姓名 1 （队长）		学号		性别	
专业		学院		校区	嘉定校区
手机号		邮箱		分工	建模 编程 写作
姓名 2		学号		性别	男
专业		学院		校区	嘉定校区
手机号		邮箱		分工	
姓名 3		学号		性别	男
专业		学院		校区	
手机号		邮箱		分工	建模

游乐园客流疏导方案

摘要：

对景区游客进行最高效的疏导分流、对酒店人员入住数目进行最准确的预测一直是国内外旅游业研究的重点。本文在问题（1）中，通过结合具体的景区方位图，通过对传统引力模型和排队模型的创新性改进，分初态和稳态两个阶段对该 Youth 游乐园游客建立不同的分流疏导模型，从而达到让游客获得最佳游园体验的目的。在问题（2）中，本文在整理研究了 2015 年酒店预定量数据特性的基础上建立了具有充分逼近任意复杂的非线性关系特点的 NAR 神经网络预测模型，然后根据影响因素（如季节，节假日等）作用大小的不同对待预测部分进行了分组处理，再利用相应的 2015 年预定量数据进行了重新估测，并通过观察估测误差及误差自相关系数，证明了这一模型的合理性。

关键词：引力模型　排队论　泊松过程　分流疏导　NAR 神经网络　酒店人数预测

1、问题重述

Youth 游乐园即将盛大开园，作为本市建有最多过山车的游乐园，受到了青少年的热捧。预计届时园区将迎来每天 1 万的大客流。如何根据客流情况，及时分流人群，为顾客提供游园线路引导，保障游客的游园体验显得尤为重要。
试就园区的整体规划，建立数学模型分析研究下面的问题：

（1）附件 1 为 Youth 乐园的规划图，共设 A-J 共 10 个项目点，游客可沿着图中标出的线路往返下个游乐项目。在保障每位游客体验游乐设施的前提下，建立对每个游乐项目的等候游客进行游览提醒和疏导的模型，以达到游园体验最优。每个游乐项目安排请参见表 1。

表 1. 每个游乐项目的时间安排

游乐项目	每场容纳游客数	每场持续时间
A	400	33分
B	30	1分15秒
C	50	2分30秒
D	30	2分30秒
E	100	5分
F	50	2分30秒
G	30	2分
H	30	1分30秒
I	20	1分30秒
J	50	2分

（2）皇冠假日酒店是游乐园内的酒店，目前已开业，为有需要的游客提供住宿便利。请根据该酒店历史预订数据信息,综合考虑影响房间预定量的主要因素(比如季节,工作日/周末,法定假日,暑期等)建立数学模型。并根据酒店 2015 年全年预定数据(附件 2),预测 2016 年 1 月至 3 月每天预定房间数.

2、问题分析

2.1 游园分流与疏导

要使游客的游园体验达到最佳，那如何让游客少走弯路，并在较短的时间内游玩最多的游乐项目显得尤为关键，即路程尽量短、效率尽量高是影响游客游园体验的两大主要影响因素，这就需要对游客进行游园方向上的引导和分流。

就路程尽量短而言，可结合各个景点之间的距离利用图论求出玩完十个景点的最短路径；就效率尽量高而言，因为景点之间的路程固定、各个景点一场活动的时间相对稳定，故而完成相同数目景点游玩时间的长短取决于在各个景点的排队等待时间的多少（假设游客在不间断地游玩，忽略人为疲劳因素），但往往两者不可兼得。

假若游乐园先有一个宏观的推荐路线，再加以时刻动态的监控，则会比仅仅依靠动态监控分流的效果要好，即：调控在有向引导下因主观思想带来的无序扩散会比无向无序的扩散更为简单，也可以节省资本。因此，本文以两大因素中的路径为基点，利用图论的知识寻求较短路径，再通过分别建立初态和稳态阶段的排队模型，根据各个不同时刻的客流量预估游客在各个景点不同时刻上的排队等待时间，建立引力模型寻求最佳的游园疏导线路，并通过评价体系验证其可行性。

2.2 预定酒店数预测

通过对该酒店 2015 年每天房间预定量变化趋势的分析，发现预定量受季节气候，假日，寒暑假等多重因素影响而呈现非线性变化。故我们认为可以通过考虑影响预定量的主要因素，对 2015、2016 年 1-3 月进行分区处理，从而使预测更为合理。预测工具则选择具有强大自学习功能及充分逼近任意复杂的非线性关系特点的 NAR 动态神经网络预测。

3、模型的基本假设

3.1 游园疏导模型假设

（1）假设游乐场营业时间为 8:00——22:00，其中 11:00——19:00 为稳态高峰期，其余时间段为非高峰期（各个景点开始运营无时间差，视为开园后即可接收游客）。日均客流量为 10000 人，通过数据查阅及比较，假设稳态时客流量为 15 人/min ，初态和非高峰期时为 7.8 人/min。

（2）无特殊说明时，假设每位游客第一次到达一个景点时不会绕开，会排队等到游玩结束后才离开；若不是第一次到达该景点，则仅仅是路过，不会再次游玩。

（3）假设游客游玩时不间断的，忽略其人为疲劳因素（包括中途休息与吃饭等）。

（4）假设初态过程游客到达率服从参数为λ_1的泊松分布，为使得景点设施能得到充分利用，初始时必须达到 60%或以上额定容量才运行。

（5）假设稳态过程超负荷景点只能向与其相邻的景点分流游客，并且只能转移一次；而为保障系统稳定运行且节省资源，每一个分流景点每次只接受一个超载荷景点的分流。

（6）假设各个景点稳态过程游客到达率服从参数为λ_2的泊松分布，且独立

互不影响。

（7）假设稳态时游客流量足够大，每场次都可以满员，可将一批额定容量的人视为一个整体，批次内部等待时间的误差不计，将各批游客间的平均等待时间视为各个游客间的平均等待时间；。

（8）假设稳态时各个景点的服务时间服从期望为$T_{额}$，方差为δ的正太分布。

3.2 预定酒店数模型假设

（1）皇冠假日酒店及 youth 游乐园在 2015 和 2016 年两年内规模，设施环境，价格，服务及周边交通条件基本保持一致

（2）该游乐园及酒店所在区域 2015,2016 两年对应月份天气情况基本一致。

4、游园疏导模型的建立与分析求解

4.1 最短路径

先忽略人流量及景点容量对游园时间的影响（此假设的合理性在于当人流量极其少，即刚开园和接近闭园时，路径是影响游客游园体验的唯一客观因素），则因路程一定、各个景点游玩时间一定，最佳体验应该为最短路径方式的游园体验。

将入口（即出口）视为第 11 个点，则入口（即出口）与 10 个景点构成一共含 11 个点的无向赋权图，符合“中国邮递员模型”。

记入口（即出口）为 1，A 点为 2，依次推导，J 点为 11，将各个入口与景点、景点与景点之间的距离作为其间边的权，运用哈密顿算法，利用 lingo 软件可得出最短哈密顿路径如下：

X(1, 3) X(3, 6) X(6, 4) X(4, 5) X(5, 7) X(7, 8) X(8, 9) X(9, 10) X(11, 2) X(2, 1)

即：出入口→B→E→C→D→F→G→H→I→J→A→出入口。

游园疏导路线想法：为了先在宏观上有一个有序的引导，不妨将最短路径作为游园的官方推荐路径，并根据可行性，勘测是否能运用在接下来的初态和稳态过程中，亦是验证其合理性。

4.2 初态过程

初态过程因人流量较少，等待时间对游客游园体验造成的影响极小，可以近似忽略不计，因此可以从最短路径及游乐场收益最大化，即：保障游客游玩路线较短时，使得游客在开园后尽快分布到各个景点，不至于资源空置过久的角度设计游乐园推荐旅游路线（显然此过程持续短暂）。而这就需要考察各个景点初始状态的服务效率，即需要建立模型以知道其投入运行的时间。

设{N(t),t≥0}是参数为λ的泊松过程，各个景点的设施容量为 M，则与其对应下的初始状态下的平均排队时间序列$\{W_n, n \geq 1\}$是服从参数为 n 和λ的Γ分布，其概率密度函数为：

$$f_{w_n}(t)=\begin{cases}\lambda e^{-\lambda t}\dfrac{(\lambda t)^{n-1}}{(n-1)!}, t \geq 0\\ 0, t<0\end{cases}$$

则平均排队时间 $W_n=\int_0^{+\infty} t\lambda e^{-\lambda t}\frac{(\lambda t)^{n-1}}{(n-1)!}dt$

参考表 1 数据，由假设（4）知（ λ=7.8，各个景点达到 60%额定容量才开始运行），利用 matlab 程序编程求解可得出下表 2 数据：

表 2：初态过程各个景点投入运营时间

景点	A	B	C	D	E	F	G	H	I	J
等待时间（min）	/	2.31	3.85	2.31	7.69	3.85	2.31	2.31	1.54	3.85

从表中可以看出 A 点容量过大，在初态人流量不高的情况下要达到 60%额定容量始运行所需的时间过长，在一段有限的时间内发生的概率近乎为 0，因此，为了不饶路，在初态过程中游客应该避免先游玩 A 点，而最短路径游园法恰好也是将 A 点置于最后。

同时，为了考虑游乐园效益，游乐园可以给游客推荐建立在最小路径上的逆序游玩法，此情况第一次经过 A 点并没有直接游玩，而是游玩完其他所有景点后由 B 点再返回 A 点，最后离开。

验证结果：从初态过程分析，最小路径及建立在其反序基础上的路径适合作为游园一开场的游客疏导路径，从而一定程度上证明上面游园疏导路线想法的合理性。

4.3 稳态过程

稳态过程因人流量达到高峰，无法忽略排队时间对游客游园体验造成的影响，某些景点甚至会出现超负荷运载，故在稳态阶段需建立一个实时动态疏导模型，以减小超载景点的负荷，保障游客最佳的游园体验。

4.3.1 引力模型

当某个景区处于超负荷状态时，可以用引力模型:找出其最佳的减荷路线。传统的引力模型往往只考虑了分流的距离，忽略游客体验，但根据本例实际情况，在设定引力公式各个参数时除了考虑超载景点的时空负荷率、分流景点的分流能力外，还需要综合考虑游客体验度。

故由万有引力定理可设定此无量纲模型的公式： $F_{ij}^t=G\bullet\frac{Q_i^t\bullet B_j^t}{Z}$

其中 F_{ij}^t 为 t 时刻超载景点 i 对分流景点 j 的吸引力，$i,j\in\{A,B,...,J\}$，G 为引力常数（可取任意正整数），Q_i^t 为 t 时刻超载景点 i 的时空负荷率，B_j^t 为 t 时刻分流景点 j 的分流能力，Z 为按照 i→j 分流路线时游客的游园体验度。

（1）t 时刻超载景点 i 的时空负荷率 $Q_i^t=\frac{N_i^t}{M_i/T_i}=\frac{T_i\bullet N_i^t}{M_i}$

其中 M_i 为超载景点 i 的每场游客容量，N_i^t 为超载 t 时刻景点 i 的实际人数，它包含排队人数和正在游玩人数，T_i 为游客在景点 i 的平均逗留时间。

（2）t 时刻分流景点 j 的分流能力由其在该时刻的可用空闲概率与服务速度

的乘积来决定。可用即为了保障系统的稳定性，不能将所有空闲容量占用，故在空闲概率的基础上需增加一个影响因子 a，因此分流能力 $B_j^t=a(1-\rho)\mu$（ρ 和 μ 的定义在下文出现）。

（3）按照 i→j 分流路线时的阻力 Z 将由两景点之间的距离和游客满意度共同决定。将目标函数设定为 $Z=b(\alpha L+\beta S)$，其中 b 为放大系数，α 为距离疲劳系数（即单位路程游客疲劳感），L 为 i、j 间的距离，β 为惩罚系数，其中 α 和 β 均需要历史观测数据来确定，S 为满意度，参考文献[3]中调查数据图，游客心理随时间变化关系可近似为周期为 $2(T_{min}-T_{max})$ 的 cos 函数，故设定

$$S=\begin{cases}1 & 0<T_j<T_{max}\\ \cos\dfrac{\pi}{T_{min}-T_{max}}(T_j-T_{max}) & T_{max}\le T_j<T_{min}\\ 0 & T_j\ge T_{min}\end{cases},$$

T_j 为分流景点 j 的平均等待时间，T_{max} 为游客满意度为 1 的最大时刻，T_{min} 为游客满意度为 0 的最短时刻。

4.3.2 M/G/1 型排队模型

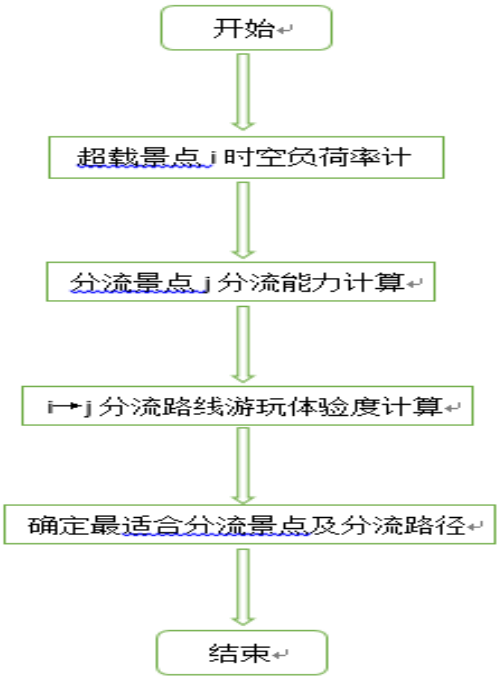
在引力模型公式中分流景点的分流能力和游客满意度都涉及到平均排队时间，因此还需建立一个排队模型来预估稳态时期游客在各个景点所需的排队时间。排队模型的简化:根据上文假设（6），从理论上来说每个景点应该符合 M/D/1/k 型排队模型，即游客到达率服从参数为 λ_2 的泊松分布、服务时间为常数、单服务台的批量服务模型。然而考虑到实际情况下，各批游客进出场所花的时间是不定的，在 $T_{额}$ 附近波动，故根据上文假设（8），可视此排队模型为 M/G/1/K 型排队模型。又考虑到稳态过程流量足够大，根据上文假设（7），进一步将此排队模型视为 M/G/1 型排队模型，即游客到达率服从参数为 $\frac{\lambda_2}{M}$ 的泊松分布、服务时间服从期望为 $T_{额}$ 且方差为 δ 的正太分布的单服务台单个服务模型。

记 $\mu=\frac{M}{T_{额}}$ 为单位时间离开系统的人数（人/分钟），定义 $\rho=\frac{\lambda_2}{\mu}$，则根据参考文献[4]知当 $\rho<1$ 时系统为生灭过程，可稳态发展，根据参考文献[4]可得景点 j 的平均等待时间计算公式为 $T_j=\frac{\rho^2+\lambda_3^2\delta^2}{2\lambda_3(1-\rho)}$。若 $\rho>1$，则该景点会入大于出，将

处于超载荷状态造成拥堵，即此时该景点需要其他景点的协助分流。

求出 T_j 后根据引力模型即可求出各个景点对相邻景点的吸引力 F_{ij}^t，将 F_{ij}^t 按大小排序，综合游乐园布局，从而得出最适合为景点 i 的分流的景点 j。

算法过程图：



根据上文假设（1）稳态时客流量为 15 人/min，取 δ 为 1，依据稳态生灭过程的公式，可得出此流量下各个景点游客的平均等待时间如表（3）所示：

表 3：稳态过程客流量为 15 人/min 时各个景区游客平均排队时间

景点	A	B	C	D	E	F	G	H	I	J
等待时间（min）	/	1.71	4.35	/	7.80	4.35	/	3.25	/	1.88

其中景点 A、D、G、I 的 ρ 值大于 1，不符合生灭过程，即这四个景点在高客流流量期间出现超载荷的概率极大，对这四个景点需要用引力模型对其分流。取引力常数 G=1，可用空闲概率影响因子 a=0.8，放大系数 b 取 10，距离疲劳系数 α 为 0.2， 惩罚系数 β 为-40，游客满意度为 1 的最大时刻 T_{max} 为 4min（实际情况需要考察，这里根据表 3 数据暂定），游客满意度为 0 的最短时刻 T_{min} 为 10min，则可计算出超载景点 A、D、G、I 对与其相邻景点的吸引力，如表（4）所示：

表 4：超载景点对其相邻景点的吸引力

吸引力 分流景点 / 超载荷景点	B	C	E	F	H	J
A	0.036 Q_A		0.013 Q_A			0.08 Q_A
D		0.008 Q_D		0.007 Q_D		
G			0.006 Q_G	0.004 Q_G	0.01 Q_G	
I			0.008 Q_I		0.008 Q_I	0.013 Q_I

注：因 A、D、G、I 不符合生灭过程，无法由排队模型求出其平均排队时间（从而求出平均逗留时间），但不影响吸引力大小的比较，故以字母视为常数出现在表达式中。实际上该逗留时间是可以根据游园的历史数据测得的。

表 4 显示，理论上来说超载荷景点可以向所有与其相邻的可分流景点分流，但充分考虑到游乐园的效益，为了减少资本以及运行稳定，根据上文假设（5），在游客疏导时可参考一开始设定的有向路径去重点调配资源，得出较好的调配路线。

验证结果：从稳态过程分析发现，当某个景点超载荷时其可分流的相邻景点均具备一定的分流能力，即疏导方案是多样的。其中，存在一种方案与最小路径基本相符合，即 I→J， G→H，D→F，A→B 若一开始的有向游园疏导路径设定为最小路径，大多数游客按照疏导路线游玩，则此方案会是在稳态疏导方案中最为经济的。

5、预定酒店数预测模型的建立与分析求解

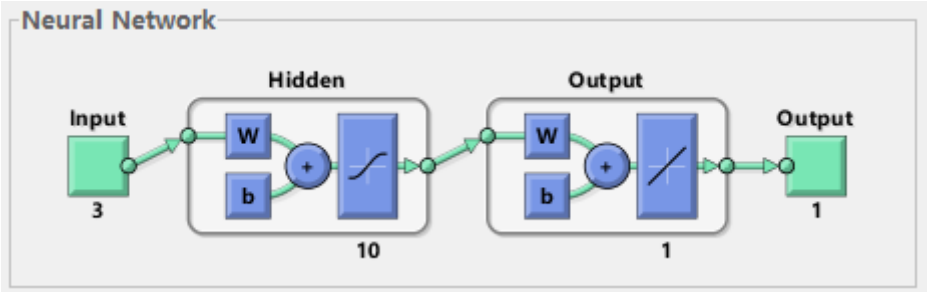
5.1 动态神经网络概述

人工神经网络是一种应用类似于大脑神经突触联接的结构进行信息处理的数学模型。其可分为动态神经网络和静态神经网络。所谓动态神经网络，是指其输入输出之间的变量关系并不仅仅是一种静态方式的映射，每一时刻的输出是基于当前时刻以前系统的动态综合结果而得，因此具有反馈和记忆的功能及逼近任意非线性函数的特点。NAR 回归神经网络是动态神经网络的一种，它由静态神经元与网络的输出反馈两部分构成。

5.2 酒店预定量模型建立

5.2.1 NAR 模型建立原理

一个完整的 NAR 神经网络主要由输入层、隐藏层、输出层等构成。其具体结构示意图如图所示：



左边代表输入数据；右边的代表输出数据；W 代表连接权；b 代表阈值。NAR 两种输出模式 ，一种是 Parallel 模式，在该模式下，输出的数据被反馈到输入端，在隐藏层中继续学习；另一种是 Series-Parallel 模式，该模式下，将期望输出反馈到输入端。本模型采用后者。

NAR 神经网络模型可以表示为：

$$y_n = G(y_{n-1},\cdots,y_{n-k},x_n,\cdots,x_{n-1}) \qquad (1)$$

式中：x—模型输入；y—模型输出；n—离散时间;G—非线性函数。由于该题中实测数据为时间序列，模型输入是未知时，故将式（ 1）改写成式（ 2）：

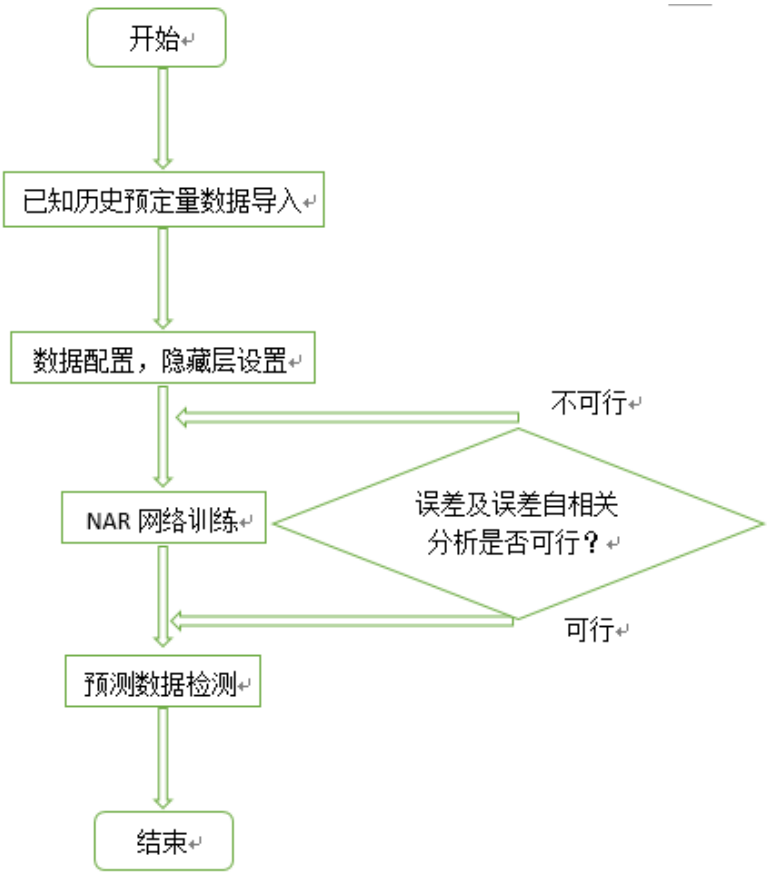
$$y_n = F(y_{n-1},\cdots,y_{n-k}) + k\varepsilon_n \qquad (2)$$

式中：F—非线性函数；k—常数； ε n—服从高斯分配的随机变量。

由神经网络特点知，NAR 模型中每一个输出，都指向元神经网络层的输入当中，作为下一次输出的调整参数，并完成对神经网络的调整。

5.2.2 NAR 模型对酒店预订房间量进行预测

其预测模型如图所示：



5.2.3 相关参数配置

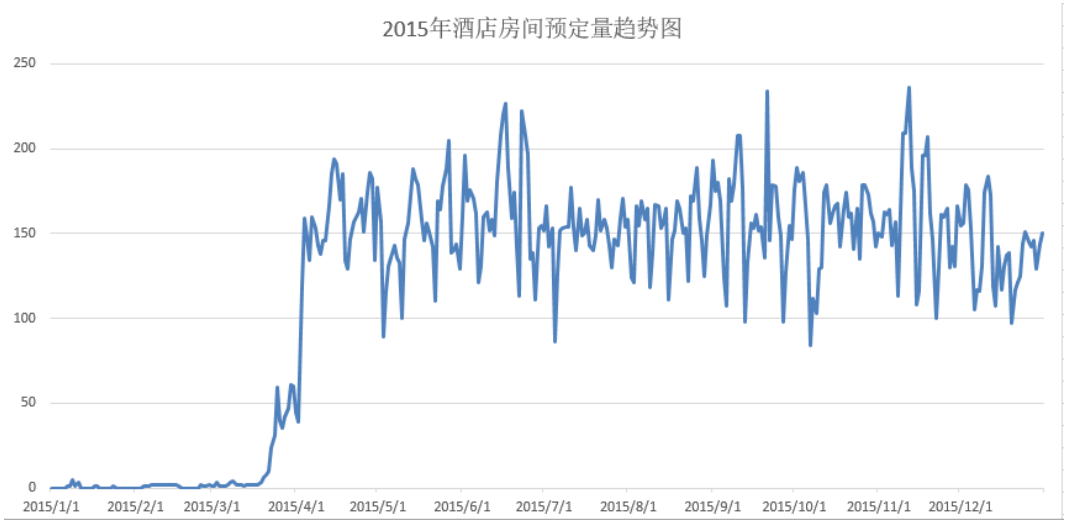
网络训练的数据可分为以下三类：训练集，即用来训练建立预测模型的数据；验证集，即用来验证此网络是否可行；测试集，即用来评估模型预测能力的数据。本文数据设置参数分别为：训练集 70%；验证集 15%；测试集 15%。隐藏层设置为 10。

NAR 神经网络预测 matlab 程序实现见附录 4

5.3 模型求解

5.3.1 原始数据处理

首先对初始数据进行处理，利用 java 编程统计出 2015 年每天预定房间数。程序见附录 5。预定量趋势变化图如下：



由于预定时间与到店时间有一个差值，故可知 2015 年初的部分预订数据是有缺失的。预测时为了尽可能呈现出数据的非线性变化，综合考虑以 7 天为一个预测单元。

5.3.2 分区预测求解

第一部分：2016.1.1-2016.1.7

这部分的房间预定量主要受元旦假期的影响，由于 2015 年初预订数据尤其是 2015.1.1-2015.1.7 严重缺失，考虑到 2015 年年末的预定量能在一定程度上反映和影响 2016 年年初预定情况，故输入数据选择 2015.12.25-2015.12.31。

第二部分：2016.1.8-2016.1.31

这部分的房间预定量主要受季节气候的影响，在假设天气情况基本一致的前提下，选择相应的 2015.1.8-2015.1.31 的数据作为输入数据。

第三部分：2016.2.1-2016.2.29

这部分的房间预定量主要受过年放假回家，置办年货，走亲访友因素的影响，故与之对应的应该选择 2015.2.11-2015.3.12 作为输入数据

第四部分：2016.3.1-2016.3.31

这段时间的房间预定量主要受季节气候的影响，同第二部分，选择 2015.3.1-2015.3.31 作为输入数据。

综上，运用 NAR 神经网络预测得结果如下：

2016年1至3月酒店房间预定量预测结果					
日期	房间数	日期	房间数	日期	房间数
2016/1/1	166	2016/2/1	2	2016/3/3	7
2016/1/2	147	2016/2/2	2	2016/3/4	6
2016/1/3	154	2016/2/3	2	2016/3/5	6
2016/1/4	147	2016/2/4	2	2016/3/6	6
2016/1/5	127	2016/2/5	2	2016/3/7	6
2016/1/6	110	2016/2/6	2	2016/3/8	6
2016/1/7	111	2016/2/7	0	2016/3/9	4
2016/1/8	21	2016/2/8	0	2016/3/10	10
2016/1/9	23	2016/2/9	0	2016/3/11	18
2016/1/10	19	2016/2/10	0	2016/3/12	7
2016/1/11	21	2016/2/11	0	2016/3/13	4
2016/1/12	21	2016/2/12	0	2016/3/14	18
2016/1/13	21	2016/2/13	0	2016/3/15	13
2016/1/14	21	2016/2/14	3	2016/3/16	11
2016/1/15	3	2016/2/15	3	2016/3/17	18
2016/1/16	2	2016/2/16	4	2016/3/18	13
2016/1/17	3	2016/2/17	4	2016/3/19	4
2016/1/18	2	2016/2/18	4	2016/3/20	19
2016/1/19	3	2016/2/19	4	2016/3/21	23
2016/1/20	1	2016/2/20	4	2016/3/22	15
2016/1/21	3	2016/2/21	4	2016/3/23	27
2016/1/22	0	2016/2/22	2	2016/3/24	15
2016/1/23	0	2016/2/23	4	2016/3/25	26
2016/1/24	0	2016/2/24	2	2016/3/26	34
2016/1/25	1	2016/2/25	3	2016/3/27	44
2016/1/26	0	2016/2/26	2	2016/3/28	43
2016/1/27	0	2016/2/27	4	2016/3/29	42
2016/1/28	0	2016/2/28	6	2016/3/30	35
2016/1/29	1	2016/2/29	5	2016/3/31	40
2016/1/30	2	2016/3/1	6		
2016/1/31	2	2016/3/2	6		

6、模型的评价与改进

6.1 游园疏导模型：

6.1.1 优点

（1）为了寻求最佳的游园体验，从路程和效率两大影响因素出发，分阶段（初态和稳态）建立不同模型，为游客的分流和疏导路线做出了合理性的推荐，而且初稳态不同的出发点得到了相似的最佳疏导路线结果，证明该模型存在一定的理论可行性；

（2）将路程和效率两大因素之一的路程定为基础，通过建立效率的模型去验证路程作为单因素时路线规划的合理性，比多目标线性规划更为简便；

（3）引力模型在分流规划中精确度最高，并结合经典的排队模型，从而保证了本模型的准切性；

（4）传统的引力模型在规划分流路线时仅以最大吸引力为标准，但此模型

将其合理的区间化，在考虑游客的游园体验时，还最大化地将游乐园的效益考虑在内；

6.1.2 缺点：

（1）排队模型在假设的基础上做了比较多的近似代替，因没有具体的数据统计，无法判断服务时间是否符合正太分布，用批次间的平均等待时间代替各个游客间的平均等待时间，在流量不够足够大时会存在一定理论上的偏差，且会随容纳量增大而增大；

（2）排队模型的推导公式只能预测符合生灭过程的平均等待时间，高峰期不符合生灭过程的景点只能依靠历史观测数据测得；

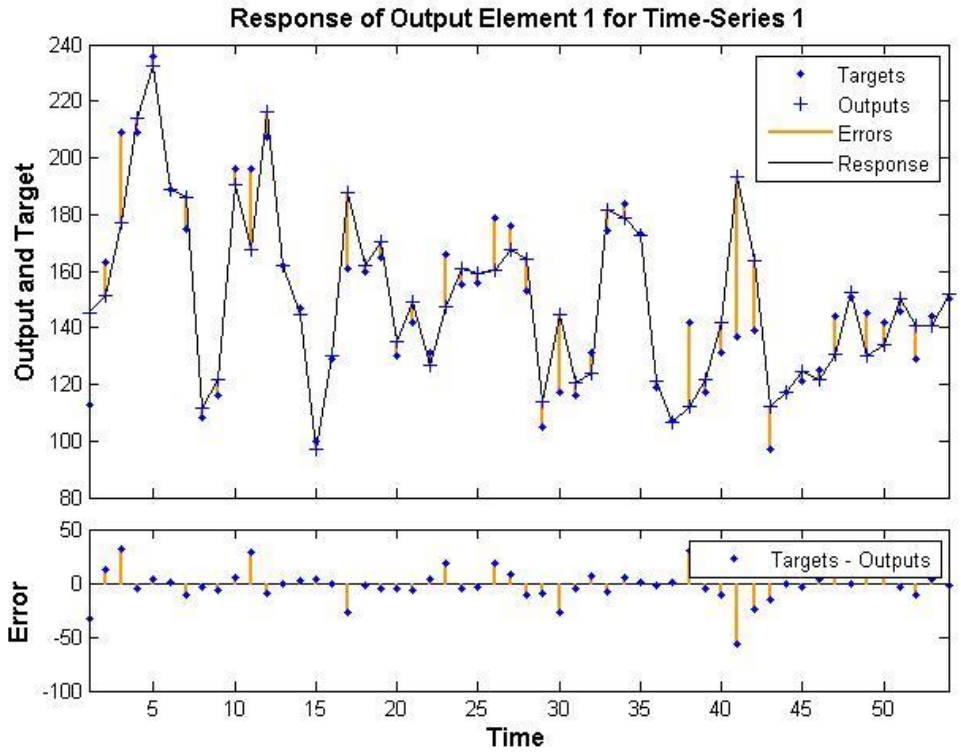
（3）引力模型考虑的因素还偏于简单化，比如说分流能力的大小，除了考虑入流景点自身的承载能力外，还应充分考虑道路的宽度等，阻力除了满足游客的游园体验外还应当充分考虑游乐园运输设施可供量及成本等；

（4）引力模型中有很多影响因子，但由于缺乏现实中的历史数据，使得其假设的线性关系在合理性上存在一些质疑，只能从理论上推导其真实存在。

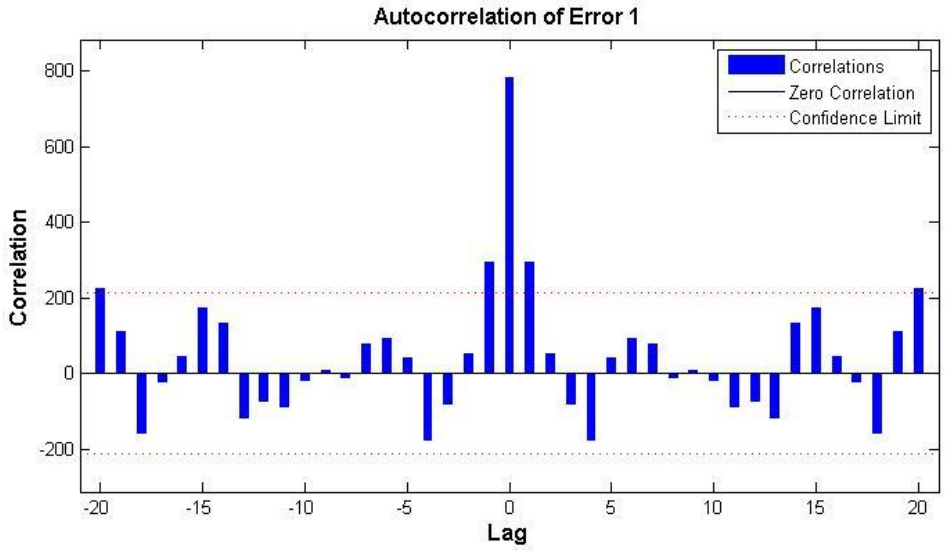
6.2 预定酒店数预测模型：

6.2.1 优点

下图为在预测 2016.1.1-2016.1.7 时得到的误差图及误差自相关图。由误差图可以看出，神经网络训练时误差保持在一个相对合理的范围内，绝大部分误差接近于 0。由误差自相关图则能发现，除了 lag 为 0 及其附近和 lag 取±20 时，其余误差自相关系数均在上下置信区间范围内，按照动态神经网络预测要求可知此模型预测效果较合理。



误差图



误差自相关图

从预测结果来看，由题目原始数据我们得到 2016.1.1 房间预定量为 145，而我们预测结果为 166，误差为 14.5%，对于一个受众多因素影响且样本数据较少的预测而言，这个误差是可以接受的。这进一步说明该模型的预测结果是有一定程度的可信性的。

6.2.2 缺点

经过反思，我们的模型主要有以下几点不足：

（1） 该模型以 7 天为一个预测单位，而 2015 年年初存在着数据缺失，而模型不具备利用已预测的前一单元的数据对现预测单元的修正功能，导致两个单元预测出来的结果在单元衔接处变化较大，与现实情况拟合较差，如 2016.1.7 和 2016.1.8 这两天的预测结果，两者相差接近 80，这显然不合理。

（2） 由于 NAR 神经网络在每一次训练开始时，都会根据一定的方法的给出一组初始的权值，而该组权值每次都是不一样的，再在这个初始的权值的基础上开始训练。所以几乎每次都收敛到不同的局部最优解然后停止训练。这就导致了对同一组数据会产生不同预测结果，且有时这些结果会相差较大。

【参考文献】
[1]姜启源. 数学模型 高等教育出版社,1987
[2]胡家骥. 运筹学 中南工业大学出版社,1987.11
[3]肖雄辉,戈鹏,朱虹明,任佩瑜,郑伟民,章小平. 旅游高峰期景区游客引力分流调度模型研究——以九寨沟风景区为例[J]. 旅游科学,2013,v.27;No.13906:39-51.
[4]刘俊杰,霍光,刘裕博. 基于M/G/1模型的停车场出口排队延误[J]. 交通科学与工程,2015,v.31;No.11601:92-97.
[5]罗芬,廖薇. 主题公园游乐设施游客等待心理变化研究——以长沙世界之窗为例[J]. 中南林业科技大学学报(社会科学版),2010,v.4;No.2003:51-54.
[6]邱厌庆,戈鹏,刘柱胜,任佩瑜. 基于复杂系统控制的景区游客时空分流导航管理研究[J]. 软科学,2011,v.25;No.14109:54-57.
[7]饶亚玲.基于客流疏导的景区游览线路优化研究[D].华侨大学,2015.
[8](美) 哈拉里 (Harary, F.)著 李慰萱译 上海科学技术出版社,1980
[9]吴普,葛全胜.海南旅游客流量年内变化与气候的相关性分析[J].地理研究,2009,v.2804:1078-1084.
[10]汪翔,何吉祥,佘磊,张静.基于NAR神经网络对养殖水体亚硝酸盐预测模型的研究[J].渔业现代化,2015,v.42;No.23704:30-34.
[11]张庆春,赵树魁,金玉子,范晓东.基于季节模型的入境旅游人数的时间序列分析与预测[J].吉林化工学院学报,2014,v.31;No.16501:101-104.
[12]韩路跃,杜行检.基于MATLAB的时间序列建模与预测[J].计算机仿真,2005,04:105-107+182.
[13]杨春波.基于灰色模型与人工神经网络的改进组合预测模型及其应用研究[D].山东师范大学,2009.
[14]肖启伟,杨秀芝.基于NAR模型的电视频道收视率预测[J].电视技术,2015,v.39;No.45704:79-81.

附录：

1、最短路径 lingo 程序

```
model:
sets:
   spot/1..11/:u;
  link(spot,spot):
    dist,
    x;
 endsets
 n=@size(spot);
data:
   dist=
0 300 400 10000 10000 10000 10000 10000 10000 10000 10000
300 0 300 10000 10000 350 10000 10000 10000 10000 250
400 300 0 300 10000 350 10000 10000 10000 10000 10000
10000 10000 300 0 450 500 10000 10000 10000 10000 10000
10000 10000 10000 450 0 10000 500 10000 10000 10000 10000
10000 350 350 500 10000 0 10000 550 10000 450 10000
10000 10000 10000 10000 500 10000 0 650 10000 10000 10000
10000 10000 10000 10000 10000 550 650 0 400 10000 10000
10000 10000 10000 10000 10000 10000 10000 400 0 450 10000
10000 10000 10000 10000 10000 450 10000 10000 450 0 350
10000 250 10000 10000 10000 10000 10000 10000 10000 350 0
 ;
enddata
  min=@sum(link:dist*x);
 @FOR(spot(K):
 @sum(spot(I)|I #ne# K:x(I,K))=1;
 @sum(spot(J)|J #ne# K:x(K,J))=1;
);
@for(spot(I)|I #gt# 1:
  @for(spot(J)|J #gt# 1 #and# I #ne# J:
    u(I)-u(J)+n*x(I,J)<=n-1);
);
@for(spot(I)|I #gt# 1:u(I)<=n-2);
@for(link:@bin(x));
End
```

输出结果为

```
Global optimal solution found.
  Objective value:                              4600.000
  Objective bound:                              4600.000
  Infeasibilities:                           0.8881784E-15
  Extended solver steps:                                0
```

Total solver iterations: 398

Variable	Value	Reduced Cost
N	11.00000	0.000000
U(1)	0.000000	0.000000
U(2)	9.000000	0.000000
U(3)	0.000000	0.000000
U(4)	2.000000	0.000000
U(5)	3.000000	0.000000
U(6)	1.000000	0.000000
U(7)	4.000000	0.000000
U(8)	5.000000	0.000000
U(9)	6.000000	0.000000
U(10)	7.000000	0.000000
U(11)	8.000000	0.000000
DIST(1, 1)	0.000000	0.000000
DIST(1, 2)	300.0000	0.000000
DIST(1, 3)	400.0000	0.000000
DIST(1, 4)	10000.00	0.000000
DIST(1, 5)	10000.00	0.000000
DIST(1, 6)	10000.00	0.000000
DIST(1, 7)	10000.00	0.000000
DIST(1, 8)	10000.00	0.000000
DIST(1, 9)	10000.00	0.000000
DIST(1, 10)	10000.00	0.000000
DIST(1, 11)	10000.00	0.000000
DIST(2, 1)	300.0000	0.000000
DIST(2, 2)	0.000000	0.000000
DIST(2, 3)	300.0000	0.000000
DIST(2, 4)	10000.00	0.000000
DIST(2, 5)	10000.00	0.000000
DIST(2, 6)	350.0000	0.000000
DIST(2, 7)	10000.00	0.000000
DIST(2, 8)	10000.00	0.000000
DIST(2, 9)	10000.00	0.000000
DIST(2, 10)	10000.00	0.000000
DIST(2, 11)	250.0000	0.000000
DIST(3, 1)	400.0000	0.000000
DIST(3, 2)	300.0000	0.000000
DIST(3, 3)	0.000000	0.000000
DIST(3, 4)	300.0000	0.000000
DIST(3, 5)	10000.00	0.000000
DIST(3, 6)	350.0000	0.000000

DIST(3, 7)	10000.00	0.000000
DIST(3, 8)	10000.00	0.000000
DIST(3, 9)	10000.00	0.000000
DIST(3, 10)	10000.00	0.000000
DIST(3, 11)	10000.00	0.000000
DIST(4, 1)	10000.00	0.000000
DIST(4, 2)	10000.00	0.000000
DIST(4, 3)	300.0000	0.000000
DIST(4, 4)	0.000000	0.000000
DIST(4, 5)	450.0000	0.000000
DIST(4, 6)	500.0000	0.000000
DIST(4, 7)	10000.00	0.000000
DIST(4, 8)	10000.00	0.000000
DIST(4, 9)	10000.00	0.000000
DIST(4, 10)	10000.00	0.000000
DIST(4, 11)	10000.00	0.000000
DIST(5, 1)	10000.00	0.000000
DIST(5, 2)	10000.00	0.000000
DIST(5, 3)	10000.00	0.000000
DIST(5, 4)	450.0000	0.000000
DIST(5, 5)	0.000000	0.000000
DIST(5, 6)	10000.00	0.000000
DIST(5, 7)	500.0000	0.000000
DIST(5, 8)	10000.00	0.000000
DIST(5, 9)	10000.00	0.000000
DIST(5, 10)	10000.00	0.000000
DIST(5, 11)	10000.00	0.000000
DIST(6, 1)	10000.00	0.000000
DIST(6, 2)	350.0000	0.000000
DIST(6, 3)	350.0000	0.000000
DIST(6, 4)	500.0000	0.000000
DIST(6, 5)	10000.00	0.000000
DIST(6, 6)	0.000000	0.000000
DIST(6, 7)	10000.00	0.000000
DIST(6, 8)	550.0000	0.000000
DIST(6, 9)	10000.00	0.000000
DIST(6, 10)	450.0000	0.000000
DIST(6, 11)	10000.00	0.000000
DIST(7, 1)	10000.00	0.000000
DIST(7, 2)	10000.00	0.000000
DIST(7, 3)	10000.00	0.000000
DIST(7, 4)	10000.00	0.000000
DIST(7, 5)	500.0000	0.000000
DIST(7, 6)	10000.00	0.000000

DIST(7, 7)	0.000000	0.000000
DIST(7, 8)	650.0000	0.000000
DIST(7, 9)	10000.00	0.000000
DIST(7, 10)	10000.00	0.000000
DIST(7, 11)	10000.00	0.000000
DIST(8, 1)	10000.00	0.000000
DIST(8, 2)	10000.00	0.000000
DIST(8, 3)	10000.00	0.000000
DIST(8, 4)	10000.00	0.000000
DIST(8, 5)	10000.00	0.000000
DIST(8, 6)	550.0000	0.000000
DIST(8, 7)	650.0000	0.000000
DIST(8, 8)	0.000000	0.000000
DIST(8, 9)	400.0000	0.000000
DIST(8, 10)	10000.00	0.000000
DIST(8, 11)	10000.00	0.000000
DIST(9, 1)	10000.00	0.000000
DIST(9, 2)	10000.00	0.000000
DIST(9, 3)	10000.00	0.000000
DIST(9, 4)	10000.00	0.000000
DIST(9, 5)	10000.00	0.000000
DIST(9, 6)	10000.00	0.000000
DIST(9, 7)	10000.00	0.000000
DIST(9, 8)	400.0000	0.000000
DIST(9, 9)	0.000000	0.000000
DIST(9, 10)	450.0000	0.000000
DIST(9, 11)	10000.00	0.000000
DIST(10, 1)	10000.00	0.000000
DIST(10, 2)	10000.00	0.000000
DIST(10, 3)	10000.00	0.000000
DIST(10, 4)	10000.00	0.000000
DIST(10, 5)	10000.00	0.000000
DIST(10, 6)	450.0000	0.000000
DIST(10, 7)	10000.00	0.000000
DIST(10, 8)	10000.00	0.000000
DIST(10, 9)	450.0000	0.000000
DIST(10, 10)	0.000000	0.000000
DIST(10, 11)	350.0000	0.000000
DIST(11, 1)	10000.00	0.000000
DIST(11, 2)	250.0000	0.000000
DIST(11, 3)	10000.00	0.000000
DIST(11, 4)	10000.00	0.000000
DIST(11, 5)	10000.00	0.000000
DIST(11, 6)	10000.00	0.000000

DIST(11, 7)	10000.00	0.000000
DIST(11, 8)	10000.00	0.000000
DIST(11, 9)	10000.00	0.000000
DIST(11, 10)	350.0000	0.000000
DIST(11, 11)	0.000000	0.000000
X(1, 1)	0.000000	0.000000
X(1, 2)	0.000000	300.0000
X(1, 3)	1.000000	400.0000
X(1, 4)	0.000000	10000.00
X(1, 5)	0.000000	10000.00
X(1, 6)	0.000000	10000.00
X(1, 7)	0.000000	10000.00
X(1, 8)	0.000000	10000.00
X(1, 9)	0.000000	10000.00
X(1, 10)	0.000000	10000.00
X(1, 11)	0.000000	10000.00
X(2, 1)	1.000000	300.0000
X(2, 2)	0.000000	0.000000
X(2, 3)	0.000000	300.0000
X(2, 4)	0.000000	10000.00
X(2, 5)	0.000000	10000.00
X(2, 6)	0.000000	350.0000
X(2, 7)	0.000000	10000.00
X(2, 8)	0.000000	10000.00
X(2, 9)	0.000000	10000.00
X(2, 10)	0.000000	10000.00
X(2, 11)	0.000000	250.0000
X(3, 1)	0.000000	400.0000
X(3, 2)	0.000000	300.0000
X(3, 3)	0.000000	0.000000
X(3, 4)	0.000000	300.0000
X(3, 5)	0.000000	10000.00
X(3, 6)	1.000000	350.0000
X(3, 7)	0.000000	10000.00
X(3, 8)	0.000000	10000.00
X(3, 9)	0.000000	10000.00
X(3, 10)	0.000000	10000.00
X(3, 11)	0.000000	10000.00
X(4, 1)	0.000000	10000.00
X(4, 2)	0.000000	10000.00
X(4, 3)	0.000000	300.0000
X(4, 4)	0.000000	0.000000
X(4, 5)	1.000000	450.0000
X(4, 6)	0.000000	500.0000

X(4, 7)	0.000000	10000.00
X(4, 8)	0.000000	10000.00
X(4, 9)	0.000000	10000.00
X(4, 10)	0.000000	10000.00
X(4, 11)	0.000000	10000.00
X(5, 1)	0.000000	10000.00
X(5, 2)	0.000000	10000.00
X(5, 3)	0.000000	10000.00
X(5, 4)	0.000000	450.0000
X(5, 5)	0.000000	0.000000
X(5, 6)	0.000000	10000.00
X(5, 7)	1.000000	500.0000
X(5, 8)	0.000000	10000.00
X(5, 9)	0.000000	10000.00
X(5, 10)	0.000000	10000.00
X(5, 11)	0.000000	10000.00
X(6, 1)	0.000000	10000.00
X(6, 2)	0.000000	350.0000
X(6, 3)	0.000000	350.0000
X(6, 4)	1.000000	500.0000
X(6, 5)	0.000000	10000.00
X(6, 6)	0.000000	0.000000
X(6, 7)	0.000000	10000.00
X(6, 8)	0.000000	550.0000
X(6, 9)	0.000000	10000.00
X(6, 10)	0.000000	450.0000
X(6, 11)	0.000000	10000.00
X(7, 1)	0.000000	10000.00
X(7, 2)	0.000000	10000.00
X(7, 3)	0.000000	10000.00
X(7, 4)	0.000000	10000.00
X(7, 5)	0.000000	500.0000
X(7, 6)	0.000000	10000.00
X(7, 7)	0.000000	0.000000
X(7, 8)	1.000000	650.0000
X(7, 9)	0.000000	10000.00
X(7, 10)	0.000000	10000.00
X(7, 11)	0.000000	10000.00
X(8, 1)	0.000000	10000.00
X(8, 2)	0.000000	10000.00
X(8, 3)	0.000000	10000.00
X(8, 4)	0.000000	10000.00
X(8, 5)	0.000000	10000.00
X(8, 6)	0.000000	550.0000

X(8, 7)	0.000000	650.0000
X(8, 8)	0.000000	0.000000
X(8, 9)	1.000000	400.0000
X(8, 10)	0.000000	10000.00
X(8, 11)	0.000000	10000.00
X(9, 1)	0.000000	10000.00
X(9, 2)	0.000000	10000.00
X(9, 3)	0.000000	10000.00
X(9, 4)	0.000000	10000.00
X(9, 5)	0.000000	10000.00
X(9, 6)	0.000000	10000.00
X(9, 7)	0.000000	10000.00
X(9, 8)	0.000000	400.0000
X(9, 9)	0.000000	0.000000
X(9, 10)	1.000000	450.0000
X(9, 11)	0.000000	10000.00
X(10, 1)	0.000000	10000.00
X(10, 2)	0.000000	10000.00
X(10, 3)	0.000000	10000.00
X(10, 4)	0.000000	10000.00
X(10, 5)	0.000000	10000.00
X(10, 6)	0.000000	450.0000
X(10, 7)	0.000000	10000.00
X(10, 8)	0.000000	10000.00
X(10, 9)	0.000000	450.0000
X(10, 10)	0.000000	0.000000
X(10, 11)	1.000000	350.0000
X(11, 1)	0.000000	10000.00
X(11, 2)	1.000000	250.0000
X(11, 3)	0.000000	10000.00
X(11, 4)	0.000000	10000.00
X(11, 5)	0.000000	10000.00
X(11, 6)	0.000000	10000.00
X(11, 7)	0.000000	10000.00
X(11, 8)	0.000000	10000.00
X(11, 9)	0.000000	10000.00
X(11, 10)	0.000000	350.0000
X(11, 11)	0.000000	0.000000

2、初态过程平均等待时间 matlab 程序

```
function f=low(a,k)
 syms t
f=int((a^k*t^k*exp(-a*t))/factorial(k-1),t,0,inf);
end
```

3、稳态过程平均等待时间 matlab 程序

```
function t=high(a,k,e)
v=a/k;
p=v*e;
t=(p^2+v^2)/(2*v*(1-p));
end
```

4、matlab 实现 NAR 神经网络预测程序

```
clear
clc
x=[ ]'; % 输入 7 个输入数据
lag=3;
iinput=x;
n=length(iinput);
inputs=zeros(lag,n-lag);
for i=1:n-lag
    inputs(:,i)=iinput(i:i+lag-1)';
end
targets=x(lag+1:end);
hiddenLayerSize = 10;  %隐藏层神经元个数
net = fitnet(hiddenLayerSize);
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
[net,tr] = train(net,inputs,targets);
yn=net(inputs);
errors=targets-yn;
figure, ploterrcorr(errors)
figure, parcorr(errors)
%[h,pValue,stat,cValue]= lbqtest(errors)
figure,plotresponse(con2seq(targets),con2seq(yn))
%figure, ploterrhist(errors)
%figure, plotperform(tr)
fn=7;  %预测步数为 fn。
f_in=iinput(n-lag+1:end)';
f_out=zeros(1,fn);
for i=1:fn
    f_out(i)=net(f_in);
    f_in=[f_in(2:end);f_out(i)];
end
figure,plot(1:7,iinput,'b',7:14,[iinput(end),f_out],'r')
```

5、java 实现 2015 年预定房间数统计程序

```
//ReadData.java
```

```
package readDataFromExcel;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import jxl.Cell;
import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;

import jxl.*;
import jxl.format.UnderlineStyle;
import jxl.write.*;
import jxl.write.Number;
import jxl.write.Boolean;
import java.io.*;


/**
 * 读取 excel 公共方法
 *
 *
 */
public class ReadData {
    /**
     *
     * @param excelFile
     *            读取文件对象
     * @param rowNum
     *            从第几行开始读，如果有一行表头则从第二行开始读
     * @return
     * @throws BiffException
     * @throws IOException
     */
    public static List<String[]> readExcel(File excelFile, int rowNum)
            throws BiffException, IOException {
        // 创建一个 list 用来存储读取的内容
        List<String[]> list = new ArrayList<String[]>();
```

```
        Workbook rwb = null;
        Cell cell = null;
        // 创建输入流
        InputStream stream = new FileInputStream(excelFile);
        // 获取 Excel 文件对象
        rwb = Workbook.getWorkbook(stream);
        // 获取文件的指定工作表 默认的第一个
        Sheet sheet = rwb.getSheet(0);
        // 行数(表头的目录不需要，从 1 开始)
        for (int i = rowNum - 1; i < sheet.getRows(); i++) {
            // 创建一个数组 用来存储每一列的值
            String[] str = new String[sheet.getColumns()];
            // 列数
            for (int j = 0; j < sheet.getColumns(); j++) {
                // 获取第 i 行，第 j 列的值
                cell = sheet.getCell(j, i);
                str[j] = cell.getContents();
//              System.out.println(str[j]);
            }
            // 把刚获取的列存入 list
            list.add(str);
        }
        // 返回值集合
        return list;
    }

    //输入查询日期
    public static String[] getDate(String target){
//      Scanner scan = new Scanner(System.in);
//      String target = scan.nextLine();
//      System.out.println(read);
        String subData[]=target.split("-");
//      for (int i = 0 ; i <subData.length ; i++ ) {
//            System.out.println("--"+subData[i]);
//          }
        return subData;
    }

    //比较是否住宿
    public static boolean isInHotel(String p_beginDate,String p_endDate,String
p_targetDate){
        String beginDate[]=getDate(p_beginDate);
        String endDate[]=getDate(p_endDate);
        String targetDate[]=getDate(p_targetDate);
```

```
        if (Integer.parseInt(targetDate[0])< Integer.parseInt(beginDate[0]) ||
Integer.parseInt(targetDate[0]) > Integer.parseInt(endDate[0])) {
            //年不满足
            return false;
        } else {
            //没有跨年
            if (Integer.parseInt(beginDate[0]) == Integer.parseInt(endDate[0])) {
                if (Integer.parseInt(targetDate[1])<
Integer.parseInt(beginDate[1]) || Integer.parseInt(targetDate[1]) >
Integer.parseInt(endDate[1])) {
                    //月不满足
                    return false;
                } else {
                    //同月
                    if (Integer.parseInt(beginDate[1])==
Integer.parseInt(endDate[1])) {
                        if (Integer.parseInt(targetDate[2])<
Integer.parseInt(beginDate[2]) || Integer.parseInt(targetDate[2]) >=
Integer.parseInt(endDate[2])) {
                            return false;
                        } else {
                            return true;
                        }
                    } else {
                        //不同月
                        if (Integer.parseInt(beginDate[1])==
Integer.parseInt(targetDate[1])) {
                            //与前月同月
                            if (Integer.parseInt(beginDate[2]) <=
Integer.parseInt(targetDate[2])) {
                                return true;
                            } else {
                                return false;
                            }
                        } else {
                            //与后月同月
                            if (Integer.parseInt(endDate[2]) >
Integer.parseInt(targetDate[2])) {
                                return true;
                            } else {
                                return false;
                            }
                        }
                    }
```

```
                }

            } else {
                //跨年
                if (Integer.parseInt(beginDate[0]) ==
Integer.parseInt(targetDate[0])) {
                    //与前年同年
                    if (Integer.parseInt(beginDate[1]) >
Integer.parseInt(targetDate[1])) {
                        return false;
                    } else if(Integer.parseInt(beginDate[1]) ==
Integer.parseInt(targetDate[1])){
                        if (Integer.parseInt(beginDate[2]) >
Integer.parseInt(targetDate[2])) {
                            return false;
                        } else {
                            return true;
                        }
                    }else{
                        return true;
                    }
                } else {
                    //与后年同年
                    if (Integer.parseInt(endDate[1]) <
Integer.parseInt(targetDate[1])) {
                        return false;
                    } else if(Integer.parseInt(endDate[1]) ==
Integer.parseInt(targetDate[1])){
                        if (Integer.parseInt(endDate[2]) <=
Integer.parseInt(targetDate[2])) {
                            return false;
                        } else {
                            return true;
                        }
                    }else{
                        return true;
                    }
                }
            }
        }
    }


    public static void main(String[] args) {
```

```
        String excelFileName = "C:\\data5.xls";
        String excelFileName2 = "C:\\date.xls";

        try {
            List<String[]> list = ReadData.readExcel(
                    new File(excelFileName), 2);
            List<String[]> list2 = ReadData.readExcel(new File(excelFileName2),
1);
            for (int j = 0; j < list2.size(); j++) {
                String[] str2 = (String[]) list2.get(j);
                int count = 0;
                for (int i = 0; i < list.size(); i++) {
                    String[] str = (String[]) list.get(i);
//                  for (int j = 0; j < str.length; j++) {
//                      System.out.println(str[j]);
//                      System.out.println(str.length);
                        if (isInHotel(str[1], str[2],str2[0])) {
                            count=count+Integer.parseInt(str[3]);

                        }
//                  }
                }
                System.out.println(str2[0]+"\t"+count);
            }

//
        } catch (BiffException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```