# OR assignment

April 21, 2025

## 1 M/M/s:FCFS/inf/inf

[2]:
```
import math

def mm_s_metrics(lambd, mu, s):
    rho = lambd / mu  # traffic intensity (total)
    r = rho / s  # traffic intensity per server

    if r >= 1:
        raise ValueError("System is unstable (rho/s >= 1). Ensure that λ/μ < s.")

    # Compute P0 (probability of zero customers)
    sum_terms = sum((rho ** n) / math.factorial(n) for n in range(s))
    last_term = (rho ** s) / (math.factorial(s) * (1 - r))
    S = sum_terms + last_term
    P0 = 1 / S

    # Compute Lq (average number in queue)
    Lq = P0 * ((rho ** (s + 1)) / (s * math.factorial(s) * ((1 - r) ** 2)))

    # Compute Wq (average waiting time in queue)
    Wq = Lq / lambd

    # Compute W (total time in system)
    W = Wq + (1 / mu)

    # Compute L (average number in system)
    L = lambd * W

    return {
        'L': L,
        'Lq': Lq,
        'W': W,
        'Wq': Wq,
        'P0': P0
    }
```

```python
# Example usage:
# λ = 5 customers/min, μ = 2 customers/min/server, s = 3 servers
results = mm_s_metrics(lambd=1.1750, mu=1.0901, s=3)

for key, value in results.items():
    print(f"{key} = {value:.4f}")
```

```
L = 1.1391
Lq = 0.0612
W = 0.9694
Wq = 0.0521
P0 = 0.3351
```

## 2 M/M/s:FCFS/m/inf

[8]:
```
import math

def mm_s_m_metrics(lambd, mu, s, m):
    rho = lambd / mu
    r = rho / s

    # Compute normalization constant S
    sum1 = sum((rho**n) / math.factorial(n) for n in range(s))
    if r != 1:
        geometric_sum = ((1 - (r)**(m - s + 1)) / (1 - r))
    else:
        geometric_sum = m - s + 1

    sum2 = ((rho ** s) / math.factorial(s)) * geometric_sum
    S = sum1 + sum2

    # P0
    P0 = 1 / S

    # Probability for all states up to m
    P = [0] * (m + 1)
    for n in range(m + 1):
        if n < s:
            P[n] = ((rho ** n) / math.factorial(n)) * P0
        else:
            P[n] = ((rho ** n) / (math.factorial(s) * (s ** (n - s)))) * P0

    # Blocking probability P_m
    P_m = P[m]

    # Lq calculation (queue length)
    numerator = (rho ** (s + 1)) * P0
    denominator = s * math.factorial(s)
    if r != 1:
        bracket = (1 - (m - s + 1)*(r)**(m - s) + (m - s)*(r)**(m - s + 1)) /
 ↪((1 - r)**2)
    else:
        bracket = (m - s)*(m - s + 1) / 2

    Lq = (numerator / denominator) * bracket

    # Effective arrival rate
    lambda_eff = lambd * (1 - P_m)

    # Wq, W, L
```

```python
    Wq = Lq / lambd
    W = Wq + (1 / mu)
    L = lambda_eff * W

    return {
        'P0': P0,
        # 'Pm (blocking)': P_m,
        'Lq': Lq,
        'Wq': Wq,
        'W': W,
        'L': L,
        # 'lambda_eff': lambda_eff
    }

# Example usage:
# lambda = 4, mu = 2, s = 2 servers, m = 5 capacity
results = mm_s_m_metrics(lambd=1.1750, mu=1.0901, s=3, m=1)

for key, value in results.items():
    print(f"{key} = {value:.4f}")
```

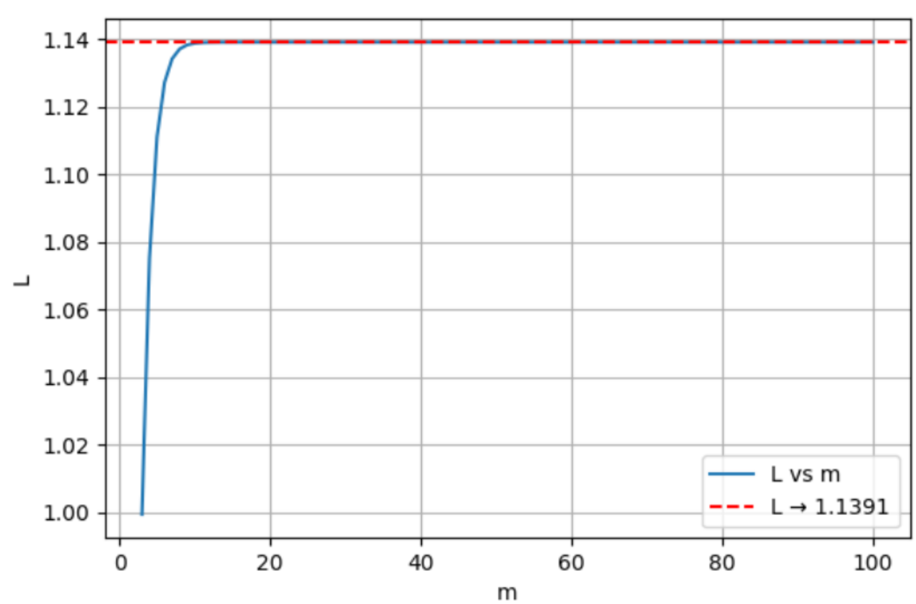
```
P0 = 0.4813
Lq = 0.2796
Wq = 0.2379
W = 1.1553
L = 0.6533
```

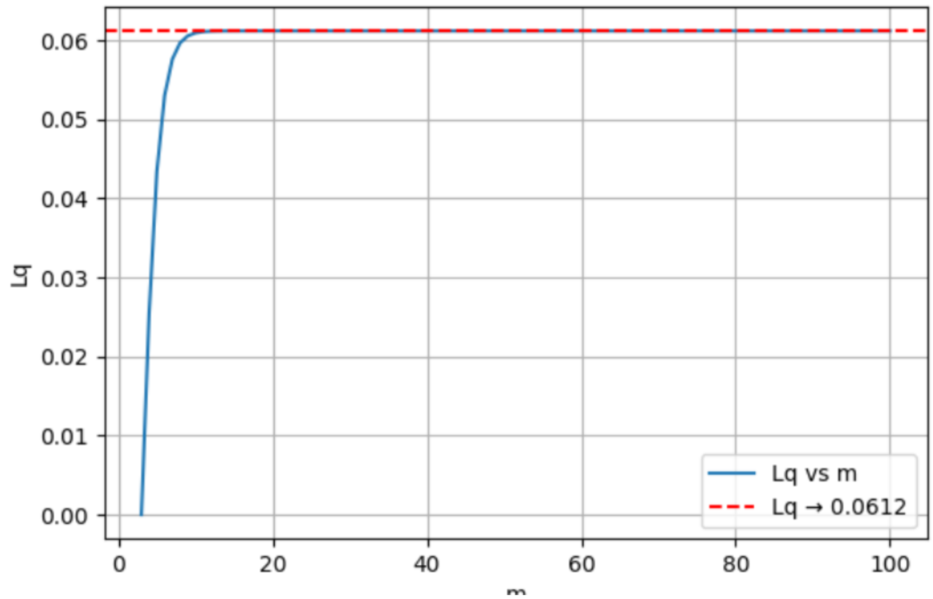
## 3 Plotting graphs

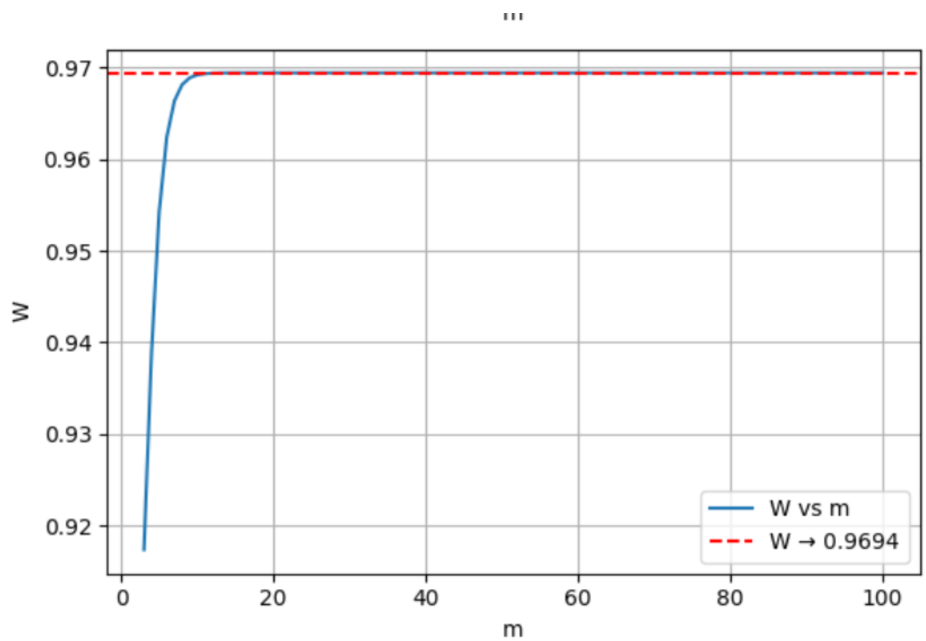
[9]:
```
import math
import matplotlib.pyplot as plt

# Provided function to compute metrics
def mm_s_m_metrics(lambd, mu, s, m):
    rho = lambd / mu
    r = rho / s

    # Compute normalization constant S
    sum1 = sum((rho**n) / math.factorial(n) for n in range(s))
    if r != 1:
        geometric_sum = ((1 - (r)**(m - s + 1)) / (1 - r))
    else:
        geometric_sum = m - s + 1

    sum2 = ((rho ** s) / math.factorial(s)) * geometric_sum
    S = sum1 + sum2

    # P0
    P0 = 1 / S

    # Probability for all states up to m
    P = [0] * (m + 1)
    for n in range(m + 1):
        if n < s:
            P[n] = ((rho ** n) / math.factorial(n)) * P0
        else:
            P[n] = ((rho ** n) / (math.factorial(s) * (s ** (n - s)))) * P0

    # Blocking probability P_m
    P_m = P[m]

    # Lq calculation (queue length)
    numerator = (rho ** (s + 1)) * P0
    denominator = s * math.factorial(s)
    if r != 1:
        bracket = (1 - (m - s + 1)*(r)**(m - s) + (m - s)*(r)**(m - s + 1)) /
 ↪((1 - r)**2)
    else:
        bracket = (m - s)*(m - s + 1) / 2

    Lq = (numerator / denominator) * bracket

    # Effective arrival rate
    lambda_eff = lambd * (1 - P_m)
```

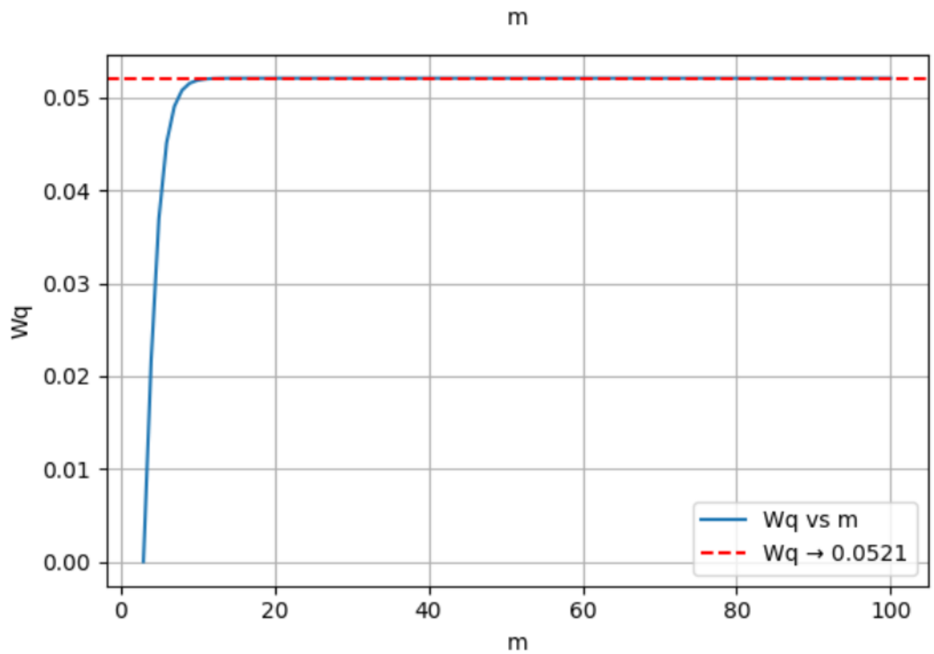
```
    # Wq, W, L
    Wq = Lq / lambd
    W = Wq + (1 / mu)
    L = lambda_eff * W

    return {
        'Lq': Lq,
        'Wq': Wq,
        'W': W,
        'L': L,
        'P0': P0
    }

# Constants
lambd = 1.1750
mu = 1.0901
s = 3

# Prepare data for plotting
m_values = list(range(3, 101))
L_vals = []
Lq_vals = []
W_vals = []
Wq_vals = []
P0_vals = []

for m in m_values:
    results = mm_s_m_metrics(lambd, mu, s, m)
    L_vals.append(results['L'])
    Lq_vals.append(results['Lq'])
    W_vals.append(results['W'])
    Wq_vals.append(results['Wq'])
    P0_vals.append(results['P0'])

# Plotting
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.plot(m_values, L_vals, label="L vs m")
plt.axhline(y=1.1391, color='r', linestyle='--', label="L → 1.1391")
plt.xlabel("m")
plt.ylabel("L")
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 2)
```

```
plt.plot(m_values, Lq_vals, label="Lq vs m")
plt.axhline(y=0.0612, color='r', linestyle='--', label="Lq → 0.0612")
plt.xlabel("m")
plt.ylabel("Lq")
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 3)
plt.plot(m_values, W_vals, label="W vs m")
plt.axhline(y=0.9694, color='r', linestyle='--', label="W → 0.9694")
plt.xlabel("m")
plt.ylabel("W")
plt.legend()
plt.grid(True)

plt.subplot(2, 2, 4)
plt.plot(m_values, Wq_vals, label="Wq vs m")
plt.axhline(y=0.0521, color='r', linestyle='--', label="Wq → 0.0521")
plt.xlabel("m")
plt.ylabel("Wq")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```




Lq
m
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100
Lq vs m
Lq → 0.0612


W
m
0.97
0.96
0.95
0.94
0.93
0.92
0
20
40
60
80
100
W vs m
W → 0.9694


m
Wq
0.00
0.01
0.02
0.03
0.04
0.05
0
20
40
60
80
100
m
Wq vs m
Wq → 0.0521

## 4 Real time data

[10]:
```python
import random
import pandas as pd

# PARAMETERS
SIM_TIME = 120    # Total time in seconds (2 minutes)
LAMBDA = 1.2      # Arrival rate: signals per second (adjustable)
MU = 1.0          # Service rate: signals per second (adjustable)

# INITIAL STATE
current_time = 0
next_arrival_time = random.expovariate(LAMBDA)
server_available_time = 0
signal_log = []
signal_number = 1

# SIMULATION LOOP
while next_arrival_time < SIM_TIME:
    arrival_time = next_arrival_time
    service_start = max(arrival_time, server_available_time)
    service_duration = random.expovariate(MU)
    service_end = service_start + service_duration
    wait_time = service_start - arrival_time

    # Log this signal
    signal_log.append({
        "Signal #": signal_number,
        "Arrival Time (s)": round(arrival_time, 2),
        "Service Start (s)": round(service_start, 2),
        "Service End (s)": round(service_end, 2),
        "Wait Time (s)": round(wait_time, 2),
        "Service Duration (s)": round(service_duration, 2)
    })

    # Update state
    signal_number += 1
    server_available_time = service_end
    next_arrival_time += random.expovariate(LAMBDA)

# CONVERT TO TABLE
df = pd.DataFrame(signal_log)

# CALCULATE LAMBDA AND MU
total_time = SIM_TIME
lambda_sim = len(df) / total_time
total_service_time = df["Service Duration (s)"].sum()
```

```python
mu_sim = len(df) / total_service_time

# PRINT TABLE AND STATS
print(df.to_string(index=False))

print("\n--- SIMULATION SUMMARY ---")
print(f"Total simulation time: {SIM_TIME} seconds")
print(f"Total signals arrived: {len(df)}")
print(f"Total signals served: {len(df)}")
print(f"Estimated λ (arrival rate): {lambda_sim:.4f} signals/sec")
print(f"Estimated μ (service rate): {mu_sim:.4f} signals/sec")
```

| Signal # | Arrival Time (s) | Service Start (s) | Service End (s) | Wait Time (s) | Service Duration (s) |
|---|---|---|---|---|---|
| 1 | 0.23 | 0.23 | 0.75 | 0.00 | 0.52 |
| 2 | 0.85 | 0.85 | 3.08 | 0.00 | 2.23 |
| 3 | 0.88 | 3.08 | 3.20 | 2.19 | 0.13 |
| 4 | 1.38 | 3.20 | 3.50 | 1.82 | 0.30 |
| 5 | 1.49 | 3.50 | 5.79 | 2.01 | 2.29 |
| 6 | 1.71 | 5.79 | 6.38 | 4.09 | 0.59 |
| 7 | 1.73 | 6.38 | 6.41 | 4.65 | 0.04 |
| 8 | 2.73 | 6.41 | 7.56 | 3.68 | 1.15 |
| 9 | 2.77 | 7.56 | 8.74 | 4.79 | 1.18 |
| 10 | 3.32 | 8.74 | 9.62 | 5.42 | 0.88 |
| 11 | 3.88 | 9.62 | 9.71 | 5.74 | 0.09 |
| 12 | 3.89 | 9.71 | 10.18 | 5.83 | 0.47 |
| 13 | 4.29 | 10.18 | 11.45 | 5.89 | 1.27 |
| 14 | 5.47 | 11.45 | 11.50 | 5.98 | 0.05 |
| 15 | 7.52 | 11.50 | 12.86 | 3.98 | 1.35 |
| 16 | 9.39 | 12.86 | 13.30 | 3.47 | 0.44 |
| 17 | 9.89 | 13.30 | 14.45 | 3.40 | 1.16 |
| 18 | 11.08 | 14.45 | 15.16 | 3.38 | 0.70 |
| 19 | 11.50 | 15.16 | 15.17 | 3.66 | 0.01 |
| 20 | 12.12 | 15.17 | 15.59 | 3.05 | 0.42 |
| 21 | 12.50 | 15.59 | 16.04 | 3.09 | 0.46 |
| 22 | 13.06 | 16.04 | 16.43 | 2.99 | 0.38 |
| 23 | 13.84 | 16.43 | 16.65 | 2.59 | 0.22 |
| 24 | 15.95 | 16.65 | 17.49 | 0.69 | 0.84 |
| 25 | 16.70 | 17.49 | 19.12 | 0.79 | 1.63 |
| 26 | 17.17 | 19.12 | 19.57 | 1.96 | 0.45 |
| 27 | 17.35 | 19.57 | 20.42 | 2.22 | 0.85 |
| 28 | 20.47 | 20.47 | 22.74 | 0.00 | 2.27 |
| 29 | 21.92 | 22.74 | 23.05 | 0.83 | 0.31 |
| 30 | 23.23 | 23.23 | 24.86 | 0.00 | 1.63 |
| 31 | 25.32 | 25.32 | 25.41 | 0.00 | 0.09 |
| 32 | 25.67 | 25.67 | 25.70 | 0.00 | 0.04 |

*Continued on next page*

Table 1 – *Continued from previous page*

| Signal # | Arrival Time (s) | Service Start (s) | Service End (s) | Wait Time (s) | Service Duration (s) |
|---|---|---|---|---|---|
| 33 | 26.46 | 26.46 | 29.55 | 0.00 | 3.09 |
| 34 | 29.43 | 29.55 | 30.04 | 0.12 | 0.49 |
| 35 | 30.14 | 30.14 | 30.43 | 0.00 | 0.29 |
| 36 | 31.37 | 31.37 | 31.69 | 0.00 | 0.33 |
| 37 | 33.32 | 33.32 | 33.65 | 0.00 | 0.33 |
| 38 | 33.36 | 33.65 | 33.91 | 0.29 | 0.26 |
| 39 | 34.02 | 34.02 | 34.07 | 0.00 | 0.05 |
| 40 | 34.20 | 34.20 | 34.99 | 0.00 | 0.79 |
| 41 | 34.39 | 34.99 | 35.44 | 0.60 | 0.44 |
| 42 | 35.46 | 35.46 | 38.71 | 0.00 | 3.25 |
| 43 | 38.48 | 38.71 | 39.23 | 0.23 | 0.52 |
| 44 | 38.91 | 39.23 | 40.66 | 0.32 | 1.43 |
| 45 | 41.02 | 41.02 | 41.74 | 0.00 | 0.72 |
| 46 | 44.90 | 44.90 | 45.33 | 0.00 | 0.43 |
| 47 | 46.26 | 46.26 | 46.61 | 0.00 | 0.35 |
| 48 | 46.39 | 46.61 | 48.87 | 0.22 | 2.26 |
| 49 | 46.75 | 48.87 | 49.72 | 2.12 | 0.85 |
| 50 | 47.76 | 49.72 | 50.35 | 1.96 | 0.63 |
| 51 | 49.14 | 50.35 | 51.17 | 1.21 | 0.82 |
| 52 | 51.83 | 51.83 | 52.39 | 0.00 | 0.56 |
| 53 | 54.64 | 54.64 | 55.46 | 0.00 | 0.82 |
| 54 | 54.92 | 55.46 | 56.93 | 0.54 | 1.47 |
| 55 | 55.60 | 56.93 | 57.78 | 1.33 | 0.85 |
| 56 | 55.97 | 57.78 | 58.28 | 1.81 | 0.49 |
| 57 | 56.87 | 58.28 | 58.96 | 1.40 | 0.69 |
| 58 | 57.26 | 58.96 | 59.62 | 1.70 | 0.66 |
| 59 | 57.64 | 59.62 | 62.92 | 1.98 | 3.30 |
| 60 | 58.07 | 62.92 | 63.39 | 4.85 | 0.46 |
| 61 | 58.97 | 63.39 | 63.99 | 4.42 | 0.60 |
| 62 | 60.14 | 63.99 | 64.31 | 3.85 | 0.32 |
| 63 | 60.44 | 64.31 | 64.54 | 3.87 | 0.24 |
| 64 | 62.18 | 64.54 | 66.87 | 2.37 | 2.33 |
| 65 | 63.39 | 66.87 | 69.20 | 3.49 | 2.33 |
| 66 | 63.77 | 69.20 | 69.92 | 5.43 | 0.72 |
| 67 | 63.97 | 69.92 | 70.26 | 5.95 | 0.34 |
| 68 | 64.49 | 70.26 | 72.16 | 5.77 | 1.90 |
| 69 | 65.00 | 72.16 | 72.40 | 7.16 | 0.24 |
| 70 | 66.00 | 72.40 | 74.61 | 6.40 | 2.21 |
| 71 | 66.30 | 74.61 | 75.47 | 8.31 | 0.86 |
| 72 | 66.71 | 75.47 | 76.29 | 8.76 | 0.82 |
| 73 | 66.83 | 76.29 | 76.62 | 9.46 | 0.34 |
| 74 | 67.63 | 76.62 | 77.51 | 8.99 | 0.88 |
| 75 | 67.83 | 77.51 | 78.09 | 9.68 | 0.58 |
| 76 | 68.07 | 78.09 | 78.17 | 10.01 | 0.08 |

*Continued on next page*

Table 1 – *Continued from previous page*

| Signal # | Arrival Time (s) | Service Start (s) | Service End (s) | Wait Time (s) | Service Duration (s) |
|---|---|---|---|---|---|
| 77 | 68.59 | 78.17 | 78.26 | 9.58 | 0.09 |
| 78 | 69.06 | 78.26 | 79.96 | 9.20 | 1.70 |
| 79 | 69.29 | 79.96 | 80.95 | 10.67 | 0.99 |
| 80 | 69.38 | 80.95 | 81.03 | 11.57 | 0.08 |
| 81 | 69.49 | 81.03 | 81.63 | 11.53 | 0.60 |
| 82 | 69.60 | 81.63 | 84.75 | 12.02 | 3.12 |
| 83 | 70.08 | 84.75 | 85.35 | 14.68 | 0.60 |
| 84 | 70.53 | 85.35 | 87.26 | 14.82 | 1.91 |
| 85 | 71.50 | 87.26 | 88.87 | 15.77 | 1.61 |
| 86 | 71.61 | 88.87 | 90.30 | 17.26 | 1.43 |
| 87 | 72.07 | 90.30 | 91.69 | 18.23 | 1.39 |
| 88 | 72.60 | 91.69 | 92.35 | 19.09 | 0.66 |
| 89 | 73.00 | 92.35 | 92.99 | 19.35 | 0.64 |
| 90 | 73.23 | 92.99 | 98.74 | 19.76 | 5.75 |
| 91 | 73.40 | 98.74 | 100.49 | 25.33 | 1.75 |
| 92 | 74.62 | 100.49 | 101.20 | 25.88 | 0.70 |
| 93 | 74.74 | 101.20 | 101.26 | 26.46 | 0.06 |
| 94 | 74.76 | 101.26 | 102.44 | 26.50 | 1.18 |
| 95 | 75.09 | 102.44 | 102.53 | 27.35 | 0.09 |
| 96 | 75.20 | 102.53 | 103.08 | 27.33 | 0.55 |
| 97 | 78.44 | 103.08 | 103.70 | 24.64 | 0.61 |
| 98 | 78.46 | 103.70 | 104.04 | 25.24 | 0.34 |
| 99 | 79.04 | 104.04 | 104.57 | 25.00 | 0.53 |
| 100 | 79.76 | 104.57 | 104.60 | 24.80 | 0.03 |
| 101 | 80.12 | 104.60 | 104.98 | 24.47 | 0.38 |
| 102 | 82.63 | 104.98 | 105.38 | 22.35 | 0.41 |
| 103 | 82.80 | 105.38 | 105.53 | 22.58 | 0.15 |
| 104 | 83.88 | 105.53 | 106.02 | 21.65 | 0.49 |
| 105 | 85.11 | 106.02 | 106.16 | 20.91 | 0.14 |
| 106 | 85.54 | 106.16 | 107.51 | 20.62 | 1.36 |
| 107 | 86.12 | 107.51 | 107.76 | 21.40 | 0.25 |
| 108 | 86.22 | 107.76 | 108.19 | 21.54 | 0.43 |
| 109 | 86.68 | 108.19 | 111.22 | 21.51 | 3.02 |
| 110 | 87.16 | 111.22 | 111.63 | 24.06 | 0.41 |
| 111 | 87.24 | 111.63 | 111.71 | 24.38 | 0.08 |
| 112 | 88.28 | 111.71 | 112.08 | 23.43 | 0.37 |
| 113 | 88.71 | 112.08 | 112.44 | 23.37 | 0.36 |
| 114 | 88.75 | 112.44 | 112.88 | 23.69 | 0.44 |
| 115 | 88.91 | 112.88 | 113.83 | 23.97 | 0.96 |
| 116 | 91.73 | 113.83 | 114.73 | 22.10 | 0.90 |
| 117 | 93.12 | 114.73 | 115.83 | 21.61 | 1.10 |
| 118 | 93.17 | 115.83 | 116.32 | 22.66 | 0.49 |
| 119 | 95.20 | 116.32 | 116.70 | 21.12 | 0.37 |
| 120 | 96.09 | 116.70 | 119.31 | 20.60 | 2.61 |

*Continued on next page*

Table 1 – *Continued from previous page*

| Signal # | Arrival Time (s) | Service Start (s) | Service End (s) | Wait Time (s) | Service Duration (s) |
|---|---|---|---|---|---|
| 121 | 96.39 | 119.31 | 119.45 | 22.92 | 0.14 |
| 122 | 97.36 | 119.45 | 119.53 | 22.09 | 0.08 |
| 123 | 98.57 | 119.53 | 119.66 | 20.96 | 0.13 |
| 124 | 101.58 | 119.66 | 120.51 | 18.07 | 0.85 |
| 125 | 102.10 | 120.51 | 120.66 | 18.41 | 0.15 |
| 126 | 102.64 | 120.66 | 121.33 | 18.02 | 0.66 |
| 127 | 103.15 | 121.33 | 121.35 | 18.17 | 0.02 |
| 128 | 104.43 | 121.35 | 121.78 | 16.92 | 0.43 |
| 129 | 105.91 | 121.78 | 122.17 | 15.87 | 0.39 |
| 130 | 110.47 | 122.17 | 122.83 | 11.70 | 0.66 |
| 131 | 111.08 | 122.83 | 122.98 | 11.75 | 0.15 |
| 132 | 111.26 | 122.98 | 126.90 | 11.72 | 3.92 |
| 133 | 116.25 | 126.90 | 126.95 | 10.65 | 0.05 |
| 134 | 116.38 | 126.95 | 128.73 | 10.57 | 1.78 |
| 135 | 116.41 | 128.73 | 129.53 | 12.32 | 0.80 |
| 136 | 116.85 | 129.53 | 132.72 | 12.68 | 3.19 |
| 137 | 117.20 | 132.72 | 135.09 | 15.52 | 2.37 |
| 138 | 117.55 | 135.09 | 135.60 | 17.54 | 0.52 |
| 139 | 118.38 | 135.60 | 137.15 | 17.22 | 1.55 |
| 140 | 119.99 | 137.15 | 139.19 | 17.16 | 2.04 |

| **Metric** | **Value** |
|---|---|
| Total simulation time | 120 seconds |
| Total signals arrived | 140 |
| Total signals served | 140 |
| Estimated arrival rate (λ) | 1.1667 signals/sec |
| Estimated service rate (μ) | 1.1035 signals/sec |

```
[ ]:
```