文章编号:1000-0887(2019)03-0237-13

非 Hermite 线性方程组的若干预处理迭代算法*

张迎春[1], 李 英[2], 肖曼玉[3], 谢公南[1]

(1. 西北工业大学 航海学院 机械与动力工程系, 西安 710072;
2. 商丘师范学院 信息技术学院, 河南 商丘 476000;
3. 西北工业大学 应用数学系, 西安 710129)

(我刊编委谢公南来稿)

摘要: 非 Hermite 线性方程组在科学和工程计算中有着重要的理论研究意义和使用价值,因此如何高效求解该类线性方程组,一直是研究者所探索的方向.通过提出一种预处理方法,对非 Hermite 线性方程组和具有多个右端项的复线性方程组求解的若干迭代算法进行预处理,旨在提高原算法的收敛速度.最后通过数值试验表明,所提出的若干预处理迭代算法与原算法相比较,预处理算法迭代次数大大降低,且收敛速度明显优于原算法.除此之外,广义共轭 *A*- 正交残量平方法(GCORS2)的预处理算法与其他算法相比,具有良好的收敛性行为和较好的稳定性.

关 键 词: 非 Hermite 线性方程组; 广义共轭 *A*- 正交残量平方法; 预处理方法

中图分类号: O246 **文献标志码:** A DOI: 10.21656/1000-0887.390222

引 言

在声散射[1]、流体力学[2]、量子化学[3]、Helmholtz 方程[4]、Maxwell 方程[5-6]等诸多科学计算和工程应用领域都会遇到各类微分方程或积分方程,通过采用有限元、有限差分等离散化技术,这些问题最终都归结于以下两类复线性方程组的求解.

第一类复线性方程组为

$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}, \tag{1}$$

其中 $\boldsymbol{A}\in C^{n\times n}$ 为非 Hermite 矩阵, $\boldsymbol{x},\boldsymbol{b}\in C^{n}$.

第二类具有多个右端项的复线性方程组为

$$\boldsymbol{A}\boldsymbol{X}=\boldsymbol{B}, \tag{2}$$

其中 $\boldsymbol{A}\in C^{n\times n}$ 为非 Hermite 矩阵, $\boldsymbol{X},\boldsymbol{B}\in C^{n\times p}$, $p\ll n$.

由于在实际工程问题的数值计算中,大规模的线性方程组求解所占比重较大,因此高效求解上述两类线性方程组一直是各领域追求的目标,并且具有重要的理论意义和使用价值,故而吸引了众多学者的关注.目前,对于非 Hermite 线性方程组和具有多个右端项的非 Hermite 线

* **收稿日期:** 2018-08-23; **修订日期:** 2018-09-02
基金项目: 国家自然科学基金(51676163)
作者简介: 张迎春(1992—),女,博士生(E-mail: zhangyingchun@ mail.nwpu.edu.cn);
谢公南(1980—),男,教授,博士,博士生导师(通讯作者. E-mail: xgn@ nwpu.edu.cn).

性方程组的求解主要集中于 Krylov 子空间方法的研究.

针对大规模的复非对称线性方程组法的求解,若不考虑存储量限制,则被广泛应用的一类 Krylov 子空间方法有广义最小残量(GMRES)法[7]以及相关 GMRES 的改进方法[8],该类方法具有非常好的鲁棒性.若考虑存储量的限制,目前主要常用的一类 Krylov 子空间方法有稳定化双共轭梯度(BiCGSTAB)法[9]、广义积型双共轭梯度(GPBiCG)法[10]、双共轭 A- 正交残量稳定化(BiCORSTAB)法[11]、广义积型双共轭 A- 正交残量(GPBiCOR)法[12]、GCORS2 法[13]等.针对多右端项大规模线性方程组的求解,目前主要的迭代法有总体 CGS(Gl-CGS)方法[14]、总体广义积型 BiCG(Gl-GPBiCG)方法[15]、总体 GPBiCG_plus(Gl-GPBiCG_plus)方法[15-16]等.

然而,由于实际工程计算中计算规模、存储量、计算效率等因素的限制,直接采用上述方法往往不能满足高效这一要求.故而预处理技术就成为了当前广泛应用的方法[13,15-17],并通过该预处理方法达到算法收敛速度快、稳定性和鲁棒性好的目的.但针对不同类型的方程组,设计相应的高效预处理方法依旧是目前众多研究者研究的方向.因此,本文主要针对求解非 Hermite 线性方程组(1)和具有多个右端项的复线性方程组(2)的若干迭代算法(GCORS2[13]、BiCORSTAB[11]、Gl-GPBiCG[15]和 Gl-GPBiCG_plus[15]),提出了一种预处理方法,并通过对这些迭代算法进行预处理,进而加速其算法的收敛速度,改善了算法稳定性和鲁棒性.

文中内积定义为:若 $\boldsymbol{x},\boldsymbol{y}\in C^n$,则其内积为 $\langle \boldsymbol{x},\boldsymbol{y}\rangle = \boldsymbol{y}^{\mathrm{H}}\boldsymbol{x}$,由此导出向量的 2-范数为 $\|\boldsymbol{x}\|_2 = \sqrt{\langle \boldsymbol{x},\boldsymbol{x}\rangle}$.若 $\boldsymbol{X},\boldsymbol{Y}\in C^{n\times p}$,则其内积为 $\langle \boldsymbol{X},\boldsymbol{Y}\rangle = \mathrm{tr}(\boldsymbol{Y}^{\mathrm{H}}\boldsymbol{X})$,由此导出矩阵的 Frobenius 范数为 $\|\boldsymbol{X}\|_{\mathrm{F}} = \sqrt{\langle \boldsymbol{X},\boldsymbol{X}\rangle}$.

1 预处理方法

为了加快非 Hermite 线性方程组(1)和具有多个右端项的复线性方程组(2)的求解速度,目前最常用的方法是采用预处理技术对相应算法进行预处理.因此,本文通过对若干迭代算法采用预处理技术求解线性方程组.

1.1 预处理思想

该预处理方法的思想为:采用与实线性方程组加速求解的类似方法,通过构造预处理条件子 $\boldsymbol{M}^{-1}$,使其近似于系数矩阵 $\boldsymbol{A}$ 的逆矩阵,即 $\boldsymbol{M}^{-1}\approx\boldsymbol{A}^{-1}$,其中 $\boldsymbol{M}^{-1} = (\boldsymbol{M}_1\boldsymbol{M}_2)^{-1}$.而后对非 Hermite 线性方程组(1)预处理,得到相应的预处理方程

$$\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{b}},$$

其中 $\tilde{\boldsymbol{A}} = \boldsymbol{M}_1^{-1}\boldsymbol{A}\boldsymbol{M}_2^{-1}, \tilde{\boldsymbol{x}} = \boldsymbol{M}_2\boldsymbol{x}, \tilde{\boldsymbol{b}} = \boldsymbol{M}_1^{-1}\boldsymbol{b}$.对于具有多个右端项的复线性方程组(2)的求解采用同样的预处理技术.本文仅考虑右预处理,故令 $\boldsymbol{M}_1 = \boldsymbol{I}, \boldsymbol{M}_2 = \boldsymbol{M}$,拟通过这种预处理技术,使得系数矩阵的奇异值分布更加集中,从而加快了 GCORS2[13]、BiCORSTAB[11]、Gl-GPBiCG[15]和 Gl-GPBiCG_plus[15]的收敛速度.

1.2 预处理条件子的选取

令 $\boldsymbol{A} = (a_{ij})_{n\times n} = \boldsymbol{D} - \boldsymbol{N}$,其中

$$\boldsymbol{D} = \mathrm{diag}(d_1, d_2, \cdots, d_n), \qquad d_i = \begin{cases} a_{ii}, & a_{ii}\neq 0, \\ 1, & a_{ii} = 0. \end{cases}$$

故

$$\boldsymbol{A}^{-1} = (\boldsymbol{D} - \boldsymbol{N})^{-1} = (\boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{N})\boldsymbol{D}^{-1},$$

其中 $\boldsymbol{I}$ 为单位矩阵.又由于

$$(\boldsymbol{I}-\boldsymbol{D}^{-1}\boldsymbol{N})^{-1}\approx\boldsymbol{I}+\boldsymbol{D}^{-1}\boldsymbol{N}+\cdots+(\boldsymbol{D}^{-1}\boldsymbol{N})^{q-1},$$

因此选取预处理条件子 $\boldsymbol{M}^{-1}$ 为

$$\boldsymbol{M}^{-1}=(\boldsymbol{I}+\boldsymbol{D}^{-1}\boldsymbol{N}+\cdots+(\boldsymbol{D}^{-1}\boldsymbol{N})^{q-1})\boldsymbol{D}^{-1},$$

其中 $q\geqslant 1$.

注 1 基于预处理条件子 $\boldsymbol{M}^{-1}$ 的选取,可发现,随着内迭代次数 q 的增加,计算量也相应增加.因此,如何适当选取内迭代次数对于提升预处理算法的收敛速度至关重要.此外,本文仅考虑右预处理,若考虑左预处理,则令 $\boldsymbol{M}_1=\boldsymbol{M},\boldsymbol{M}_2=\boldsymbol{I}$, 预处理条件子选取一样,唯一区别在于预处理算法中迭代步骤的处理,但左预处理与右预处理技术目的均是提高算法的收敛速度和改善算法的稳定性.因此,左预处理有着同样的效果,在此不再赘述.

2 若干预处理迭代算法

首先考虑非 Hermite 线性方程组(1),拟采用预处理技术对 GCORS2[13] 和 BiCORSTAB[11] 进行预处理.

2.1 预处理 GCORS2 算法

下面给出 GCORS2 算法[13]的预处理算法(记作 PGCORS2 算法):

第 1 步 给定初始向量 $\boldsymbol{x}_0$, 计算 $\boldsymbol{r}_0=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}_0$;选择 $\boldsymbol{r}_0^*=p(\boldsymbol{A}\boldsymbol{M}^{-1})\boldsymbol{r}_0$, 使得 $\langle\boldsymbol{r}_0^*,\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{r}_0\rangle\neq 0$, 其中 $p(t)$ 是以 t 为变量的任意阶多项式.

第 2 步 置 $\boldsymbol{u}_0=\boldsymbol{t}_0=\boldsymbol{M}^{-1}\boldsymbol{r}_0,\boldsymbol{q}_0=\hat{\boldsymbol{u}}_0=\hat{\boldsymbol{t}}_0=\hat{\boldsymbol{r}}_0=\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{r}_0,\hat{\boldsymbol{q}}_0=\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{q}_0,\rho_0=\langle\boldsymbol{r}_0^*,\hat{\boldsymbol{r}}_0\rangle,\hat{\rho}_0=\langle\boldsymbol{s}_0^*,\hat{\boldsymbol{r}}_0\rangle,k:=0$.

第 3 步 若 $\|\boldsymbol{r}_k\|_2\leqslant\varepsilon\|\boldsymbol{r}_0\|_2$, 计算停止,其中 $\varepsilon\in\mathbf{R}_+$; 否则,计算

$$\sigma_k=\langle\boldsymbol{r}_0^*,\hat{\boldsymbol{q}}_k\rangle,\ \hat{\sigma}_k=\langle\boldsymbol{s}_0^*,\hat{\boldsymbol{q}}_k\rangle,\ \alpha_k=\rho_k/\sigma_k,\ \hat{\alpha}_k=\hat{\rho}_k/\hat{\sigma}_k,\ \boldsymbol{s}_k=\boldsymbol{t}_k-\alpha_k\boldsymbol{M}^{-1}\boldsymbol{q}_k,$$
$$\hat{\boldsymbol{s}}_k=\hat{\boldsymbol{t}}_k-\alpha_k\hat{\boldsymbol{q}}_k,\ \boldsymbol{h}_k=\boldsymbol{u}_k-\hat{\alpha}_k\boldsymbol{M}^{-1}\boldsymbol{q}_k,\ \hat{\boldsymbol{h}}_k=\hat{\boldsymbol{u}}_k-\hat{\alpha}_k\hat{\boldsymbol{q}}_k,\ \boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\alpha_k\boldsymbol{u}_k+\hat{\alpha}_k\boldsymbol{s}_k,$$
$$\boldsymbol{r}_{k+1}=\boldsymbol{r}_k-\alpha_k\hat{\boldsymbol{u}}_k-\hat{\alpha}_k\hat{\boldsymbol{s}}_k,\ \hat{\boldsymbol{r}}_{k+1}=\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{r}_{k+1},\ \rho_{k+1}=\langle\boldsymbol{r}_0^*,\hat{\boldsymbol{r}}_{k+1}\rangle,\ \hat{\rho}_{k+1}=\langle\boldsymbol{s}_0^*,\hat{\boldsymbol{r}}_{k+1}\rangle,$$
$$\beta_{k+1}=\rho_{k+1}\alpha_k/(\rho_k\hat{\alpha}_k),\ \hat{\beta}_{k+1}=\hat{\rho}_{k+1}\hat{\alpha}_k/(\hat{\rho}_k\alpha_k),\ \boldsymbol{t}_{k+1}=\boldsymbol{M}^{-1}\boldsymbol{r}_{k+1}+\hat{\beta}_{k+1}\boldsymbol{s}_k,$$
$$\hat{\boldsymbol{t}}_{k+1}=\hat{\boldsymbol{r}}_{k+1}+\hat{\beta}_{k+1}\hat{\boldsymbol{s}}_k,\ \boldsymbol{u}_{k+1}=\boldsymbol{M}^{-1}\boldsymbol{r}_{k+1}+\beta_{k+1}\boldsymbol{h}_k,\ \hat{\boldsymbol{u}}_{k+1}=\hat{\boldsymbol{r}}_{k+1}+\beta_{k+1}\hat{\boldsymbol{h}}_k,$$
$$\boldsymbol{q}_{k+1}=\hat{\boldsymbol{t}}_{k+1}+\beta_{k+1}(\hat{\boldsymbol{h}}_k+\hat{\beta}_{k+1}\boldsymbol{q}_k),\ \hat{\boldsymbol{q}}_{k+1}=\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{q}_{k+1}.$$

第 4 步 置 $k:=k+1$, 转第 2 步.

从 PGCORS2 算法可以看出,该算法相较于 GCORS2 算法多求解了两个线性方程组 $\tilde{\boldsymbol{r}}_k=\boldsymbol{M}^{-1}\boldsymbol{r}_k$ 与 $\tilde{\boldsymbol{q}}_k=\boldsymbol{M}^{-1}\boldsymbol{q}_k$.下面首先给出求解线性方程 $\tilde{\boldsymbol{r}}_k=\boldsymbol{M}^{-1}\boldsymbol{r}_k$ 的迭代格式.

根据预处理条件子 $\boldsymbol{M}^{-1}$ 的形式,选取 $\tilde{\boldsymbol{r}}_k^0=\boldsymbol{0}$, 则相应的迭代格式如下:

for $l=0,1,\cdots,q-1$,

$$\tilde{\boldsymbol{r}}_k^{l+1}=\boldsymbol{D}^{-1}(\boldsymbol{N}\tilde{\boldsymbol{r}}_k^l+\boldsymbol{r}_k),$$

end

$$\tilde{\boldsymbol{r}}_k=\tilde{\boldsymbol{r}}_k^q.$$

类似地,选取 $\tilde{\boldsymbol{q}}_k^0=\boldsymbol{0}$, 可得到求解线性方程 $\tilde{\boldsymbol{q}}_k=\boldsymbol{M}^{-1}\boldsymbol{q}_k$ 的迭代格式如下:

for $l=0,1,\cdots,q-1$,

$$\tilde{\boldsymbol{q}}_k^{l+1}=\boldsymbol{D}^{-1}(\boldsymbol{N}\tilde{\boldsymbol{q}}_k^l+\boldsymbol{q}_k),$$

end

$$\tilde{\boldsymbol{q}}_k=\tilde{\boldsymbol{q}}_k^q.$$

对于 PGCORS2 算法,当 q 处于一定范围内时,系数矩阵特征值的密集程度会随着内迭代次数 q 的增大而更集中,进而加快了 GCORS2 算法的收敛速度.根据类似的预处理方式,可得到 BiCORSTAB 算法[11]的预处理算法.

2.2 预处理 BiCORSTAB 算法

下面给出 BiCORSTAB 算法[11]的预处理算法(记作 PBiCORSTAB 算法):

第 1 步 给定初始向量 $\boldsymbol{x}_0$, 计算 $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$;选择 $\boldsymbol{r}_0^*$, 使得 $\langle \boldsymbol{r}_0^*, \boldsymbol{A}\boldsymbol{r}_0 \rangle \neq 0$, 置 $k := 0$.

第 2 步 若 $\| \boldsymbol{r}_k \|_2 \leqslant \varepsilon \| \boldsymbol{r}_0 \|_2$, 计算停止,其中 $\varepsilon \in \mathbf{R}_+$; 否则,计算

$$\hat{\boldsymbol{r}}_k = \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{r}_k,\ \rho_k = \langle \boldsymbol{r}_0^*, \hat{\boldsymbol{r}}_k \rangle,$$

若 $\rho_k = 0$, 方法失效;否则,计算

$$\beta_k = \rho_k \alpha_k / (\rho_{k-1}\omega_k),\ \boldsymbol{p}_k = \boldsymbol{M}^{-1}\boldsymbol{r}_k + \beta_k(\boldsymbol{p}_{k-1} - \omega_{k-1}\boldsymbol{M}^{-1}\boldsymbol{q}_{k-1}),$$

$$\boldsymbol{q}_k = \hat{\boldsymbol{r}}_k + \beta_k(\boldsymbol{q}_{k-1} - \omega_{k-1}\hat{\boldsymbol{q}}_{k-1}),$$

(若 $k = 0$, 则 $\boldsymbol{p}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0, \boldsymbol{q}_0 = \hat{\boldsymbol{r}}_0$),

$$\hat{\boldsymbol{q}}_{k+1} = \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{q}_{k+1},\ \alpha_k = \rho_k / \langle \boldsymbol{r}_0^*, \hat{\boldsymbol{q}}_k \rangle,\ \boldsymbol{s}_k = \boldsymbol{r}_k - \alpha_k \boldsymbol{q}_k,\ \tilde{\boldsymbol{s}}_k = \boldsymbol{M}^{-1}\boldsymbol{r}_k - \alpha_k \boldsymbol{M}^{-1}\boldsymbol{q}_k,$$

若 $\| \boldsymbol{x}_k \|_2 \leqslant \varepsilon, \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$; 否则,计算

$$\boldsymbol{t}_k = \hat{\boldsymbol{r}}_k - \alpha_k \hat{\boldsymbol{q}}_k,\ \omega_k = \langle \boldsymbol{t}_k, \boldsymbol{s}_k \rangle / \langle \boldsymbol{t}_k, \boldsymbol{t}_k \rangle,\ \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k + \omega_k \tilde{\boldsymbol{s}}_k,\ \boldsymbol{r}_{k+1} = \boldsymbol{s}_k - \omega_k \boldsymbol{t}_k.$$

第 3 步 置 $k := k + 1$, 转第 2 步.

从 PBiCORSTAB 算法可以看出,该算法相较于 BiCORSTAB 算法多求解了两个线性方程组 $\tilde{\boldsymbol{r}}_k = \boldsymbol{M}^{-1}\boldsymbol{r}_k$ 与 $\tilde{\boldsymbol{q}}_k = \boldsymbol{M}^{-1}\boldsymbol{q}_k$.对于这两个线性方程组的预处理迭代格式与 2.1 小节的迭代格式类似,这里不再赘述.

其次,考虑具有多个右端项的复线性方程组(2),拟采用预处理技术对 Gl-GPBiCG 算法[15]和 Gl-GPBiCG_plus 算法[15]进行预处理.

2.3 预处理 Gl-GPBiCG 算法

下面给出 Gl-GPBiCG 算法[15]的预处理算法(记作 PGl-GPBiCG 算法):

第 1 步 给定初始元素 $\boldsymbol{X}_0$, 计算 $\boldsymbol{R}_0 = \boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_0$, 选择 $\hat{\boldsymbol{R}}_0$, 使得 $\langle \boldsymbol{R}_0, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}} \neq 0$.

第 2 步 置 $\boldsymbol{T}_{-1} = \boldsymbol{W}_{-1} = \boldsymbol{O}_{n\times p}$, $\beta_{-1} = 0$, $k := 0$.

第 3 步 若 $\| \boldsymbol{R}_k \|_{\mathrm{F}} \leqslant \varepsilon \| \boldsymbol{R}_0 \|_{\mathrm{F}}$, 计算停止,其中 $\varepsilon \in \mathbf{R}_+$; 否则,计算

$$\boldsymbol{P}_k = \boldsymbol{M}^{-1}\boldsymbol{R}_k + \beta_{k-1}(\boldsymbol{P}_{k-1} - \boldsymbol{U}_{k-1}),\ \alpha_k = \langle \boldsymbol{R}_k, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}} / \langle \boldsymbol{A}\boldsymbol{P}_k, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}},$$

$$\boldsymbol{Y}_k = \boldsymbol{T}_{k-1} - \boldsymbol{R}_k - \alpha_k \boldsymbol{W}_{k-1} + \alpha_k \boldsymbol{A}\boldsymbol{P}_k,$$

$$\boldsymbol{T}_k = \boldsymbol{R}_k - \alpha_k \boldsymbol{A}\boldsymbol{P}_k,\ \boldsymbol{M}^{-1}\boldsymbol{T}_k = \boldsymbol{M}^{-1}\boldsymbol{R}_k - \alpha_k \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{P}_k,$$

$$\zeta_k = \frac{\langle \boldsymbol{Y}_k, \boldsymbol{Y}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{T}_k \rangle_{\mathrm{F}} - \langle \boldsymbol{Y}_k, \boldsymbol{T}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{Y}_k \rangle_{\mathrm{F}}}{\langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k \rangle_{\mathrm{F}} \langle \boldsymbol{Y}_k, \boldsymbol{Y}_k \rangle_{\mathrm{F}} - \langle \boldsymbol{Y}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{Y}_k \rangle_{\mathrm{F}}},$$

$$\eta_k = \frac{\langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k \rangle_{\mathrm{F}} \langle \boldsymbol{Y}_k, \boldsymbol{T}_k \rangle_{\mathrm{F}} - \langle \boldsymbol{Y}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{T}_k \rangle_{\mathrm{F}}}{\langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k \rangle_{\mathrm{F}} \langle \boldsymbol{Y}_k, \boldsymbol{Y}_k \rangle_{\mathrm{F}} - \langle \boldsymbol{Y}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{Y}_k \rangle_{\mathrm{F}}},$$

(若 $k = 0$, 则 $\zeta_k = \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{T}_k \rangle_{\mathrm{F}} / \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k \rangle_{\mathrm{F}}, \eta_k = 0$),

$$\boldsymbol{U}_k = \zeta_k \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{P}_k + \eta_k(\boldsymbol{M}^{-1}\boldsymbol{T}_{k-1} - \boldsymbol{M}^{-1}\boldsymbol{R}_k + \beta_{k-1}\boldsymbol{U}_{k-1}),$$

$$\boldsymbol{Z}_k = \zeta_k \boldsymbol{M}^{-1}\boldsymbol{R}_k + \eta_k \boldsymbol{Z}_{k-1} - \alpha_k \boldsymbol{U}_k,$$

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \alpha_k \boldsymbol{P}_k + \boldsymbol{Z}_k,\ \boldsymbol{R}_{k+1} = \boldsymbol{T}_k - \eta_k \boldsymbol{Y}_k - \zeta_k \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k,$$

$$\beta_k = \alpha_k \langle \boldsymbol{R}_{k+1}, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}} / \zeta_k \langle \boldsymbol{R}_k, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}},\ \boldsymbol{W}_k = \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{T}_k + \beta_k \boldsymbol{A}\boldsymbol{P}_k.$$

第 4 步 置 $k := k + 1$, 转第 2 步.

从 PGl-GPBiCG 算法可以看出,该算法相较于 Gl-GPBiCG 算法多求解了两个具有右端项的线性方程组 $\tilde{\boldsymbol{R}}_k = \boldsymbol{M}^{-1}\boldsymbol{R}_k$ 与 $\tilde{\boldsymbol{V}}_k = \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{P}_k$.对于这两个线性方程组的迭代格式与 2.1 小节的迭代格式类似.值得注意的是,对于 $\tilde{\boldsymbol{V}}_k = \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{P}_k$ 的求解,需先求解 $\boldsymbol{A}\boldsymbol{P}_k$,而后采用预处理迭代格式.

2.4 预处理 Gl-GPBiCG_plus 算法

下面给出 Gl-GPBiCG_plus 算法[15]的预处理算法(记作 PGl-GPBiCG_plus 算法):

第 1 步 给定初始元素 $\boldsymbol{X}_0$,计算 $\boldsymbol{R}_0 = \boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_0$,选择 $\hat{\boldsymbol{R}}_0$,使得 $\langle \boldsymbol{R}_0, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}} \neq 0$.

第 2 步 置 $\boldsymbol{P}_{-1} = \boldsymbol{U}_{-1} = \boldsymbol{Z}_{-1} = \boldsymbol{O}_{n\times p}, \beta_{-1} = 0, k := 0$.

第 3 步 若 $\| \boldsymbol{R}_k \|_{\mathrm{F}} \leqslant \varepsilon \| \boldsymbol{R}_0 \|_{\mathrm{F}}$,计算停止,其中 $\varepsilon \in \mathbf{R}_+$;否则,计算

$$\boldsymbol{P}_k = \boldsymbol{M}^{-1}\boldsymbol{R}_k + \beta_{k-1}(\boldsymbol{P}_{k-1} - \boldsymbol{M}^{-1}\boldsymbol{U}_{k-1}),\ \alpha_k = \langle \boldsymbol{R}_k, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}} / \langle \boldsymbol{A}\boldsymbol{P}_k, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}},$$

$$\zeta_k = \frac{\langle \boldsymbol{A}\boldsymbol{Z}_{k-1}, \boldsymbol{A}\boldsymbol{Z}_{k-1} \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{R}_k \rangle_{\mathrm{F}} - \langle \boldsymbol{A}\boldsymbol{Z}_{k-1}, \boldsymbol{R}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{A}\boldsymbol{Z}_{k-1} \rangle_{\mathrm{F}}}{\langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{Z}_{k-1}, \boldsymbol{A}\boldsymbol{Z}_{k-1} \rangle_{\mathrm{F}} - \langle \boldsymbol{A}\boldsymbol{Z}_{k-1}, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{A}\boldsymbol{Z}_{k-1} \rangle_{\mathrm{F}}},$$

$$\eta_k = \frac{\langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{Z}_{k-1}, \boldsymbol{R}_k \rangle_{\mathrm{F}} - \langle \boldsymbol{A}\boldsymbol{Z}_{k-1}, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{R}_k \rangle_{\mathrm{F}}}{\langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{Z}_{k-1}, \boldsymbol{A}\boldsymbol{Z}_{k-1} \rangle_{\mathrm{F}} - \langle \boldsymbol{A}\boldsymbol{Z}_{k-1}, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k \rangle_{\mathrm{F}} \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{A}\boldsymbol{Z}_{k-1} \rangle_{\mathrm{F}}},$$

(若 $k = 0$,则 $\zeta_k = \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{R}_k \rangle_{\mathrm{F}} / \langle \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{R}_k \rangle_{\mathrm{F}}, \eta_k = 0$),

$$\boldsymbol{U}_k = \zeta_k \boldsymbol{A}\boldsymbol{P}_k + \eta_k(\boldsymbol{A}\boldsymbol{Z}_{k-1} + \beta_{k-1}\boldsymbol{U}_{k-1}),\ \boldsymbol{T}_k = \boldsymbol{R}_k - \alpha_k \boldsymbol{A}\boldsymbol{P}_k,$$

$$\boldsymbol{Z}_k = \zeta_k \boldsymbol{M}^{-1}\boldsymbol{R}_k + \eta_k \boldsymbol{Z}_{k-1} - \alpha_k \boldsymbol{M}^{-1}\boldsymbol{U}_k,$$

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \alpha_k \boldsymbol{P}_k + \boldsymbol{Z}_k,\ \boldsymbol{R}_{k+1} = \boldsymbol{T}_k - \boldsymbol{A}\boldsymbol{Z}_k,\ \beta_k = \alpha_k \langle \boldsymbol{R}_{k+1}, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}} / \zeta_k \langle \boldsymbol{R}_k, \hat{\boldsymbol{R}}_0 \rangle_{\mathrm{F}}.$$

第 4 步 置 $k := k + 1$,转第 2 步.

从 PGl-GPBiCG_plus 算法可以看出,该算法相较于 Gl-GPBiCG_plus 算法多求解了两个具有右端项的线性方程组 $\tilde{\boldsymbol{R}}_k = \boldsymbol{M}^{-1}\boldsymbol{R}_k$ 与 $\tilde{\boldsymbol{U}}_k = \boldsymbol{M}^{-1}\boldsymbol{U}_k$.对于这两个线性方程组的迭代格式与 2.1 小节的迭代格式类似.

注 2 通过观察上述给出的 4 种预处理算法,可发现,随着内迭代次数 q 的增加,矩阵乘法运算和加法运算随之增多,这也就意味着内迭代计算量的增加.然而,由于内迭代次数在一定范围内的增加会减少整体算法的循环次数,进而导致外循环计算量的减少.因此从整体角度出发,为了确保算法精度的同时缩短计算时间,需寻求内迭代计算量增加与外循环计算量减少的平衡点.本文通过算例验证(未在本文列出),当内迭代次数 $q > 4$ 时,整体迭代次数减少,但计算时间增多,因此后续提供的算例仅考虑 $q = 1,2,4$ 的情况.

3 数值算例与结果分析

对于非 Hermite 线性方程组(1)和具有多个右端项的复线性方程组(2),分别采用 GCORS2 算法[13]、BiCORSTAB 算法[11]、Gl-GPBiCG 算法[15]、Gl-GPBiCG_plus 算法[15]、PGCORS2 算法、PBiCORSTAB 算法、PGl-GPBiCG 算法及 PGl-GPBiCG_plus 算法进行对比分析,以此验证预处理算法的有效性.此外通过观察预处理条件子,可知该类预处理仅对部分元素进行处理,适用于大型稀疏矩阵线性方程组的求解,如果对稠密矩阵进行预处理,将会大大增加计算量,导致算法效率的下降,故对于稠密矩阵线性方程组的求解并不适用.因此,本文算例主要针对稀疏矩阵线性方程.在算例中,初始向量 $\boldsymbol{x}_0 = \boldsymbol{0} \in C^n$($\boldsymbol{X}_0 = \boldsymbol{0} \in C^{n\times p}$),$\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$($\boldsymbol{R}_k = \boldsymbol{B} - \boldsymbol{A}\boldsymbol{X}_k$),其中 $\boldsymbol{x}_k$($\boldsymbol{X}_k$) 表示第 k 步迭代向量,$\boldsymbol{r}_k$($\boldsymbol{R}_k$) 表示第 k 步迭代误差,终止条件满足 $\| \boldsymbol{r}_k \|_2 / \| \boldsymbol{r}_0 \|_2 \leqslant 10^{-8}$(即 $\| \boldsymbol{R}_k \|_{\mathrm{F}} / \| \boldsymbol{R}_0 \|_{\mathrm{F}} \leqslant 10^{-8}$),即算法中提及的 $\varepsilon = 10^{-8}$,此外 GCORS2 算法[13]、BiCORSTAB 算法[11]中选取 $\boldsymbol{r}_0^* = \boldsymbol{A}\boldsymbol{r}_0$,Gl-GPBiCG 算法[15]、Gl-GPBiCG_plus

算法[15]中选取 $\hat{\boldsymbol{R}}_0 = \boldsymbol{R}_0$.算例中 k,t 及 l_{res} 分别表示计算近似解的迭代次数、迭代时间和相对残量范数,其中相对残量范数定义为 $l_{\mathrm{res}} = \lg(\|\boldsymbol{r}_k\|_2/\|\boldsymbol{r}_0\|_2)$(即 $l_{\mathrm{res}} = \lg(\|\boldsymbol{R}_k\|_{\mathrm{F}}/\|\boldsymbol{R}_0\|_{\mathrm{F}})$).

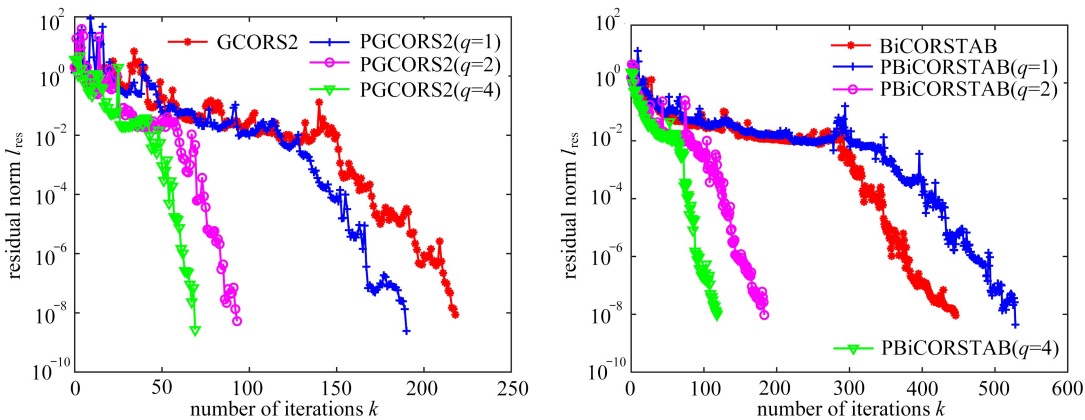
本文所有算例均使用双精度浮点运算并在 MATLAB R2017b 平台上运行,电脑配置:CPU 为 PC-Interl(R) Core(TM) i7-6700,3.40 GHz,内存为 16 GB.

例 1 为了验证 PGCORS2 算法和 PBiCORSTAB 算法求解非 Hermite 线性方程组(1)的有效性和鲁棒性,该算例主要针对源于声散射、二维对流扩散和有限差分 Laplace(拉普拉斯)算子问题通过转换得到的 YOUNG1C、CDDE1 和 GR_30_30 矩阵进行测试,更多细节请详见文献[18],方程(1)中右端项选取 $\boldsymbol{b} = (i,\cdots,i)^{\mathrm{T}}$.3 个测试矩阵的计算结果对比详见表 1,相应算法的收敛性行为详见图 1~3.

表 1 预处理方法求解 YOUNG1C、CDDE1 及 GR_30_30 的数值结果

Table 1 Numerical results of YOUNG1C, CDDE1 and GR_30_30 obtained with preconditioning methods

method	YOUNG1C			CDDE1			GR_30_30		
	k	t/s	l_{res}	k	t/s	l_{res}	k	t/s	l_{res}
GCORS2[13]	219	0.023	−8.061	103	0.019	−8.859	50	0.013	−8.231
PGCORS2 (q = 1)	191	0.025	−8.611	103	0.023	−8.872	50	0.016	−8.231
PGCORS2 (q = 2)	94	0.017	−8.281	51	0.014	−8.047	28	0.011	−8.074
PGCORS2 (q = 4)	70	0.015	−8.572	36	0.009	−8.207	20	0.009	−8.355
BiCORSTAB[11]	447	0.034	−8.040	198	0.020	−8.321	50	0.010	−8.071
PBiCORSTAB (q = 1)	529	0.048	−8.357	183	0.026	−8.310	50	0.013	−8.071
PBiCORSTAB (q = 2)	184	0.020	−8.024	77	0.013	−8.202	31	0.009	−8.139
PBiCORSTAB (q = 4)	119	0.018	−8.008	52	0.009	−8.312	18	0.006	−8.137



(a) PGCORS2 (b) PBiCORSTAB

图 1 例 1 中 YOUNG1C 的收敛结果

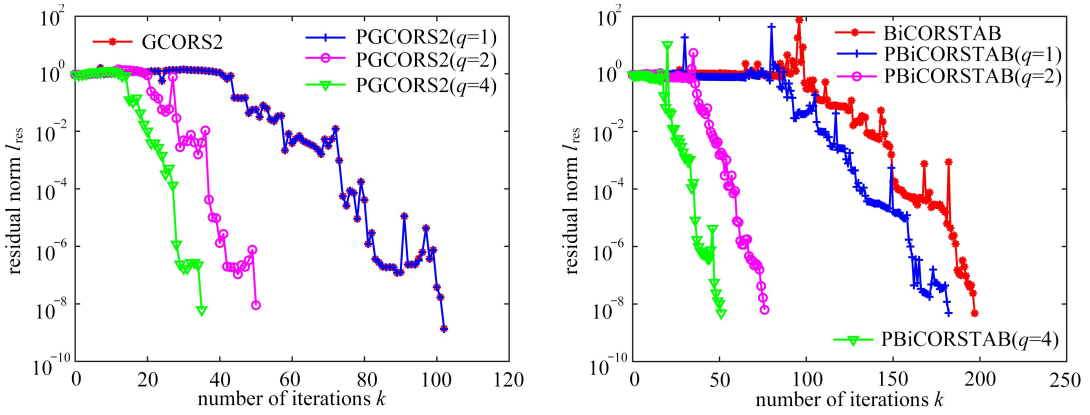
Fig. 1 Convergence histories of YOUNG1C for example 1

基于表 1 和图 1~3 的数值结果,采用 GCORS2、BiCORSTAB 及相应的预处理算法求解线性方程组(1)可得到如下相关结论:

1) 对于测试矩阵 YOUNG1C,GCORS2 算法在迭代时间和迭代次数两方面所表现出的收敛性行为均优于 BiCORSTAB 算法,该结论与文献[13]保持一致.类似地,测试矩阵 CDDE1 和 GR_30_30 同样呈现出相关收敛性行为.另外从图表中的数据可知,PGCORS2 算法的收敛性行为同样优于 PBiCORSTAB 算法,迭代时间较少,且稳定性较好.

2）从图与表中的数据可知，当预处理算法的内迭代次数 $q=1$ 时，预处理算法（PGCORS2、PBiCORSTAB）与原算法的迭代次数比较接近，特别是测试矩阵 CDDE1 和 GR_30_30，这是由于此时的预处理条件子 $\boldsymbol{M}^{-1}=\boldsymbol{D}^{-1}$ 为对角阵，预处理算法相较于原算法影响很小.因此，当 $q=1$ 时，预处理算法与原算法的迭代次数几乎相同.

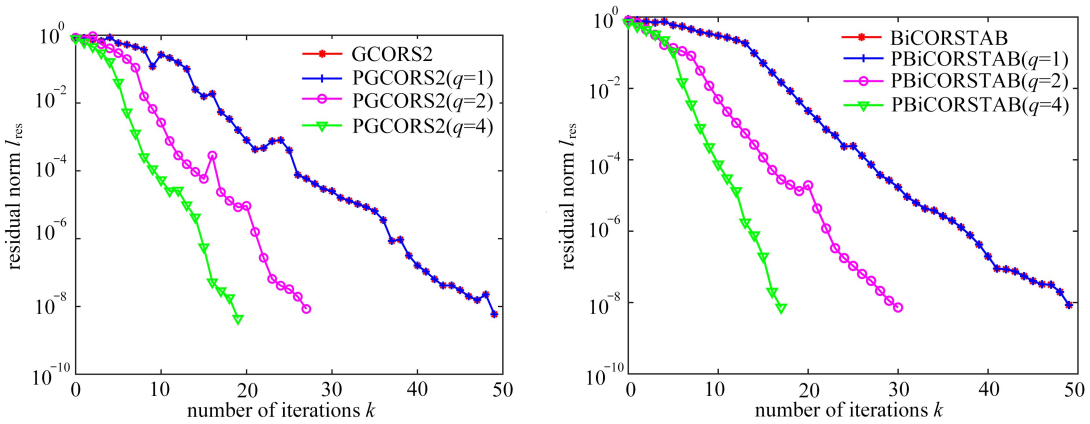
3）从图与表中的数据可知，对于同一测试算例，当内迭代次数 q 在一定范围时，PGCORS2 和 PBiCORSTAB 算法的收敛速度随着内迭代次数的增加而相应的加快，所用迭代时间相应减少.特别地，当内迭代次数 $q=4$ 时，预处理算法收敛速度最快，且迭代时间和迭代次数达到最小.



（a）PGCORS2　　（b）PBiCORSTAB

图 2　例 1 中 CDDE1 的收敛结果

Fig. 2　Convergence histories of CDDE1 for example 1



（a）PGCORS2　　（b）PBiCORSTAB

图 3　例 1 中 GR_30_30 的收敛结果

Fig. 3　Convergence histories of GR_30_30 for example 1

例 2　对于带有多个右端项的非 Hermite 线性方程组(2)，考虑阶数为 4 000 的复 Toeplitz 矩阵[13,15]：

$$\boldsymbol{A}=\begin{pmatrix} 4 & 0 & 1 & 0.7 & & \\ \mathrm{i}\gamma & 4 & 0 & 1 & 0.7 & \\ & \mathrm{i}\gamma & 4 & 0 & 1 & \ddots \\ & & \mathrm{i}\gamma & 4 & 0 & \ddots \\ & & & \ddots & \ddots & \ddots \end{pmatrix},$$

其符号函数为 $f(z)=\mathrm{i}\gamma z^{-1}+4+z^2+0.7z^3$.通过改变参数 γ, 可得到与符号函数 $f(z)$ 相关的谱和伪谱.因此,参数 γ 会影响到 Gl-GPBiCG[15]、Gl-GPBiCG_plus[15]、GCORS2[13] 及相关预处理方法的收敛性行为.方程(2)中右端项选取 $\boldsymbol{B}=\boldsymbol{A}\times\boldsymbol{E}, p=5\ll n$, 其中 $\boldsymbol{E}=(e_{ij})_{n\times p}$, 且 $e_{ij}=1$. 算例的计算结果对比详见表 2,相应算法的收敛性行为详见图 4 和图 5.

表 2 不同参数 γ 下预处理方法求解算例 2 的数值结果

Table 2 Numerical results for example 2 with different values of parameter γ obtained with preconditioning methods

method	$\gamma=2.0$			$\gamma=2.5$			$\gamma=2.7$		
	k	t/s	l_{res}	k	t/s	l_{res}	k	t/s	l_{res}
Gl-GPBiCG[15]	22	0.075	−8.010	46	0.155	−8.052	146	0.396	−8.043
PGl-GPBiCG ($q=1$)	22	0.076	−8.010	46	0.159	−8.052	146	0.419	−8.043
PGl-GPBiCG ($q=2$)	21	0.073	−8.283	21	0.093	−8.042	506	1.949	−8.001
PGl-GPBiCG ($q=4$)	10	0.048	−8.362	17	0.091	−8.337	43	0.183	−8.118
Gl-GPBiCG_plus[15]	23	0.061	−8.148	72	0.198	−8.075	302	0.787	−8.026
PGl-GPBiCG_plus ($q=1$)	23	0.063	−8.148	72	0.201	−8.075	302	0.792	−8.026
PGl-GPBiCG_plus ($q=2$)	15	0.052	−8.286	59	0.182	−8.205	486	1.662	−8.193
PGl-GPBiCG_plus ($q=4$)	10	0.048	−8.351	33	0.172	−8.153	110	0.479	−8.365
GCORS2[13]	17	0.053	−8.257	25	0.059	−8.059	34	0.082	−8.155
PGCORS2 ($q=1$)	17	0.054	−8.257	25	0.067	−8.059	34	0.090	−8.155
PGCORS2 ($q=2$)	12	0.046	−8.403	16	0.047	−8.289	19	0.079	−8.150
PGCORS2 ($q=4$)	7	0.043	−8.813	11	0.039	−8.725	13	0.076	−8.034

基于表 2、图 4 和图 5 的数值结果,采用 Gl-GPBiCG、Gl-GPBiCG_plus、GCORS2 及相应的预处理算法求解带有多个右端项的线性方程组(2)可得到如下相关结论:

1) 从表 2 中的数据可知,对于不同的参数 γ, GCORS2 算法所用的迭代时间和迭代次数均少于 Gl-GPBiCG 和 Gl-GPBiCG_plus 算法.且从图 4、图 5 中看出,当预处理算法的内迭代次数 $q=1$ 时,3 种预处理算法(PGCORS2、PGl-GPBiCG 和 PGl-GPBiCG_plus)与原算法的迭代次数几乎接近,收敛曲线几近重合,这一现象与例 1 中结果保持一致.

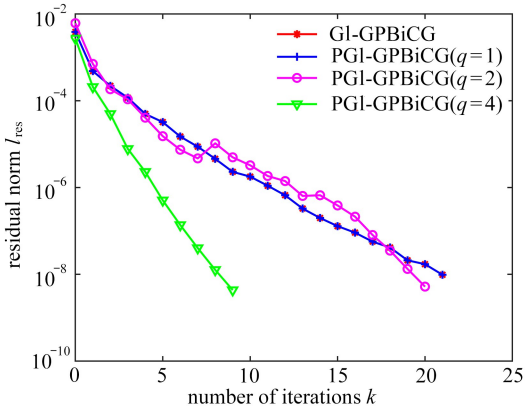
2) 从表 2 数据可知,当 $\gamma=2.0, 2.5$ 时,3 种算法的预处理算法的收敛速度均随着内迭代次数的增加而相应加快,所用迭代时间相应减少.特别地,当内迭代次数 $q=4$ 时,预处理算法收敛速度最快,且迭代时间和迭代次数达到最小.而当 $\gamma=2.7$ 时,可从图 5(a)、(b)中可知,PGl-GPBiCG 和 PGl-GPBiCG_plus 算法在内迭代次数为 $q=2$ 时,收敛性行为呈现骤增骤降趋势,且高于 $q=1$ 时的迭代次数,而 PGCORS2 算法却保持良好的收敛性行为.这说明参数 γ 会影响到 PGl-GPBiCG 和 PGl-GPBiCG_plus 算法的收敛行为,但对 PGCORS2 算法影响极小,因此 PGCORS2 算法呈现较好的收敛性行为.

基于算例 2 的结果,可知 GCORS2 算法的预处理算法表现出较好的收敛性行为.因此,以下算例进一步验证预处理算法求解带有多个右端项的线性方程组(2)的有效性和可行性.

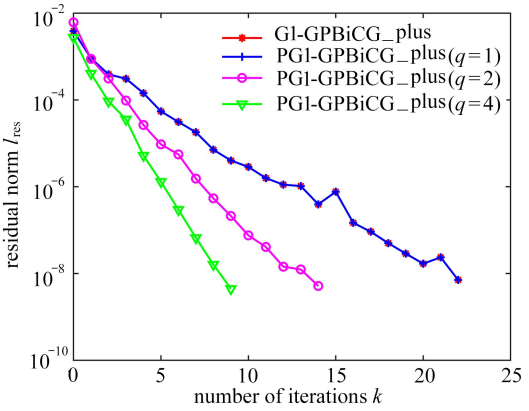
例 3 对于带有多个右端项的非 Hermite 线性方程组(2),考虑 Helmholtz 方程[4,13]:

$$\begin{cases}\Delta u+\alpha^2 u=0, & (x,y)\in[0,\pi]\times[0,\pi],\\ u_x(0,y)=\mathrm{i}\sqrt{\alpha^2-1/4}\cos(y/2),\\ u_x(\pi,y)-\mathrm{i}\sqrt{\alpha^2-1/4}\,u=0,\\ u_y(x,0)=u(x,\pi)=0.\end{cases}$$

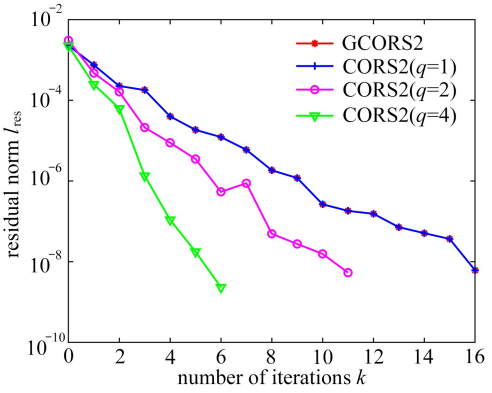
通过中心差分格式离散 Helmholtz 方程，离散后所得线性方程组系数矩阵阶数为 $(m+1)\times m$，其中步长选取 $h=1/m$，右端项选取 $\boldsymbol{B}=\boldsymbol{A}\times\boldsymbol{E}$，$p=5\ll n$，其中 $\boldsymbol{E}=(e_{ij})_{n\times p}$，且 $e_{ij}=1$.该算例仅考虑 $\alpha=2.77,4.16$ 两种情况.算例的计算结果对比详见表 3，相应算法的收敛性行为详见图 6.



(a) PGl-GPBiCG



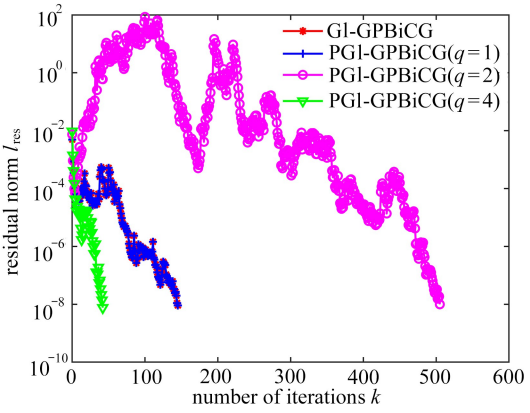
(b) PGl-GPBiCG_plus



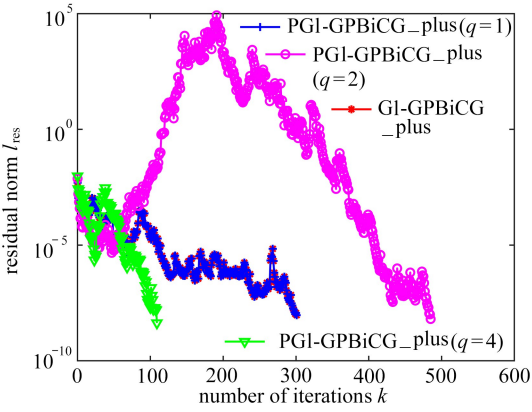
(c) PGCORS2

图 4 例 2 中参数 $\gamma=2.0$ 的收敛结果

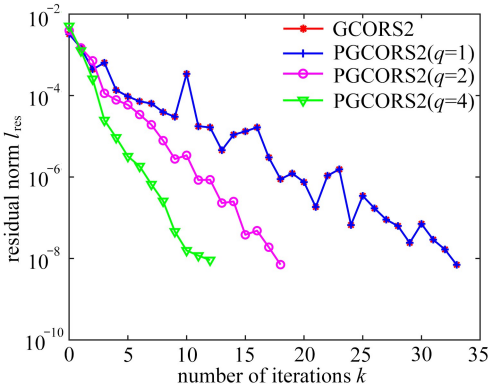
Fig. 4 Convergence histories for example 2 with $\gamma=2.0$



(a) PGl-GPBiCG



(b) PGl-GPBiCG_plus



(c) PGCORS2

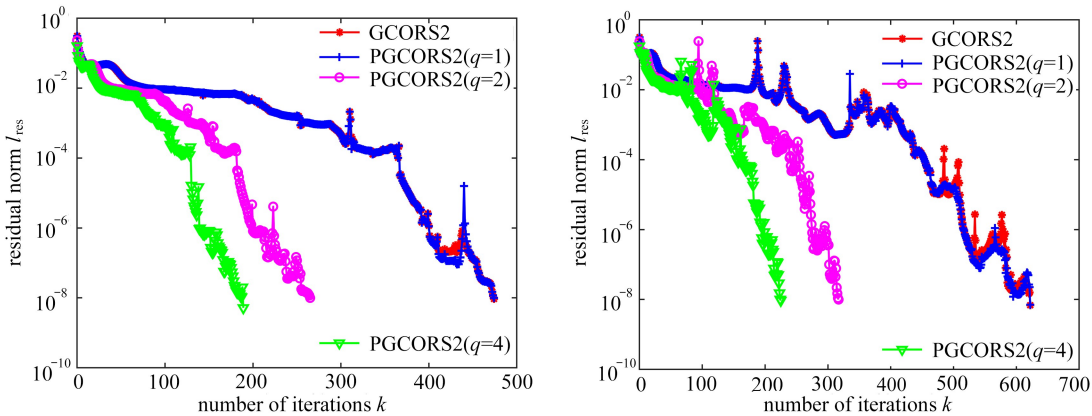
图 5 例 2 中参数 $\gamma = 2.7$ 的收敛结果

Fig. 5 Convergence histories for example 2 with $\gamma = 2.7$

表 3 不同参数 α 下预处理方法求解算例 3 的数值结果

Table 3 Numerical results for example 3 with different values of parameter α obtained with preconditioning methods

m	α	GCORS2[13]			PGCORS2 ($q = 1$)			PGCORS2 ($q = 2$)			PGCORS2 ($q = 4$)		
		k	t/s	l_{res}	k	t/s	l_{res}	k	t/s	l_{res}	k	t/s	l_{res}
80	2.77	317	1.77	−8.04	317	1.80	−8.03	188	1.23	−8.26	133	1.20	−8.11
100	2.77	396	2.14	−8.09	396	2.16	−8.15	223	1.76	−8.00	158	1.67	−8.22
120	2.77	475	7.65	−8.03	475	7.81	−8.00	266	5.75	−8.00	190	5.14	−8.29
80	4.16	431	2.21	−8.01	423	2.32	−8.02	331	1.93	−8.10	232	1.78	−8.01
100	4.16	490	2.33	−8.04	495	2.35	−8.02	374	2.05	−8.16	269	1.95	−8.14
120	4.16	623	6.38	−8.16	623	6.55	−8.15	318	4.76	−8.00	226	4.26	−8.03



(a) $\alpha = 2.77$ (b) $\alpha = 4.16$

图 6 例 3 中 PGCORS2 算法的收敛结果 ($m = 120$)

Fig. 6 Convergence histories for example 3 with $m = 120$

基于表 3 和图 6 数值结果,采用 GCORS2 算法及相应的预处理算法求解带有多个右端项的线性方程组(2)可得到如下相关结论:

1) 从表 3 中的数据可知,对于不同的参数 α, 当计算规模从 6 480($m = 80$) 增加到 14 520 ($m = 120$) 时,PGCORS2 依旧保持良好的收敛性行为.当 $q = 1$ 时,PGCORS2 与 GCORS2 的迭代

次数几乎接近,收敛曲线几近重合(可从图 6 中看出);当 $q = 4$ 时,PGCORS2 算法收敛速度最快,且迭代时间和迭代次数达到最小.

2) 从图 6 中可知,对于同一规模的 Helmholtz 方程,随着参数 α 的增加,PGCORS2 和 GCORS2 算法所用迭代次数增多.这说明参数 α 影响 GCORS2 和 PGCORS2 的收敛性行为,但 PGCORS2 算法相较于 GCORS2 算法依旧具有良好的收敛速度.

4 结束语

本文主要采用若干预处理迭代算法求解非 Hermite 线性方程组(1)和具有多个右端项的复线性方程组(2).通过预处理技术,有效地提高了求解线性方程组的效率,大大降低计算时间.以下针对相关预处理算法进行总结:

1) 考虑非 Hermite 线性方程组(1)的求解,PGCORS2 收敛速度和迭代时间均优于 PBi-CORSTAB 算法,具有良好的收敛性行为.考虑具有多个右端项的复线性方程组(2)的求解,PGCORS2 算法的收敛速度和迭代时间均优于 PGl-GPBiCG 算法和 PGl-GPBiCG_plus 算法,具有良好的收敛性行为.这进一步说明预处理算法加快了原算法的收敛速度.

2) 针对于文中所提到的所有预处理算法,当内迭代次数 $q = 1$ 时,由于此时的预处理条件子 $\boldsymbol{M}^{-1} = \boldsymbol{D}^{-1}$ 为对角阵,预处理算法相较于原算法影响很小.因此,预处理算法与原算法的迭代次数几乎接近,收敛曲线较为相似.当内迭代次数 $q = 4$ 时,预处理算法收敛速度均达到最快,迭代时间和迭代次数达到最小.

参考文献(References):

[1] BAZÁN F S V, KLEEFELD A, LEEM K H, et al. Sampling method based projection approach for the reconstruction of 3D acoustically penetrable scatterers[J]. *Linear Algebra and Its Applications*, 2016, **495**(15): 289-323.

[2] SADD Y. A flexible inner-outer preconditioned GMRES algorithm[J]. *SIAM Journal on Scientific Computing*, 1993, **14**(2): 461-469.

[3] BAI A, DAY D, DONGARRA J, et al. A test matrix collection for non-Hermitian eigenvalue problems: CS-97-355[R]. Knoxville, TN: Department of Computer Science, University of Tennessee, 1997.

[4] BAYLISS A, GLODSTEIN C I, TURKEL E. An iteration method for the Helmholtz equation [J]. *Journal of Computational Physics*, 1983, **49**(3): 443-457.

[5] HU Q Y, YUAN L. A plane-wave least-squares method for time-harmonic Maxwell's equations in absorbing media[J]. *SIAM Journal on Scientific Computing*, 2014, **36**(4): 1937-1959.

[6] HUTTUNEN T, MALINEN M, MONK P. Solving Maxwell's equations using the ultra weak variational formulation[J]. *Journal of Computational Physics*, 2007, **223**(2): 731-758.

[7] SAAD Y, SCHULTZ M H. A generalized minimum residual algorithm for solving nonsymmetirc linear systems[J]. *SIAM Journal on Scientific and Statistical Computing*, 1986, **7**(3): 856-869.

[8] DU K. GMRES with adaptively deflated restarting and its performance on an electromagnetic cavity problem[J]. *Applied Numerical Mathematics*, 2011, **61**(9): 977-988.

[9] VAN DERVORST H A. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems[J]. *SIAM Journal on Scientific and Statistical Computing*, 1992, **13**(2): 631-644.

[10] DEHGHAN M, MOHAMMADI-ARANI R. Generalized product-type methods based on bi-conjugate gradient (GPBiCG) for solving shifted linear systems[J]. *Computational and Applied Mathematics*, 2017, **36**(4): 1591-1606.

[11] ZHAO L, HUANG T Z, JING Y F, et al. A generalized product-type BiCOR method and its application in signal deconvolution[J]. *Computers and Mathematics With Applications*, 2013, **66**(8): 1372-1388.

[12] GU X M, HUANG T Z, CARPENTIERI B, et al. A hybridized iterative algorithm of the Bi-CORSTAB and GPBiCOR methods for solving non-Hermitian linear systems[J]. *Computers and Mathematics With Applications*, 2015, **70**(12): 3019-3031.

[13] ZHANG J H, DAI H. Generalized conjugate *A*-orthogonal residual squared method for complex non-Hermitian linear systems[J]. *Journal of Computational Mathematics*, 2014, **32**(3): 248-265.

[14] 张建华, 戴华. 求解具有多个右端项线性方程组的总体 CGS 算法[J]. 高等学校计算数学学报, 2008, **30**(4): 390-399.(ZHANG Jianhua, DAI Hua. Global CGS algorithm for linear systems with multiple right-hand sides[J]. *Numerical Mathematics a Journal of Chinese Universities*, 2008, **30**(4): 390-399.(in Chinese))

[15] ZHANG J H, DAI H. Global GPBiCG method for complex non-Hermitian linear systems with multiple right-hand sides[J]. *Computational and Applied Mathematics*, 2016, **35**(1): 171-185.

[16] 张建华. 非 Hermitian 线性方程组的若干迭代方法及其预处理[D]. 博士学位论文. 南京: 南京航空航天大学, 2016.(ZHANG Jianhua. Some iterative methods and their preconditioned variants for non-Hermitian linear systems[D]. PhD Thesis. Nanjing: Nanjing University of Aeronautics and Astronautics, 2016.(in Chinese))

[17] 富明慧, 李勇息. 求解病态线性方程组的预处理精细积分法[J]. 应用数学和力学, 2018, **39**(4): 462-469.(FU Minghui, LI Yongxi. A preconditioned precise integration method for solving ill-conditioned linear equations[J]. *Applied Mathematics and Mechanics*, 2018, **39**(4): 462-469.(in Chinese))

[18] DAVIS T A, HU Y F. The university of Florida sparse matrix collection[J]. *ACM Transactions on Mathematical Software*, 2011, **38**(1): 1-25.

Some Preconditioning Iterative Algorithms for Non-Hermitian Linear Equations

ZHANG Yingchun[1], LI Ying[2], XIAO Manyu[3], XIE Gongnan[1]

(1. *Department of Mechanical and Power Engineering, School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an* 710072, *P.R.China*;
2. *School of Information Technology, Shangqiu Normal University, Shangqiu, Henan* 476000, *P.R.China*;
3. *Department of Applied Mathematics, Northwestern Polytechnical University, Xi'an* 710129, *P.R.China*)

(Contributed by XIE Gongnan, M. AMM Editorial Board)

Abstract: Non-Hermitian linear equations have extensive application in scientific and engineering calculations and are expected to be solved with high efficiency. To accelerate the convergence rate of original algorithms, a preconditioning technique was developed and applied to some iterative methods chosen to solve the non-Hermitian linear equations and complex linear systems with multiple right-hand sides. Several numerical experiments show that the preconditioned iterative methods are superior to the original methods in terms of both the convergence rate and the number of iterations. In addition, the preconditioned generalized conjugate A- orthogonal residual squared method (GCORS2) has better convergent behavior and stability than other preconditioned methods.

Key words: non-Hermitian linear equations; generalized conjugate A-orthogonal residual squared method; preconditioning method

Foundation item: The National Natural Science Foundation of China(51676163)

引用本文/Cite this paper:

张迎春, 李英, 肖曼玉, 谢公南. 非 Hermite 线性方程组的若干预处理迭代算法[J]. 应用数学和力学, 2019, **40**(3): 237-249.

ZHANG Yingchun, LI Ying, XIAO Manyu, XIE Gongnan. Some preconditioning iterative algorithms for non-Hermitian linear equations[J]. *Applied Mathematics and Mechanics*, 2019, **40**(3): 237-249.