IEOR 4404 Simulation **Fall 2024**

Homework 8

Instructor: Henry Lam

Problem 1 Messages arrive at a communications facility according to a Poisson process with rate 2 per hour.

(a) Write down the pseudo-codes for simulating the Poisson process until the 10-th hour, using both the interarrival time generation approach and the conditional representation approach.

(b) Suppose the facility consists of three channels, and an arriving message will either go to a free channel if any of them are free, in which case the corresponding channel becomes busy, or else will be lost if all channels are busy. The amount of time that a message occupies a channel is an exponential random variable with mean 1 hour, independent from everything else. Suppose that the system is initially empty at time 0. By using any of the simulators in part (a) and in addition tracing the message occupancy, use simulation to estimate the probability that all three channels are busy at the 10-th hour. Use a replication size (i.e., number of simulated trajectories) 100.

Problem 2 Consider a life insurance company that works as follows. Customers arrive according to a Poisson process with rate ρ. The ith individual stays in the system for an amount of time T_i, which is uniformly distributed in the interval $[0, t]$ independent of all other randomness. Suppose the system is empty at time zero.

(a) What is the distribution of the number of customers who are still in the system at time t?

(b) Suppose that at time t the company has 5 customers in its portfolio. Conditional on this information, explain with a pseudo-code how to simulate the remaining times of these customers.

(c) Implement the pseudo-code in part (b) with $\rho = 1$ and $t = 10$, and plot a histogram of the average remaining times among the 5 customers in the portfolio. Use a replication size 100.

Problem 3 Suppose customers arrive according to a time inhomogeneous Poisson process with intensity function $\lambda(t) = 2 + \cos(2\pi t)$.

(a) Write down the pseudo-codes for simulating the process until time 10, using both the interarrival time generation approach and the conditional representation approach.

(b) Argue that, conditional on the number of customers at time 10, each arrival time is independently distributed according to a density proportional to $\lambda(t)$. Using this, derive the pseudo-code of an alternative method to simulate the process until time 10 in addition to the two approaches in part (a).

(c) Implement the two approaches in part (a) and the alternative approach in part (b), to estimate the expected time-averaged number of customers during the time $[0, 10]$, i.e., the expectation of $\int_0^{10} N(t)dt$. Use a replication size 100.

Alexandre
Duhamel
afd2153
MSOR

HW 8: Simulation

Problem 1:

a) 1. Inter-arrival time approach:
- Set $t, I = 0, 0$
- while $t < 10$:
  - Generate $U \sim Unif(0,1)$
  - $t -= \frac{1}{\lambda} \log U$ where $\lambda = 2$ ITM exp
  - $I += 1$
  - $A(I) = t$
- return $I-1$ and $A[:-1]$

2. Conditional operation approach:
- Generate $N(10) \sim Pois(20)$ XT
- Suppose $N(10) = n$, generate n iid $U(0,1) : U_1, \dots, U_n$
- Sort $U_1, \dots, U_n$ to get : $U_{(1)} < \dots < U_{(n)}$
- return n and $A_i = T U_{(i)}$ for $i = 1, \dots, n$

Alternatively:
- Generate $N(10) \sim Pois(20)$ XT
- Suppose $N(10) = n$, generate n iid $U(0,10) : U_1, \dots, U_n$
- Sort $U_1, \dots, U_n$ to get : $U_{(1)} < \dots < U_{(n)}$
- return n and $U_{(i)}$ for $i = 1, \dots, n$

b)

```python
arrival_rate = 2
num_channels = 3
channel_mean_time = 1
end_time = 10
num_simulations = 100

def simulate_poisson_process_interarrival(end_time, rate):
    arrival_times = []
    current_time = 0
    while current_time < end_time:
        interarrival_time = np.random.exponential(1 / rate)
        current_time += interarrival_time
        if current_time < end_time:
            arrival_times.append(current_time)
    return arrival_times

def probability():
    all_busy_count = 0

    for _ in range(num_simulations):
        arrival_times = simulate_poisson_process_interarrival(end_time, arrival_rate)

        channel_occupancy = [0] * num_channels

        for arrival_time in arrival_times:
            free_channel = None
            for i in range(num_channels):
                if channel_occupancy[i] <= arrival_time:
                    free_channel = i
                    break

            if free_channel is not None:
                occupancy_time = np.random.exponential(channel_mean_time)
                channel_occupancy[free_channel] = arrival_time + occupancy_time

        if all(end_time <= occupancy for occupancy in channel_occupancy):
            all_busy_count += 1

    return all_busy_count / num_simulations

busy_probability = probability()
print(f"Estimated probability that all channels are busy at the 10th hour: {busy_proba
```

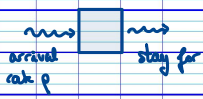
occupy the free channel with new occupancy end time

checks if all channels are busy at the end

```
Estimated probability that all channels are busy at the 10th hour: 0.1700
```

Problem 2:

a)



So total number of arrivals by time t is $N(t) \overset{\text{dist.}}{=} \text{Pois}(\rho t)$

Each customer stays a time $T_i \in [0,t]$

If a customer arrives at time $s < t$, at time t, they'll still be in the system if $T_i \geq t-s$.

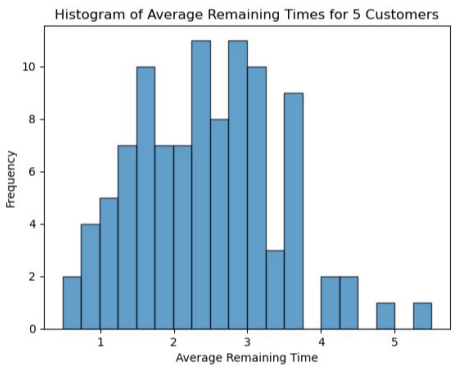
ie the probability of a customer that arrived before t, to still be in the system at time t is:

$$P(T_i \geq t-s) = \frac{t-s}{t}$$ since the T_i's are uniformly distributed

So, the total number of customers in the system at time t is a Binomial RV : $\text{Bin}(N(t), 1/2)$ (trials; probability of coming out before t as a mean)

$$\mathbb{E}\left[\frac{t-s}{t}\right] = \frac{1}{t}\int_0^t \frac{t-s}{t}\,ds = \frac{1}{t}\left(t - \frac{t}{2}\right) = \frac{1}{2}$$

b) Initialize observation time t
   For each customer: Sample an arrival time $S \sim \text{Unif}(0,t)$
                      Given S, the remaining time in the system is $\sim \text{Unif}(0, t-S)$
   return the remaining times




```python
rho = 1
t = 10
num_customers = 5
num_replications = 100

average_remaining_times = []

for _ in range(num_replications):
    remaining_times = []

    for _ in range(num_customers):
        arrival_time = np.random.uniform(0, t)

        remaining_time = np.random.uniform(0, t - arrival_time)

        remaining_times.append(remaining_time)

    average_remaining_times.append(np.mean(remaining_times))

plt.hist(average_remaining_times, bins=20, edgecolor='k', alpha=0.7)
plt.title("Histogram of Average Remaining Times for 5 Customers")
plt.xlabel("Average Remaining Time")
plt.ylabel("Frequency")
plt.show()
```

Problem 3:

a) 1. Inter-arrival time approach:

- Find λ such that $\lambda(t) \leq \lambda$, $\forall t \in [0, 10]$ (3)
- Set $t, I = 0, 0$
- while $t < 10$:
  - Generate $U \sim Unif(0,1)$
  - $t \mathrel{-}= \frac{1}{\lambda} \log U$ where $\lambda = 3$ ITM exp
  - Generate $V \sim Unif(0,1)$
  - If $V \leq \frac{\lambda(t)}{\lambda}$: $I \mathrel{+}= 1$, $A(I) = t$
- return $I-1$ and $A[:-1]$

2. Conditional generation approach:

- Find λ such that $\lambda(t) \leq \lambda$, $\forall t \in [0, 10]$ (3)
- Generate $N(10) \sim Pois(20)$ (λT)
- Suppose $N(10) = n$, generate n iid $U(0,1)$: $U_1, \dots, U_n$
- Sort $U_1, \dots, U_n$ to get : $U_{(1)} < \dots < U_{(n)}$
- Set $A_i = T U_{(i)}$ for $i = 1, \dots, n$
- Generate n iid $V_i \sim Unif(0,1)$
- If $V_i \leq \frac{\lambda(A_i)}{\lambda}$: put A_i into A
- return A

b) In an inhomogeneous Poisson process, if we know the total number of arrivals, the individual arrival times are distributed independently with a density proportional to the rate $\lambda(t)$ over the interval. So, given 2 arrival times by time t, each arrival is independently drawn from $2 + \cos(2\pi t)$. This means, arrivals are more likely when $\lambda(t)$ is higher.

Pseudo-code:

- Generate $N(10) \sim Poisson\left(\int_0^{10} \lambda(t)\,dt\right)$; $\Delta = 2 \times 10 + \frac{1}{2\pi}[\sin(2\pi t)]_0^{10} = 20$
- $N \overset{def}{=} Pois(\Delta)$
- if $N > 0$:
  - $U \sim Unif(0, N)$
  - T_i = Inverse CDF(U, Δ, T)
  - add T_i to arrival times
- return sorted(arrival times), N

```python
def lambda_t(t):
    return 2 + np.cos(2 * np.pi * t)

# Inter-arrival time approach to simulate the inhomogeneous Poisson process
def simulate_interarrival_approach(T=10, lambda_max=3):
    times = []
    current_time = 0
    while current_time < T:
        U = np.random.uniform(0, 1)
        delta_t = -np.log(U) / lambda_max
        current_time += delta_t
        if current_time >= T:
            break
        if np.random.uniform(0, 1) <= lambda_t(current_time) / lambda_max:
            times.append(current_time)
    return times

# Estimate E[∫_0^10 N(t) dt] using replication
def estimate_time_averaged_customers(replications=100, T=10):
    integrals = []
    for _ in range(replications):
        arrival_times = simulate_interarrival_approach(T)

        # Add start and end times for convenience
        arrival_times = [0] + arrival_times + [T]

        # Sum over intervals between arrival times
        integral_estimate = sum((arrival_times[i+1] - arrival_times[i]) * i
                                for i in range(len(arrival_times) - 1)) / T
        integrals.append(integral_estimate)

    # Average across replications
    return np.mean(integrals)

# Run the estimation
expected_time_averaged_customers_interarrival = estimate_time_averaged_customers()
print(f"Expected time-averaged number of customers (inter-arrival): {expected_time_
```

```
Expected time-averaged number of customers (inter-arrival): 9.6898
```

```python
import numpy as np

def cumulative_intensity(T):
    return 2 * T  # Since integral of lambda(t) over [0, T] is 2*T

def simulate_conditional_representation(T=10):
    Lambda_T = cumulative_intensity(T)
    N_T = np.random.poisson(Lambda_T)
    arrival_times = np.sort(np.random.uniform(0, T, N_T))
    return arrival_times

# Run the estimation
expected_time_averaged_customers_conditional = estimate_time_averaged_customers()
print(f"Expected time-averaged number of customers (conditional): {expected_time_av
```

```
Expected time-averaged number of customers (conditional): 10.1266
```

```python
import numpy as np

def inverse_cdf(U, T=10):
    Lambda_T = cumulative_intensity(T)
    lower, upper = 0, T
    epsilon = 1e-6
    while upper - lower > epsilon:
        mid = (lower + upper) / 2
        F_mid = (2 * mid + (1 / (2 * np.pi)) * np.sin(2 * np.pi * mid)) / Lambda_T
        if F_mid < U:
            lower = mid
        else:
            upper = mid
    return (lower + upper) / 2

def simulate_alternative_method(T=10):
    Lambda_T = cumulative_intensity(T)
    N_T = np.random.poisson(Lambda_T)
    arrival_times = [inverse_cdf(np.random.uniform(0, 1), T) for _ in range(N_T)]
    return sorted(arrival_times)

# Run the estimation
expected_time_averaged_customers_alternative = estimate_time_averaged_customers()
print(f"Expected time-averaged number of customers (alternative): {expected_time_av
```

```
Expected time-averaged number of customers (alternative): 10.0416
```