.421798349978289

[1009]:
```
WIP1=lam1*CT1
WIP1
```

[1009]: 3.20444567912602

[1010]:
```
WIP2=lam2*CT2
WIP2
```

[1010]: 5.03795381307854

[1011]:
```
WIP3=lam3*CT3
WIP3
```

[1011]: 2.34332416654605

[1012]:
```
WIP=WIP1+WIP2+WIP3
WIP
```

[1012]: 10.5857236587506

[1013]:
```
CT=WIP/5
CT
```

[1013]: 2.11714473175012

[ ]:
```

```