组成、特征

一般的排队过程都由**输入过程、排队规则、服务过程**三部分组成

根据排队规则和服务规则的不同可分为多中模型。

运行指标

(i)平均队长：指系统内顾客数（包括正被服务的顾客与排队等待服务的顾客）的 数学期望，记作 Ls 。

(ii)平均排队长:指系统内等待服务的顾客数的数学期望，记作 Lq 。

(iii)平均逗留时间：顾客在系统内逗留时间（包括排队等待的时间和接受服务的 时间）的数学期望，记作Ws 。

（iv）平均等待时间：指一个顾客在排队系统中排队等待时间的数学期望，记作 Wq 。

（v）平均忙期：指服务机构连续繁忙时间（顾客到达空闲服务机构起，到服务机 构再次空闲止的时间）长度的数学期望，记为Tb 。

基本参数

队长、排队长、逗留时间、等待时间、忙期（顾客等待）概率=服务强度等

模型（时间间隔按照泊松流分布）

单服务台、多服务台（顾客不会离去）

M / M / s/ s 损失制排队模型

当 s 个服务台被占用后，顾客自动离去。

混合型模型

系统容量有限的情况，当系统容量足够，顾客就排队；不够就离开

以上的模型均有不同的公式

```
'''
    排队论仿真
'''

import numpy as np
```

```
import matplotlib.pyplot as plt

class Q(object):
    def __init__(self, total_time, N, lambd, mu):
        self.total_time = total_time
        self.N = N
        # 平均到达时间和平均服务时间
        self.arr_mean = 1/lambd
        self.ser_mean = 1/mu
        # 仿真时间内总的顾客数
        self.arr_num = round(total_time*lambd*2)
        # 按负指数分布产生各顾客达到时间间隔
        self.customer_time = np.random.exponential(self.arr_mean, self.arr_num)
        # 各顾客的到达时刻等于时间间隔的累积和
        self.customer_time = np.array(np.cumsum(self.customer_time))
        # 按负指数分布产生各顾客服务时间
        self.serve_time = np.random.exponential(self.ser_mean, self.arr_num)
        # 计算仿真顾客数，到达时刻在仿真时间内的顾客数
        self.customer_num = np.sum(self.customer_time <= total_time)
        self.wait_time = None
        self.all_time = None

    def simulation(self):
        # 计算第1个顾客的信息，直接服务，不用等待
        self.wait_time = np.zeros(self.customer_num)
        self.wait_time[0] = 0
        # 其离开时刻等于其到达时刻与服务时间之和
        self.all_time = np.zeros(self.customer_num)
        self.all_time[0] = self.customer_time[0] + self.wait_time[0]
        # 此时系统内共有
        system_customer = np.zeros(self.customer_num)
        # 记录第i个顾客到达时候的顾客数
        system_customer[0] = 1
        member = [0]
        number = 0

        for i in range(1, self.arr_num):
            # 如果第 i个顾客的到达时间超过了仿真时间，则跳出循环
            if self.customer_time[i] > self.total_time:
                break
            else:
                for j in member:
                    if self.all_time[j] > self.customer_time[i]:
                        number = number + 1
                # 系统已满
                if number >= self.N+1:
                    system_customer[i] = 0

                # 系统未满
                else:
                    if number == 0:
                        self.wait_time[i] = 0
                        self.all_time[i] = self.customer_time[i] +
self.serve_time[i]
                        system_customer[i] = 1
                        member.append(i)
                    else:
                        len_men = len(member)
```

```
                        self.wait_time[i] = self.all_time[member[len_men-1]] -
self.customer_time[i]
                        self.all_time[i] = self.all_time[member[len_men-1]] +
self.serve_time[i]
                        system_customer[i] = number + 1
                        member.append(i)
        return len(member)

    def draw(self):
        # 绘图
        # 设置分辨率
        plt.rcParams['figure.dpi'] = 100
        print(self.simulation())
        y = np.arange(0, self.simulation(), 1)
        plt.plot(self.customer_time[0:self.simulation()], y, 'r', label='arrive
time')
        plt.plot(self.all_time, y, 'g4--', label='leave time')
        plt.legend()
        plt.grid()
        plt.show()
        plt.plot(self.wait_time,'y', label='wait time')
        plt.plot(self.serve_time[0:self.simulation()] + self.wait_time, 'g4--',
label='stop time')
        plt.legend()
        plt.grid()
        plt.show()

if __name__ == '__main__':
    # 仿真时间、队伍长度、到达、服务
    qq = Q(10, 10000000000, 15,8)
    qq.draw()
```

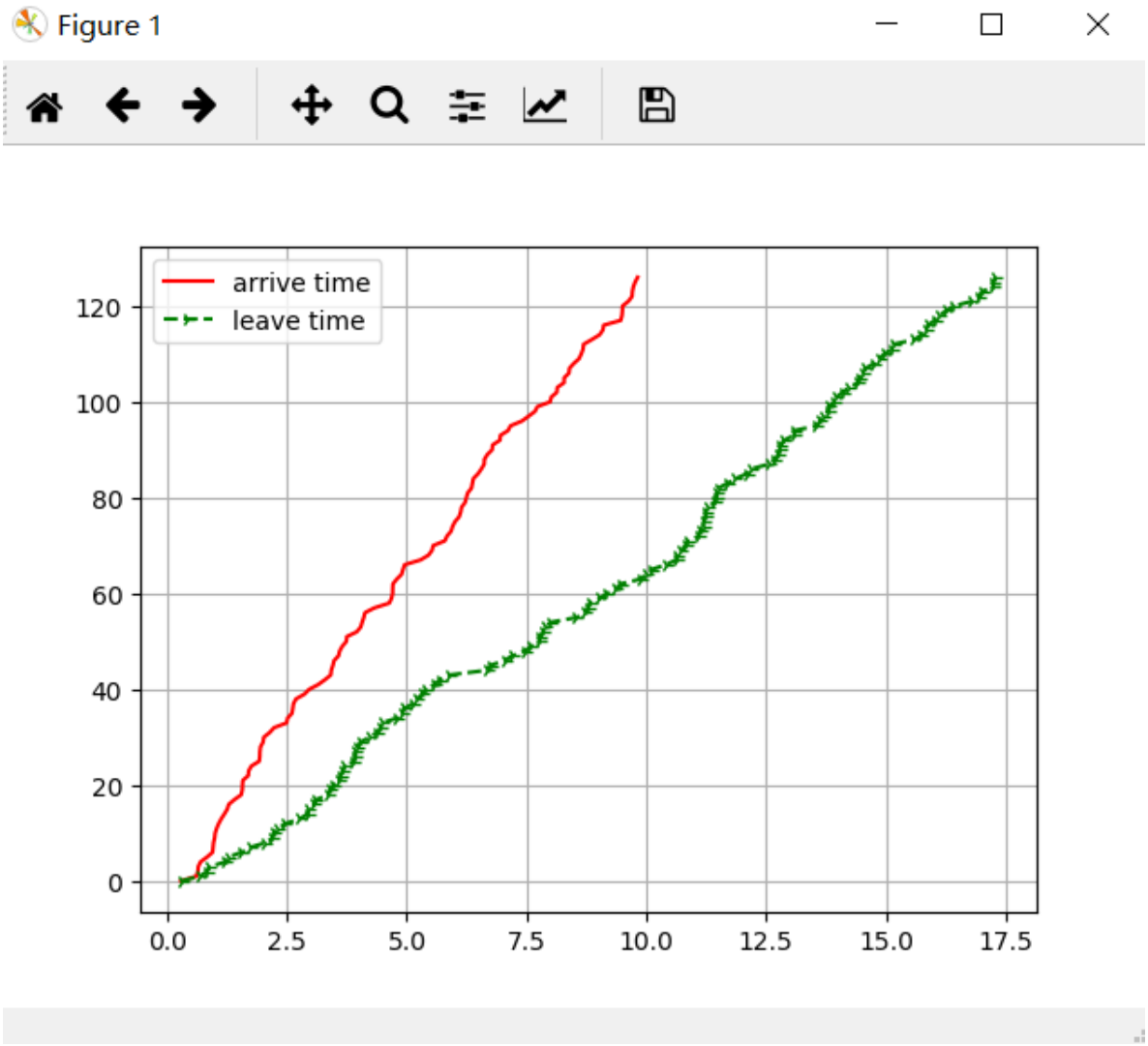
Figure 1
arrive time
leave time
120
100
80
60
40
20
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

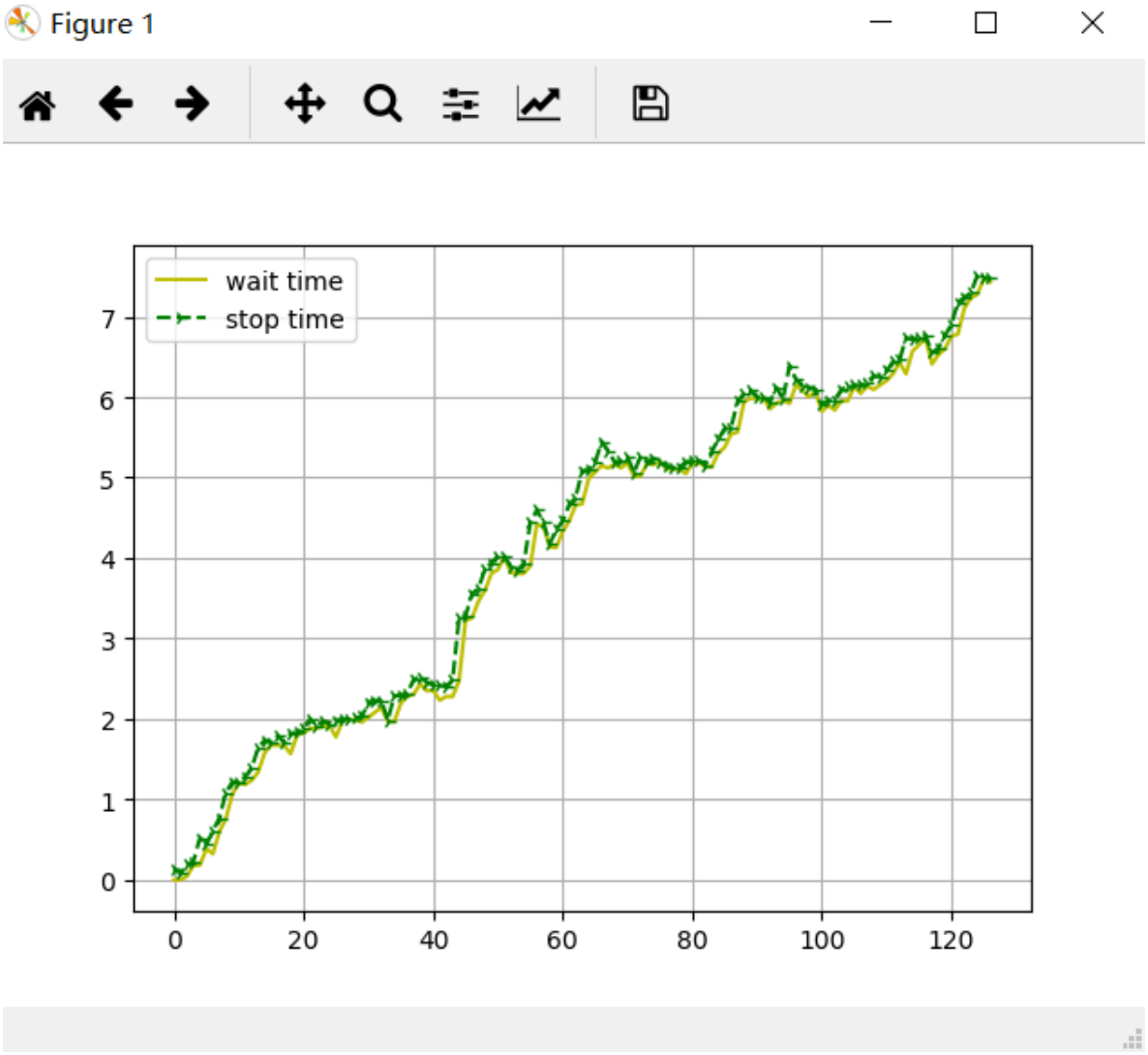
Figure 1
wait time
stop time
0
1
2
3
4
5
6
7
0
20
40
60
80
100
120