文章编号:1000-0887(2014)09-1011-09

一种求解鞍点问题的预处理并行算法*

姜晓林[1], 吕全义[1], 谢公南[2]

(1. 西北工业大学 应用数学系, 西安 710129;
2. 西北工业大学 机电学院;工程仿真与宇航计算技术联合实验室, 西安 710072)

(我刊编委谢公南来稿)

摘要: 研究了一种求解鞍点问题的并行预处理变形共轭梯度算法.通过应用迭代法进行预处理后,再采用变形共轭梯度求解的模式.首先构造系数矩阵近似逆的多项式表达式,以此作为预处理矩阵的逆矩阵,对方程组进行预处理;然后采用变形共轭梯度法并行求解预处理后的线性方程组.为减少运算量,采用迭代方式并行计算多项式与向量的乘法运算.通过调整迭代次数,即调整多项式次数,检验各种次数的多项式进行预处理后的求解方程的效果.数值试验结果表明,该算法明显优于未预处理的变形共轭梯度法,且当预处理迭代次数取 4 时效果最好.

关 键 词: 鞍点问题; 并行算法; 变形共轭梯度法; 预处理方法

中图分类号: O246 **文献标志码:** A

doi: 10.3879/j.issn.1000-0887.2014.09.007

引 言

鞍点问题是一类非常重要的线性方程组的求解问题,常见于科学计算以及工程应用领域,如 Navier-Stokes 方程的离散、第二类椭圆形方程的混合离散、计算流体力学问题、电磁学问题、最小二乘问题、约束优化问题和非线性规划问题等.它的应用已经渗透到应用数学、力学、电学等众多领域.对于鞍点问题,虽然串行算法已经比较完善,但是随着计算规模的增大,求解鞍点问题的存储要求和计算量快速增加,单台处理机往往难以承受.目前并行算法被认为是解决大规模鞍点问题最有效的途径之一,因此研究求解鞍点问题的高效并行算法具有重要的理论意义和实际意义.

一般来说,鞍点问题的求解方法主要分为两类:基于矩阵分裂的定常迭代法和基于 Krylov 子空间的不定常迭代法.对于定常迭代法,其中最为著名的就是 Uzawa-type 迭代法[1]和 HSS 类迭代法[2].对于不定常迭代法,Krylov 子空间法[3]是求解鞍点问题行之有效的方法.为了提高 Krylov 子空间迭代法的收敛速度,往往需要对系数矩阵进行预处理.国内外很多学者做了大量的工作,并给出了许多行之有效的预条件子,主要有块对角预条件子、约束预条件子、HSS 预条件子等.共轭梯度法(conjugate gradient method,简称为 CG 法)作为 Krylov 子空间法的一种基本方法更使得到了广泛的应用.CG 法具有存储量少、计算量少和适合并行计算等优点.但是它

* **收稿日期:** 2014-04-25; **修订日期:** 2014-06-30
基金项目: 陕西省自然科学基金(2009JM1008);国家自然科学基金(11202164)
作者简介: 姜晓林(1989—),男,山东人,硕士生(E-mail: jiangxiaolin1234@126.com);
吕全义(1963—),女,沈阳人,副教授(通讯作者. E-mail: luquan@nwpu.edu.cn).

只适用于系数矩阵为对称正定的情况,当系数矩阵非对称正定采用变形共轭梯度法(modified conjugate gradient method,简称为 MCG 法).MCG 法的收敛速度与系数矩阵的奇异值的密集度紧密相关,奇异值的密集度越大,收敛性越好.于是便出现了预处理变形共轭梯度(predisposed modified conjugate gradient,简称为 PMCG)法[4-5],它是通过引入预处理矩阵 $\boldsymbol{M}$ 将方程组进行恒等变形,使系数矩阵的奇异值分布更为集中,已达到提高收敛速度的目的.

本文主要求解 Navier-Stokes 方程离散化得到的线性方程组,其具有如下形式:

$$\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^{\mathrm{T}} & \boldsymbol{O} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{g} \end{pmatrix}, \tag{1}$$

其中 $\boldsymbol{A} \in R^{n\times n}$ 是对称正定矩阵,$\boldsymbol{B} \in R^{n\times m}$ 是列满秩矩阵,$m \leqslant n$, $\boldsymbol{f} \in R^n$,$\boldsymbol{g} \in R^m$.

为方便起见,设 $\boldsymbol{H}$ 为线性方程组(1) 的系数矩阵.本文采用矩阵多项式近似 $\boldsymbol{H}^{-1}$, 这样可以对预处理线性方程采取若干次迭代方法来求解,同时也可以通过调整迭代次数,使整个求解方程的计算效果达到最优.最后在 Lenovo(联想)深腾 1800 集群并行机上进行了数值试验,并与未预处理 MCG 法的计算结果进行了比较.本文涉及的向量和矩阵都是实的,向量范数为 2-范数.

1 变形共轭梯度法及并行实现

1.1 变形共轭梯度法

记线性方程组(1)为 $\boldsymbol{H}\tilde{\boldsymbol{x}} = \boldsymbol{b}$,其中 $\boldsymbol{H} \in R^{(m+n)\times(m+n)}$,$\tilde{\boldsymbol{x}} = (\boldsymbol{x}^{\mathrm{T}} \quad \boldsymbol{y}^{\mathrm{T}})^{\mathrm{T}}$,$\boldsymbol{b} = (\boldsymbol{f}^{\mathrm{T}} \quad \boldsymbol{g}^{\mathrm{T}})^{\mathrm{T}}$.变形共轭梯度算法如下[4]:

1) 任意给定初始向量 $\tilde{\boldsymbol{x}}^{(0)} \in R^{(m+n)}$,置 $k := 1$ 计算

$\boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{H}\tilde{\boldsymbol{x}}^{(0)}$, $\boldsymbol{q}^{(0)} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{r}^{(0)}$, $\boldsymbol{z}^{(0)} = \boldsymbol{q}^{(0)}$;

2) 若 $\boldsymbol{r}^{(k)} = \boldsymbol{0}$ 或者 $\boldsymbol{r}^{(k)} \neq \boldsymbol{0}$ 而 $\boldsymbol{z}^{(k)} = \boldsymbol{0}$ 停止计算,否则计算

$\boldsymbol{\alpha}_k = [\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)}] / [\boldsymbol{z}^{(k)}, \boldsymbol{z}^{(k)}]$, $\tilde{\boldsymbol{x}}^{(k+1)} = \tilde{\boldsymbol{x}}^{(k)} + \boldsymbol{\alpha}_k \boldsymbol{z}^{(k)}$;

3) 计算

$\boldsymbol{r}^{(k+1)} = \boldsymbol{b} - \boldsymbol{H}\tilde{\boldsymbol{x}}^{(k+1)}$, $\boldsymbol{q}^{(k+1)} = \boldsymbol{H}^{\mathrm{T}}\boldsymbol{r}^{(k+1)}$,

$\boldsymbol{\beta}_k = [\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(k+1)}] / [\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)}]$, $\boldsymbol{z}^{(k+1)} = \boldsymbol{q}^{(k+1)} + \boldsymbol{\beta}_k \boldsymbol{z}^{(k)}$;

4) 置 $k := k + 1$, 转 2).

注 $(\cdot)^{(k)}$ 表示迭代第 k 步计算得到的结果.

1.2 算法并行实现

为叙述方便,设处理机台数为 p,p 整除 n 及 m,且 $n = pl$ 与 $m = ps$,p_i($i = 1,2,\cdots,p$) 表示第 i 台处理机.将系数矩阵子块 $\boldsymbol{A}$,$\boldsymbol{B}$,$\boldsymbol{B}^{\mathrm{T}}$,常向量 $\boldsymbol{f}$,$\boldsymbol{g}$ 以及所需的一些向量表示成分块形式[6-7],记

$$\boldsymbol{A} = (\boldsymbol{A}_1^{\mathrm{T}}, \boldsymbol{A}_2^{\mathrm{T}}, \cdots, \boldsymbol{A}_p^{\mathrm{T}})^{\mathrm{T}},\ \boldsymbol{B} = (\boldsymbol{B}_1^{\mathrm{T}}, \boldsymbol{B}_2^{\mathrm{T}}, \cdots, \boldsymbol{B}_p^{\mathrm{T}})^{\mathrm{T}},\ \boldsymbol{B}^{\mathrm{T}} = (\tilde{\boldsymbol{B}}_1^{\mathrm{T}}, \tilde{\boldsymbol{B}}_2^{\mathrm{T}}, \cdots, \tilde{\boldsymbol{B}}_p^{\mathrm{T}})^{\mathrm{T}},$$

$$\tilde{\boldsymbol{x}}^{(k)} = ((\tilde{\boldsymbol{x}}_1^{(k)})^{\mathrm{T}}, (\tilde{\boldsymbol{x}}_2^{(k)})^{\mathrm{T}}, \cdots, (\tilde{\boldsymbol{x}}_p^{(k)})^{\mathrm{T}})^{\mathrm{T}},\ \tilde{\boldsymbol{y}}^{(k)} = ((\tilde{\boldsymbol{y}}_1^{(k)})^{\mathrm{T}}, (\tilde{\boldsymbol{y}}_2^{(k)})^{\mathrm{T}}, \cdots, (\tilde{\boldsymbol{y}}_p^{(k)})^{\mathrm{T}})^{\mathrm{T}},$$

$$\boldsymbol{r}_u^{(k)} = ((\boldsymbol{r}_{u1}^{(k)})^{\mathrm{T}}, (\boldsymbol{r}_{u2}^{(k)})^{\mathrm{T}}, \cdots, (\boldsymbol{r}_{up}^{(k)})^{\mathrm{T}})^{\mathrm{T}},\ \boldsymbol{r}_v^{(k)} = ((\boldsymbol{r}_{v1}^{(k)})^{\mathrm{T}}, (\boldsymbol{r}_{v2}^{(k)})^{\mathrm{T}}, \cdots, (\boldsymbol{r}_{vp}^{(k)})^{\mathrm{T}})^{\mathrm{T}},$$

$$\boldsymbol{q}_u^{(k)} = ((\boldsymbol{q}_{u1}^{(k)})^{\mathrm{T}}, (\boldsymbol{q}_{u2}^{(k)})^{\mathrm{T}}, \cdots, (\boldsymbol{q}_{up}^{(k)})^{\mathrm{T}})^{\mathrm{T}},\ \boldsymbol{q}_v^{(k)} = ((\boldsymbol{q}_{v1}^{(k)})^{\mathrm{T}}, (\boldsymbol{q}_{v2}^{(k)})^{\mathrm{T}}, \cdots, (\boldsymbol{q}_{vp}^{(k)})^{\mathrm{T}})^{\mathrm{T}},$$

$$\boldsymbol{z}_u^{(k)} = ((\boldsymbol{z}_{u1}^{(k)})^{\mathrm{T}}, (\boldsymbol{z}_{u2}^{(k)})^{\mathrm{T}}, \cdots, (\boldsymbol{z}_{up}^{(k)})^{\mathrm{T}})^{\mathrm{T}},\ \boldsymbol{z}_v^{(k)} = ((\boldsymbol{z}_{v1}^{(k)})^{\mathrm{T}}, (\boldsymbol{z}_{v2}^{(k)})^{\mathrm{T}}, \cdots, (\boldsymbol{z}_{vp}^{(k)})^{\mathrm{T}})^{\mathrm{T}},$$

$$\boldsymbol{f} = ((\boldsymbol{f}_1)^{\mathrm{T}}, (\boldsymbol{f}_2)^{\mathrm{T}}, \cdots, (\boldsymbol{f}_p)^{\mathrm{T}})^{\mathrm{T}},\ \boldsymbol{g} = ((\boldsymbol{g}_1)^{\mathrm{T}}, (\boldsymbol{g}_2)^{\mathrm{T}}, \cdots, (\boldsymbol{g}_p)^{\mathrm{T}})^{\mathrm{T}},$$

其中 $\boldsymbol{A}_i$,$\boldsymbol{B}_i$ 分别是 $l \times n$ 与 $l \times m$ 矩阵,$\tilde{\boldsymbol{B}}_i$ 是 $s \times n$ 矩阵,$\boldsymbol{f}_i$,$\tilde{\boldsymbol{x}}_i^{(k)}$,$\boldsymbol{r}_{ui}^{(k)}$,$\boldsymbol{q}_{ui}^{(k)}$,$\boldsymbol{z}_{ui}^{(k)}$ 均是 l 维列向量,$\boldsymbol{g}_i$, $\tilde{\boldsymbol{y}}_i^{(k)}$,$\boldsymbol{r}_{vi}^{(k)}$,$\boldsymbol{q}_{vi}^{(k)}$,$\boldsymbol{z}_{vi}^{(k)}$ 均是 s 维列向量.

1）存储方式：

将 $\boldsymbol{A}_i,\boldsymbol{B}_i,\tilde{\boldsymbol{B}}_i,\boldsymbol{f}_i,\boldsymbol{g}_i,\tilde{\boldsymbol{x}}_i^{(0)}$ 和 $\tilde{\boldsymbol{y}}_i^{(0)}$ 均存储在 $p_i(i=1,2,\cdots,p)$ 处理机中.

2）准备工作：

① $\boldsymbol{r}_u^{(0)}=\boldsymbol{f}-(\boldsymbol{A}\tilde{\boldsymbol{x}}^{(0)}+\boldsymbol{B}\tilde{\boldsymbol{y}}^{(0)})$ 和 $\boldsymbol{r}_v^{(0)}=\boldsymbol{g}-\boldsymbol{B}^{\mathrm{T}}\tilde{\boldsymbol{x}}^{(0)}$ 的计算：在 p_i 中，对初始向量 $\tilde{\boldsymbol{x}}_i^{(0)},\tilde{\boldsymbol{y}}_i^{(0)}$ 进行一次全收集后得到 $\tilde{\boldsymbol{x}}^{(0)}$ 和 $\tilde{\boldsymbol{y}}^{(0)}$，计算 $\boldsymbol{r}_{ui}^{(0)}=\boldsymbol{f}_i-(\boldsymbol{A}_i\tilde{\boldsymbol{x}}^{(0)}+\boldsymbol{B}_i\tilde{\boldsymbol{y}}^{(0)})$ 和 $\boldsymbol{r}_{vi}^{(0)}=\boldsymbol{g}_i-\tilde{\boldsymbol{B}}_i\tilde{\boldsymbol{x}}^{(0)}$.

② $\boldsymbol{q}_u^{(0)}=\boldsymbol{A}\boldsymbol{r}_u^{(0)}+\boldsymbol{B}\boldsymbol{r}_v^{(0)}$ 和 $\boldsymbol{q}_v^{(0)}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{r}_u^{(0)}$ 的计算：在 p_i 中，对向量 $\boldsymbol{r}_{ui}^{(0)},\boldsymbol{r}_{vi}^{(0)}$ 进行一次全收集后得到 $\boldsymbol{r}_u^{(0)}$ 和 $\boldsymbol{r}_v^{(0)}$，计算 $\boldsymbol{q}_{ui}^{(0)}=\boldsymbol{A}_i\boldsymbol{r}_u^{(0)}+\boldsymbol{B}_i\boldsymbol{r}_v^{(0)},\boldsymbol{q}_{vi}^{(0)}=\tilde{\boldsymbol{B}}_i\boldsymbol{r}_u^{(0)}$ 和 $(\boldsymbol{r}_u^{(0)},\boldsymbol{r}_u^{(0)})+(\boldsymbol{r}_v^{(0)},\boldsymbol{r}_v^{(0)})$；令 $\boldsymbol{z}_{ui}^{(0)}=\boldsymbol{q}_{ui}^{(0)}$ 和 $\boldsymbol{z}_{vi}^{(0)}=\boldsymbol{q}_{vi}^{(0)}$.

3）循环过程：

步骤 1 在 p_i 中，计算内积 $(\boldsymbol{z}_{ui}^{(k)},\boldsymbol{z}_{ui}^{(k)})+(\boldsymbol{z}_{vi}^{(k)},\boldsymbol{z}_{vi}^{(k)})$，全规约后得到 $(\boldsymbol{z}_u^{(k)},\boldsymbol{z}_u^{(k)})+(\boldsymbol{z}_v^{(k)},\boldsymbol{z}_v^{(k)})$，再计算 $\boldsymbol{\alpha}_k=((\boldsymbol{r}_u^{(k)},\boldsymbol{r}_u^{(k)})+(\boldsymbol{r}_v^{(k)},\boldsymbol{r}_v^{(k)}))/((\boldsymbol{z}_u^{(k)},\boldsymbol{z}_u^{(k)})+(\boldsymbol{z}_v^{(k)},\boldsymbol{z}_v^{(k)}))$.

步骤 2 在 p_i 中，计算 $\tilde{\boldsymbol{x}}_i^{(k+1)}=\tilde{\boldsymbol{x}}_i^{(k)}+\boldsymbol{\alpha}_k\boldsymbol{z}_{ui}^{(k)},\tilde{\boldsymbol{y}}_i^{(k+1)}=\tilde{\boldsymbol{y}}_i^{(k)}+\boldsymbol{\alpha}_k\boldsymbol{z}_{vi}^{(k)}$，对 $\tilde{\boldsymbol{x}}_i^{(k+1)}$ 和 $\tilde{\boldsymbol{y}}_i^{(k+1)}$ 进行一次全收集得到 $\tilde{\boldsymbol{x}}^{(k+1)}$ 和 $\tilde{\boldsymbol{y}}^{(k+1)}$；再计算 $\boldsymbol{r}_{ui}^{(k+1)}=\boldsymbol{f}_i-(\boldsymbol{A}_i\tilde{\boldsymbol{x}}^{(k+1)}+\boldsymbol{B}_i\tilde{\boldsymbol{y}}^{(k+1)}),\boldsymbol{r}_{vi}^{(k+1)}=\boldsymbol{g}_i-\tilde{\boldsymbol{B}}_i\tilde{\boldsymbol{x}}^{(k+1)}$.

步骤 3 在 p_i 中，对 $\boldsymbol{r}_{ui}^{(k+1)}$ 和 $\boldsymbol{r}_{vi}^{(k+1)}$ 进行一次全收集得到 $\boldsymbol{r}_u^{(k+1)}$ 和 $\boldsymbol{r}_v^{(k+1)}$，再计算 $\boldsymbol{q}_{ui}^{(k+1)}=\boldsymbol{A}_i\boldsymbol{r}_u^{(k+1)}+\boldsymbol{B}_i\boldsymbol{r}_v^{(k+1)},\boldsymbol{q}_{vi}^{(k+1)}=\tilde{\boldsymbol{B}}_i\boldsymbol{r}_u^{(k+1)}$ 和 $(\boldsymbol{r}_u^{(k+1)},\boldsymbol{r}_u^{(k+1)})+(\boldsymbol{r}_v^{(k+1)},\boldsymbol{r}_v^{(k+1)})$；若 $(\boldsymbol{r}_u^{(k+1)},\boldsymbol{r}_u^{(k+1)})+(\boldsymbol{r}_v^{(k+1)},\boldsymbol{r}_v^{(k+1)})<\varepsilon$，停止，否则继续计算.

步骤 4 在 p_i 中，计算 $\boldsymbol{M}\tilde{\boldsymbol{r}}^{(k)}=\boldsymbol{r}^{(k)}$，计算 $\boldsymbol{z}_{ui}^{(k+1)}=\boldsymbol{q}_{ui}^{(k+1)}+\boldsymbol{\beta}_k\boldsymbol{z}_{ui}^{(k)},\boldsymbol{z}_{vi}^{(k+1)}=\boldsymbol{q}_{vi}^{(k+1)}+\boldsymbol{\beta}_k\boldsymbol{z}_{vi}^{(k)}$.

步骤 5 置 $k:=k+1$，返回步骤 1.

由上面的并行实现过程可以发现：每迭代 1 步需要的通讯是 1 次全归约，2 次全收集，串行运算是 1 次内积；可见随着处理器的增多，并行性将会急速下降.

2 预处理变形共轭梯度法及并行实现

预处理技术是指在方程两边同时乘以预处理矩阵，将其化为一个与之等价但系数矩阵奇异值分布更集中的方程的技术.本文采用矩阵多项式构造预处理条件子 $\boldsymbol{M}^{-1}$，使其近似地等于系数矩阵 $\boldsymbol{H}^{-1}$.

2.1 预处理变形共轭梯度法

若 $\boldsymbol{M}$ 是预处理条件子，则预处理变形共轭梯度算法如下[5]：

1）任意给定初始向量 $\tilde{\boldsymbol{x}}^{(0)}\in R^{(m+n)}$，置 $k:=1$ 计算

$\boldsymbol{r}^{(0)}=\boldsymbol{b}-\boldsymbol{H}\tilde{\boldsymbol{x}}^{(0)}$，解 $\boldsymbol{M}\tilde{\boldsymbol{r}}^{(0)}=\boldsymbol{r}^{(0)}$，置 $\boldsymbol{p}^{(0)}=(\boldsymbol{M}^{-1}\boldsymbol{H})^{\mathrm{T}}\tilde{\boldsymbol{r}}^{(0)}$；

2）若 $\boldsymbol{r}^{(k)}=\boldsymbol{0}$ 或者 $\boldsymbol{r}^{(k)}\neq\boldsymbol{0}$ 而 $\boldsymbol{p}^{(k)}=\boldsymbol{0}$ 停止计算，否则计算

$\boldsymbol{\alpha}_k=[\tilde{\boldsymbol{r}}^{(k)},\tilde{\boldsymbol{r}}^{(k)}]/[\boldsymbol{p}^{(k)},\boldsymbol{p}^{(k)}]$，$\tilde{\boldsymbol{x}}^{(k+1)}=\tilde{\boldsymbol{x}}^{(k)}+\boldsymbol{\alpha}_k\boldsymbol{p}^{(k)}$；

3）计算

$\boldsymbol{r}^{(k+1)}=\boldsymbol{b}-\boldsymbol{H}\tilde{\boldsymbol{x}}^{(k+1)}$；

4）若 $(\boldsymbol{r}^{(k+1)},\boldsymbol{r}^{(k+1)})<\varepsilon$，停止计算，否则继续计算；

5）求解 $\boldsymbol{M}\tilde{\boldsymbol{r}}^{(k+1)}=\boldsymbol{r}^{(k+1)}$，计算

$\boldsymbol{\beta}_k=[\tilde{\boldsymbol{r}}^{(k+1)},\tilde{\boldsymbol{r}}^{(k+1)}]/[\tilde{\boldsymbol{r}}^{(k)},\tilde{\boldsymbol{r}}^{(k)}]$，

$\boldsymbol{p}^{(k+1)}=(\boldsymbol{M}^{-1}\boldsymbol{H})^{\mathrm{T}}\tilde{\boldsymbol{r}}^{(k+1)}+\boldsymbol{\beta}_k\boldsymbol{p}^{(k)}$；

6）置 $k:=k+1$，转 2）.

从上面的算法可以知道：PMCG 算法只是在 MCG 算法的基础上多求解了 2 个线性方程组

方程 $\boldsymbol{M}\tilde{\boldsymbol{r}}^{(k)}=\boldsymbol{r}^{(k)}$ 和 $\boldsymbol{M}^{\mathrm{T}}\boldsymbol{z}^{(k+1)}=\tilde{\boldsymbol{r}}^{(k+1)}$.为了加速 MCG 算法的收敛性,且尽量保持 MCG 算法的并行性,需要构造合适的预条件子 $\boldsymbol{M}$.

2.2 预处理方法

设 $\boldsymbol{H}=\boldsymbol{D}-\boldsymbol{N}$,且 $\boldsymbol{D}$ 可逆,若 $\boldsymbol{D}^{-1}\boldsymbol{N}$ 的谱半径 $\rho(\boldsymbol{D}^{-1}\boldsymbol{N})<1$, 则有

$$\boldsymbol{H}^{-1}=(\boldsymbol{I}-\boldsymbol{D}^{-1}\boldsymbol{N})^{-1}\boldsymbol{D}^{-1}=\sum_{i=0}^{\infty}(\boldsymbol{D}^{-1}\boldsymbol{N})^{i}\boldsymbol{D}^{-1},$$

取预处理条件子

$$\boldsymbol{M}^{-1}=(\boldsymbol{I}+\boldsymbol{D}^{-1}\boldsymbol{N}+(\boldsymbol{D}^{-1}\boldsymbol{N})^{2}+\cdots+(\boldsymbol{D}^{-1}\boldsymbol{N})^{q-1})\boldsymbol{D}^{-1}.$$

若 $\boldsymbol{D}$ 给定,便有 $\boldsymbol{N}=\boldsymbol{D}-\boldsymbol{H}$;则求解方程组 $\boldsymbol{M}\tilde{\boldsymbol{r}}^{(k)}=\boldsymbol{r}^{(k)}$ 的迭代格式为

任选 $\tilde{\boldsymbol{r}}^{(k,0)}$,

for $h=0,1,\cdots,q-1$(q 为常数)

$\tilde{\boldsymbol{r}}^{(k,h+1)}=\boldsymbol{D}^{-1}((\boldsymbol{D}-\boldsymbol{H})\tilde{\boldsymbol{r}}^{(k,h)}+\boldsymbol{r}^{(k)})$

end

$\tilde{\boldsymbol{r}}^{(k)}=\tilde{\boldsymbol{r}}^{(k,q)}$.

因为要求 $\boldsymbol{D}^{-1}$ 容易计算,且具有并行性,所以这里取 $\boldsymbol{D}$ 是对角矩阵.但由于系数矩阵 $\boldsymbol{H}$ 的特殊结构,故采用以下形式:

$$\boldsymbol{D}=\begin{pmatrix}\boldsymbol{D}_1 & \\ & \boldsymbol{D}_2\end{pmatrix},$$

其中 $\boldsymbol{D}_1$ 为矩阵 $\boldsymbol{A}$ 的对角矩阵,$\boldsymbol{D}_2$ 为矩阵 $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}$ 的对角矩阵.

类似地,便有求解方程组 $\boldsymbol{M}^{\mathrm{T}}\boldsymbol{z}^{(k+1)}=\tilde{\boldsymbol{r}}^{(k+1)}$ 的迭代格式:

任选 $\boldsymbol{z}^{(k+1,0)}$,

for $h=0,1,\cdots,q-1$(q 为常数)

$\boldsymbol{z}^{(k+1,h+1)}=\boldsymbol{D}^{-1}((\boldsymbol{D}-\boldsymbol{H})^{\mathrm{T}}\boldsymbol{z}^{(k+1,h)}+\tilde{\boldsymbol{r}}^{(k+1)})$

end

$\boldsymbol{z}^{(k+1)}=\boldsymbol{z}^{(k+1,q)}$.

通过构造系数矩阵近似逆的多项式作预处理条件矩阵的逆矩阵,并利用上面的迭代法迭代 q 次实现,这样使系数矩阵奇异值密集度会随着 q 的增大而更集中,从而加快 MCG 算法的收敛速度.

2.3 算法并行实现

为叙述方便,设处理机台数为 p,p 整除 n 及 m,且 $n=pl$ 与 $m=ps$,$p_i(i=1,2,\cdots,p)$ 表示第 i 台处理机.在 1.2 小节符号的基础上,还需引进

$$\boldsymbol{D}_1=(\boldsymbol{D}_{11}^{\mathrm{T}},\boldsymbol{D}_{12}^{\mathrm{T}},\cdots,\boldsymbol{D}_{1p}^{\mathrm{T}})^{\mathrm{T}},\ \boldsymbol{D}_2=(\boldsymbol{D}_{21}^{\mathrm{T}},\boldsymbol{D}_{22}^{\mathrm{T}},\cdots,\boldsymbol{D}_{2p}^{\mathrm{T}})^{\mathrm{T}},$$

$$\tilde{\boldsymbol{r}}_u^{(k)}=((\tilde{\boldsymbol{r}}_{u1}^{(k)})^{\mathrm{T}},(\tilde{\boldsymbol{r}}_{u2}^{(k)})^{\mathrm{T}},\cdots,(\tilde{\boldsymbol{r}}_{up}^{(k)})^{\mathrm{T}})^{\mathrm{T}},\ \tilde{\boldsymbol{r}}_v^{(k)}=((\tilde{\boldsymbol{r}}_{v1}^{(k)})^{\mathrm{T}},(\tilde{\boldsymbol{r}}_{v2}^{(k)})^{\mathrm{T}},\cdots,(\tilde{\boldsymbol{r}}_{vp}^{(k)})^{\mathrm{T}})^{\mathrm{T}},$$

$$\boldsymbol{p}_u^{(k)}=((\boldsymbol{p}_{u1}^{(k)})^{\mathrm{T}},(\boldsymbol{p}_{u2}^{(k)})^{\mathrm{T}},\cdots,(\boldsymbol{p}_{up}^{(k)})^{\mathrm{T}})^{\mathrm{T}},\ \boldsymbol{p}_v^{(k)}=((\boldsymbol{p}_{v1}^{(k)})^{\mathrm{T}},(\boldsymbol{p}_{v2}^{(k)})^{\mathrm{T}},\cdots,(\boldsymbol{p}_{vp}^{(k)})^{\mathrm{T}})^{\mathrm{T}},$$

其中 $\boldsymbol{D}_{1i},\boldsymbol{D}_{2i}$ 分别是 $l\times n$ 与 $s\times m$ 矩阵.$\tilde{\boldsymbol{r}}_{ui}^{(k)},\boldsymbol{p}_{ui}^{(k)}$ 均是 l 维列向量,$\tilde{\boldsymbol{r}}_{vi}^{(k)},\boldsymbol{p}_{vi}^{(k)}$ 均是 s 维列向量.

1) 存储方式

将 $\boldsymbol{A}_i,\boldsymbol{B}_i,\tilde{\boldsymbol{B}}_i,\boldsymbol{f}_i,\boldsymbol{g}_i,\tilde{\boldsymbol{x}}_i^{(0)}$ 和 $\tilde{\boldsymbol{y}}_i^{(0)}$ 均存储在 $p_i(i=1,2,\cdots,p)$ 处理机中.

2) 准备工作

① $\boldsymbol{r}_u^{(0)}=\boldsymbol{f}-(\boldsymbol{A}\tilde{\boldsymbol{x}}^{(0)}+\boldsymbol{B}\tilde{\boldsymbol{y}}^{(0)})$ 和 $\boldsymbol{r}_v^{(0)}=\boldsymbol{g}-\boldsymbol{B}^{\mathrm{T}}\tilde{\boldsymbol{x}}^{(0)}$ 的计算:在 p_i 中,对初始向量 $\tilde{\boldsymbol{x}}_i^{(0)},\tilde{\boldsymbol{y}}_i^{(0)}$ 进

行一次全收集后得到 $\tilde{\boldsymbol{x}}^{(0)}$ 和 $\tilde{\boldsymbol{y}}^{(0)}$；计算 $\boldsymbol{r}_{ui}^{(0)}=\boldsymbol{f}_i-(\boldsymbol{A}_i\tilde{\boldsymbol{x}}^{(0)}+\boldsymbol{B}_i\tilde{\boldsymbol{y}}^{(0)})$ 和 $\boldsymbol{r}_{vi}^{(0)}=\boldsymbol{g}_i-\tilde{\boldsymbol{B}}_i\tilde{\boldsymbol{x}}^{(0)}$.

② 预处理过程

$\boldsymbol{M}\begin{pmatrix}\tilde{\boldsymbol{r}}_u^{(0)}\\ \tilde{\boldsymbol{r}}_v^{(0)}\end{pmatrix}=\begin{pmatrix}\boldsymbol{r}_u^{(0)}\\ \boldsymbol{r}_v^{(0)}\end{pmatrix}$ 的求解：记 $\boldsymbol{D}_1=\mathrm{diag}(\tilde{\boldsymbol{D}}_{11},\tilde{\boldsymbol{D}}_{12},\cdots,\tilde{\boldsymbol{D}}_{1p})$，$\boldsymbol{D}_2=\mathrm{diag}(\tilde{\boldsymbol{D}}_{21},\tilde{\boldsymbol{D}}_{22},\cdots,\tilde{\boldsymbol{D}}_{2p})$，选初始向量 $\tilde{\boldsymbol{r}}_{ui}^{(0,0)}$，$\tilde{\boldsymbol{r}}_{vi}^{(0,0)}$，

for $h=0,1,\cdots,q-1$（q 为常数）

在 p_i 中，对 $\tilde{\boldsymbol{r}}_{ui}^{(h,0)}$ 和 $\tilde{\boldsymbol{r}}_{vi}^{(h,0)}$ 进行一次全收集得到 $\tilde{\boldsymbol{r}}_u^{(h,0)}$ 和 $\tilde{\boldsymbol{r}}_v^{(h,0)}$，计算

$\tilde{\boldsymbol{r}}_{ui}^{(h+1,0)}=\tilde{\boldsymbol{D}}_{1i}^{-1}((\boldsymbol{D}_{1i}-\boldsymbol{A}_i)\tilde{\boldsymbol{r}}_u^{(h,0)}-\boldsymbol{B}_i\tilde{\boldsymbol{r}}_v^{(h,0)}+\boldsymbol{r}_u^{(0)})$

$\tilde{\boldsymbol{r}}_{vi}^{(h+1,0)}=\tilde{\boldsymbol{D}}_{2i}^{-1}(-\tilde{\boldsymbol{B}}_i\tilde{\boldsymbol{r}}_u^{(h,0)}+\boldsymbol{D}_{2i}\tilde{\boldsymbol{r}}_v^{(h,0)}+\boldsymbol{r}_v^{(0)})$

end

$\tilde{\boldsymbol{r}}_{ui}^{(0)}=\tilde{\boldsymbol{r}}_{ui}^{(q,0)}$，$\tilde{\boldsymbol{r}}_{vi}^{(0)}=\tilde{\boldsymbol{r}}_{vi}^{(q,0)}$.

$\begin{pmatrix}\boldsymbol{p}_u^{(0)}\\ \boldsymbol{p}_v^{(0)}\end{pmatrix}=(\boldsymbol{M}^{-1}\boldsymbol{H})^{\mathrm{T}}\begin{pmatrix}\tilde{\boldsymbol{r}}_u^{(0)}\\ \tilde{\boldsymbol{r}}_v^{(0)}\end{pmatrix}$ 的计算：首先求解方程组 $\boldsymbol{M}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{z}_u^{(0)}\\ \boldsymbol{z}_v^{(0)}\end{pmatrix}=\begin{pmatrix}\tilde{\boldsymbol{r}}_u^{(0)}\\ \tilde{\boldsymbol{r}}_v^{(0)}\end{pmatrix}$，然后计算 $\begin{pmatrix}\boldsymbol{p}_u^{(0)}\\ \boldsymbol{p}_v^{(0)}\end{pmatrix}=\boldsymbol{H}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{z}_u^{(0)}\\ \boldsymbol{z}_v^{(0)}\end{pmatrix}$.在 p_i 中，选初始向量 $\tilde{\boldsymbol{z}}_{ui}^{(0,0)}$，$\tilde{\boldsymbol{z}}_{vi}^{(0,0)}$，求解方法跟上面迭代方法一样，迭代 q 次后，得到 $\boldsymbol{z}_{ui}^{(0)}$ 和 $\boldsymbol{z}_{vi}^{(0)}$.

③ 在 p_i 中，计算 $(\tilde{\boldsymbol{r}}_{ui}^{(0)},\tilde{\boldsymbol{r}}_{ui}^{(0)})+(\tilde{\boldsymbol{r}}_{vi}^{(0)},\tilde{\boldsymbol{r}}_{vi}^{(0)})$，全规约后得到 $(\tilde{\boldsymbol{r}}_u^{(0)},\tilde{\boldsymbol{r}}_u^{(0)})+(\tilde{\boldsymbol{r}}_v^{(0)},\tilde{\boldsymbol{r}}_v^{(0)})$.

④ 在 p_i 中，对 $\boldsymbol{z}_{ui}^{(0)}$，$\boldsymbol{z}_{vi}^{(0)}$ 进行一次全收集得到 $\boldsymbol{z}_u^{(0)}$ 和 $\boldsymbol{z}_v^{(0)}$，计算 $\boldsymbol{p}_{ui}^{(0)}=\boldsymbol{A}_i\boldsymbol{z}_u^{(0)}+\boldsymbol{B}_i\boldsymbol{z}_v^{(0)}$ 和 $\boldsymbol{p}_{vi}^{(0)}=\tilde{\boldsymbol{B}}_i\boldsymbol{z}_u^{(0)}$.

3）循环过程

步骤 1 在 p_i 中，计算内积 $(\boldsymbol{p}_{ui}^{(k)},\boldsymbol{p}_{ui}^{(k)})+(\boldsymbol{p}_{vi}^{(k)},\boldsymbol{p}_{vi}^{(k)})$，全规约后得到 $(\boldsymbol{p}_u^{(k)},\boldsymbol{p}_u^{(k)})+(\boldsymbol{p}_v^{(k)},\boldsymbol{p}_v^{(k)})$.

步骤 2 在 p_i 中，计算 $\boldsymbol{\alpha}_k=((\tilde{\boldsymbol{r}}_u^{(k)},\tilde{\boldsymbol{r}}_u^{(k)})+(\tilde{\boldsymbol{r}}_v^{(k)},\tilde{\boldsymbol{r}}_v^{(k)}))/((\boldsymbol{p}_u^{(k)},\boldsymbol{p}_u^{(k)})+(\boldsymbol{p}_v^{(k)},\boldsymbol{p}_v^{(k)}))$，再计算 $\tilde{\boldsymbol{x}}_i^{(k+1)}=\tilde{\boldsymbol{x}}_i^{(k)}+\boldsymbol{\alpha}_k\boldsymbol{p}_{ui}^{(k)}$，$\tilde{\boldsymbol{y}}_i^{(k+1)}=\tilde{\boldsymbol{y}}_i^{(k)}+\boldsymbol{\alpha}_k\boldsymbol{p}_{vi}^{(k)}$；然后对 $\tilde{\boldsymbol{x}}_i^{(k+1)}$ 和 $\tilde{\boldsymbol{y}}_i^{(k+1)}$ 进行一次全收集得到 $\tilde{\boldsymbol{x}}^{(k+1)}$ 和 $\tilde{\boldsymbol{y}}^{(k+1)}$，再计算 $\boldsymbol{r}_{ui}^{(k+1)}=\boldsymbol{f}_i-(\boldsymbol{A}_i\tilde{\boldsymbol{x}}^{(k+1)}+\boldsymbol{B}_i\tilde{\boldsymbol{y}}^{(k+1)})$，$\boldsymbol{r}_{vi}^{(k+1)}=\boldsymbol{g}_i-\tilde{\boldsymbol{B}}_i\tilde{\boldsymbol{x}}^{(k+1)}$.

步骤 3 在 p_i 中，计算 $(\boldsymbol{r}_{ui}^{(k+1)},\boldsymbol{r}_{ui}^{(k+1)})$ 和 $(\boldsymbol{r}_{vi}^{(k+1)},\boldsymbol{r}_{vi}^{(k+1)})$，全规约后得到 $(\boldsymbol{r}_u^{(k+1)},\boldsymbol{r}_u^{(k+1)})$ 和 $(\boldsymbol{r}_v^{(k+1)},\boldsymbol{r}_v^{(k+1)})$.若 $(\boldsymbol{r}_u^{(k+1)},\boldsymbol{r}_u^{(k+1)})+(\boldsymbol{r}_v^{(k+1)},\boldsymbol{r}_v^{(k+1)})<\varepsilon$，停止，否则继续计算.

步骤 4 求解 $\boldsymbol{M}\begin{pmatrix}\tilde{\boldsymbol{r}}_u^{(k+1)}\\ \tilde{\boldsymbol{r}}_v^{(k+1)}\end{pmatrix}=\begin{pmatrix}\boldsymbol{r}_u^{(k+1)}\\ \boldsymbol{r}_v^{(k+1)}\end{pmatrix}$.任选初始向量 $\tilde{\boldsymbol{r}}_{ui}^{(k+1,0)}$，$\tilde{\boldsymbol{r}}_{vi}^{(k+1,0)}$，求解方法跟上述迭代方法一样，迭代 q 次后在 p_i 中得到 $\tilde{\boldsymbol{r}}_{ui}^{(k+1)}$ 和 $\tilde{\boldsymbol{r}}_{vi}^{(k+1)}$.

步骤 5 在 p_i 中，计算内积 $(\tilde{\boldsymbol{r}}_{ui}^{(k+1)},\tilde{\boldsymbol{r}}_{ui}^{(k+1)})+(\tilde{\boldsymbol{r}}_{vi}^{(k+1)},\tilde{\boldsymbol{r}}_{vi}^{(k+1)})$，全规约后得到 $(\tilde{\boldsymbol{r}}_u^{(k+1)},\tilde{\boldsymbol{r}}_u^{(k+1)})+(\tilde{\boldsymbol{r}}_v^{(k+1)},\tilde{\boldsymbol{r}}_v^{(k+1)})$；再计算

$$\boldsymbol{\beta}_k=((\tilde{\boldsymbol{r}}_u^{(k+1)},\tilde{\boldsymbol{r}}_u^{(k+1)})+(\tilde{\boldsymbol{r}}_v^{(k+1)},\tilde{\boldsymbol{r}}_v^{(k+1)}))/((\tilde{\boldsymbol{r}}_u^{(k)},\tilde{\boldsymbol{r}}_u^{(k)})+(\tilde{\boldsymbol{r}}_v^{(k)},\tilde{\boldsymbol{r}}_v^{(k)})).$$

步骤 6 求解 $\boldsymbol{M}^{\mathrm{T}}\begin{pmatrix}\boldsymbol{z}_u^{(k+1)}\\ \boldsymbol{z}_v^{(k+1)}\end{pmatrix}=\begin{pmatrix}\tilde{\boldsymbol{r}}_u^{(k+1)}\\ \tilde{\boldsymbol{r}}_v^{(k+1)}\end{pmatrix}$.在 p_i 中，任选初始向量 $\boldsymbol{z}_{ui}^{(k+1,0)}$，$\boldsymbol{z}_{vi}^{(k+1,0)}$，求解方法与上述迭代方法一样，迭代 q 次后得到 $\boldsymbol{z}_{ui}^{(k+1)}$ 和 $\boldsymbol{z}_{vi}^{(k+1)}$.

步骤 7 在 p_i 中，对 $\boldsymbol{z}_{ui}^{(k+1)}$ 和 $\boldsymbol{z}_{vi}^{(k+1)}$ 进行一次全收集得到 $\boldsymbol{z}_u^{(k+1)}$ 和 $\boldsymbol{z}_v^{(k+1)}$，再计算

$$\boldsymbol{p}_{ui}^{(k+1)} = \boldsymbol{A}_i \boldsymbol{z}_u^{(k+1)} + \boldsymbol{B}_i \boldsymbol{z}_v^{(k+1)} + \boldsymbol{\beta}_k \boldsymbol{p}_{ui}^{(k)},\ \boldsymbol{p}_{vi}^{(k+1)} = \tilde{\boldsymbol{B}}_i \boldsymbol{z}_u^{(k+1)} + \boldsymbol{\beta}_k \boldsymbol{p}_{vi}^{(k)}.$$

步骤 8 置 $k := k+1$, 返回步骤 1.

从上面的并行实现过程可以知道,本文算法每迭代 1 步需要通讯是:2 次全收集,2 次全归约,并无串行计算,与 MCG 算法的并行性基本相同;另外,预处理迭代过程中,每迭代 1 步只需要 1 次全收集,并行性明显优于 MCG 算法,如果能够有效地提高收敛速度,本文算法一定优于 MCG 算法.

3 数值算例与结果分析

为了验证算法的有效性与并行性,在 Lenovo 深腾 1800 集群并行机上分别采用 MCG 法与本文给出的 PMCG 法对算例进行计算并比较.在迭代计算过程中,所有的初始向量均取零向量,终止条件 $\varepsilon = 10^{-8}$.

算例 考虑 Stokes 问题[8]

$$\begin{cases} -\mu\Delta \boldsymbol{u} + \nabla w = \tilde{\boldsymbol{f}}, & (x,y) \in \Omega, \\ \nabla \boldsymbol{u} = \tilde{\boldsymbol{g}}, & (x,y) \in \Omega, \\ \boldsymbol{u} = \boldsymbol{0}, & (x,y) \in \partial\Omega, \\ \int_\Omega w(x)\,\mathrm{d}x = 0, \end{cases}$$

其中, $\Omega = (0,1) \times (0,1) \subset R^2$, $\partial\Omega$ 是 Ω 的边界.采用流体力学离散化方法对其离散化,可将本题转化为求解方程组(1)的线性方程问题,其中

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I} \otimes \boldsymbol{T} + \boldsymbol{T} \otimes \boldsymbol{I} & 0 \\ 0 & \boldsymbol{I} \otimes \boldsymbol{T} + \boldsymbol{T} \otimes \boldsymbol{I} \end{bmatrix} \in R^{2l^2 \times 2l^2},\ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{I} \otimes \boldsymbol{F} \\ \boldsymbol{F} \otimes \boldsymbol{I} \end{bmatrix} \in R^{2l^2 \times l^2},$$

$$\boldsymbol{T} = \frac{1}{h^2} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix} \in R^{l \times l},$$

$$\boldsymbol{F} = \frac{1}{h} \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix} \in R^{l \times l},$$

$\otimes$ 是 Kronecker 内积, $h = 1/(l+1)$ 是离散网格尺寸.其中右端项的值取系数矩阵 $\boldsymbol{H}$ 每行元素之和.

显然,线性方程组(1)中的 $n = 2l^2$, $m = l^2$.当 $l = 20, 40$ 时,采用 MCG 法与 PMCG 法计算结果分别见表 1 和 2.

表 1,2 中 p 表示处理机台数, $T(s)$ 表示时间, K 表示迭代次数, E 表示相对并行效率, $\bar{E}$ 表示绝对并行效率, Δ 表示误差.由于一台处理机的计算时间较长,因此我们采用多台计算机的计算时间,并且使用多台处理机中最小的计算时间作为基准来计算加速比和并行效率.

表 1 计算结果 (l = 20)

Table 1 Numerical results (l = 20)

	p	1	2	4	8
MCG algorithm	T/s	39.21	19.53	9.91	7.32
	K	2 803	2 803	2 803	2 803
	E		0.99	0.99	0.67
	$\bar{E}$	0.67	0.67	0.66	0.45
	Δ	9.50E-09	9.50E-09	9.50E-09	9.50E-09
algorithm of this paper q = 1	T/s	43.70	21.03	10.45	8.19
	K	2 833	2 833	2 833	2 833
	E		0.99	0.99	0.67
	$\bar{E}$	0.61	0.63	0.63	0.40
	Δ	8.55E-09	8.55E-09	8.55E-09	8.55E-09
algorithm of this paper q = 2	T/s	30.02	14.26	7.09	6.39
	K	1 132	1 132	1 132	1 132
	E		0.99	0.99	0.59
	$\bar{E}$	0.88	0.93	0.93	0.52
	Δ	7.05E-09	7.05E-09	7.05E-09	7.05E-09
algorithm of this paper q = 4	T/s	26.45	12.18	6.12	5.48
	K	622	622	622	622
	E		0.99	0.99	0.60
	$\bar{E}$	1	0.99	0.99	0.60
	Δ	6.90E-09	6.90E-09	6.90E-09	6.90E-09

表 2 计算结果 (l = 40)

Table 2 Numerical results (l = 40)

	p	1	2	4	8
MCG algorithm	T/s	797.59	432.42	278.71	278.71
	K	13 642	13 642	13 642	13 642
	E		0.92	0.72	0.66
	$\bar{E}$	0.79	0.73	0.57	0.52
	Δ	7.40E-09	7.40E-09	7.40E-09	7.40E-09
algorithm of this paper q = 1	T/s	878.10	475.09	312.95	269.53
	K	13 704	13 704	13 704	13 704
	E		0.92	0.70	0.65
	$\bar{E}$	0.72	0.67	0.51	0.47
	Δ	6.99E-09	6.99E-09	6.99E-09	6.99E-09
algorithm of this paper q = 2	T/s	720.78	374.19	240.37	216.55
	K	5 704	5 704	5 704	5 704
	E		0.96	0.75	0.67
	$\bar{E}$	0.88	0.85	0.66	0.58
	Δ	8.44E-09	8.44E-09	8.44E-09	8.44E-09
algorithm of this paper q = 4	T/s	633.28	318.26	204.19	182.50
	K	2 921	2 921	2 921	2 921
	E		0.99	0.78	0.70
	$\bar{E}$	1	0.99	0.78	0.70
	Δ	7.15E-09	7.15E-09	7.15E-09	7.15E-09

由以上计算结果可以得到如下结论：

1）本文算法在预处理内迭代两步以上时，减少了迭代次数与运行时间，表明该预处理方

法提高了变形共轭梯度法的收敛速度,具有良好的并行性,且内迭代次数 $q = 4$ 时效果最好.

2) 比较表中数据,只迭代一次的预处理方法失效.当预处理内迭代一步时,就是取 $\boldsymbol{M}^{-1} = \boldsymbol{D}^{-1}$,由于该算例中矩阵 $\boldsymbol{A}$ 与 $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}$ 的主对角元素均相等,所以 $\boldsymbol{D}$ 是数量矩阵,故 $\boldsymbol{D}^{-1}\boldsymbol{H}$ 的奇异值分布与 $\boldsymbol{H}$ 相同,所以对本例只迭代一次的预处理方法无效.

3) 比较表中数据,当计算规模较小时,处理机的台数较少时,本文算法的并行效率可高达 0.98 以上,但是随着处理机台数的增加,由于整体内积的计算与全收集而引起的大规模通讯使得并行效率大大降低.当计算规模适当大时,处理机台数的增加,不会快速减少并行效率.可见在具有一定的计算规模时,且采用适当数量处理机的情况下,本文算法确实具有良好的并行性.

4 结束语

本文主要运用 PMCG 法求解 Stokes 方程离散化的线性方程组.首先构造系数矩阵近似逆的多项式表达式,以此作为预处理矩阵的逆矩阵,对方程组进行预处理;然后采用变形共轭梯度法并行求解预处理后的线性方程组.最后通过分别采用本文算法与 MCG 算法计算实例,测试本文算法的有效性.计算结果表明,本文算法不仅有效地提高了 MCG 算法的收敛速度,而且提高了并行效率.

致谢 作者对西北工业大学高性能计算研究与发展中心对本文的支持表示衷心的感谢.

参考文献(References):

[1] Elman H C, Golub G H. Inexact and preconditioned Uzawa algorithm for saddle point problems[J]. *SIAM Journal on Numerical Analysis*, 1994, **31**(6): 1645-1661.

[2] Benzi M, Gander M J, Golub G H. Optimization of the Hermitian and skew-Hermitian splitting iteration for saddle-point problems[J]. *BIT Numerical Mathematics*, 2003, **43**(5): 881-900.

[3] 李晓梅, 吴建平. 数值并行算法与软件[M]. 北京: 科学出版社, 2007.(LI Xiao-mei, WU Jianping. *Numerical Parallel Algorithm and Software*[M]. Beijing: Science Press, 2007.(in Chinese))

[4] 张凯院, 徐仲. 数值代数[M]. 第二版修订本. 北京: 科学出版社, 2010.(ZHANG Kai-yuan, XU Zhong. *Numerical Algebra*[M]. revised 2nd ed. Beijing: Science Press, 2010.(in Chinese))

[5] 胡家赣. 解线性代数方程组的迭代解法[M]. 北京: 科学出版社, 1999: 173-201.(HU Jia-gan. *Iterative Solution of Linear Algebraic Equations*[M]. Beijing: Science Press, 1999: 173-201.(in Chinese))

[6] 陈国良, 安虹, 陈俊, 郑启龙, 单九龙. 并行算法实践[M]. 北京: 高等教育出版社, 2004.(CHEN Guo-liang, AN Hong, CHEN Jun, ZHENG Qi-long, SHAN Jiu-long. *The Parallel Algorithm*[M]. Beijing: Higher Education Press, 2004.(in Chinese))

[7] 侯俊霞, 吕全义, 曹方颖 , 谢公南. 一种求解大型 Lyapunov 矩阵方程的预处理并行算法[J]. 应用数学和力学, 2013, **34**(5): 454-461.(HOU Jun-xia, LÜ Quan-yi, CAO Fang-ying, XIE Gong-nan. A preconditioned parallel method for solving large Lyapunov matrix equation[J]. *Applied Mathematics and Mechanics*, 2013, **34**(5): 454-461.(in Chinese))

[8] BAI Zhong-Zhi, Golub G H, PAN Jian-yu. Preconditioned Hermitian and skew-Hermian splitting methods for non-Hermitian positive semidefinite linear systems[J]. *Numerische Mathematik*, 2004, **98**(1): 1-32.

A Preconditioned Parallel Method for Solving Saddle Point Problems

JIANG Xiao-lin[1], LÜ Quan-yi[1], XIE Gong-nan[2]

(1. *Department of Applied Mathematics*, *Northwestern Polytechnical University*, *Xi'an* 710129, *P.R.China*;

2. *Engineering Simulation and Aerospace Computing*; *School of Mechanical Engineering*, *Northwestern Polytechnical University*, *Xi'an* 710072, *P.R.China*)

(Contributed by XIE Gong-nan, M. AMM Editorial Board)

Abstract: A parallel algorithm with preconditioned modified conjugate gradient method for solving saddle point problems was studied. It is a model that by using iterative method for preconditioning and applying modified conjugate gradient method for solving the problems. Firstly the approximate inverse of the coefficient matrix's polynomial expressions is constructed and become the inverse matrix of the preconditioned matrix, secondly the modified conjugate gradient method is used for parallel solving the preconditioned linear equations. In order to reduce the amount of calculation, we have to parallel compute the polynomials and vector multiplication by using iterative method. By adjusting the number of iterations and polynomials to exam the effect of preconditioning. The results show that our algorithm is superior to the modified conjugate gradient method and it has the best effect when the number of iterations is four.

Key words: saddle point problem; parallel algorithm; modified conjugate gradient method; precondition method

Foundation item: The National Natural Science Foundation of China(11202164)