2016 年同济大学数学建模竞赛

题目:游客在游乐场最佳路线选择及酒店房间预订数预测

姓名　　学号　　学院

联系方式：17717093975

摘　要

本文通过满意度的建立与最佳路线的确立及规划设计，对于游乐园的庞大客流进行了合理配置，在使游乐园分布合理并使资源最大化利用的同时，让游客达到体验最高。并且通过对于庞大数据的归纳处理、回归拟合，对于游乐园酒店客流的影响因素进行分析确定，并预测出来年的客流数量。

对于第一个问题，通过建立满意度 V 的函数来量化评价客人感受的指标，此处从游玩时间、等待时间、行走时间以及协调系数四个方面来衡量满意度 V。游玩时间已由表给出，等待时间与行走时间通过分析可给出定义式，而其中涉及到较为复杂的情况与算法，如在行走时间内，进出景点的人数，还有与人流密度、景点热度等因素有关，综合一系列因素考虑；同时，为确定最佳路线，在 floyd 最短距离路线算法上优化，加入速度的变化因素，优化最佳路线的选择；考虑到价值度与游客偏好、景点拥挤度、道路阻抗对于游客心里选择的影响，提出协调系数的概念，并且根据查阅的大量资料可确定协调系数的算法与需要确定的数据；对于以上算法都可用 c++与 matlab 程序计算。

最终我们以游客从入口出发和从 E 景点出发两种情况进行模拟，得出了我们预期的成果：选择了一个最佳景点，并推荐了最佳行走路线。并且通过对比发现，相比于常理中“选择距离最近”、“选择最不拥挤”、“选择自己最喜欢”的一些方案，来对比他们的价值度，发现我们的方案价值度更高，并且与上述任意一种方案的结果不尽相同，我们的方案综合考虑了各种因素，提出了一个平衡所有因素的不同结果与路线选择。

对于第二个问题，首先要将如此杂乱无章的数据进行有序处理。我们用 excel 的函数统计出了每一天的入住量，并制成散点、曲线图观察了大致的分布情况。从大致分布情况与常理出发，确立了“淡旺季”、“暑期”、“双休日”、“节假日”四个主要因素来进行探究，将他们设为我们的回归变量，分别设为 x1，x2，x3，x4。我们根据四元回归变量，统计了对应的情况下的当天订房数，如“淡季，工作日，非暑期，非节假日”的平均入住数量。为了确定回归变量的取值，依据层次分析法得出每种影响因素的权重，确定回归变量的取值。接下来采用统计回归模型，先根据散点图大致确定模型方程种类，再利用 MATLAB 统计工具箱中 regress 函数进行多元回归分析，得出回归系数，并用残差分析方法检验模型的缺陷。由于影响因素中（例如：节假日与工作日）有相互耦合关系，并非独立变量，所以优化过程将引入交互作用项给予模型改正。在优化的过程中，我们对于异常值进行了取舍，最后提高了结果的合理性。

最终的预测数据与函数曲线和我们设想的近乎相似，也与之前 2015 年的数据有非常大的相似度，证明我们的模型是合理的，但同时也存在一些缺陷，如 1 月份前几天突变的地方吻合度较低等。

关键词：时空分流，满意度，数据筛选，回归方程

一、问题重述

1.1 问题一

对于如同 youth 游乐园一样的大型游乐园，每天有大量的客流量涌入园区，因此，对于有限的游玩项目与园区容量，如何合理的分配客流，并提供一套合理的模型方案，从而达到游乐园不会过分拥挤，并且使游客体验最佳的两者动态平衡，是一个亟需解决的问题。

1.2 问题二

对于园区的大型五星级酒店，酒店方需要了解影响酒店入住量的影响因素。而对于已有的全年预订与入住的一组庞大数据，作出数据的归纳处理，以此来分析出一些主要因素，用数学模型确定该因素对于入住量的影响方式与程度，建立出数学模型。并用此模型来预测出下一年的预订与入住人数。

二、问题分析

2.1 问题一

对于问题一中选取使游客体验程度最高，因此需要建立一套标准去量化游客体验。将游客体验用满意度 V 衡量，通过分析影响满意度的各个因素，建立量化模型。

为了使 V 最大，我们需要引导游客一条最佳的游园体验路线建议，使游客在等待时间、游乐偏好、人群拥挤等问题中得到一个最佳的平衡方案，即在任意时刻，对于任意游客 i，在其还未游览过的景点中，为其挑选最适合的景点 n 以及去到 n 并能玩上项目的总时间(由排队时间+行走时间构成)最短的一条路线。为达到这个目的，我们的算法由以下几步组成：

1.确定最短路线：对于游客 i 剩余未游览过的景点$S_1, S_2, \ldots, S_n$中，计算出游客 i 从此时所在地 m 去到对应景点的 n 条行走时间最短的路线，$L_{m\to s1}, L_{s1\to s2,}\ldots$。

2.计算协调系数：综合考虑游客偏好、景点拥挤度、道路阻抗对游客心理的影响，提出一个协调系数，由公式计算出从出发点 m 到未游览过景点$S_1, S_2, \ldots, S_n$分别对应的的协调系数。

3.计算价值度：由公式计算出从出发点 m 到未游览过景点$S_1, S_2, \ldots, S_n$分别对应的价值度。

4.根据价值度比较，选出最佳推荐景点与对应路线。

2.2 问题二

本问题有如下要点：

1.数据处理：将如此庞大且杂乱无章的数据先进行有序处理归纳，观察出数据规律，并根据几种较为直观与合乎常理的因素进行数据观测。

2.确定影响因素：根据数据与因素的相关度确定主要的影响因素，并确定相关因素的定义域。这一步是本问题中最关键与最困难的步骤，因为影响因素诸多且变幻莫测，故需要通过多种尝试与拟合来确定主要的影响因素。

3.回归分析：根据 matlab 与概率统计知识，根据已经确定的影响因素及数

据，用统计回归模型来建立函数曲线来拟合大数据。

三、模型假设

3.1 问题一

1、不存在车辆调度问题，在游乐场中只能步行。

2、所有游客被相同对待。

3、游客每个项目都只会游玩一次，且游客不会走回头路。

4、一个项目的每次运转的时间相同，游客上下所花费的时间相同。

5、所有项目都是满载情况。

6、假设钢筋与 y 轴平行，且抗弯刚度 $EI=\infty$(钢筋不会出现受弯变形).

3.2 问题二

1、全部预订房间没有退房的情况。

2、不考虑酒店促销，天气等随机事件影响。

3、酒店容量足够，不存在没房间的情况。

四、符号说明

符号	符号说明
x_i	游客 i
t_n	某个游玩项目每次运转时间
$t_{mn}^{i}\left(T\right)$	游玩总时间（步行时间+等待时间+游玩时间）
$t_{m\to n}^{i}$	从游玩项目 m 到游玩项目 n 的总时间
C_m	m 景点每场的容纳数
t_n^i	游客 i 在游玩项目 n 的排队等候时间
$t_{nr}\left(T\right)$	T 时刻游玩项目 n 本场已经运转的时间
$P_{in}\left(T\right)$	T 时刻游玩项目 n 的排队人数
P_{in}	新到达人数
P_{out}	离开人数

ρ	偏好度
V	满意度
λ	协调系数

五、模型建立及结果分析

5.1. 问题一

5.1.1 模型建立过程

满意度定义

将任一游客 x_i 在 T 时刻第 j 次做出的决策记为 x_{ij}，即 x_i 在 T 决定下一个游玩地点是。y_1，并将 x_{ij} 作为决策向量 $X_i = \left(x_{i1}, x_{i2}, x_{i3}, \ldots, x_{ij}\right) j \leq 10$，里面每一个元素代表每次游览的项目。

游客在某一 m 游玩项目到 n 游玩项目的价值为 $V_{mn}^{i} = \frac{t_n \times \lambda_{mn}^{i}}{t_{mn}^{i}\left(T\right)}$，$\lambda_{jh}^{i}$ 为协调系数。

若游客 i 此时刻（T）在 y_m 游玩项目，则他在上一次做出的决策为 x_{ij-1}，即 $y_m = x_{ij-1}$，他将做出 j 次决策到下一个游玩项目 $y_n = x_{ij}$。

1. 实时监测数据信息

在计算过程需要利用 RFID 技术进行实时监测游乐场内所有道路的人数密度，建立一个人流密度矩阵（将游玩项目与出入口之间任一连线看成一条道路）

$$\rho_{i,j} = \begin{pmatrix} \rho_{1,1} & \rho_{1,2} & \cdots\cdots & \rho_{1,11} \\ \rho_{2,1} & \rho_{2,2} & \cdots\cdots & \rho_{2,11} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \rho_{11,1} & \rho_{11,2} & \cdots\cdots & \rho_{11,11} \end{pmatrix}$$

由于有 $A \rightarrow J$ 10 个游玩项目，加上出入口有 11 个地点，所以矩阵是 11×11 的阶数，当两个点之间实际没有道路直接到达时 $\rho_{i,j}$=0，中间 t+1 段路程的人流密度 $\rho_{m,s1}$，$\rho_{s1,s2}$，$\rho_{s2,s3}$，……，$\rho_{st,n}$。此外还需要通过检票系统等措施监测出 T 时刻 y_n 游玩项目的排队等候人数 $P_n\left(T\right)$。

2. 选择路线

包括以下几个步骤：

（1）计算行走速度

人步行的速度与年龄，性别，个人健康状况等内在因素有关，还与

当前道路拥挤等外在因素有关，在此定义当前每个人的行走速度都为

$$V = V_0 \rho^{-0.8}$$

V_0 为在空旷路面上人平均行走速度，在此取 $V_0 = 1.34m/s$，ρ 为当前道路的拥挤程度，可以根据检测所得的人流密度而定。根据公式可以求出各段路上的行走速度 $V^i_{m\to s1}(T), V^i_{s1\to s2}(T), \ldots, V^i_{st\to n}(T)$

（2）计算行走时间

从 m 到 n 中要经过若干条线段，如 AB、BC，则经过的任一条线段上的行走时间（如相邻景点 S1 与 S2）

$$t^i_{s1\to s2}(T) = \frac{L_{s1s2}}{V^i_{s1\to s2}(T)}$$

求出从游玩项目 m 到游玩项目 n 过程中每两个相邻游玩项目路步行所花费的时间 $t^i_{m\to s1}(T), t^i_{s1\to s2}(T), \ldots, t^i_{st\to n}(T)$，从而求出游玩项目 m 走到游玩项目 n 的任一路线的总、行走时间:

$$t^i_{m\to n} = \sum_{k=1}^{t} t^i_{sk\to sk+1}(T)$$

（3）选取最优路线

考虑到达某游玩项目步行时间最短原则，取上述求出的所有总行走时间中最小值所对应的路线即为最优路线。

3. 计算排队时间。

（1）计算从游玩项目 m 走到游玩项目 n 时间段 $t^i_{m\to n}$ 中新到达游玩项目 n 的人数 $P^n_{in}(t^i_{m\to n})$。

在行走到下一个游玩项目时，距离决策已经有一段 $t^i_{m\to n}$ 的时间差，在这段时间里会有原来在路上的人到达 n，从而使 n 的排队人数发生改变，这个人数与人群行走速度，人流密度以及该游玩项目是几条道路交汇口有关（将游玩项目所在的点看成一个路口），可以得出单位时间到达游玩项目 n 的人数为

$$P^n_{in} = \sum_{k=1}^{l} \alpha_k V_k \rho_k$$

时间段 $t^i_{m\to n}$ 中新到达游玩项目 n 的人数

$$P^n_{in}(t^i_{m\to n}) = P^n_{in} \cdot t^i_{m\to n} = t^i_{m\to n} \cdot \sum_{k=1}^{l} \alpha_k V_k \rho_k$$

l 为该游玩项目是 l 条道路的交汇口，V_k 是 k 号道路的人群行走速

度，ρ_k 是 k 号道路的人群密度，可以从人流密度矩阵中得出。此外，考虑到不会所有在与 n 交汇的路上的人都进入游玩项目 n，所以还需要乘以比例系数 α_k 。

（2）计算从游玩项目 m 走到游玩项目 n 时间段 $t^i_{m\to n}$ 中，从游玩项目 n 出去的人数 $P^n_{out}(t^i_{m\to n})$ 。

时间段 $t^i_{m\to n}$ 中，从游玩项目 n 出去的人数应该为在该时间段中游玩目 n 运行了几轮，用运行的次数乘以每次容纳人数即为离开人数。

$$P^n_{out}(t^i_{m\to n})=\left[\frac{t^i_{m\to n}-\left(t_n-t_{nr}\left(T\right)\right)}{t_n}+1\right]\cdot C_n$$

$\left[\frac{t^i_{m\to n}-\left(t_n-t_{nr}\left(T\right)\right)}{t_n}+1\right]$ 表示取整，由此可以求出离开人数 $P^n_{out}(t^i_{m\to n})$

（3）计算到达游玩项目 n 时 n 的总人数。

由 T 时刻的人数加上进来的人数减去游玩过后离开的人数。

$$P_n(T+t^i_{m\to n})=P_n\left(T\right)+P^n_{in}(t^i_{m\to n})-P^n_{out}(t^i_{m\to n})$$

（4）计算总排队时间。

到达游玩项目 n 后的排队时间，等于此时排队人数与每次容纳人数的商乘以每次持续时间。

$$t^i_n=\left[\frac{P_n(T+t^i_{m\to n})}{C_n}\right]\cdot t_h+t_n-t_{nr}\left(T+t^i_{m\to n}\right)$$

$$t_{nr}\left(T+t^i_{m\to n}\right)=\left(t^i_{m\to n}-\left(t_n-t_{nr}\left(T\right)\right)\right)\%t_n$$

4. 计算从在 m 做出决策到在能在游玩项目 n 进行游玩的总时间。

总时间应等于步行时间加上排队等待时间。

$$t^i_{mn}\left(T\right)=t^i_n+t^i_{m\to n}$$

5. 计算协调系数。

研究表明，游客在旅游过程中具有一定的行为特征，比如游客会根据自己的偏好选择节点参观、游客经常会回避那些拥挤的景点而选择去游览不拥挤的景点、游客总是倾向先游览离自己距离近的景点。然而，基于任何单一行为特征来选择目标景点的方式有不能很好解决游客在景区内分布不均衡的问题，为此采用协调控制的方法，引入了协调系数 λ^i_{mn} ，该系数表示游客 i 在游玩 m 时选择游玩项目 n 的可能性，该值综合反映了对游

客偏好、景点的拥挤状况和道路阻抗因素的平衡，其值越大说明越能满足游客偏好，景点越不拥挤，道路的阻抗越小。

首先定义游客偏好矩阵 p_{ij}，游客 i 对景点 j 的偏好程度记为 p_{ij}，值越大表明游客 i 越希望到景点 j 进行游览，游客 i 对景区内所有景点偏好数组。游客偏好向量为

$$p_{ij} = \left(p_{i1} \quad p_{i2} \quad \cdots\cdots \quad p_{i10}\right)$$

偏好向量可以事前对游客的采集或者统计而得到，因为本题有十个游玩项目，所以应该有十个元素。

定义道路阻抗，其反映了道路的拥挤程度，从而会影响游客的游览体验。

$$\Omega_{mn} = \frac{t^{i}_{m\to n}\left(T\right)}{K_{mn}}$$

K_{mn} 为 m 到 n 的路线总数。

则协调系数

$$\lambda^{i}_{mn} = \left(\frac{p_{ij}}{\overline{p_i}}\right)^{\alpha} \cdot \left(1 + \frac{t_{nr}\left(T + t^{i}_{m\to n}\right)}{\overline{t_{nr}\left(T + t^{i}_{m\to n}\right)}}\right)^{\beta} \cdot \left(\frac{\overline{\Omega_n}}{\Omega_{mn}}\right)^{\gamma}$$

$\overline{p_i}$ 为游客 i 尚未游览过游玩项目的偏好 p_{ij} 的平均值，$\overline{t_{nr}\left(T + t^{i}_{m\to n}\right)}$ 为游客 i 尚未游览过游玩项目等待时间 $t_{nr}\left(T + t^{i}_{m\to n}\right)$ 平均值，$\overline{\Omega_n}$ 为游客 i 尚未游览过游玩项目的阻抗平均值，平均值均应为统计大量游客在一定时间内的平均值。根据相关文献取 $\alpha = \gamma = 1$，$\beta = -0.3$。

6. 满意度（价值）计算

$$W^{i}_{mn} = \frac{\lambda^{i}_{mn} \cdot t_n}{t^{i}_{mn}\left(T\right)}$$

即可算出从 m 到 n 的满意度，将所有没有游玩过的项目均计算出满意度，取满意度最大的情况，即为当前时刻下最优路线。

为了验证我们模型的准确性，本文将定义不同的初始条件从而得出在此条件下的最优路线并进行比较。

5.1.2 仿真模拟

模型一

初始条件

假定在 T 时刻，游客 i 在游乐园门口进行决策到下一个景点 j，此时每条路的人流密度随机生成，并计算出单位时间内进入各个游玩项目的流量相对值，如下表所示（0 代表出入口）：

游玩项目	单位时间进入相对人数
0	0.24
A	0.61
B	0.56
C	0.49
D	0.36
E	0.83
F	0.4
G	0.55
H	0.37
I	0.42
J	0.33

表 1 单位时间进入景点相对人数

偏好向量的元素取整数，如下表：

游玩项目	偏好系数
0	0
A	4
B	7
C	12
D	9
E	6
F	5
G	8
H	8
I	7
J	6

表 2 偏好向量

所有游玩项目本场运行时间均为 0，即都同时开始运行，设任一项目排队时间与该项目平均排队时间相同。

建模过程

1. 利用到达每个游玩项目的最短距离，结合当前道路拥挤度（人流密度），再考虑到直接能够到达各个游玩项目的路线数量的差异性，利用 matlab 求出到达各个游玩项目的步行时间，再用 C++编程求出两点之间的最佳路线，求出从起点到各个游玩项目的最优路线。

终点	路线图	时间(t/s)
A	0-A	241
B	0-B	368
C	0-B-C	677
D	0-B-C-D	1040
E	0-A-E	564
F	0-B-C-D-F	1817
G	0-A-E-G	1162
H	0-A-J-I-H	1288
I	0-A-J-I	874
J	0-A-J	592

表 3 最快路线表

2. 在行走的时间里，每个游玩项目新到达与游玩结束后离开的人数，利用之前定义的人流密度以及每个项目的持续时间，可以得出以下表格

游玩项目	新到达的人数	离开的人数
A	321	0
B	491	120
C	1550	200
D	1786	180
E	645	100
F	1733	600
G	1773	270
H	1966	420
I	1167	180
J	678	200

表 4 景区人数变动表

A 的离开人数是 0 的原因是游玩项目 A 的持续时间是 33 分钟，比从入口到 A 的时间短，因此走到 A 时 A 的一次运转时间还没结束，所以没有人离开。

3. 到达后游玩项目后排队等候时间

游玩项目	等待时间(t/s)
A	5699
B	2332
C	6373
D	12010
E	2736
F	9133
G	9518
H	8882
I	7586
J	2768

表 5 等待时间

4. 以之前所求的等待时间，计算各条道路阻抗，以及根据之前对平均值的假设，计算出各个景点的协调系数为

游玩项目	协调系数
A	3.119
B	8.02
C	10.55
D	11.58
E	4.02
F	6.143
G	7.64
H	17.02
I	14.39
J	24.11

表 6 协调系数

5. 根据以上数据得出从入口到各个游玩项目游客的满意度值曲线，如下图所示（横坐标 1-10 分别与 A-J 相对应）。

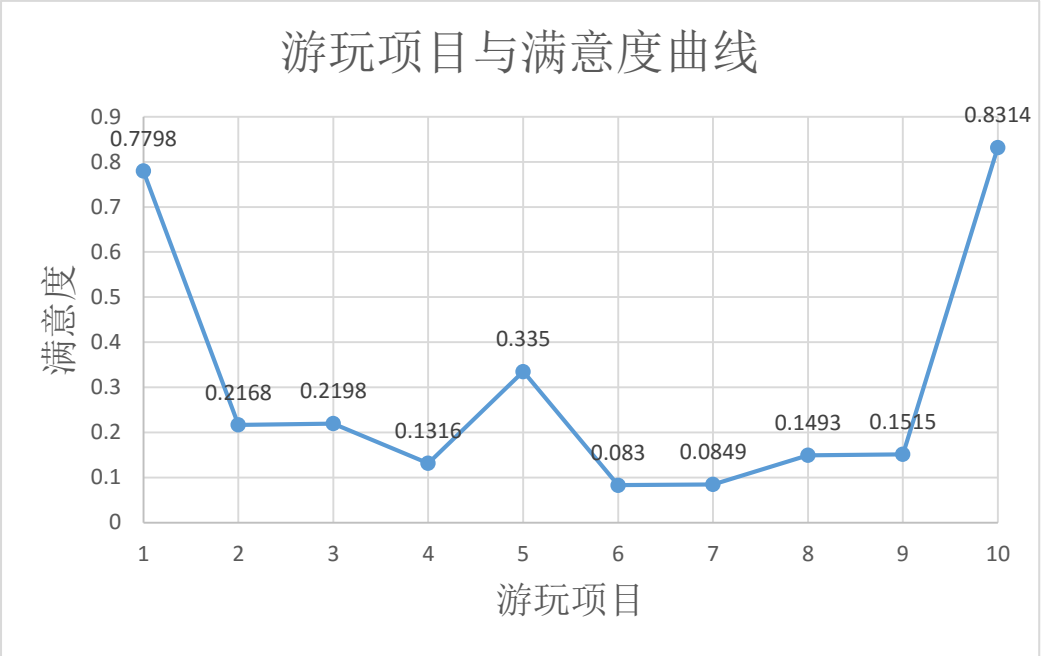


图 1 游玩项目与满意度曲线 1

从游客满意度值我们可以得出此名游客在该时刻选择游玩项目 J 会有最大体验度，因此可以将此游客引导到游玩项目 J。

模型二

初始条件

在模型一的基础上，深化模型以体现模型适用范围的普遍性。假设游客已经游玩了 A 与 E，在结束 E 的时刻进行决策到下一个游玩项目，由于假设同一个游玩项目游客不会玩两次，因此计算过程与结果均去掉游玩项目 A 与 E 的影响，其他初始情况与模型一相同。

与之前的建模过程相同，得出各个游玩项目的游客满意度指数曲线（游玩过的项目 A 和 E 满意度为零）

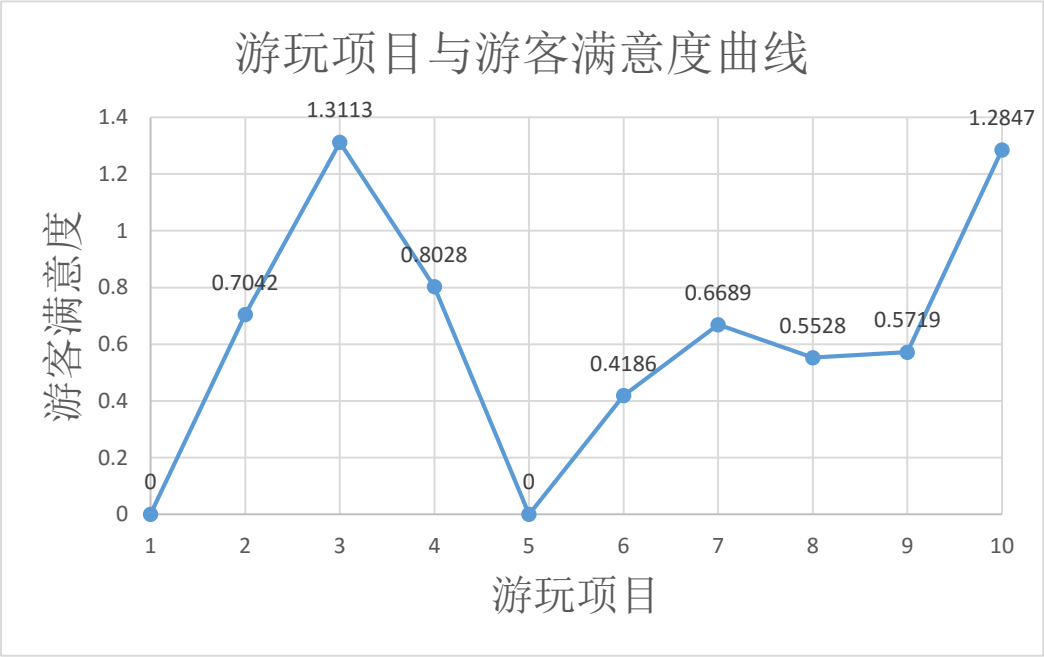


图 2 游玩项目与满意度曲线 2

从图中我们可以得出 C 景点的游客满意度最高，因此选择 C 作为下一个游玩项目。

5.1.3 结果分析与模型评价

从最终决策结果来看，可以得出以下结论。

1. 从模型一来看，在此满意度标准下，最优决策行为是从起点到较近的游玩项目 J，不是与出入口相距最近的 A，这是因为与 A 相比，J 的偏好程度要比 A 更高，因此选择 J 要比 A 的体验度更高。

2. 模型一中从入口到 J 经过了 A，但是没有选择 A，说明当前排队人数对游客满意度指标的影响。

3. 从模型一的游客满意度曲线图可以看出，虽然 A 的偏好指标比较低，但是在推荐路线上是排在第二位，这也说明了在模型中的行走路程对于游客体验度也有影响。

4. 在模型二中，最优决策不是与 E 相近的两个项目，这是因为在道路拥挤程度上，到达 C 的道路要比到离 E 最近的游玩项目的道路更通畅，人流密度更小，从而体现道路拥挤程度的影响。

从分析结果来看，当前时刻最优线路综合考虑了游客的主观偏好程度，项目之间的距离，道路拥挤程度，排队等候时间以及项目运转时间，容纳人数等因素，说明最优路线在此满意度的标准下给游客的体验程度最好。

此外因为当某一排队时间增加时，会影响协调系数的大小，从而影响游客体验程度，即满意度，所以当某一个游乐项目人数过多时，程序会引导其他不在这个游玩项目的游客到其他游玩项目，从而减小了该项目的负荷人数，不会造成游乐场内某个项目过大负载的情况，实现游乐场各个项目的符合平衡。

5.2 问题二

5.2.1 模型建立过程

1. 数据筛选

题目所给数据是预定日期与订房数量之间的关系，但是实际入住日期并不是预定日期，我们需要的应该是当天被预定的房间数，因此根据所给表格求出每天房间的被预订数并画成折线图。

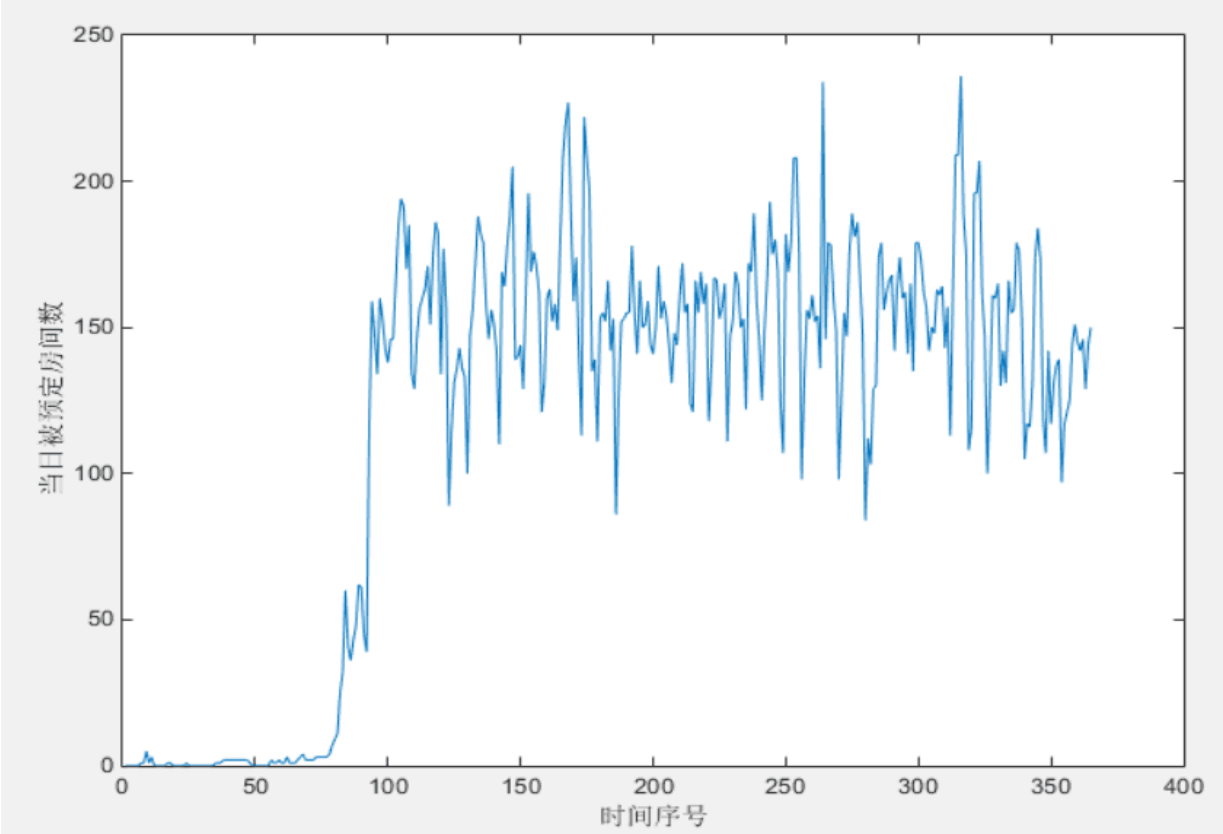


图 3 全年每天预定房间数折线

2. 影响因素定义

(1) 季度

季节的种类在很大程度上会影响到游乐场的人数，从而对入住酒店的人数也会有影响，比如人们可能不想在天气寒冷的时候去游玩，因为本题所给数据是 2015 年 1 到 12 月的预定数据，因此本文将根据月份顺序将一年划分成四个季度，如下表所示:

月份	季度
1-3 月	第一季度
4-6 月	第二季度
7-9 月	第三季度
10-12 月	第四季度

表 7 季度因素划分

统计每个季度的订房数:

季度	订房数
第一季度	523
第二季度	14168
第三季度	14178
第四季度	13962

表 8 季度与订房数

（2）节假日

根据生活经验以及研究分析，节假日种类与放假时间的长短与人们是否选择出行有很大的关系，因此本文把节假日按照放假日期来区分节假日种类，分成只是节日但不放假（某些节日虽然不放假但人们可能在当天更倾向于去游乐场游玩），放假三天，放假七天三个种类。

节假日种类	节假日名称
不放假	儿童节 万圣节 圣诞节
放假三天	元旦 清明节 端午节 劳动节 中秋节
放假七天	春节 国庆节

表 9 节假日因素划分

统计各个节假日的订房数量：

节假日类型	统计天数	订房数量
不放假	3	453
放假三天	15	1233
放假七天	14	1190

表 10 节假日与订房数量

（3）工作日与双休日影响。

是否是双休日会在一定程度上影响人们的出行计划，但是从常理来看，周五晚上到酒店入住周六早上直接去游乐场游玩的情况并不少见，此外在周日的时候由于第二天是工作日，所以选择入住酒店的人数可能会受到影响，所以本文将周五，周六，周日分开统计，周一至周四当成一种类型一起统计。

时间类型	订房总数
工作日	23976
周五	5467
周六	5467
周日	4605

表 11 双休日因素划分与订房数

(4) 寒暑期的影响

寒假日期定为二月一号至二月二十八号，暑假日期定为七月一号至八月三十一号。但是从图 来看，二月的房间入住数与周围两个月的数量差不多，并且波动比较小，所以认为寒假这个因素对于酒店房间的预订没有影响，只统计暑假的预订数。

时间类型	订房总数
暑期	14178
非暑期	29277

表 12 暑期与订房总数

基本模型

因变量（房间数）	季度	法定假	双休日	暑期
13	3.45561	21.6793	27.1605	0
5	3.45561	21.6793	27.9835	0
6	3.45561	21.6793	23.8683	0
7	3.45561	21.6793	20.9877	0
2	3.45561	23.7672	23.8683	0
154.3	32.3232	21.6793	27.1605	0
150.3	32.3232	21.6793	27.9835	0
157.7	32.3232	21.6793	23.8683	0
139.4	32.3232	21.6793	20.9877	0
160	32.3232	23.7672	27.1605	0
152	32.3232	24.3447	27.1605	0
163	32.5359	21.6793	27.1605	0
180	32.5359	21.6793	27.9835	0
157	32.5359	21.6793	27.1605	1
166	32.5359	21.6793	23.8683	0
153	32.5359	21.6793	27.9835	1
154	32.5359	21.6793	23.8683	1
117	32.5359	21.6793	20.9877	0
126	32.5359	21.6793	20.9877	1
208	32.5359	23.7672	27.1605	0
160	32.5359	23.7672	27.9835	0
149	32.5359	23.7672	23.8683	0
98	32.5359	23.7672	20.9877	0
162	32.5359	24.3447	27.1605	0
170	31.6853	30.2088	27.1605	0
142	31.6853	23.7672	23.8683	0
151	31.6853	23.7672	27.9835	0
153	31.6853	21.6793	27.9835	0
123	31.6853	21.6793	20.9877	0
154	31.6853	21.6793	23.8683	0
152	31.6853	21.6793	27.1605	0

表 13 房间

为了大致地分析 y 与 x1、x2、x3、x4 的关系，首先利用表 13 中的数据分别作出 y 对 x1、x1、x2、x3、x4 的散点图。

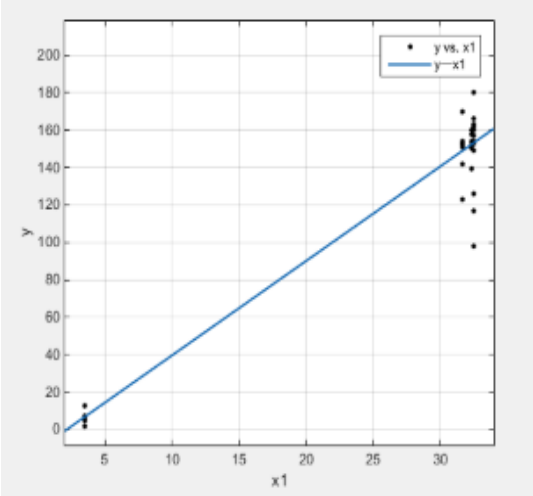


图 4 y 对 x_1 的散点图

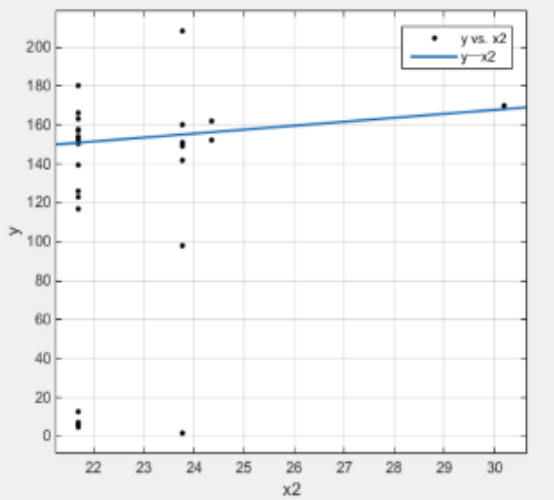


图 5 y 对 x_2 散点图

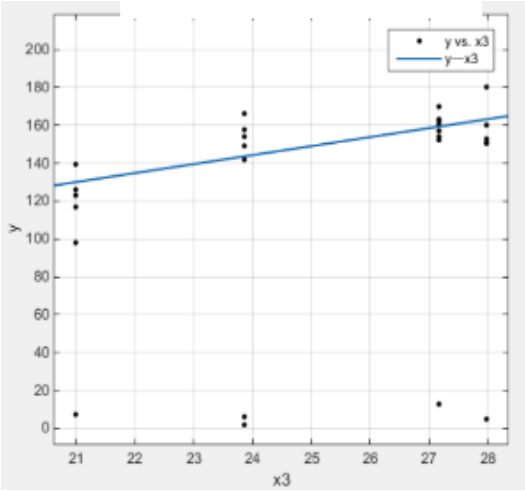


图 6 y 对 x_3 的散点图

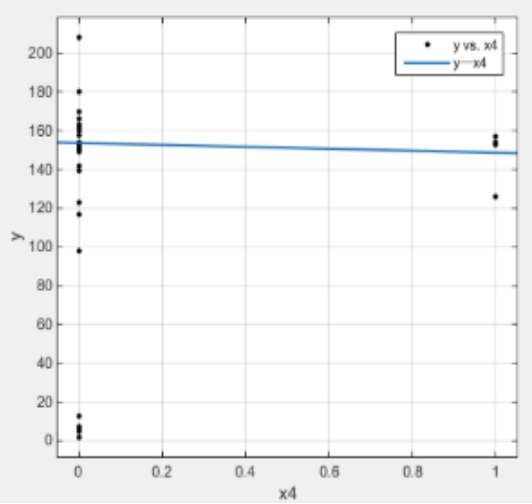


图 7 y 对 x_4 的散点图

从图 4 中可以发现，随着 x_1 的增加，Y 的值有比较明显的增长趋势，由于 x_1 权重取值不均匀，而散点大部分分布在 x_1>30 的区域，所以重新对 x_1>30 的区域作散点图。

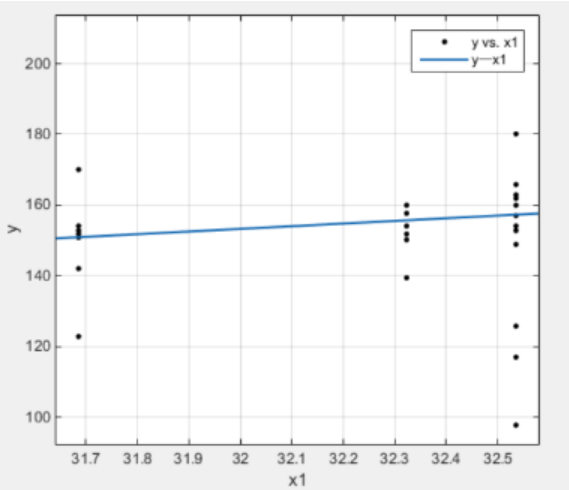


图 8 改正后 y 对 x_1 的散点图

观察散点集中部分，发现 x1 与有较为明显的线性增长关系，图 4 和图 5 中的直线是用线性模型

$$Y = b_0 + b_1 * x_1 + e_1 \quad (1)$$

拟合的，其中 e_1 是随机误差。

在图 6 中，随着 x_3 的增加，y 的值有比较明显的线性增长趋势，图 6 中的直线是用线性模型

$$Y = b_0 + b_3 * x_3 + e_3 \quad (2)$$

拟合的，其中 e_3 是随机误差。

而在图 5 和图 7 中，散点在横向较为集中，在纵向较为分散，但大体上也满足单调相关性，在这里将也先简化为线性模型

$$Y = b_0 + b_2 * x_2 + e_2 \quad (3)$$

$$Y = b_0 + b_4 * x_4 + e_4 \quad (4)$$

而对于这一部分的模型误差，将在后续的模型优化中进行模型的修正。
综合上面的分析，结合模型(1)、(2)、(3)和(4)建立如下的回归模型

$$Y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_4 * x_4 + e \quad (5)$$

(5)式右端的 x_1, x_2, x_3, x_4 为回归变量，$b_0 + b_1 * 1 + b_2 * 2 + b_3 * 3 + b_4 * 4$ 是给定季度指数 x_1、法定节假 n0.。。。。。日指数 x_2、双休日指数 x_3 和暑期指数 x_4 时，酒店当天预定数 Y 的平均值，其中的参数 b_0, b_1, b_2, b_3, b_4 为回归系数，由表 13 的数据估计，影响 Y 的其他因素作用都包含在随机误差 e 中，如果模型选择得合适，e 应大致服从均值为 0 的正态分布。

参数求解

对于多元回归分析，在 MATLAB 里用统计工具箱中的命令 regress 求解。

定义变量及参数

Y 为 33 维向量:

Y =[13 5 6 7 2 97 0 154.3 150.3 157.7 139.4 160 152 163
180 157 166 153 154 117 126 208 160 149 98 162 170 142 151 153 123 154
152]^T

x_1 为 33 维向量:

x_1 =[3.4556087 3.4556087 3.4556087 3.4556087 3.4556087 3.4556087
3.4556087 32.3232323 32.3232323 32.3232323 32.3232323 32.3232323
32.3232323 32.5358852 32.5358852 32.5358852 32.5358852 32.5358852
32.5358852 32.5358852 32.5358852 32.5358852 32.5358852 32.5358852
32.5358852 32.5358852 31.6852738 31.6852738 31.6852738 31.6852738
31.6852738 31.6852738 31.6852738]^T

x_2 为 33 维向量:

x_2 =[21.6792537 21.6792537 21.6792537 21.6792537 23.7672146 27.1604938
27.1604938 21.6792537 21.6792537 21.6792537 21.6792537 23.7672146
24.3447357 21.6792537 21.6792537 21.6792537 21.6792537 21.6792537
21.6792537 21.6792537 21.6792537 23.7672146 23.7672146 23.7672146
23.7672146 24.3447357 30.2087961 23.7672146 23.7672146 21.6792537
21.6792537 21.6792537 21.6792537]^T

x_3 为 33 维向量:

x_3 =[27.1604938 27.9835391 23.8683128 20.9876543 23.8683128 27.1604938
27.1604938 27.1604938 27.9835391 23.8683128 20.9876543 27.1604938
27.1604938 27.1604938 27.9835391 27.1604938 23.8683128 27.9835391
23.8683128 20.9876543 20.9876543 27.1604938 27.9835391 23.8683128
20.9876543 27.1604938 27.1604938 23.8683128 27.9835391 27.9835391

20.9876543 23.8683128 27.1604938 $]^T$

x_4 为 33 维向量:

x_4 =[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1

0 1 0 0 0 0 0 0 0 0 0 0 0 0 $]^T$

x 为 33 X 5 维回归变量矩阵：

x=[1 x_1 x_2 x_3 x_4]

b 为对应于回归变量矩阵 x 的回归系数向量:

b =[b_0 b_1 b_2 b_3 b_4 $]^T$

即有：

$y = x * b$

输入代码：

[b,bint,r,rint,stats]=regress(y,x,0.05)

其中置信水平为 0.05，返回值 b 为回归系数向量的估计值，bint 为 b 的置信区间，r 为残差向量，rint 为 r 的置信区间，stats 为该回归模型的检验统计量，返回值有 3 个，第一个值为该模型回归方程的相关系数 R^2，第二个值为 F 的统计量值，第三个是与 F 统计量对应的概率值 p。

得到模型(5)的回归系数估计值及其在置信水平为 0.05 下的置信区间、检验统计量 R^2，F，p 的结果如下：

回归系数	回归系数估计值	回归系数置信区间
b0	-171.68	(-276.26 -67.1)
b1	4.71	(4.08 5.34)
b2	2.35	(-1.55 6.26)
b3	4.65	(1.74 7.55)
b4	-1.61	(-24.79 21.56)
R^2=0.9006 F=63.405 p<0.0001		

表 14 模型（4）的计算结

表 14 显示，R^2=0.9006 指因变量 y(当日预定房间数)的 90.06%可由模型确定，F 值远远超过 F 检验的临界值，p 远小于 0.05，因而确定模型（4）从整体来看是可用的。

表 14 的回归系数给出了模型（4）中 b_0,b_1,b_2,b_3,b_4 的估计值。检查他们的置信区间发现，b_2, b_4 的置信区间是包含零点的，其中 b_2 的置信区间左端点离零点很近，表明回归变量 x_2 对因变量 y 的影响不太显著，符合前文讨论的模型(3)是存在较大误差的，需要修正模型。而对于 b4 的置信区间，左端点和右端点距离

零点近似相等，而 b_4 的估计值为-1.61，

在 x_4 的取值范围为 0 或 1 的定义域内，对 y 值的影响波动也在 2 之内。通过观察残差 r 与 x_4 的关系散点图：

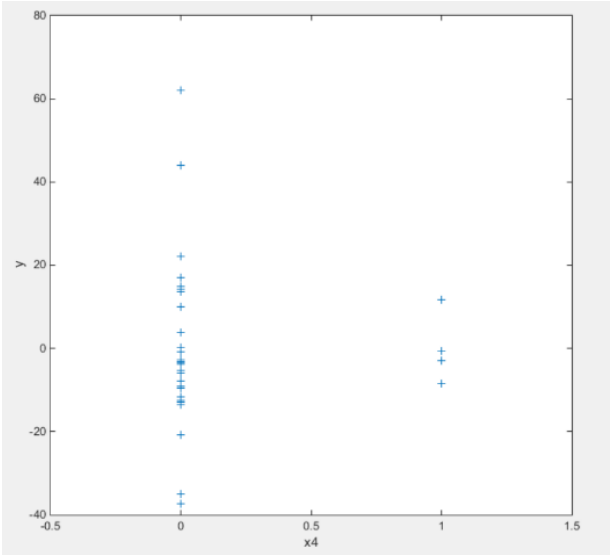


图 9　残差与 x_4 的关系

发现散点大部分都分布在 $x_4 = 0$ 的区域内，集中分布在 0 的两侧，说明 x_4 对于 y 的取值影响极小，几乎可以忽略。

模型改进

由对模型(4)的分析可得，x_4 的影响基本可以忽略，而模型中 x_3 对于因变量 y 的影响存在较大的误差，需要修正，因此需要对 x_3 进行改进的模型建立。

模型(5)中回归变量 x_1、x_2 和 x_3 对于因变量 y 的影响是相互独立的，即当天预定房间数 y 的均值与季度指数 x_1 的线性关系由 b_1 确定，不依赖于 x_2 和 x_3，同样，y 的均值与 x_2、x_3 的线性关系也分别由 b_2 和 b_3 确定，不与除自身以外的其余变量有关。根据直觉和经验可以猜想，节假日指数 x_2 与双休日指数 x_3 之间会有交互作用的，实际上是不独立的，假若当天为节假日，在一定程度上人们不再关注当天是否为双休日，也就是会削弱 x_2 的作用，也就是说 x_2 与 x_3 的交互作用会对 y 有影响，不妨简单地用 x_2 与 x_3 的乘积代表他们的交互作用，于是将模型(4)

增加一项，得到

$$Y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_{23} * x_2 x_3 + e \quad (6)$$

利用表 13 的数据估计模型(6)的系数，利用 MATLAB 的统计工具箱得到的结果如下表：

回归系数	回归系数估计值	回归系数估计值置信区间
b_0	1202.33	(354.68,2759. 4)
b_1	4.72	(4.13,5.31)
b_2	-59.59	(-129.75,-10.59)
b_3	-47.6	(-104.82,-11.39)
b_{23}	2.31	(0.306,4.92)
R^2=0.9110 F=71.6 P=0		

表 15 模型(6)的计算结果

表 15 与表 14 的结果相比，R^2 有所提高，说明模型(6)比模型(5)有所改进，并且所有的参数的置信区间，特别是 x_2 与 x_3 的交互作用项的系数 b_{23} 的置信区间不包含零点，所以有理由相信模型(6)比模型(5)更符合实际。
与模型(5)类似，做模型(6)的残差分析图：

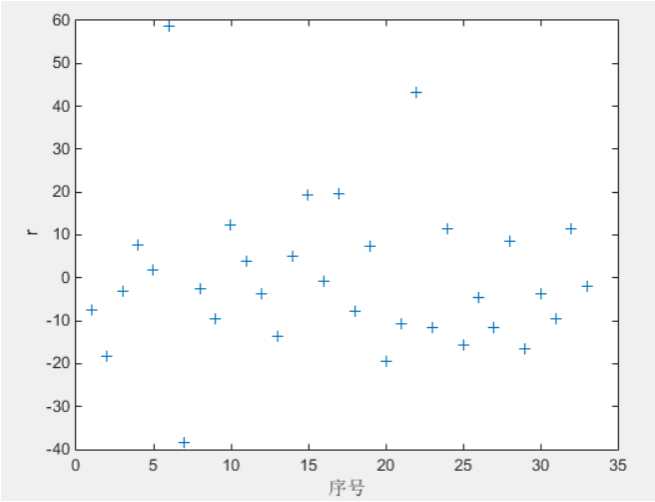


图 15 残差分布图

可以看出，残差 r 散点在 0 附近的带状分布，靠近 0 较密，距离 0 远的散点较稀疏，总体上在-20 到 20 之间分布，接近正态分布，说明这个模型是可靠有效

的。从图 15 中还可以看到两个异常点 r_6 和 r_7，这两个点表示在当天的预定房间数突然的剧增或剧减，有可能是类似于旅游团预定因素等，由我们未知的原因造成的，为了使个别的数据不致影响整个模型，应该将这个异常数据去掉，对于模型(6)重新估计回归系数，得到的结果如下表：

回归系数	回归系数估计值	回归系数估计值置信区间
b_0	1072.33	(95.26,2239.94)
b_1	4.89	(4.39,5.39)
b_2	-53.84	(-106.43,-1.25)
b_3	-41.1	(-84.86,-2.65)
b_{23}	2.05	(0.086,4.03)
R^2=0.9474 F=117 P=0		

表 16

残差分析图：

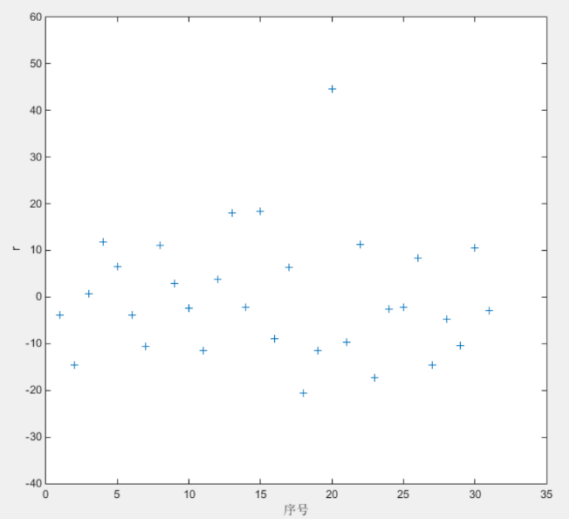


图 11 残差分布图

可以看出，去掉异常数据后结果又有改善，相关系数 R^2 显著提高，由原来的 0.9110 提高到了 0.9474，残差散点图分布更加均匀，对于这样的回归系数是满意的。此回归方程为：

$$Y = 1072.23 + 4.89 * x_1 - 53.84 * x_2 - 41.1 * x_3 + 2.05 * x_2 * x_3 \quad (7)$$

5.2.2 模型检验与仿真模拟

用模型(7)做为检验模型，将回归变量 x 代入(7)式，计算出拟合数据 y_1，并

将实际数据 y 和拟合数据 y_1 在同一张图上作出如下折线图：

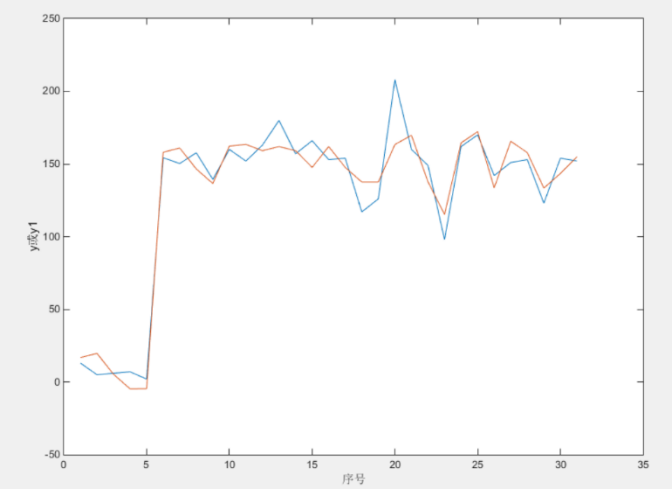


图 12 检验折线图

其中蓝线为实际数据，红线为拟合数据。对比发现：拟合折线与实际折线在增长和下降趋势都是一致的，数据值也几乎较为贴合，对于这种拟合结果，是满意的。

用模型(7)做为检验模型，将回归变量 x 代入(7)式，计算出拟合数据 y_1，并将实际数据 y 和拟合数据 y_1 在同一张图上作出如下折线图：

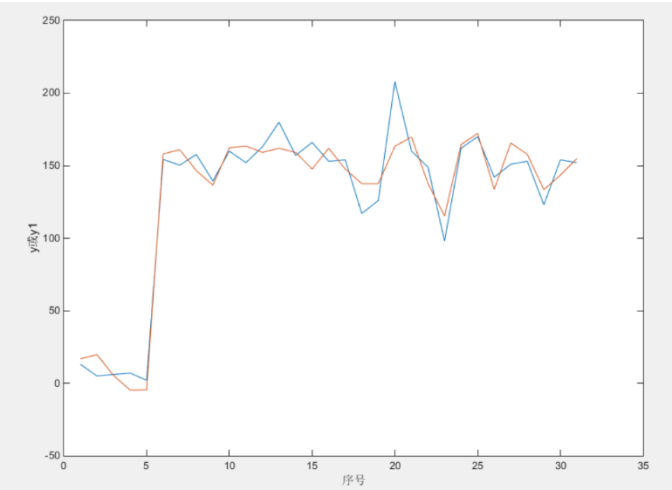


图 13 检验折线图

其中蓝线为实际数据，红线为拟合数据。对比发现：拟合折线与实际折线在增长和下降趋势都是一致的，数据值也几乎较为贴合，对于这种拟合结果，是满意的。

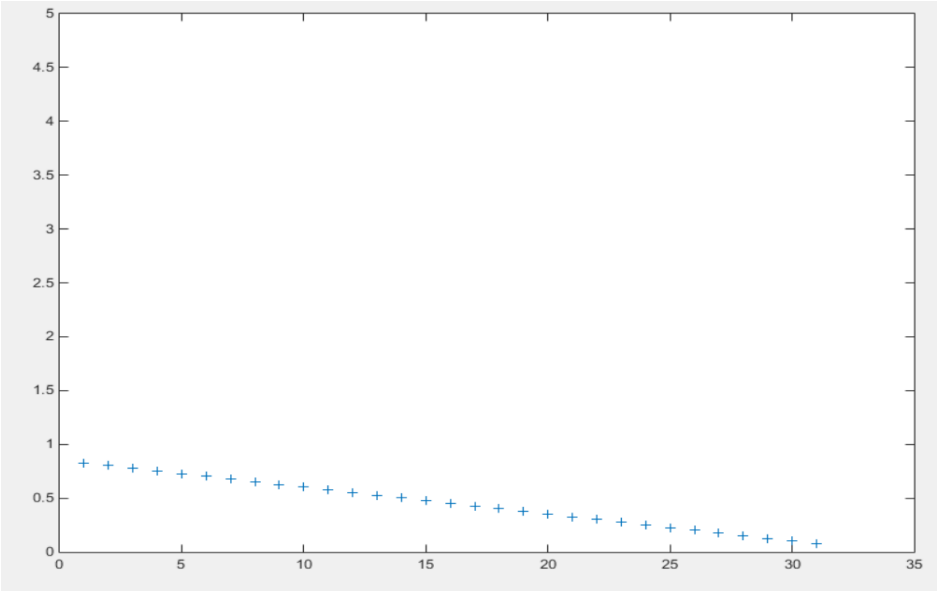
由上述公式可以拟合出 1-3 月三个月的时间与订房量的关系方程：

$$y_1 = -0.025 * x + 0.8516?$$

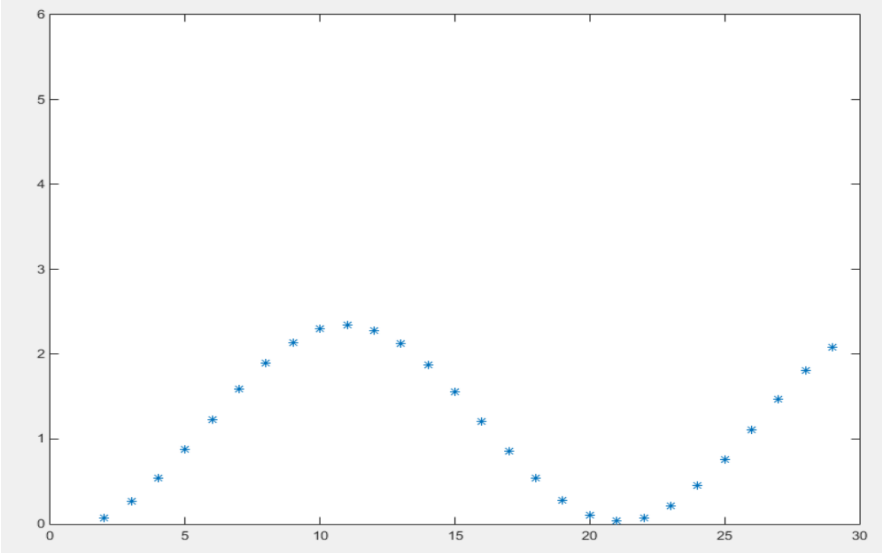
$$y_2 = 1.574 * sin(0.003014 * x + 0.8082) + 1.171 * sin(0.3061 * x + 4.512)$$

$$y_3 = 0.007943 * x^3 - 0.2381 * x^2 + 2.055 * x - 2.425$$

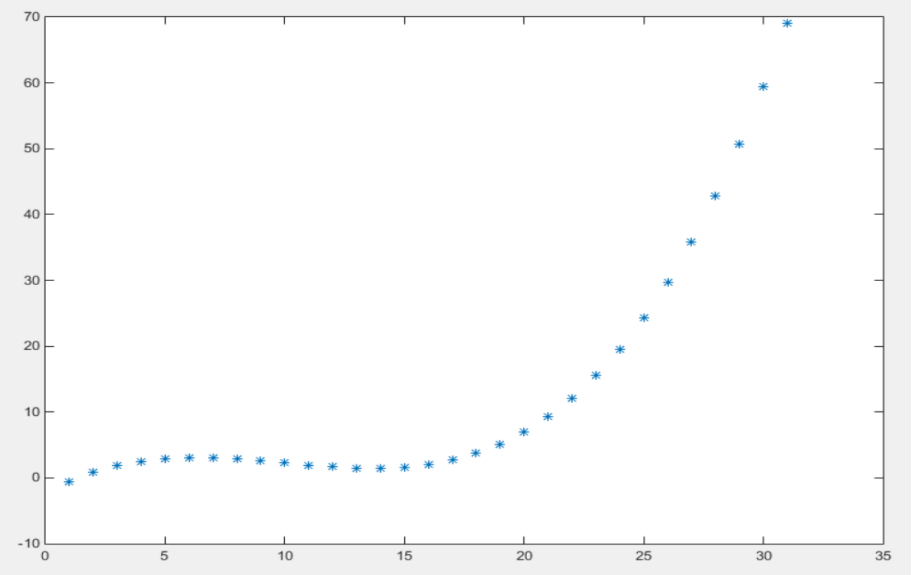
从而预测出各个月的订房数量与时间关系图



一月



二月



三月

5.2.3 结果分析

从预测结果来看，一月份前段时间与实际相差比较大，原因可能是一月份虽然处于淡季，但是受元旦的影响较大，在划分影响关系是没有考虑好与季度之间的关系变化。

一二月以及三月前半个月订房数量较少，说明冬季的影响因素比较大，而三月下半月订房数迅速上升可能因为开学或者大家从春节假期回来预订数量上升，这些结果都与 2015 年的数据趋势一致，说明模型的准确性。

此模型很好地模拟了在季度，节假日以及双休日（工作日）影响下的订房数量，但是这三种因素都是客观因素，一些主观或者不可抗因素其实也会很大程度地影响订房数量，比如酒店促销，价格上涨，恶劣天气等等，但是由于数据的不足所以很难以在考虑更多因素的情况下的订房数量。

六、参考文献

[1]冯刚，任佩瑜，戈鹏，朱忠福，冉建华，基于管理熵与 RFID 的九寨沟游客高峰期“时空分流”导航管理模式研究，《旅游科学》，第 24 卷第 2 期，2010 年 4 月.

[2] 任竞斐，郑伟民, 景区旅游高峰期游客旅游路线动态实时调度仿真研究，42 到 45 页，《统计与决策》，2013 年第 8 期.

[3]姜启源，谢金星，叶俊，《数学模型》，高等教育出版社，2003.

附录：

Matlab 源代码：

从入口出发做决策：

```
tplay=[1980 75 150 150 300 150 120 90 90 120];
twalk=[241 368 677 1040 564 1817 1162 1288 874 592];
carry=[400 30 50 30 100 50 30 30 20 50];
pwait=[670 320 360 435 230 690 340 570 340 278];
pin=[321 491 1550 1786 645 1733 1773 1966 1167 678];
lambda=[3.119 8.02 10.55 11.58 4.02 6.143 7.64 17.02
14.39 24.11];
for i=1:10
    pout(i)=floor(twalk(i)/tplay(i))*carry(i);

twait(i)=floor((pwait(i)+pin(i)+pout(i))/carry(i))*tplay(
i)+tplay(i)-mod((twalk(i)-tplay(i)),tplay(i));

value(i)=tplay(i)*lambda(i)/(twalk(i)+tplay(i)+twait(i))
end
```

从 E 出发做决策：

```
tplay=[1980 75 150 150 300 150 120 90 90 120];
twalk=[322 322 631 994 0 1269 598 952 537 672];
carry=[400 30 50 30 100 50 30 30 20 50];
pwait=[670 320 360 435 230 690 340 570 340 278];
pin=[67 116 300 239 0 175 194 206 145 106];
lambda=[0 0.0169 0.03016 0.0289 0 0.01465 0.01806 0.02598
0.02059 0.02441];
done=[1,5];
for i=1:10
    value(i)=0
    if( i~=done(1) && i~=done(2))
           pout(i)=floor(twalk(i)/tplay(i))*carry(i);

twait(i)=floor((pwait(i)+pin(i)+pout(i))/carry(i))*tplay(
i)+tplay(i)-mod((twalk(i)-tplay(i)),tplay(i));

value(i)=tplay(i)*lambda(i)/(twalk(i)+tplay(i)+twait(i));
    end
end
plot(value,'*')
```

按月划分入住房间数：

```
y1=[0   0   0   0   0   0   1   1   5   1   3   0   0   0   0   0
1   1   0   0   0   0   0   1   0   0   0   0   0   0   0];
y2=[0   0   0   1   1   1   2   2   2   2   2   2   2   2   2   2
1   0   0   0   0   0   0   0   2   1   1   2];
```

```
y3=[1   1   3   1   1   1   2   3   4   2   2   2   2   3   3   3
3   3   4   7   9   11  25  32  60  41  36  43  48  62  61];
y4=[45  39  118 159 148 134 160 153 143 138 146 146 166
185 194 191 170 185 134 129 147 157 160 163 171 151 174
186 182 134];
y5=[177 157 89  115 131 136 143 136 133 100 147 156 172
188 182 179 158 146 156 151 142 110 169 164 178 188 205
139 140 144 129];
y6=[160 196 169 176 171 162 121 131 160 163 152 158 149
180 208 220 227 189 159 174 142 113 222 208 197 135 139
111 153 155];
y7=[152 166 142 153 86  126 152 153 155 155 178 155 141
166 150 151 159 144 141 150 171 153 159 154 144 131 148
144 160 172 155];
y8=[158 124 121 166 155 169 158 165 118 138 167 166 153
157 165 111 147 152 169 165 150 153 122 172 169 189 159
146 125 149 167];
y9=[193 175 180 169 124 107 182 169 179 208 208 177 98
133 156 153 161 152 154 136 234 146 179 178 160 149 98
127 155 147];
y10=[176   189 181 186 167 147 84  112 103 129 130 174
179 156 162 166 168 142 164 174 160 162 141 165 135 179
179 173 162 157 142];
y11=[150   148 163 161 164 143 157 113 163 209 209 236
189 175 108 116 196 196 207 162 147 100 129 161 160 165
130 142 131 166];
y12=[155   156 179 176 153 105 117 116 131 174 184 173
119 107 142 117 131 137 139 97  117 121 125 144 151 145
142 146 129 144 150];
x1=1:31;
x2=1:28;
x3=1:31;
x4=1:30;
x5=1:31;
x6=1:30;
x7=1:31;
x8=1:31;
x9=1:30;
x10=1:31;
x11=1:30;
x12=1:31;
plot(x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6,x7,y7,x8,y8,x9,y
9,x10,y10,x11,y11,x12,y12)
```

数据参数回归分析：

```
y=[13
5
6
7
2
154.3
150.3
157.7
139.4
160
152
163
180
157
166
153
154
117
126
208
160
149
98
162
170
142
151
153
123
154
152
];
x1=[3.4556087
3.4556087
3.4556087
3.4556087
32.3232323
32.3232323
32.3232323
32.3232323
32.3232323
32.3232323
32.5358852
```

```
32.5358852
32.5358852
32.5358852
32.5358852
32.5358852
32.5358852
32.5358852
32.5358852
32.5358852
32.5358852
32.5358852
32.5358852
31.6852738
31.6852738
31.6852738
31.6852738
31.6852738
31.6852738
31.6852738
];
x2=[21.6792537
21.6792537
21.6792537
23.7672146
21.6792537
21.6792537
21.6792537
21.6792537
21.6792537
23.7672146
24.3447357
21.6792537
21.6792537
21.6792537
21.6792537
21.6792537
21.6792537
21.6792537
21.6792537
21.6792537
23.7672146
23.7672146
23.7672146
23.7672146
24.3447357
```

```
30.2087961
23.7672146
23.7672146
21.6792537
21.6792537
21.6792537
21.6792537
];
x3=[27.1604938
27.9835391
23.8683128
20.9876543
23.8683128
27.1604938
27.9835391
23.8683128
20.9876543
27.1604938
27.1604938
27.1604938
27.9835391
27.1604938
23.8683128
27.9835391
23.8683128
20.9876543
20.9876543
27.1604938
27.9835391
23.8683128
20.9876543
27.1604938
27.1604938
23.8683128
27.9835391
27.9835391
20.9876543
23.8683128
27.1604938
];
x=[x1./x1 x1 x2 x3 x2.*x3 ];
[b,bint,r, rint,stats]=regress(y,x,0.05)
```

C++源代码：

第一题:

（1） 行走时间——基于 floyd 的算法

```
#include <iostream>
#include "math.h"
using namespace std;
const int INF = 100000;
int n=10;
double v[17],row[17],path[32][32],dist[32][32],map[32][32];
void init(){
   int i,j;
     for(i=0;i<=n;i++)
        for(j=0;j<=n;j++)
       map[i][j]=(i==j)?0:99999;
     map[0][1]=300,map[0][2]=400;

map[1][0]=300,map[1][2]=300,map[1][5]=350,map[1][10]=250;

map[2][1]=300,map[2][0]=400,map[2][3]=300,map[2][5]=350;
     map[3][2]=300,map[3][4]=450,map[3][5]=500;
     map[4][3]=450,map[4][6]=500;

map[5][1]=350,map[5][2]=350,map[5][3]=500,map[5][7]=550,map[5][
9]=450;
   map[6][4]=500,map[6][7]=650;
   map[7][6]=650,map[7][5]=550,map[7][8]=400;
     map[8][7]=400,map[8][9]=450;
     map[9][5]=450,map[9][8]=450,map[9][10]=350;
   map[10][1]=250,map[10][9]=350;
   cout<<"input the row of each road"<<endl;
        double rsum,rowavg;
     for(int p=1;p<=16;p++)
   {v[0]=1.34;
     cin>>row[p];
   }

     for( p=1;p<=16;p++)
   {v[0]=1.34;
     v[p]=v[0]/pow(row[p]/0.1,0.8);
   }

     map[0][1]=300/v[1],map[0][2]=400/v[2];
```

```
map[1][0]=300/v[1],map[1][2]=300/v[3],map[1][5]=350/v[4],map[1]
[10]=250/v[5];

map[2][1]=300/v[3],map[2][0]=400/v[2],map[2][3]=300/v[6],map[2]
[5]=350/v[7];

map[3][2]=300/v[6],map[3][4]=450/v[8],map[3][5]=500/v[9];
        map[4][3]=450/v[8],map[4][6]=500/v[10];

map[5][1]=350/v[4],map[5][2]=350/v[7],map[5][3]=500/v[9],map[5]
[7]=550/v[11],map[5][9]=450/v[12];
     map[6][4]=500/v[10],map[6][7]=650/v[13];

map[7][6]=650/v[13],map[7][5]=550/v[11],map[7][8]=400/v[14];
        map[8][7]=400/v[14],map[8][9]=450/v[15];

map[9][5]=450/v[12],map[9][8]=450/v[15],map[9][10]=350/v[16];
     map[10][1]=250/v[5],map[10][9]=350/v[16];


    }
    void floyd(){
        int i,j,k;
        for(i=0;i<=n;i++)
            for(j=0;j<=n;j++)
                dist[i][j]=map[i][j],path[i][j]=0;
        for(k=0;k<=n;k++)
            for(i=0;i<=n;i++)
                for(j=0;j<=n;j++)
                    if(dist[i][k]+dist[k][j]<dist[i][j])

dist[i][j]=dist[i][k]+dist[k][j],path[i][j]=k; }
    void output(int i,int j){
        if(i==j)
        cout<<"you have choose the same spot"<<endl;
       else if(path[i][j]==0)
        cout<<"-->"<<j;
        else{
            output(i,path[i][j]);
            output(path[i][j],j);

        }
    }
```

```
int main(){
    int a,b,u,z,x;
 int spots[11]={0,1,2,3,4,5,6,7,8,9,10};
    int unvisit[11];

 cout<<"please input your starting point"<<endl;
 cin>>u;
 cout<<"please input the spots you have NOT visited,ended
with the number'999'"<<endl;
    for(a=0;a<11;a++)
 {
    cin>>x;
    if (x==999)
       break;
    else if(x<0 && x>10)
       cout<<"there is no such spots,please input it
correctly"<<endl;
    else
       unvisit[a]=x;
 }


    init();
    floyd();
 for(b=0;b<a;b++)
 {z=unvisit[b];
    if(u||z)
 {
        if(dist[u][z]==INF) cout<<"No path"<<endl;
        else
    {
            cout<<"start from: "<<u;
            output(u,z);
       cout<<"   "<<"the whole time is "<<dist[u][z]<<"
seconds";
        cout<<endl;
    }
 }
 }
    return 0;
}
```

（2） 等待时间、进入人数与协调系数

```
#include <iostream>
#include "math.h"
```

```
using namespace std;
const int INF = 100000;
int n=10;

double p[11]={0,4,7,12,9,6,5,8,8,7,6};
double pa=70;
double wp=14335;
double o[10];
double
row[17],disc[10],w[11],path[32][32],dist[32][32],map[32][32];
double nameta[11][10],zk[10],wait[10],ROW[11];
void main()
{        int unvisit[11];
  int a,b,u,x;
  cout<<"please input your starting point"<<endl;
  cin>>u;
      int un[10];
      int spots[11]={0,1,2,3,4,5,6,7,8,9,10};
      for(a=0;a<11;a++)
      {
          if (spots[a]==u)
              break;
          else un[a]=spots[a];

      }
      for(b=a;b<11;b++)
      {
          un[b]=spots[b+1];
      }

 double arfa[11];
    int i,j;
    for(i=0;i<=10;i++)
        arfa[i]=p[i]/pa;
    cout<<"input the walking time of each spot"<<endl;
    for(i=0;i<=10;i++)
        cin>>w[i];

     for(i=0;i<=n;i++)
        for(j=0;j<=n;j++)
       map[i][j]=(i==j)?0:99999;
     map[0][1]=300,map[0][2]=400;

map[1][0]=300,map[1][2]=300,map[1][5]=350,map[1][10]=250;
```

```
map[2][1]=300,map[2][0]=400,map[2][3]=300,map[2][5]=350;
          map[3][2]=300,map[3][4]=450,map[3][5]=500;
          map[4][3]=450,map[4][6]=500;

map[5][1]=350,map[5][2]=350,map[5][3]=500,map[5][7]=550,map[5][
9]=450;
      map[6][4]=500,map[6][7]=650;
      map[7][6]=650,map[7][5]=550,map[7][8]=400;
          map[8][7]=400,map[8][9]=450;
          map[9][5]=450,map[9][8]=450,map[9][10]=350;
      map[10][1]=250,map[10][9]=350;
    cout<<"input row of road"<<endl;
        for(int q=1;q<17;q++)
        {
            cin>>row[q];}

      double R[11][11];
           for(i=0;i<11;i++)
          for(j=0;j<11;j++)
              R[i][j]=0;
          R[0][1]=row[1],R[0][2]=row[2];

R[1][0]=row[1],R[1][2]=row[3],R[1][5]=row[4],R[1][10]=row[5];

R[2][1]=row[3],R[2][0]=row[2],R[2][3]=row[6],R[2][5]=row[7];
          R[3][2]=row[6],R[3][4]=row[8],R[3][5]=row[9];
          R[4][3]=row[8],R[4][6]=row[10];

R[5][1]=row[4],R[5][2]=row[7],R[5][3]=row[9],R[5][7]=row[11],R[
5][9]=row[12];
      R[6][4]=row[10],R[6][7]=row[13];
      R[7][6]=row[13],R[7][5]=row[11],R[7][8]=row[14];
          R[8][7]=row[14],R[8][9]=row[15];
          R[9][5]=row[12],R[9][8]=row[15],R[9][10]=row[16];
      R[10][1]=row[5],R[10][9]=row[16];
    double Pin[11];

      for (int k=0;k<11;k++)
      {   double sumrow=0;
      ROW[k]=0;
      int count=0;
          for(i=0;i<11;i++)
           for(j=0;j<11;j++)
```

```
            {
                if(i==k || j==k)
                {
                    ROW[k]+=R[i][j];
                count++;
                     sumrow =sumrow+pow(R[i][j],0.2);
                }

            }
             sumrow=sumrow/2;
             ROW[k]=ROW[k]/2*count;
                 Pin[k]=arfa[k]*w[k]*1.34*sumrow;
                cout<<k<<"Pin"<<Pin[k]<<endl;
        }
        double sum=0;
        cout<<"input waiting time for each spot"<<endl;
        for(int icount=0;icount<=10;icount++)
              {cin>>wait[icount];
              }
        for(i=0;i<=10;i++)
        { double x1,y1,z1;
        x1=(p[i]/7.3);
        if (i==u)
            x1=0;
            y1=(1+pow(wait[i]/1800,-0.3));
            z1=(0.1/ROW[i]);
            cout<<x1<<"    "<<y1<<"   "<<z1<<"  row is
"<<ROW[i]<<endl;
            nameta[u][i]=x1*y1*z1;

            cout<<"for start "<<u<<" to the spot "<<i<<"协调系数为
"<<nameta[u][i]<<endl;
        }
}
```