Advances in Applied Mathematics 应用数学进展, 2025, 14(4), 155-165
Published Online April 2025 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2025.144148

Hans汉斯

耦合Sylvester转置矩阵方程的多迭代因子梯度算法

冯文慧

广东工业大学数学与统计学院，广东 广州

收稿日期：2025年3月3日；录用日期：2025年3月26日；发布日期：2025年4月8日

摘　要

本文研究了耦合Sylvester转置矩阵方程的数值求解问题。通过充分利用方程的耦合性，引入了多个参数，并提出了多迭代因子梯度(MGI)算法。此外，还推导出了确保算法生成的迭代解对于任意初始矩阵都收敛到精确解的充要条件。最后，通过一个数值例子验证了所提算法的有效性。

关键词

耦合Sylvester转置矩阵方程，迭代算法，多迭代因子，收敛性分析

Multi-Iterative Factors Gradient Algorithm for the Coupled Sylvester-Transpose Matrix Equations

Wenhui Feng

School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou Guangdong

Received: Mar. 3rd, 2025; accepted: Mar. 26th, 2025; published: Apr. 8th, 2025

Abstract

This paper investigates the numerical solution of coupled Sylvester-transpose matrix equations. By fully exploiting the coupling of the equations, several parameters are introduced, and an multi-iterative factors gradient algorithm is proposed. Additionally, the necessary and sufficient conditions that ensure the convergence of the iterative solutions generated by the algorithm to the exact solution for any initial matrix are derived. Finally, the effectiveness of the proposed algorithm has been verified through a numerical example.

文章引用: 冯文慧. 耦合 Sylvester 转置矩阵方程的多迭代因子梯度算法[J]. 应用数学进展, 2025, 14(4): 155-165.
DOI: 10.12677/aam.2025.144148

Keywords

Coupled Sylvester-Transpose Matrix Equation, Iterative Algorithm, Multi-Iterative Factors, Convergence Analysis

1. 引言

矩阵方程作为矩阵理论的重要组成部分，在数学与工程科学的多个领域中发挥着至关重要的作用，广泛应用于信号处理、稳定性分析、控制和系统理论等方面[1]-[4]。深入研究 Sylvester 矩阵方程的数值求解方法，不仅能够丰富矩阵理论的应用领域，还可以推动相关数学理论的发展。因此，提出高效的数值求解算法，对于提高解决实际工程问题的效率和精度至关重要。本文主要研究耦合 Sylvester 转置矩阵方程，这对实际问题中的观测器设计[5]等方面具有重要意义。

早期对矩阵方程的研究，主要通过克罗内克积和向量化算子，将矩阵方程转化为非齐次线性方程以获取解析解。然而，随着问题规模的不断扩大，矩阵维数呈指数增加，导致计算复杂度显著上升，存储需求也随之增大。因此，一些迭代方法来求矩阵方程的数值解成为了主流研究。在文献[6] [7]中，基于层次识别原理，Ding 和 Chen 等人提出了一种基于梯度的迭代(GI)算法和一种基于最小二乘的迭代(LSI)算法来求解广义 Sylvester 矩阵方程 $AXB=F$ 和 $AXB+CXD=F$ 。文献[8]用梯度法求解 Sylvester 转置矩阵方程 $AXB+CX^{\mathrm{T}}D=F$ 。后来，一些研究者对梯度法进行了一系列的改进，以加快算法的收敛速度。文献[9]提出了一种求解 Sylvester 矩阵方程的基于松弛梯度的迭代(RGI)算法。文献[10]提出了一种基于加速梯度的迭代(AGI)算法求解 Sylvester 矩阵方程。这两种改进的算法通过引入松弛因子和权重因子来调整算法参数和迭代格式，有效提升了收敛速度与精度，并在不同类型的矩阵方程中得到应用。在耦合 Sylvester 转置矩阵方程的研究方面，也取得了一些成果。文献[11]求解方程 $\sum_{i=1}^{r}A_iXB_i+\sum_{j=1}^{s}C_jX^{\mathrm{T}}D_j=E$，通过共轭梯度法得到了方程的最小二乘解和最小范数解。文献[12]利用层次识别原理，引入了单个迭代因子，提出了一种求解方程 $\sum_{\eta=1}^{p}\left(A_{i\eta}X_{\eta}B_{i\eta}+C_{i\eta}X_{\eta}^{\mathrm{T}}D_{i\eta}\right)=F_i, i\in I[1,N]$ 的迭代算法。

基于上述研究，本文继续研究耦合 Sylvester 转置矩阵方程的数值算法。针对此类方程的特点，基于梯度算法和单因子收敛法，引入多个参数，进而提出了多迭代因子梯度算法。

2. 主要问题和预备知识

本文我们主要通过构造 MGI 算法来求解如下形式的耦合 Sylvester 转置矩阵方程：

$$\sum_{j=1}^{p}\left(A_{ij}X_jB_{ij}+C_{ij}X_j^{\mathrm{T}}D_{ij}\right)=F_i,\quad i=[1,s] \tag{1}$$

$A_{ij}\in R^{m_i\times r_j}$，$C_{ij}\in R^{m_i\times t_j}$，$B_{ij}\in R^{t_j\times n_i}$，$D_{ij}\in R^{r_j\times n_i}$，$F_i\in R^{m_i\times n_i}$，$X_j\in R^{r_j\times t_j}$，$i=[1,s]$，$j=[1,p]$，其中 X_j 为未知待测矩阵。

首先我们介绍一些符号，定义实矩阵 $A\in R^{m\times m}$，$B\in R^{n\times n}$，A^{T}，$\bar{A}$，A^H，$tr(A)$，$\lambda(A)$，$\rho(A)$ 分别表示矩阵 A 的转置，共轭，共轭转置，迹，特征值，谱范数。矩阵 A 的 2-范数和 F-范数定义为

$\|A\|_2=\sqrt{\lambda_{\max}\left(A^{\mathrm{T}}A\right)}=\sigma_{\max}(A)$，$\|A\|=\sqrt{\langle A,A\rangle}=\sqrt{\mathrm{tr}\left(A^{\mathrm{T}}A\right)}$。$\langle A,B\rangle=\mathrm{tr}\left(A^{\mathrm{T}}B\right)=\mathrm{tr}\left(B^{\mathrm{T}}A\right)$为矩阵 A 和 B 内积的定义，同时克罗内克积定义为 $A\otimes B=\left(a_{ij}B\right)$，即

$$A\otimes B=\begin{pmatrix}
a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1p} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1p}\\
a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2p} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1}b_{2p}\\
\vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots\\
a_{11}b_{p1} & a_{11}b_{p1} & \cdots & a_{11}b_{pp} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p1} & \cdots & a_{1n}b_{pp}\\
\vdots & \vdots & & \vdots & \ddots & & & & & \\
\vdots & \vdots & & \vdots & & \ddots & & & & \\
a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1p} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1p}\\
a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2p} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2p}\\
\vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots\\
a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pp} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}bb_{pp}
\end{pmatrix}$$

其次，我们将提供一些有用的结果，这些结果将在后续部分中发挥重要作用。参考[13]的工作，设 $P_{mn}\in R^{mn\times mn}$ 为一个给定的方阵，可以划分为 $m\times n$ 个子矩阵，且有

$$P(m,n)=\sum_{i=1}^{m}\sum_{j=1}^{n}E_{ij}\otimes E_{ij}^{\mathrm{T}},\quad E_{ij}=e_ie_j^{\mathrm{T}}。$$

则根据定义，我们有

$$\mathrm{vec}\left(X^{\mathrm{T}}\right)=P(m,n)\mathrm{vec}(X),\quad P(m,n)P(n,m)=I_{mn},\quad P(m,n)^{\mathrm{T}}=P(m,n)^{-1}=P(n,m)。$$

在此基础上，我们给出了所求矩阵方程的可解条件，通过运用克罗内克积，我们可以将方程(1)的两边同时列向量化，可以得到：

$$\mathrm{vec}(F_i)=\sum_{j=1}^{p}\left[B_{ij}^{\mathrm{T}}\otimes A_{ij}+\left(D_{ij}^{\mathrm{T}}\otimes C_{ij}\right)P_{r_it_j}\right]\mathrm{vec}(X_j),\quad i=[1,s],\quad j=[1,p]\tag{2}$$

展开可以写成 $Sx=f$，其中

$$S=\begin{bmatrix}
\left(D_{11}^{\mathrm{T}}\otimes C_{11}\right)P_{r_1t_1}+B_{11}^{\mathrm{T}}\otimes A_{11} & \cdots & \left(D_{1l}^{\mathrm{T}}\otimes C_{1l}\right)P_{r_1,t_l}+B_{1l}^{\mathrm{T}}\otimes A_{1l}\\
\left(D_{21}^{\mathrm{T}}\otimes C_{21}\right)P_{r_1t_1}+B_{21}^{\mathrm{T}}\otimes A_{21} & \cdots & \left(D_{2l}^{\mathrm{T}}\otimes C_{2l}\right)P_{r_1,t_l}+B_{2l}^{\mathrm{T}}\otimes A_{2l}\\
\vdots & & \vdots\\
\left(D_{s1}^{\mathrm{T}}\otimes C_{s1}\right)P_{r_1t_1}+B_{s1}^{\mathrm{T}}\otimes A_{s1} & \cdots & \left(D_{sl}^{\mathrm{T}}\otimes C_{sl}\right)P_{r_1,t_l}+B_{sl}^{\mathrm{T}}\otimes A_{sl}
\end{bmatrix},\quad x=\begin{bmatrix}\mathrm{vec}(X_1)\\ \mathrm{vec}(X_2)\\ \vdots\\ \mathrm{vec}(X_l)\end{bmatrix}$$

$$f=\begin{bmatrix}\mathrm{vec}(F_1)\\ \mathrm{vec}(F_2)\\ \vdots\\ \mathrm{vec}(F_s)\end{bmatrix}$$

当 S 为方阵，即满足

$$\sum_{i=1}^{s}m_in_i=\sum_{j=1}^{p}t_jr_j$$

则方程(1)有唯一解当且仅当 S 非奇异，此时方程(1)的解可以表示为 $x=R^{-1}f$。

此外，我们将给出几个引理，简要回顾一下之前使用梯度型算法来求解矩阵方程的一些工作。对于

矩阵方程的求解，如果系数矩阵是方阵，早期的方法是将系数矩阵利用“雅可比迭代和高斯–赛德尔迭代”的思想分解进行求解。后来，Ding 等人提出了 GI 算法和 LSI 算法，这两种算法将矩阵方程系数矩阵的适用性扩展到了非方阵进行求解，对于此类算法的一些收敛定理由**引理 3.1** 给出：

引理 3.1 [14] 考虑以下矩阵方程：

$$AXB = F,$$

其中 $A \in R^{m\times r}$，$B \in R^{s\times n}$ 和 $F \in R^{m\times n}$ 是未知矩阵，$X \in R^{r\times s}$ 是未知待测矩阵。对于这个矩阵方程，可以构造一种如下所示的迭代格式：

$$X(k+1) = X(k) + \mu A^{\mathrm{T}}\left(F - AX(k)B\right)B^{\mathrm{T}},$$

其中

$$0 < \mu < \frac{2}{\|A\|_2^2\|B\|_2^2}。$$

接下来给出的**引理 3.2** 和**引理 3.3** 讨论了块矩阵的谱范数与弗罗比尼乌斯范数之间的关系，并分析了算法的收敛性。

引理 3.2 [15] 对于任意矩阵 A，有以下关系成立

$$\|A\|_2 = \sigma_{\max}(A) \le \|A\|。$$

引理 3.3 [15] 设矩阵 $A = \left[A_{ij}\right]_{m\times n}$ 为一个分块矩阵，其中矩阵 A_{ij}，$i = [1,m]$，$j = [1,n]$ 的维数是兼容的，则对于任何 P-范数，有

$$\|A\|_p \le \left\|\left[\|A_{ij}\|_p\right]_{n\times n}\right\|_p。$$

3. 多迭代因子梯度算法

3.1. MGI 算法的构造

基于前面对梯度迭代算法的分析，本节我们提出了 MGI 算法来求解方程(1)，并讨论了该算法的收敛性和收敛速率。对于方程(1)的求解，之前一些研究中提出的单因子收敛法因为没有充分利用好方程组的耦合性和子方程的独立性之间的关系，导致收敛性的证明结果以及算法的收敛速度不太理想。因此针对此类方程的特点，基于梯度算法和单因子收敛法，我们引入多个参数，提出了改进的多迭代因子梯度算法。

其主要思想是搜索矩阵组 $X = (X_1, X_2, \cdots, X_l)$，最小化以下目标函数：

$$G(X) = \frac{1}{2}\sum_{i=1}^{\varpi}\mu_i\|R_i\|^2, \tag{3}$$

其中，$\mu_i > 0$，$R_i = \sum_{j=1}^{p}\left(A_{ij}X_jB_{ij} + C_{ij}X_j^{\mathrm{T}}D_{ij}\right) - F_i$，$i = [1,s]$。

我们提出 mMGI 算法如下：

步骤 1 输入初始矩阵 $A_{ij} \in R^{m_i\times r_j}$，$C_{ij} \in R^{m_i\times t_j}$，$B_{ij} \in R^{t_j\times n_i}$，$D_{ij} \in R^{r_j\times n_i}$，$F_i \in R^{m_i\times n_i}$，$X_j \in R^{r_j\times t_j}$，$i = [1,s]$，$j = [1,p]$。给定任意小的正数 ε，和合适的正数 ω_i，$i = [1,s]$，且满足 $\sum_{i=1}^{s}\omega_i = 1$。选取初始矩阵 $X_j(0)$，

$j=[1,p]$，设置 $k:=1$。

步骤 2 如果：

$$\delta_k=\frac{\sum_{i=1}^{s}\left\|F_i-\sum_{j=1}^{p}\left(A_{ij}X_jB_{ij}+C_{ij}X_j^{\mathrm{T}}D_{ij}\right)\right\|_F}{\sum_{i=1}^{s}\|F_i\|_F}<\varepsilon, \tag{4}$$

停止计算；否则，转到**步骤 3**。

步骤 3 计算对于，$i=[1,s]$，$l=[1,p]$，更新序列：

$$\begin{aligned}X_l^i(k+1)&=X_l^i(k)+\mu_iA_{il}^{\mathrm{T}}\left[F_i-\sum_{j=1}^{p}\left(A_{ij}X_j(k)B_{ij}+C_{ij}\left(X_j(k)\right)^{\mathrm{T}}D_{ij}\right)\right]B_{il}^{\mathrm{T}}\\&\quad+\mu_iD_{il}\left[F_i^{\mathrm{T}}-\sum_{j=1}^{p}\left(B_{ij}^{\mathrm{T}}X_j^{\mathrm{T}}(k)A_{ij}^{\mathrm{T}}+D_{ij}^{\mathrm{T}}X_j(k)C_{ij}^{\mathrm{T}}\right)\right]C_{il},\\X_l(k+1)&=\omega_1X_l^1(k+1)+\cdots+\omega_sX_l^s(k+1)。\end{aligned} \tag{5}$$

步骤 4 令 $k:=k+1$，返回**步骤 2** 继续计算。

3.2. mMGI 算法的收敛性

首先，我们给出 MGI 算法的收敛性分析，给出了使算法收敛的充分条件。

定理 3.1 假设耦合 Sylvester 转置矩阵方程(1)存在唯一解，如果选取迭代因子 μ_i 满足不等式

$$0<\mu_i<\frac{2}{s\omega_i\sum_{l=1}^{p}\left(\|A_{il}\|_2^2\|B_{il}\|_2^2+\|C_{il}\|_2^2\|D_{il}\|_2^2\right)}, \tag{6}$$

则由 mMGI 算法生成的迭代解 $X(k)=\left(X_1(k),X_2(k),\cdots,X_l(k)\right)$ 在任意初始条件下都收敛于方程的唯一解 X_j^*，$j=[1,p]$，即 $\lim_{k\to\infty}X_j(k)=X_j^*$，$j\in\overline{1,p}$。

证明 首先，我们定义迭代误差矩阵

$$\tilde{X}_l(k)=X_l(k)-X_l^*,\quad \tilde{X}_l^{(i)}(k)=X_l^{(i)}(k)-X_l^*,\quad i=[1,s],\quad l=[1,p]。 \tag{7}$$

将 mMGI 算法中的(5)前两行的两边同时减去 X_l^*，有

$$\begin{aligned}\tilde{X}_l^{(i)}(k)&=X_l^{(i)}(k)-X_l^*\\&=X_l^{(i)}(k-1)-X_l^*+\mu_i\left\{A_{il}^{\mathrm{T}}\left[\sum_{j=1}^{p}\left(A_{ij}X_j^*B_{ij}+C_{ij}\left(X_j^*\right)^{\mathrm{T}}D_{ij}\right)\right.\right.\\&\quad-\sum_{j=1}^{p}\left(A_{ij}X_j(k-1)B_{ij}+C_{ij}X_j^{\mathrm{T}}(k-1)D_{ij}\right)B_{il}^{\mathrm{T}}+D_{il}\left[\sum_{j=1}^{p}\left(A_{ij}X_j^*B_{ij}+C_{ij}\left(X_j^*\right)^{\mathrm{T}}D_{ij}\right)\right.\\&\quad\left.\left.\left.-\sum_{j=1}^{p}\left(A_{ij}X_j(k-1)B_{ij}+C_{ij}X_j^{\mathrm{T}}(k-1)D_{ij}\right)\right]^{\mathrm{T}}C_{il}\right\}\\&=\tilde{X}_l^{(i)}(k-1)-\mu_i\left\{A_{il}^{\mathrm{T}}\left[\sum_{j=1}^{p}\left(A_{ij}\tilde{X}_j(k-1)B_{ij}+C_{ij}\tilde{X}_j^{\mathrm{T}}(k-1)D_{ij}\right)\right]B_{il}^{\mathrm{T}},\ l=[1,p].\right.\end{aligned}$$

则

$$\begin{aligned}\tilde{X}_l(k) &= X_l(k) - X_l^* = \sum_{i=1}^{s} \omega_i \tilde{X}_l^{(i)}(k) \\ &= \tilde{X}_l(k-1) - \sum_{i=1}^{s} \mu_i \omega_i \left\{ A_{il}^{\mathrm{T}} \left[\sum_{j=1}^{p} \left(A_{ij} \tilde{X}_j(k-1) B_{ij} + C_{ij} \tilde{X}_j^{\mathrm{T}}(k-1) D_{ij} \right) \right] B_{il}^{\mathrm{T}} \right. \\ &\quad \left. + D_{il} \left[\sum_{j=1}^{p} \left(A_{ij} \tilde{X}_j(k-1) B_{ij} + C_{ij} \tilde{X}_j^{\mathrm{T}}(k-1) D_{ij} \right) \right]^{\mathrm{T}} C_{il} \right\} \\ &= \tilde{X}_l(k-1) - \sum_{i=1}^{s} \sum_{j=1}^{p} \mu_i \omega_i \left(A_{il}^{\mathrm{T}} A_{ij} \tilde{X}_j(k-1) B_{ij} B_{il}^{\mathrm{T}} + A_{il}^{\mathrm{T}} C_{ij} \tilde{X}_j^{\mathrm{T}}(k-1) D_{ij} B_{il}^{\mathrm{T}} \right. \\ &\quad \left. + D_{il} B_{ij}^{\mathrm{T}} \tilde{X}_j^{\mathrm{T}}(k-1) A_{ij}^{\mathrm{T}} C_{il} + D_{il} D_{ij}^{\mathrm{T}} \tilde{X}_j(k-1) C_{ij}^{\mathrm{T}} C_{il} \right), \ l = [1, p].\end{aligned}$$

定义

$$\begin{aligned}\theta_i(k) &= F_i - \sum_{j=1}^{p} \left(A_{ij} X_j(k) B_{ij} + C_{ij} X_j^{\mathrm{T}}(k) D_{ij} \right) \\ &= -\sum_{j=1}^{p} \left(A_{ij} \tilde{X}_j(k) B_{ij} + C_{ij} \tilde{X}_j^{\top}(k) D_{ij} \right),\end{aligned}$$

则

$$\tilde{X}_l(k) = \tilde{X}_l(k-1) + \sum_{i=1}^{s} \mu_i \omega_i \left(A_{il}^{\mathrm{T}} \theta_i(k-1) B_{il}^{\mathrm{T}} + D_{il} \theta_i^{\mathrm{T}}(k-1) C_{il} \right) 。 \tag{8}$$

取(8)两边 F 范数的平方，可以得到：

$$\begin{aligned}\left\| \tilde{X}_l(k) \right\|_F^2 &= \left\| \tilde{X}_l(k-1) + \sum_{i=1}^{s} \mu_i \omega_i \left(A_{il}^{\mathrm{T}} \theta_i(k-1) B_{il}^{\mathrm{T}} + D_{il} \theta_i^{\mathrm{T}}(k-1) C_{il} \right) \right\|_F^2 \\ &= \left\| \tilde{X}_l(k-1) \right\|_F^2 + \left\| \sum_{i=1}^{s} \mu_i \omega_i \left(A_{il}^{\mathrm{T}} \theta_i(k-1) B_{il}^{\mathrm{T}} + D_{il} \theta_i^{\mathrm{T}}(k-1) C_{il} \right) \right\|_F^2 \\ &\quad + \mathrm{tr} \left[\tilde{X}_l^{\mathrm{T}}(k-1) \sum_{i=1}^{s} \mu_i \omega_i \left(A_{il}^{\mathrm{T}} \theta_i(k-1) B_{il}^{\mathrm{T}} + D_{il} \theta_i^{\mathrm{T}}(k-1) C_{il} \right) \right] \\ &\quad + \mathrm{tr} \left[\sum_{i=1}^{s} \mu_i \omega_i \left(B_{il} \theta_i^{\mathrm{T}}(k-1) A_{il} + C_{il}^{\mathrm{T}} \theta_i(k-1) D_{il}^{\mathrm{T}} \right) \tilde{x}_l(k-1) \right] \\ &= \left\| \tilde{X}_l(k-1) \right\|_F^2 + \left\| \sum_{i=1}^{s} \mu_i \omega_i \left(A_{il}^{\mathrm{T}} \theta_i(k-1) B_{il}^{\mathrm{T}} + D_{il} \theta_i^{\mathrm{T}}(k-1) C_{il} \right) \right\|_F^2 \\ &\quad + \mathrm{tr} \left(\sum_{i=1}^{s} \mu_i \omega_i \tilde{X}_l^{\mathrm{T}}(k-1) A_{il}^{\mathrm{T}} \theta_i(k-1) B_{il}^{\mathrm{T}} \right) + \mathrm{tr} \left(\sum_{i=1}^{s} \mu_i \omega_i \tilde{X}_l^{\mathrm{T}}(k-1) D_{il} \theta_i^{\mathrm{T}}(k-1) C_{il} \right) \\ &\quad + \mathrm{tr} \left(\sum_{i=1}^{s} \mu_i \omega_i B_{il} \theta_i^{\mathrm{T}}(k-1) A_{il} \tilde{X}_l(k-1) \right) + \mathrm{tr} \left(\sum_{i=1}^{s} \mu_i \omega_i C_{il}^{\mathrm{T}} \theta_i(k-1) D_{il}^{\mathrm{T}} \tilde{X}_l^{\mathrm{T}}(k-1) \right).\end{aligned}$$

由于 $tr(AB) = tr(BA)$，则上式可以进一步转化为

$$\begin{aligned}\left\|\tilde{X}_l(k)\right\|_F^2 &= \left\|\tilde{X}_l(k-1)+\sum_{i=1}^{s}\mu_i\omega_i\left(A_{il}^{\mathrm{T}}\theta_i(k-1)B_{il}^{\mathrm{T}}+D_{il}\theta_i^{\mathrm{T}}(k-1)C_{il}\right)\right\|_F^2 \\ &\quad+\mathrm{tr}\left(\sum_{i=1}^{s}\mu_i\omega_i B_{il}^{\mathrm{T}}\tilde{X}_l^{\mathrm{T}}(k-1)A_{il}^{\mathrm{T}}\theta_i(k-1)\right)+\mathrm{tr}\left(\sum_{i=1}^{s}\mu_i\omega_i\theta_i^{\mathrm{T}}(k-1)C_{il}\tilde{X}_l^{\mathrm{T}}(k-1)D_{il}\right) \\ &\quad+\mathrm{tr}\left(\sum_{i=1}^{s}\mu_i\omega_i\theta_i^{T}(k-1)A_{il}\tilde{X}_l(k-1)B_{il}\right)+\mathrm{tr}\left(\sum_{i=1}^{s}\mu_i\omega_i D_{il}^{\mathrm{T}}\tilde{X}_l^{\mathrm{T}}(k-1)C_{il}^{\mathrm{T}}\theta_i(k-1)\right) \\ &= \left\|\tilde{X}_l(k-1)+\sum_{i=1}^{s}\mu_i\omega_i\left(A_{il}^{\mathrm{T}}\theta_i(k-1)B_{il}^{\mathrm{T}}+D_{il}\theta_i^{\mathrm{T}}(k-1)C_{il}\right)\right\|_F^2 \\ &\quad+\mathrm{tr}\left[\sum_{i=1}^{s}\mu_i\omega_i\left(B_{il}^{\mathrm{T}}\tilde{X}_l^{\mathrm{T}}(k-1)A_{il}^{\mathrm{T}}+D_{il}^{\mathrm{T}}\tilde{X}_l(k-1)C_{il}^{\mathrm{T}}\right)\theta_i(k-1)\right] \\ &\quad+\mathrm{tr}\left[\sum_{i=1}^{s}\mu_i\omega_i\theta_i^{\mathrm{T}}(k-1)\left(A_{il}\tilde{X}_l(k-1)B_{il}+C_{il}X_l^{\mathrm{T}}(k-1)D_{il}\right)\right].\end{aligned} \tag{9}$$

将(9)两侧所有的$\left\|\tilde{X}_l(k)\right\|_F^2$，$l=[1,p]$相加，可得：

$$\begin{aligned}\sum_{l=1}^{p}\left\|\tilde{X}_l(k)\right\|_F^2 &\le \left\|\tilde{X}_l(k-1)\right\|_F^2+\sum_{i=1}^{s}\mu_i\omega_i\,\mathrm{tr}\left[\sum_{l=1}^{p}\left(B_{il}^{\mathrm{T}}\tilde{X}_l^{\mathrm{T}}(k-1)A_{il}^{\mathrm{T}}+D_{il}^{\mathrm{T}}\tilde{X}_l(k-1)C_{il}^{\mathrm{T}}\right)\theta_i(k-1)\right] \\ &\quad+\sum_{i=1}^{s}\mu_i\omega_i\,\mathrm{tr}\left[\sum_{l=1}^{p}\theta_i^{\mathrm{T}}(k-1)\left(A_{il}\tilde{X}_l(k-1)B_{il}+C_{il}X_l^{\mathrm{T}}(k-1)D_{il}\right)\right] \\ &\le \left\|\tilde{X}_l(k-1)\right\|_F^2-2\sum_{i=1}^{s}\mu_i\omega_i\left\|\theta_i(k-1)\right\|_F^2 \\ &\quad+s\sum_{i=1}^{s}\mu_i^2\omega_i^2\sum_{l=1}^{p}\left(\|A_{il}\|_2^2\|B_{il}\|_2^2+\|C_{il}\|_2^2\|D_{il}\|_2^2\right)\left\|\theta_i(k-1)\right\|_F^2 \\ &\le \sum_{l=1}^{P}\left\|\tilde{X}_l(0)\right\|_F^2-2\sum_{i=1}^{s}\mu_i\omega_i\left[1-\frac{s}{2}\mu_i\omega_i\sum_{l=1}^{p}\left(\|A_{il}\|_2^2\|B_{il}\|_2^2+\|C_{il}\|_2^2\|D_{il}\|_2^2\right)\right]\sum_{j=0}^{k}\left\|\theta_i(j)\right\|_F^2.\end{aligned}$$

显然，如果收敛因子μ_i满足(6)，我们可以推断出：

$$2\sum_{i=1}^{s}\mu_i\omega_i\left[1-\frac{s}{2}\mu_i\omega_i\sum_{l=1}^{p}\left(\|A_{il}\|_2^2\|B_{il}\|_2^2+\|C_{il}\|_2^2\|D_{il}\|_2^2\right)\right]\sum_{j=0}^{k}\left\|\theta_i(j)\right\|_F^2\le\sum_{l=1}^{P}\left\|\tilde{X}_l(0)\right\|_F^2。$$

从级数收敛的必要条件可知当$k\to\infty$时，

$$\lim_{k\to\infty}\left\|\theta_i(k)\right\|_F^2=0。$$

因此，可以得到：

$$\lim_{k\to\infty}\tilde{X}_j(k)=0，\quad j=[1,p]。$$

所以，MGI 算法生成的迭代解收敛于方程(1)的精确解，即

$$\lim_{k\to\infty}X_j(k)=X_j^{*}，\quad j=[1,p]。$$

定理证明完毕。

在证明了 MGI 算法收敛的充分性后，我们进一步分析了迭代因子μ_i与算法的收敛速度之间的关系，提出了**定理 3.2** 为算法的收敛提供了充分必要条件，扩展了迭代因子的范围，在保持算法稳定性的同时加快了算法的收敛速度，接下来提供了**定理 3.2** 的证明。

定理 3.2 假设耦合 Sylvester 转置矩阵方程(1)存在唯一解，则由 MGI 算法生成的迭代解

$X(k)=(X_1(k),X_2(k),\cdots,X_l(k))$在任意初始条件下都收敛于方程的唯一解$X_j^*$，$j=[1,p]$，即$\lim\limits_{k\to\infty}X_j(k)=X_j^*$，$j=[1,p]$，当且仅当迭代因子$\mu_i$满足不等式

$$0<\mu_i<\frac{2}{\left\|T^{\frac{1}{2}}S\right\|_2^2}, \tag{10}$$

证明 首先，用向量化算子对(8)的两侧进行列向量化，得到

$$\mathrm{vec}\left[\tilde{X}_l(k)\right]=\mathrm{vec}\left[\tilde{X}_l(k-1)\right]-\sum_{i=1}^{s}\mu_i\omega_i\left[B_{il}\otimes A_{il}^{\mathrm{T}}+P_{r_it_l}^{\mathrm{T}}\left(D_{il}\otimes C_{il}^{\mathrm{T}}\right)\right]$$

$$\sum_{j=1}^{p}\left[B_{ij}^{\mathrm{T}}\otimes A_{ij}+\left(D_{ij}^{\mathrm{T}}\otimes C_{ij}\right)P_{r_jt_j}\right]\mathrm{vec}\left[\tilde{X}_j(k-1)\right],\quad l=[1,p]。 \tag{11}$$

令

$$\mathrm{vec}\left[\tilde{X}(k)\right]=\left[\left[\mathrm{vec}\left(\tilde{X}_1(k)\right)\right]^{\mathrm{T}},\left[\mathrm{vec}\left(\tilde{X}_2(k)\right)\right]^{\mathrm{T}},\cdots,\left[\mathrm{vec}\left(\tilde{X}_p(k)\right)\right]^{\mathrm{T}}\right]^{\mathrm{T}},$$

定义$T=\mathrm{blkdiag}\left\{\omega_1I_{m_1n_1},\omega_2I_{m_2n_2},\cdots,\omega_sI_{m_sn_s}\right\}$，则(11)可以写成

$$\mathrm{vec}\left[\tilde{X}(k+1)\right]=\mathrm{vec}\left[\tilde{X}(k)\right]-\mu_iS^{\mathrm{T}}TS\,\mathrm{vec}\left[\tilde{X}(k)\right]=\left[I-\mu_iS^{\mathrm{T}}T^{\frac{1}{2}}T^{\frac{1}{2}}S\right]\mathrm{vec}\left[\tilde{X}(k)\right], \tag{12}$$

显然(12)是一个迭代方程，它收敛当且仅当

$$\rho\left(I-\mu_iS^{\mathrm{T}}T^{\frac{1}{2}}T^{\frac{1}{2}}S\right)<1,$$

因此，当且仅当迭代因子μ_i满足(10)，即(12)谱半径小于1，方程(1)具有唯一解。

定理证明完毕。

注释 3.1 在证明了我们所提出的 MGI 算法是收敛的之后，为了定量分析 MGI 算法的收敛速度，我们进行理论分析来估计其收敛阶数。收敛阶数描述了迭代误差随着迭代步数增加而减少的速率，能很好地衡量迭代算法的收敛效率。

假设迭代误差在第k步和第$(k+1)$步之间的关系可以表示为：

$$e_{(k+1)}=\alpha e_k^p,$$

其中e_k是第k步的误差，α是一个常数，p是收敛阶数。

对于 MGI 算法，我们可以通过分析迭代公式(5)来推导误差之间的关系。假设初始误差e_0已知，通过迭代公式，可以得到$e_1,e_2,\cdots,e_k$的表达式。通过数学归纳法，可以得到：

$$e_{k+1}=\alpha\left(\prod_{i=1}^{k}\mu_i\right)^p e_0^p,$$

其中μ_i是第i步的迭代因子。

为了使算法收敛，我们需要$\alpha\left(\prod\limits_{i=1}^{k}\mu_i\right)^p<1$对所有的$k$成立。即需要$p>1$，这表明算法具有超线性收敛性。

可以通过选择迭代因子μ_i来进一步确定p的值。例如选择$\mu_i=\frac{2}{i+2}$时，则$p=1.5$。此时，MGI 算法

具有 1.5 阶的收敛速度，这表明算法在每次迭代中能显著减少误差，即算法具有较好的收敛性。

3.3. 迭代因子的选择方法

在本节中，我们将讨论如何选择多迭代因子以优化算法的性能。迭代因子的选择对算法的收敛速度和稳定性有重要影响。我们提出以下几种方法来选择迭代因子：

3.3.1. 理论分析

基于算法的收敛性分析，选择满足**定理 3.2** 中不等式(10)的迭代因子。

3.3.2. 根据经验

通过数值实验，观察不同迭代因子对算法收敛速度的影响，并选择表现最佳的因子。

3.3.3. 优化方法

使用优化算法(如梯度下降法)来自动调整迭代因子，以最小化目标函数(3)。

具体方法可以根据实际问题的需求和计算资源来确定。

4. 数值例子

4.1. 数值实验验证算法收敛性

我们给出了一个数值例子来说明所提算法的收敛性。考虑以下耦合 Sylvester 转置矩阵方程，

$$\begin{cases} A_{11}X_1B_{11}+C_{11}X_1^{\mathrm{T}}D_{11}+A_{12}X_2B_{12}+C_{12}X_2^{\mathrm{T}}D_{12}=F_1, \\ A_{21}X_1B_{21}+C_{21}X_1^{\mathrm{T}}D_{21}+A_{22}X_2B_{22}+C_{22}X_2^{\mathrm{T}}D_{22}=F_2, \end{cases}$$

其中系数矩阵的取值来自于文献[16]：

$$A_{11}=\begin{bmatrix}-1 & 2\\ -2 & 3\end{bmatrix}, B_{11}=\begin{bmatrix}0 & -1\\ 0.5 & 3\end{bmatrix}, C_{11}=\begin{bmatrix}1 & -2\\ 3 & 3\end{bmatrix}, D_{11}=\begin{bmatrix}-2 & 0\\ 1 & 2\end{bmatrix},$$

$$A_{21}=\begin{bmatrix}2 & 3\\ 1 & 2\end{bmatrix}, B_{21}=\begin{bmatrix}4 & 1\\ 0.5 & -2\end{bmatrix}, C_{21}=\begin{bmatrix}1 & 2\\ 2 & 0\end{bmatrix}, D_{21}=\begin{bmatrix}0 & -2\\ -1 & -1\end{bmatrix},$$

$$A_{12}=\begin{bmatrix}1 & -3\\ 0 & 2\end{bmatrix}, \quad B_{12}=\begin{bmatrix}2 & 3\\ 1 & -2\end{bmatrix}, C_{12}=\begin{bmatrix}2.5 & -4\\ -2 & 0\end{bmatrix}, D_{12}=\begin{bmatrix}1 & 0\\ 3 & -1\end{bmatrix},$$

$$A_{22}=\begin{bmatrix}-1 & 0\\ 3 & 1\end{bmatrix}, \quad B_{22}=\begin{bmatrix}0 & 2\\ 2 & -2\end{bmatrix}, C_{22}=\begin{bmatrix}2 & 3\\ 1 & 2\end{bmatrix}, D_{22}=\begin{bmatrix}1 & -3\\ 2 & -1\end{bmatrix},$$

$$F_1=\begin{bmatrix}-26 & 35\\ -21 & -25\end{bmatrix}, \quad F_2=\begin{bmatrix}19 & -29\\ 3 & 11\end{bmatrix}.$$

此方程的精确解为：

$$X_1^*=\begin{bmatrix}1 & 2\\ -1 & 0\end{bmatrix}, X_2^*=\begin{bmatrix}2 & -2\\ 0 & 3\end{bmatrix}。$$

在本例中，我们将初始矩阵设为：

$$X_1^{(i)}(0)=X_2^{(i)}(0)=10^{-6}\times I_2, \quad i=1,2,$$

迭代误差设为：

$$\delta(k)=\sqrt{\frac{\left\|X_1(k)-X_1^*\right\|^2+\left\|X_2(k)-X_2^*\right\|^2}{\left\|X_1^*\right\|^2+\left\|X_2^*\right\|^2}}。$$

接下来，我们对所提出的 MGI 算法的收敛性进行了验证。首先，我们设定参数 $\omega_1 = 0.4, \omega_2 = 0.6$，选定两个参数后，迭代因子的选择方法我们结合上一节的 3.3.1 和 3.3.2 的两种方法，图 1 中选取的三组迭代因子为：$\mu_1 = 2\times10^{-5}, \mu_2 = 1.5\times10^{-5}; \mu_1 = 3\times10^{-5}, \mu_2 = 2\times10^{-5}; \mu_1 = 1.5\times10^{-5}, \mu_2 = 1\times10^{-5}$。

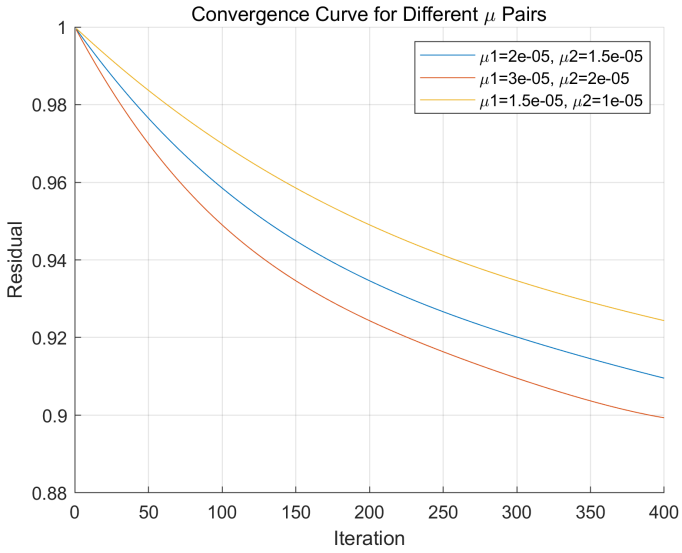

Figure 1. Iteration error
图 1. 迭代误差

从图 1 可以看出，随着迭代步数的增加，迭代误差持续减小，这表明迭代解正在逐渐接近精确解，从而证明了本文所提出的算法的收敛性。此外，我们发现迭代因子的数值越大，算法的收敛速度越快，然而，如果迭代因子选取过大，算法将直接发散。因此，如何选择最佳的迭代因子以优化收敛性能，仍需进一步研究。

4.2. MGI 算法的优势

1) 收敛速度更快：通过引入多个迭代因子，MGI 算法能够更灵活地调整迭代步长，从而加速收敛。

2) 计算复杂度更低：MGI 算法避免了复杂的矩阵分解操作，仅需进行矩阵乘法和加法运算，因此计算复杂度远低于克罗内克积的直接解法。

3) 适用范围更广：MGI 算法不仅适用于方阵，还可以处理非方阵情况，扩展了算法的应用范围。

5. 结论

本文针对耦合 Sylvester 转置矩阵方程的数值求解问题，引入了多个参数，并提出了一种多迭代因子梯度算法。通过分析矩阵范数之间的递推关系以及系数矩阵列向量化后的谱半径，推导出了确保算法生成的迭代解对于任意初始矩阵都能收敛到精确解的充分条件和充分必要条件。此外，我们还提供了一个具体的数值例子来验证算法的有效性和理论结果的正确性。值得注意的是，不同参数的选择对收敛速率有着显著的影响。因此，如何选取最优参数以提高收敛速率，仍是一个需要进一步研究的问题。

参考文献

[1] Costa, O.L.V. and Fragoso, M.D. (1993) Stability Results for Discrete-Time Linear Systems with Markovian Jumping Parameters. *Journal of Mathematical Analysis and Applications*, **179**, 154-178. https://doi.org/10.1006/jmaa.1993.1341

[2] Zhou, B. (2015) Global Stabilization of Periodic Linear Systems by Bounded Controls with Applications to Spacecraft Magnetic Attitude Control. *Automatica*, **60**, 145-154. https://doi.org/10.1016/j.automatica.2015.07.003

[3] Ding, F. (2023) Least Squares Parameter Estimation and Multi-Innovation Least Squares Methods for Linear Fitting Problems from Noisy Data. *Journal of Computational and Applied Mathematics*, **426**, Article ID: 115107. https://doi.org/10.1016/j.cam.2023.115107

[4] Zhou, B. (2020) Finite-Time Stabilization of Linear Systems by Bounded Linear Time-Varying Feedback. *Automatica*, **113**, Article ID: 108760. https://doi.org/10.1016/j.automatica.2019.108760

[5] Dai, L. (1989) Singular Control Systems. Springer.

[6] Ding, F. and Chen, T. (2005) Iterative Least-Squares Solutions of Coupled Sylvester Matrix Equations. *Systems & Control Letters*, **54**, 95-107. https://doi.org/10.1016/j.sysconle.2004.06.008

[7] Ding, F., Liu, P.X. and Ding, J. (2008) Iterative Solutions of the Generalized Sylvester Matrix Equations by Using the Hierarchical Identification Principle. *Applied Mathematics and Computation*, **197**, 41-50. https://doi.org/10.1016/j.amc.2007.07.040

[8] Xie, L., Ding, J. and Ding, F. (2009) Gradient Based Iterative Solutions for General Linear Matrix Equations. *Computers & Mathematics with Applications*, **58**, 1441-1448. https://doi.org/10.1016/j.camwa.2009.06.047

[9] Niu, Q., Wang, X. and Lu, L. (2010) A Relaxed Gradient Based Algorithm for Solving Sylvester Equations. *Asian Journal of Control*, **13**, 461-464. https://doi.org/10.1002/asjc.328

[10] Xie, Y. and Ma, C. (2016) The Accelerated Gradient Based Iterative Algorithm for Solving a Class of Generalized Sylvester-Transpose Matrix Equation. *Applied Mathematics and Computation*, **273**, 1257-1269. https://doi.org/10.1016/j.amc.2015.07.022

[11] Li, Z., Wang, Y., Zhou, B. and Duan, G. (2010) Least Squares Solution with the Minimum-Norm to General Matrix Equations via Iteration. *Applied Mathematics and Computation*, **215**, 3547-3562. https://doi.org/10.1016/j.amc.2009.10.052

[12] Song, C., Chen, G. and Zhao, L. (2011) Iterative Solutions to Coupled Sylvester-Transpose Matrix Equations. *Applied Mathematical Modelling*, **35**, 4675-4683. https://doi.org/10.1016/j.apm.2011.03.038

[13] Al Zhour, Z. and Kiliçman, A. (2007) Some New Connections between Matrix Products for Partitioned and Non-Partitioned Matrices. *Computers & Mathematics with Applications*, **54**, 763-784. https://doi.org/10.1016/j.camwa.2006.12.045

[14] Wang, W. and Song, C. (2021) Iterative Algorithms for Discrete-Time Periodic Sylvester Matrix Equations and Its Application in Antilinear Periodic System. *Applied Numerical Mathematics*, **168**, 251-273. https://doi.org/10.1016/j.apnum.2021.06.006

[15] Khalil, I., Doyle, J. and Glover, K. (1996) Robust and Optimal Control. Vol. 2, Prentice Hall.

[16] Huang, B. and Ma, C. (2022) On the Relaxed Gradient-Based Iterative Methods for the Generalized Coupled Sylvester-Transpose Matrix Equations. *Journal of the Franklin Institute*, **359**, 10688-10725. https://doi.org/10.1016/j.jfranklin.2022.07.051