连续 Sylvester 矩阵方程求解的分裂迭代算法*

李 英

(商丘师范学院 信息技术学院, 河南 商丘 476000)

摘要: 有效求解连续的 Sylvester 矩阵方程对于科学和工程计算有着重要的应用价值,因此该文提出了一种可行的分裂迭代算法.该算法的核心思想是外迭代将连续 Sylvester 矩阵方程的系数矩阵分裂为对称矩阵和反对称矩阵,内迭代求解复对称矩阵方程.相较于传统的分裂算法,该文所提出的分裂迭代算法有效地避免了最优迭代参数的选取,并利用了复对称方程组高效求解的特点,进而提高了算法的易实现性、易操作性.此外,从理论层面进一步证明了该分裂迭代算法的收敛性.最后,通过数值算例表明分裂迭代算法具有良好的收敛性和鲁棒性,同时也证实了分裂迭代算法的收敛性很大程度依赖于内迭代格式的选取.

关 键 词: Sylvester 矩阵方程; 复对称矩阵方程; 分裂迭代算法; 收敛性
中图分类号: O246 **文献标志码:** A DOI: 10.21656/1000-0887.400133

引 言

考虑如下连续的 Sylvester 矩阵方程:

$$\boldsymbol{AX} + \boldsymbol{XB} = \boldsymbol{F}, \tag{1}$$

其中 $\boldsymbol{A} \in C^{m\times m}$,$\boldsymbol{B} \in C^{n\times n}$,$\boldsymbol{F} \in C^{m\times n}$,且 $\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{F}$ 均为稀疏矩阵,$\boldsymbol{A}$,$\boldsymbol{B}$ 为正定矩阵.作为矩阵方程中极其重要的一类方程,连续 Sylvester 方程在工程计算等诸多领域都有着广泛的应用,如控制与系统理论[1-2]、图形恢复[3]、信号处理[4]等方面,因此求解连续 Sylvester 方程具有非常重要的实际工程意义和学术价值.

对于连续 Sylvester 方程(1)的求解,目前主要有两种方式进行处理,第一种是将 Sylvester 矩阵方程(1)进行拉直,转化为形如 $\tilde{\boldsymbol{A}}\boldsymbol{x} = \boldsymbol{b}$ 的形式求解,其中 $\tilde{\boldsymbol{A}} = \boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{B}^{\mathrm{T}}$,$\boldsymbol{I}_n$,$\boldsymbol{I}_m$ 分别是阶数为 n 和 m 的单位矩阵;第二种是直接对 Sylvester 方程(1)进行迭代求解.然而当 Sylvester 方程(1)的阶数较大时,会导致线性方程系数矩阵 $\tilde{\boldsymbol{A}}$ 的维数急剧增大,一方面会引发病态方程的产生,另一方面会造成计算机存储量的增大,计算时间的增加.因此,直接对连续的 Sylvester 矩阵方程(1)的求解成为了当前众多学者关注的热点.

目前直接法和迭代法是求解连续 Sylvester 矩阵方程(1)的主要工具.直接法主要用于矩阵

* **收稿日期:** 2019-04-04; **修订日期:** 2019-06-19
作者简介: 李英(1981—),女,实验师,硕士(E-mail: sqsyjkx@126.com).
引用格式: 李英. 连续 Sylvester 矩阵方程求解的分裂迭代算法[J]. 应用数学和力学, 2020, **41**(1): 115-124.

方程(1)阶数较小的情况,如 Bartels-Stewart 和 Hessenberg-Schur 方法[5-6].迭代法一般用于矩阵阶数较大或系数矩阵为稀疏矩阵的情况.近些年,众多学者基于系数矩阵分裂的思想提出了大量有效可行的算法[7-13]:如 Bai 提出了求解连续 Sylvester 方程(1)的 HSS 迭代算法(Hermitian and skew-Hermitian splitting),该算法是求解线性方程 $\tilde{\boldsymbol{A}}\boldsymbol{x}=\boldsymbol{b}$ 的 HSS 算法的拓展[7].为了提高 HSS 算法的收敛性,修正 HSS(MHSS)算法避免了 HSS 算法每次迭代求解复线性矩阵方程的过程,而只需在每次迭代中求解两个实线性子系统,进而大大加速了算法的收敛性,减少迭代时间[8].之后,由于预处理技术的引入,相继提出预处理 HSS(PHSS)算法、非交错 PHSS(NPHSS)算法、非精确 PHSS(IPHSS)算法以及非精确 NPHSS(INPHSS)算法[9],大大改善了原 HSS 算法的收敛速度.然而,上述的分裂迭代算法均面临一个最优迭代参数选取的问题,并且由于最优迭代参数的选取并没有一个统一的公式去定性计算,这在一定程度上增加了此类分裂迭代算法计算的复杂性.

为了避免最优迭代参数的选取,本文借鉴上述分裂迭代算法的思想,拟采用内外迭代进行处理,通过将连续的 Sylvester 矩阵方程(1)的系数矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 分裂为对称矩阵和反对称矩阵,使内迭代过程转化为复对称矩阵方程组的求解.而近些年来对于复对称线性方程组的求解已有大量丰富的学术成果,从宏观角度出发,主要分为分裂迭代法[14-19]和 Krylov 子空间迭代法[20-24]两大类.但基于最优迭代参数选取的考虑,现主要针对 Krylov 子空间迭代法进行研究,如共轭正交共轭梯度(COCG)法[20-21]、共轭 A-正交共轭残量(COCR)法[21-22]、复对称矩阵的双共轭梯度(SCBiCG(Γ, m)) 法[23]以及复对称矩阵的双共轭残量(SCBiCR(Γ, m)) 法[24]等 Krylov 子空间迭代法.COCG 法和 COCR 法分别是共轭梯度(CG)法和共轭残量(CR)法的延拓,但在实际应用中由于 COCG 法有时会导致不规则的收敛行为,因此 COCR 法相较于 COCG 法具有更好的鲁棒性.SCBiCG(Γ, m) 法和 SCBiCR(Γ, m) 法分别由双共轭梯度(BiCG)法和双共轭残量(BiCR)法推导得到,但相较于 SCBiCG(Γ, m) 法,SCBiCR(Γ, m) 法克服了 SCBiCG(Γ, m) 法需要确定系数 c_i 的不足,大大提高了收敛速度.此外,丰富的预处理技术应用也大大加快了 Krylov 子空间迭代法的收敛速度[21].

基于上述的调研,本文主要针对连续的 Sylvester 矩阵方程(1)的求解,提出了一种新型的分裂迭代算法,该算法采用内外迭代方式进行求解,外迭代基于分裂思想,将系数矩阵 $\boldsymbol{A}$,$\boldsymbol{B}$ 分裂为一个对称矩阵和一个反对称矩阵,该分裂方式区别于 HSS 一类方法的分裂方式(Hermitian 矩阵和反 Hermitian 矩阵);内迭代采用复对称线性方程组的求解算法,有效地避免了最优迭代参数的选取.基于这样一种新型的分裂迭代算法,可有效地加快连续的 Sylvester 矩阵方程(1)的收敛速度.

文中内积定义:若 $\boldsymbol{X},\boldsymbol{Y}\in C^{m\times n}$,则其内积为 $\langle \boldsymbol{X},\boldsymbol{Y}\rangle=\mathrm{tr}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{Y})$,由此导出矩阵的 Frobenius 范数为 $\|\boldsymbol{X}\|_{\mathrm{F}}=\sqrt{\langle \boldsymbol{X},\boldsymbol{X}\rangle}$.$\boldsymbol{X}^{\mathrm{T}}$ 表示 $\boldsymbol{X}$ 的转置矩阵,$\bar{\boldsymbol{X}}$ 表示 $\boldsymbol{X}$ 的共轭矩阵,$\boldsymbol{X}^{\mathrm{H}}$ 表示 $\boldsymbol{X}$ 的共轭转置矩阵,$\rho(\boldsymbol{X})$ 表示矩阵 $\boldsymbol{X}$ 的谱半径,$\vec{\boldsymbol{X}}$ 表示矩阵 $\boldsymbol{X}$ 的拉直,$\boldsymbol{X}\otimes\boldsymbol{Y}$ 表示矩阵 $\boldsymbol{X}$ 与 $\boldsymbol{Y}$ 的直积或 Kronecker 积.若 $x\in\mathbf{C}$,$\mathrm{Re}(x)$ 表示 x 的实部,$\mathrm{Im}(x)$ 表示 x 的虚部.

1 分裂迭代算法

针对连续 Sylvester 矩阵方程(1)的求解,拟基于分裂迭代思想,采用内外迭代格式进行处理,其中外迭代主要对系数矩阵进行分裂,内迭代执行复对称矩阵方程组的求解,以下给出详细的推导过程.

1.1 外迭代格式

首先将连续 Sylvester 矩阵方程(1)的系数矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 分裂为

$$\boldsymbol{A} = \boldsymbol{M}_1 - \boldsymbol{N}_1,\ \boldsymbol{B} = \boldsymbol{M}_2 - \boldsymbol{N}_2, \tag{2}$$

其中

$$\boldsymbol{M}_1 = (\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{A})/2,\ \boldsymbol{N}_1 = (\boldsymbol{A}^{\mathrm{T}} - \boldsymbol{A})/2,\ \boldsymbol{M}_2 = (\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{B})/2,\ \boldsymbol{N}_2 = (\boldsymbol{B}^{\mathrm{T}} - \boldsymbol{B})/2, \tag{3}$$

满足 $\boldsymbol{M}_1 = \boldsymbol{M}_1^{\mathrm{T}}, \boldsymbol{M}_2 = \boldsymbol{M}_2^{\mathrm{T}}, \boldsymbol{N}_1 = -\boldsymbol{N}_1^{\mathrm{T}}, \boldsymbol{N}_2 = -\boldsymbol{N}_2^{\mathrm{T}}$.故矩阵方程(1)等价于

$$\boldsymbol{M}_1\boldsymbol{X} + \boldsymbol{X}\boldsymbol{M}_2 = \boldsymbol{N}_1\boldsymbol{X} + \boldsymbol{X}\boldsymbol{N}_2 + \boldsymbol{F}. \tag{4}$$

然后根据矩阵方程(4),建立相应的分裂迭代格式.

给定初始矩阵 $\boldsymbol{X}_0 \in C^{m\times n}$, 外迭代格式如下:

$$\boldsymbol{M}_1\boldsymbol{X}_{k+1} + \boldsymbol{X}_{k+1}\boldsymbol{M}_2 = \boldsymbol{N}_1\boldsymbol{X}_k + \boldsymbol{X}_k\boldsymbol{N}_2 + \boldsymbol{F}. \tag{5}$$

令 $\tilde{\boldsymbol{F}}_k = \boldsymbol{N}_1\boldsymbol{X}_k + \boldsymbol{X}_k\boldsymbol{N}_2 + \boldsymbol{F}$, 则有

$$\boldsymbol{M}_1\boldsymbol{X}_{k+1} + \boldsymbol{X}_{k+1}\boldsymbol{M}_2 = \tilde{\boldsymbol{F}}_k, \tag{6}$$

计算 $\boldsymbol{X}_{k+1} \in C^{m\times n}, k = 0,1,2,\cdots$, 直至迭代序列 $\{\boldsymbol{X}_k\}_{k=0}^{\infty}$ 收敛.

通过上述外迭代格式的分析,可知外迭代过程已将连续 Sylvester 矩阵方程(1)的求解转化为复对称矩阵方程组(6)的求解,相应地,内迭代过程主要针对这一问题进行处理.

1.2 内迭代格式

基于文献调研,大量的研究成果主要针对复对称线性方程组 $\tilde{\boldsymbol{A}}\boldsymbol{x} = \boldsymbol{b}$ 的形式进行求解,其中 $\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{A}}^{\mathrm{T}}$, 故为了有效求解复对称矩阵方程(6),本文拟通过线性变换形式,将求解 $\tilde{\boldsymbol{A}}\boldsymbol{x} = \boldsymbol{b}$ 的可行算法进行延拓.

1.2.1 复对称线性变换

首先给出复对称线性变换的定义.

定义 1 设 $\boldsymbol{T}$ 是酉空间 V 上的线性变换,对于任意 $\boldsymbol{X},\boldsymbol{Y} \in V$, 满足

$$\langle \boldsymbol{T}(\bar{\boldsymbol{X}}), \boldsymbol{Y}\rangle = \overline{\langle \boldsymbol{X}, \boldsymbol{T}(\bar{\boldsymbol{Y}})\rangle}, \tag{7}$$

则称 $\boldsymbol{T}$ 为酉空间 V 中一个对称变换.

根据复对称线性变换的定义,给出相应的复对称线性变换方程:

$$\boldsymbol{T}(\boldsymbol{X}) = \boldsymbol{C}, \tag{8}$$

其中 $\boldsymbol{T}(\boldsymbol{X}) = \boldsymbol{M}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{N}$, $\boldsymbol{M} \in C^{m\times m}, \boldsymbol{N} \in C^{n\times n}, \boldsymbol{X} \in C^{m\times n}$,且 $\boldsymbol{M} = \boldsymbol{M}^{\mathrm{T}}, \boldsymbol{N} = \boldsymbol{N}^{\mathrm{T}}$.

下面验证酉空间 V 上的线性变换 $\boldsymbol{T}(\boldsymbol{X}) = \boldsymbol{M}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{N}$ 为复对称变换.

根据酉空间的内积定义满足 $\langle \boldsymbol{X},\boldsymbol{Y}\rangle = \mathrm{tr}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{Y}) = \mathrm{tr}(\boldsymbol{Y}\boldsymbol{X}^{\mathrm{H}}) = \overline{\mathrm{tr}(\boldsymbol{Y}^{\mathrm{H}}\boldsymbol{X})} = \overline{\mathrm{tr}(\boldsymbol{X}\boldsymbol{Y}^{\mathrm{H}})}$, 则

$$\begin{aligned}\langle \boldsymbol{T}(\bar{\boldsymbol{X}}),\boldsymbol{Y}\rangle &= \langle \boldsymbol{M}\bar{\boldsymbol{X}} + \bar{\boldsymbol{X}}\boldsymbol{N},\boldsymbol{Y}\rangle = \mathrm{tr}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{M}^{\mathrm{H}}\boldsymbol{Y} + \boldsymbol{N}^{\mathrm{H}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}) = \mathrm{tr}(\boldsymbol{X}^{\mathrm{T}}\bar{\boldsymbol{M}}\boldsymbol{Y}) + \mathrm{tr}(\bar{\boldsymbol{N}}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{Y}) = \\ &\overline{\mathrm{tr}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{M}\bar{\boldsymbol{Y}})} + \overline{\mathrm{tr}(\boldsymbol{X}^{\mathrm{H}}\bar{\boldsymbol{Y}}\boldsymbol{N})} = \overline{\mathrm{tr}(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{M}\bar{\boldsymbol{Y}} + \boldsymbol{X}^{\mathrm{H}}\bar{\boldsymbol{Y}}\boldsymbol{N})} = \\ &\overline{\mathrm{tr}(\boldsymbol{X}^{\mathrm{H}}(\boldsymbol{M}\bar{\boldsymbol{Y}} + \bar{\boldsymbol{Y}}\boldsymbol{N}))} = \overline{\langle \boldsymbol{X},\boldsymbol{T}(\bar{\boldsymbol{Y}})\rangle},\end{aligned}$$

故线性变换 $\boldsymbol{T}(\boldsymbol{X}) = \boldsymbol{M}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{N}$ 为酉空间 V 中一个对称变换.

1.2.2 算法延拓

为了提升分裂迭代算法的收敛性,这里仅针对文献中收敛性较好的 3 种算法进行代表性阐明拓展,即 COCR 算法[21-22]、COCG 算法[20-21]以及 CCC of SCBiCR 算法(简记为 CCC_SCBiCR 算法)[24].如下给出求解复对称线性变换方程(8)的 3 种拓展算法.

首先给出求解复对称线性变换方程(8)的 COCR 算法.

第 1 步 给定初始矩阵 $\boldsymbol{X}_0 \in C^{m\times n}$, 计算 $\boldsymbol{R}_0=\boldsymbol{C}-\boldsymbol{T}(\boldsymbol{X}_0)$, 置 $\boldsymbol{P}_0=\boldsymbol{R}_0, \boldsymbol{U}_0=\boldsymbol{T}(\boldsymbol{P}_0), \boldsymbol{S}_0=\boldsymbol{U}_0$, $k:=0$.

第 2 步 若 $\|\boldsymbol{R}_k\|_F \leqslant \varepsilon\|\boldsymbol{R}_0\|_F$, 计算停止,其中 $\varepsilon \in \mathbf{R}_+$; 否则,计算

$$\alpha_k=\langle\bar{\boldsymbol{R}}_k,\boldsymbol{S}_k\rangle/\langle\bar{\boldsymbol{U}}_k,\boldsymbol{U}_k\rangle,\ \boldsymbol{X}_{k+1}=\boldsymbol{X}_k+\alpha_k\boldsymbol{P}_k,\ \boldsymbol{R}_{k+1}=\boldsymbol{R}_k-\alpha_k\boldsymbol{U}_k,\ \boldsymbol{S}_{k+1}=\boldsymbol{T}(\boldsymbol{R}_{k+1}),$$

$$\beta_k=\langle\bar{\boldsymbol{R}}_{k+1},\boldsymbol{S}_{k+1}\rangle/\langle\bar{\boldsymbol{R}}_k,\boldsymbol{S}_k\rangle,\ \boldsymbol{P}_{k+1}=\boldsymbol{R}_{k+1}+\beta_k\boldsymbol{P}_k,\ \boldsymbol{U}_{k+1}=\boldsymbol{S}_{k+1}+\beta_k\boldsymbol{U}_k.$$

第 3 步 置 $k:=k+1$, 转第 2 步.

其次给出求解复对称线性变换方程(8)的 COCG 算法.

第 1 步 给定初始矩阵 $\boldsymbol{X}_0 \in C^{m\times n}$,计算 $\boldsymbol{R}_0=\boldsymbol{C}-\boldsymbol{T}(\boldsymbol{X}_0)$,置 $\boldsymbol{P}_0=\boldsymbol{R}_0, k:=0$.

第 2 步 若 $\|\boldsymbol{R}_k\|_F \leqslant \varepsilon\|\boldsymbol{R}_0\|_F$, 计算停止,其中 $\varepsilon \in \mathbf{R}_+$; 否则,计算

$$\boldsymbol{U}_k=\boldsymbol{T}(\boldsymbol{R}_k),\ \alpha_k=\langle\bar{\boldsymbol{R}}_k,\boldsymbol{R}_k\rangle/\langle\bar{\boldsymbol{P}}_k,\boldsymbol{U}_k\rangle,\ \boldsymbol{X}_{k+1}=\boldsymbol{X}_k+\alpha_k\boldsymbol{P}_k,\ \boldsymbol{R}_{k+1}=\boldsymbol{R}_k-\alpha_k\boldsymbol{U}_k,$$

$$\beta_k=\langle\bar{\boldsymbol{R}}_{k+1},\boldsymbol{R}_{k+1}\rangle/\langle\bar{\boldsymbol{R}}_k,\boldsymbol{R}_k\rangle,\ \boldsymbol{P}_{k+1}=\boldsymbol{R}_{k+1}+\beta_k\boldsymbol{P}_k.$$

第 3 步 置 $k:=k+1$, 转第 2 步.

最后给出求解复对称线性变换方程(8)的 CCC_SCBiCR 算法.

第 1 步 给定初始矩阵 $\boldsymbol{X}_0 \in C^{m\times n}$, 计算 $\boldsymbol{R}_0=\boldsymbol{C}-\boldsymbol{T}(\boldsymbol{X}_0)$, 置 $\boldsymbol{P}_0=\boldsymbol{R}_0, \boldsymbol{U}_0=\boldsymbol{T}(\boldsymbol{R}_0), \boldsymbol{S}_0=\boldsymbol{U}_0$, $k:=0$.

第 2 步 若 $\|\boldsymbol{R}_k\|_F \leqslant \varepsilon\|\boldsymbol{R}_0\|_F$, 计算停止,其中 $\varepsilon \in \mathbf{R}_+$; 否则,计算

$$\boldsymbol{V}_k=\boldsymbol{T}(\boldsymbol{S}_k),\ \alpha_k=\langle\bar{\boldsymbol{U}}_k,\boldsymbol{R}_k\rangle/\langle\bar{\boldsymbol{S}}_k,\boldsymbol{S}_k\rangle,\ \boldsymbol{X}_{k+1}=\boldsymbol{X}_k+\alpha_k\boldsymbol{P}_k,\ \boldsymbol{R}_{k+1}=\boldsymbol{R}_k-\alpha_k\boldsymbol{S}_k,$$

$$\boldsymbol{U}_{k+1}=\boldsymbol{U}_k-\alpha_k\boldsymbol{V}_k,\ \zeta_k=(\langle\bar{\boldsymbol{U}}_k,\boldsymbol{R}_k\rangle-\alpha_k\langle\bar{\boldsymbol{S}}_k,\boldsymbol{S}_k\rangle)/(\alpha_k\langle\bar{\boldsymbol{V}}_k,\boldsymbol{S}_k\rangle-\langle\bar{\boldsymbol{U}}_k,\boldsymbol{U}_k\rangle),$$

$$\beta_k=(\langle\bar{\boldsymbol{U}}_{k+1},\boldsymbol{R}_{k+1}\rangle+\zeta_k\langle\bar{\boldsymbol{U}}_{k+1},\boldsymbol{U}_{k+1}\rangle)/(\langle\bar{\boldsymbol{U}}_k,\boldsymbol{R}_k\rangle+\zeta_k\langle\bar{\boldsymbol{U}}_k,\boldsymbol{U}_k\rangle),$$

$$\boldsymbol{P}_{k+1}=\boldsymbol{R}_{k+1}+\beta_k\boldsymbol{P}_k,\ \boldsymbol{S}_{k+1}=\boldsymbol{U}_{k+1}+\beta_k\boldsymbol{S}_k.$$

第 3 步 置 $k:=k+1$, 转第 2 步.

1.3 分裂迭代算法

通过内外迭代格式的推导,如下给出整体的分裂迭代算法.

第 1 步 给定一个初始矩阵 $\boldsymbol{X}_0 \in C^{m\times n}$,通过式(3),计算 $\boldsymbol{M}_1,\boldsymbol{M}_2,\boldsymbol{N}_1,\boldsymbol{N}_2,\boldsymbol{R}_0=\boldsymbol{F}-\boldsymbol{A}\boldsymbol{X}_0-\boldsymbol{X}_0\boldsymbol{B}$,置 $k:=0$.

第 2 步(外迭代) 若 $\|\boldsymbol{R}_k\|_F \leqslant \varepsilon\|\boldsymbol{R}_0\|_F$, 计算停止,其中 $\varepsilon \in \mathbf{R}_+$; 否则,计算

$$\tilde{\boldsymbol{F}}_k=\boldsymbol{N}_1\boldsymbol{X}_k+\boldsymbol{X}_k\boldsymbol{N}_2+\boldsymbol{F}.$$

第 3 步(内迭代) 采用 COCR、COCG 及 CCC_SCBiCR 中任意一种方法求解

$$\boldsymbol{M}_1\boldsymbol{X}_{k+1}+\boldsymbol{X}_{k+1}\boldsymbol{M}_2=\tilde{\boldsymbol{F}}_k,$$

计算 $\boldsymbol{R}_{k+1}=\boldsymbol{F}-\boldsymbol{A}\boldsymbol{X}_{k+1}-\boldsymbol{X}_{k+1}\boldsymbol{B}$.

第 4 步 置 $k:=k+1$, 转第 2 步.

为了后续描述方便,采用不同内迭代方法的分裂迭代算法分别记作 TS_COCR 算法、TS_COCG 算法以及 TS_CCC_SCBiCR 算法.

2 收敛性分析

针对文中所提出的分裂迭代算法,给出其相应的收敛性质.

定理 1 假定 $A \in C^{m\times m}$ 是正定矩阵,且 $\boldsymbol{A}\boldsymbol{M}-\boldsymbol{N}$,其中 $\boldsymbol{M}=(\boldsymbol{A}^{\mathrm{T}}+\boldsymbol{A})/2, \boldsymbol{N}=(\boldsymbol{A}^{\mathrm{T}}-\boldsymbol{A})/2$.

若对于任意的 $\boldsymbol{x} \in C^m$ 且 $\boldsymbol{x} \neq \boldsymbol{0}$,满足 $\mathrm{Re}(\boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{A}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{x}) > 0$,则 $\rho(\boldsymbol{M}^{-1}\boldsymbol{N}) < 1$.

证明 设 $\boldsymbol{M}^{-1}\boldsymbol{N}$ 的特征值为 λ,相应的特征向量为 $\boldsymbol{x}$,则满足 $\boldsymbol{M}^{-1}\boldsymbol{N}\boldsymbol{x} = \lambda\boldsymbol{x}$,等价于 $\lambda\boldsymbol{M}\boldsymbol{x} = \boldsymbol{N}\boldsymbol{x}$.故 $|\lambda| = |\boldsymbol{x}^{\mathrm{H}}\boldsymbol{N}\boldsymbol{x}|/|\boldsymbol{x}^{\mathrm{H}}\boldsymbol{M}\boldsymbol{x}|$.又由于 $\boldsymbol{M} = \boldsymbol{M}^{\mathrm{T}}, \boldsymbol{N} = -\boldsymbol{N}^{\mathrm{T}}$, 所以

$$|\lambda|^2 = \frac{|\boldsymbol{x}^{\mathrm{H}}\boldsymbol{N}\boldsymbol{x}|^2}{|\boldsymbol{x}^{\mathrm{H}}\boldsymbol{M}\boldsymbol{x}|^2} = \frac{(\boldsymbol{x}^{\mathrm{H}}\boldsymbol{N}\boldsymbol{x})^{\mathrm{H}}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{N}\boldsymbol{x}}{(\boldsymbol{x}^{\mathrm{H}}\boldsymbol{M}\boldsymbol{x})^{\mathrm{H}}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{M}\boldsymbol{x}} = -\frac{\boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{N}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{N}\boldsymbol{x}}{\boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{M}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{M}\boldsymbol{x}}. \tag{9}$$

而若要满足 $|\lambda|^2 < 1$, 则需满足 $\boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{M}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{M}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{N}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{N}\boldsymbol{x} > 0$, 而

$$\begin{aligned}
&\boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{M}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{M}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{N}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{N}\boldsymbol{x} = \\
&\quad \frac{1}{4}(\boldsymbol{x}^{\mathrm{H}}(\boldsymbol{A}^{\mathrm{H}} + \bar{\boldsymbol{A}})\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}(\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{A})\boldsymbol{x} + \boldsymbol{x}^{\mathrm{H}}(\boldsymbol{A}^{\mathrm{H}} - \bar{\boldsymbol{A}})\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}(\boldsymbol{A}^{\mathrm{T}} - \boldsymbol{A})\boldsymbol{x}) = \\
&\quad \frac{1}{2}(\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}^{\mathrm{H}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{A}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{x}) = \\
&\quad \frac{1}{2}((\boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{A}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{x})^{\mathrm{H}} + \boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{A}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{x}) = \\
&\quad \mathrm{Re}(\boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{A}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{x}),
\end{aligned}$$

则如果对任意的 $\boldsymbol{x} \in C^m$ 且 $\boldsymbol{x} \neq \boldsymbol{0}$, 满足 $\mathrm{Re}(\boldsymbol{x}^{\mathrm{H}}\bar{\boldsymbol{A}}\boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{x}) > 0$, 则 $\rho(\boldsymbol{M}^{-1}\boldsymbol{N}) < 1$.证毕. □

定理 2 假定连续的 Sylvester 方程 $\boldsymbol{AX} + \boldsymbol{XB} = \boldsymbol{F}$ 中的 $\boldsymbol{A} \in C^{m\times m}$,$\boldsymbol{B} \in C^{n\times n}$ 是正定矩阵,且 $\boldsymbol{A} = \boldsymbol{M}_1 - \boldsymbol{N}_1$,$\boldsymbol{B} = \boldsymbol{M}_2 - \boldsymbol{N}_2$,其中 $\boldsymbol{M}_1 = (\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{A})/2$,$\boldsymbol{N}_1 = (\boldsymbol{A}^{\mathrm{T}} - \boldsymbol{A})/2$,$\boldsymbol{M}_2 = (\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{B})/2$,$\boldsymbol{N}_2 = (\boldsymbol{B}^{\mathrm{T}} - \boldsymbol{B})/2$.若对任意的 $\boldsymbol{X} \in C^{m\times n}$ 且 $\boldsymbol{X} \neq \boldsymbol{0}$,满足 $\mathrm{Re}(\langle \boldsymbol{X},\bar{\boldsymbol{A}}\boldsymbol{X} + \boldsymbol{X}\bar{\boldsymbol{B}}\rangle\langle \boldsymbol{X},\boldsymbol{AX} + \boldsymbol{XB}\rangle) > 0$,则 $\rho((\boldsymbol{M}_1 \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{M}_2)^{-1}(\boldsymbol{N}_1 \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{N}_2)) < 1$.

证明 将连续的 Sylvester 方程 $\boldsymbol{AX} + \boldsymbol{XB} = \boldsymbol{F}$ 两端拉直,可得

$$(\boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{B}^{\mathrm{T}})\vec{\boldsymbol{X}} = \vec{\boldsymbol{F}}, \tag{10}$$

其中 $\vec{\boldsymbol{X}},\vec{\boldsymbol{F}} \in C^{mn}$.由 $\boldsymbol{A} = \boldsymbol{M}_1 - \boldsymbol{N}_1$,$\boldsymbol{B} = \boldsymbol{M}_2 - \boldsymbol{N}_2$ 可知,方程(10)等价于

$$(\boldsymbol{M}_1 \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{M}_2)\vec{\boldsymbol{X}} = (\boldsymbol{N}_1 \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{N}_2)\vec{\boldsymbol{X}} + \vec{\boldsymbol{F}}. \tag{11}$$

令 $\tilde{\boldsymbol{A}} = \boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{B}^{\mathrm{T}}$,$\tilde{\boldsymbol{M}} = \boldsymbol{M}_1 \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{M}_2$,$\tilde{\boldsymbol{N}} = \boldsymbol{N}_1 \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{N}_2$, 则满足

$$\begin{aligned}
&\tilde{\boldsymbol{M}} = \boldsymbol{M}_1 \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{M}_2 = \frac{\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{A}}{2} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \frac{\boldsymbol{B}^{\mathrm{T}} + \boldsymbol{B}}{2} = \\
&\quad \frac{1}{2}(\boldsymbol{A}^{\mathrm{T}} \otimes \boldsymbol{I}_n + \boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{B}^{\mathrm{T}} + \boldsymbol{I}_m \otimes \boldsymbol{B}) = \frac{1}{2}(\tilde{\boldsymbol{A}}^{\mathrm{T}} + \tilde{\boldsymbol{A}}),
\end{aligned} \tag{12}$$

$$\begin{aligned}
&\tilde{\boldsymbol{N}} = \boldsymbol{N}_1 \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{N}_2 = \frac{\boldsymbol{A}^{\mathrm{T}} - \boldsymbol{A}}{2} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \frac{\boldsymbol{B}^{\mathrm{T}} - \boldsymbol{B}}{2} = \\
&\quad \frac{1}{2}(\boldsymbol{A}^{\mathrm{T}} \otimes \boldsymbol{I}_n - \boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{B}^{\mathrm{T}} - \boldsymbol{I}_m \otimes \boldsymbol{B}) = \frac{1}{2}(\tilde{\boldsymbol{A}}^{\mathrm{T}} - \tilde{\boldsymbol{A}}).
\end{aligned} \tag{13}$$

由定理 1 可知, 对于任意的 $\vec{\boldsymbol{X}} \in C^{mn}$ 且 $\vec{\boldsymbol{X}} \neq \boldsymbol{0}$, 满足 $\mathrm{Re}(\vec{\boldsymbol{X}}^{\mathrm{H}}\overline{\tilde{\boldsymbol{A}}}\vec{\boldsymbol{X}}\vec{\boldsymbol{X}}^{\mathrm{H}}\tilde{\boldsymbol{A}}\vec{\boldsymbol{X}}) > 0$, 则 $\rho(\tilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{N}}) < 1$.而

$$\begin{aligned}
&\mathrm{Re}(\vec{\boldsymbol{X}}^{\mathrm{H}}\overline{\tilde{\boldsymbol{A}}}\vec{\boldsymbol{X}}\vec{\boldsymbol{X}}^{\mathrm{H}}\tilde{\boldsymbol{A}}\vec{\boldsymbol{X}}) = \mathrm{Re}(\vec{\boldsymbol{X}}^{\mathrm{H}}(\bar{\boldsymbol{A}} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{B}^{\mathrm{H}})\vec{\boldsymbol{X}}\vec{\boldsymbol{X}}^{\mathrm{H}}(\boldsymbol{A} \otimes \boldsymbol{I}_n + \boldsymbol{I}_m \otimes \boldsymbol{B}^{\mathrm{T}})\vec{\boldsymbol{X}}) = \\
&\quad \mathrm{Re}(\mathrm{tr}(\boldsymbol{X}^{\mathrm{H}}(\bar{\boldsymbol{A}}\boldsymbol{X} + \boldsymbol{X}\bar{\boldsymbol{B}}))\mathrm{tr}(\boldsymbol{X}^{\mathrm{H}}(\boldsymbol{AX} + \boldsymbol{XB}))) =
\end{aligned}$$

$$\operatorname{Re}(\langle X,\bar{A}X+X\bar{B}\rangle\langle X,AX+XB\rangle), \tag{14}$$

故对任意的 $X \in C^{m\times n}$ 且 $X \neq 0$，当满足 $\operatorname{Re}(\langle X,\bar{A}X+X\bar{B}\rangle\langle X,AX+XB\rangle) > 0$ 时，

$$\rho((M_1 \otimes I_n + I_m \otimes M_2)^{-1}(N_1 \otimes I_n + I_m \otimes N_2)) < 1.$$

证毕. □

3 数值算例与结果分析

对于连续 Sylvester 矩阵方程(1)的求解，分别采用 INPHSS 算法[9]、CGS 算法[25]、TS_COCR 算法、TS_COCG 算法和 TS_CCC_SCBiCR 算法进行对比分析，以此验证本文所提出分裂迭代算法的可行性和有效性. 本文算例主要针对稀疏复线性方程组. 算例中，初始矩阵 $X_0 = 0 \in C^{m\times n}$，$R_k = F - AX_k - X_kB$，其中 X_k 表示第 k 步迭代矩阵，R_k 表示第 k 步迭代误差，终止条件满足 $\|R_k\|_F / \|R_0\|_F \leqslant 10^{-8}$，即算法中提及的 $\varepsilon = 10^{-8}$. 此外对于 INPHSS 算法中所涉及的矩阵 P_1 和 P_2 满足 $P_1 = I_m$，$P_2 = \beta I_n/\alpha$，其中 $\alpha = \beta$，I_m 和 I_n 分别为 m 阶和 n 阶单位矩阵. 内迭代过程采用求解 Hermite 正定方程组的 CG 算法. CGS 算法中选取 $\tilde{R}_0 = R_0$. 算例中 k，t 分别表示计算近似解的迭代次数和迭代时间，且算例中的迭代次数 k 表示的是 INPHSS 算法、TS_COCR 算法、TS_COCG 算法及 TS_CCC_SCBiCR 算法的外迭代次数.

本文所有算例均使用双精度浮点运算并在 MATLAB R2017b 平台上运行，电脑配置为 PC-Intel(R) Core(TM) i7-6700 CPU 3.40 GHz，16 GB 内存.

例 1 考虑连续 Sylvester 矩阵方程(1)，且 $m = n$，方程(1)中系数矩阵为

$$A = M_A + 2qN_A + 100I/(n+1)^2,\ B = M_B + 2qN_B + 100I/(n+1)^2,$$

其中三对角矩阵 $M_A, N_A, M_B, N_B \in R^{n\times n}$，满足

$$M_A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix},\ M_B = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix},$$

$$N_A = \begin{pmatrix} 0 & -1.5 & & & \\ 1.5 & 0 & -1.5 & & \\ & \ddots & \ddots & \ddots & \\ & & 1.5 & 0 & -1.5 \\ & & & 1.5 & 0 \end{pmatrix},\ N_B = \begin{pmatrix} 0 & -3 & & & \\ 3 & 0 & -3 & & \\ & \ddots & \ddots & \ddots & \\ & & 3 & 0 & -3 \\ & & & 3 & 0 \end{pmatrix}.$$

右端项 F 选取随机稀疏矩阵. 该算例的系数矩阵来源于文献[9]. 针对该算例，INPHSS 算法的理论最优迭代参数 α 如表 1 所示[9]，INPHSS 算法、CGS 算法、TS_COCR 算法、TS_COCG 算法及 TS_CCC_SCBiCR 算法的计算结果对比详见表 2.

表 1 INPHSS 的理论最优迭代参数

Table 1 The theoretical quasi-optimal iteration parameters for INPHSS

n	α		
	$q = 0.05$	$q = 0.1$	$q = 0.2$
80	0.033 0	0.131 9	0.527 6
160	0.033 6	0.134 2	0.536 8

表 2 INPHSS、CGS、TS_COCR、TS_COCG 及 TS_CCC_SCBiCR 求解例 1 的数值结果

Table 2 Numerical results of example 1 obtained with the INPHSS, CGS, TS_COCR, TS_COCG and TS_CCC_SCBiCR methods

n	q	INPHSS		CGS		TS_COCR		TS_COCG		TS_CCC_SCBiCR	
		k	t/s	k	t/s	k	t/s	k	t/s	k	t/s
80	0.05	11	0.003 3	13	0.010 3	11	0.002 9	11	0.002 4	11	0.003 3
	0.1	17	0.015 0	17	0.027 7	18	0.002 6	18	0.002 3	18	0.019 6
	0.2	39	0.028 3	22	0.056 6	75	0.002 3	75	0.001 9	75	0.002 9
160	0.05	11	0.011 4	14	0.059 6	11	0.011 2	11	0.010 0	11	0.015 0
	0.1	18	0.013 3	18	0.071 5	19	0.010 5	19	0.010 4	19	0.013 9
	0.2	42	0.038 9	22	0.106 9	79	0.009 0	79	0.007 2	79	0.011 3

例 2 考虑连续 Sylvester 矩阵方程(1),且 $m = n$, 方程(1)中系数矩阵为

$$\boldsymbol{A} = \begin{pmatrix} 4 & \mathrm{i} & & & \\ -\mathrm{i} & 4 & \mathrm{i} & & \\ & \ddots & \ddots & \ddots & \\ & & -\mathrm{i} & 4 & \mathrm{i} \\ & & & -\mathrm{i} & 4 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} \gamma_s & -1 & & & \\ 1 & \beta & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & \beta & -1 \\ & & & 1 & \gamma_s \end{pmatrix},$$

其中$\beta = 2 - k^2h^2$,$\gamma_s = 2 - k^2h^2 - (1 + \mathrm{i}kh)/(1 + k^2h^2)$,$h = 1/(n + 1)$,$k = 1$.右端项 $\boldsymbol{F} = \boldsymbol{A}\boldsymbol{X}^* + \boldsymbol{X}^*\boldsymbol{B}$, 其中 $\boldsymbol{X}^* = \boldsymbol{I} + \mathrm{i}\boldsymbol{E}$,$\boldsymbol{I}$ 为单位矩阵, $\boldsymbol{E} = (e_{ij})_{m\times n}$,且 $e_{ij} = 1$.CGS 算法、TS_COCR 算法、TS_COCG 算法及 TS_CCC_SCBiCR 算法的计算结果对比详见表 3.

表 3 CGS、TS_COCR、TS_COCG 及 TS_CCC_SCBiCR 求解例 2 的数值结果

Table 3 Numerical results of example 2 obtained with the CGS, TS_COCR, TS_COCG and TS_CCC_SCBiCR methods

n	CGS		TS_COCR		TS_COCG		TS_CCC_SCBiCR	
	k	t/s	k	t/s	k	t/s	k	t/s
2^8	8	0.344 7	20	0.057 2	20	0.049 3	20	0.084 3
2^{10}	8	2.459 2	19	1.037 8	19	0.681 8	19	1.651 6
2^{12}	8	129.220 2	18	64.383 8	18	46.332 3	18	70.753 6

例 3 考虑连续 Sylvester 矩阵方程(1),且 $m = n$, 方程(1)中系数矩阵为

$$\boldsymbol{A} = \mathrm{triu}(\mathrm{rand}(n),1) \times \mathrm{i} + \mathrm{diag}(q\boldsymbol{I}_n + \mathrm{diag}(\mathrm{rand}(n))),$$
$$\boldsymbol{B} = \mathrm{tril}(\mathrm{rand}(n),1) + \mathrm{diag}(q\boldsymbol{I}_n + \mathrm{diag}(\mathrm{rand}(n))) \times 5\mathrm{i},$$

其中 $q = \sqrt{n}$,rand(n) 表示 n 阶随机矩阵,triu(rand(n),1) 表示提取 n 阶随机矩阵 rand(n) 的上三角矩阵, 对角线及对角线以下元素均为 0,tril(rand(n),1) 表示提取 n 阶随机矩阵 rand(n) 的下三角矩阵,对角线及对角线以上元素均为0.右端项矩阵 $\boldsymbol{F} = \boldsymbol{A}\boldsymbol{X}^* + \boldsymbol{X}^*\boldsymbol{B}$,其中 $\boldsymbol{X}^* = \boldsymbol{I} + \mathrm{i}\boldsymbol{E}$,$\boldsymbol{I}$ 为单位矩阵,$\boldsymbol{E} = (e_{ij})_{m\times n}$,且 $e_{ij} = 1$.CGS 算法、TS_COCR 算法、TS_COCG 算法及 TS_CCC_SCBiCR 算法的计算结果对比详见表 4.

基于表 1~4 的数值结果,采用 INPHSS、CGS、TS_COCR、TS_COCG 及 TS_CCC_SCBiCR 求解连续的 Sylvester 矩阵方程(1)可得到如下相关结论:

1) 表 1 表明随着参数 q 的增加,系数矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 次对角线元素增大,进而 INPHSS 算法所涉及的理论拟最优参数值增加,且在同一矩阵维数下,INPHSS 算法的迭代次数和迭代时间也随着参数 q 的增加而增加;相较于 CGS 算法和 TS_CCC_SCBiCR 算法,INPHSS 算法具有良好

的收敛性,然而 TS_COCG 算法和 TS_COCR 算法的收敛速度明显优于 INPHSS 算法,并且避免了最优迭代参数的选取.

2)表 2~4 的数值结果表明,相较于直接求解矩阵方程(1)的 CGS 算法,分裂迭代法(TS_COCR、TS_COCG、TS_CCC_SCBiCR)明显提高了方程(1)的求解速度;此外通过观察文中所提出的分裂迭代法,可以发现随着阶数 n 的增加,TS_COCG 算法具有良好的收敛性,特别是例 2,当矩阵阶数达到 2^{12} 时,TS_COCG 算法相较于 TS_COCR 和 TS_CCC_SCBiCR 算法,迭代速度加快了至少 0.25 倍,这也进一步说明了 TS_COCG 算法的有效性和优越性.

表 4 CGS、TS_COCR、TS_COCG 及 TS_CCC_SCBiCR 求解例 3 的数值结果

Table 4 Numerical results of example 3 obtained with the CGS, TS_COCR, TS_COCG and TS_CCC_SCBiCR methods

n	CGS		TS_COCR		TS_COCG		TS_CCC_SCBiCR	
	k	t/s	k	t/s	k	t/s	k	t/s
100	7	0.014 5	18	0.007 6	18	0.006 6	18	0.012 1
200	8	0.056 3	24	0.046 9	24	0.039 9	24	0.063 7
400	9	0.337 0	39	0.335 5	39	0.308 7	39	0.438 2

基于上述结论,可知分裂迭代法中内迭代格式所采用的算法直接影响到整体算法的收敛速度,但对于外迭代次数却毫无影响,这进一步说明内迭代过程针对上述 3 个算例均进行了精确求解;同时算例结果也说明了文中所提出的分裂迭代法的可行性和有效性.

4 结 束 语

本文主要通过将系数矩阵分裂为对称矩阵和反对称矩阵,而后利用复对称矩阵求解高效的特点,提出了针对连续 Sylvester 矩阵方程(1)求解的分裂迭代算法,有效提高了算法的计算效率,大大降低了计算时间.以下是针对分裂迭代算法的总结:

1)相较于传统的 CGS 算法,分裂迭代算法(TS_COCR、TS_COCG、TS_CCC_SCBiCR)因其利用了系数矩阵分裂后复对称的特点,具有良好的收敛性;此外,所提出的算法与传统的分裂思想不同(分裂为 Hermite 矩阵和反 Hermite 矩阵),巧妙避免了最优迭代参数的选取,提高了算法的易实现性、易操作性.

2)分裂迭代算法由内外迭代格式组成,并且整体算法的收敛性很大程度依赖于内迭代格式的选取,而 COCG 法和 COCR 法相较于 CCC_SCBiCR 法具有较好的收敛性和鲁棒性,因此,从理论层面来看 TS_COCR 和 TS_COCG 算法的收敛速度优于 TS_CCC_SCBiCR 算法的收敛速度,同时数值算例也验证了这一点.

参考文献(References):

[1] LANCASTER P, RODMAN L. *Algebraic Riccati Equations*[M]. Oxford: The Clarendon Press, 1995.

[2] CHIANG C Y. On the Sylvester-like matrix equation $\boldsymbol{AX} + f(\boldsymbol{X})\boldsymbol{B} = \boldsymbol{C}$[J]. *Journal of the Franklin Institute*, 2016, **353**: 1061-1074.

[3] CALVETTI D, REICHEL L. Application of ADI iterative methods to the restoration of noisy images[J]. *SIAM Journal on Matrix Analysis and Applications*, 1996, **17**(1): 165-186.

[4] ANDERSON B, AGATHOKLIS P, JURY E, et al. Stability and the matrix Lyapunov equation for discrete 2-dimensional systems[J]. *IEEE Transactions on Circuits and Systems*, 1986,

33(3): 261-267.

[5] BARTELS R H, STEWART G W. Solution of the matrix equation $\boldsymbol{AX}+\boldsymbol{XB}=\boldsymbol{C}$ [F4][J]. *Communications of the ACM*, 1972, **15**(9): 820-826.

[6] GOLUB G, NASH S, VAN LOAN C. A Hessenberg-Schur method for the problem $\boldsymbol{AX}+\boldsymbol{XB}=\boldsymbol{C}$ [J]. *IEEE Transactions on Automatic Control*, 1979, **24**(6): 909-913.

[7] BAI Z Z. On Hermitian and skew-Hermitian splitting iteration methods for continuous Sylvester equations[J]. *Journal of Computational Mathematics*, 2011, **29**(2): 185-198.

[8] ZHOU D M, CHEN G L, CAI Q Y. On modified HSS iteration methods for continuous Sylvester equations[J]. *Applied Mathematics and Computation*, 2015, **263**: 84-93.

[9] XU L, HUO H F, YANG A L. Preconditioned HSS iteration method and its non-alternating variant for continuous Sylvester equations[J]. *Computers and Mathematics With Applications*, 2018, **75**(4): 1095-1106.

[10] DEHGHAN M, SHIRILORD A. A generalized modified Hermitian and skew-Hermitian splitting (GMHSS) method for solving complex Sylvester matrix equation[J]. *Applied Mathematics and Computation*, 2019, **348**: 632-651.

[11] ZHOU R, WANG X, TANG X B. A generalization of the Hermitian and skew-Hermitian splitting iteration method for solving Sylvester equations[J]. *Applied Mathematics and Computation*, 2015, **271**: 609-617.

[12] ZHOU R, WANG X, TANG X B. Preconditioned positive-definite and skew-Hermitian splitting iteration methods for continuous Sylvester equations $\boldsymbol{AX}+\boldsymbol{XB}=\boldsymbol{C}$ [J]. *East Asian Journal on Applied Mathematics*, 2017, **7**(1): 55-69.

[13] ZHENG Q Q, MA C F. On normal and skew-Hermitian splitting iteration methods for large sparse continuous Sylvester equations[J]. *Journal of Computational and Applied Mathematics*, 2014, **268**: 145-154.

[14] XIAO X Y, WANG X, YIN H W. Efficient single-step preconditioned HSS iteration methods for complex symmetric linear systems[J]. *Computers and Mathematics With Applications*, 2017, **74**(10): 2269-2280.

[15] XIAO X Y, WANG X, YIN H W. Efficient preconditioned NHSS iteration methods for solving complex symmetric linear systems[J]. *Computers and Mathematics With Applications*, 2018, **75**(1): 235-247.

[16] 杨丽, 李军. Hilbert 空间中分裂可行性问题的改进 Halpern 迭代和黏性逼近算法[J]. 应用数学和力学, 2017, **38**(9): 1072-1080.(YANG Li, LI Jun. Modified Halpern iteration and viscosity approximation methods for split feasibility problems in Hilbert spaces[J]. *Applied Mathematics and Mechanics*, 2017, **38**(9): 1072-1080.(in Chinese))

[17] LI C L, MA C F. On semi-convergence of parameterized SHSS method for a class of singular complex symmetric linear systems[J]. *Computers and Mathematics With Applications*, 2019, **77**: 466-475.

[18] HUANG Z G, WANG L G, XU Z, et al. The generalized double steps scale-SOR iteration method for solving complex symmetric linear systems[J]. *Journal of Computational and Applied Mathematics*, 2019, **346**: 284-306.

[19] HUANG Z G, WANG L G, XU Z, et al. Preconditioned accelerated generalized successive overrelaxation method for solving complex symmetric linear systems[J]. *Computers and Mathematics With Applications*, 2019, **77**(7): 1902-1916.

[20] VAN DER VORST H A, MELISSEN J B M. A Petrov-Galerkin type method for solving $\boldsymbol{Ax}=\boldsymbol{b}$,

where *A* is symmetric complex[J]. *IEEE Transactions on Magnetics*, 1990, **26**(2): 706-708.

[21] GU X M, CLEMENS M, HUANG T Z, et al. The SCBiCG class of algorithms for complex symmetric linear systems with applications in several electromagnetic model problems[J]. *Computer Physics Communications*, 2015, **191**: 52-64.

[22] SOGABE T, ZHANG S L. A COCR method for solving complex symmetric linear systems[J]. *Journal of Computational and Applied Mathematics*, 2007, **199**(2): 297-303.

[23] CLEMENS M, WEILAND T, VAN RIENEN U. Comparison of Krylov-type methods for complex linear systems applied to high-voltage problems[J]. *IEEE Transactions on Magnetics*, 1998, **34**(5): 3335-3338.

[24] ABE K, FUJINO S. Converting BiCR method for linear equations with complex symmetric matrices[J]. *Applied Mathematics and Computation*, 2018, **321**: 564-576.

[25] HAJARIAN M. Matrix form of the CGS method for solving general coupled matrix equations [J]. *Applied Mathematics Letters*, 2014, **34**: 37-42.

A Splitting Iterative Algorithm for Solving Continuous Sylvester Matrix Equations

LI Ying

(*School of Information Technology*, *Shangqiu Normal University*, *Shangqiu*, *Henan* 476000, *P.R.China*)

Abstract: The solution of continuous Sylvester matrix equations has significant application value in scientific and engineering calculations, hence, a splitting iterative algorithm was proposed. The core idea of the algorithm is to split the coefficient matrix of the continuous Sylvester matrix equation into a symmetric matrix and an antisymmetric matrix with an outer iterative scheme, and to solve the complex symmetric matrix equation with the inner iterative scheme. Compared with the traditional splitting algorithms, the proposed splitting algorithm effectively avoids the selection of optimal iterative parameters and takes advantages of the efficient solution of complex symmetric equations, which improves the easy implementation and easy operation of the algorithm. In addition, the convergence of the splitting iterative algorithm was further proved theoretically. Numerical examples show that, the splitting iterative algorithm has good convergence and robustness, and the convergence of the splitting iterative algorithm depends on the selection of the inner iterative schemes.

Key words: Sylvester matrix equation; complex symmetric matrix equation; splitting iterative algorithm; convergence