第十讲: 收费广场收费亭的最优数量

数学模型和算法的应用与 MATLAB 实现

周吕文

中国科学院力学研究所

2017 年 7 月 29 日

微信公众号: 超级数学建模

Part I

题目与分析

提要

1. 赛题
   - 英文赛题
   - 中文赛题
2. 赛题分析
   - 正确理解题义

2005 MCM PROBLEM B: Tollbooths

Heavily-traveled toll roads such as the Garden State Parkway, Interstate 95, and so forth, are multi-lane divided highways that are interrupted at intervals by toll plazas. Because collecting tolls is usually unpopular, it is desirable to minimize motorist annoyance by limiting the amount of traffic disruption caused by the toll plazas. Commonly, a much larger number of tollbooths is provided than the number of travel lanes entering the toll plaza. Upon entering the toll plaza, the flow of vehicles fans out to the larger number of tollbooths, and when leaving the toll plaza, the flow of vehicles is required to squeeze back down to a number of travel lanes equal to the number of travel lanes before the toll plaza. Consequently, when traffic is heavy, congestion increases upon departure from the toll plaza. When traffic is very heavy, congestion also builds at the entry to the toll plaza because of the time required for each vehicle to pay the toll.

Make a model to help you determine the optimal number of tollbooths to deploy in a barrier-toll plaza. **Explicitly consider the scenario where there is exactly one tollbooth per incoming travel lane. Under what conditions is this more or less effective than the current practice?** Note that the definition of "optimal" is up to you to determine.

2005 MCM 问题 B: 收费亭的最优数量

像 Garden State Parkway, Interstate 95 等等这样的长途收费公路, 通常是多条车道的高速公路, 这些高速公路被通行税收费广场分成几段. 因为征收通行税通常不受欢迎, 所以应尽量减少通过通行税收费广场引起的交通混乱给汽车司机带来的烦恼. 通常, 收费亭的数量要多于进入收费广场道路的数量. 进入通行税收费广场的时候, 驶向大量收费亭的车流呈扇形展开, 当离开通行税收费广场的时候, 车流只能汇聚到与进入收费广场车道数相同的高速公路上. 从而, 当交通是拥挤的时, 拥挤在通行税广场的出口处增加. 当交通非常拥挤的时候, 因为车辆需要时间来付通行费, 因此阻塞也会出现在通行税收费广场的入口处.

建立一个模型来确定在一个容易造成阻塞的通行税收费广场中应该部署的最优的收费亭数量. 需要额外的考虑每一个进入收费广场的车道上都仅有一个收费亭的情况. 与通常情况相比, 在什么条件下这或多或少有效? 注意: "最佳" 的定义由你自己决定.

提要

正确理解题义

建立模型来确定公路收费广场中应该部署的最优的收费亭的数量，“最优”是自己定义.

- 一般情况下，收费亭数量要多于进入广场的车道数量.
- 在车流量比较大时，堵塞不仅会发生在收费亭后的瓶颈，也会由于等待收费而发生成收费亭前.
- 要额外的考虑一条车道仅对应一个收费亭的情况，并说明这种情况在何时效率高于现行的方式.

Part II

The Booth Tolls for Thee

提要

成功模型的性质

- 通过降低车辆通过的时间来最大化效率.
- 给出收费站操作者合理的可执行的政策.
- 要有足够的鲁棒性, 能够处理不同的交通流量.
- 如果车道增加, 收费亭的数量不应减少.

假设和目标

全局假设

- 所有的车辆性质相同, 收费站对其的服务也都都相同的.
- 进入收费广场的车流量与车道数相关, 与收费亭的数量无关.
- 同一天中, 运行的收费亭数量不变.

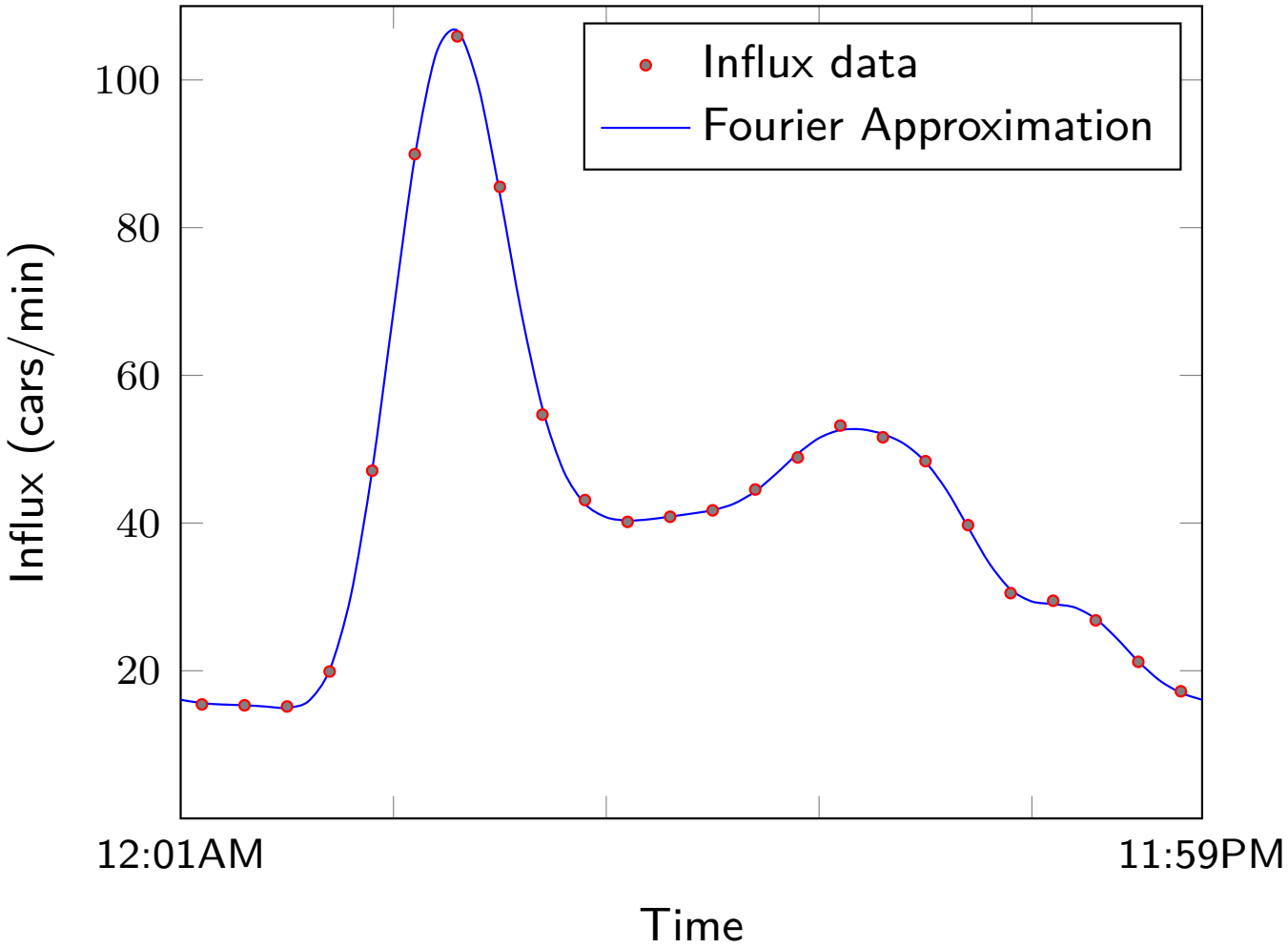
优化目标: 乘客时间价值 $W\alpha N\gamma$ 及收费站运营成本 QB

$$C(B) = W\alpha N\gamma + QB$$

- W 为车辆的平均等待时间; α 平均时间价值;
- N 是一天中通过广场的车辆数; γ 是车辆平均载员数.
- B 为收费亭的数量; Q 单个收费亭一天的操作成本.

车流量函数

$$F(t) = a_0 + \sum_{i=1}^{8} a_i \cos(it\omega) + b_i \sin(it\omega),\ \omega = 2\pi/24$$



提要

模型假设 & 期望

假设

- 收费亭对车辆的服务速度服从指数分布.
- 交通流量来自每一条车道, 各车道上的交通流量相同.
- 本模型不考虑瓶颈: 当收费亭数多于车道数时存在瓶颈效应.
- 若增加一个收费亭不能将时间降低一个**阈值**, 则无需增加.
- 若收费亭都被占用, 新来的车将选择最早结束服务的收费亭.
- 所有车辆 (行为和决定) 目标: 减少等待时间.

期望

- 增加收费亭不应该增加车辆在收费亭前排队等待的时间.
- 每一个额外增加的收费亭所节省的待等时间递减.

模型建立

服务时刻

$$\text{服务时刻}(i) = \max\left\{\text{到达时刻}(i),\ \min\{\text{收费亭空闲时刻}(j)\}\right\}$$

离开时刻

$$\text{离开时刻}(i) = \text{服务时刻}(i) + \text{服务时长}(i)$$

等待时间

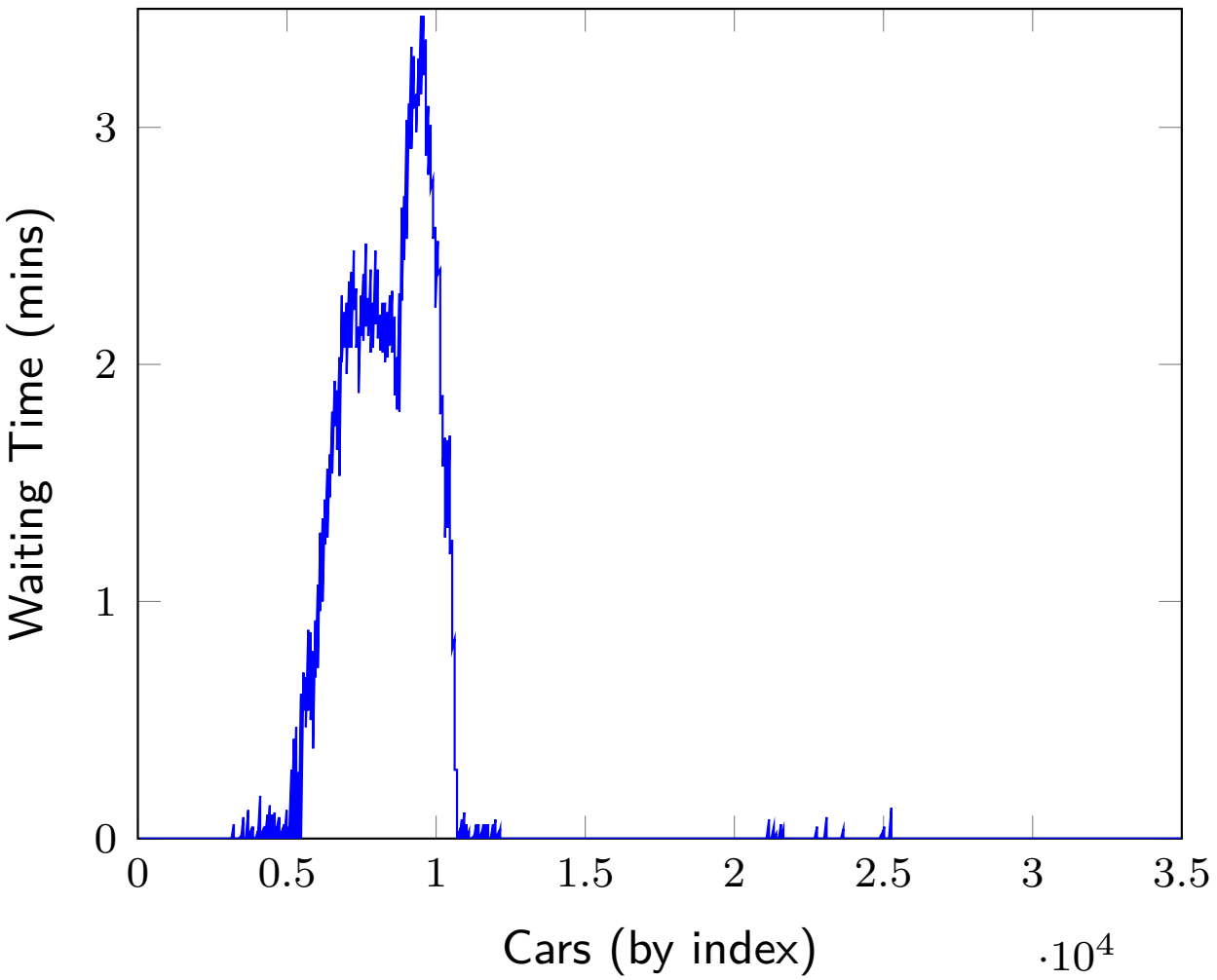
$$\text{等待时间}(i) = \text{离开时刻}(i) - \text{到达时刻}(i)$$

模型求解: 6 车道公路的最佳收费亭数量确定

收费亭	平均等待	平均等待 2	最大等待	边际效用
6	28.2683	43.0457	98.6196	N/A
7	12.1237	27.7573	55.1100	43.5096
8	5.9544	16.5089	31.6650	23.4450
9	2.4111	8.3740	15.8357	15.8293
10	0.2450	1.2188	2.7815	13.0542
11	0.0223	0.1666	0.7482	2.0333
12	0.0039	0.0653	0.3100	0.4382
13	0.0012	0.0410	0.2699	0.0401

- 阈值: 车辆接收服务的平均时间 (0.2 分钟).
- 结果: $L = 6 \longrightarrow B = 12$

模型求解: 6 车道 10 收费亭每辆车的等待时间



结果与分析

车道	收费亭
1	4
2	5
3	7
4	8
5	10
6	12
7	13
8	16
16	29

满足期望

- 高速公路车道数不会超过收费亭的数量.
- 增加收费亭不增加车辆排队等待的时间.
- 每一个增加的收费亭所节省时间递减.

贡献 & 缺点

- 贡献：给出了收费亭数量的上限.
- 缺点：没有考虑瓶颈效应.

模型方法 & 变量定义

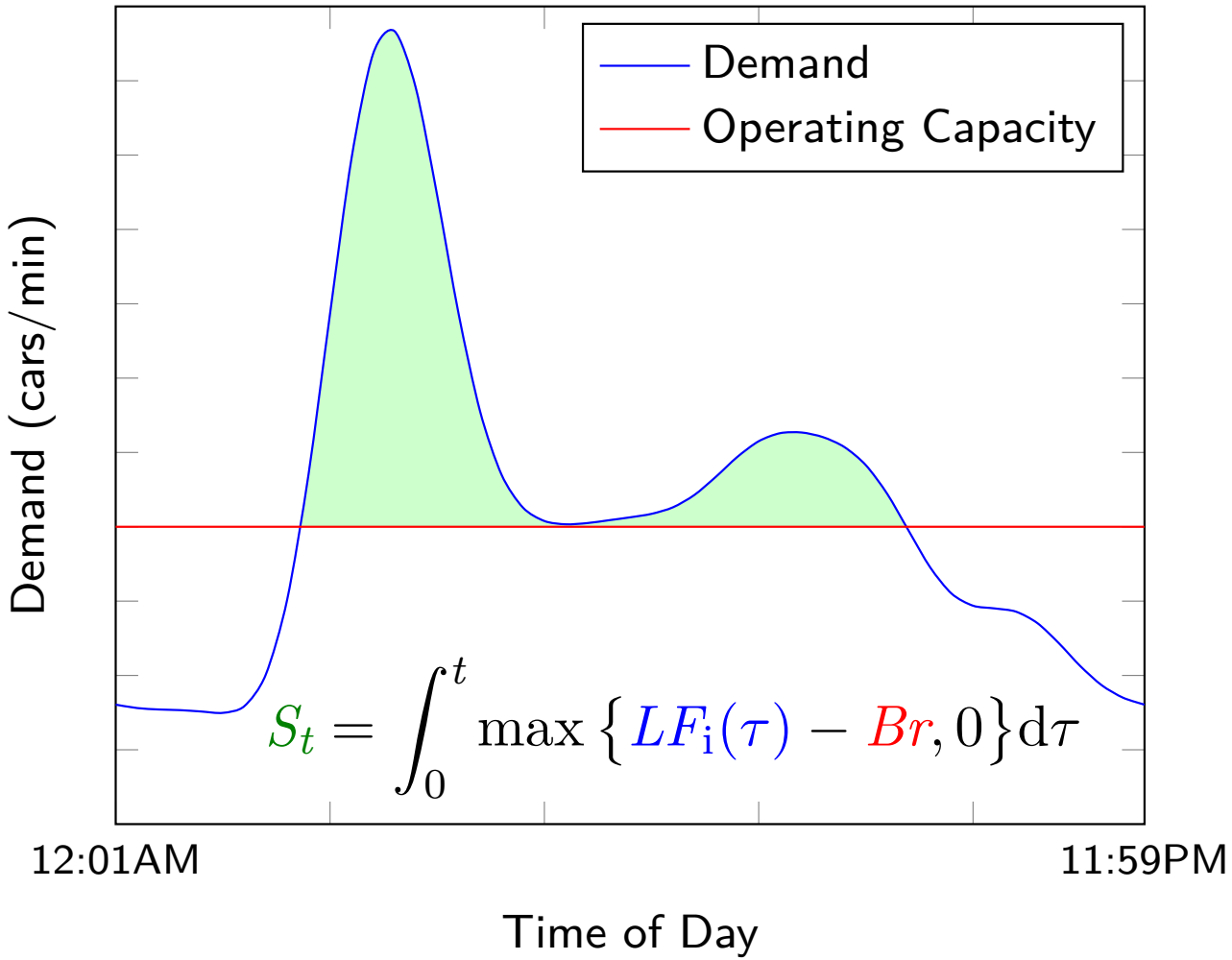
模型方法

- 整体效果: 不关注个体运动细节, 只在意所有车辆整体效果.
- 瓶颈效应: 忽略并道细节; 车流量超过阈值时, 产生瓶颈.

变量定义

- 平均等待时间 $W =$ 队列中的时间 $W_1 +$ 服务时间 $W_2 +$ 瓶颈时间 W_3.
- F_{i}: 一条车道每分钟进入广场的车辆数. 则总入流量为 LF_{i}.
- F_{o}: 每分钟从收费亭出来的车辆数.
- r [车/分钟]: 最大服务率. 则有 $W_2 = 1/r$.
- 出流障碍 K [车/分钟]: 当出流量超过它时发生瓶颈效应.

模型建立


Demand
Operating Capacity
Demand (cars/min)
$S_t = \int_0^t \max\{LF_i(\tau) - Br, 0\}\mathrm{d}\tau$
12:01AM
11:59PM
Time of Day

模型建立

每辆车的平均等待时间

$$W_1 = \frac{3600}{N} \int_0^{24} \int_0^t \max\left(LF_i(\tau) - Br,\ 0\right) \mathrm{d}\tau\, \mathrm{d}t$$

每辆车的平均服务时间

$$W_2 = \frac{1}{r}$$

出口的瓶颈引起的时间

$$W_3 = \frac{3600}{N} \int_0^{24} \int_0^t \max\left(F_{\mathrm{o}}(\tau, B) - K,\ 0\right) \mathrm{d}\tau\, \mathrm{d}t$$

模型求解: 6 车道 $F_o(\tau, B)$ 曲面拟合

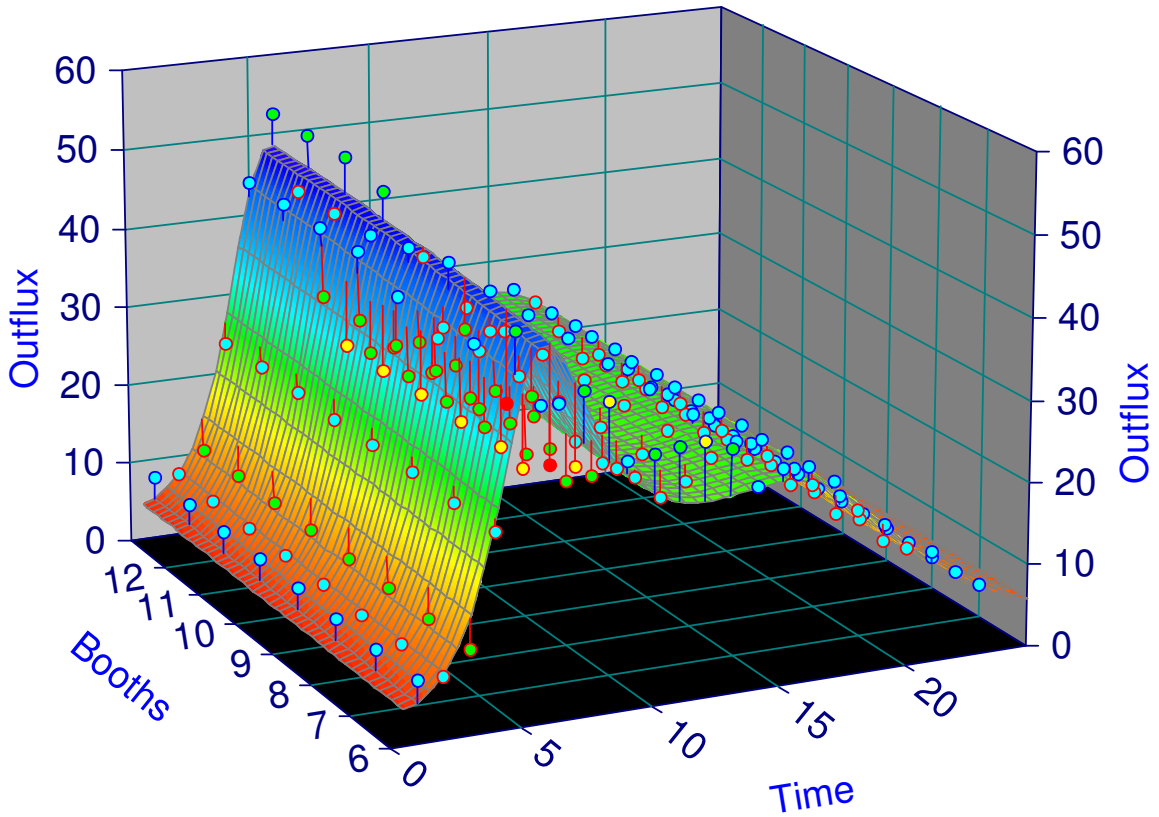
Outflux (cars/min) -- 6 Lane

Rank 4 Eqn 317016996 z^(-1)=a+bx^(1.5)+cx^2+dx^2lnx+ex^(2.5)+fx^3+ge^(x/wx)+h/lny

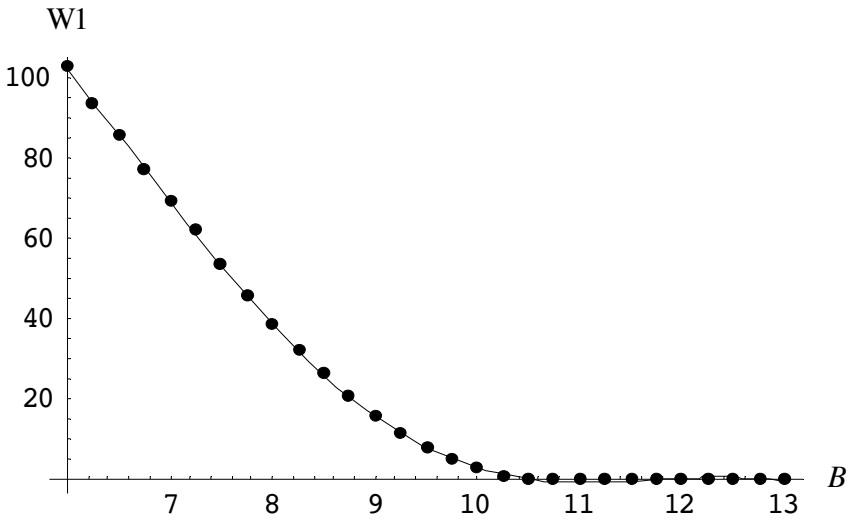
r^2=0.89795631 DF Adj r^2=0.89349539 FitStdErr=3.5049628 Fstat=231.30704

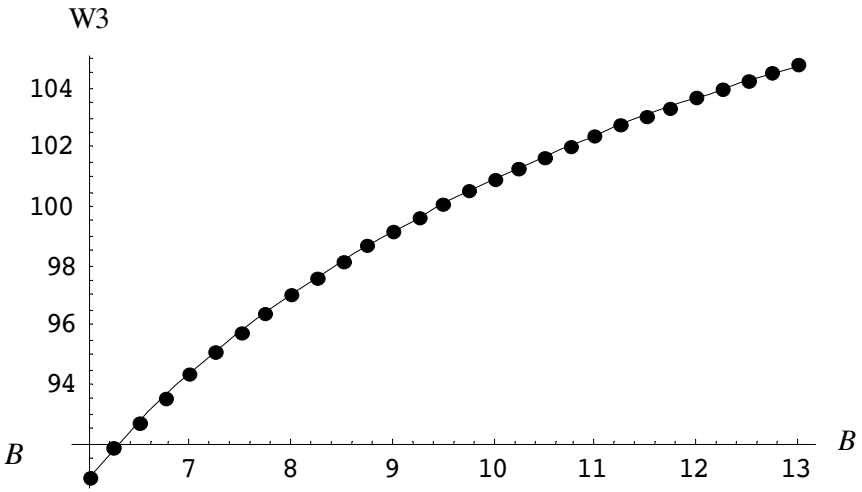
a=-220.32191 b=-2.9096518 c=1.5952541 d=-0.63655404

e=0.23714718 f=-0.0079954663 g=220.43421 h=0.010602912



模型求解: 6 车道 W_1, W_3 与 B 的拟合





最佳收费亭数量的确定

- 四次多项式拟合: W_1-B; W_3-B.
- 令 $\mathrm{d}W/\mathrm{d}B = 0$(三次多项式函数) 可求得最佳 B.

结果与分析

车道	收费亭
1	3
2	5
3	6
4	7
5	9
6	11
7	12
8	14
16	27

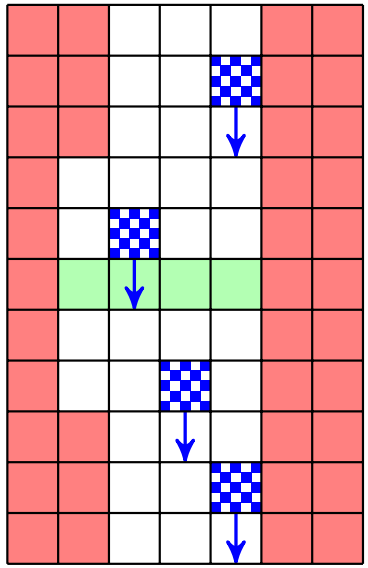
满足期望

- 收费亭数量 B 随 L 单调递增.
- 结果在模型 1 给出的“上界”之下.
- 每车道一个收费的设置从来都不是最佳.

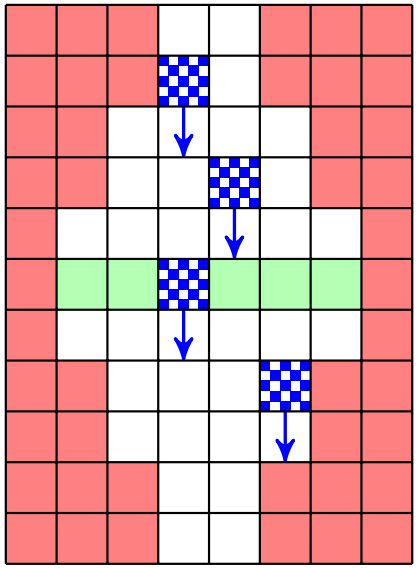
优点 & 缺点

- 优点：考虑瓶颈，能给出问题的公式描述.
- 缺点：缺乏鲁棒性，掩盖了交通行为细节.

模型假设



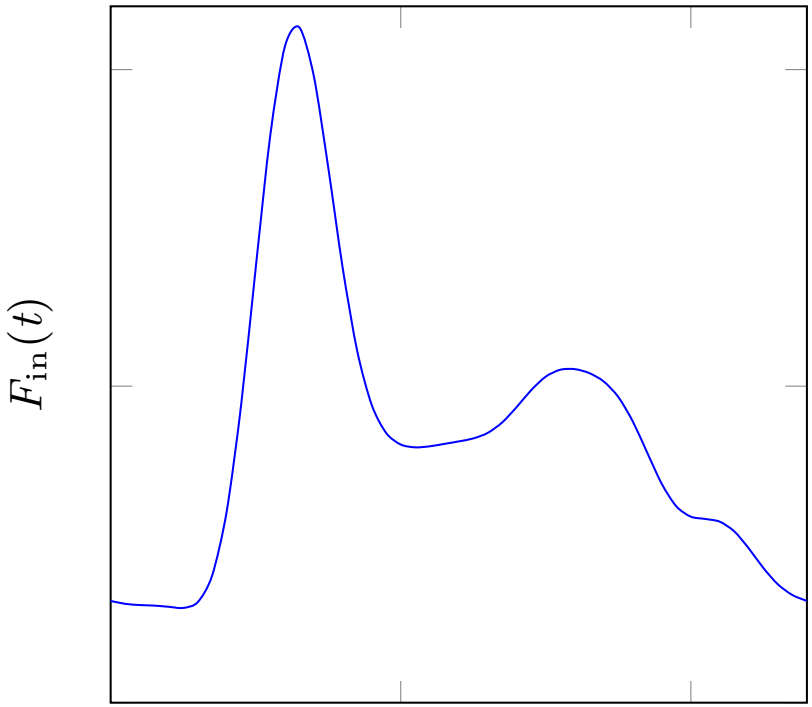
$\mathrm{mod}(B-L,2)=0$

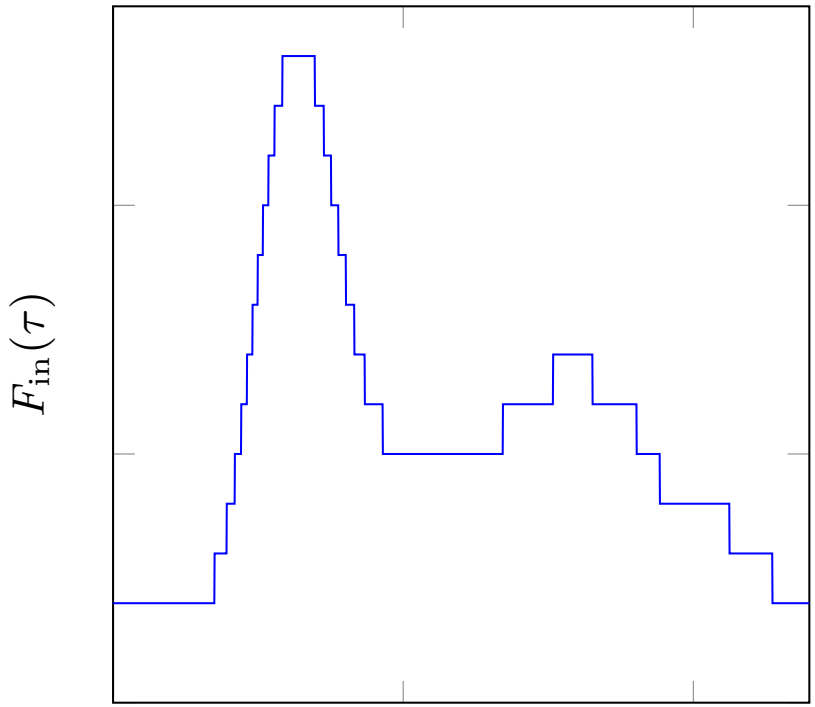


$\mathrm{mod}(B-L,2)=1$

- 收费广场中有三种元胞：车辆元胞，空置元胞，禁止元胞.
- 元胞代表一个可容纳标准车辆的物理空间以及车辆前后缓冲区域.
- 所有的车都具有相同的尺寸.

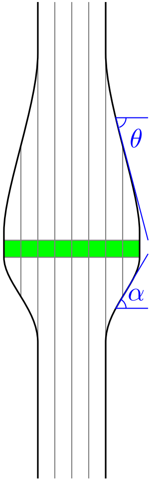
模型建立：入流量的离散标准化

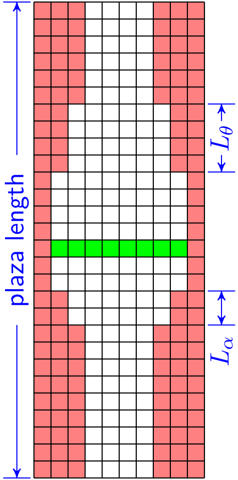




$$F'_{\text{in}}(\tau) = \min\left[\text{round}\left(\frac{F_{\text{in}}(t)}{\eta}\right), L\right]$$

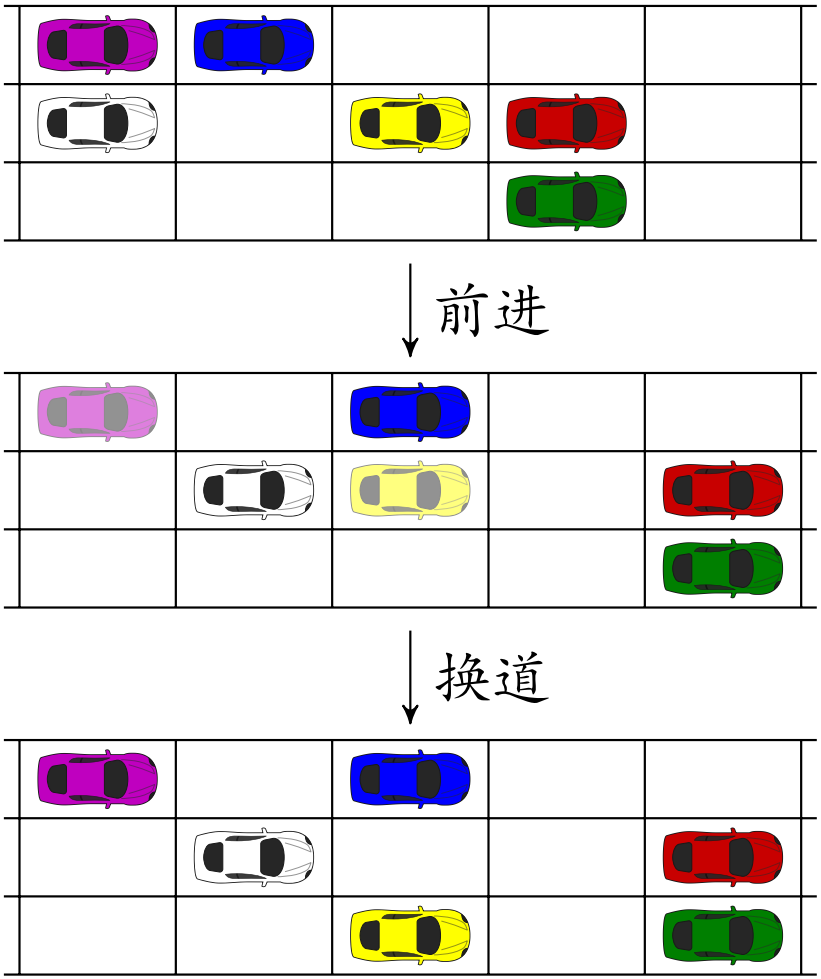
模型建立：收费站平面图数组化





-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	0	0	0	0	0	0	-88	-88
-88	-88	0	0	0	0	0	0	-88	-88
-88	-88	0	0	0	0	0	0	-88	-88
-88	-88	0	0	0	0	0	0	-88	-88
-88	0	0	0	0	0	0	0	0	-88
-88	0	0	0	0	0	0	0	0	-88
-88	0	0	0	0	0	0	0	0	-88
-88	0	0	0	0	0	0	0	0	-88
-88	0	0	0	0	0	0	0	0	-88
-88	0	0	0	0	0	0	0	0	-88
-88	0	0	0	0	0	0	0	0	-88
-88	-88	0	0	0	0	0	0	-88	-88
-88	-88	0	0	0	0	0	0	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88
-88	-88	-88	0	0	0	0	-88	-88	-88

模型建立：动力学规则



前进规则

- 若 t 时刻 i 位置状态是车，且 $i+1$ 位置为空，
- 则 $t+1$ 时刻 i 位置变为空，$i+1$ 位置变为车.

换道规则

- 若 t 时刻 i 位置和 $i+1$ 位置状态都为车，
- 则 $t+1$ 时刻 i 位置的车尝试换道，左右机率相等.

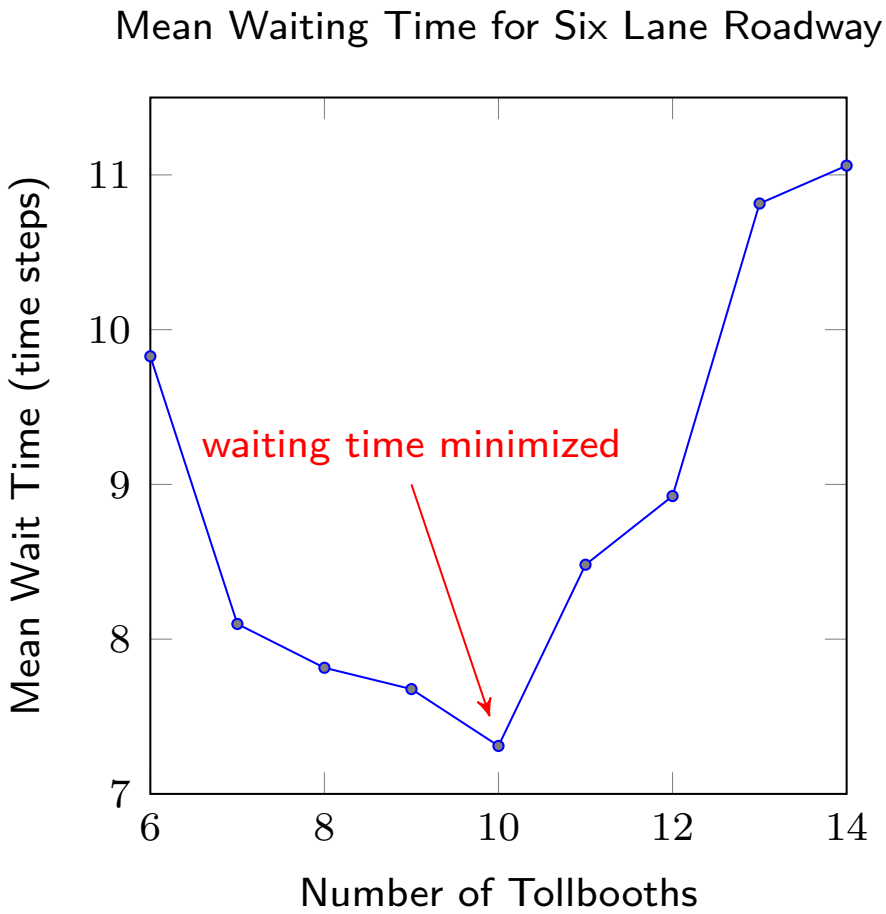
模型建立：花费总时间的计算

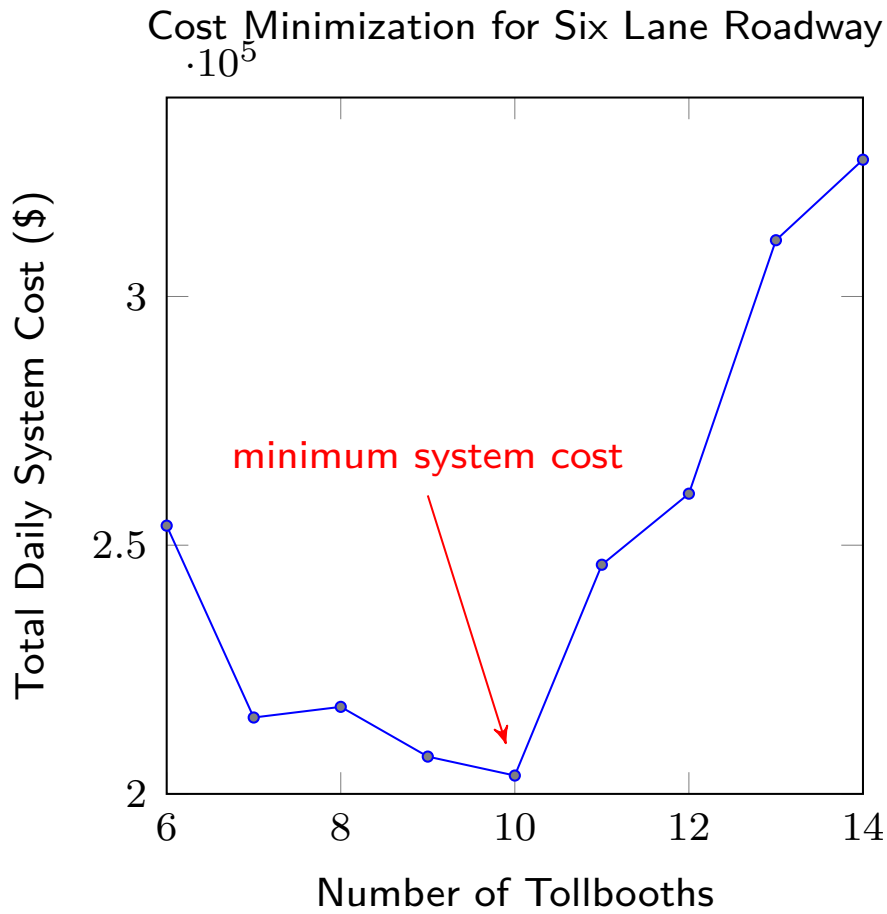
花费总时间的计算

$$\forall x, \forall y$$

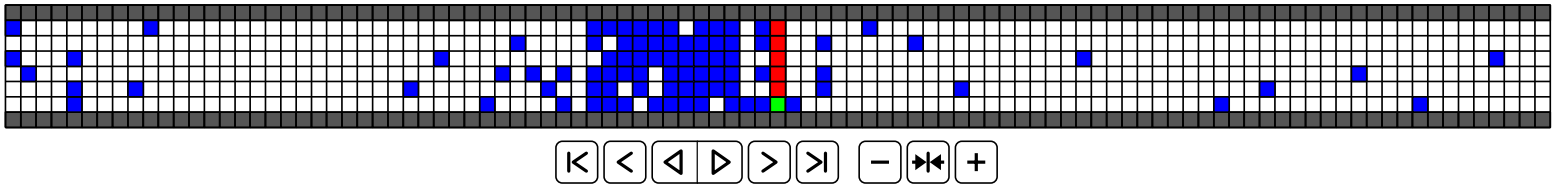
$$W_i = W_{i-1} + 1(\text{plaza}(x, y) > 0)$$

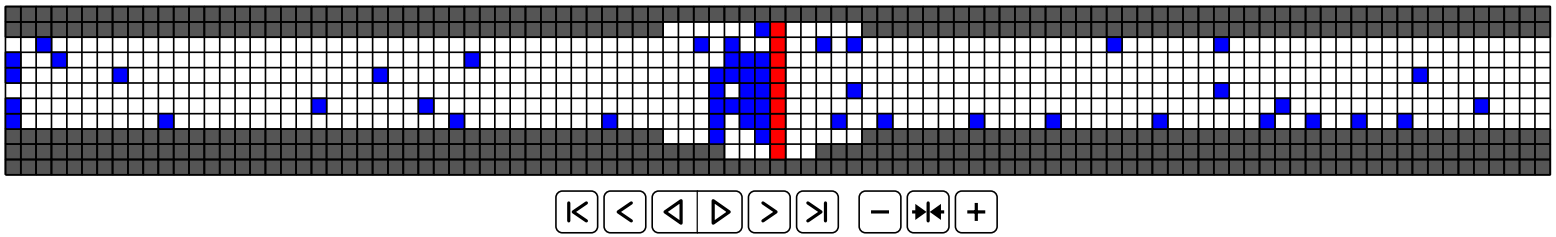
结果求解

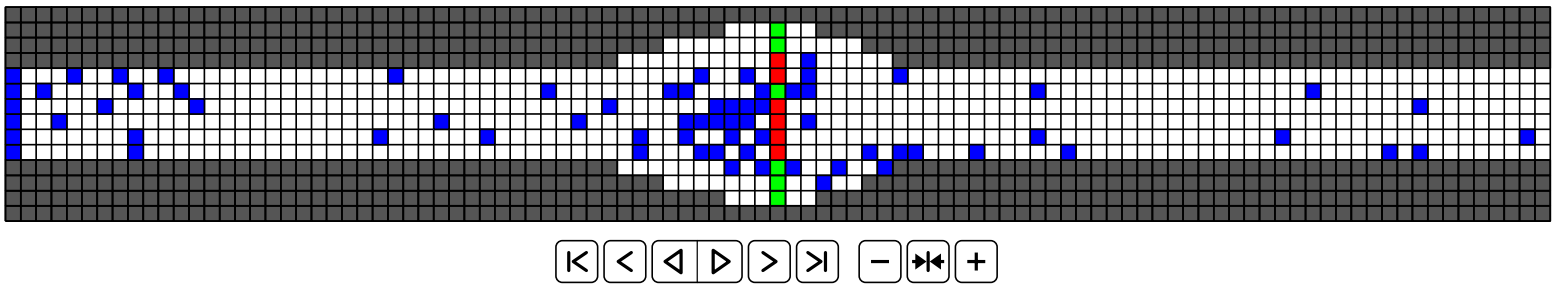

Mean Waiting Time for Six Lane Roadway
Mean Wait Time (time steps)
11
10
9
8
7
6
8
10
12
14
Number of Tollbooths
waiting time minimized


Cost Minimization for Six Lane Roadway
$\cdot 10^5$
Total Daily System Cost ($)
3
2.5
2
6
8
10
12
14
Number of Tollbooths
minimum system cost

结果与分析







结果与分析：假设评价

车道数	一天	高峰
1	2	2
2	4	4
3	5	6
4	7	7
5	8	9
6	10	11
7	12	13
8	14	15
16	27	29

假设的合理性

- 元胞类型：模型可通过增加元胞类型来得到改进，但新的特征可能不会改变系统的基本行为.
- 元胞尺寸：无法明确评估这种假设是否在某些重要方面限制了模型.
- 车辆类型：作为扩展，没有探讨使用多个元胞表示较大车辆.

提要

结论

车道数	模型一	模型二	模型三	推荐值
1	4	3	2	2
2	5	5	4	4
3	7	6	5	5
4	8	7	7	6
5	10	9	8	9
6	12	11	10	10
7	13	12	12	12
8	16	14	14	14
16	29	27	27	27

Part III

程序实现

提要

拟合

Command Window

f_x >>

拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
fx >>
```

拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
fx >>
```

拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

    1.0200    0.0400
```

拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

    1.0200    0.0400

fx >>
```

拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

    1.0200    0.0400

>> xi = 1:0.1:3;
fx >>
```

拟合

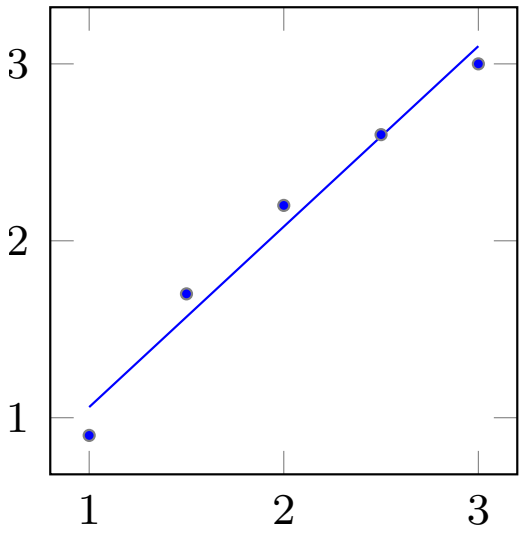
Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

    1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
fx >>
```

拟合

Command Window

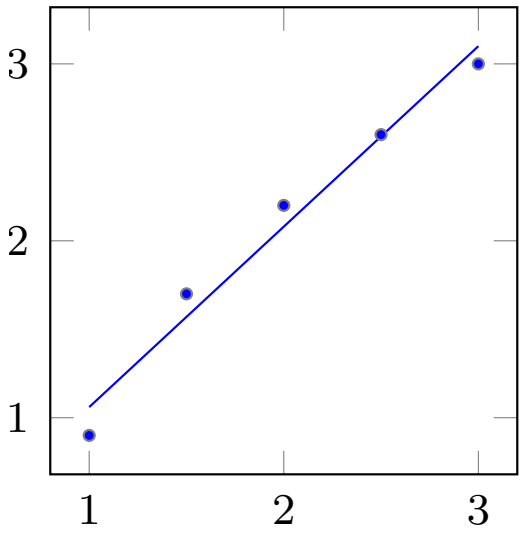
```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
>> plot(x,y,'o',xi,yi);
fx >>
```



拟合

Command Window

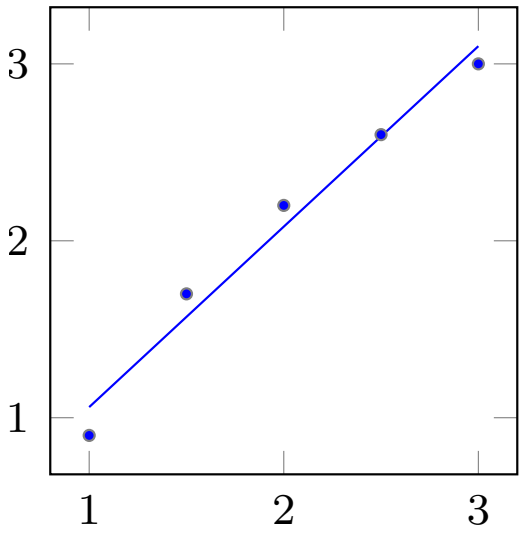
```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
>> plot(x,y,'o',xi,yi);
>> p = fittype('a*x+b*sin(x)+c');
fx >>
```



拟合

Command Window

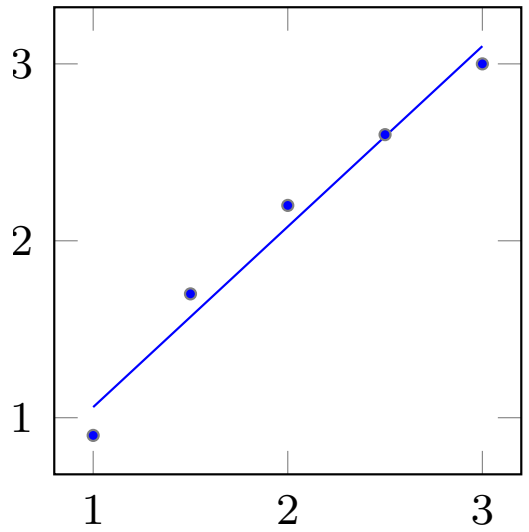
```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
>> plot(x,y,'o',xi,yi);
>> p = fittype('a*x+b*sin(x)+c');
>> f = fit(x,y,p)

f =
   General model:
   f(x) = a*x+b*sin(x)+c
   Coefficients (with 95% confidence bounds):
   a = 1.249 (0.9856, 1.512)
   b = 0.6357 (0.03185, 1.24)
   c = -0.8611 (-1.773, 0.05094)

fx >>
```

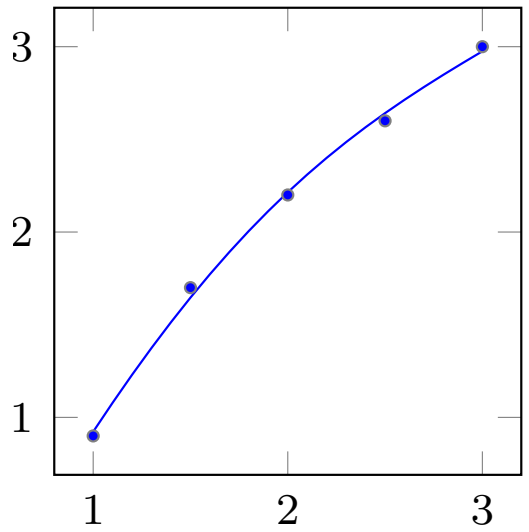


拟合

Command Window

```
>> x = [1.0, 1.5, 2.0, 2.5, 3.0]';
>> y = [0.9, 1.7, 2.2, 2.6, 3.0]';
>> a = polyfit(x,y,1)

a =

      1.0200    0.0400

>> xi = 1:0.1:3;
>> yi = polyval(a,xi);
>> plot(x,y,'o',xi,yi);
>> p = fittype('a*x+b*sin(x)+c');
>> f = fit(x,y,p)

f =
   General model:
   f(x) = a*x+b*sin(x)+c
   Coefficients (with 95% confidence bounds):
   a = 1.249 (0.9856, 1.512)
   b = 0.6357 (0.03185, 1.24)
   c = -0.8611 (-1.773, 0.05094)

>> plot(f,x,y);
```





多元线性回归

定义

$$Y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_k x_k$$

regress

```
[B,Bint,R,Rint,Stats] = regress(Y,X)
```

- B: 回归得到的自变量系数.
- Bint: B 的 95% 的置信区间矩阵

案例

$$F(t) = b_0 + b_1 \cos(\omega t) + b_2 \sin(2\omega t)$$

多元线性回归

定义

$$Y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_k x_k$$

regress

```
[B,Bint,R,Rint,Stats] = regress(Y,X)
```

- B: 回归得到的自变量系数.
- Bint: B 的 95% 的置信区间矩阵

案例

$$F(t) = b_0 + b_1 \cos(\omega t) + b_2 \sin(2\omega t)$$

提要

车流量: 傅里叶级数近似 (拟合)

```
01 T=[0.5:1:23.5]';
02 influx=[15.44 15.32 15.16 19.9  47.09 89.95 ... % 0.5- 5.5
03         105.9 85.52 54.68 43.11 40.16 40.85 ... % 6.5-11.5
04         41.72 44.54 48.88 53.2  51.61 48.38 ... %12.5-17.5
05         39.72 30.51 29.48 26.82 21.21 17.22]';  %18.5-23.5
06 omega =2*pi/24;              % 周期
07 fc=zeros(24,8); fs=zeros(24,8);
08 for n = 1:8
09     fc(:,n)=cos(n*omega*T); fs(:,n)=sin(n*omega*T);
10 end
11 [B,BINT,R]=regress(influx,[ones(24,1),fc,fs],0.05);
12 t=0:0.01:24;
13 a0 = B(1); a = B(2:9); b = B(10:end);
14 inrate = a0
15 for n = 1:8
16     inrate = inrate+a(n)*cos(n*t.*omega)+b(n)*sin(n*t.*omega);
17 end
```

车流量: 傅里叶级数近似 (拟合)

```
01 T=[0.5:1:23.5]';
02 influx=[15.44 15.32 15.16 19.9  47.09 89.95 ... % 0.5- 5.5
03         105.9 85.52 54.68 43.11 40.16 40.85 ... % 6.5-11.5
04         41.72 44.54 48.88 53.2  51.61 48.38 ... %12.5-17.5
05         39.72 30.51 29.48 26.82 21.21 17.22]';  %18.5-23.5
06 omega =2*pi/24;              % 周期
07 fc=zeros(24,8); fs=zeros(24,8);
08 for n = 1:8
09     fc(:,n)=cos(n*omega*T); fs(:,n)=sin(n*omega*T);
10 end
11 [B,BINT,R]=regress(influx,[ones(24,1),fc,fs],0.05);
12 t=0:0.01:24;
13 a0 = B(1); a = B(2:9); b = B(10:end);
14 inrate = a0
15 for n = 1:8
16     inrate = inrate+a(n)*cos(n*t.*omega)+b(n)*sin(n*t.*omega);
17 end
```

车流量: 傅里叶级数近似 (拟合)

```
01 T=[0.5:1:23.5]';
02 influx=[15.44 15.32 15.16 19.9  47.09 89.95 ... % 0.5- 5.5
03         105.9 85.52 54.68 43.11 40.16 40.85 ... % 6.5-11.5
04         41.72 44.54 48.88 53.2  51.61 48.38 ... %12.5-17.5
05         39.72 30.51 29.48 26.82 21.21 17.22]';  %18.5-23.5
06 omega =2*pi/24;              % 周期
07 fc=zeros(24,8); fs=zeros(24,8);
08 for n = 1:8
09     fc(:,n)=cos(n*omega*T); fs(:,n)=sin(n*omega*T);
10 end
11 [B,BINT,R]=regress(influx,[ones(24,1),fc,fs],0.05);
12 t=0:0.01:24;
13 a0 = B(1); a = B(2:9); b = B(10:end);
14 inrate = a0
15 for n = 1:8
16     inrate = inrate+a(n)*cos(n*t.*omega)+b(n)*sin(n*t.*omega);
17 end
```

车流量: 傅里叶级数近似 (拟合)

```
01 T=[0.5:1:23.5]';
02 influx=[15.44 15.32 15.16 19.9  47.09 89.95 ... % 0.5- 5.5
03         105.9 85.52 54.68 43.11 40.16 40.85 ... % 6.5-11.5
04         41.72 44.54 48.88 53.2  51.61 48.38 ... %12.5-17.5
05         39.72 30.51 29.48 26.82 21.21 17.22]';  %18.5-23.5
06 omega =2*pi/24;              % 周期
07 fc=zeros(24,8); fs=zeros(24,8);
08 for n = 1:8
09     fc(:,n)=cos(n*omega*T); fs(:,n)=sin(n*omega*T);
10 end
11 [B,BINT,R]=regress(influx,[ones(24,1),fc,fs],0.05);
12 t=0:0.01:24;
13 a0 = B(1); a = B(2:9); b = B(10:end);
14 inrate = a0
15 for n = 1:8
16     inrate = inrate+a(n)*cos(n*t.*omega)+b(n)*sin(n*t.*omega);
17 end
```

车流量: 傅里叶级数近似 (拟合)

```
01 T=[0.5:1:23.5]';
02 influx=[15.44 15.32 15.16 19.9  47.09 89.95 ... % 0.5- 5.5
03         105.9 85.52 54.68 43.11 40.16 40.85 ... % 6.5-11.5
04         41.72 44.54 48.88 53.2  51.61 48.38 ... %12.5-17.5
05         39.72 30.51 29.48 26.82 21.21 17.22]';  %18.5-23.5
06 omega =2*pi/24;              % 周期
07 fc=zeros(24,8); fs=zeros(24,8);
08 for n = 1:8
09     fc(:,n)=cos(n*omega*T); fs(:,n)=sin(n*omega*T);
10 end
11 [B,BINT,R]=regress(influx,[ones(24,1),fc,fs],0.05);
12 t=0:0.01:24;
13 a0 = B(1); a = B(2:9); b = B(10:end);
14 inrate = a0
15 for n = 1:8
16     inrate = inrate+a(n)*cos(n*t.*omega)+b(n)*sin(n*t.*omega);
17 end
```

模型一: 并联服务台排队论模拟

Basic Car-Tracking Model Code

```
for j = 1:i                    % i: 车的数量
    for k = 1:booths           % 找出最快将空闲的收费亭
        if (last(k) == min(last))
            B(j) = k;
        end
    end
    if A(j) > last(B(j)) % 如果有空闲的收费亭, 则
        Start(j) = A(j);           % 开始时刻=到达时刻
        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
        last(B(j)) = L(j);         % 空闲时刻 = 离开时刻
    else                       % 如果没有空闲收费亭, 则
        Start(j) = last(B(j));  % 开始时刻=空闲时刻
        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
        last(B(j)) = L(j);         % 空闲时刻 = 离开时刻
    end
end
```

模型一: 并联服务台排队论模拟

Basic Car-Tracking Model Code

```
01 for j = 1:i                    % i: 车的数量
02     for k = 1:booths           % 找出最快将空闲的收费亭
03         if (last(k) == min(last))
04            B(j) = k;
05         end
06     end
07     if A(j) > last(B(j)) % 如果有空闲的收费亭, 则
08        Start(j) = A(j);              % 开始时刻=到达时刻
09        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
10        last(B(j)) = L(j);            % 空闲时刻 = 离开时刻
11     else                          % 如果没有空闲收费亭, 则
12        Start(j) = last(B(j));  % 开始时刻=空闲时刻
13        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
14        last(B(j)) = L(j);            % 空闲时刻 = 离开时刻
15     end
16 end
```

模型一: 并联服务台排队论模拟

Basic Car-Tracking Model Code

```
01 for j = 1:i                    % i: 车的数量
02     for k = 1:booths           % 找出最快将空闲的收费亭
03         if (last(k) == min(last))
04            B(j) = k;
05         end
06     end
07     if A(j) > last(B(j)) % 如果有空闲的收费亭, 则
08        Start(j) = A(j);              % 开始时刻=到达时刻
09        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
10        last(B(j)) = L(j);            % 空闲时刻 = 离开时刻
11     else                          % 如果没有空闲收费亭, 则
12        Start(j) = last(B(j));  % 开始时刻=空闲时刻
13        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
14        last(B(j)) = L(j);            % 空闲时刻 = 离开时刻
15     end
16 end
```

模型一: 并联服务台排队论模拟

Basic Car-Tracking Model Code

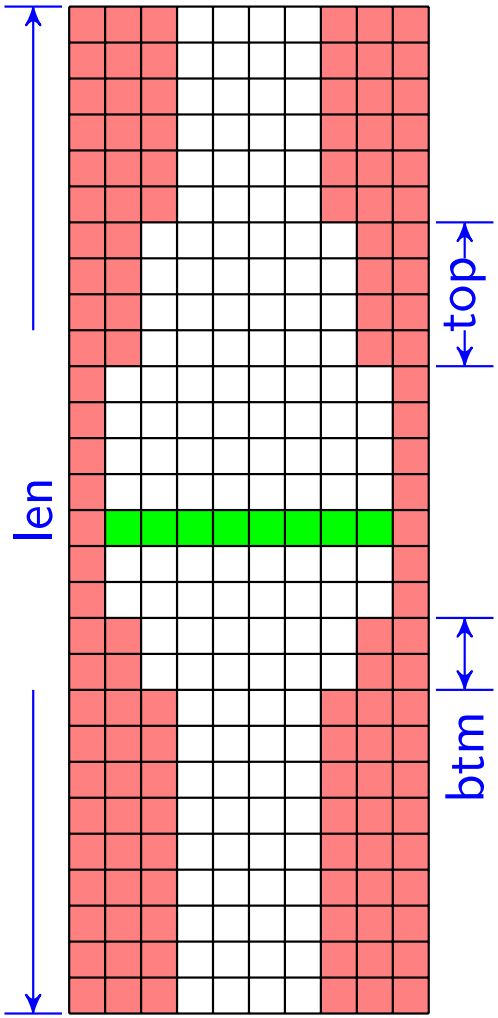
```
for j = 1:i                    % i: 车的数量
    for k = 1:booths           % 找出最快将空闲的收费亭
        if (last(k) == min(last))
            B(j) = k;
        end
    end
    if A(j) > last(B(j)) % 如果有空闲的收费亭, 则
        Start(j) = A(j);           % 开始时刻=到达时刻
        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
        last(B(j)) = L(j);         % 空闲时刻 = 离开时刻
    else                       % 如果没有空闲收费亭, 则
        Start(j) = last(B(j));  % 开始时刻=空闲时刻
        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
        last(B(j)) = L(j);         % 空闲时刻 = 离开时刻
    end
end
```

模型一: 并联服务台排队论模拟

Basic Car-Tracking Model Code

```
01 for j = 1:i                    % i: 车的数量
02     for k = 1:booths           % 找出最快将空闲的收费亭
03         if (last(k) == min(last))
04            B(j) = k;
05         end
06     end
07     if A(j) > last(B(j)) % 如果有空闲的收费亭, 则
08        Start(j) = A(j);              % 开始时刻=到达时刻
09        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
10        last(B(j)) = L(j);            % 空闲时刻 = 离开时刻
11     else                           % 如果没有空闲收费亭, 则
12        Start(j) = last(B(j));  % 开始时刻=空闲时刻
13        L(j) = Start(j) + S(j); % 离开时刻=开始时刻+服务时长
14        last(B(j)) = L(j);            % 空闲时刻 = 离开时刻
15     end
16 end
```

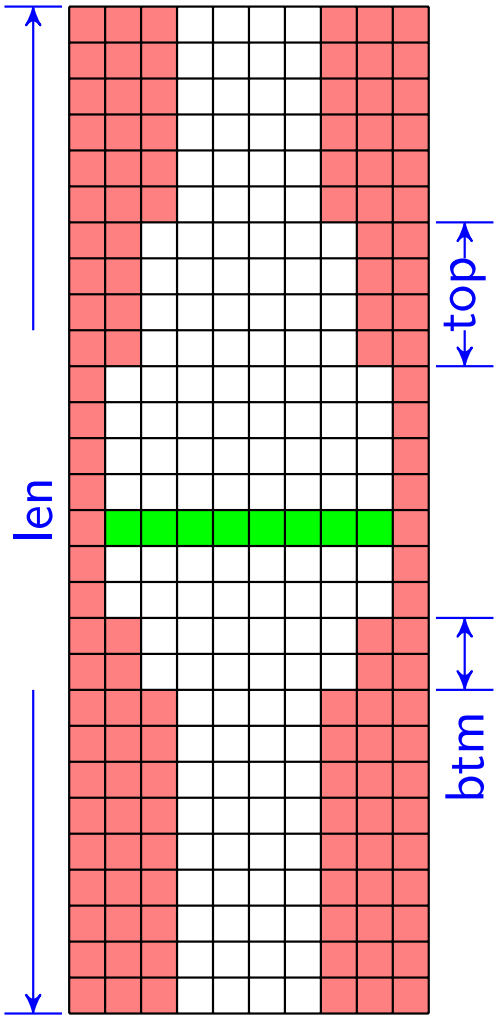
模型三: 收费站平面图的矩阵化



create_plaza.m

```
01 B = 8;                          % 收费亭数量
02 L = 4;                          % 车道数量
03 len=27;                         % 广场长度
04 top = 4;                        % 上斜坡长
05 btm = 2;                        % 下斜坡长
06 plaza = zeros(len,B+2);
07 plaza(1:len,[1,2+B]) = -88;
08 for c = 2:B/2 - L/2 + 1
09     for r = 1:(len-1)/2-top*(c-1)
10         plaza(r,[c, B+3-c]) = -88;
11     end
12     for r = (len+3)/2+btm*(c-1):len
13         plaza(r,[c, B+3-c]) = -88;
14     end
15 end
```

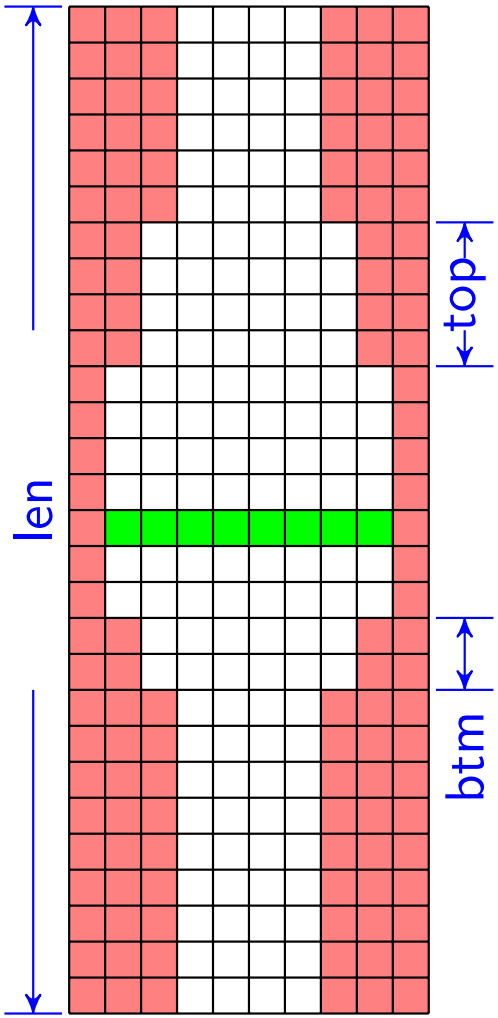
模型三: 收费站平面图的矩阵化



create_plaza.m

```
01 B = 8;                          % 收费亭数量
02 L = 4;                          % 车道数量
03 len=27;                         % 广场长度
04 top = 4;                        % 上斜坡长
05 btm = 2;                        % 下斜坡长
06 plaza = zeros(len,B+2);
07 plaza(1:len,[1,2+B]) = -88;
08 for c = 2:B/2 - L/2 + 1
09     for r = 1:(len-1)/2-top*(c-1)
10         plaza(r,[c, B+3-c]) = -88;
11     end
12     for r = (len+3)/2+btm*(c-1):len
13         plaza(r,[c, B+3-c]) = -88;
14     end
15 end
```

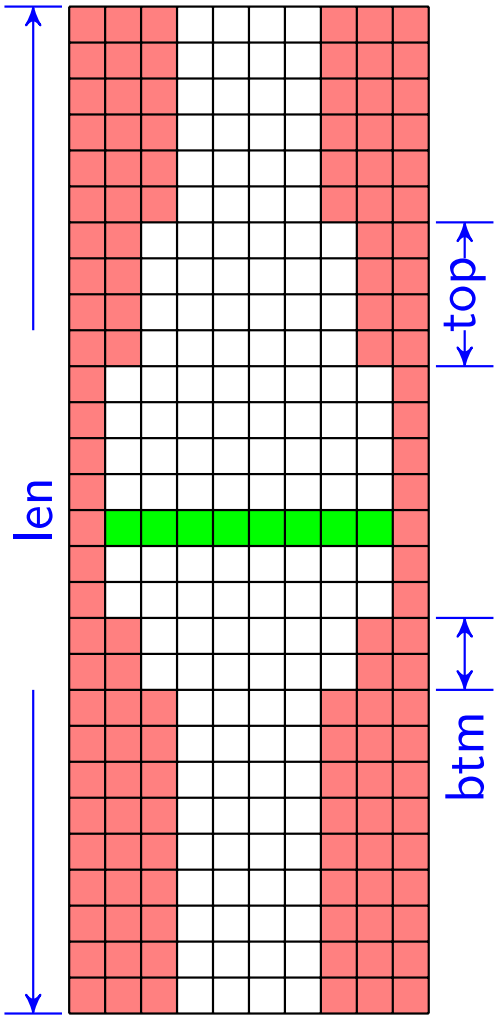
模型三: 收费站平面图的矩阵化



create_plaza.m

```
01 B = 8;                          % 收费亭数量
02 L = 4;                          % 车道数量
03 len=27;                         % 广场长度
04 top = 4;                        % 上斜坡长
05 btm = 2;                        % 下斜坡长
06 plaza = zeros(len,B+2);
07 plaza(1:len,[1,2+B]) = -88;
08 for c = 2:B/2 - L/2 + 1
09     for r = 1:(len-1)/2-top*(c-1)
10         plaza(r,[c, B+3-c]) = -88;
11     end
12     for r = (len+3)/2+btm*(c-1):len
13         plaza(r,[c, B+3-c]) = -88;
14     end
15 end
```

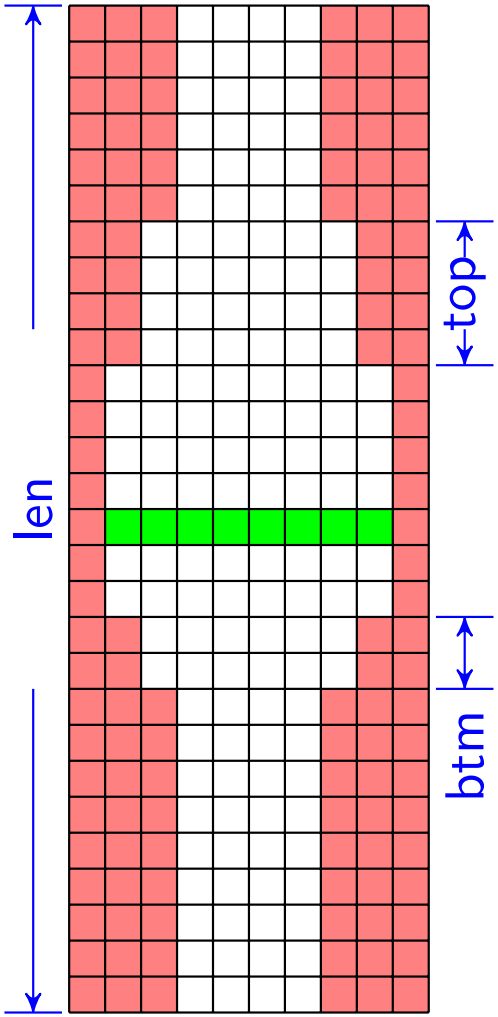
模型三: 收费站平面图的矩阵化



create_plaza.m

```
01 B = 8;                          % 收费亭数量
02 L = 4;                          % 车道数量
03 len=27;                         % 广场长度
04 top = 4;                        % 上斜坡长
05 btm = 2;                        % 下斜坡长
06 plaza = zeros(len,B+2);
07 plaza(1:len,[1,2+B]) = -88;
08 for c = 2:B/2 - L/2 + 1
09     for r = 1:(len-1)/2-top*(c-1)
10         plaza(r,[c, B+3-c]) = -88;
11     end
12     for r = (len+3)/2+btm*(c-1):len
13         plaza(r,[c, B+3-c]) = -88;
14     end
15 end
```

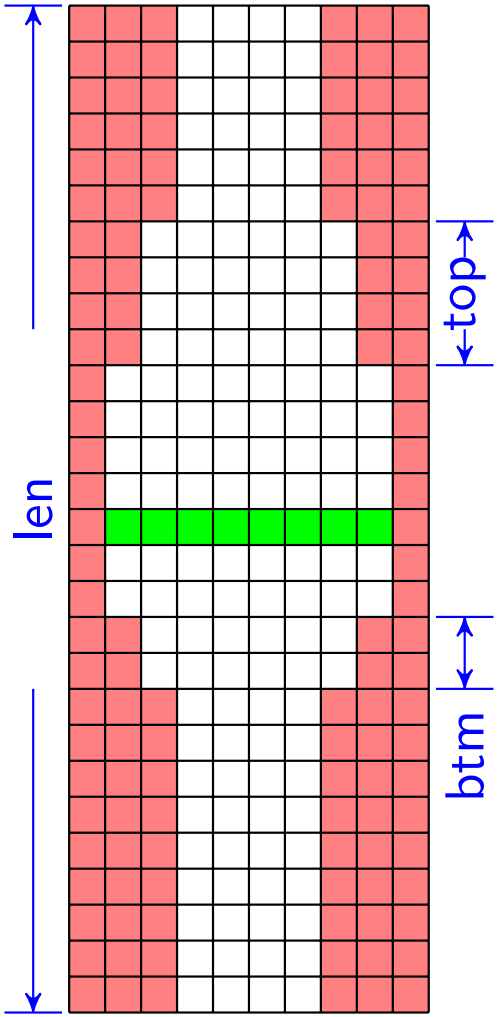
模型三: 收费站平面图的矩阵化



create_plaza.m

```
01 B = 8;                          % 收费亭数量
02 L = 4;                          % 车道数量
03 len=27;                         % 广场长度
04 top = 4;                        % 上斜坡长
05 btm = 2;                        % 下斜坡长
06 plaza = zeros(len,B+2);
07 plaza(1:len,[1,2+B]) = -88;
08 for c = 2:B/2 - L/2 + 1
09     for r = 1:(len-1)/2-top*(c-1)
10         plaza(r,[c, B+3-c]) = -88;
11     end
12     for r = (len+3)/2+btm*(c-1):len
13         plaza(r,[c, B+3-c]) = -88;
14     end
15 end
```

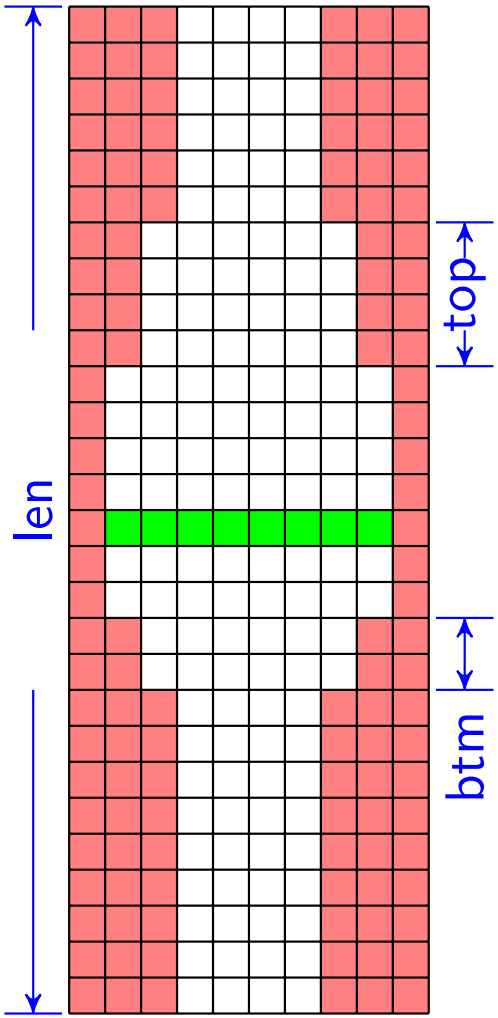
模型三: 收费站平面图的矩阵化



create_plaza.m

```
01 B = 8;                          % 收费亭数量
02 L = 4;                          % 车道数量
03 len=27;                         % 广场长度
04 top = 4;                        % 上斜坡长
05 btm = 2;                        % 下斜坡长
06 plaza = zeros(len,B+2);
07 plaza(1:len,[1,2+B]) = -88;
08 for c = 2:B/2 - L/2 + 1
09     for r = 1:(len-1)/2-top*(c-1)
10         plaza(r,[c, B+3-c]) = -88;
11     end
12     for r = (len+3)/2+btm*(c-1):len
13         plaza(r,[c, B+3-c]) = -88;
14     end
15 end
```

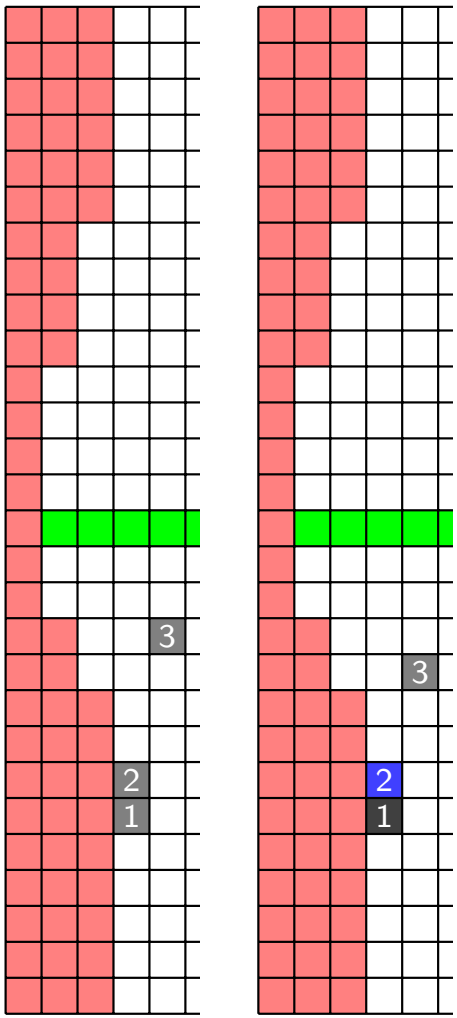
模型三: 收费站平面图的矩阵化



create_plaza.m

```
01 B = 8;                          % 收费亭数量
02 L = 4;                          % 车道数量
03 len=27;                         % 广场长度
04 top = 4;                        % 上斜坡长
05 btm = 2;                        % 下斜坡长
06 plaza = zeros(len,B+2);
07 plaza(1:len,[1,2+B]) = -88;
08 for c = 2:B/2 - L/2 + 1
09     for r = 1:(len-1)/2-top*(c-1)
10         plaza(r,[c, B+3-c]) = -88;
11     end
12     for r = (len+3)/2+btm*(c-1):len
13         plaza(r,[c, B+3-c]) = -88;
14     end
15 end
```

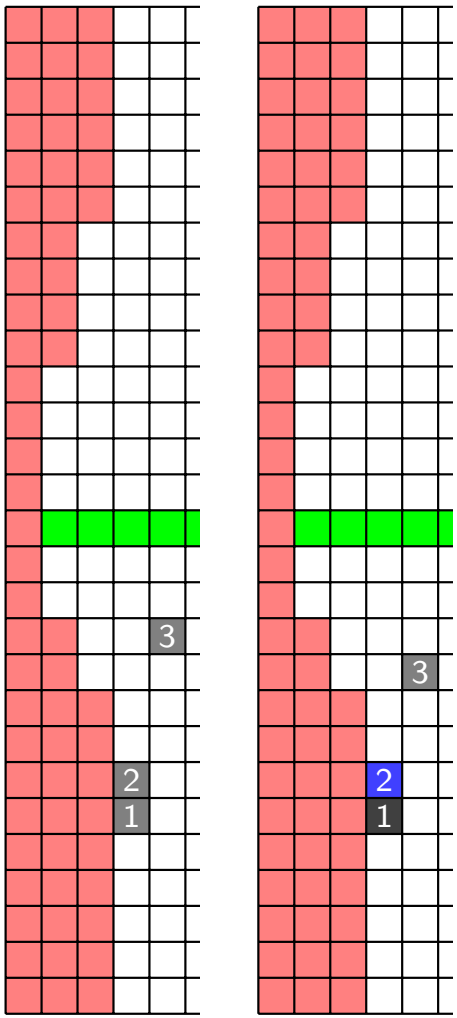
模型三: 前进规则



move_forward.m

```
01 [L, W] = size(plaza);
02 prob = 0.7;
03 for i = (L-1):-1:((L + 1)/2 + 1)
04     for j = 1:W
05         if plaza(i,j) == 1
06             if plaza(i+1,j) ~= 0
07                 % 若不能前进标记为-2
08                 plaza(i,j) = -2;
09             elseif prob >= rand
10                 plaza(i,j) = 0;
11                 plaza(i+1,j) = 1;
12             end
13         end
14     end
15 end
```

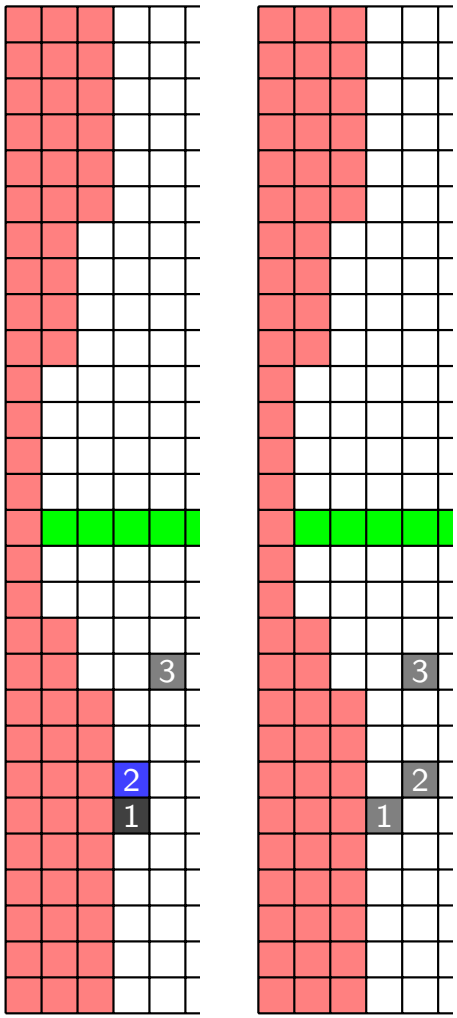
模型三: 前进规则



move_forward.m

```
01 [L, W] = size(plaza);
02 prob = 0.7;
03 for i = (L-1):-1:((L + 1)/2 + 1)
04     for j = 1:W
05         if plaza(i,j) == 1
06             if plaza(i+1,j) ~= 0
07                 % 若不能前进标记为-2
08                 plaza(i,j) = -2;
09             elseif prob >= rand
10                 plaza(i,j) = 0;
11                 plaza(i+1,j) = 1;
12             end
13         end
14     end
15 end
```

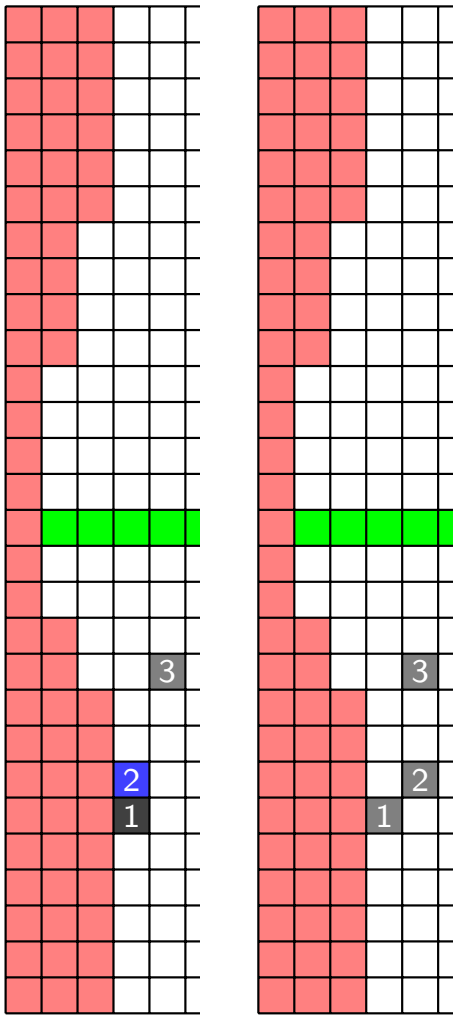
模型三: 换道规则



switch_lanes.m

```
01 if plaza(i,j) == -2
02     if rand > 0.5  % 先尝试向左
03         if plaza(i,j-1) == 0
04             plaza(i,j-1) = 1;
05             plaza(i,j) = 0;
06         elseif plaza(i,j+1) == 0
07             plaza(i,j+1) = 1;
08             plaza(i,j) = 0;
09         else
10             plaza(i,j) = 1;
11         end
12     else              % 先尝试向右
13         ......
14     end
15 end
```

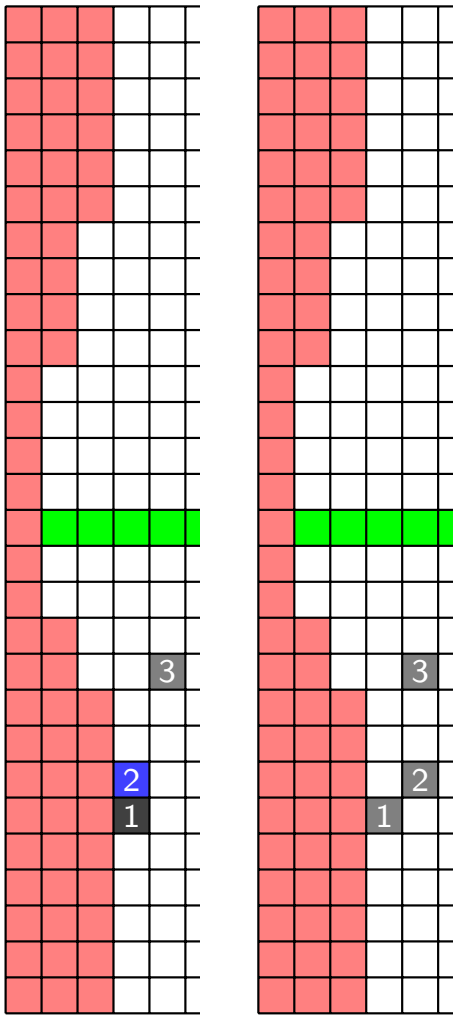
模型三: 换道规则



switch_lanes.m

```
01 if plaza(i,j) == -2
02     if rand > 0.5  % 先尝试向左
03         if plaza(i,j-1) == 0
04             plaza(i,j-1) = 1;
05             plaza(i,j) = 0;
06         elseif plaza(i,j+1) == 0
07             plaza(i,j+1) = 1;
08             plaza(i,j) = 0;
09         else
10             plaza(i,j) = 1;
11         end
12     else              % 先尝试向右
13         ......
14     end
15 end
```

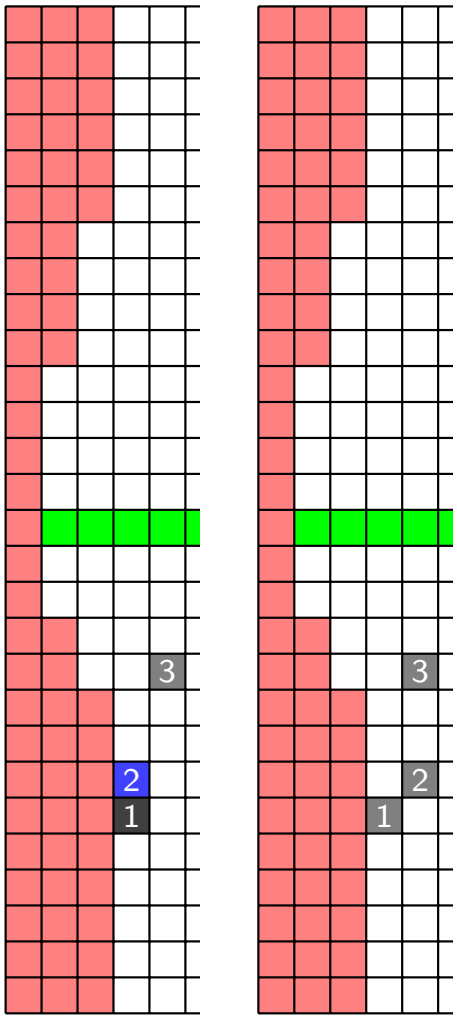
模型三: 换道规则



switch_lanes.m

```
01 if plaza(i,j) == -2
02     if rand > 0.5  % 先尝试向左
03         if plaza(i,j-1) == 0
04             plaza(i,j-1) = 1;
05             plaza(i,j) = 0;
06         elseif plaza(i,j+1) == 0
07             plaza(i,j+1) = 1;
08             plaza(i,j) = 0;
09         else
10             plaza(i,j) = 1;
11         end
12     else              % 先尝试向右
13         ......
14     end
15 end
```

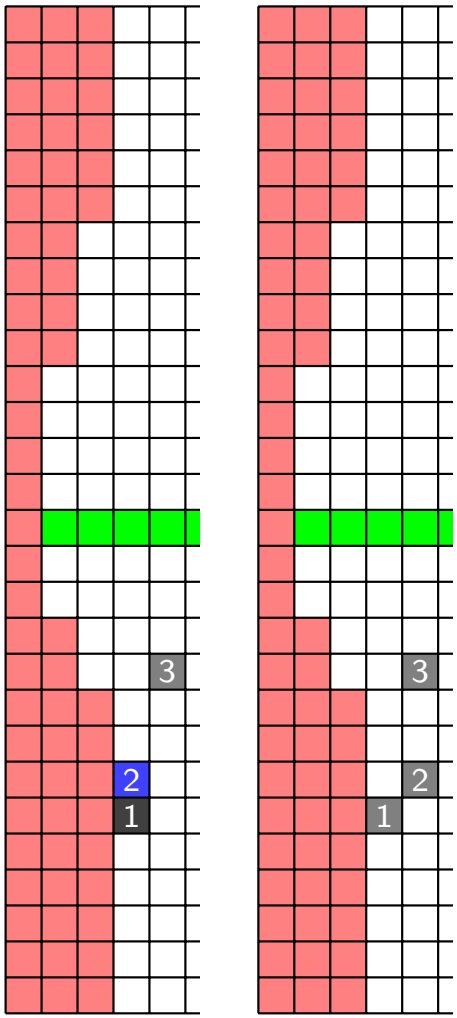
模型三: 换道规则



switch_lanes.m

```
01 if plaza(i,j) == -2
02     if rand > 0.5  % 先尝试向左
03         if plaza(i,j-1) == 0
04             plaza(i,j-1) = 1;
05             plaza(i,j) = 0;
06         elseif plaza(i,j+1) == 0
07             plaza(i,j+1) = 1;
08             plaza(i,j) = 0;
09         else
10             plaza(i,j) = 1;
11         end
12     else              % 先尝试向右
13         ......
14     end
15 end
```

模型三: 换道规则



switch_lanes.m

```
01 if plaza(i,j) == -2
02     if rand > 0.5  % 先尝试向左
03         if plaza(i,j-1) == 0
04             plaza(i,j-1) = 1;
05             plaza(i,j) = 0;
06         elseif plaza(i,j+1) == 0
07             plaza(i,j+1) = 1;
08             plaza(i,j) = 0;
09         else
10             plaza(i,j) = 1;
11         end
12     else              % 先尝试向右
13         ......
14     end
15 end
```

Part IV

写作分析和总结

提要

8 论文评价
- 优点
- 缺点
- 改进

9 写作分析
- 文章结构
- 行文特点
- 摘要写作

10 要点总结
- 知识要点
- 参考作业

优点

模型

- 成本考虑全面: 顾客时间成本 + 收费站运营成本
- 模型丰富: 微观 + 宏观

写作

- 行文幽默
- 模型关联

缺点

模型三

- 收费广场结构失真: 对于收费亭-车道数 = 奇数的情况, 生成的收费站矩阵很不对称.
- 车辆前进规则单一: 所有车的速度都一样, 要么 1, 要么 0. 不能很好的模拟车的减速, 加速等行为.
- 更换车道具有方向性及盲目性:
  - 由于循环具有方向, 因此使得改换车道具有了方向性.
  - 若车辆更换车道后情况不比之前好, 那么它就没必要更换.
- 元胞自动机模型对花费时间的统计太过宏观.

改进

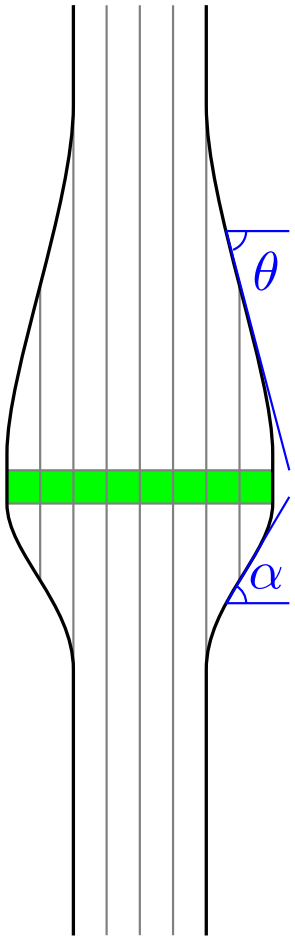
模型一: 考虑瓶颈

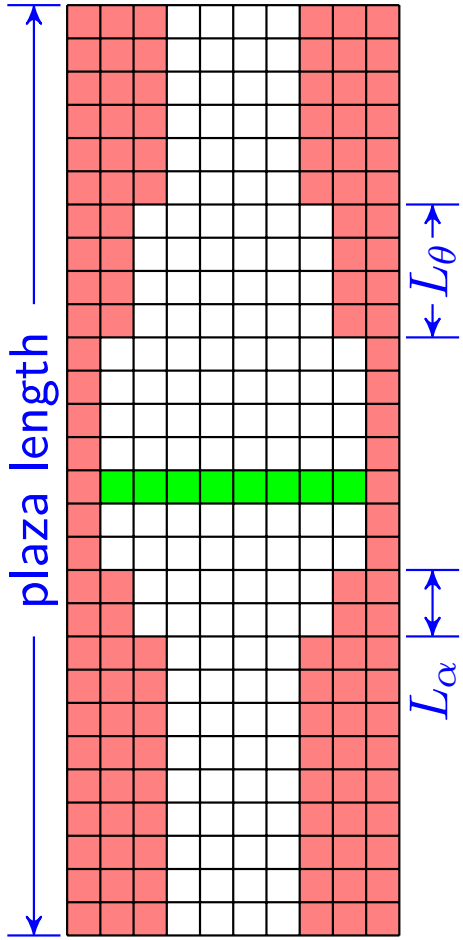
- 将瓶颈处也想像成一个服务台, 即 B 个收费亭服务台并联后再与瓶颈服务台串联.
- 顾客离开收费亭的时刻作为到达瓶颈服务台的时刻.

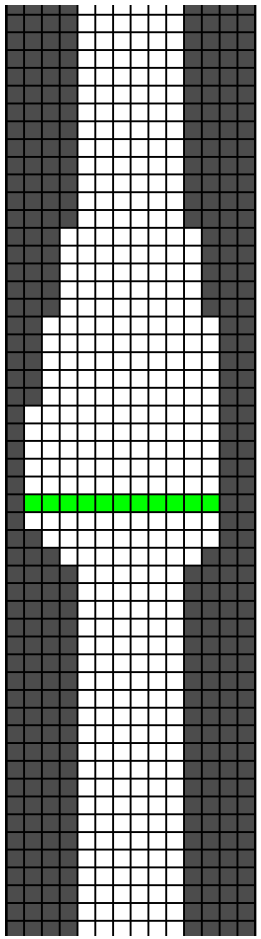
模型三

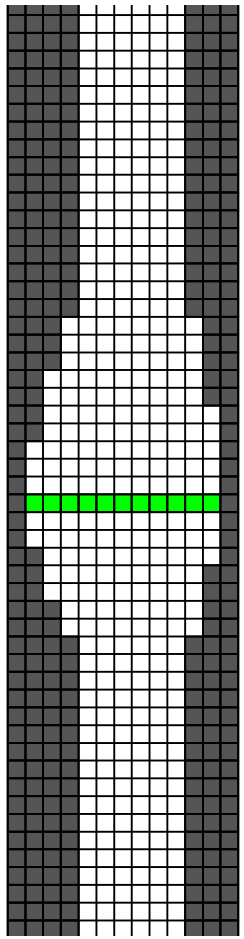
- 车辆前进规则: 改用 NS 规则.
- 车辆换道规则: 将所有需要换道的车辆找出, 随机排序, 按照新顺序逐个换道.

改进: 收费广场结构









$$L_\theta = [\tan\theta],\ L_\alpha = [\tan\alpha]$$

提要

8 论文评价
- 优点
- 缺点
- 改进

9 写作分析
- 文章结构
- 行文特点
- 摘要写作

10 要点总结
- 知识要点
- 参考作业

文章结构

原文的结构是并列式，即用不同的法方解决同一个问题，分别得到结果

- 微观. 排队模型. 仿真. 上界
- 宏观. 积分模型. 解析
- 微观. 元胞自动机. 仿真. 推荐.

注意：这种模式的论文不能写成罗列式，要比较结果和联系，并说明结果差异的原因.

后前照应

文中的前后照应，是一种比较好的写作手法，这种手法通常可以采用“后前照应”

- 文章开头先给出一个好的模型的标准，后面模型算出来后发现都符合标准. 作者先得到结果，后写的论文.
- 对于前一个模型的结果，作者强调其是上界，也就是后来的模型结果应比这个第一个模型的结果小. 后面发现确实如此，显得作者前面分析的很有道理. 作者先得到全部结果，后写的论文及分析.

这提示我们，论文一定要在全部模型算完后，再全局性的改改.

摘要写作

格式一

- 对要解决的整个问题作简单描述，以及问题的意义.
- 问题是如何解决的，从什么角度，考虑了哪些因素...
- 问题的结论，并简单评价结论.

格式二

- 对要解决的整个问题作简单描述，以及问题的意义.
- 问题一是如何解决的，结果如何，简单评价问题一的模型.
- 问题二是如何解决的，结果如何，简单评价问题二的模型.
- 问题三是如何解决的，结果如何，简单评价问题三的模型.
- ⋮

摘要范例

In this paper, we address the problems associated with heavy demands on toll plazas such as lines, backups, and traffic jams. We consider several models in hopes of minimizing the "cost to the system," which includes the time-value of time wasted by drivers as well as the cost of daily operations of the toll plaza.

One model yields a microscopic simulation of line formation in front of the toll booths when the service rate cannot match the demand. Using hourly demand data from a major New Jersey parkway, the simulation is limited in not taking bottlenecking effects into consideration. The results, however, when subjected to threshold analysis can serve to set upper bounds on the number of booths that could potentially be suggested by any other models.

After presenting this basic model, a more general, macroscopic framework for analyzing toll plaza design is introduced. In analyzing "total cost" and allowing bottlenecking, this model is more complete than the first, and it is able to make recommendations for booth number based on data obtained from the first model. This computation melds the macro- and micro-levels, a strategy that is helpful in looking at toll booth situations.

Finally, a model for traffic flow through a plaza is formulated in the world of "cellular automata." An interesting take on microscopic ideas, the cellular automata model can serve as an independent validation of our other models.

In fact, the models mostly agree that given L lanes, a number of booths around $B = [[1.65L + 0.9]]$, where $[[x]]$ is the greatest integer less than x, will minimize the total human cost associated with the plaza.

提要

8 论文评价
- 优点
- 缺点
- 改进

9 写作分析
- 文章结构
- 行文特点
- 摘要写作

10 要点总结
- 知识要点
- 参考作业

知识要点

模型

- 掌握并列服务台的排队论模型.
- 掌握元胞自动机交通模型.

写作

- 掌握并列式摘要写作方式.
- 学习本文对结果的分析讨论.

程序

- 能够实现多项式曲线拟合, 了解复杂拟合的实现.
- 掌握排队论中, 并列服务台的 MatLab 模拟.
- 了解论文中的元胞自动机交通流模拟程序.

参考作业

- 查阅美赛 09 年交通环岛和 14 年右行规则的交通问题, 找问题的共性和差异.
- 研究周老师改进后的收费站的程序.
- 尝试用本文中学到的模型解决美赛 09 年交通环岛或 14 年右行规则的交通问题.

Thank You!!!