ECE 5371 ENGINEERING ANALYSIS

Assignment 6

Rishikesh
R11643260
Texas Tech University
04/25/2024

Contents

1 Probelm 1

Parts arrive at station 1 from a conveyor belt with a Poisson distribution at a mean rate of 1-per-second. Here they mesh with parts arriving from another conveyor belt at a steady and constant rate of 1 part in 1.5 seconds, to form a more complicated assembly and emerge from Station 1. Assume the complicated assembly can be completely by Station 1 instantaneously. Station 1 is known to have a failure rate of 0.01 per second that is characterized by the exponential distribution and stays "down" for 5 seconds. Based on the above, write a Monte Carlo simulator of this scenario that can then yield the discrete event average output rate for the assembled parts from Station 1.

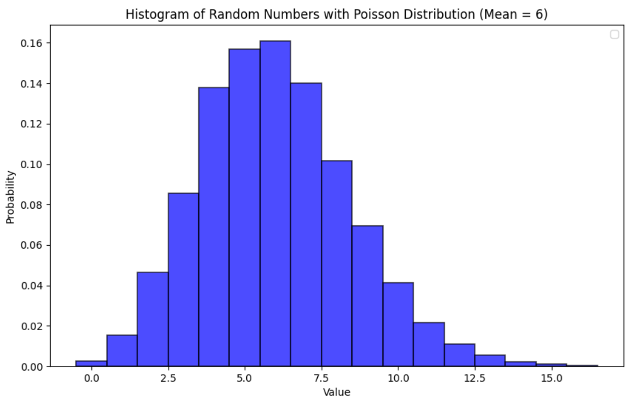

Figure 1: Histogram of Poisson Distribution with $\mu = 6$

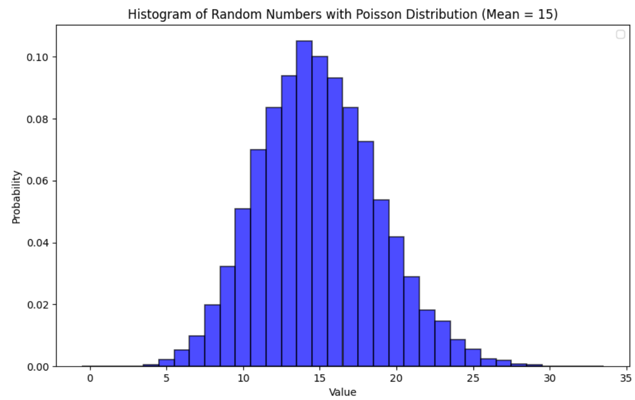

Figure 2: Histogram of Poisson Distribution with $\mu = 15$

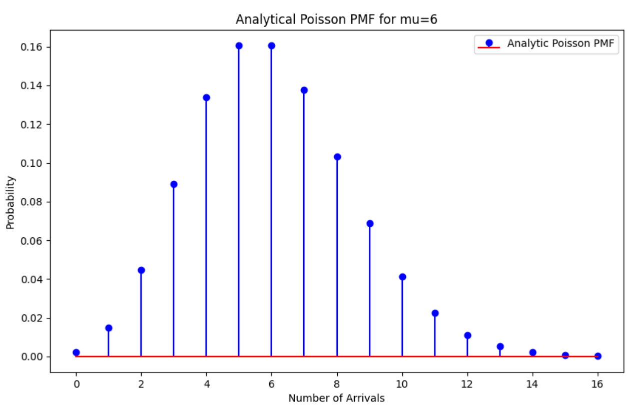

Figure 3: Analytical Formula for the Poisson Distribution Histogram with $\mu = 6$

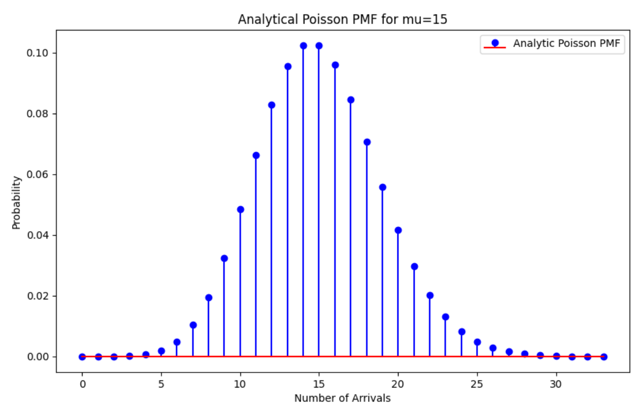

Figure 4: Analytical Formula for the Poisson Distribution Histogram with $\mu = 15$

```python
def generate_poisson_arrivals(rate, max_time):
    arrivals = []
    current_time = 0
    while current_time < max_time:
        inter_arrival_time = np.random.exponential(1/rate)
        current_time += inter_arrival_time
        if current_time < max_time:
            arrivals.append(current_time)
    return arrivals

def generate_constant_rate_arrivals(rate, max_time):
    interval = rate
    return np.arange(interval, max_time, interval)

def simulate_system(max_time, poisson_rate, constant_rate, failure_rate, down_time):
    poisson_arrivals = generate_poisson_arrivals(poisson_rate, max_time)
    constant_arrivals = generate_constant_rate_arrivals(constant_rate, max_time)
    next_failure_time = np.random.exponential(1/failure_rate)

    assembled_count = 0
    current_time = 0
    failure_active_until = -1

    events = sorted([(t, 'poisson') for t in poisson_arrivals] + [(t, 'constant') for t in
    constant_arrivals])

    for event_time, event_type in events:
        if event_time >= next_failure_time:
            failure_active_until = event_time + down_time
            next_failure_time += np.random.exponential(1/failure_rate)

        if event_time >= failure_active_until:
            assembled_count += 1

    return assembled_count, assembled_count / max_time

def plot_poisson(mu, max_time, num_samples):
    samples = [len(generate_poisson_arrivals(mu, max_time)) for _ in range(num_samples)]

    max_val = max(samples)
    x = np.arange(0, max_val + 1)
    y = poisson.pmf(x, mu * max_time)


    # Plot for Histogram
    plt.figure(figsize=(10, 6))
    plt.hist(samples, bins=np.arange(-0.5, max(samples)+1), density=True, alpha=0.7, color='blue',
    edgecolor='black', linewidth=1.2)
    plt.title(f'Histogram of Random Numbers with Poisson Distribution (Mean = {mu})')
    plt.xlabel('Value')
    plt.ylabel('Probability')
    plt.legend()
    plt.show()

    # Plot for Analytical PMF
    plt.figure(figsize=(10, 6))
```

```python
    plt.stem(x, y, 'b', markerfmt='bo', basefmt="r-", use_line_collection=True, label='Analytic Poisson PMF')
    plt.title(f'Analytical Poisson PMF for mu={mu}')
    plt.xlabel('Number of Arrivals')
    plt.ylabel('Probability')
    plt.legend()
    plt.show()

# Parameters
mu_values = [6, 15]
max_time = 1  # time interval to count arrivals
num_samples = 10000  # number of samples to generate for the histograms

for mu in mu_values:
    plot_poisson(mu, max_time, num_samples)
```

Listing 1: Python code Monte Carlo simulator of this scenario that can then yield the discrete event average output rate for the assembled parts from Station 1

1.1 For the "exponential distribution of the failure rate", simply obtain the relation between the "time-to-failure" and a random number.

In an exponential distribution, the time T until the next event can be described by the probability density function:

$$f(t;\lambda) = \lambda e^{-\lambda t}$$

where:

- λ is the rate parameter = 0.01 per second, which is the reciprocal of the mean time between events.
- t is the time.

The cumulative distribution function, which gives the probability that the time until the next event is less than or equal to t, is:

$$F(t;\lambda) = 1 - e^{-\lambda t}$$

1.2 Generating Time to Failure

To generate a random time-to-failure:

1. We can use a uniformly distributed random number U in the interval [0, 1].
2. Transform this random number using the inverse of the cumulative distribution function of the exponential distribution:

$$T = F^{-1}(U;\lambda) = -\frac{\ln(1-U)}{\lambda}$$

Since $1-U$ is statistically identical to U for a uniform distribution, We can simplify this to:

$$T = -\frac{\ln(U)}{\lambda}$$

1.3 Result from simulations

Time to next failure: 186.3597 seconds
Total assembled parts: 15632
Average output rate: 1.5632 parts per second

```
def simulate_system(max_time, poisson_rate, constant_rate, failure_rate, down_time):
    poisson_arrivals = generate_poisson_arrivals(poisson_rate, max_time)
    constant_arrivals = generate_constant_rate_arrivals(constant_rate, max_time)
    next_failure_time = np.random.exponential(1/failure_rate)

    assembled_count = 0
    current_time = 0
    failure_active_until = -1

    events = sorted([(t, 'poisson') for t in poisson_arrivals] + [(t, 'constant') for t in constant_arrivals])

    for event_time, event_type in events:
        if event_time >= next_failure_time:
            failure_active_until = event_time + down_time
            next_failure_time += np.random.exponential(1/failure_rate)

```

```python
        if event_time >= failure_active_until:
            assembled_count += 1

    return assembled_count, assembled_count / max_time

def generate_time_to_failure(failure_rate):
    U = np.random.uniform()
    T = -np.log(U) / failure_rate
    return T


failure_rate = 0.01  # failures per second
time_to_failure = generate_time_to_failure(failure_rate)
print(f"Time to next failure: {time_to_failure} seconds")

# Simulation parameters
max_time = 10000  # seconds
poisson_rate = 1  # parts per second
constant_rate = 1.5  # time in seconds per part
failure_rate = 0.01  # failures per second
down_time = 5  # seconds

output_count, average_rate = simulate_system(max_time, poisson_rate, constant_rate, failure_rate, down_time)
print(f"Total assembled parts: {output_count}")
print(f"Average output rate: {average_rate} parts per second")
```

Listing 2: Python code Monte Carlo simulator of this scenario that can then yield the discrete event average output rate for the assembled parts from Station 1