Centre universitaire Abedelhafid Boussouf – Mila
Institut des sciences et technologie
Département de mathématiques et informatique
1er année Master : Intelligence artificielle et ses applications – Année: 2023-2024
Matière : Modélisation et simulation
Resp. Dr. Sadek Benhammada

TP 3

Exercice 01

L'objectif de ce TP est la réalisation d'un simulateur en langage Python d'une file d'attente M/M/1, i.e., avec un seul serveur, des arrivées des clients distribués suivant une loi Poisson, et les durées de service distribuées suivant une loi exponentielle. Le simulateur reçois en entrée le taux d'arrivée des clients λ, le taux de service μ, et une durée de simulation « duration », puis, calcule et affiche les mesures de performance suivantes :

1. Le nombre moyen de clients dans le système ($\overline{N}$)
2. Le nombre moyen de clients en attente ($\overline{N}_Q$)
3. Le temps d'attente moyen dans le système ($\overline{T}$))
4. Le temps d'attente moyen dans la file d'attente ($\overline{T}_Q$)
5. Le temps d'attente moyen dans la file d'attente ($\overline{T}_S$)
6. Le taux d'utilisation du serveur.

Pour vous guider dans l'implémentation du simulateur, une partie du code est fournie, accompagnée d'un tableau explicatif détaillant l'utilité de chaque variable dans le code.

Variables de simulation	
clock	L'horloge de la simulation
queue	File d'attente : liste contenant les temps d'arrivée des clients en attente.
next_arrival	Le temps du prochain événement d'arrivée, initialisé à partir d'une distribution exponentielle.
next_departure	Le temps du prochain événement de départ, initialement défini à l'infini. S'il n'y a pas de départ planifié, l'initialisation à l'infini permet de vérifier le test `next_arrival <= next_departure`, afin d'exécuter le prochain arriver planifié.
busy	Un booléen indiquant si le serveur est occupé ou non
Métriques du temps	
T	Temps moyen de séjour des clients dans le système
Tq	Temps moyen passé dans la file d'attente
Ts	Temps moyen de service.
Métriques de file d'attente	
N	Nombre moyen de clients dans le système
Nq	Nombre moyen de clients dans la file d'attente
Ns	Nombre moyen clients en service.
SU	Coefficient d'utilisation du serveur : Proportion du temps pendant laquelle le serveur est occupé.
Variables utilisées pour le calcul des métriques	
sumN	La somme pondérée des nombres de clients dans le système.
nbN	Le nombre de clients actuellement dans le système.
upN	Les temps des mises à jour du variable **nbN**
sumQ	La somme pondérée par le temps, des nombres de clients dans la file d'attente
nbQ	Le nombre de clients actuellement en file d'attente.
upQ	Les temps des mises à jour du variable **nbQ**
sumS	La somme pondérée des nombres de clients en service
upS	Les temps des mises à jour du variable **sumS**.
sumT, sumTq, sumTs	Les sommes respectives des temps de séjour, d'attente, et de service de tous les clients.
nb_arr, nb_dep	Compteurs d'arrivées et de départs de clients respectivement.

```
def exponential(rate):
    return ……
   # implémenter la fonction exponentielle qui génère des valeurs X ~> Exp(rate), en appliquant la méthode inverse



def simulate(lam, mu, end_time):
    # Initialisation des paramètres de la simulation
    clock = 0  # Horloge de la simulation
    queue = []  # File d'attente des entités
    next_arrival = exponential(lam)  # Prochain événement d'arrivée
    next_departure = float('inf')  # Prochain événement de départ
    T, Tq, Ts = 0, 0, 0  # Initialisation des métriques temporelles
    Nq, N = 0, 0  # Initialisation des métriques liées à la file d'attente
    SU = 0  # Initialisation de l'utilisation du serveur
    nb_arr, nb_dep = 0, 0  # Compteurs d'arrivées et de départs

    # Variables supplémentaires pour le suivi calcul métriques
    sumN, nbN, upN, sumQ, nbQ, upQ, sumS, upS = 0, 0, 0, 0, 0, 0, 0, 0
    sumT, sumTq, sumTs = 0, 0, 0
    busy = False  # booléen indiquant si le serveur est occupé

    # Boucle de simulation
    while queue or next_arrival <= end_time or next_departure != float('inf'):
        if (next_arrival <= next_departure and next_arrival < end_time):
            # Si l'evenement suivant est l'arrivée d'un client à next_arrival, exécution de l'événement d'arrivée
            clock = next_arrival

            # Mise à jour des variable pour le suivi du nombre de dans le système
            sumN = sumN + nbN * (clock - upN)  # Mise à jour de la somme sumN : On ajoute le produit du nombre de clients dans
le système (nbN) par la durée écoulée (clock - upN)
            nbN = nbN + 1  # Incrémentation du du nombre de clients dans le systeme nbN (arrivée d'un client)
            upN = clock  # Mise à jour de upN : Affectation de la valeur actuelle de l'horloge pour conserver le temps de la
dernière mise à jour de nbN

            nb_arr += 1  # Incrémentation du compteur d'arrivées

            # Planification du prochain événement d'arrivée
            interArrT = exponential(lam)
            next_arrival = next_arrival + interArrT

            if not busy:  # Si le serveur est libre, servir du client (Exécution du Debut de service)
                # Planification du prochain départ
```

```
                serviceT = ……
                next_departure = ……

                # Mise à jour des métrique du temps
                sumT = …… # Mise à jour du temps total de séjour dans le systeme
                sumTs =…… # Mise à jour du temps total de service
                upS = …… # Conservation du  temps de début d'utilisation du serveur
# Le temps total d'attente "sumQ" n'est pas mise à jour ici, car le client est servi immédiatement (son temps d'attente = 0)

                busy = …… # Mettre le serveur à l'état actif
            else:
                # Si le serveur est occupé, mettre le client arrivant en file d'attente
                queue.append(clock)  # Mettre le temps d'arrivée en file d'attente
                sumQ = …… # Mise à jour du temps total d'attente sumQ
                ……# Incrémentation du du nombre de clients en attente
                ……# Mise à jour de upQ

        else:
            # Exécution de l'événement de départ à next_departure
            clock = ……

            # Mise à jour des métriques pour le nombre de clients dans le système
            sumN = …… # Mise à jour de la somme sumN
            nbN = ……
            upN = ……

            …… # Incrémentation du compteur de départs

            if nbQ > 0:
            # Si la file d'attente n'est pas vide, servir la prochain client en attente (Exécution du Debut de service)
                sumQ = ……
                ……
                ……
                first = queue.pop(0)  # Retirer le client servie de la file d'attente
                ……)  # Generation du temps de départ du client

                # Mise à jour des métriques pour le temps total, le temps en file d'attente et le temps de service
                sumT = ……   # Mise à jour du temps total de sejour dans le systeme
                sumTq = ……   # Mise à jour du temps total d'attente
                sumTs = …… # Mise à jour du temps total de service

            else:
                # Si la file d'attente est vide, mettre à jour les métriques pour l'utilisation du serveur
                sumS += clock - upS
                # upS = clock
```

```
            busy = ……  # Mettre le serveur à l'état inactif
            next_departure = ……  # Réinitialiser le temps de départ (aucun départ planifié)

# Calculer et afficher les résultats de la simulation
N = ……
Nq = ……
SU = ……
T = ……
Tq = ……
Ts = ……
print(" ==================Simulateur du modele M/M/1====================")
print("N =", round(N, 3))
print("Nq =", round(Nq, 3))
print("T =", round(T, 3))
print("Tq =", round(Tq, 3))
print("Ts =", round(Ts, 3))
print("Utilisation du serveur :", round(SU, 3))

return N, Nq, T, Tq, Ts, SU
```