使用马尔科夫过程建模美国犹他州 Lehi 地区的交通流

姓　　名：黄　博

学　　号：521030910068

院　　系：电子信息与电气工程学院

学 科/专 业：计算机科学与技术 (IEEE 试点班)

2024 年 5 月 31 日

摘 要

本文旨在通过马尔科夫过程模型对美国 Utah 州 Lehi 地区及 Highland 地区的交通流进行建模与预测，从而帮助市政部门更有效地管理交通。研究区域被划分为八个部分，包括四个城区（东北、西北、西南、东南）和四个城外区域（对应各城区以外的部分）。利用 2023 年全年以 1 小时为间隔，通过位于不同区域边界的连续计数站（Continous Counting Stations）统计的车辆数量数据，我们对区域间的交通流动进行了分析。

我们尝试了时齐 / 非时齐的离散 / 连续马尔科夫过程模型。对于离散马尔科夫过程建模, 我们计算了离散时间下的马尔科夫过程的 k 步转移概率矩阵; 验证了查普曼-柯尔莫哥洛夫方程; 在时齐离散马尔科夫建模下, 我们计算了极限分布与平稳分布; 我们尝试了使用线性模型拟合转移概率矩阵, 从而设计了简单的非时齐离散马尔科夫过程模型; 在时齐 / 非时齐的假设下, 我们都对交通流量进行了预测。对于连续马尔科夫过程建模, 我们根据数据使用线性模型拟合 $\frac{p_{i,j}(t)-\delta_{i,j}}{t}$ 计算得到 Q 矩阵; 在时齐假设下, 通过 Q 矩阵计算了极限分布与平稳分布; 在非时齐假设下, 我们尝试了使用线性模型拟合 Q 矩阵, 从而设计了简单的非时齐连续马尔科夫过程模型; 在时齐 / 非时齐的假设下, 我们都对交通流量进行了预测。

模型结果显示非时齐连续马尔科夫过程在预测交通流方面具有较好的效果。该模型可以为市政部门提供精准的交通流预测，帮助其在高峰时段优化交通信号灯设置、合理规划交通疏导路线、提升道路利用效率以及制定应急预案等方面提供科学依据。

本次研究中通过对 Lehi 地区交通流的成功建模, 充分展示了马尔科夫过程在交通管理中的巨大潜力. 由于数据管制，本研究未能获得更细粒度（如某个街区）的数据。然而，如果市政部门采用本研究方法，并基于更细粒度的数据在街道层次进行建模，便能实现更精确的交通预测和更加有效的交通管控，极大地提高城市交通管理的效果。

关键词： 马尔科夫过程，交通流，非时齐马尔科夫过程，转移速率矩阵，线性模型

目 录

第一章 简介

1.1 马尔科夫过程

马尔科夫过程是由俄罗斯数学家安德烈 · 马尔可夫（Andrey Markov）提出的一种随机过程，其满足马尔科夫性质。这一性质表明，给定当前状态，未来状态与过去状态条件独立。形式上，如果 X_t 表示时间 t 的状态，那么对于任何时间序列 $t_0 < t_1 < \cdots < t_n < t$,

$$P(X_t = x_t | X_{t_0} = x_{t_0}, X_{t_1} = x_{t_1}, \ldots, X_{t_n} = x_{t_n}) = P(X_t = x_t | X_{t_n} = x_{t_n}).$$

这种无记忆性质简化了对各种复杂系统的分析和建模。根据时间参数的不同，马尔科夫过程可以分为离散时间马尔科夫过程和连续时间马尔科夫过程。

1.1.1 离散时间马尔科夫过程

设 $X_{\mathbb{T}} := \{X(n), n \in \mathbb{N}\}$ 为随机序列, 状态空间为 $\mathbb{S} \subset \mathbb{Z}$, 如果对于任意非负整数 k 和 $n_1 < n_2 < \cdots < n_l < m$, 以及 $i_{n1}, i_{n2}, \ldots, i_{nl}, i_m, i_{m+k} \in \mathbb{S}$, 有

$$\begin{aligned} &P(X(m+k) = i_{m+k} | X(n_1) = i_1, \ldots, X(n_l) = i_l, X(m) = i_m) \\ &= P(X(m+k) = i_{m+k} | X(m) = i_m) \end{aligned}$$

成立, 就称 $X_{\mathbb{T}}$ 为离散时间马尔科夫过程 [1]

1.1.2 连续时间马尔科夫过程

设 $X_{\mathbb{T}} := \{X(t), t \geq 0\}$ 为随机序列, 状态空间为 $\mathbb{S} \subset \mathbb{Z}$, 若对任意时刻 $0 < t_1 < \cdots < t_n < t_{n+1}$ 与 $i_{t1}, \ldots, i_{t_n}, i_{t_{n+1}} \in \mathbb{S}$, 有 $P(X(t_1) = i_1, \ldots, X(t_n) = i_n) > 0$, 有

$$\begin{aligned} &P(X(t_{n+1}) = i_{n+1} | X(t_1) = i_1, \ldots, X(t_n) = i_n) \\ &= P(X(t_{n+1}) = i_{n+1} | X(t_n) = i_n) \end{aligned}$$

成立, 就称 $X_{\mathbb{T}}$ 为连续时间马尔科夫过程 [1]

1.1.3 时齐和非时齐马尔科夫过程

根据转移概率是否随时间变化，马尔科夫过程可以进一步分为齐次和非时齐马尔科夫过程。

1.1.3.1　时齐马尔科夫过程

如果一个马尔科夫过程的转移概率与时间无关，则称为时齐马尔科夫过程。对于离散时间马尔科夫链，齐次性表示 $P(X_{n+1} = j|X_n = i)$ 是不随 n 变化的常数。对于连续时间马尔科夫过程，齐次性表示 $P(X(t+s) = j|X(t) = i)$ 不随 t 变化。

1.1.3.2　非时齐马尔科夫过程

如果一个马尔科夫过程的转移概率随时间变化，则称为非时齐马尔科夫过程。对于离散时间马尔科夫链，非齐次性表示 $P(X_{n+1} = j|X_n = i)$ 是 n 的函数。对于连续时间马尔科夫过程，非齐次性表示 $P(X(t+s) = j|X(t) = i)$ 是 t 的函数。

1.2　交通流

交通流理论研究车辆、驾驶员和基础设施之间的相互作用，旨在理解和优化交通流动。交通流通常使用各种数学和计算技术进行建模，以预测和管理车辆运动，减少拥堵，提高道路安全。

在城市地区，有效的交通流管理对于最小化旅行时间、减少燃料消耗和降低排放至关重要。交通流受道路布局、交通信号、天气条件和驾驶员行为等多种因素影响。传统的交通流模型包括确定性模型（如运动波模型）和随机模型（如排队论和元胞自动机）。

近年来，数据收集技术的进步，如连续计数站（Continuous Counting Stations，简称 CCS），提供了高分辨率的交通数据。这些数据使得能够开发出更复杂的模型，更准确地捕捉交通流的动态和随机性。

本研究聚焦于利用非时齐马尔科夫过程对犹他州 Lehi 地区的交通流进行建模。通过将该地区划分为八个不同的区域，并利用 2023 年的 CCS 数据，我们旨在预测不同区域内的车辆数量变化。我们的目标是提供一个模型，帮助市政部门进行交通管理，提高道路效率，减少拥堵。

本文的以下部分将详细介绍我们的研究方法，展示研究结果，并讨论研究结果对城市交通管理的意义。

第二章 Problem Setting

在本研究中，我们的目标是通过马尔科夫过程对美国 Utah 州 Lehi 地区的交通流进行建模和预测。为此，我们首先需要对研究区域进行合理的划分，并引入相关数据以便进行建模和分析。

2.1 Lehi 地区交通网络的数学建模

2.1.1 基于 Lehi+Highland 地区道路网络和 CCS 数据的建模方法

图 2-1 即为本次研究的 Lehi 地区及周边地区的 Google 地图。我们研究的核心区域是红色区域内部的交通流, 即主要包括 Lehi 市区域 +Highland 市区域.

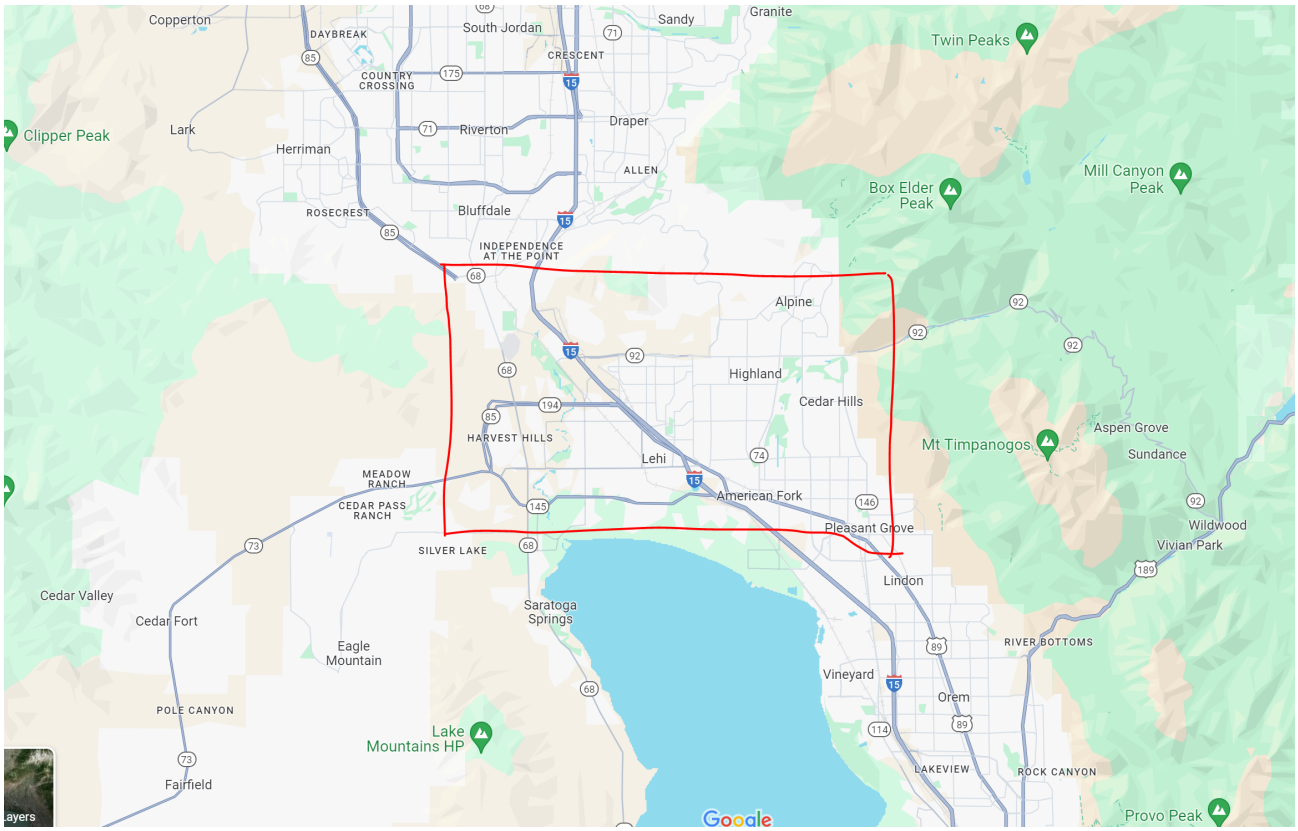

图 2–1 Utah, Lehi 地区的 Google 地图, 红圈部分即为本次研究的 Lehi 地区及周边地区

由地图可以发现, Lehi 地区的外部主要有四条通道, 分别对应东北通道, 东南通向 Orem 市通道, 西北通向 Bluffdale 市通道, 以及西南通向 Eagle Moutain 方向的通道.

我们的模型数据基于连续计数站（Continous Counting Station, CCS）提供的数据。CCS 是一种用于收集车辆流量数据的设备，通常安装在道路上，能够实时记录通过该点的车辆数量。图 2-2 展示了 Lehi 地区的 CCS 点的分布情况。

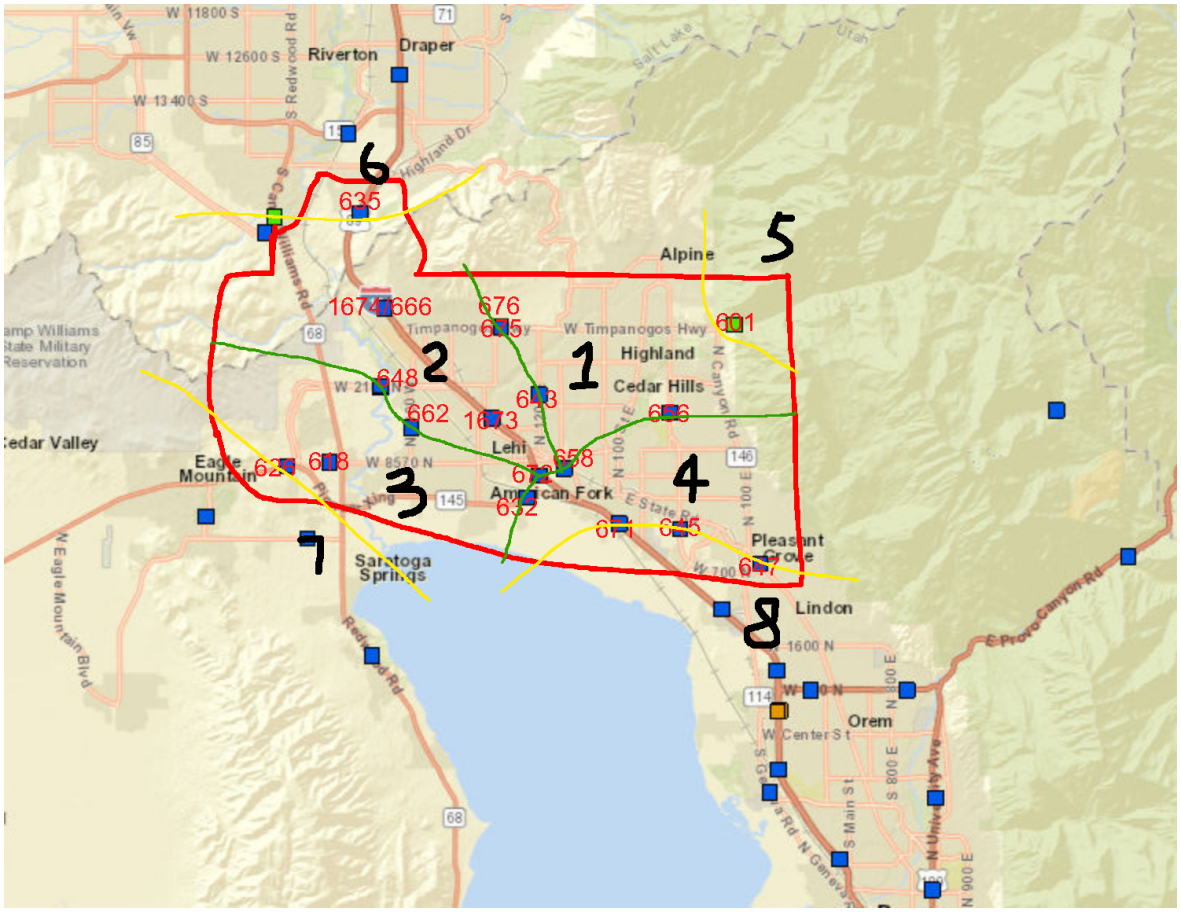

图 2–2 Utah, Lehi 地区的 World Street 格式地图,
description: 图中蓝色和橙色的方块点表示连续计数站 (CCS), 计数站上的数字表示该站点的 CCS 编号,
红圈内部围成的区域表示我们具体建模研究交通流的区域 (基本与 Lehi 地区 +Highland 地区的行政区域相重合),
四条绿线表示对于 CCS 的分布对于 Lehi 地区的内部城区的划分, 划分形成了 Region 1, Region 2, Region 3, Region 4 四个城区区域
四条淡黄线表示基于 CCS 对 Lehi 地区的车辆进出关卡的划分, 基于四条黄线划分形成了 Region 5, Region 6, Region 7, Region 8 四个城外区域

详细的数据收集情况以及区域的划分原则见附录中的数据收集情况说明。

这些 CCS 点位于不同区域的边界，能够记录通过这些点的车辆数量。因此，CCS 数据可以视为不同区域之间车辆流动的反映。

2.1.2 建模补充 Assumption

为了简化问题, 我们做出以下假设:

2.1.2.1 离散时间粒度假设

我们使用的 CCS 数据是以小时为单位, 即 CCS 返回的是从上一个小时到当前小时这个时间段内通过该 CCS 点的车辆数量。

结合地图上的距离, 车辆从一个区域到相邻的区域的时间一般不会超过一个小时, 而且不同区域之间的距离又使得车辆在一个小时内跨越两个以上的区域的概率极低.

因而我们认为以小时为单位的离散时间粒度是合理的, 在这个时间粒度下, 我们可以认为车辆仅能在一个小时内从当前区域移动到相邻的区域或者停留在当前区域.

2.1.2.2 马尔科夫性假设

离散时间粒度假设已经保证了车辆在一个小时内的移动是有限的, 即车辆下一时刻的所处状态仅能是当前时刻的状态或者相邻区域的状态.

基于大量车辆 (整个区域内的车辆) 的统计学角度, 我们假设在任一时间的车辆状态仅依赖于当前时刻的车辆状态, 在已知当前时刻的车辆状态下, 未来时刻的车辆状态与过去时刻的车辆状态无关.

则车辆的位置状态满足马尔科夫性质.

2.1.3 建模结果

基于上述的地区交通网络分析以及建模补充假设, 我们可以研究区域内任一一个车辆的位置状态建模为一个有限状态的马尔科夫过程.

根据图 2-2 中, 我们将研究区域划分为八个区域:

四个城区区域：Region 1, Region 2, Region 3, Region 4.

四个城外区域：Region 5, Region 6, Region 7, Region 8.

马尔科夫过程 $X_{\mathbb{T}} := \{X(t), t \in \mathbb{T}\}$ 中 X(T) 表示研究区域内任一车辆在时刻 t 的位置状态.

$X_{\mathbb{T}}$ 满足马尔科夫性质.

建立的有限状态马尔科夫状态空间定义为下:

- 状态 1(S_1): 车辆位于 Region 1
- 状态 2(S_2): 车辆位于 Region 2

- 状态 3(S_3): 车辆位于 Region 3
- 状态 4(S_4): 车辆位于 Region 4
- 状态 5(S_5): 车辆位于 Region 5
- 状态 6(S_6): 车辆位于 Region 6
- 状态 7(S_7): 车辆位于 Region 7
- 状态 8(S_8): 车辆位于 Region 8

状态空间 $S = \{S_1, S_2, \ldots, S_8\}$.

因此建模得到有限状态马尔科夫过程 $X_{\mathbb{T}}$.

第三章 离散时间马尔科夫过程建模

3.1 数据的时间间隔

离散时间马尔科夫建模遵循原始数据中的 1 小时的时间颗粒度.

3.2 原始数据处理

3.2.1 某日内不同时间段内不同区域车辆数量

表 3–1 2023-01-04 部分时间节点不同区域车辆数量

时间节点	区域 1	区域 2	区域 3	区域 4	区域 5	区域 6	区域 7	区域 8
00:00	16916	35858	35858	63663	15185	109641	46026	103826
01:00	16943	35946	35707	63602	15179	109742	46116	103738
02:00	16967	35905	35620	63649	15183	109826	46160	103663
03:00	16969	35914	35586	63688	15181	109895	46145	103595
…	…	…	…	…	…	…	…	…
23:00	16035	44588	33862	61584	15192	110606	47168	97938
24:00(00:00 of 01-05)	16108	44623	33656	61377	15184	110833	47354	97838

3.2.2 某日内部分时间段内不同区域之间车辆转移数量

表 3–2 2023-01-04 10:00 至 11:00 时间段内不同区域之间车辆转移数量

10:00 区域 \ 11:00 区域	区域 1	区域 2	区域 3	区域 4	区域 5	区域 6	区域 7	区域 8
区域 1	13611	1375	0	0	29	0	0	0
区域 2	1166	23197	1508	4404	0	5125	0	0
区域 3	0	1068	36894	1413	0	0	755	0
区域 4	0	5118	993	61746	0	0	0	5709
区域 5	20	0	0	0	15200	0	0	0
区域 6	0	5334	0	0	0	100451	0	0
区域 7	0	0	1094	0	0	0	40212	0
区域 8	0	0	0	6723	0	0	0	93828

3.3　k 步转移概率计算

1 小时的时间颗粒度下，1 步转移概率即为 1 小时内的转移概率。根据2.1.2, 我们假定一个小时内车辆仅能保持在当前区域或者移动到相邻区域。

因而 1 步转移概率矩阵 P 可以通过以下方式计算：

$$P_{i,j}^{1}(m) = \frac{\text{m:00 - (m+1):00 内从 i 区域转移到 j 区域的车辆的数量}}{\text{m 时刻 i 区域内车辆的总数}}$$

根据马尔科夫性质, k 步转移概率可以通过以下递推关系计算：

$$P_{i,j}^{k}(m) = \sum_{s=1}^{8} P_{i,s}^{k-1}(m) \cdot P_{s,j}^{1}(m+k-1)$$

3.4　k 步转移概率计算实验结果及 Chapman-Kolmogorov 方程验证

Chapman-Kolmogorov 方程是马尔科夫链理论中的一个基本方程，它描述了任意两个时间步之间的转移概率与中间时间步的关系。对于任意起始时间 m, 任意状态 i, j 和时间步 k, l，Chapman-Kolmogorov 方程表示为：

$$p_{i,j}^{(k+l)}(m) = \sum_{r \in \mathbb{S}} p_{i,r}^{(k)}(m) \cdot p_{r,j}^{(l)}(m+k)$$

为了验证我们的模型，我们计算实际的 $P^{k+l}(m)$ 矩阵, 与任意的 $k', l', k'+l' = k+l$ 得到的 $P^{k'}(m)$ 和 $P^{l'}(m+k')$ 矩阵相乘得到的 $P^{k'+l'}(m)$ 矩阵进行比较。

k 步转移概率计算实验结果及 Chapman-Kolmogorov 方程验证实验结果

表 3–3 实验基础设置

日期	m	k	l
2023-01-04	5	10	4

$P^{k}(m), P^{l}(m+k)$ 的实验结果见代码部分中的输出.

经验证, $P^{k+l}(m) = P^{k}(m) \cdot P^{l}(m+k)$, Chapman-Kolmogorov 方程成立, 说明我们的模型是正确的。

表 3–4 $P^{k+l}(m)$

区域	区域 1	区域 2	区域 3	区域 4	区域 5	区域 6	区域 7	区域 8
区域 1	0.2506058	0.1693671	0.0646380	0.1358869	0.0158096	0.2600794	0.016994	0.0866184
区域 2	0.0495052	0.1262049	0.0994541	0.1593780	0.0013739	0.3448476	0.0438547	0.1753812
区域 3	0.0276242	0.1102931	0.2518682	0.1556875	0.0003426	0.1305146	0.1850447	0.1386246
区域 4	0.0322355	0.1092912	0.0846435	0.2240014	0.0005065	0.163675	0.0274415	0.3582043
区域 5	0.0174510	0.0049318	0.0005777	0.0017172	0.972232	0.0025630	0.0000694	0.0004570
区域 6	0.0360223	0.1325303	0.0443310	0.0942868	0.0005163	0.6213757	0.0119326	0.0590044
区域 7	0.0063437	0.0370743	0.1562929	0.0430774	0.0000470	0.0269624	0.7060095	0.0241923
区域 8	0.0160048	0.0814676	0.0502231	0.2398649	0.0001346	0.069027	0.0101358	0.5331414

3.5 时齐离散马尔科夫链建模

3.5.1 小范围时齐假设

在本研究中，我们首先假定在较小的时间范围内，交通流的转移概率是时齐的。这意味着在这一时间范围内，转移概率矩阵 P 不随时间变化。基于这一假设，我们可以利用 k 步转移概率矩阵进行交通流的预测。

我们进行了实验，计算了 k 步转移概率矩阵，并用其预测了不同时间步的交通流情况。实验结果如下：

我们设定 $P = P^1(10)$

3.5.2 极限分布与平稳分布

基于时齐的马尔科夫链，我们还计算了从初始分布开始的极限分布和平稳分布。极限分布表示随着时间趋向无穷，系统状态的分布趋近于某一稳定分布。平稳分布则是满足 $\pi P = \pi$ 的分布，其中 π 表示平稳分布向量。

我们进行了实验，计算了极限分布和平稳分布，结果如下：

表 3–5 10:00 时刻的车流分布, 极限分布, 平稳分布

区域	区域 1	区域 2	区域 3	区域 4	区域 5	区域 6	区域 7	区域 8
10:00 时刻的车流分布	15015	35400	40130	73566	15220	105785	41306	100551
极限分布	21742	35895	26030	51147	7349	125712	83900	75196
平稳分布	21623	35913	26113	51185	6470	125768	84649	75250

平稳分布和极限分布基本相等 (存在部分计算误差), 但实际上和3–1 中任何时刻的车流分布都有较大差距, 这实际上也符合实际情况和我们的假设, 即交通流本身随

时间变化的幅度极大, 将交通流建模为时齐马尔科夫链是不合理的.

3.6　非时齐离散马尔科夫链建模

3.6.1　时齐马尔科夫链的局限性

通过上述实验结果可以看出，在较小的时间范围内，时齐马尔科夫链的预测效果是可以接受的。然而，随着时间的延长，预测偏差会逐渐增大。因此，基于时齐马尔科夫链的方法只能用于较短时间范围的预测。

3.6.2　非时齐马尔科夫链的引入

为了提高预测的准确性，我们尝试了非时齐马尔科夫链的建模方法。具体来说，我们将转移概率矩阵建模为线性模型，通过之前的两个转移概率矩阵来预测第三个转移概率矩阵。

一种简单的建模方法是线性插值。

我们建立如下的关于时间的转移概率矩阵的线性模型:

$$P = tA + P_0,$$

P 是转移概率矩阵, t 是时间, 单位是小时, A 是待定参数矩阵, P_0 是初始转移概率矩阵.

通过最小二乘法确定 A 和 P_0 的每个 unit 的, 从而得到 A 和 P_0.

非时齐马尔科夫链的 k 步转移概率矩阵及预测结果

实验部分利用非时齐马尔科夫链模型计算了 k 步转移概率矩阵，并预测了交通流情况。实验结果如下:

表 3–6 20:00 - 21:00 内从区域 1 出发的真实交通流, 非时齐离散建模预测交通流, 时齐离散建模预测交通流

区域 1 出发到达的区域	区域 1	区域 2	区域 3	区域 4	区域 5	区域 6	区域 7	区域 8
真实交通流	14671	814	0	0	21	0	0	0
非时齐建模预测交通流	14877	1595	0	0	64	0	0	0
时齐建模预测交通流	14010	4168	0	0	21	0	0	0

实验结果分析

实验结果表明，即使是简单线性插值构造的非时齐马尔科夫链模型在交通流预测方面也能取得较时齐马尔科夫链更好的效果。

这符合我们对于交通流的实际认识，交通流量存在显著的随时间变化的性质。早晚高峰、节假日等因素都会导致转移概率的变化，因此采用非时齐的方式能够更好地捕捉时间变化的动态特性，因此能够更有效地辅助交通管理和决策。

另一方面，交通流的非时齐变化性质比较复杂，当前简单的线性插值模型仍然存在较大的误差，需要更复杂的模型来进行建模.

第四章 连续时间马尔科夫过程建模

4.1 连续时间马尔科夫链建模的原因

在上一章节中,我们使用离散时间马尔科夫链进行了一些初步建模，但由于交通流本身是连续的，离散时间建模中 1 小时的时间跨度可能过大，不足以捕捉到交通流的细微变化。因此，为了更精确地反映交通流的动态变化，我们引入了连续时间马尔科夫链模型。

4.2 连续时间马尔科夫链的基本概念

连续时间马尔科夫链（Continuous-Time Markov Chain, CTMC）是一类随机过程，其中状态转移发生在连续时间点上。设 $\{X(t), t \geq 0\}$ 是一个随机过程，如果对于任意时间 t 和任意状态 i, j，都有

$$P(X(t+\Delta t) = j | X(t) = i, X(s), s \leq t) = P(X(t+\Delta t) = j | X(t) = i),$$

则称 $\{X(t)\}$ 为连续时间马尔科夫链。这里，转移概率 $P(X(t+\Delta t) = j | X(t) = i)$ 仅依赖于当前状态 i ，而与之前的状态无关，这体现了马尔科夫性质。

连续时间马尔科夫链的核心是生成矩阵 Q，它描述了状态转移的速率。Q 矩阵的定义为:

$$q_{ij} = \lim_{\Delta t \to 0} \frac{P(X(t+\Delta t) = j | X(t) = i) - \delta_{ij}}{\Delta t},$$

其中 δ_{ij} 是 Kronecker delta 函数。当 $i \neq j$ 时，q_{ij} 表示从状态 i 转移到状态 j 的速率；当 $i = j$ 时，q_{ii} 表示离开状态 i 的总速率。

4.3 连续时间马尔科夫链的 Q 矩阵计算

为了计算 Q 矩阵，我们使用线性拟合的方法。具体步骤如下：1. 计算一个小时、两个小时、三个小时等对应的 k 步转移概率矩阵 P^1, P^2, P^3。2. 使用线性拟合的方法，拟合得到函数

$$\frac{p_{i,j}(t) - \delta_{i,j}}{t}$$

3. 带入 $t = \epsilon$，得到拟合的 Q 矩阵。

表 4–1 线性拟合得到的 12:00 的 Q 矩阵

状态	区域 1	区域 2	区域 3	区域 4	区域 5	区域 6	区域 7	区域 8
区域 1	-0.1441	0.1525	-0.0013	-0.0031	0.0024	-0.0051	-0.0001	-0.0009
区域 2	0.0531	-0.4390	0.0492	0.1823	-0.0000	0.1584	-0.0005	-0.0037
区域 3	-0.0005	0.0424	-0.1043	0.0395	-0.0000	-0.0021	0.0265	-0.0015
区域 4	-0.0008	0.0987	0.0186	-0.2109	-0.0000	-0.0036	-0.0003	0.0984
区域 5	0.0019	-0.0001	-0.0000	-0.0000	-0.0017	-0.0000	-0.0000	-0.0000
区域 6	-0.0006	0.0565	-0.0007	-0.0018	-0.0000	-0.0530	-0.0000	-0.0004
区域 7	-0.0000	-0.0004	0.0285	-0.0004	-0.0000	-0.0001	-0.0275	-0.0000
区域 8	-0.0001	-0.0016	-0.0007	0.0782	-0.0000	-0.0005	-0.0000	-0.0752

4.4 时齐连续时间马尔科夫链的建模

4.4.1 局部范围内假设时齐

在局部时间范围内，我们假设交通流的状态转移速率是时齐的，即 Q 矩阵不随时间变化。基于这一假设，我们可以用 Q 矩阵预测特定时间段内的交通流量。

通过矩阵指数 e^{Qt}，我们可以得到时间 t 内的转移概率矩阵：

$$P(t) = e^{Qt}.$$

我们进行了实验，利用 Q 矩阵预测了不同时间段内的交通流情况，结果如下：

时齐假设的平稳分布计算

表 4–2 8:00 时刻的车流分布, 根据 8:00 Q 矩阵计算得到的平稳分布

区域	区域 1	区域 2	区域 3	区域 4	区域 5	区域 6	区域 7	区域 8
8:00 时刻的车流分布	16150	36407	39751	68752	15191	104467	42768	103487
平稳分布	9936	36558	34845	80286	27936	119933	17330	100144

4.5 非时齐连续时间马尔科夫链的建模

与离散时间马尔科夫链类似，我们在连续时间马尔科夫链中也尝试了非时齐建模的方法。我们将转移速率矩阵 Q 建模为线性模型.

我们建立如下的关于时间的 Q 矩阵的线性模型：

$$Q = tB + Q_0,$$

Q 是速率转移矩阵, t 是时间, 单位是小时, B 是待定参数矩阵, Q_0 是初始 Q 矩阵.

通过最小二乘法确定 B 和 Q_0 的每个元素的, 从而得到 B 和 Q_0.

我们进行了实验，利用非时齐马尔科夫链模型计算了 k 步转移概率矩阵，并预测了交通流情况。实验结果如下:

表 4–3 20:00 - 21:00 内从区域 1 出发的真实交通流, 非时齐连续建模预测交通流, 时齐连续建模预测交通流

区域 1 出发到达的区域	区域 1	区域 2	区域 3	区域 4	区域 5	区域 6	区域 7	区域 8
真实交通流	14671	814	0	0	21	0	0	0
非时齐建模预测交通流	14821	1685	10	28	75	0	0	0

第五章 Summary

5.1 工作总结与反思

在本次研究中，我们针对美国 Utah 州 Lehi 地区的交通流进行了详细的建模与分析。我们将 Lehi 地区划分为八个区域，利用马尔科夫过程对不同区域之间的交通流进行了建模。通过使用 2023 年全年的交通数据，我们分别采用了离散时间马尔科夫链和连续时间马尔科夫链对交通流进行了预测。

主要工作

1 基于离散时间马尔科夫链对交通流进行建模，并计算了 k 步转移概率以进行交通流的预测。

2 验证了 Chapman-Kolmogorov 方程，确保了模型的有效性。

3 探讨了时齐和非时齐的离散时间马尔科夫链建模，发现非时齐模型在长时间预测中表现更佳。

4 引入了连续时间马尔科夫链模型，通过拟合的方法计算 Q 矩阵，并验证

5 探讨了非时齐的连续时间马尔科夫链建模，进一步提高了预测精度。

主要成果

1 离散时间和连续时间马尔科夫链模型均能够较好地捕捉交通流的动态变化。

2 非时齐模型在长时间预测中表现出更高的准确性，证明了其在交通流建模中的潜力。

研究的不足之处

1 数据的时间间隔较大，1 小时的跨度限制了模型的精度。

2 由于数据权限的限制，我们只能获得一定区域内的车辆总数，缺乏更细粒度的数据。

5.2 对后续工作的思考

基于本次研究的总结，我们对未来的工作提出了以下几点思考和建议：

数据权限的提升

当前研究中，我们使用的数据粒度较粗，主要是每小时的车辆数。如果未来能够获得更加实时的交通数据，将能够显著提高模型的精度和预测能力。

泊松分布建模不同区域之间的状态转移

如果能够获得实时的交通数据，我们可以将每个车辆的到达过程建模为泊松分布。连续时间马尔科夫链可以结合泊松分布，来更精确地描述车辆到达的随机过程。具体来说，在每个区域内，车辆到达的过程可以被看作是泊松过程，而每个区域内车辆的总数则可以通过连续时间马尔科夫链来描述。这种方法能够弥补当前马尔科夫建模中粒度较大的不足，使得模型能够更精细地捕捉交通流的变化。

更细粒度的建模

未来的研究可以进一步细化交通流的建模，例如对每一条街道进行建模。通过利用街道上的摄像头进行车辆识别，可以实时监测每条街道的交通情况。这样不仅可以提供更细粒度的数据，还能够更准确地反映交通流的变化和拥堵情况，从而为城市交通管理提供更有力的支持。

未来，随着数据获取能力的提升和模型的不断优化，我们有理由相信，基于马尔科夫过程的交通流建模方法将在城市交通管理和优化中发挥更加重要的作用。

附录 A 参考文献及参考数据

应用随机过程第二版韩东, 熊德文, 高等教育出版社, 2023.

美国 Utah 州连续计数站的每小时交通量 (Hourly Volumes From Continuous Counting Stations (CCS)) 数据来源 https://udot.utah.gov/connect/business/traffic-data/traffic-statistics/, https://drive.google.com/drive/folders/1ZYy-WkICLOp1482vwEbTc5UvLItbWs4y

美国 Utah 州 CCS 的拓扑结构数据来源 https://hub.arcgis.com/datasets/017ab39594bc4e37964884a explore?location=37.365873%2C-112.725309%2C6.00

美国 Utah 州人均车辆数据来源 https://en.wikipedia.org/wiki/List_of_U.S._states_by_vehicles_per_capita

美国 Utah 州 Lehi 地区, Highland 地区, American Fork 地区, Pleasant Grove 地区, Heber City, Bluffdale, Herriman, Draper, Eagle Mountain, Lindon, Orem, Vineyard 地区人口数据来源 https://datacommons.org/place/geoId/4944320

美国 Utah 州的 Google 地图数据来源 https://www.google.com/maps/@40.41817, -111.8277396,11.25z?entry=ttu

附录 B 数据处理

B.1 Utah 州 2023 年 Lehi 地区 CCS 数据

根据附录 A 获得的 Utah 州 2023 年 Lehi 地区 CCS 数据，格式如下.

2023-Station-501-733.xlsx 文件的部分如下表所示:

表 B–1 Utah 州 2023 年 Lehi 地区 CCS 数据

STATION	DATE	ROUTE	MP	LANE	H0000	H0100
-0501	2023/1/1	0215	026.400	NLane1	28	23
-0501	2023/1/1	0215	026.400	NLane2	231	181
-0501	2023/1/1	0215	026.400	NLane3	199	188
-0501	2023/1/1	0215	026.400	NLane4	42	69
-0501	2023/1/1	0215	026.400	PLane1	141	115

其中，STATION 表示 CCS 编号，DATE 表示日期，ROUTE 表示路线，MP 表示里程，LANE 表示车道，H0000 表示 00:00-01:00 通过这个技术站的车辆数，H0100 表示 01:00-02:00 通过这个技术站的车辆数。

Nlane, Plane 分别代表不同方向的车道, 即如 Plane 代表从 Region 1 进入 Region 2 的车道, 则 Nlane 代表从 Region 2 进入 Region 1 的车道. (具体的 Plane 和 Nlane 的代表的方向可见附录 C 代码部分)

因此, 分析上述数据可以获得 2023 年内, 以小时为单位的通过 CSS 的车辆的数量.

B.2 不同 Region 总车辆数量数据预估

因为数据管制, 我们无法获得每个 Region 实时的车辆数量数据. 因此我们采用了结合 Utah 州 2022 年 Lehi 地区等各个城市的人口数据, 以及 Utah 州 2022 年的人均车辆拥有量数据, 来估计 2023 年 Lehi 地区初始状态各个 Region 的车辆数量.

附录 C 代码

代码和数据包含在本次上交的文件中, 下列了实验的部分关键代码及输出. 完整的代码请见随作业上传的附件 code, 或https://github.com/BobHuangC/MATH4704-MarkovTraffic.

selected_code

May 31, 2024

[ ]:
```
import model
import numpy as np

_initial_traffic_amouts_list = [region_1_initial_traffic_amount,␣
 ↪region_2_initial_traffic_amount,
                                region_3_initial_traffic_amount,␣
 ↪region_4_initial_traffic_amount,
                                region_5_initial_traffic_amount,␣
 ↪region_6_initial_traffic_amount,
                                region_7_initial_traffic_amount,␣
 ↪region_8_initial_traffic_amount]

# initialization of CCSs
CCS_601, CCS_626, CCS_632, CCS_635, CCS_643,\
CCS_645, CCS_647, CCS_648, CCS_656, CCS_658,\
CCS_662, CCS_671, CCS_672, CCS_675, CCS_676,\
Region_1, Region_2, Region_3, Region_4,\
Region_5, Region_6, Region_7, Region_8 = \
model.initialize_CSSs_and_Regions(date=Date,
                                  ␣
 ↪initials_traffic_amount_list=_initial_traffic_amouts_list)

model.initialize_region_transition_matrix_v2()
```

```
CCS idx:  645
Warning: the data for this date is empty
CCS idx:  656
Warning: the data for this date is empty
The initial traffic amount of Region  1  is  16916
The initial traffic amount of Region  2  is  35858
The initial traffic amount of Region  3  is  35858
The initial traffic amount of Region  4  is  63663
The initial traffic amount of Region  5  is  15185
The initial traffic amount of Region  6  is  109641
The initial traffic amount of Region  7  is  46026
The initial traffic amount of Region  8  is  103826
```

[ ]:
```
print("The traffic amount of each region during the day: ")
print(model.current_time_traffic_amount)
```

```
The traffic amount of each region during the day:
[[ 16916.  35858.  35858.  63663.  15185. 109641.  46026. 103826.]
 [ 16943.  35946.  35707.  63602.  15179. 109742.  46116. 103738.]
 [ 16967.  35905.  35620.  63649.  15183. 109826.  46160. 103663.]
 [ 16969.  35914.  35586.  63688.  15181. 109895.  46145. 103595.]
 [ 16968.  35946.  35642.  63821.  15178. 109753.  46082. 103583.]
 [ 16930.  35971.  35902.  64095.  15178. 109400.  45834. 103663.]
 [ 16932.  35833.  36694.  65058.  15179. 108329.  45168. 103780.]
 [ 16559.  36233.  37943.  66497.  15183. 106229.  44296. 104033.]
 [ 16150.  36407.  39751.  68752.  15191. 104467.  42768. 103487.]
 [ 15477.  35416.  40055.  71770.  15211. 105269.  41848. 101927.]
 [ 15015.  35400.  40130.  73566.  15220. 105785.  41306. 100551.]
 [ 14797.  36092.  40489.  74286.  15229. 105576.  40967.  99537.]
 [ 14551.  36569.  40764.  74423.  15236. 105759.  40824.  98847.]
 [ 14319.  37533.  40645.  74251.  15243. 105610.  40808.  98564.]
 [ 14232.  38345.  40489.  74243.  15251. 105610.  40897.  97906.]
 [ 14317.  38893.  40266.  73557.  15267. 105994.  40991.  97688.]
 [ 14470.  39389.  39771.  72068.  15271. 106720.  41637.  97647.]
 [ 14572.  41414.  38993.  69768.  15250. 106958.  42518.  97500.]
 [ 14901.  42893.  37977.  67421.  15210. 106314.  43538.  98719.]
 [ 15237.  43090.  36368.  65161.  15191. 108393.  44984.  98549.]
 [ 15506.  43821.  35333.  64029.  15205. 109208.  45811.  98060.]
 [ 15732.  44064.  34637.  62871.  15195. 109772.  46380.  98322.]
 [ 15934.  44384.  34205.  62110.  15197. 110170.  46813.  98160.]
 [ 16035.  44588.  33862.  61584.  15192. 110606.  47168.  97938.]
 [ 16108.  44623.  33656.  61377.  15184. 110833.  47354.  97838.]]
```

[ ]:
```
np.set_printoptions(precision=5, suppress=True)
np.set_printoptions(threshold=np.inf)
print("The transition amount among different regions: ")
print()
# for _ in range(model.hourly_traffic_among_regions.shape[0]):
for _ in range(2):
    print(f'The transition amount during {_}:00 to {_+1}:00')
    print(model.hourly_traffic_among_regions[_])
    print()
```

```
The transition amount among different regions:

The transition amount during 0:00 to 1:00
[[ 16837.     76.      0.      0.      3.      0.      0.      0.]
 [    97.  34210.    111.    644.      0.    796.      0.      0.]
 [     0.    260.  35351.     98.      0.      0.    149.      0.]
 [     0.    705.    186.  61967.      0.      0.      0.    805.]
```

```
 [     9.      0.      0.      0.  15176.      0.      0.      0.]
 [     0.    695.      0.      0.      0. 108946.      0.      0.]
 [     0.      0.     59.      0.      0.      0.  45967.      0.]
 [     0.      0.      0.    893.      0.      0.      0. 102933.]]

The transition amount during 1:00 to 2:00
[[ 16910.     28.      0.      0.      5.      0.      0.      0.]
 [    56.  34918.     63.    411.      0.    498.      0.      0.]
 [     0.    125.  35459.     54.      0.      0.     69.      0.]
 [     0.    420.     73.  62624.      0.      0.      0.    485.]
 [     1.      0.      0.      0.  15178.      0.      0.      0.]
 [     0.    414.      0.      0.      0. 109328.      0.      0.]
 [     0.      0.     25.      0.      0.      0.  46091.      0.]
 [     0.      0.      0.    560.      0.      0.      0. 103178.]]
```

[ ]:
```
# experiment 0: the calculation of k-step transition probability matrix of␣
 ↪discrete time Markov Chain
exp0_begin_hour = 0
exp0_k = 5

print("Experiment 0")
print('The construction of Markov Chain and the calculation of k-step␣
 ↪transition probability matrix')
print('--------------------------------------------------')
# # print(Region_1.getTransitionMatrix2Regions(begin_hour=exp0_hour1,␣
 ↪end_hour=exp0_hour3))

print(f"The {exp0_k}-step transition matrix from hour {exp0_begin_hour}")
print(model.get_hour_k_step_transition_matrix(begin_hour=exp0_begin_hour,␣
 ↪k=exp0_k))
```

```
Experiment 0
The construction of Markov Chain and the calculation of k-step transition
probability matrix
--------------------------------------------------
The 5-step transition matrix from hour 0
[[0.9844283927 0.0141073345 0.0001000432 0.0003684498 0.0006464278
  0.0003457101 0.0000001108 0.0000035311]
 [0.0068681506 0.8444942206 0.014918352  0.0639403918 0.0000015379
  0.0685430255 0.0000392506 0.0011950709]
 [0.000051876  0.0170542394 0.9569735787 0.0144051392 0.000000002
  0.0005869097 0.010720576  0.000207679 ]
 [0.000087474  0.0330524964 0.0070290145 0.9138798042 0.000000003
  0.0009898659 0.0000304349 0.044930907 ]
 [0.0011743182 0.0000102107 0.0000000329 0.0000001071 0.9988152403
  0.0000000903 0.           0.0000000004]
```

```
[0.000053063  0.0245519889 0.000180023  0.0006495359 0.0000000017
 0.9745595506 0.000000179  0.0000056578]
[0.0000000352 0.0000319309 0.012518772  0.0000458253 0.
 0.0000003978 0.9874028041 0.0000002347]
[0.0000004876 0.0003968684 0.00006541   0.0291429578 0.
 0.0000054856 0.0000001309 0.9703886598]]
```

[ ]:
```
# experiment 1: the verification of the Chapman-Kolmogorov equation
exp1_m = 5
exp1_k = 10
exp1_l = 4

exp1_P_begin_m_step_k_plus_l = model.get_hour_k_step_transition_matrix(
    ␣
 ↪                                                          begin_hour=exp1_m,
                                    k=exp1_k+exp1_l)

exp1_P_begin_m_step_k = model.get_hour_k_step_transition_matrix(
                                                                  begin_hour=exp1_m,
                                                                  k=exp1_k)

exp1_P_begin_m_plus_k_step_l = model.get_hour_k_step_transition_matrix(
                                                                  begin_hour=exp1_m+exp1
                                                                  k=exp1_l)


print("Experiment 1")
print('The Chapman-Kolmogorov equation verification')
print('-----------------------------------------------')
print("The left side of the Chapman-Kolmogorov equation")
print(exp1_P_begin_m_step_k_plus_l)
print()
print("The right side of the Chapman-Kolmogorov equation")
print(np.dot(exp1_P_begin_m_step_k, exp1_P_begin_m_plus_k_step_l))
print()
print("The equality of the left side and the right side of the␣
 ↪Chapman-Kolmogorov equation")
print(np.allclose(exp1_P_begin_m_step_k_plus_l,
                                  np.dot(exp1_P_begin_m_step_k,␣
 ↪exp1_P_begin_m_plus_k_step_l)))

print("exp1_P_begin_m_step_k_plus_l")
print(exp1_P_begin_m_step_k_plus_l)
print()
print("exp1_P_begin_m_step_k")
print(exp1_P_begin_m_step_k)
print()
print("exp1_P_begin_m_plus_k_step_l")
```

```
print(exp1_P_begin_m_plus_k_step_l)
print()
```

```
Experiment 1
The Chapman-Kolmogorov equation verification
--------------------------------------------------
The left side of the Chapman-Kolmogorov equation
[[0.2506058269 0.1693671176 0.0646380422 0.1358869916 0.0158096418
  0.2600794715 0.016994452  0.0866184563]
 [0.0495052355 0.1262049758 0.0994541392 0.1593780286 0.0013739797
  0.3448476217 0.0438547831 0.1753812364]
 [0.0276242593 0.1102931417 0.2518682041 0.1556875836 0.0003426976
  0.1305146811 0.1850447622 0.1386246704]
 [0.0322355785 0.1092912909 0.0846435845 0.2240014307 0.0005065758
  0.163675691  0.0274415028 0.3582043458]
 [0.0174510871 0.0049318644 0.0005777729 0.0017172704 0.972232541
  0.0025630259 0.0000694184 0.0004570199]
 [0.0360223586 0.1325303742 0.0443310845 0.0942868838 0.0005163503
  0.6213757889 0.0119326711 0.0590044885]
 [0.0063437213 0.0370743964 0.1562929921 0.0430774328 0.0000470468
  0.0269624525 0.7060095619 0.0241923962]
 [0.0160048253 0.0814676324 0.0502231965 0.2398649274 0.0001346311
  0.069027515  0.0101358561 0.5331414163]]

The right side of the Chapman-Kolmogorov equation
[[0.2506058269 0.1693671176 0.0646380422 0.1358869916 0.0158096418
  0.2600794715 0.016994452  0.0866184563]
 [0.0495052355 0.1262049758 0.0994541392 0.1593780286 0.0013739797
  0.3448476217 0.0438547831 0.1753812364]
 [0.0276242593 0.1102931417 0.2518682041 0.1556875836 0.0003426976
  0.1305146811 0.1850447622 0.1386246704]
 [0.0322355785 0.1092912909 0.0846435845 0.2240014307 0.0005065758
  0.163675691  0.0274415028 0.3582043458]
 [0.0174510871 0.0049318644 0.0005777729 0.0017172704 0.972232541
  0.0025630259 0.0000694184 0.0004570199]
 [0.0360223586 0.1325303742 0.0443310845 0.0942868838 0.0005163503
  0.6213757889 0.0119326711 0.0590044885]
 [0.0063437213 0.0370743964 0.1562929921 0.0430774328 0.0000470468
  0.0269624525 0.7060095619 0.0241923962]
 [0.0160048253 0.0814676324 0.0502231965 0.2398649274 0.0001346311
  0.069027515  0.0101358561 0.5331414163]]

The equality of the left side and the right side of the Chapman-Kolmogorov
equation
True
exp1_P_begin_m_step_k_plus_l
[[0.2506058269 0.1693671176 0.0646380422 0.1358869916 0.0158096418
```

```
  0.2600794715 0.016994452  0.0866184563]
 [0.0495052355 0.1262049758 0.0994541392 0.1593780286 0.0013739797
  0.3448476217 0.0438547831 0.1753812364]
 [0.0276242593 0.1102931417 0.2518682041 0.1556875836 0.0003426976
  0.1305146811 0.1850447622 0.1386246704]
 [0.0322355785 0.1092912909 0.0846435845 0.2240014307 0.0005065758
  0.163675691  0.0274415028 0.3582043458]
 [0.0174510871 0.0049318644 0.0005777729 0.0017172704 0.972232541
  0.0025630259 0.0000694184 0.0004570199]
 [0.0360223586 0.1325303742 0.0443310845 0.0942868838 0.0005163503
  0.6213757889 0.0119326711 0.0590044885]
 [0.0063437213 0.0370743964 0.1562929921 0.0430774328 0.0000470468
  0.0269624525 0.7060095619 0.0241923962]
 [0.0160048253 0.0814676324 0.0502231965 0.2398649274 0.0001346311
  0.069027515  0.0101358561 0.5331414163]]

exp1_P_begin_m_step_k
[[0.4005255823 0.1957855316 0.0514273197 0.1258562575 0.0124194497
  0.173988575  0.0045836281 0.0354136561]
 [0.0493455937 0.128926448  0.1229724874 0.2037515056 0.0009053714
  0.3394136555 0.0214460581 0.1332388804]
 [0.0166751698 0.0986345964 0.4190169135 0.1779382236 0.0001593687
  0.0761537386 0.1239353765 0.087486613 ]
 [0.0236217974 0.1099074607 0.0818064328 0.3151652489 0.0002604237
  0.1155309517 0.0100301026 0.3436775822]
 [0.0108409078 0.0019306334 0.0001318286 0.0004363272 0.9861250936
  0.0004853273 0.0000044721 0.00004541  ]
 [0.0241182857 0.1224395523 0.0353019444 0.0813054306 0.0002601054
  0.7065911312 0.0035854793 0.0263980711]
 [0.002430745  0.0218099086 0.1723466473 0.0322113687 0.000016689
  0.0097605954 0.7517241107 0.0096999354]
 [0.0070795362 0.0572163085 0.0281835732 0.2617176897 0.0000494966
  0.0284044816 0.001955505  0.6153934092]]

exp1_P_begin_m_plus_k_step_l
[[0.5709600048 0.212641907  0.0213281927 0.0751337537 0.0087319926
  0.099375168  0.0012679087 0.0105610723]
 [0.0818966876 0.1876900158 0.0844013349 0.2044213077 0.0003300085
  0.3481103087 0.0122214089 0.0809289279]
 [0.0121497469 0.1147258433 0.5164217268 0.114288705  0.0000086348
  0.0452020938 0.1683054484 0.028897801 ]
 [0.0204337429 0.1480006181 0.0804933266 0.3609520312 0.0000168592
  0.0761182666 0.0093278371 0.3046573183]
 [0.0112416982 0.0021526962 0.0000748936 0.0003107232 0.985815323
  0.0003891397 0.0000014669 0.0000140593]
 [0.0142647108 0.1235355002 0.0113461493 0.040357885  0.0000102517
  0.8043976932 0.0006503853 0.0054374245]
 [0.0003452269 0.0084395956 0.083207715  0.0062369044 0.0000000499
```

```
 0.0012853271 0.8998233601 0.0006618209]
[0.0019177303 0.0385606346 0.0148473065 0.2092801588 0.0000003075
 0.0070847566 0.0007553283 0.7275537773]]
```

[ ]:
```
# experiment 2: the construction of time-homogeneous Markov Chain and the calculation of limit distribution and stationary distribution

# # Suppose the traffic flow is a time-homogeneous Markov chain
# # The transition matrix is the same for all hours
exp2_begin_hour = 0
exp2_k = 1
exp2_initial_distribution_time = 10

exp2_initial_distribution = model.current_time_traffic_amount[exp2_initial_distribution_time]
exp2_initial_distribution = exp2_initial_distribution / np.sum(exp2_initial_distribution)
exp2_transition_matrix_P = model.get_hour_k_step_transition_matrix(
                                                                    begin_hour=exp2_begin_hour,
                                                                    k=exp2_k)

# compute the limit distribution of the traffic flow according to the transition matrix and the initial distribution
def compute_limit_distribution(transition_matrix, initial_distribution, alltol= 1e-6):
        next_distribution = np.dot(initial_distribution, transition_matrix)
        while not np.allclose(initial_distribution, next_distribution,atol=alltol):
                initial_distribution = next_distribution
                next_distribution = np.dot(initial_distribution,transition_matrix)
        return next_distribution

print("Experiment 2")
print('The construction of time-homogeneous Markov Chain and the calculation of limit distribution')
print('--------------------------------------------------')
print('The transition matrix of the traffic flow:')
print(exp2_transition_matrix_P)
print()
print('The initial distribution of the traffic flow:')
print(exp2_initial_distribution)
print(exp2_initial_distribution * np.sum(_initial_traffic_amouts_list))
print()
exp2_limit_distribution = compute_limit_distribution(exp2_transition_matrix_P,exp2_initial_distribution)
```

```
print('The limit distribution of the traffic flow:')
print(exp2_limit_distribution)
print(exp2_limit_distribution * np.sum(_initial_traffic_amouts_list))



# compute the stationary distribution of the traffic flow according to the␣
 ↪transition matrix
def compute_stationary_distribution(transition_matrix):
        eigenvalues, eigenvectors = np.linalg.eig(transition_matrix.T)
        stationary_index = np.where(np.isclose(eigenvalues, 1))[0][0]
        stationary_distribution = np.real(eigenvectors[:, stationary_index])
        stationary_distribution /= np.sum(stationary_distribution)
        return stationary_distribution


print("Experiment 2")
print('The construction of time-homogeneous Markov Chain and the calculation of␣
 ↪stationary distribution')
print('-------------------------------------------------')
exp2_stationary_distribution =␣
 ↪compute_stationary_distribution(exp2_transition_matrix_P)
print('The stationary distribution of the traffic flow:')
print(exp2_stationary_distribution)
print(exp2_stationary_distribution * np.sum(_initial_traffic_amouts_list))
```

```
Experiment 2
The construction of time-homogeneous Markov Chain and the calculation of limit
distribution
-------------------------------------------------
The transition matrix of the traffic flow:
[[0.9953298652 0.0044927879 0.           0.           0.0001773469
  0.           0.           0.          ]
 [0.0027051146 0.9540409393 0.0030955435 0.01795973   0.
  0.0221986725 0.           0.          ]
 [0.           0.0072508227 0.9858608958 0.0027330024 0.
  0.           0.0041552792 0.          ]
 [0.           0.0110739362 0.0029216342 0.9733597223 0.
  0.           0.           0.0126447073]
 [0.0005926902 0.           0.           0.           0.9994073098
  0.           0.           0.          ]
 [0.           0.0063388696 0.           0.           0.
  0.9936611304 0.           0.          ]
 [0.           0.           0.0012818842 0.           0.
  0.           0.9987181158 0.          ]
 [0.           0.           0.           0.0086009285 0.
  0.           0.           0.9913990715]]
```

```
The initial distribution of the traffic flow:
[0.035166158  0.0829092238 0.0939872076 0.1722966089 0.0356462821
 0.2477557129 0.0967414801 0.2354973265]
[ 15015.  35400.  40130.  73566.  15220. 105785.  41306. 100551.]

The limit distribution of the traffic flow:
[0.0509213378 0.0840687562 0.060964514  0.11979023   0.0172132835
 0.2944262904 0.1964999378 0.1761156503]
[ 21742.0363622036  35895.089035811   26030.2014381436  51147.1938933037
   7349.6073096246 125712.0764899111  83900.1679231133  75196.6275478868]
Experiment 2
The construction of time-homogeneous Markov Chain and the calculation of
stationary distribution
--------------------------------------------------
The stationary distribution of the traffic flow:
[0.0506437051 0.0841115653 0.061160225  0.1198787901 0.0151537925
 0.2945580551 0.1982533349 0.176240532 ]
[ 21623.4946878873  35913.3673916747  26113.764748294   51185.0066352778
   6470.2602434536 125768.3364677012  84648.8211549889  75249.9486707226]
```

[ ]:
```
# experiment 3: the construction of non-time-homogeneous Markov Chain

# exp3_begin_hour - exp3_begin_hour + 3.
# We Use the linear model to construct the transition matrix of␣
 ↪non-time-homogeneous Markov Chain
exp3_begin_hour = 18
exp3_end_hour = exp3_begin_hour + 3


# We combine the transition matrix of exp3_begin_hour - exp3_begin_hour + 1 and␣
 ↪exp3_begin_hour + 1 - exp3_begin_hour + 2
# to construct the transition matrix of exp3_begin_hour + 2 - exp3_begin_hour +␣
 ↪3
exp3_transition_matrix_6_7 = model.
 ↪get_hour_k_step_transition_matrix(begin_hour=exp3_begin_hour, k=1)
exp3_transition_matrix_7_8 = model.
 ↪get_hour_k_step_transition_matrix(begin_hour=exp3_begin_hour+1, k=1)

# Use the linear interpolation to construct the transition matrix of␣
 ↪exp3_begin_hour + 2 - exp3_begin_hour + 3
exp3_model_transition_matrix_8_9 = 2 * exp3_transition_matrix_7_8 -␣
 ↪exp3_transition_matrix_6_7
exp3_real_transition_matrix_8_9 = model.
 ↪get_hour_k_step_transition_matrix(begin_hour=exp3_begin_hour+2, k=1)
```

```
print("Experiment 3")
print('The construction of non-time-homogeneous Markov Chain')
print('-------------------------------------------------')
print(f'The transition matrix of the traffic flow from {exp3_begin_hour}:00 to␣
 ↪{exp3_begin_hour+1}:00:')
print(exp3_transition_matrix_6_7)
print()
print(f"The transition matrix of the traffic flow from {exp3_begin_hour+1}:00␣
 ↪to {exp3_begin_hour+2}:00:")
print(exp3_transition_matrix_7_8)
print()
print(f'The model transition matrix of the traffic flow from␣
 ↪{exp3_begin_hour+2}:00 to {exp3_begin_hour+3}:00:')
print(exp3_model_transition_matrix_8_9)
print()
print(f'The real transition matrix of the traffic flow from {exp3_begin_hour+2}:
 ↪00 to {exp3_begin_hour+3}:00:')
print(exp3_real_transition_matrix_8_9)
print()


print('The real transition number is ')
print(model.hourly_traffic_among_regions[exp3_begin_hour+2][0])

print('The Non Time Alignment model transition number is ')
print(model.current_time_traffic_amount[exp3_begin_hour+2] *␣
 ↪exp3_model_transition_matrix_8_9[0])

print('The Time Alignment model transition number is ')
print(model.current_time_traffic_amount[exp3_begin_hour+2] *␣
 ↪(exp3_transition_matrix_6_7+exp3_transition_matrix_7_8)[0]/2)
```

```
Experiment 3
The construction of non-time-homogeneous Markov Chain
-------------------------------------------------
The transition matrix of the traffic flow from 18:00 to 19:00:
[[0.8849070532 0.1146902892 0.           0.           0.0004026575
  0.           0.           0.          ]
 [0.0472338144 0.5854568344 0.0332222041 0.1458046768 0.
  0.1882824703 0.           0.          ]
 [0.           0.0650920294 0.8404560655 0.0338362693 0.
  0.           0.0606156358 0.          ]
 [0.           0.1156909568 0.0321709853 0.7338959671 0.
  0.           0.           0.1182420907]
 [0.0016436555 0.           0.           0.           0.9983563445
  0.           0.           0.          ]
```

```
 [0.           0.0564083752 0.           0.           0.
  0.9435916248 0.           0.          ]
 [0.           0.           0.0196609858 0.           0.
  0.           0.9803390142 0.          ]
 [0.           0.           0.           0.0824765243 0.
  0.           0.           0.9175234757]]

The transition matrix of the traffic flow from 19:00 to 20:00:
[[0.9221631555 0.0755398044 0.           0.           0.0022970401
  0.           0.           0.          ]
 [0.0332791831 0.7441169645 0.0239266651 0.093153864  0.
  0.1055233233 0.           0.          ]
 [0.           0.0490541135 0.8856687198 0.0257094149 0.
  0.           0.0395677519 0.          ]
 [0.           0.0781142094 0.0227129725 0.8215190068 0.
  0.           0.           0.0776538113]
 [0.0013823975 0.           0.           0.           0.9986176025
  0.           0.           0.          ]
 [0.           0.0344302676 0.           0.           0.
  0.9655697324 0.           0.          ]
 [0.           0.           0.0136048373 0.           0.
  0.           0.9863951627 0.          ]
 [0.           0.           0.           0.0563070148 0.
  0.           0.           0.9436929852]]

The model transition matrix of the traffic flow from 20:00 to 21:00:
[[0.9594192577 0.0363893196 0.           0.           0.0041914227
  0.           0.           0.          ]
 [0.0193245518 0.9027770945 0.0146311261 0.0405030513 0.
  0.0227641762 0.           0.          ]
 [0.           0.0330161976 0.930881374  0.0175825604 0.
  0.           0.018519868  0.          ]
 [0.           0.040537462  0.0132549597 0.9091420464 0.
  0.           0.           0.0370655319]
 [0.0011211395 0.           0.           0.           0.9988788605
  0.           0.           0.          ]
 [0.           0.0124521601 0.           0.           0.
  0.9875478399 0.           0.          ]
 [0.           0.           0.0075486887 0.           0.
  0.           0.9924513113 0.          ]
 [0.           0.           0.           0.0301375053 0.
  0.           0.           0.9698624947]]

The real transition matrix of the traffic flow from 20:00 to 21:00:
[[0.9461498775 0.0524958081 0.           0.           0.0013543145
  0.           0.           0.          ]
 [0.0235047124 0.8009173684 0.0175030237 0.0758768627 0.
  0.0821980329 0.           0.          ]
```

```
 [0.           0.0402173605 0.9126312512 0.0182548892 0.
  0.           0.028896499  0.          ]
 [0.           0.057692608  0.0183042059 0.8570960034 0.
  0.           0.           0.0669071827]
 [0.002038803  0.           0.           0.           0.997961197
  0.           0.           0.          ]
 [0.           0.0278184748 0.           0.           0.
  0.9721815252 0.           0.          ]
 [0.           0.           0.0098666259 0.           0.
  0.           0.9901333741 0.          ]
 [0.           0.           0.           0.0410157047 0.
  0.           0.           0.9589842953]]

The real transition number is
[14671.   814.     0.     0.    21.     0.     0.     0.]
The Non Time Alignment model transition number is
[14876.7550104497  1594.6163743911     0.               0.
    63.7305815415     0.               0.               0.          ]
The Time Alignment model transition number is
[14010.21532801    4168.036467264      0.               0.
    20.5244513044     0.               0.               0.          ]
```

[ ]:
```
# experiment 4: the construction of time-homogeneous continuous Markov Chain
 ↪and the calculation of Q matrix and the limit distribution and stationary
 ↪distribution

# The calculation of Q matrix
# input a list of x and a list of matrix, for each element of matrix, build the
 ↪linear model of x to the element
# return a function, the input of this function is an element of x, the output
 ↪is a matrix
def linearModelMatrix(X, matrix):
    def linearModel(x, y):
        p = np.polyfit(x, y, 1)
        f = np.poly1d(p)
        return f
    _matrix_shape = matrix[0].shape
    _matrix_len = len(matrix)
    _f_list = []
    for i in range(_matrix_shape[0]):
        _f_list.append([])
        for j in range(_matrix_shape[1]):
            _f_list[i].append(linearModel(X, [matrix[k][i][j] for k in
 ↪range(_matrix_len)]))

    def _f(x):
        _result = np.zeros(_matrix_shape)
```

```
        for i in range(_matrix_shape[0]):
            for j in range(_matrix_shape[1]):
                _result[i][j] = _f_list[i][j](x)
        return _result
    return _f


# return a Q-matrix
# begin_hour:
# n_hour_steps, the number of hours used to fit the Q matrix
# epsilon: use the epsilon that tends to 0 to calculate the Q matrix
def get_hour_Q_matrix(begin_hour, n_hour_steps, epsilon):
    trans_matrixs_list = [model.get_hour_k_step_transition_matrix(begin_hour,␣
 ↪k) for k in range(1, n_hour_steps+1)]
    phi_t_list = [i - np.eye(8) for i in trans_matrixs_list]
    phi_t_list = [phi_t_list[i] / (i+1) for i in range(n_hour_steps)]
    Phi_t = linearModelMatrix(list(range(1, 1+n_hour_steps)),
                              phi_t_list)
    return Phi_t(epsilon)




exp4_begin_hour1 = 12
exp4_fitting_hour_steps = 4
exp4_epsilon = 0.1
exp4_Q = get_hour_Q_matrix(begin_hour=exp4_begin_hour1,␣
 ↪n_hour_steps=exp4_fitting_hour_steps, epsilon=exp4_epsilon)

exp4_transition_matrix_P1 = model.
 ↪get_hour_k_step_transition_matrix(begin_hour=exp4_begin_hour1, k=1)
exp4_transition_matrix_P2 = model.
 ↪get_hour_k_step_transition_matrix(begin_hour=exp4_begin_hour1, k=2)
exp4_transition_matrix_P3 = model.
 ↪get_hour_k_step_transition_matrix(begin_hour=exp4_begin_hour1, k=3)
exp4_transition_matrix_P4 = model.
 ↪get_hour_k_step_transition_matrix(begin_hour=exp4_begin_hour1, k=4)


print("Experiment 4")
print('The construction of time-homogeneous continuous Markov Chain')
print('------------------------------------------------')
print('The Q matrix of the traffic flow:')
print(exp4_Q)
print()
print('The transition matrix of the traffic flow from 12:00 to 13:00:')
```

```
print(exp4_transition_matrix_P1)
print()
print('The transition matrix of the traffic flow from 12:00 to 14:00:')
print(exp4_transition_matrix_P2)
print()
print('The transition matrix of the traffic flow from 12:00 to 15:00:')
print(exp4_transition_matrix_P3)
print()
print('The transition matrix of the traffic flow from 12:00 to 16:00:')
print(exp4_transition_matrix_P4)
print()
```

```
Experiment 4
The construction of time-homogeneous continuous Markov Chain
--------------------------------------------------
The Q matrix of the traffic flow:
[[-0.144417628   0.1525599482 -0.0013237875 -0.0031431007  0.0024705162
  -0.0051508894 -0.0000878363 -0.0009072224]
 [ 0.0531883339 -0.4390173745  0.0492818905  0.1823882414 -0.0000486955
   0.1584489658 -0.0005061463 -0.0037352153]
 [-0.0005188432  0.0424664971 -0.1043975108  0.0395343983 -0.0000031929
  -0.002108823   0.0265823295 -0.001554855 ]
 [-0.0008582823  0.098701552   0.0186082181 -0.210926484  -0.0000067846
  -0.0035760849 -0.0003403118  0.0983981775]
 [ 0.0019016641 -0.0001069153 -0.0000056071 -0.000019649  -0.0017470169
  -0.0000218963 -0.0000000499 -0.0000005295]
 [-0.0006457253  0.0565556642 -0.0006759171 -0.0017924488 -0.000004178
  -0.0530322571 -0.0000357028 -0.0003694351]
 [-0.0000237106 -0.0004258436  0.0285481394 -0.0004166275 -0.0000000265
  -0.0000888994 -0.0275428522 -0.0000501796]
 [-0.0001291818 -0.0015974542 -0.000746743   0.0782213154 -0.0000001651
  -0.0004838838 -0.0000294138 -0.0752344737]]

The transition matrix of the traffic flow from 12:00 to 13:00:
[[0.8628960209 0.1346299223 0.           0.           0.0024740568
  0.           0.           0.          ]
 [0.0474172113 0.605458175  0.0445459269 0.1589871202 0.
  0.1435915666 0.           0.          ]
 [0.           0.0392748504 0.8965508782 0.0370670199 0.
  0.           0.0271072515 0.          ]
 [0.           0.0864248955 0.0181126802 0.8031388146 0.
  0.           0.           0.0923236096]
 [0.0019033867 0.           0.           0.           0.9980966133
  0.           0.           0.          ]
 [0.           0.0510594843 0.           0.           0.
  0.9489405157 0.           0.          ]
 [0.           0.           0.0274593376 0.           0.
```

```
  0.           0.9725406624 0.          ]
 [0.           0.           0.           0.0723744777 0.
  0.           0.           0.9276255223]]

The transition matrix of the traffic flow from 12:00 to 14:00:
[[0.7623138042 0.1847490682 0.0052728528 0.0216330446 0.0061414554
  0.0198897748 0.           0.          ]
 [0.0677987061 0.3979758827 0.0668703888 0.2258401921 0.0002020008
  0.2255007311 0.0012625639 0.0145495344]
 [0.0017014602 0.0614271539 0.8073367395 0.068528109  0.
  0.0058023352 0.051812054  0.0033921483]
 [0.0037440887 0.1256394412 0.0357622782 0.6627318371 0.
  0.0127681253 0.0005133671 0.1588408624]
 [0.0051390248 0.0002266411 0.           0.           0.9946343341
  0.           0.           0.          ]
 [0.0022119927 0.0809235004 0.001999772  0.0082045067 0.
  0.9066602282 0.           0.          ]
 [0.           0.0010464882 0.0499689041 0.000999197  0.
  0.           0.9479854107 0.          ]
 [0.           0.0065160521 0.0014533184 0.1279250215 0.
  0.           0.           0.864105608 ]]

The transition matrix of the traffic flow from 12:00 to 15:00:
[[0.674189352  0.205574917  0.0125442881 0.0470066237 0.0094970208
  0.0488053886 0.0001699492 0.0022124605]
 [0.0783681027 0.2787029191 0.0802308151 0.2446519552 0.0005014991
  0.2776782803 0.003380476  0.0364859524]
 [0.0044484584 0.0802175329 0.7206368932 0.0927990785 0.0000075318
  0.0154613596 0.0762991    0.0101300457]
 [0.0093290199 0.1382555711 0.0515098875 0.5528163199 0.0000165737
  0.0324742786 0.001650819  0.2139475303]
 [0.0075608947 0.000764693  0.0000090668 0.0000366692 0.9915918533
  0.0000368229 0.           0.          ]
 [0.0058346149 0.0988559272 0.0051918161 0.0195840675 0.0000097917
  0.8696202345 0.0000644546 0.0008390935]
 [0.0000504891 0.0030118409 0.0723959652 0.0027443032 0.
  0.0001700253 0.921525186  0.0001021901]
 [0.0003143747 0.015791073  0.0044030917 0.1701360196 0.
  0.0010586779 0.0000468419 0.8082499212]]

The transition matrix of the traffic flow from 12:00 to 16:00:
[[0.5924052324 0.2040838123 0.0226166021 0.073190514  0.0124734652
  0.0866942032 0.0007523641 0.0077838067]
 [0.0823682646 0.2035027395 0.0900284241 0.2380903733 0.00084985
  0.3149627851 0.0070361077 0.0631614556]
 [0.00808961   0.0931672531 0.6248353486 0.1150581104 0.0000273877
  0.0304845357 0.1077445791 0.0205931755]
 [0.0153760066 0.1393456307 0.0671875705 0.4540410381 0.0000582113
```

```
 0.057978921  0.0040135279 0.261999094 ]
[0.0104585966 0.001404182  0.0000479768 0.0001682657 0.9877286513
 0.0001873964 0.0000004247 0.0000045066]
[0.0102732953 0.1117260678 0.0100695091 0.0329177292 0.0000358352
 0.831503949  0.0003056818 0.0031679326]
[0.0002032249 0.0058916822 0.0899669741 0.0057070502 0.0000002257
 0.0007614141 0.8970394691 0.0004299597]
[0.0011083384 0.0276299901 0.0095104934 0.2033249218 0.0000014053
 0.004148193  0.0002516592 0.7540249989]]
```

[ ]:
```
# experiment 5: the calculation of stationary distribution by Q matrix

# compute the stationary distribution of the traffic flow according to the Q␣
 ↪matrix
from copy import deepcopy
def compute_stationary_distribution_by_Q(Q):
        _Q = deepcopy(Q)
        _Q = _Q.T
        _Q[-1] = np.ones(8)
        b = np.zeros(8)
        b[-1] = 1
        return np.linalg.solve(_Q, b)


exp5_begin_hour = 8
exp5_fitting_hour_steps = 4
exp5_epsilon = 0.5
exp5_Q = get_hour_Q_matrix(begin_hour=exp5_begin_hour,␣
 ↪n_hour_steps=exp5_fitting_hour_steps, epsilon=exp5_epsilon)

exp5_stationary_distribution = compute_stationary_distribution_by_Q(exp5_Q)

print("Experiment 5")
print('The calculation of stationary distribution by Q matrix')
print('--------------------------------------------------')
print('The Q matrix of the traffic flow:')
print(exp5_Q)
print()
print('The stationary distribution of the traffic flow:')
print(exp5_stationary_distribution)
print(exp5_stationary_distribution * np.sum(_initial_traffic_amouts_list))
print("The begin traiffic amount of the traffic flow:")
print(model.current_time_traffic_amount[exp5_begin_hour])
```

```
Experiment 5
The calculation of stationary distribution by Q matrix
--------------------------------------------------
```

```
The Q matrix of the traffic flow:
[[-0.1129871796  0.1128079849 -0.0001750714 -0.0004957066  0.001904023
  -0.0007102407 -0.0000308647 -0.0003129449]
 [ 0.0319879105 -0.4721416894  0.0656971621  0.1779638607 -0.0000083202
   0.196921598  -0.0001276774 -0.0002928441]
 [-0.0001884196  0.0360562339 -0.1262041902  0.0697545931 -0.0000007084
  -0.0004609613  0.0214542247 -0.0004107722]
 [-0.0002629089  0.0954781154  0.0161999205 -0.2268261359 -0.0000016847
  -0.0002894043 -0.000074788   0.1157768858]
 [ 0.0006878843 -0.0000244755 -0.000001043  -0.0000032954 -0.0006555843
  -0.0000034205 -0.0000000056 -0.00000006  ]
 [-0.0002618247  0.0604338016 -0.0001439154 -0.0004201793 -0.0000011258
  -0.0594209578 -0.0000166558 -0.0001691429]
 [-0.0000129222 -0.0000497927  0.0425708733 -0.0000775598 -0.00000001
  -0.0000493722 -0.0423403743 -0.000040842 ]
 [-0.0000665106 -0.0002836668 -0.0002350701  0.0931763786 -0.0000000504
  -0.0002537265 -0.0000082302 -0.0923291239]]

The stationary distribution of the traffic flow:
[0.0232720826 0.085623622  0.0816109942 0.1880372292 0.06543035
 0.2808913891 0.0405895801 0.2345447529]
[  9936.5509338028  36558.9747455205  34845.6910067174  80286.8198650094
  27936.9928173364 119933.0390845458  17330.6547701383 100144.2767769294]
The begin traiffic amount of the traffic flow:
[ 16150.  36407.  39751.  68752.  15191. 104467.  42768. 103487.]
```

[ ]:
```python
# expereiment 6: the construction of non-time-homogeneous continuous Markov␣
 ↪Chain
# use linear interpolation to construct the non-time-homogeneous Q matrix
exp6_begin_hour1 = 18
exp6_begin_hour2 = exp6_begin_hour1 + 1

exp6_fitting_hour_steps = 3
exp6_epsilon = 0.1

exp6_Q1 = get_hour_Q_matrix(begin_hour=exp6_begin_hour1,␣
 ↪n_hour_steps=exp6_fitting_hour_steps, epsilon=exp6_epsilon)
exp6_Q2 = get_hour_Q_matrix(begin_hour=exp6_begin_hour2,␣
 ↪n_hour_steps=exp6_fitting_hour_steps, epsilon=exp6_epsilon)
exp6_model_Q3 = 2 * exp6_Q2 - exp6_Q1
exp6_real_Q3 =␣
 ↪get_hour_Q_matrix(begin_hour=exp6_begin_hour2+exp6_fitting_hour_steps,␣
 ↪n_hour_steps=exp6_fitting_hour_steps, epsilon=exp6_epsilon)

print("Experiment 6")
print('The construction of non-time-homogeneous continuous Markov Chain')
print('-------------------------------------------------')
```

```
print('The Q matrix of the traffic flow from 6:00 to 7:00:')
print(exp6_Q1)
print()
print('The Q matrix of the traffic flow from 7:00 to 8:00:')
print(exp6_Q2)
print()
print('The model Q matrix of the traffic flow from 8:00 to 9:00:')
print(exp6_model_Q3)
print()
print('The real Q matrix of the traffic flow from 8:00 to 9:00:')
print(exp6_real_Q3)
print()
```

```
Experiment 6
The construction of non-time-homogeneous continuous Markov Chain
--------------------------------------------------
The Q matrix of the traffic flow from 6:00 to 7:00:
[[-0.1323579182  0.1384299806 -0.000645184  -0.0024856186  0.0001948444
  -0.0029728679 -0.0000162998 -0.0001469364]
 [ 0.0566465991 -0.4967122138  0.0393446746  0.1780239307 -0.0000217236
   0.2257988689 -0.0003261035 -0.0027540323]
 [-0.0005062468  0.0768223944 -0.1822340121  0.0386286444 -0.000000603
  -0.0017874835  0.0699201994 -0.0008428927]
 [-0.0008076479  0.1408908459  0.0374215142 -0.3123837733 -0.0000010718
  -0.0028547973 -0.000331617   0.1380665472]
 [ 0.0016103748 -0.000030996  -0.0000004467 -0.0000019365 -0.0015748978
  -0.0000020979  0.             0.          ]
 [-0.0004119914  0.0672045019 -0.0003159235 -0.0012164438 -0.0000005226
  -0.0651793357 -0.0000080168 -0.0000722681]
 [-0.0000046598 -0.0002582442  0.0226198344 -0.0001352395  0.
  -0.0000162956 -0.0221984435 -0.0000069518]
 [-0.0000311275 -0.001418579  -0.0005117047  0.0972471128  0.
  -0.0001088557 -0.000011127  -0.0951657189]]

The Q matrix of the traffic flow from 7:00 to 8:00:
[[-0.0889752522  0.0898485505 -0.0003290105 -0.0014322923  0.0026507989
  -0.0016946932 -0.0000056417 -0.0000624596]
 [ 0.0390360851 -0.2963976064  0.027797752   0.1098798496 -0.0000107046
   0.1215635789 -0.0001820351 -0.0016869195]
 [-0.0003040121  0.0569365646 -0.12920341    0.0293742831 -0.0000002109
  -0.0011776309  0.0448856096 -0.0005111933]
 [-0.0004463688  0.0920610716  0.0257388082 -0.2032325886 -0.0000003359
  -0.0017286205 -0.0001831368  0.0877911709]
 [ 0.0015065122 -0.0000243998 -0.0000001848 -0.0000008951 -0.0014800447
  -0.0000009878  0.             0.          ]
 [-0.0002162505  0.0393316268 -0.0001621753 -0.0007119788 -0.000000148
  -0.0382100343 -0.0000025714 -0.0000284685]
```

```
 [-0.0000019169 -0.0001441411  0.0156457158 -0.0000661913  0.
  -0.000007448  -0.0154233122 -0.0000027064]
 [-0.0000113807 -0.0008997269 -0.0002988117  0.064278385   0.
  -0.0000442194 -0.0000043978 -0.0630198485]]

The model Q matrix of the traffic flow from 8:00 to 9:00:
[[-0.0455925861  0.0412671205 -0.000012837  -0.0003789659  0.0051067533
  -0.0004165184  0.0000050165  0.0000220172]
 [ 0.0214255711 -0.0960829989  0.0162508293  0.0417357686  0.0000003144
   0.0173282889 -0.0000379666 -0.0006198067]
 [-0.0001017775  0.0370507348 -0.0761728079  0.0201199218  0.0000001812
  -0.0005677783  0.0198510198 -0.000179494 ]
 [-0.0000850897  0.0432312972  0.0140561021 -0.0940814039  0.0000004001
  -0.0006024437 -0.0000346567  0.0375157946]
 [ 0.0014026496 -0.0000178036  0.000000077   0.0000001464 -0.0013851916
   0.0000001222  0.            0.          ]
 [-0.0000205096  0.0114587517 -0.0000084271 -0.0002075137  0.0000002265
  -0.0112407329  0.000002874   0.0000153312]
 [ 0.000000826  -0.000030038   0.0086715973  0.0000028568  0.
   0.0000013997 -0.0086481809  0.0000015391]
 [ 0.0000083662 -0.0003808748 -0.0000859187  0.0313096571  0.
   0.0000204168  0.0000023315 -0.030873978 ]]

The real Q matrix of the traffic flow from 8:00 to 9:00:
[[0.1739940564 0.0298844623 0.0000162272 0.000078933  0.0014864228
  0.0000954539 0.           0.          ]
 [0.0140913648 0.0423096407 0.0150405822 0.058447112  0.0000002873
  0.0755155583 0.0000126772 0.0001383331]
 [0.0000181309 0.0331474539 0.136771505  0.0126827013 0.
  0.0001056957 0.0228004002 0.0000296684]
 [0.000028425  0.0521904971 0.0147266315 0.0838787683 0.
  0.0001657061 0.0000123478 0.0545531798]
 [0.0021218833 0.0000022693 0.           0.           0.2034314029
  0.           0.           0.          ]
 [0.0000132029 0.0243911187 0.0000130844 0.000063646  0.
  0.1810745036 0.           0.          ]
 [0.           0.0000062338 0.0050208552 0.0000028659 0.
  0.           0.2005256007 0.          ]
 [0.           0.0000825663 0.0000214085 0.037936156  0.
  0.           0.           0.1675154247]]


The construction of non-time-homogeneous continuous Markov Chain
------------------------------------------------
The Q matrix of the traffic flow from 6:00 to 7:00:
[[-0.1323579182  0.1384299806 -0.000645184  -0.0024856186  0.0001948444
  -0.0029728679 -0.0000162998 -0.0001469364]
 [ 0.0566465991 -0.4967122138  0.0393446746  0.1780239307 -0.0000217236
```

```
   0.2257988689 -0.0003261035 -0.0027540323]
 [-0.0005062468  0.0768223944 -0.1822340121  0.0386286444 -0.000000603
  -0.0017874835  0.0699201994 -0.0008428927]
 [-0.0008076479  0.1408908459  0.0374215142 -0.3123837733 -0.0000010718
  -0.0028547973 -0.000331617   0.1380665472]
 [ 0.0016103748 -0.000030996  -0.0000004467 -0.0000019365 -0.0015748978
  -0.0000020979  0.             0.          ]
 [-0.0004119914  0.0672045019 -0.0003159235 -0.0012164438 -0.0000005226
  -0.0651793357 -0.0000080168 -0.0000722681]
 [-0.0000046598 -0.0002582442  0.0226198344 -0.0001352395  0.
  -0.0000162956 -0.0221984435 -0.0000069518]
 [-0.0000311275 -0.001418579  -0.0005117047  0.0972471128  0.
  -0.0001088557 -0.000011127  -0.0951657189]]

The Q matrix of the traffic flow from 7:00 to 8:00:
[[-0.0889752522  0.0898485505 -0.0003290105 -0.0014322923  0.0026507989
   -0.0016946932 -0.0000056417 -0.0000624596]
 [ 0.0390360851 -0.2963976064  0.027797752   0.1098798496 -0.0000107046
    0.1215635789 -0.0001820351 -0.0016869195]
 [-0.0003040121  0.0569365646 -0.12920341    0.0293742831 -0.0000002109
  -0.0011776309  0.0448856096 -0.0005111933]
 [-0.0004463688  0.0920610716  0.0257388082 -0.2032325886 -0.0000003359
  -0.0017286205 -0.0001831368  0.0877911709]
 [ 0.0015065122 -0.0000243998 -0.0000001848 -0.0000008951 -0.0014800447
  -0.0000009878  0.             0.          ]
 [-0.0002162505  0.0393316268 -0.0001621753 -0.0007119788 -0.000000148
  -0.0382100343 -0.0000025714 -0.0000284685]
 [-0.0000019169 -0.0001441411  0.0156457158 -0.0000661913  0.
  -0.000007448  -0.0154233122 -0.0000027064]
 [-0.0000113807 -0.0008997269 -0.0002988117  0.064278385   0.
  -0.0000442194 -0.0000043978 -0.0630198485]]

The model Q matrix of the traffic flow from 8:00 to 9:00:
[[-0.0455925861  0.0412671205 -0.000012837  -0.0003789659  0.0051067533
   -0.0004165184  0.0000050165  0.0000220172]
 [ 0.0214255711 -0.0960829989  0.0162508293  0.0417357686  0.0000003144
    0.0173282889 -0.0000379666 -0.0006198067]
 [-0.0001017775  0.0370507348 -0.0761728079  0.0201199218  0.0000001812
  -0.0005677783  0.0198510198 -0.000179494 ]
 [-0.0000850897  0.0432312972  0.0140561021 -0.0940814039  0.0000004001
  -0.0006024437 -0.0000346567  0.0375157946]
 [ 0.0014026496 -0.0000178036  0.000000077   0.0000001464 -0.0013851916
    0.0000001222  0.             0.          ]
 [-0.0000205096  0.0114587517 -0.0000084271 -0.0002075137  0.0000002265
  -0.0112407329  0.000002874   0.0000153312]
 [ 0.000000826  -0.000030038   0.0086715973  0.0000028568  0.
    0.0000013997 -0.0086481809  0.0000015391]
 [ 0.0000083662 -0.0003808748 -0.0000859187  0.0313096571  0.
```

```
   0.0000204168  0.0000023315 -0.030873978 ]]

The real Q matrix of the traffic flow from 8:00 to 9:00:
[[0.1739940564 0.0298844623 0.0000162272 0.000078933  0.0014864228
  0.0000954539 0.           0.          ]
 [0.0140913648 0.0423096407 0.0150405822 0.058447112  0.0000002873
  0.0755155583 0.0000126772 0.0001383331]
 [0.0000181309 0.0331474539 0.136771505  0.0126827013 0.
  0.0001056957 0.0228004002 0.0000296684]
 [0.000028425  0.0521904971 0.0147266315 0.0838787683 0.
  0.0001657061 0.0000123478 0.0545531798]
 [0.0021218833 0.0000022693 0.           0.           0.2034314029
  0.           0.           0.          ]
 [0.0000132029 0.0243911187 0.0000130844 0.000063646  0.
  0.1810745036 0.           0.          ]
 [0.           0.0000062338 0.0050208552 0.0000028659 0.
  0.           0.2005256007 0.          ]
 [0.           0.0000825663 0.0000214085 0.037936156  0.
  0.           0.           0.1675154247]]
```

[ ]:
```
# experiment 6 extension: the prediction of traffic flow by Q matrix

from scipy.linalg import expm

# the number of hours to predict the traffic flow
exp6_predict_hour = 1
exp6_model_hour_P = np.clip(expm(exp6_model_Q3 * exp6_predict_hour), a_min=0,␣
 ↪a_max=None)
exp6_real_hour_P = np.clip(expm(exp6_real_Q3 * exp6_predict_hour), a_min=0,␣
 ↪a_max=None)

print("Experiment 6 extension")
print('The prediction of traffic flow by Q matrix')
print('-------------------------------------------------')
print(f'The real transition matrix of the traffic flow from␣
 ↪{exp6_begin_hour1+2}:00 to {exp6_begin_hour1+3}:00:')
print(exp6_real_hour_P)
print()
print("The real transition number is")
print(model.hourly_traffic_among_regions[exp6_begin_hour1+2][0])
print(f'The model transition matrix of the traffic flow from␣
 ↪{exp6_begin_hour1+2}:00 to {exp6_begin_hour1+3}:00:')
print(exp6_model_hour_P)
print()
print('The model transition number is')
```

```
print(model.current_time_traffic_amount[exp6_begin_hour1+2] *␣
↪exp6_model_hour_P[0])
```

```
Experiment 6 extension
The prediction of traffic flow by Q matrix
--------------------------------------------------
The real transition matrix of the traffic flow from 20:00 to 21:00:
[[1.1902904662 0.033357726  0.0002774673 0.0010574323 0.0017953357
  0.0014036288 0.0000024379 0.0000227186]
 [0.015728903  1.0463446861 0.0169433158 0.0624537109 0.0000123851
  0.0845948133 0.000213222  0.0019161767]
 [0.0002860987 0.0366663442 1.1470227279 0.0152366812 0.0000001553
  0.0015470815 0.0269973619 0.0004507621]
 [0.0004406893 0.0559415085 0.0168919393 1.090402555  0.0000002404
  0.0023828594 0.0002113229 0.0619360981]
 [0.0025628708 0.0000390806 0.0000002261 0.0000008738 1.2256029961
  0.0000011491 0.0000000016 0.0000000147]
 [0.0002121783 0.0273237087 0.000226823  0.0008643115 0.0000001149
  1.1995595319 0.0000019913 0.0000185532]
 [0.0000005567 0.0001022982 0.0059450695 0.0000420014 0.0000000002
  0.0000030148 1.222113388  0.000000884 ]
 [0.0000065306 0.0011886343 0.0003493048 0.0430700935 0.0000000028
  0.0000353583 0.0000030651 1.1835542235]]

The real transition number is
[14671.   814.     0.     0.    21.     0.     0.     0.]
The model transition matrix of the traffic flow from 20:00 to 21:00:
[[0.9558500061 0.0384612367 0.0003010828 0.00044492   0.0049893355
  0.           0.0000061118 0.0000124792]
 [0.0199714481 0.9099909127 0.0151879491 0.0381170506 0.0000524911
  0.0164168569 0.0001170793 0.0001462122]
 [0.0002755004 0.034408943  0.9271506113 0.0191957949 0.0000005764
  0.           0.0190326616 0.0001810417]
 [0.0003500441 0.0395697152 0.0132378133 0.9117104288 0.0000009265
  0.           0.0000999842 0.0352548047]
 [0.0013702154 0.0000106344 0.0000000769 0.           0.9986192917
  0.           0.0000000039 0.0000000132]
 [0.000096735  0.0108647902 0.0000792148 0.0000273711 0.0000003765
  0.9889176477 0.0000031397 0.0000107249]
 [0.000001164  0.0001240291 0.0083143363 0.0000864572 0.000000003
  0.           0.9914725507 0.0000018366]
 [0.0000074661 0.0002724971 0.0001248719 0.0294258967 0.0000000239
  0.000011504  0.0000023251 0.9701554153]]

The model transition number is
[14821.4101940969  1685.4098535761    10.6381583465    28.4877823067
    75.8628455908     0.               0.2799892895     1.2237080344]
```