这是一个jupyter notebook脚本，用于演示仿真结果

1 单灶台的仿真结果

1.1 平均队列长度

系统中的顾客到达为一个泊松流，且间隔时间分布为到达率为λ的负指数分布。 单个灶台做完一顿黄焖鸡的时间固定为τ_0 则可以得到队列平均长度为：

$$\bar{k} = \rho + \frac{\lambda^2 b^2}{2(1-\rho)} = \frac{\rho(2-\rho)}{2(1-\rho)} = \frac{2\lambda\tau_0 - (\lambda\tau_0)^2}{2-2\lambda\tau_0}$$

In [1]:

```
import plotly
import plotly.graph_objects as go
import pandas as pd
import math
```

定义平均队列长度计算函数

In [2]:

```
def func_k_avg(tau_, lambda_):
    rho = tau_ * lambda_
    k_avg = rho * (2 - rho)/(2 *(1 - rho))
    return k_avg
```

定义测试数据集

顾客到达率λ从0.009人/分钟到0.09人/分钟

黄焖鸡制作时间τ_0从6分钟到15分钟

注意为了保持系统稳定，要使$\rho = \lambda \times \tau_0$小于1

In [3]:

```
lambda_list = [round(0.009 * (i+1), 3) for i in range(10) ]
print(lambda_list)
```

```
[0.009, 0.018, 0.027, 0.036, 0.045, 0.054, 0.063, 0.072, 0.081, 0.09]
```

In [4]:

```
tau_list = [1 * (i+1) for i in range(10) ]
print(tau_list)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

初始化数据帧用于存放计算结果

In [5]:

```
k_avg = pd.DataFrame(columns = lambda_list, index = tau_list)
k_avg
```

Out[5]:

	0.009	0.018	0.027	0.036	0.045	0.054	0.063	0.072	0.081	0.090
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

计算结果并存放进数据帧

In [6]:

```
for tau_ in k_avg.index:
    for lambda_ in k_avg.columns:
        k_avg.loc[tau_, lambda_] = func_k_avg(tau_, lambda_)
k_avg
```

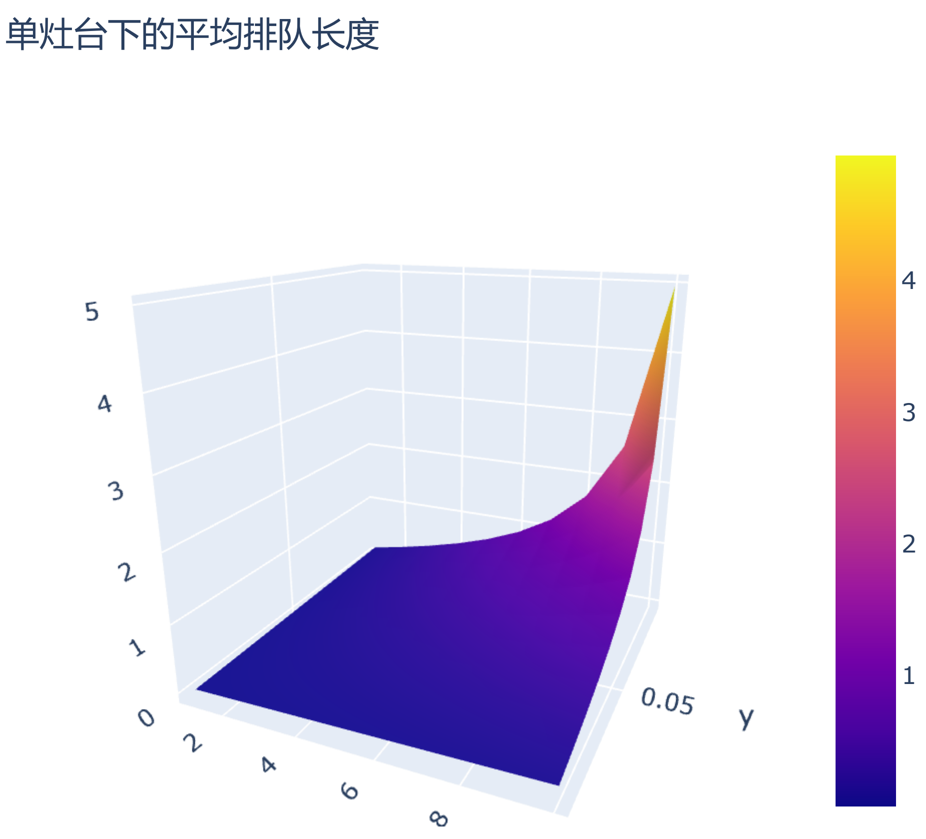
Out[6]:

	0.009	0.018	0.027	0.036	0.045	0.054	0.063	0.072	0.081
1	0.009041	0.018165	0.027375	0.036672	0.04606	0.055541	0.065118	0.074793	0.08457
2	0.018165	0.036672	0.055541	0.074793	0.094451	0.114538	0.135082	0.156112	0.177659
3	0.027375	0.055541	0.08457	0.114538	0.145535	0.177659	0.211023	0.245755	0.282002
4	0.036672	0.074793	0.114538	0.156112	0.199756	0.245755	0.294449	0.346247	0.401645
5	0.04606	0.094451	0.145535	0.199756	0.257661	0.319932	0.387427	0.46125	0.542836
6	0.055541	0.114538	0.177659	0.245755	0.319932	0.401645	0.492859	0.596282	0.715763
7	0.065118	0.135082	0.211023	0.294449	0.387427	0.492859	0.614954	0.760065	0.938234
8	0.074793	0.156112	0.245755	0.346247	0.46125	0.596282	0.760065	0.967245	1.244455
9	0.08457	0.177659	0.282002	0.401645	0.542836	0.715763	0.938234	1.244455	1.709518
10	0.094451	0.199756	0.319932	0.46125	0.634091	0.856957	1.166351	1.645714	2.536579

绘制仿真结果

In [7]:

```
fig = go.Figure(data=[go.Surface(z=k_avg.values, x=k_avg.index, y=k_avg.columns)])
fig.update_layout(title='单灶台下的平均排队长度', autosize=False,
                  width=500, height=500,
                  margin=dict(l=65, r=50, b=65, t=90))
fig.show()
```



1.2 平均排队时间

根据M/G/1模型中的推导，可以得到表达式：

$$\bar{w} = \frac{\lambda b^2}{2(1-\rho)} = \frac{b}{2}\frac{\rho}{1-\rho} = \frac{\bar{\tau}}{2}\frac{\rho}{1-\rho} = \frac{\lambda\tau_0}{2\left(1-\lambda\tau_0\right)}\tau_0$$

定义平均排队时间计算表达式

In [8]:

```
def func_w_avg(tau_, lambda_):
    rho = tau_ * lambda_
    w_avg = tau_ * rho/(2 *(1 - rho))
    return w_avg
```

计算结果并存放进数据帧

In [9]:

```
w_avg = pd.DataFrame(columns = lambda_list, index = tau_list)
for tau_ in w_avg.index:
    for lambda_ in w_avg.columns:
        w_avg.loc[tau_, lambda_] = func_w_avg(tau_, lambda_)
w_avg
```

Out[9]:

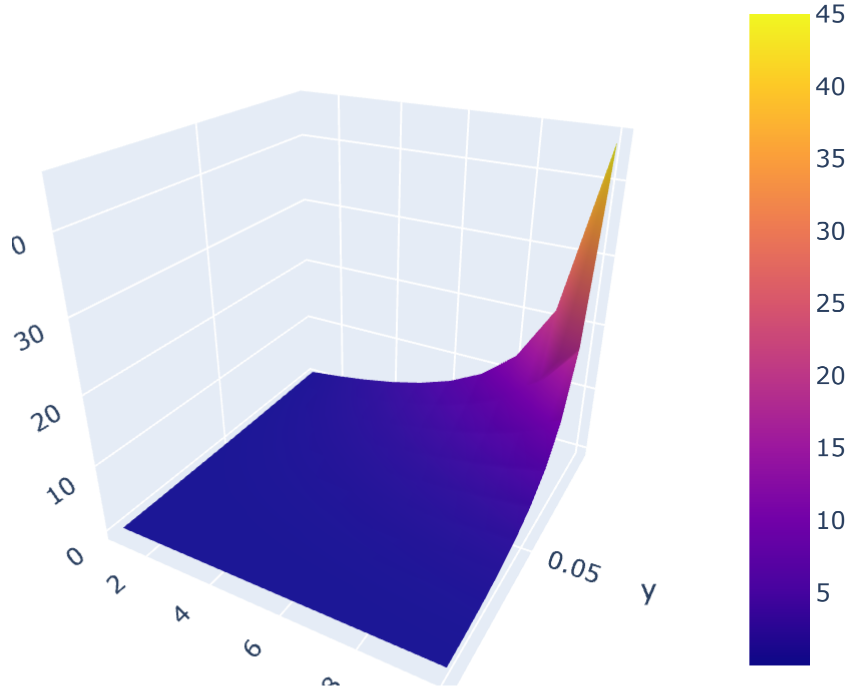
	0.009	0.018	0.027	0.036	0.045	0.054	0.063	0.072	0.081
1	0.004541	0.009165	0.013875	0.018672	0.02356	0.028541	0.033618	0.038793	0.04407
2	0.01833	0.037344	0.057082	0.077586	0.098901	0.121076	0.144165	0.168224	0.193317
3	0.041624	0.085624	0.132209	0.181614	0.234104	0.289976	0.349568	0.413265	0.481506
4	0.074689	0.155172	0.242152	0.336449	0.439024	0.55102	0.673797	0.808989	0.95858
5	0.117801	0.247253	0.390173	0.54878	0.725806	0.924658	1.149635	1.40625	1.701681
6	0.171247	0.363229	0.579952	0.826531	1.109589	1.43787	1.823151	2.28169	2.836576
7	0.235326	0.504577	0.81566	1.179144	1.609489	2.12701	2.761181	3.556452	4.583141
8	0.310345	0.672897	1.102041	1.617978	2.25	3.042254	4.064516	5.433962	7.363636
9	0.396627	0.869928	1.444518	2.156805	3.063025	4.254864	5.89261	8.284091	12.105166
10	0.494505	1.097561	1.849315	2.8125	4.090909	5.869565	8.513514	12.857143	21.315789

绘制结果

In [10]:

```
fig = go.Figure(data=[go.Surface(z=w_avg.values, x=w_avg.index, y=w_avg.columns)])
fig.update_layout(title='单灶台下的平均排队时间', autosize=True,
                  width=500, height=500,
                  margin=dict(l=65, r=50, b=65, t=90))
fig.show()
```

单灶台下的平均排队时间



我们可以从图中很清楚的看到，在单灶台情况下，排队的平均队列长度和平均等待时间会随着顾客到来的强度λ（y）以及做完一个黄焖鸡米饭的服务时间τ_0（x）的增加而增加。并且这种增长趋势并不是线性的，而是在ρ更接近1的时候会极速增长。

2 多灶台分析

在第一部分我们队单服务台的情况进行了仿真，并在仿真中可以看出，当$\lambda = 0.09$，$\tau_0 = 10$时，排队情况已经相当恶化，平均排队长度达到了4.95人，平均排队时间已经高达45分钟，那么此时可以通过增加灶台的方式缓解排队压力。

2.1 平均排队长度

平均队伍长度的计算公式如下

$$L_{\mathrm{q}}^{\mathrm{M}} = \frac{(k\rho)^k \rho}{k!(1-\rho)^2} P_0$$

$$P_0 = \left[\sum_{i=0}^{k-1} \frac{1}{i!} \left(\frac{\lambda}{\mu} \right)^i + \frac{1}{k!} \frac{1}{1-\rho} \left(\frac{\lambda}{\mu} \right)^k \right]^{-1}$$

$$\mu = 1/\tau_0, \quad \rho = \lambda/(k\mu)$$

$$L_q^G = \frac{1}{2} L_q^M$$

2.1 平均排队时间

$$W_q^M = \frac{(k\rho)^k \rho}{k!(1-\rho)^2 \lambda} P_0$$

$$W_q^G = \frac{1}{2} W_q^M$$

列出计算公式函数

In [11]:

```
def fun_Lq(tau_, lambda_, K):
    mu_ = 1/tau_
    rho_ = lambda_/(K*mu_)
    P_0 = 0
    for i in range(K):
        P_0 += 1 / (math.factorial(i)) * math.pow(lambda_/mu_, i)
    P_0 += 1/math.factorial(K) * (1/(1-rho_)) * math.pow(lambda_/mu_, i)
    P_0 = 1 / P_0
    L_M = math.pow(K * rho_, K) * rho_ /(math.factorial(K)*math.pow(1 - rho_, 2)) * P_0

    return L_M/2
#     Lq_up = lambda_*math.pow(tau_,2)/(2*tau_+K-lambda_*tau_)
#     Lq_down = 1
#     for i in range(K):
#         Lq_down += math.factorial(K-1)*(K-lambda_*tau_)/(math.factorial(i) * math.pow(lambda_ * ta
#     return Lq_up/Lq_down

def fun_Wq(tau_, lambda_, K):
    mu_ = 1/tau_
    rho_ = lambda_/(K*mu_)
    P_0 = 0
    for i in range(K):
        P_0 += 1 / (math.factorial(i)) * math.pow(lambda_/mu_, i)
    P_0 += 1/math.factorial(K) * (1/(1-rho_)) * math.pow(lambda_/mu_, i)
    P_0 = 1 / P_0
    L_M = math.pow(K * rho_, K) * rho_ /(math.factorial(K)*math.pow(1 - rho_, 2)) * P_0
    W_M = L_M/lambda_
    return W_M/2
```

由于我们讨论多服务台情况，为了使结果更加明显，在参数设计上继续增加顾客到达强度，注意此时的稳态条件变为：

$$\rho = \lambda/(k\mu) < 1$$

k从3变化到10，则在$\tau_0 = 10$的情况下，λ最大可以是0.27。

In [12]:

```
K_list = [i + 3 for i in range(8)]
K_list
```

Out[12]:

```
[3, 4, 5, 6, 7, 8, 9, 10]
```

In [13]:

```
lambda_ = 0.27
tau_ = 10

L_list = [fun_Lq(tau_, lambda_, k) for k in K_list]
W_list = [fun_Wq(tau_, lambda_, k) for k in K_list]
```

In [14]:

```
L_list
```

Out[14]:

```
[7.572326237496783,
 0.5380955681023253,
 0.1105414088990324,
 0.02779759953283408,
 0.007216536985289514,
 0.001817199270783968,
 0.00043283042228857524,
 9.662762319603705e-05]
```

In [15]:

```
W_list
```

Out[15]:

```
[28.045652731469563,
 1.9929465485271305,
 0.40941262555197183,
 0.10295407234382993,
 0.026727914760331532,
 0.006730367669570251,
 0.001603075638105834,
 0.0003578800859112483]
```

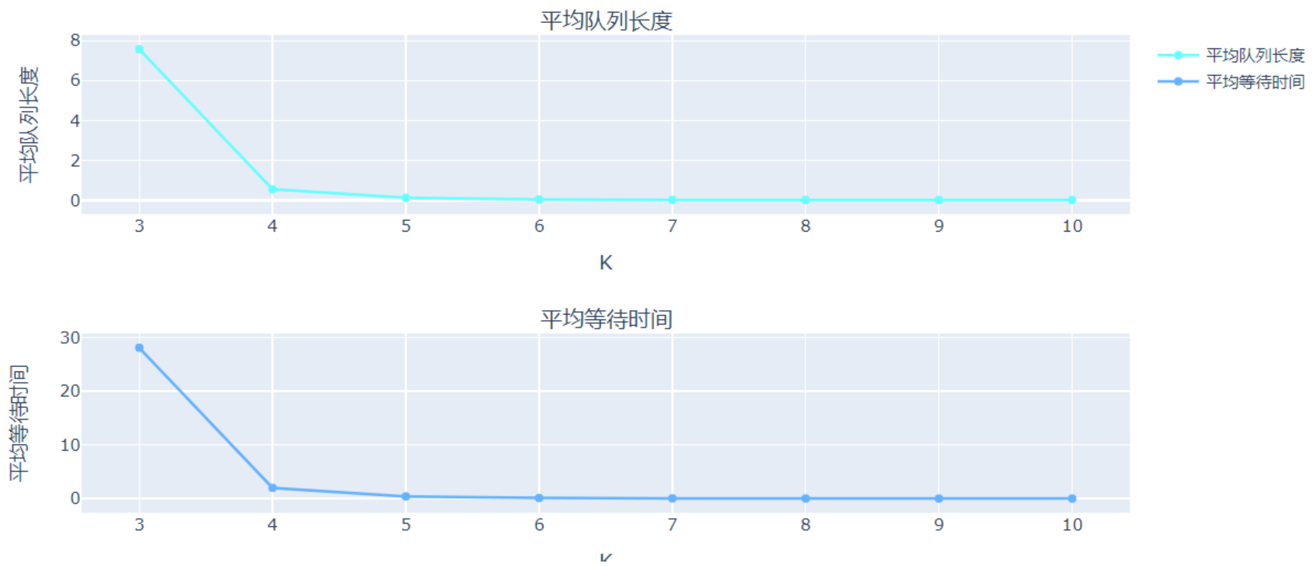
In [17]:

```
trace1 = go.Scatter(
    x = K_list,
    y = L_list,
    mode = "lines+markers",
    name = "平均队列长度",
    marker = dict(color = 'rgb(102,255,255)'),
    )
trace2 = go.Scatter(
    x = K_list,
    y = W_list,
    mode = "lines+markers",
    name = "平均等待时间",
    marker = dict(color = 'rgb(102,178,255)'),
    )
layout = go.Layout(title = 'Line Plot: Mean House Values by Bedrooms and Year',
    xaxis= dict(title= 'K',ticklen= 1,zeroline= False),
    yaxis= dict(title= '平均队列长度',ticklen= 1,zeroline= False))
fig = plotly.subplots.make_subplots(rows=2, cols=1, subplot_titles=("平均队列长度", "平均等待时间",
                                                                  ))
fig.append_trace(trace1, 1, 1)
fig.append_trace(trace2, 2, 1)

fig['layout']['xaxis1'].update(title='K')
fig['layout']['xaxis2'].update(title='K')
fig['layout']['yaxis1'].update(title='平均队列长度')
fig['layout']['yaxis2'].update(title='平均等待时间')
# fig.show()
```

Out[17]:

```
layout.YAxis({
    'anchor': 'x2', 'domain': [0.0, 0.375], 'title': {'text': '平均等待时间'}
})
```



从上图中可以清楚地看到，增加灶台可以显著提高服务效率，当只有3个灶台时，平均队列长度达到了8人，等待时间达到了将近30分钟，这显然是极大程度上降低了用户用餐体验，但是只要增加一个灶台，则平均排队时间就骤降到了不到5分钟，平均队列长度也下降到了一人之内。所以增加灶台数量可以显著提高服务效率。但是要注

意到，之后如果再增加灶台数量，则对等待时间的影响就微乎其微，所以在这里建议只增加一个灶台。