实验一 排队系统仿真实验

姓名 连宇昊　　学号 2020210140　　班级 01012001

一、 实验目标

通过 MM1、MMk 排队系统的仿真和比较，理解和掌握离散事件的仿真建模方法。

二、 实验原理

1. 离散事件系统仿真的原理

① 离散时间系统的概念

离散事件系统是指系统状态只在离散时刻发生变化的系统。(指受事件驱动、系统状态跳跃式变化的动态系统，系统迁移发生在一串离散事件点上）

② 离散事件仿真的基本要素

•实体（Entity）：实体是系统的成分，如出纳员、顾客、机床、毛胚等。临时实体和永久实体。临时实体，如银行服务系统中的顾客，永久实体，如银行服务系统中的出纳员；

•属性(Attribute)： 每一实体具有的有效特征称为实体属性，如银行服务系统中顾客到达时间，队列中等待的优先级，出纳员忙与闲等；

•状态（State）：实体状态是指在某一时刻该实体的所有属性值，系统状态由系统中各实体的状态合成。在银行系统中，一般状态有服务台忙闲，等待服务的顾客数，队列长度等；

•活动（Activity）：任何使系统状态发生某种变化的过程或行为。一般活动具有一定的持续时间，如顾客在银行中接受出纳员服务的过程称为服务活动；

•事件（Event）：改变系统状态的某一瞬时事变称为事件。事件通常发生在活动的开始或结束时刻；

•进程（Process）：实体的进程是由若干个该实体若干事件及活动构成的，它包含了这些事件和活动的逻辑关系和时序关系，例如，顾客的到达，排队、再经过出纳员服务到离去构成了一个进程。

③ 离散事件系统仿真基本模式：

•面向时间间隔的仿真方法：仿真时钟按较小且相同的时间间隔向前推进，推进一步需判断在所推进的这一时间间隔内是否有事件发生，并依此来改变系统状态变量值。

•面向事件的仿真方法：仿真时钟每次推进到系统中下一事件将要发生的时刻。因为两个事件的发生间隔通常是非固定的，并大多具有随机性，因此每次时钟推进的时间间隔是不同的。由于面向时间间隔的仿真方法中，在每一时间间隔内发生的事件都被当作在该时间间隔的末端发生来处理，这为仿真带来了误差。该方法的另一个缺点是当两个相邻事件间的时间相差比所给定的时间间隔大很多时，仿真器可能要连续推进多个时间间隔，但并不会产生任何影响，系统的状态也不会产生任何改变，无疑这会降低仿真系统运行的效率。因此，当今离散事件仿真多采用面向事件的仿真方法。面向时间间隔的仿真方法适用于系统中状态变量很多时的场合。

④ 仿真钟的推进方法

•固定步长推进法：选择适当的时间单位 T 作为仿真钟推进时的增量，每推进一步作如下处理：该步内若无事件发生，则仿真钟再推进一个时间单位 T；若在该步内有若干个事件发生，则认为这些事件均发生在该步的结束时刻。为便于进行各类事件处理，用户必须规定

当出现这种情况时各类事件处理的优先顺序。

•变步长推进法：即事先没有确定时钟推进步长，而是根据随机事件的发生而进行随机步长的推进，推进的步长为最后已发生事件与下一事件之间的时间间隔。

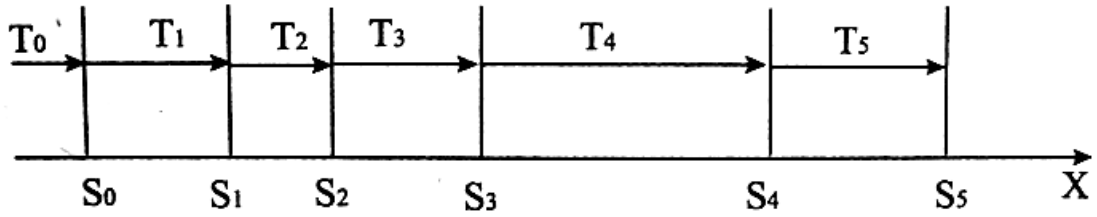

图 1 变步长推进法示意图

2. MM1、MMk 排队系统的结构

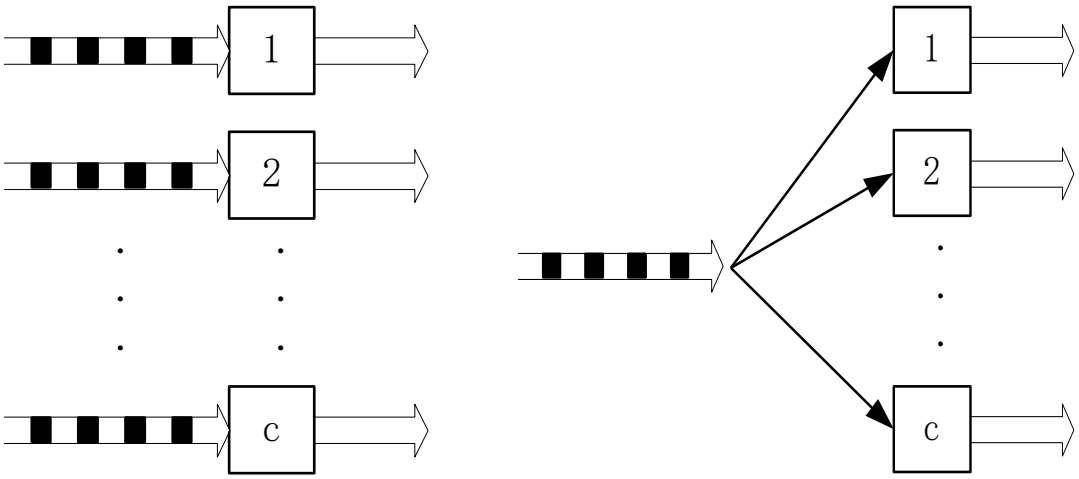

图 2 MM1、MMk 排队系统的结构示意图（左：多个 MM1 排队系统 右：MMk 排队系统）

3. MM1、MMk 排队系统的重要参数推导

① M/M/1 模型的统计特征分析

a 顾客到达模式

实体（临时实体）到达模式：顾客。实体到达模式是顾客到达模式，设到达时间间隔 A1 服从均值β_A的指数分布。

$$f(A)=\frac{1}{\beta_A}e^{-A/\beta_A}$$

b 服务模式

设服务员为每个顾客服务的时间为 Si，它也服从指数分布，均值为β_s。

$$f(S)=\frac{1}{\beta_s}e^{-S/\beta_S}$$

c 服务规则

由于是单服务台系统，考虑系统顾客按单队排列，并按 FIFO 方式服务。

d 理论分析结果

在该系统中，设$\rho = \frac{\lambda}{\mu}$，则稳态时的平均等待队长为$Q = \frac{\rho\lambda}{1-\rho}$，顾客的平均等待时间为$T = \frac{\rho}{\mu-\lambda}$。

关于顾客在系统中逗留的时间 W（随机变量），在 M/M/1 情形下，它服从参数为 μ-λ 的负指数分布，即 $W_s = E(W) = \frac{1}{\mu-\lambda}$

② M/M/k 模型的统计特征分析

对于 MMk 系统，平均逗留时间和平均排队时间的表达式与 MM1 系统类似，即顾客的平均等待时间为$T = \frac{\rho}{\mu-\lambda}$。关于顾客在系统中逗留的时间 W（随机变量），在 M/M/1 情形下，它服从参数为 μ-λ 的负指数分布，即$T = \frac{\rho}{\mu-\lambda}$ $W_s = E(W) = \frac{1}{\mu-\lambda}$。

但是平均队长的表达式会有所不同。具体来说，对于 MMk 系统，平均队长的表达式为：

$$L = \lambda E(T) + \frac{\lambda^k P_0 (k\rho)^k}{\left(k! * (1-\rho)\right)}$$

三、 算法设计

Input: Arrival rate λ, Service rate μ, Simulation time T
Output: Average number of customers in the system L, Average waiting time W
initialize $t = 0$, $n = 0$, $A = 0$, $D = \infty$, $L = 0$, $W = 0$; **while** $t < T$ **do**
  if $A < D$ **then**
    $t = A$; $n \leftarrow n + 1$; $A \leftarrow t - \ln(U)/\lambda$; **if** $D = \infty$ **then**
      $D \leftarrow t - \ln(U)/\mu$;
    end
    $L \leftarrow L + (n-1)(t - T_{last})$; $T_{last} \leftarrow t$;
  end
  else
    $t = D$; $n \leftarrow n - 1$; **if** $n > 0$ **then**
      $D \leftarrow t - \ln(U)/\mu$; $W \leftarrow W + (t - A)$;
    end
    else
      $D \leftarrow \infty$;
    end
  end
end
$L \leftarrow L/T$; $W \leftarrow W/n$;

Algorithm 1: MM1 Queueing System Simulation

Input: $\lambda\ \mu\ k\ T$
Output: $L\ W$
$t \leftarrow 0$, $n \leftarrow 0$, $A \leftarrow 0$, $D_i \leftarrow \infty$ for $i = 1, 2, \ldots, k$, $L \leftarrow 0$, $W \leftarrow 0$;
while $t < T$ **do**
  if $A < \min(D_1, D_2, \ldots, D_k)$ **then**
    $t \leftarrow A$; $n \leftarrow n + 1$; $A \leftarrow t - \ln(U)/\lambda$; **if** $n \leq k$ **then**
      $i \leftarrow theidleserver$; $D_i \leftarrow t - \ln(U)/\mu$;
    end
    $L \leftarrow L + (n-1)(t - T_{last})$; $T_{last} \leftarrow t$;
  end
  else
    $t \leftarrow \min(D_1, D_2, \ldots, D_k)$;
    $n \leftarrow n - thenumberofserversthatfinishserviceatt$; **for** $i = 1, 2, \ldots, k$ **do**
      if $D_i = t$ **then**
        if $n \geq k$ **then**
          $D_i \leftarrow \infty$;
        end
        else
          $D_i \leftarrow t - \ln(U)/\mu$; $W \leftarrow W + (t - A)$;
        end
      end
    end
  end
end
$L \leftarrow L/T$; $W \leftarrow W/n$;

Algorithm 2: MMk Queueing System Simulation

四、 仿真结果分析

1. 给定 lambda=0.2，以 mu 为变量（0.3、0.4…..0.8），画出 MM1 的平均逗留时间、平均排队时间、系统中平均队长的仿真曲线，分析曲线的走势图。

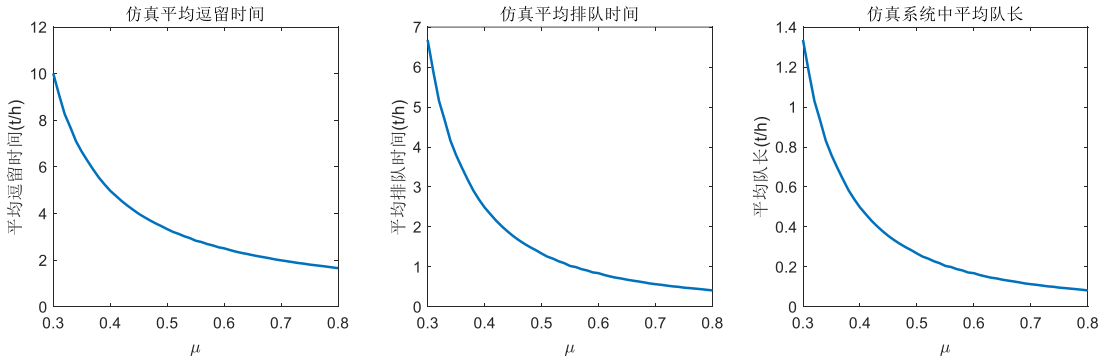

图 3 MM1 排队系统受 μ 的影响图

由上图 3 所示，随着 μ 不断变大，即平均服务速率会增大，所以此时排队时间、系统中平均队长均会随着 μ 的增大二减小，增加了排队系统的效率。

2. 给定 mu=0.8，以 lambda 为变量（0.1、0.2…..0.6），画出 MM1 的平均逗留时间、平均排队时间、系统中平均队长的仿真曲线，分析曲线的走势图。

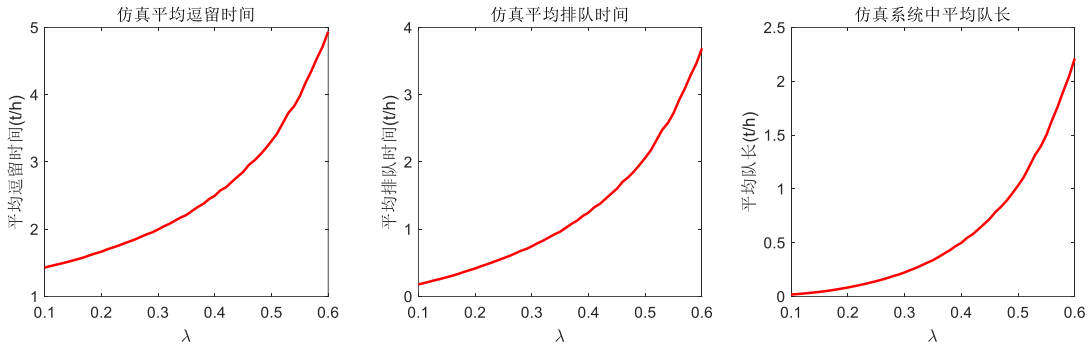

图 4 MM1 排队系统受 λ 的影响图

由上图 3 所示，随着 λ 不断变大，即平均到达速率会增大，所以此时排队时间、系统中平均队长均会随着 λ 的增大而增大，增加了排队系统的复旦。

3. 当 lambda=0.9，mu=0.4 时，如果系统是由 3 个 MM1 组成，或者 1 个 MM3 组成，分别得到两种系统各自的平均逗留时间、平均排队时间、系统中平均队长的仿真值与理论值，并对结果做比较，分析 MM3 的优势。

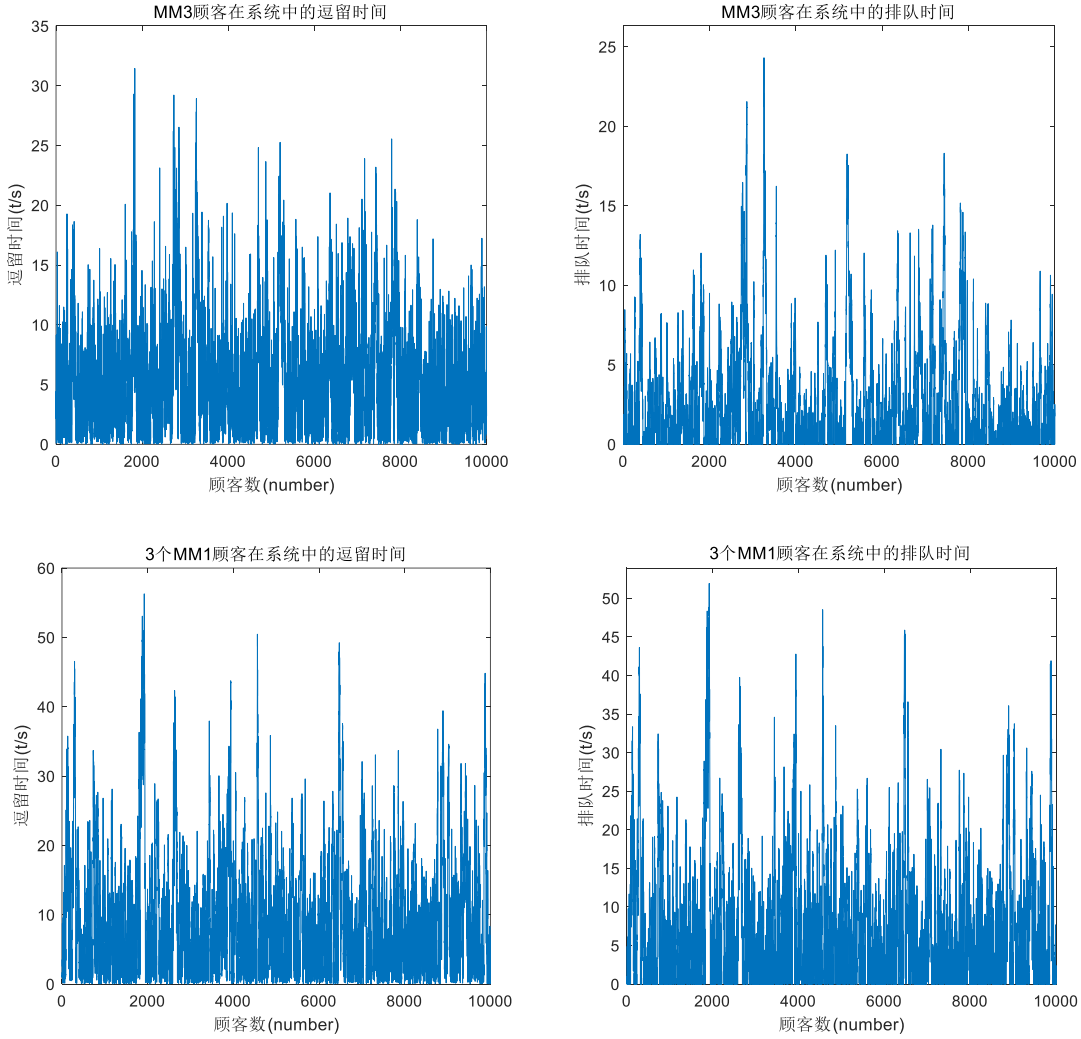

图 5 3 个 MM1 排队系统与 MM3 系统对比分析图

表 1 MM3 排队系统与 3 个 MM1 排队系统性能对比表

性能指标	MM3 排队系统	3 个 MM1 排队系统
理论平均逗留时间	**4.3925**	10
仿真平均逗留时间	**4.3887**	10.0668
理论平均排队时间	**1.8925**	7.5
仿真平均排队时间	**1.889**	7.5669
理论系统中平均队长	**1.7033**	2.25
仿真系统中平均队长	**1.6997**	2.2711

如图 5 以及表 1 所示，各指标的对比可以看出：单队比 3 个队有显著的优越性，即其余参数相同时，MM3 排队系统与 3 个 MM1 排队系统性能相比，MM3 排队系统拥有更高效的结果，这是在安排排队方式时应注意的问题。

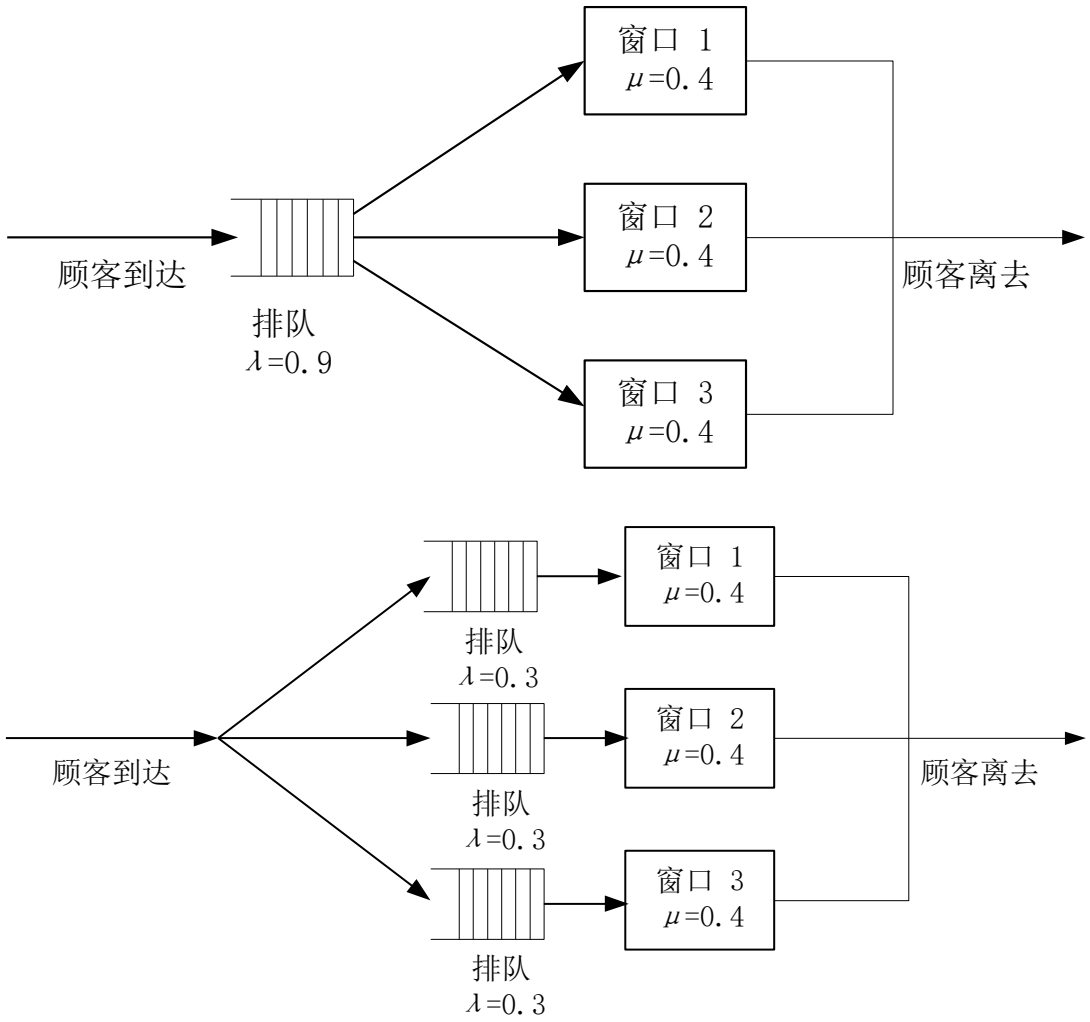

图 5 3 个 MM1 排队系统与 MM3 系统对比示意图

五、 实验心得

经过本次的 MM1 与 MMk 排队系统的仿真实验，我对排队系统的性能评估有了更深入的了解，并学习到了如何使用仿真工具进行排队系统的建模与仿真。

在仿真过程中，我们需要根据排队系统的具体特征来选择合适的仿真模型，比如针对服务时间随机、到达时间随机的排队系统，我们可以选择使用离散事件仿真（DES）模型进行仿真。其次，我们需要根据仿真结果来评估排队系统的性能，包括系统的平均等待时间、平均逗留时间、系统的利用率等指标。通过对这些指标的评估，我们可以了解到系统的优劣，并对系统进行改进和优化。

对于这门课程的建议，我认为可以进一步加强仿真实验的环节，增加更多实际案例和实验操作，让学生能够更好地理解排队系统的应用和实现过程，并提高自己的仿真能力和实践能力。同时，也可以引入一些新的排队模型和仿真工具，让学生能够更加全面地了解和掌握排队系统的相关知识。

六、 源代码

封装函数：[t_Wait_avg,t_Queue_avg,QueLength_avg]=MM1_Queue(Quene_Para)
输入 input：Quene_Para:结构体表示队列的参数
输出 output：t_Wait_avg,t_Queue_avg,QueLength_avg：仿真最终的平均逗留时间、平均排队时间、系统中平均队长
功能：MM1 排队系统的仿真

```
function [t_Wait_avg,t_Queue_avg,QueLength_avg] = MM1_Queue(Quene_Para)

Sim_Customer_Num = Quene_Para.Sim_Customer_Num;
Lambda=Quene_Para.Lambda;
Mu=Quene_Para.Mu;
Time_Arrive=Quene_Para.Time_Arrive;
Time_Leave=Quene_Para.Time_Leave;
Arrive_Num=Quene_Para.Arrive_Num;
Leave_Num=Quene_Para.Leave_Num;
Interval_Arrive=Quene_Para.Interval_Arrive;
Interval_Serve=Quene_Para.Interval_Serve;

Interval_Arrive=-log(rand(1,Sim_Customer_Num))/Lambda;%到达时间间隔
Interval_Serve=-log(rand(1,Sim_Customer_Num))/Mu;%服务时间
Time_Arrive(1)=Interval_Arrive(1);%顾客到达时间

```

```
ArriveNum(1)=1;
for i=2:Sim_Customer_Num
    Time_Arrive(i)=Time_Arrive(i-1)+Interval_Arrive(i);
    ArriveNum(i)=i;
end

Time_Leave(1)=Time_Arrive(1)+Interval_Serve(1);%顾客离开时间
LeaveNum(1)=1;
for i=2:Sim_Customer_Num
    if Time_Leave(i-1)<Time_Arrive(i)
        Time_Leave(i)=Time_Arrive(i)+Interval_Serve(i);
    else
        Time_Leave(i)=Time_Leave(i-1)+Interval_Serve(i);
    end
    LeaveNum(i)=i;
end

t_Wait=Time_Leave-Time_Arrive; %各顾客在系统中的逗留时间
t_Wait_avg=mean(t_Wait);
t_Queue=t_Wait-Interval_Serve;%各顾客在系统中的排队时间
t_Queue_avg=mean(t_Queue);

Timepoint=[Time_Arrive,Time_Leave];%系统中离开事件、到达事件发生的时间
Timepoint=sort(Timepoint);
ArriveFlag=zeros(size(Timepoint));%到达时间标志
CusNum=zeros(size(Timepoint));%到达或者离开事件发生时系统的顾客数量
temp=2;
CusNum(1)=1;
for i=2:length(Timepoint)
    if (temp<=length(Time_Arrive))&&(Timepoint(i)==Time_Arrive(temp))%这里隐含着任何事件不可能同时发生的假设
        CusNum(i)=CusNum(i-1)+1;
        temp=temp+1;
        ArriveFlag(i)=1;
    else
        CusNum(i)=CusNum(i-1)-1;
    end
end
%计算任何相邻事件的时间间隔
Time_interval=zeros(size(Timepoint));
Time_interval(1)=Time_Arrive(1);
for i=2:length(Timepoint)
    Time_interval(i)=Timepoint(i)-Timepoint(i-1);
end
%系统中平均顾客数计算
CusNum_fromStart=[0 CusNum];
CusNum_avg=sum(CusNum_fromStart.*[Time_interval 0] )/Timepoint(end);

%计算事件发生时系统队列的长度
```

```
QueLength=zeros(size(CusNum));
for i=1:length(CusNum)
    if CusNum(i)>=2
        QueLength(i)=CusNum(i)-1;%系统中只有 1 个服务台
    else
        QueLength(i)=0;
    end
end
QueLength_avg=sum([0 QueLength].*[Time_interval 0] )/Timepoint(end);%系统平均队长

%仿真值与理论值比较
% disp(['理论平均逗留时间=',num2str(1/(Mu-Lambda))]);
% disp(['理论平均排队时间=',num2str(Lambda/(Mu*(Mu-Lambda)))]);
% disp(['理论系统中平均队长=',num2str(Lambda*Lambda/(Mu*(Mu-Lambda)))]);
%
% disp(['仿真平均逗留时间=',num2str(t_Wait_avg)])
% disp(['仿真平均排队时间=',num2str(t_Queue_avg)])
% disp(['仿真系统中平均队长=',num2str(QueLength_avg)]);
end
```

主函数：完成 MM1 在不同仿真参数 λ、μ 下的仿真与绘图

```
clc;
clear;

%% ---------------- Queue Control Parameters ---------------%
% Sim_Customer_Num=input('请输入仿真顾客总数'); %仿真顾客总数；
% Lambda=input('请输入仿真参数 Lambda'); %到达模式服从参数为 λ 的泊松分布；
% Mu=input('请输入仿真参数 Mu'); %服务时间服从参数为 μ 的指数分布；
%λ 小于 μ
Sim_Customer_Num = 10000;%仿真顾客总数；
Lambda=0.2; %到达模式服从参数为 λ 的泊松分布；
Mu=0.8;%服务时间服从参数为 μ 的指数分布；

%% ---------------- Initialization of Parameters ---------------%
Time_Arrive=zeros(1,Sim_Customer_Num);
Time_Leave=zeros(1,Sim_Customer_Num);
Arrive_Num=zeros(1,Sim_Customer_Num);
Leave_Num=zeros(1,Sim_Customer_Num);
Interval_Arrive=zeros(1,Sim_Customer_Num);
Interval_Serve=zeros(1,Sim_Customer_Num);

Quene_Para = struct('Sim_Customer_Num', Sim_Customer_Num, 'Lambda', Lambda,'Mu',Mu, 'Time_Arrive', Time_Arrive,...
```

```
    'Time_Leave', Time_Leave, 'Arrive_Num', Arrive_Num, 'Leave_Num', Leave_Num, 'Interval_Arrive', Interval_Arrive,...
    'Interval_Serve', Interval_Serve);

save('Quene_Para', 'Quene_Para');


%% ---------------- Queue Generation ---------------%
[t_Wait_avg,t_Queue_avg,QueLength_avg] = MM1_Queue(Quene_Para);

%% ---------------- μ Simulation ---------------%
%---------------- Initialization of Simulation Parameters ----------------%
iter = 100;
Quene_Para.Lambda=0.2;
Mu_Simulation = 0.3:0.01:0.8;
Mu_Wait_avg_iter = zeros(iter,size(Mu_Simulation,2));
Mu_Queue_avg_iter = zeros(iter,size(Mu_Simulation,2));
Mu_QueLength_avg_iter = zeros(iter,size(Mu_Simulation,2));

for i = 1:iter
    Mu_Wait_avg = zeros(size(Mu_Simulation));
    Mu_Queue_avg = zeros(size(Mu_Simulation));
    Mu_QueLength_avg = zeros(size(Mu_Simulation));
    for j =1:size(Mu_Simulation,2)
        fprintf('== Mu is %g and iteration is %g == \n',Mu_Simulation(j),i)
        Quene_Para.Mu=Mu_Simulation(j);
        [Mu_Wait_avg(j),Mu_Queue_avg(j),Mu_QueLength_avg(j)] = MM1_Queue(Quene_Para);
    end
    Mu_Wait_avg_iter(i,:)=Mu_Wait_avg;
    Mu_Queue_avg_iter(i,:)=Mu_Queue_avg;
    Mu_QueLength_avg_iter(i,:)=Mu_QueLength_avg;
end
Mu_Wait_avg = sum(Mu_Wait_avg_iter)/iter;
Mu_Queue_avg = sum(Mu_Queue_avg_iter)/iter;
Mu_QueLength_avg = sum(Mu_QueLength_avg_iter)/iter;

figure;
subplot(1,3,1);
plot(Mu_Simulation,Mu_Wait_avg,'LineWidth',1.5,'MarkerSize',12);
xlabel('\mu');
ylabel('平均逗留时间(t/h)');
```

```
62. title('仿真平均逗留时间');
63. subplot(1,3,2);
64. plot(Mu_Simulation,Mu_Queue_avg,'LineWidth',1.5,'MarkerSize',12);
65. xlabel('\mu');
66. ylabel('平均排队时间(t/h)');
67. title('仿真平均排队时间');
68. subplot(1,3,3);
69. plot(Mu_Simulation,Mu_QueLength_avg,'LineWidth',1.5,'MarkerSize',
        12);
70. xlabel('\mu');
71. ylabel('平均队长(t/h)');
72. title('仿真系统中平均队长');
73.
74. %% ---------------- λ Simulation ---------------%
75. %---------------- Initialization of Simulation Parameters -------
        --------
76. iter = 100;
77. Quene_Para.Mu=0.8;
78. Lambda_Simulation = 0.1:0.01:0.6;
79. Lambda_Wait_avg_iter = zeros(iter,size(Lambda_Simulation,2));
80. Lambda_Queue_avg_iter = zeros(iter,size(Lambda_Simulation,2));
81. Lambda_QueLength_avg_iter = zeros(iter,size(Lambda_Simulation,2))
        ;
82.
83. for i = 1:iter
84.     Lambda_Wait_avg = zeros(size(Lambda_Simulation));
85.     Lambda_Queue_avg = zeros(size(Lambda_Simulation));
86.     Lambda_QueLength_avg = zeros(size(Lambda_Simulation));
87.     for j =1:size(Lambda_Simulation,2)
88.         fprintf('== Lambda is %g and iteration is %g == \n',Lambd
        a_Simulation(j),i)
89.         Quene_Para.Lambda=Lambda_Simulation(j);
90.         [Lambda_Wait_avg(j),Lambda_Queue_avg(j),Lambda_QueLength_
        avg(j)] = MM1_Queue(Quene_Para);
91.     end
92.     Lambda_Wait_avg_iter(i,:)=Lambda_Wait_avg;
93.     Lambda_Queue_avg_iter(i,:)=Lambda_Queue_avg;
94.     Lambda_QueLength_avg_iter(i,:)=Lambda_QueLength_avg;
95. end
96. Lambda_Wait_avg = sum(Lambda_Wait_avg_iter)/iter;
97. Lambda_Queue_avg = sum(Lambda_Queue_avg_iter)/iter;
98. Lambda_QueLength_avg = sum(Lambda_QueLength_avg_iter)/iter;
99.
100. figure;
```

```
subplot(1,3,1);
plot(Lambda_Simulation,Lambda_Wait_avg,'r','LineWidth',1.5,'Marke
        rSize',12);
xlabel('\lambda');
ylabel('平均逗留时间(t/h)');
title('仿真平均逗留时间');
subplot(1,3,2);
plot(Lambda_Simulation,Lambda_Queue_avg,'r','LineWidth',1.5,'Mark
        erSize',12);
xlabel('\lambda');
ylabel('平均排队时间(t/h)');
title('仿真平均排队时间');
subplot(1,3,3);
plot(Lambda_Simulation,Lambda_QueLength_avg,'r','LineWidth',1.5,'
        MarkerSize',12);
xlabel('\lambda');
ylabel('平均队长(t/h)');
title('仿真系统中平均队长');
```

封装函数：[t_Wait_avg,t_Queue_avg,QueLength_avg]=MMK_Queue(Quene_Para)

输入 input：Quene_Para:结构体表示队列的参数

输出 output：t_Wait_avg,t_Queue_avg,QueLength_avg：仿真最终的平均逗留时间、平均排队时间、系统中平均队长

功能：MMK 排队系统的仿真

```
function [t_Wait_avg,t_Queue_avg,QueLength_avg] = MMK_Queue(Quene
        _Para)

Sim_Customer_Num = Quene_Para.Sim_Customer_Num;
Lambda=Quene_Para.Lambda;
Mu=Quene_Para.Mu;
K = Quene_Para.K;
Time_Arrive=Quene_Para.Time_Arrive;
Time_Leave=Quene_Para.Time_Leave;
Interval_Arrive=Quene_Para.Interval_Arrive;
Interval_Serve=Quene_Para.Interval_Serve;

%按照分布，初始化到达间隔和服务间隔
Interval_Arrive=-log(rand(1,Sim_Customer_Num))/Lambda;
Interval_Serve=-log(rand(1,Sim_Customer_Num))/Mu;
%通过累计求和，计算顾客的达到时刻;
Time_Arrive = cumsum(Interval_Arrive);

%定义时间列表的三个参量的索引值:
%事件时刻，事件类型（1 代表到达，-1 代表离开），客户编号
ETIME = 1; ETYPE = 2; ECUST_NO = 3;
```

```
%初始化事件列表，加入第一个客户到达的事件
evList(1,ETIME) = Time_Arrive(1);
evList(1,ETYPE) = 1;
evList(1,ECUST_NO) = 1;
%初始化队列
qList = [];
%下一个到达的客户编号；
nextN = 2;
%繁忙服务台数量
busyN = 0;
%按照时间加权求和的队列长度
nQ = 0;
%队列长度上一次变化的时刻
tQ = 0;
%主逻辑，以事件列表非空作为循环条件
while(~isempty(evList))
    %到达事件
    if (evList(1,ETYPE) == 1)
        %如果有服务台空闲
        if (busyN < K)
            %繁忙服务台加一
            busyN = busyN + 1;
            %设置该客户的离开事件，按逻辑应该是删除到达事件，再添加离开事件
            %此处优化为直接修改事件类型和时刻，
            evList(1,ETIME ) = evList(1,ETIME) + Interval_Serve(evList(1,ECUST_NO));
            evList(1,ETYPE) = -1;
            %服务台没有空闲
        else
            %队列长度将发生变化，累积前一段时刻的队列长度求和
            nQ = nQ + length(qList)*(evList(1,ETIME) - tQ);
            tQ = evList(1,ETIME);
            %将顾客排入队列
            qList(end+1) = evList(1, ECUST_NO);
            %删除该顾客达到事件
            evList(1,:) = [];
        end
        %如果还有客户将要到达，处理新顾客到达事件，加入事件列表
        if (nextN <= Sim_Customer_Num)
            evList(end+1, ETIME) = Time_Arrive(nextN);
            evList(end, ETYPE) = 1;
            evList(end, ECUST_NO) = nextN;
            nextN = nextN + 1;
```

```
63.          end
64.          %离开事件
65.      else
66.          %繁忙服务台减一
67.          busyN = busyN - 1;
68.          %记录该顾客的离开事件
69.          Time_Leave(evList(1, ECUST_NO)) = evList(1, ETIME);
70.          %如果队列中有顾客
71.          if (~isempty(qList))
72.              %队列长度将发生变化，累积前一段时刻的队列长度求和
73.              nQ = nQ + length(qList)*(evList(1,ETIME) - tQ);
74.              tQ = evList(1,ETIME);
75.              %增加繁忙服务台数
76.              busyN = busyN + 1;
77.              %添加该顾客的离开事件
78.              evList(end+1, ETIME) = evList(1, ETIME) + Interval_Se
           rve(qList(1));
79.              evList(end, ETYPE) = -1;
80.              evList(end, ECUST_NO) = qList(1);
81.              %将该顾客移除等待队列
82.              qList(1) = [];
83.          end
84.          %删除该顾客的离开事件
85.          evList(1,:) = [];
86.      end
87.      %对事件列表排序，默认首列升序，即按照时间顺序排序
88.      evList = sortrows(evList);
89.  end
90.
91.
92.  %各顾客在系统中的逗留时间
93.  t_Wait=Time_Leave-Time_Arrive;
94.  t_Wait_avg=mean(t_Wait);
95.  %各顾客在系统中的排队时间
96.  t_Queue=t_Wait-Interval_Serve;
97.  t_Queue_avg=mean(t_Queue);
98.
99.  QueLength_avg = nQ/Time_Leave(end);
100.
101. %仿真值与理论值比较
102. ro = Lambda /(K*Mu);
103. k=0:(K-1);
104. P0 = (sum(1./factorial(k).*(Lambda/Mu).^k) +...
105.     1/factorial(K)/(1-ro)*(Lambda/Mu)^K)^(-1);
```

```
Lq = (K*ro)^K*ro/factorial(K)/(1-ro)^2*P0;
Ls = Lq + Lambda/Mu;
Wq = Lq/Lambda;
Ws = Ls/Lambda;

figure;
subplot(1,2,1);
plot(1:Sim_Customer_Num,t_Wait);
xlabel('顾客数(number)')
ylabel('逗留时间(t/s)')
title('顾客在系统中的逗留时间')
subplot(1,2,2);
plot(1:Sim_Customer_Num,t_Queue);
xlabel('顾客数(number)')
ylabel('排队时间(t/s)')
title('各顾客在系统中的排队时间')
ylim([0 max(t_Queue)+2]);

disp(['理论平均逗留时间=',num2str(Ws)]);
disp(['理论平均排队时间=',num2str(Wq)]);
disp(['理论系统中平均队长=',num2str(Lq)]);

disp(['仿真平均逗留时间=',num2str(t_Wait_avg)])
disp(['仿真平均排队时间=',num2str(t_Queue_avg)])
disp(['仿真系统中平均队长=',num2str(QueLength_avg)]);
end

```

主函数：完成 MM3 与 3 个 MM1 系统的对比

```
clc;
clear;

%% ---------------- Queue Control Parameters ---------------%%
% Sim_Customer_Num=input('请输入仿真顾客总数'); %仿真顾客总数;
% Lambda=input('请输入仿真参数 Lambda'); %到达模式服从参数为 λ 的泊松分布;
% Mu=input('请输入仿真参数 Mu'); %服务时间服从参数为 μ 的指数分布;
%λ 小于 μ
Sim_Customer_Num = 10000000;%仿真顾客总数;
Lambda=0.9; %到达模式服从参数为 λ 的泊松分布;
Mu=0.4;%服务时间服从参数为 μ 的指数分布;
K=3;%服务台数量 K

```

```
%% ---------------- Initialization of Parameters ---------------%
      %
Time_Arrive=zeros(1,Sim_Customer_Num);
Time_Leave=zeros(1,Sim_Customer_Num);
Interval_Arrive=zeros(1,Sim_Customer_Num);
Interval_Serve=zeros(1,Sim_Customer_Num);

Quene_Para = struct('Sim_Customer_Num', Sim_Customer_Num, 'Lambda
      ', Lambda,'Mu',Mu,'K',K, 'Time_Arrive', Time_Arrive,...
    'Time_Leave', Time_Leave, 'Interval_Arrive', Interval_Arrive,
      'Interval_Serve', Interval_Serve);

save('Quene_Para', 'Quene_Para');


%% ---------------- Queue Generation ---------------%%
%---------------- MM3 Queue ---------------%
[t_Wait_avg,t_Queue_avg,QueLength_avg] = MMK_Queue(Quene_Para);

%---------------- 3*MM1 Queue ---------------%
Quene_Para.K = 1;
Quene_Para.Lambda = 0.3;
[t_Wait_avg_1,t_Queue_avg_1,QueLength_avg_1] = MMK_Queue(Quene_Pa
      ra);
```