本科生毕业论文（设计）

题目：一种高效求解微分黎卡提方程的快速灵活全纯嵌入式方法

姓　　名　贾智超

学　　号　17342012

院　　系　计算机学院

专　　业　信息与计算科学

指导教师　汪涛 (副教授)

2021 年 4 月 28 日

一种高效求解微分黎卡提方程的

快速灵活全纯嵌入式方法

On an FFHE-Inspired Method

for Effectively Solving Differential Reccati Equations

姓　　名	贾 智 超
学　　号	17342012
院　　系	计算机学院
专　　业	信息与计算科学
指导教师	汪涛 (副教授)

2021 年 4 月 28 日

学术诚信声明

本人郑重声明：所呈交的毕业论文（设计），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文（设计）不包含任何其他个人或集体已经发表或撰写过的作品成果。对本论文（设计）的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本论文（设计）的知识产权归属于培养单位。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日　　期：　　年　　月　　日

【摘　要】

本文提出了一种用于求解微分黎卡提（Riccati）方程的快速灵活全纯嵌入式方法。微分黎卡提方程是一类经典的二次非线性微分方程，当未知量为矩阵形式而非标量时，相当于一类带参的微分代数方程组，它被证明没有一般的初等解法。FFHE（Fast and Flexible Holomorphic Embedding）方法是一种快速灵活的全纯嵌入式计算方法的简称，它包括对参数方程的解函数进行幂级数展开、有理逼近、以及计算区间的自适应划分等主要步骤，对求解带参的微分代数方程组具有很好的效果，可以用于求解微分黎卡提方程组。本文依据以上过程，设计并实现了相应的 FFHE 方法，成功求解未知量为任意维度矩阵的带参微分黎卡提方程，并在经典 FFHE 方法的基础上提出了一些提升算法运行效率的改进方案。例如，根据前一区间最终长度确定当前区间初始长度，取高维矩阵中的单个值或离散值预测方程式的相对误差等。数值仿真结果显示，该新方法可达到极高的计算精度，并且在长区间上其计算速度要明显快于目前常用的 Lyapunov 方法及其他传统迭代方法（如龙格-库塔法），在变量为高维矩阵的情况下在计算效率和稳定性等方面该新方法优势更明显。

【关键词】　微分黎卡提方程，带参方程，Lyapunov 方法，快速灵活全纯嵌入

[ABSTRACT]

This study develops a fast and flexible holomorphic embedding method for solving differential Riccati equations. Differential Riccati equations are a kind of classical quadratic nonlinear differential equations which are equivalent to a set of parametrized differential algebraic equations When the unknown is a matrix but not a scalar. It has been proved that there is no general elementary solution for this kind of equations. FFHE is short for the“Fast and Flexible Holomorphic Embedding” method, which consists of major steps such as deriving power series expansions for solution functions of the parametrized equations, computing rational approximants, and partitioning the interval of the parameter in an adaptive manner. The method works well on solving parametrized differential-algebraic equations and can be extended to deal with differential Ricati equations. Using the process described above, this study tailors the FFHE method for solving parametrized differential Riccati equations with the unknown being a matrix of free dimension. Moreover, this study proposes some novel approaches to improve the efficiency of the classical FFHE method. For example, one can promisingly initiate the length of the current interval by the actual final length of the previous interval and estimate the relative error using a single or few values in large matrices. The experimental results show that FFHE can achieve very high accuracy with an outstanding computing speed significantly higher than the Lyapunov method and other traditional iterative methods (e.g., the Runge-Kutta method). Furthermore, FFHE enjoys even better computational efficiency and stability when the unknowns are matrices of higher dimensions.

[Keywords] differential Riccati equations, parametrized equation, Lyapunov method, FFHE

目录

一、续论

1.1 选题背景与意义

微分黎卡提（Riccati）方程又称黎卡提方程[1]，是一类非线性微分方程，也是最优控制领域中一类经典数值计算问题。尽管其结构简单，拥有与一元二次方程组相似的结构，但是，它被证明除少数情况外没有一般的初等解法。自黎卡提方程被提出以来，越来越多的理论与实际问题[2] 都与计算求解黎卡提方程有关，因此，设计有效计算方法求解黎卡提方程也成为了自动控制领域普遍关注的重要问题之一。现有文献已提出了许多特例解法，但未能从根本上解决该问题，于是，设计高效计算方法获得它的近似解是很有意义的研究工作。

本文采用 FFHE 方法，全称为快速灵活全纯嵌入方法[3]。它是一种新颖高效的数值计算方法，能有效求解大规模代数方程组和微分代数混合方程组，它包含快速幂级数展开、构造有理逼近得到数值解、灵活划分参数区间、根据具体问题选用特定参数和优化策略等重要组成，已有效地应用于大规模电力系统潮流计算、电压稳定分析、机械臂容错型运动规划等诸多工程计算问题[3]。因此，本文旨在根据 FFHE 思想设计求解微分黎卡提方程的计算方法，相较于求解该方程的其它传统方法和较为先进的专门算法，期望在计算精度和速度等方面获得更优的结果，形成一套用于高效求解微分黎卡提方程的新颖 FFHE 计算方法。

本文设计的新颖 FFHE 计算方法有如下特点。首先，它跳出传统迭代法和当下较为流行的计算方法表现出的在参数区间上“碎步递进”的计算方式，而是采用自动划分区间且在每个区间内独立构造近似解。这样做的好处是可以根据给定的误差界自动计算满足误差要求的最长有效区间，而且因构造有理近似函数通常可以在较长区间上得到满意的近似函数，比上述“碎步递进”方式的步长要长得多。因此，利用 FFHE 方法求解黎卡提方程时，可以将计算精度设置到极高，同时在同等计算精度要求下其在计算速度方面具有明显优势。

1.2 国内外研究现状

考虑到微分黎卡提方程没有初等解法，虽然目前有不少求解方法，但都存在一些局限性。它们的共性是先求某些特定类型的黎卡提方程的特解，再通过初等变换和积分等方式求得该方程的通解[4;5;6]。而为求一般微分黎卡提方程得解，通

常需要采用一些具有普适性的方法，由于不存在初等解法，通常需要针对性强且高效的数值计算方法来求解。例如，一些方法是根据哈密尔顿方法而设计的，还有一种较新的 Lyapunov 方程方法[7]，利用到代数 Riccati 方程和微分 Lyapunov 方程的解来求解微分 Riccati 方程[8]。它与 Anderson-Moore 方法的过程比较相似[9]，但避免了后者的局限性和较长区间内可能会出现数值计算困难等缺点。经验证，它的计算效果要优于 Anderson-Moore 方法，计算精度更高。同时，它的计算速度要快于 Davidson-Maki 方法等若干方法[10]。但是，Lyapunov 方程方法仍是基于“碎步递进”的思想，没法克服传统迭代方法的弊端。

1.3 论文结构与章节安排

本文共分为六章，各章节内容安排如下：

第一章是续论，简要说明了本文章的选题背景与意义，以及国内外研究现状和相关工作。

第二章为问题描述，介绍了待求解的微分黎卡提方程的问题描述。

第三章为算法步骤描述，设计了求解微分黎卡提方程的新 FFHE 方法。

第四章为数值实验，在不同参数设置和实现方式等条件下，利用 FFHE 方法求解数值算例，展示实验结果，并与 Lyapunov 方程方法等对比分析实验结果。

第五章是总结。

二、问题描述

本文讨论的微分黎卡提方程通常描述如下：

$$\begin{aligned} &\dot{P}(t) = A^T P(t) + P(t)A + Q - P(t)SP(t), \qquad t \in [0, T], \\ &P(T) = F. \end{aligned} \tag{2.1}$$

在上述黎卡提方程中，未知量或待求量

$$P(t) \in \mathbb{R}^{n \times n}. \tag{2.2}$$

是一个与参数 t 相关的 n 维矩阵，而 A, Q, S 均为常量，$\dot{P}$ 是关于参数 t 的导数。这里，F 为 $P(t)$ 在 $t = T$ 时的值，可视为初始值。矩阵方程展开后，相当于一个包含 $n \times n$ 个方程的方程组，需要在任给参数 t 的条件下，求解出 $n \times n$ 个待求数值，从而得到 $P(t)$ 的数值解。

三、方法设计

根据 (2.1) 问题描述中给定的微分黎卡提方程形式及其初始值，一种快速灵活全纯嵌入式方法（FFHE 方法）的计算过程主要分四步：

第一步：对于待求解的矩阵 $P(t)$ 中的每个分量数值，以参数 $t = T$ 为初始状态，设定关于 $(T - t)$ 的最高次数为 q 的幂级数展开式，并将这些展开式全部带入原方程组中，逐次建立关于展开式中待定系数满足的方程组，最终解出每个幂级数展开式的各阶系数。

第二步：利用已得到的待求解矩阵 $P(t)$ 的幂级数展开式，构造有理近似函数，得到关于 $(T - t)$ 的有理函数表达式。

第三步：设定原方程的最大相对误差阈值，并根据这阈值自适应地搜索当前区间的最佳终止状态下的参数 T'，划分当前区间（长度为 $T - T'$），并计算矩阵 $P(t)$ 在该区间终点处的值，作为下一区间的起始点处解值，T' 作为下一区间起始点。

第四步：在新的起始状态参数 $t = T'$ 下重复第一步到第三步，直至区间划分到终止状态，当 $T' = 0$ 时便可终止计算。

3.1 幂级数展开

首先，设定常量矩阵中各个常数的符号表达形式:

$$A = (a_{ij})_{n\times n}, \quad Q = (b_{ij})_{n\times n}, \quad S = (s_{ij})_{n\times n}. \tag{3.1}$$

对于待求解矩阵 $P(t)$，共有 $n \times n$ 个需要幂级数展开的未知分量，设初始状态下的参数为 t_0，级数展开的最高次数为 q（$q \to \infty$）。而且，$P(t)$ 的幂级数展开式形式为：

$$P(t) = (p_{ij})_{n\times n} = \left(\sum_{k=0}^{q} p_{ijk} \cdot (t - t_0)^k \right)_{n\times n}. \tag{3.2}$$

对 t 求导后，相应地有 $\dot{P}(t)$ 的幂级数展开式形式为：

$$\dot{P}(t) = (\dot{p}_{ij})_{n\times n} = \left(\sum_{k=1}^{q} k \cdot p_{ijk} \cdot (t - t_0)^{k-1} \right)_{n\times n}. \tag{3.3}$$

这里，p_{ijk} 为函数 $p_{ij}(t)$ 的幂级数展开式系数，其中 p_{ij0} 由初始状态下待求解矩阵 $P(t)$ 的数值决定，是已知的，因此，共有 $n \times n \times q$ 个待定系数。理论上，幂级数展开式系数需要由低阶到高阶逐次计算，阶数越高的系数通常其值越小，对数值计算结果的影响也越小。因此，相当于在最高阶数达到某阶数 q 时，舍弃更高阶系数一般对计算结果影响较小。所以，当 q 越大，幂级数展开的结果越精确，同时要求解的系数的数目也越多。所以，q 的取值需兼顾数值结果精度和计算速度。

将 $P(t)$ 和 $\dot{P}(t)$ 的幂级数展开式代入微分黎卡提方程，得到一个 $n \times n$ 阶的矩阵等式，相当于一个由 $n \times n$ 个方程组成的方程组。根据方程组逐阶计算展开式系数，以矩阵 $P(t)$ 中任意一项 p_{ij} 对应的位置为例，该方程为：

$$\begin{aligned}\sum_{k=1}^{q} k\, p_{ijk}\,(t-t_0)^{k-1} =& \sum_{c=1}^{n}\left(a_{ci}\sum_{k=0}^{q} p_{cjk}(t-t_0)^k\right)+\sum_{c=1}^{n}\left(a_{cj}\sum_{k=0}^{q} p_{ick}(t-t_0)^k\right)\\ &-\sum_{d=1}^{n}\left(\sum_{c=1}^{n}\left(s_{cd}\sum_{k=0}^{q} p_{ick}(t-t_0)^k\right)\sum_{k=0}^{q} p_{djk}(t-t_0)^k\right)+b_{ij}.\end{aligned} \tag{3.4}$$

对矩阵 $P(t)$ 中其他项相应地建立起这样的方程，得到由 $n \times n$ 个方程组成的方程组。等式两边化简后，取方程中的任给次数 k 的所有项（忽略其它项），建立关于次数为 k 的所有项的等式（$\delta_k = 1, k = 0; \delta_k = 0, k = 1, 2, 3, ...$）：

$$\begin{aligned}(k+1)p_{ij(k+1)}(t-t_0)^k =& \sum_{c=1}^{n} a_{ci}p_{cjk}(t-t_0)^k+\sum_{c=1}^{n} a_{cj}p_{ick}(t-t_0)^k+b_{ij}\delta_k\\ &-\sum_{r=0}^{k}\sum_{d=1}^{n}\Big(\sum_{c=1}^{n} s_{cd}\,p_{icr}(t-t_0)^r\Big)p_{dj(k-r)}(t-t_0)^{k-r}.\end{aligned} \tag{3.5}$$

省去参数 t 的幂次，得到展开式系数满足的等式：

$$(k+1)p_{ij(k+1)} = \sum_{c=1}^{n} a_{ci}p_{cjk}+\sum_{c=1}^{n} a_{cj}p_{ick}+b_{ij}\delta_k-\sum_{r=0}^{k}\sum_{d=1}^{n}\left(\sum_{c=1}^{n} s_{cd}\,p_{icr}\right)p_{dj(k-r)}. \tag{3.6}$$

这里，各角标取值范围为

$$i = 1, 2, ..., n. \qquad j = 1, 2, ..., n. \qquad k = 0, 1, 2, ..., q-1. \tag{3.7}$$

在上面，p_{ij0} 是常数项系数，它为 $P(t)$ 在初始状态下的已知解值。首个划分区间的初始状态下的参数为 T，我们有 $P(T) = F$，此区间内方程组幂级数展开式的常数项系数 p_{ij0} 均为矩阵 F 中的对应数值。之后，每个区间的初始状态数值由

前一个区间的终止状态数值决定，区间内方程组的幂级数展开式的常数项系数对应于该区间初始状态处矩阵中相应的解值。

已知常数项 p_{ij0} 的值，接着利用 $n \times n \times q$ 个关于系数的等式，求解相同数目的待定系数 p_{ijk}，$i = 1, 2, ..., n$，$j = 1, 2, ..., n$，$k = 1, 2, ..., q$。根据式 (3.6)，高阶项系数可由比当前阶数更低的系数计算得出，除常数项外的其他系数的计算顺序为：

$$p_{cdm}\ (c, d = 1, 2, ..., n; m = 0, 1, ..., k-1) \quad \rightarrow \quad p_{ijk}\ (k \geq 1). \tag{3.8}$$

在式 (3.8) 中，计算目标系数值所依赖的一系列系数应满足 $c = i$ 或 $d = j$，不满足这条件的其余系数是不会被用到的。因此，计算任意目标系数值 p_{ijk} 时，需要用到的低次项（包括常数项）系数个数为 $(2n-1)k$。依照式 (3.8) 给出的计算顺序，最终可以逐次算出所有幂级数展开式中的各阶系数。

3.2 构造数值解

3.2.1 帕德逼近

为提高数值计算结果的精度，待求解矩阵 $P(t)$ 的幂级数展开式的最高次数一般会设置为某个比较大的数值，使得展开式为含高次项的复杂多项式。经实验验证，一般来说，当 $(t - t_0) > 1$ 时，幂级数展开式中高次项可能达到很大的值，与低次项的值产生巨大差距，可能对收敛域内数值计算结果的精度产生一定影响。而结合所得的幂级数展开式，通过有理逼近的方法，利用两个更低次的多项式的商来构造近似函数，可以在收敛域内达到更好的计算精度。这样构造的有理函数通常称为帕德近似函数[11]，具有下面的形式：

$$\sum_{k=0}^{q} p_{ijk}(t - t_0)^k = \frac{\sum_{k=0}^{m} \alpha_{ijk}(t - t_0)^k}{\sum_{k=0}^{n} \beta_{ijk}(t - t_0)^k}, \quad q = m + n. \tag{3.9}$$

这里，α_{ijk} 和 β_{ijk} 是用来构造帕德逼近的两个多项式中的待定系数。在式 (3.9) 中，左端展开式的最高次数为 q，可以在等式两端同时乘以分母多项式，并依照次数 $0, 1, ..., q$ 逐次建立多项式系数方程，共得到 $(q+1)$ 个方程式；此外，方程组中共有包括所有 α_{ijk} 和 β_{ijk} 在内的 $(q+2)$ 个待定系数。因此，当设定分母多项式的常数项 $\beta_{ij0} = 1$ 时，可以算得所有系数值。

式 (3.9) 对于 m 和 n 的实际数值没有限制。一般来说，为了使得帕德逼近后得到分子分母中多项式中的最高次项的次数尽可能小，从而更好地提升有理逼近的

计算精度，通常取 m 和 n 均为 q 的一半，m 向上取整，n 向下取整。在这种条件下，可以根据阶数为 $m+1, m+2, ..., m+n$ 的项的系数，建立起由 n 个方程组成的方程组，其可转化为如下线性方程组求解：

$$\begin{bmatrix} p_{ij(m-n+1)} & p_{ij(m-n+2)} & \cdots & p_{ijm} \\ p_{ij(m-n+2)} & p_{ij(m-n+3)} & \cdots & p_{ij(m+1)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{ijm} & p_{ij(m+1)} & \cdots & p_{ij(m+n-1)} \end{bmatrix} \begin{bmatrix} \beta_{ijn} \\ \beta_{ij(n-1)} \\ \vdots \\ \beta_{ij1} \end{bmatrix} = - \begin{bmatrix} p_{ij(m+1)} \\ p_{ij(m+2)} \\ \vdots \\ p_{ij(m+n)} \end{bmatrix}.$$

求得系数 $\beta_{ijk}(k = 1, 2, ..., n)$ 后，代入根据次数为 $1, 2, ..., m$ 的项的系数建立的 m 个方程式：

$$\alpha_{ijk} = \sum_{r=0}^{k} p_{ijr}\beta_{ij(m-r)}, \qquad k = 0, 1, 2, ..., m. \tag{3.10}$$

这样便可求得系数 $\alpha_{ijk}(k = 0, 1, ..., m)$，也就得到有理近似函数的分子分母中两个多项式的全部系数。从理论上来说，帕德逼近可以逼近和拟合任意阶的多项式，当分子与分母多项式的最高次数分别为 m 和 n 时，逼近的误差为 $\mathcal{O}((t-t_0)^{m+n+1})$。由于待求解矩阵 $P(t)$ 的幂级数展开式的最高次数为 q（即只展开到 q 次项，忽略更高次项），于是，它的幂级数展开式的误差为 $\mathcal{O}((t-t_0)^{q+1})$。因此，在有理逼近过程中，设置分子与分母多项式的最高次数 m 与 n 的和为 q，可以将这一过程的误差恰好控制在 $\mathcal{O}((t-t_0)^{q+1})$，不会增加整个计算过程中的误差。

3.2.2 计算行列式比

上述数值解的构造方法是通过计算分子与分母的两个多项式的全部系数，以得到幂级数展开式的帕德逼近式，来计算幂级数展开式在某一参数的状态下的拟合数值外。除此之外，还有另外一种构造数值解的方式，可以通过计算两个矩阵的行列式值，而且计算它们的比值来直接计算数值解值。设幂级数展开式是关于 $(t-t_0)$ 展开，令 $\Delta t = t - t_0$，所构造的两个矩阵分别为：

$$L_{ij}^{m,n}(\Delta t) = \begin{bmatrix} p_{ij(m-n+1)} & p_{ij(m-n+2)} & \cdots & p_{ij(m+1)} \\ p_{ij(m-n+2)} & p_{ij(m-n+3)} & \cdots & p_{ij(m+2)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{ijm} & p_{ij(m+1)} & \cdots & p_{ij(m+n)} \\ \sum_{k=n}^{m} p_{ij(k-n)}(\Delta t)^k & \sum_{k=n-1}^{m} p_{ij(k-n+1)}(\Delta t)^k & \cdots & \sum_{k=0}^{m} p_{ijk}(\Delta t)^k \end{bmatrix},$$

$$M_{ij}^{m,n}(\Delta t)=\begin{bmatrix} p_{ij(m-n+1)} & p_{ij(m-n+2)} & \cdots & p_{ij(m+1)} \\ p_{ij(m-n+2)} & p_{ij(m-n+3)} & \cdots & p_{ij(m+2)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{ijm} & p_{ij(m+1)} & \cdots & p_{ij(m+n)} \\ (\Delta t)^n & (\Delta t)^{n-1} & \cdots & 1 \end{bmatrix}.$$

这里，m 和 n 的取值与之前相同。根据所构造的矩阵 L 和 M，可以计算出在任意参数状态下的解函数的近似解：

$$\sum_{k=0}^{q} p_{ijk}(t-t_0)^k=\sum_{k=0}^{q} p_{ijk}(\Delta t)^k=\frac{det(L_{ij}^{m,n}(\Delta t))}{det(M_{ij}^{m,n}(\Delta t))}. \tag{3.11}$$

其中，$det(L)$ 和 $det(M)$ 分别为矩阵 L 和 M 的行列式值（通常在上述行列式的计算过程中需要事先对两个矩阵进行预处理，以提升数值稳定性）。利用这种基于矩阵行列式的方法计算解函数的近似值，与之前利用两个多项式的比构造的有理函数计算解函数近值值的方式存在区别，又称为离散化有理逼近。经实验，当幂级数展开式的最高次数 q 相同时，两种方式得到的数值解结果是一致的。当 q 较大并超出这个范围时，多项式比的形式的帕德逼近方式在实验中会失效，而离散化有理逼近方式仍然可以完成有效的计算。

3.3 区间自适应划分

3.3.1 自适应划分策略

在本文研究的微分黎卡提方程求解问题中，待求解矩阵 $P(t)$ 的参数 t 的区间范围为 $[0,T]$。在首个区间内，从参数 $t=T$ 开始向 $t=0$ 的方向进行区间划分，依次对方程组完成幂级数展开和有理逼近的步骤，并开始自适应区间划分。

自适应区间划分的总体步骤为，通过设定合理的区间长度变化策略，以及适当的区间长度增长率和缩减率，在每次改变区间长度后，判断区间在当前长度下的终止点的相对误差是否超过设定的误差阈值。如果相对误差在阈值之内，则区间划分完成；否则继续通过相应策略改变区间长度，判断相对误差。

设置首个区间长度的初始值为整个区间的总长度 T，并设置参数 $\alpha=0.1$ 为首区间的缩减率。在确定首个区间的长度时，如果其终止点的相对误差大于阈值，则每次利用缩减率 α 乘以其当前长度作为新的区间长度，直至其终止点的相对误差达到阈值以内。一般来说，在整个区间的总长度 T 较大以及设定一个正常的相对误差阈值的情况下，划分区间的长度要远远小于区间总长度，因此，将首个区

间的缩减率 α 设为较小的 0.1，可以使区间长度成数量级地缩减，快速达到一个合适值。

首个区间划分完成后，之后区间的划分步骤为：设置增长率 $\beta = 2$，以前一个区间的划分长度的 β 倍作为当前区间长度的初始值；再设置新的缩减率 $\gamma = 0.6$，如果当前区间终止点的相对误差大于阈值，则每次利用缩减率 γ 乘以其当前长度作为新的区间长度，直至其终止点的相对误差达到阈值以内。这样设计的原因是：由于待求解矩阵 $P(t)$ 中的数值是随着参数 t 连续变化的，而区间长度一般变化不大，相邻区间的数值变化规律也相差不大，所以数值近似效果较为接近，在相同的相对误差阈值的条件下，划分的区间长度也较为接近。因此，可以根据前一个区间长度确定当前区间长度的大致范围，设定其在前者的两倍以内，并通过一个较大的缩减率 γ 来缓慢缩减当前区间长度，使得当前区间长度遍历前一区间长度的 2 倍，1.2 倍，0.72 倍等，最终划分得到一个合适区间长度值。与直接利用整个区间总长度值进行缩减相比，当前策略极大地减小了区间划分长度的最大范围，在相同的区间划分与误差判断次数下，对区间长度有更精准的划分。当区间划分到接近整个区间的终止点时，即当前区间的确定划分终止点超出整个区间范围，则直接以整个区间的终止点作为当前区间的终止点，自适应区间划分过程完毕。

3.3.2 误差判断

在区间自适应划分过程中，需要多次判断区间终止点的误差是否在阈值以内，来决定是否继续调整区间划分长度。根据给定的微分黎卡提方程形式及待求解矩阵 $P(t)$ 在初始状态下的值，无法求得 $P(t)$ 在任意点处的精确值。因此，只能利用黎卡提方程式两端作差，取矩阵范数来计算相对误差，用其估计数值结果的误差。

在误差判断过程中，需要计算矩阵 $P(t)$ 及其导数 $\dot{P}(t)$ 中的全部数值，以代入黎卡提方程中计算相对误差，也从两个方面增加了计算量。一方面，需要计算 $\dot{P}(t)$ 的数值结果，可以通过两种方式实现：（a）对其进行类似 $P(t)$ 的幂级数展开和有理逼近来进行拟合，或者（b）利用有限差分的方式对其进行估计。经实验，后一种方式尽管计算量较小，但计算结果的精确度随着矩阵 $P(t)$ 维数和复杂度增加而导致数值稳定性不如前一种方式，于是，在本文数值实验中，一律采取前一种方式。另一方面，需要计算 $P(t)$ 和 $\dot{P}(t)$ 中的全部数值，可以采取计算矩阵中的某个固定位置的相对误差值，并通过其估计矩阵的相对误差范数的上限的方式，替代计算矩阵中的全部数值，大大缩减计算量。例如，任取 n 维矩阵中第 i 行第 j 列的位置，则只需计算矩阵 $P(t)$ 中的第 i 行以及第 j 列的全部数值（共 $(2n-1)$ 个），以及矩阵 $\dot{P}(t)$ 中的第 i 行第 j 列的数值，以估计相对误差，计算量缩减为原来的

$\frac{1}{n}$（从 $2n^2$ 个数值缩减到 $2n$ 个数值）。实验中设定矩阵的相对误差范数的上限为位置 $(1,1)$ 的相对误差值的 $2n$ 倍。

四、数值实验

在数值实验过程中，所有的常数矩阵都是固定或随机生成的。这里，A 为随机生成的每个元素的数值在 $[-10, 10]$ 之间的矩阵，S 为随机生成的每个元素的数值在 $[-200, 200]$ 之间的对称矩阵，比如当 $n = 5$ 时所生成的矩阵为

$$A = \begin{bmatrix} -7 & 2 & -6 & -7 & 0 \\ 1 & -6 & -9 & -6 & -8 \\ -8 & -5 & 4 & 7 & -2 \\ 2 & 3 & -4 & -8 & 7 \\ -6 & 6 & 1 & -10 & 6 \end{bmatrix}, \quad S = \begin{bmatrix} -5 & 19 & 22 & 20 & -12 \\ 19 & -106 & -107 & -102 & 56 \\ 22 & -107 & -113 & -106 & 60 \\ 20 & -102 & -106 & -100 & 56 \\ -12 & 56 & 60 & 56 & -32 \end{bmatrix}.$$

另外，Q 为单位矩阵。初始状态下的矩阵 $P(T)$ 的对角线元素的数值为为 0.01，其余元素的数值均为 0。整个区间为 $[0, T]$，由 $t = T$ 开始，划分到 $t = 0$ 时结束。此外，当 $n = 20, 35$ 时，随机生成的矩阵见附录。

由于需进行对比实验展示 FFHE 方法的计算效果，选择 Lyapunov 方程方法做对比，而且以步长为 0.0001 的龙格-库塔法作为基准，进行对比试验。为评估各方法计算效果，相对误差的计算公式为

$$error_{rel.} \doteq \|P(t) - P_{exact}(t)\|_1 / \|P_{exact}(t)\|_1,$$

这里，$P(t)$ 为 FFHE 或 Lyapunov 方程方法的计算结果，$P_{exact}(t)$ 为龙格-库塔法的计算结果（基准）。在作图过程中，需要用到区间上紧密分布的各点处的数值解值，因此，设定 FFHE 方法中相邻点间的最小间隔为 0.001，并在每次划分区间后保存区间内每点处的数值解值。

下面共进行五项实验，前三项是与设置合适的参数以便最大化 FFHE 方法性能的实验，在这三项实验中，如无特殊说明（即相关参数值非对比实验中要进行改变的量或对对比实验效果无影响），那么，默认矩阵的维数为 $n = 5$，初始状态下的参数 $T = 1$，幂级数展开式最高次数 $q = 21$，有理逼近方式采用离散化有理逼近，最大相对误差阈值 $error\_max = 10^{-5}$。后两项为对比 FFHE 方法与 Lyapunov 方程方法性能而设计的实验，在这两项实验中，尽可能采用在相同数量级的相对误差设定下对比算法运行时间，比较两种方法的性能。所有实验均在 matlab 环境下运行，所统计的运行时间为 matlab 代码的实际运行时间。

在该算例中，以待求解矩阵 $P(t)$ 的五个对角元 $p_{11}, p_{22}, p_{33}, p_{44}, p_{55}$ 为例，这五个元素在区间 $t \in [0,1]$ 上精确值如图4-1所示。

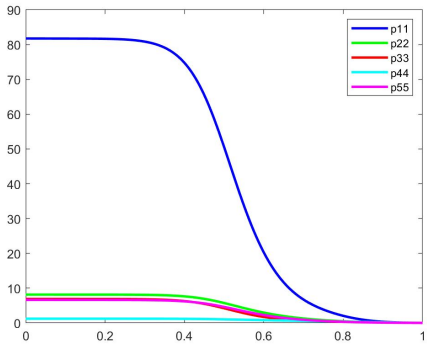

图 4-1 矩阵 $P(t)$ 中五个对角元的精确解曲线

4.1 限制展开式最高次数

下面分别令幂级数展开式最高次数 $q = 11, 21, 31$ 分别进行实验，对比 q 的取值对计算结果的影响。在这样的设置下，相对误差随参数 t 变化如图4-2所示，其中虚线为区间划分处。

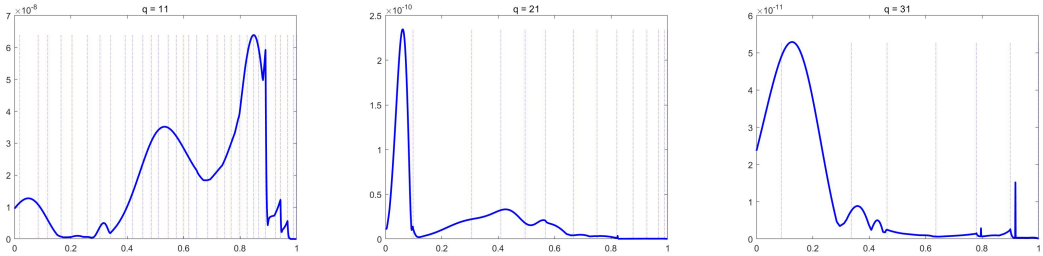

图 4-2 当 $q = 11, 21, 31$ 时各方法计算结果的相对误差

观察图4-2与表4.1可知，当 q 增大时，数值计算结果的相对误差在减小，划分区间数在减少，即划分后各区间的平均长度在增加，说明计算结果越来越精准。但是，当 q 增大时，计算复杂度增加，但需要计算的区间数减少。所以，在两方面因素的共同作用下，幂级数展开式的总计算时间维持平稳。

最高次数 q	11	21	31
划分区间数	32	13	7
展开式计算时间（秒）	0.10	0.11	0.12

表 4.1 当 $q = 11, 21, 31$ 时划分区间数和展开式计算时间

理论上，随着 q 增大，区间数目的减少倍数会先高后低，但始终维持在较高

水平，即 q 增大若干倍时，区间数目的减少倍数会远大于 q 的增大倍数。当 q 达到较高值时，最终会出现区间数目维持平稳，而且再增大 q 是区间数目几乎不变。另外，随着 q 增大，幂级数展开式的计算复杂度和计算时间是非线性增加的。因此，尽可能选取可接受的范围内较大的幂级数展开式最高次数 q（尽可能位于计算时间增加倍数和区间数减少倍数的临界值）对减少计算时间更有利，同时也会有更高的计算精度。

4.2 数值解构造方法

本节对比分别分析采用离散化有理逼近、帕德逼近、直接利用幂级数展开式构造多项式近似等这些不同数值解的构造方式对计算结果的影响。在实际实验过程中，由于帕德逼近方式下的相对误差达不到最大阈值 $error\_max = 10^{-5}$，因此，需要调整对这三种方式对比的实验设计，将这项实验分为两个子实验。一是，在原最大相对误差的阈值条件下，先对比离散化有理逼近和多项式近似的两种方式，分别令 $q = 11, 21, 31$ 时对比数值计算效果。二是，在设置 $error\_max = 10^{-3}$ 和 $q = 5$ 时，对比这三种方式的计算效果（不过，这样设置几乎已达到帕德逼近可以有效运行的极限）。为更严格地控制相对误差，这项对比实验需计算黎卡提方程的解函数在完整区间上的相对误差，而非只计算某离散点处的相对误差再用它来代表整个区间上的相对误差。

4.2.1 第一组实验

第一组实验结果如图4-3-4-5以及表4.2所示，离散化有理逼近方式带来的相对误差是小于多项式近似的结果；同时，前者的区间划分数目也少于后者。且当 q 越大时，区间划分数的差异越明显。这些观察表明在幂级数展开式最高次数较大时，有必要使用离散化有理逼近的方式以提高数值解的计算精度，也与理论上利用有理逼近来降低算式最高次数的目的相吻合。

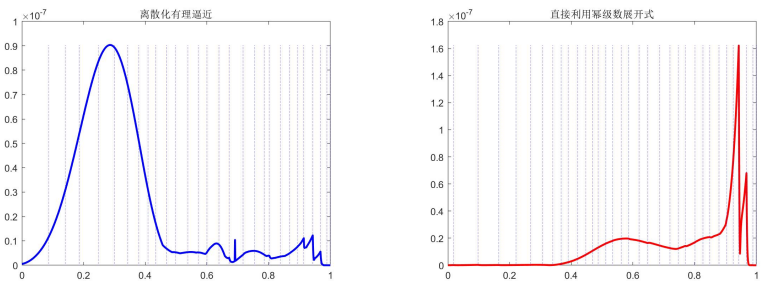

图 4-3 $q = 11$ 时各方法计算结果的相对误差

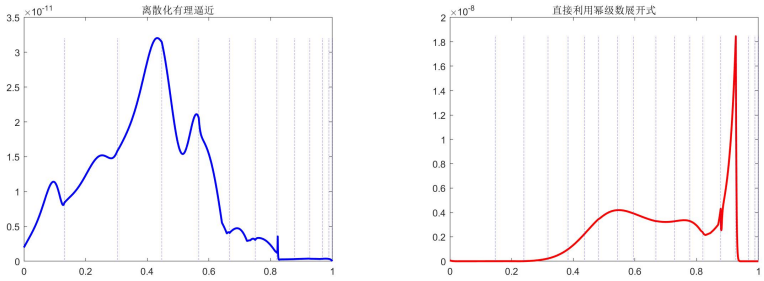

图 4-4 $q=21$ 时各方法计算结果的相对误差

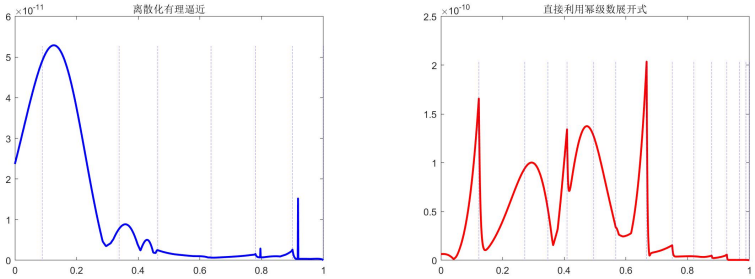

图 4-5 $q=31$ 时各方法计算结果的相对误差

最高次数 q	11	21	31
离散化有理逼近划分区间数	28	12	7
直接利用幂级数展开式划分区间数	33	17	14

表 4.2 $q=11,21,31$ 时两种方法所需划分区间数

4.2.2 第二组实验

第二组实验结果如图4-6以及表4.3所示，在低次幂级数展开以及低相对误差阈值的设置下，三种方法的划分区间数目是大致相等的，但多项式近似带来的相对误差要低于其他两种有理逼近方式，因为当幂级数展开式最高次数 q 较低时，利用多项式近似就能够达到较好的逼近效果，不必采取有理逼近来降低算式最高次数，有理逼近反而会带来低次下的较大误差 $O((t-t_0)^{q+1})$，对数值计算精度造成不利的影响。此外，可以看到两种有理逼近方式带来的相对误差和计算效果是大致相同的，但离散化有理逼近的适用性要远远好于帕德逼近。

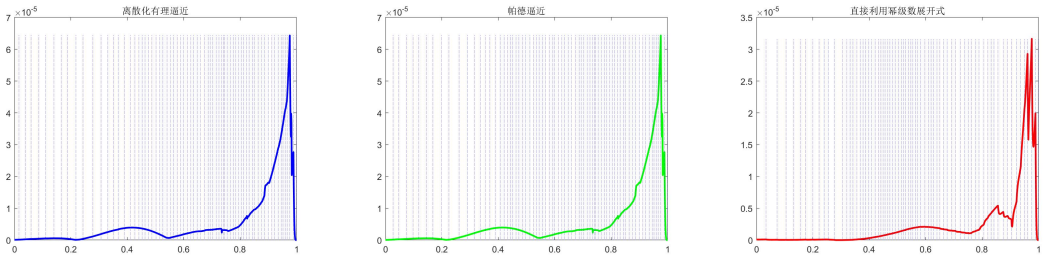

图 4-6 当 $q=5$，$error\_max=10^{-3}$ 时各方法计算结果的相对误差

离散化有理逼近	71
帕德逼近	73
多项式近似	71

表 4.3 当 $q=5$，$error\_max=10^{-3}$ 时三种方法所需划分区间数

4.3 调整相对误差阈值

本节分别取最大相对误差阈值 $error\_max = 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}$ 进行对比实验，分析 $error\_max$ 的值对数值计算效果的影响。这些条件下的相对误差随参数 t 的变化如图4-7以及表4.4所示。

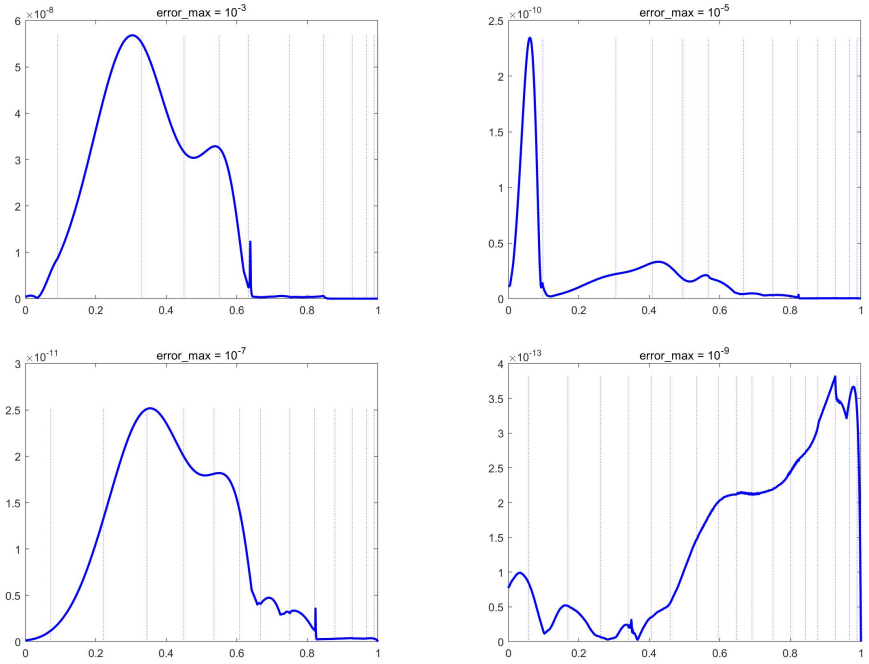

图 4-7 当 $error\_max = 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}$ 时各方法计算结果的相对误差

$error\_max$	10^{-3}	10^{-5}	10^{-7}	10^{-9}
划分区间数	11	13	14	18

表 4.4 当 $error\_max = 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}$ 时各方法所需的划分区间数

可以看出，划分区间数目随着最大相对误差阈值减小而增加，但当阈值和实际相对误差以指数倍快速减小时，划分区间数目增加量很小。这表明通过缩小最大相对误差阈值 $error\_max$，可以将相对误差控制在一个极小的数量级，同时不会显著增加划分区间数目，也不会显著增加计算量；当要求计算结果更高精度时本文提出的新方法有更好的数值计算效果。

4.4 调整问题规模

本节分别设置待求解矩阵 $P(t)$ 的维数 $n = 5, 20, 35$ 进行对比实验，分析在不同的 n 的取值下，FFHE 方法和 Lyapunov 方程方法的数值计算效果。

4.4.1 矩阵维数 $n = 5$

当 $n = 5$ 时，Lyapunov 方程方法具有很好的数值计算效果，因此，实验中控制两种方法的最大相对误差大致相等，对比它们的运行时间来比较计算效果。在控制最大相对误差的过程中，取基于传统迭代思想的 Lyapunov 方程方法的步长为 0.001。根据实验结果，尽管 Lyapunov 方程方法的步长取值并不影响其数值计算精度，但步长需要取足够小，以保证可以找到整个区间内任意参数状态下与之足够邻近的计算点，并以其计算数值作为该参数状态下的数值估计。通过调整 FFHE 方法的最大相对误差阈值 $error\_max$ 以及幂级数展开最高次数 q，可以控制 FFHE 方法的实际相对误差与 Lyapunov 方程方法大致相等，并在其实现过程中采用离散化有理逼近的方式构造数值解。其中 FFHE 方法的其余参数为 $q = 21$，$error\_max = 10^{-8}$。这两种方法计算结果的相对误差随参数 t 的变化如图4-8所示。

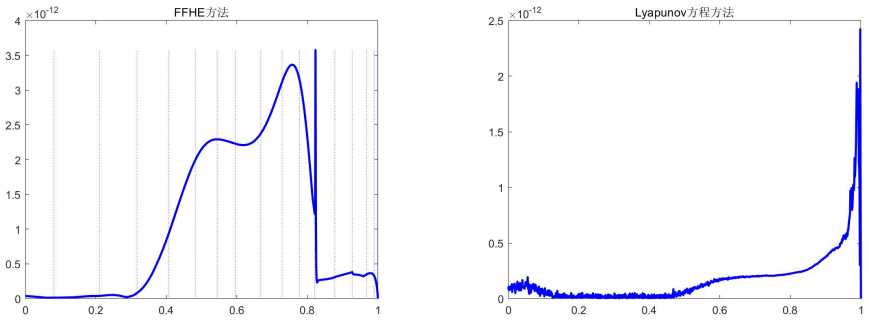

图 4-8 当 $n = 5$ 时各方法计算结果的相对误差

FFHE	0.15
Lyapunov	0.33

表 4.5 当 $n = 5$ 时各方法的运行时间（秒）

结合表4.9中结果可以看出，在矩阵维数 n 较小时，两种方法都能达到很高的数值计算精度，但 FFHE 方法的计算速度要明显快于 Lyapunov 方程方法。

4.4.2 矩阵维数 $n = 20$

当 $n = 20$ 时，Lyapunov 方程方法的数值计算结果精度很糟糕（误差在 0.014 左右），远远低于 FFHE 方法能够达到的计算精度。其中，FFHE 方法参数设置为 $q = 21$，$error\_max = 10^{-5}$。因此，在相同数值计算精度下对比算法运行时间不再有意义。两种方法的相对误差随参数 t 的变化如图4-9所示。

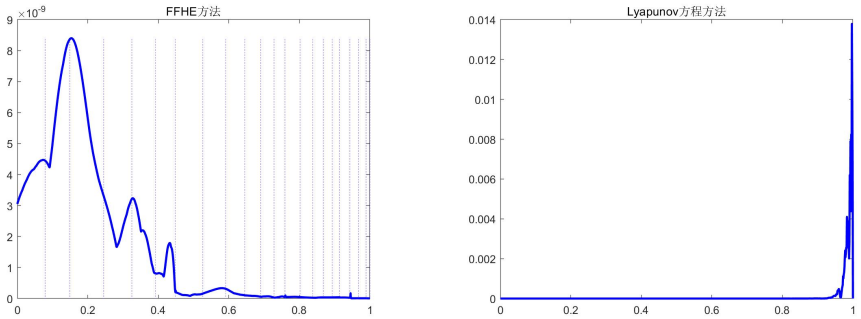

图 4-9 当 $n = 20$ 时各方法计算结果的相对误差

FFHE	5.1
Lyapunov	0.39

表 4.6 当 $n = 20$ 时各方法的运行时间（秒）

可以看出，在矩阵维数 n 较大时，FFHE 方法能够达到的数值计算精度远远高于 Lyapunov 方程方法。与此同时，为达到极高的计算精度，前者的计算速度要慢很多。在实际要求的计算精度下，FFHE 能够满足精度要求，同时可以通过适当增大阈值 $error\_max$ 以及减小幂级数展开最高次数 q 等方式加快 FFHE 计算速度，但是 Lyapunov 方程方法却无法达到这样的计算精度。

4.4.3 矩阵维数 $n = 35$

当 $n = 35$ 时，Lyapunov 方程方法的计算效果极差，在相关参数设置下其所得计算结果误差极大，而 FFHE 方法依旧能够获得高精度的解，其误差约为 Lyapunov 方程方法的 10^8 分之一。它们的相对误差随参数 t 的变化如图4-10所示。其中，FFHE 方法的参数设置为 $q = 21$，$error\_max = 10^{-3}$。

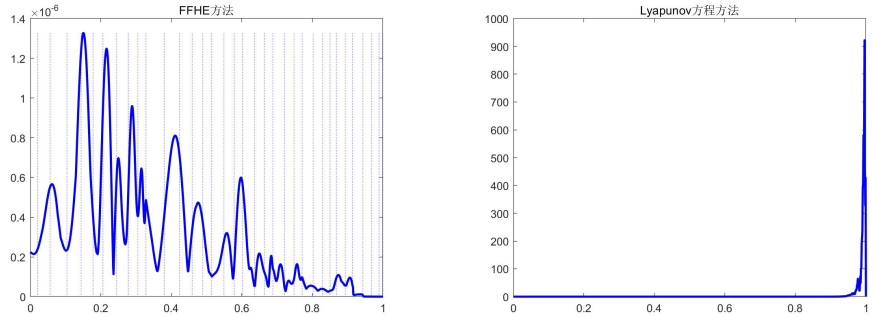

图 4-10 当 $n = 35$ 时各方法计算结果的相对误差

FFHE	56
Lyapunov	0.72

表 4.7 当 $n = 35$ 时各方法的运行时间（秒）

结合表4.7结果可以看出，在矩阵维数 n 进一步增大时，FFHE 方法虽然多耗费些计算时间，但依然能够计算得到高精度的数值解，但 Lyapunov 方程方法却无法得到精确的数值计算结果。因此，在计算微分黎卡提方程中的大规模待求解矩阵时，FFHE 方法还是能得到高精度的数值计算结果，表明其适用范围更加广泛，计算效果更好。

4.4.4 方法耗时分析

根据上述实验结果，FFHE 方法求解大规模黎卡提方程时可得到理想精度的计算结果，不过多消耗一定计算时间。因此，本节对其运行过程中的各部分耗时进行简要分析。在 FFHE 方法运行过程中，如表4.8所示，统计两个主要部分的运行时间：一是幂级数展开式求解，二是离散化有理逼近与自适应区间划分。

矩阵维数 n	5	20	35
计算幂级数展开式耗时	0.11	4.8	55
构造数值解与区间划分耗时	0.04	0.25	1.20
总耗时	0.15	5.1	56

表 4.8 当 $n = 5, 20, 35$ 时 FFHE 方法中两个主要部分的运行时间（秒）

从上表可以看出，随着问题规模增加，幂级数展开式求解部分逐渐成为 FFHE 方法的主要耗时部分。理论上，该部分具有整个算法流程中最高的计算复杂度 $\mathcal{O}(n^4q^2)$，即需要计算 n^2q 个系数值，每个系数值的计算复杂度为 $\mathcal{O}(\frac{1}{2}n^2q)$。而离散化有理逼近与自适应区间划分部分的计算复杂度取决于每个区间划分成功的平均尝试次数 s，计算复杂度为 $\mathcal{O}(nq^2s + n^2q^2)$，即构造用于有理逼近的矩阵的计算复杂度为 $\mathcal{O}(q^2)$；划分每个区间的过程中计算相对误差的有理逼近结果使用次数为 $\mathcal{O}(ns)$，计算下一区间初始状态矩阵数值的有理逼近结果使用次数为 $\mathcal{O}(n^2)$。由于采取了自适应区间划分策略使 s 降到很低，后者的复杂度可视为 $\mathcal{O}(n^2q^2)$，远远低于前者。当矩阵规模 n 增大，幂级数展开式求解部分的耗时增速更快，在达到一定规模后成为主要耗时部分，对这部分的算法优化（如并行化）变得更有必要。

4.5 设置区间长度

在矩阵维度 $n = 5$ 时，本节将参数区间从 $[0, 1]$ 扩展为 $[0, 10]$，即将 $T = 1$ 变为 $T = 10$，对比 FFHE 方法和 Lyapunov 方程方法的数值计算效果。利用与上一项实验中 $n = 5$ 对应实验的相同参数设置，控制两种方法的最大相对误差大致相等，在扩展后的参数区间上进行实验，对比它们的计算时间。采取两种方法带来的相对误差随参数 t 的变化如图4-11所示。

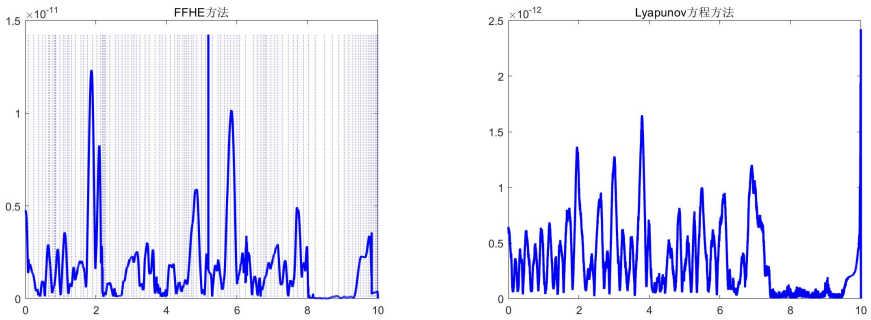

图 4-11 当 $T = 10$ 时的各方法计算结果的相对误差

参数 T 值	1	10
FFHE	0.15	0.40
Lyapunov	0.33	0.90

表 4.9 当 $T = 1, 10$ 时两种方法的运行时间（秒）

从上图及上表可以看出，随着区间总长度的增加，FFHE 方法和 Lyapunov 方程方法的运行时间是以大致相同的比例增加的，且前者的计算速度始终快于后者。因此，在小规模的微分黎卡提方程求解过程中，不论总区间长度为多大，在保证数值计算精度的前提下，FFHE 方法都具有计算速度的优越性。理论上，除非总区间长度极小，甚至小于 FFHE 方法自适应划分的区间长度，这种情况下无法充分发挥 FFHE 方法在计算速度方面的优势。二在总区间长度超过一定值，可容纳一定数目的划分区间时，FFHE 方法在计算速度方面的优势才能够充分发挥出来。

五、总结

本文提出了一种用于求解微分黎卡提方程的 FFHE 方法，它是一种高效且适用范围广的数值计算方法。本文介绍了 FFHE 方法所包含的幂级数展开、有理逼近、区间自适应划分等重要部分的基本设计框架，展示了其求解微分黎卡提方程的计算过程。本文还设计了一系列数值实验，在不同幂级数展开式最高次数、采用不同有理逼近方式、以及不同最大相对误差等条件下，对比分析了 FFHE 方法的计算效果。另外，在不同矩阵规模、不同区间长度等条件下，也对比分析了 FFHE 方法与 Lyapunov 方法等常用方法的计算效果。结果表明，在求解微分黎卡提方程问题时，FFHE 方法具有更高的结果精度、更快的计算速度、以及更广的适用范围。

参考文献

[1] STILLFJORD T. Low-rank second-order splitting of large-scale differential Riccati equations[J]. IEEE Transactions on Automatic Control, 2015, 60(10): 2791–2796.

[2] 赵龙, 田祥. 基于黎卡提微分方程的移动机器人运动误差控制研究 [J]. 机床与液压, 2019(23).

[3] CHIANG H D, WANG T, SHENG H. A novel fast and flexible holomorphic embedding power flow method[J]. IEEE Transactions on Power Systems, 2018, 33(3): 2551–2562.

[4] 刘颖. 几类特殊的黎卡提 (Riccati) 方程通解求法 [J]. 沈阳航空工业学院学报, 2000(4): 65–68.

[5] 宋丽娟, 孙礼信. 关于黎卡提方程的一种解法 [J]. 白城师范学院学报, 2007(03): 11–13.

[6] 王明建. Riccati 微分方程特解新求法的研究 [J]. 数学的实践与认识, 2006(07): 384–388.

[7] NGUYEN T, GAJIC Z. Solving the matrix differential Riccati equation: A Lyapunov equation approach[J]. IEEE Transactions on Automatic Control, 2010, 55(1): 191–194.

[8] LAUB A J. A Schur method for solving algebraic Riccati equations[J]. IEEE Transactions on Automatic Control, 1979, 24(6): 913–921.

[9] ANDERSON B, MOORE J B, MOLINARI B P. Linear optimal control[J]. IEEE Transactions on Systems Man & Cybernetics, 1971, 93(4): 559–559.

[10] DAVISON E, MAKI M. The numerical solution of the matrix Riccati differential equation[J]. IEEE Transactions on Automatic Control, 1973, 18(1): 71–73.

[11] BAKER G A, PETER G M. Pade approximants[J], 1996.

致谢

四年的本科学习转眼就将要结束，在这四年的时光中，我在理论课和实验课的课堂上学习到了很多深奥的知识和实用的技能，也在课外的一次次单人课程项目与小组合作项目活动中，体会到数学应用与计算机实践的辛苦和乐趣，收获到同学之间互相合作的宝贵经验。在此，我向所有引导和帮助过我的老师和同学们，表达最诚挚的谢意。

首先，要感谢的是我的指导老师汪涛老师。作为一名数学计算和编程基础比较薄弱、算法应用和实践经验比较匮乏的本科生，在汪老师的指导下，我对 FFHE 算法等数值计算方法产生了浓厚兴趣，对它们的应用领域有了一定了解，并夯实自己的理论基础和编程能力，尝试设计应用这些计算方法。汪老师对数值计算领域有广博的见识和深刻的理解，给了我很多极有帮助的方向性建议和技巧性指导。他严谨而创新的科研方式和孜孜不倦的工作态度是我学习的榜样，在此向汪老师致以崇高的敬意和衷心的感谢。

其次，我要感谢我的同学们，在日常学习和生活中给我提供了很多有益的建议和无私的帮助，拓展了我的知识面，让我的学习和研究过程更加顺利。尤其是我在汪老师研究组中的队友丁聿宁同学，在设计 FFHE 方法求解 Riccati 方程的研究中，他给予了我大量鼓励与帮助，提出了不少改进思路，在此表达我衷心的感谢。

最后，我要感谢我的家人，有了他们的支持，我才充满信心地投入到学习生活中，取得现在的成绩。

贾智超

2021 年 4 月 28 日

附录 A 补充 $n=20$ 和 $n=35$ 时的常数矩阵数据

首先，随机生成的 $n=20$ 维常数矩阵如下。

7	3	-1	5	-3	-7	-8	7	6	1	3	-4	-9	-10	-10	-9	-7	9	-10	-5
9	-10	-2	-5	7	6	10	3	-2	-4	-3	9	-5	8	5	4	-2	9	1	-4
-8	7	6	0	2	-4	-10	-3	-5	5	7	-1	6	9	0	-10	7	-9	8	2
9	9	6	4	1	1	6	0	-2	-7	1	-7	-10	6	0	-9	6	5	4	-5
3	4	-7	8	9	-7	7	-2	-8	4	-3	9	9	-8	8	0	-9	-5	-7	7
-8	5	0	10	-4	2	8	-9	-8	-7	9	10	5	-5	2	-8	-2	-2	-3	10
-5	5	-1	1	5	-5	-9	-5	9	-3	8	-1	0	-3	2	7	1	1	-1	5
1	-2	3	-8	5	3	-2	-8	10	3	1	-8	2	4	8	7	-2	9	10	-3
10	3	4	-7	-3	4	-5	-7	2	6	3	-5	-6	-8	6	5	3	-2	-7	2
10	-7	5	-5	1	5	6	-5	-9	-9	2	-2	-1	5	2	-7	3	10	7	-8
-7	4	-5	7	-9	-1	-1	-2	-6	9	-6	2	10	-8	-7	3	-4	-4	3	9
10	-10	4	-5	-9	-9	9	-9	-3	6	-4	-5	1	3	-5	0	-1	4	-3	8
10	-5	3	7	1	-6	-7	8	7	0	-1	2	0	0	8	10	-10	3	-6	7
0	-10	-7	-5	6	9	-5	9	-10	-1	-6	4	-6	6	-10	3	10	1	-2	-5
6	-8	-8	9	9	-7	-7	0	-10	-1	7	-6	0	5	0	6	-7	4	0	2
-8	7	0	-3	-8	7	-8	0	-7	-4	-6	-8	3	8	-7	-1	-8	3	-8	-10
-2	4	10	-6	1	1	8	-3	3	0	-6	-4	4	8	10	-1	-3	-7	2	-2
9	-4	-3	-5	-1	10	2	8	5	0	-7	-4	-2	-3	4	7	-6	-8	-6	-4
6	9	2	2	-10	-9	1	-3	3	7	-6	-2	-3	4	0	-9	0	10	-2	-7
10	-10	-6	-1	-3	-1	-7	-8	-1	6	-1	0	10	-6	-1	-8	-3	-7	2	-7

图 A-1 矩阵 A

-40	-6	58	28	50	38	12	-26	-12	-56	2	-48	-22	-46	40	-4	26	-32	-40	58
-6	-85	0	-19	22	26	25	-88	-83	-20	9	45	-12	83	35	98	1	-28	-64	0
58	0	-85	-43	-71	-53	-15	29	9	80	-2	75	31	76	-55	16	-38	44	52	-85
28	-19	-43	-26	-31	-21	-2	-5	-14	36	1	48	13	57	-20	30	-19	16	12	-43
50	22	-71	-31	-65	-51	-19	47	29	72	-4	51	29	42	-55	-12	-32	44	60	-71
38	26	-53	-21	-51	-41	-17	45	31	56	-4	33	23	22	-45	-20	-24	36	52	-53
12	25	-15	-2	-19	-17	-10	31	26	20	-3	0	9	-11	-20	-26	-7	16	28	-15
-26	-88	29	-5	47	45	31	-101	-89	-48	10	21	-23	60	55	96	14	-44	-84	29
-12	-83	9	-14	29	31	26	-89	-82	-28	9	36	-15	73	40	94	5	-32	-68	9
-56	-20	80	36	72	56	20	-48	-28	-80	4	-60	-32	-52	60	8	36	-48	-64	80
2	9	-2	1	-4	-4	-3	10	9	4	-1	-3	2	-7	-5	-10	-1	4	8	-2
-48	45	75	48	51	33	0	21	36	-60	-3	-90	-21	-111	30	-66	33	-24	-12	75
-22	-12	31	13	29	23	9	-23	-15	-32	2	-21	-13	-16	25	8	14	-20	-28	31
-46	83	76	57	42	22	-11	60	73	-52	-7	-111	-16	-149	15	-110	33	-12	16	76
40	35	-55	-20	-55	-45	-20	55	40	60	-5	30	25	15	-50	-30	-25	40	60	-55
-4	98	16	30	-12	-20	-26	96	94	8	-10	-66	8	-110	-30	-116	6	24	64	16
26	1	-38	-19	-32	-24	-7	14	5	36	-1	33	14	33	-25	6	-17	20	24	-38
-32	-28	44	16	44	36	16	-44	-32	-48	4	-24	-20	-12	40	24	20	-32	-48	44
-40	-64	52	12	60	52	28	-84	-68	-64	8	-12	-28	16	60	64	24	-48	-80	52
58	0	-85	-43	-71	-53	-15	29	9	80	-2	75	31	76	-55	16	-38	44	52	-85

图 A-2 矩阵 S

其次，随机生成的 $n=35$ 维的常数矩阵如下。

-9	-3	6	-8	1	1	5	0	9	-7	4	-9	8	5	0	-4	-7	-10	-8	-6	1	3	-4	8	3	-9	6	5	-2	8	6	4	10	10	4
1	-6	-1	-1	8	8	2	-4	10	9	-3	7	-4	10	-5	10	-5	-1	3	6	-2	4	-7	4	2	1	2	-5	-4	-6	-9	-2	4	6	-2
9	6	4	-5	6	-3	-2	9	-10	3	-5	-7	-5	-3	10	8	-9	-10	-3	-9	-8	9	-3	6	-8	4	0	6	9	-2	5	3	-7	-6	-9
-4	5	-2	10	-9	-10	0	10	0	-7	-5	-3	2	5	-7	-7	2	-1	-3	9	5	4	5	1	-8	0	-5	-8	-1	8	4	-6	8	10	-6
6	-5	-3	2	-5	-8	0	0	8	1	6	3	-1	-8	8	-10	6	9	2	-7	2	1	4	10	-5	2	-5	0	-4	-5	-1	2	9	-9	-5
2	2	-6	-5	-3	-9	3	10	-5	9	-9	10	-3	2	-10	4	0	-5	8	-4	6	-3	4	-9	5	-6	0	2	-9	10	2	4	10	-2	-6
6	-2	-2	-8	8	5	5	-1	7	6	4	0	3	-1	-4	8	-10	-2	-6	4	1	3	-10	1	-2	3	-6	4	2	3	-3	2	1	-2	4
6	-9	-10	1	-2	-1	-4	-2	7	4	3	10	9	5	5	-8	10	-1	4	-8	7	-3	-3	-5	7	-3	-9	-2	-7	-7	-7	-3	1	-2	5
-9	-10	-1	7	-5	4	-9	-6	3	-1	5	-6	3	-5	3	-10	0	7	8	0	2	-2	8	0	9	-8	-7	5	-9	7	-2	-3	-7	-8	1
-5	0	-3	7	2	4	7	-6	-10	-5	10	-2	-5	-9	-7	2	-4	7	-6	5	9	-3	-4	4	-2	-3	-5	8	2	3	9	-7	0	-2	1
3	-5	10	7	0	1	-2	7	-3	1	-2	4	-6	-1	9	2	-7	-1	-4	9	8	-1	2	-6	-9	-5	-6	-9	-4	1	-6	6	1	2	3
0	-3	1	-6	8	3	6	5	8	-5	-5	-9	-1	3	1	0	3	-3	0	5	0	0	-4	2	1	-5	2	-8	-3	-5	-3	0	-7	10	10
10	-4	7	1	9	8	-7	1	4	-4	8	5	-10	-8	-5	-10	2	9	8	5	6	9	-8	-4	5	3	7	8	5	-10	-4	-3	0	-6	3
5	-7	-2	8	7	3	-8	7	-3	9	6	-2	8	-1	2	4	5	5	0	-2	-1	-6	-2	8	-4	0	7	-2	8	-6	-9	6	1	-3	2
1	-2	-1	-8	4	8	9	0	3	8	-1	7	2	3	9	9	-7	5	-5	-5	7	-3	7	-8	0	-2	6	-2	-3	-3	0	-6	-6	-5	9
-4	4	7	7	5	-1	8	1	-5	6	-9	-3	-6	7	8	8	-3	9	1	0	-4	2	10	-8	-6	-6	4	10	10	3	7	7	3	-4	1
-5	-6	10	8	8	-8	10	-6	1	9	1	-4	0	-4	0	-8	-7	0	2	-6	10	0	8	10	9	8	8	5	10	10	8	7	-10	-7	-3
8	3	1	-6	-9	-9	8	2	10	-7	-7	10	-9	-2	-1	-3	7	6	-2	7	-5	-5	-9	-7	0	8	6	10	-6	-6	5	7	-4	-10	10
-1	-1	9	-9	-3	4	-10	-8	7	0	1	5	8	8	-4	-5	-3	-1	-10	3	-9	2	-3	-7	1	-5	-6	-6	6	5	-2	-3	1	-1	-1
7	-1	5	-1	-10	-4	1	-9	-5	3	-3	-2	-1	4	-9	1	-5	7	4	7	5	8	-6	-7	-6	2	10	-8	2	8	-4	-2	-4	-5	2
-8	-7	1	7	7	4	10	4	4	9	-4	-8	1	-5	5	2	-10	-2	0	6	7	-9	-4	-9	-8	6	-9	-2	9	-1	4	8	2	8	5
8	-6	10	-1	0	3	0	2	-3	3	5	5	1	5	-8	-4	6	5	-2	-1	9	-1	10	-1	-9	10	3	0	2	-7	8	-1	-3	1	4
-10	2	7	-2	-3	1	8	-6	10	-2	0	3	4	-4	-3	6	0	10	-9	-4	-4	-9	1	4	7	9	-6	1	4	7	9	1	-8	9	-6
8	-5	10	0	-6	5	-9	-2	-9	5	6	-9	-3	-5	-2	6	1	0	-3	4	6	1	5	7	6	-2	-5	10	5	0	-9	4	9	10	-9
8	1	3	-8	1	0	10	3	-5	7	-6	-8	-9	-10	8	6	8	-1	-6	10	1	1	7	6	-6	-7	-3	0	-2	-8	0	10	3	2	-5
1	9	-3	-8	-4	0	9	1	7	2	4	10	-1	-3	-10	-3	5	-6	-6	6	-1	6	-7	4	8	6	-6	-2	-5	8	-8	1	3	-8	8
-8	5	0	8	-9	0	1	-8	-9	-9	-9	0	-9	-1	7	-9	4	-4	7	7	7	-6	8	-1	4	-8	0	10	-10	3	-5	3	4	9	1
-7	4	9	2	-9	9	-1	-7	1	9	6	-10	5	-4	-4	4	-7	-8	-2	4	9	2	-8	4	3	-7	9	-5	-1	-8	8	1	5	2	-1
4	10	-10	-5	-9	-2	-3	-10	6	-9	4	-9	-10	-4	-5	10	1	-3	-9	2	-9	-1	5	10	10	-3	-2	3	3	9	-6	9	7	8	-1
7	6	-7	8	-2	-8	5	-2	5	1	9	-8	10	-5	0	-7	10	-4	-3	5	4	8	5	0	-10	7	2	10	-5	-10	-8	8	0	-2	6
-10	2	-1	-9	-8	-5	-10	0	-8	-8	-9	8	5	9	5	-8	2	-3	6	0	-6	3	5	-4	-2	2	2	4	0	-10	1	-7	-7	2	3
5	9	1	-1	-1	4	-3	-7	-1	-3	9	-1	9	8	-6	9	6	7	-7	8	-2	-3	-8	6	0	2	5	-4	-5	10	-4	-7	-3	-9	3
10	-9	-9	5	8	7	9	4	-3	7	0	1	0	-2	-6	0	10	1	9	-9	-5	-3	-1	-6	2	9	-2	1	-4	4	-2	-5	7	9	-7
-3	-5	3	-3	-3	10	1	-10	0	-5	2	0	-5	0	10	7	8	0	-4	10	7	-5	0	-6	-9	2	8	-10	3	-3	6	5	-7	3	-1
3	10	8	-2	-8	-6	-1	-8	2	8	-4	-9	-5	1	2	7	-6	6	-4	-8	10	10	1	-1	3	-10	9	8	8	0	7	-6	-4	-8	0

图 A-3 矩阵 A

-45	-30	39	36	36	12	-36	-3	3	-18	-6	-66	6	15	-15	12	-36	33	-3	-33	18	-54	-18	54	-24	69	-45	-21	42	45	63	18	24	-90	87
-30	-25	40	15	35	0	-30	-15	-20	-15	-25	-40	-15	35	-10	35	-45	45	15	-10	5	-30	-5	30	-20	35	-15	-15	30	35	35	35	30	-50	50
39	40	-73	-6	-62	12	48	39	59	24	64	46	48	-83	13	-86	90	-93	-45	-5	4	30	-4	-30	32	-29	-3	21	-42	-53	-35	-80	-60	50	-53
36	15	-6	-45	-9	-24	18	-21	-42	9	-33	60	-39	33	12	39	-9	15	33	48	-27	54	27	-54	12	-75	63	15	-30	-27	-63	27	6	90	-84
36	35	-62	-9	-53	8	42	31	46	21	51	44	37	-67	12	-69	75	-77	-35	0	1	30	-1	-30	28	-31	3	19	-38	-47	-35	-65	-50	50	-52
12	0	12	-24	8	-16	0	-20	-36	0	-32	24	-32	36	4	40	-24	28	28	28	-16	24	16	-24	0	-36	36	4	-8	-4	-28	32	16	40	-36
-36	-30	48	18	42	0	-36	-18	-24	-18	-30	-48	-18	42	-12	42	-54	54	18	-12	6	-36	-6	36	-24	42	-18	-18	36	42	42	42	36	-60	60
-3	-15	39	-21	31	-20	-18	-34	-57	-9	-55	6	-49	66	-1	71	-57	62	44	29	-17	12	17	-12	-12	-24	36	-4	8	16	-14	61	38	20	-15
3	-20	59	-42	46	-36	-24	-57	-97	-12	-92	22	-84	109	1	118	-90	99	75	55	-32	30	32	-30	-16	-53	69	-3	6	19	-35	100	60	50	-41
-18	-15	24	9	21	0	-18	-9	-12	-9	-15	-24	-9	21	-6	21	-27	27	9	-6	3	-18	-3	18	-12	21	-9	-9	18	21	21	21	18	-30	30
-6	-25	64	-33	51	-32	-30	-55	-92	-15	-89	8	-79	107	-2	115	-93	101	71	46	-27	18	27	-18	-20	-37	57	-7	14	27	-21	99	62	30	-22
-66	-40	46	60	44	24	-48	6	22	-24	8	-100	24	2	-22	-4	-36	30	-18	-58	32	-84	-32	84	-32	110	-78	-30	60	62	98	8	24	-140	134
6	-15	48	-39	37	-32	-18	-49	-84	-9	-79	24	-73	93	2	101	-75	83	65	50	-29	30	29	-30	-12	-51	63	-1	2	13	-35	85	50	50	-42
15	35	-83	33	-67	36	42	66	109	21	107	2	93	-130	5	-139	117	-126	-84	-49	29	-12	-29	12	28	32	-60	12	-24	-40	14	-121	-78	-20	11
-15	-10	13	12	12	4	-12	-1	1	-6	-2	-22	2	5	-5	4	-12	11	-1	-11	6	-18	-6	18	-8	23	-15	-7	14	15	21	6	8	-30	29
12	35	-86	39	-69	40	42	71	118	21	115	-4	101	-139	4	-149	123	-133	-91	-56	33	-18	-33	18	28	41	-69	11	-22	-39	21	-129	-82	-30	20
-36	-45	90	-9	75	-24	-54	-57	-90	-27	-93	-36	-75	117	-12	123	-117	123	69	24	-15	-18	15	18	-36	9	27	-21	42	57	21	111	78	-30	36
33	45	-93	15	-77	28	54	62	99	27	101	30	83	-126	11	-133	123	-130	-76	-31	19	12	-19	-12	36	0	-36	20	-40	-56	-14	-119	-82	20	-27
-3	15	-45	33	-35	28	18	44	75	9	71	-18	65	-84	-1	-91	69	-76	-58	-43	25	-24	-25	24	12	42	-54	2	-4	-14	28	-77	-46	-40	33
-33	-10	-5	48	0	28	-12	29	55	-6	46	-58	50	-49	-11	-56	24	-31	-43	-53	30	-54	-30	54	-8	77	-69	-13	26	21	63	-42	-16	-90	83
18	5	4	-27	1	-16	6	-17	-32	3	-27	32	-29	29	6	33	-15	19	25	30	-17	30	17	-30	4	-43	39	7	-14	-11	-35	25	10	50	-46
-54	-30	30	54	30	24	-36	12	30	-18	18	-84	30	-12	-18	-18	-18	12	-24	-54	30	-72	-30	72	-24	96	-72	-24	48	48	84	-6	12	-120	114
-18	-5	-4	27	-1	16	-6	17	32	-3	27	-32	29	-29	-6	-33	15	-19	-25	-30	17	-30	-17	30	-4	43	-39	-7	14	11	35	-25	-10	-50	46
54	30	-30	-54	-30	-24	36	-12	-30	18	-18	84	-30	12	18	18	18	-12	24	54	-30	72	30	-72	24	-96	72	24	-48	-48	-84	6	-12	120	-114
-24	-20	32	12	28	0	-24	-12	-16	-12	-20	-32	-12	28	-8	28	-36	36	12	-8	4	-24	-4	24	-16	28	-12	-12	24	28	28	28	24	-40	40
69	35	-29	-75	-31	-36	42	-24	-53	21	-37	110	-51	32	23	41	9	0	42	77	-43	96	43	-96	28	-130	102	30	-60	-58	-112	23	-6	160	-151
-45	-15	-3	63	3	36	-18	36	69	-9	57	-78	63	-60	-15	-69	27	-36	-54	-69	39	-72	-39	72	-12	102	-90	-18	36	30	84	-51	-18	-120	111
-21	-15	21	15	19	4	-18	-4	-3	-9	-7	-30	-1	12	-7	11	-21	20	2	-13	7	-24	-7	24	-12	30	-18	-10	20	22	28	13	14	-40	39
42	30	-42	-30	-38	-8	36	8	6	18	14	60	2	-24	14	-22	42	-40	-4	26	-14	48	14	-48	24	-60	36	20	-40	-44	-56	-26	-28	80	-78
45	35	-53	-27	-47	-4	42	16	19	21	27	62	13	-40	15	-39	57	-56	-14	21	-11	48	11	-48	28	-58	30	22	-44	-50	-56	-41	-38	80	-79
63	35	-35	-63	-35	-28	42	-14	-35	21	-21	98	-35	14	21	21	21	-14	28	63	-35	84	35	-84	28	-112	84	28	-56	-56	-98	7	-14	140	-133
18	35	-80	27	-65	32	42	61	100	21	99	8	85	-121	6	-129	111	-119	-77	-42	25	-6	-25	6	28	23	-51	13	-26	-41	7	-113	-74	-10	2
24	30	-60	6	-50	16	36	38	60	18	62	24	50	-78	8	-82	78	-82	-46	-16	10	12	-10	-12	24	-6	-18	14	-28	-38	-14	-74	-52	20	-24
-90	-50	50	90	50	40	-60	20	50	-30	30	-140	50	-20	-30	-30	-30	20	-40	-90	50	-120	-50	120	-40	160	-120	-40	80	80	140	-10	20	-200	190
87	50	-53	-84	-52	-36	60	-15	-41	30	-22	134	-42	11	29	20	36	-27	33	83	-46	114	46	-114	40	-151	111	39	-78	-79	-133	2	-24	190	-181

图 A-4 矩阵 S