文章编号:1000-0887(2010)05-0609-09

定常 Stokes 方程一种基于完全区域分解的有限元并行算法*

尚月强[1], 何银年[2]

(1. 贵州师范大学 数学与计算机科学学院,贵阳 550001;
2. 西安交通大学 理学院,西安 710049)

(郭兴明推荐)

摘要: 基于完全区域分解技巧,提出了一种求解定常 Stokes 方程的有限元并行算法.该算法中,所有子问题都是定义在整个求解区域上,但绝大部分自由度来自其所负责的子区域,从而使得算法稍加修改现有的串行程序即可实现相应的并行计算,实现简单,通信需求少.数值结果验证了算法的高效性.

关 键 词: Stokes 方程; 有限元方法; 并行算法; 完全区域分解

中图分类号: O246;O242.21 **文献标志码:** A

DOI: 10.3879/j.issn.1000-0887.2010.05.012

引 言

随着并行机的发展,有限元并行计算引起了越来越多的关注.特别是在计算流体力学领域,由于流体流动的复杂性,使得数值模拟中需要复杂的网格、足够多的网格节点和长时间的积分运算,从而导致计算规模很大,需借助高性能并行机才能满足其内存及计算需求.因此,探索高效的有限元并行算法对流体的数值模拟至关重要.

完全区域分解是一种有效的自适应网格生成方法[1-2].在该方法中,每个处理器除了生成其所负责的子区域部分的网格外,还生成少许粗网格单元以覆盖整个求解区域(关于完全区域分解技巧的详细描述,参见文献[1]).本文中,我们利用完全区域分解技巧,设计出求解定常 Stokes 方程的一种并行有限元离散方法.其基本思想是使用类似于局部加密的全局网格去计算 Stokes 方程在某一给定子区域的局部解.该网格在给定的子区域上较细,而在其它区域则较粗(参见图 1).具体说来,我们首先将求解区域分解成若干个互不重叠的子区域 $D_1,\cdots,D_J$,然后将每一子区域向外扩展一定尺寸获得相互重叠的区域分解 $\Omega_1,\cdots,\Omega_J$($D_j \subset\subset \Omega_j \subset \Omega$, $j=1,2,\cdots,J$.此处 $D_j \subset\subset \Omega_j \subset \Omega$ 表示 $\mathrm{dist}(\partial D_j \backslash \partial\Omega_j, \partial\Omega_j \backslash \partial\Omega) > 0$);每个处理器负责一个子区

* **收稿日期:** 2009-12-18; **修订日期:** 2010-04-15
基金项目: 国家自然科学基金资助项目(10971166);国家基础研究基金资助项目(2005CB321703);贵州省科学技术基金资助项目(2008(2123))
作者简介: 尚月强(1976—),男,贵州人,副教授,博士(E-mail:yueqiangshang@gmail.com);
何银年(1953—),男,陕西人,教授,博士(联系人. E-mail:heyn@mail.xjtu.edu.cn).

域,相互独立地在其负责的子区域生成尺寸为 h 的细网格、而在其余区域生成尺寸为 H 的粗网格. 该算法通信需求少,能充分利用现有的串行软件进行并行编程;并且,通过适当选取粗细网格的尺寸 H 和 h, 能获得与标准有限元方法(FE)相同的关于参数 h 的渐进收敛阶.

本文内容安排如下:第 1 节给出一些基本知识;第 2 节设计并分析基于完全区域分解的有限元并行算法;第 3 节给出数值算例;第 4 节给出相关结论.

1 Stokes 问题及其有限元逼近

设 Ω 是 $R^d(d=2,3)$ 中具有 Lipschitz 连续边界 $\partial\Omega$ 的有界区域. 我们用 $H^m(\Omega)$ 表示通常意义下的 Sobolev 空间[3-4],用 $\|\cdot\|_{m,\Omega}$ 表示相应的范数,用 $H_0^1(\Omega)$ 表示 $H^1(\Omega)$ 中在 $\partial\Omega$ 上迹为 0 的函数组成的子空间. 本文中,我们用字母 c 表示一大于 0 的常数,它不依赖于网格参数,在不同地方可能代表不同的值.

1.1 Stokes 问题

我们考虑下列不可压缩 Stokes 问题:

$$-\nu\Delta\boldsymbol{u}+\nabla p=\boldsymbol{f}, \qquad \text{在 } \Omega \text{ 内}, \tag{1}$$

$$\operatorname{div}\boldsymbol{u}=0, \qquad \text{在 } \Omega \text{ 内}, \tag{2}$$

$$\boldsymbol{u}=0, \qquad \text{在 } \partial\Omega \text{ 上}, \tag{3}$$

其中, $\boldsymbol{u}=(u_1,\cdots,u_d)$ 为速度向量,p 是压力, $\boldsymbol{f}=(f_1,\cdots,f_d)$ 是外力,ν 是粘性系数.

为了引进方程(1) ~ (3)的变分形式,我们设

$$X=H_0^1(\Omega)^d,\ Y=L^2(\Omega)^d,\ M=L_0^2(\Omega)=\left\{q\in L^2(\Omega):\int_\Omega q\mathrm{d}x=0\right\},$$

并定义 $a(\boldsymbol{u},\boldsymbol{v}),d(\boldsymbol{v},q)$ 如下:

$$a(\boldsymbol{u},\boldsymbol{v})=\nu(\nabla\boldsymbol{u},\nabla\boldsymbol{v}),\ d(\boldsymbol{v},q)=(\operatorname{div}\boldsymbol{v},q), \qquad \forall\boldsymbol{u},\boldsymbol{v}\in X,\ q\in M,$$

其中$(\cdot,\cdot)$为 $L^2(\Omega)^d(d=1,2,3)$ 上的标准内积.

在上述符号下,我们可得方程(1) ~ (3)的变分形式:

求 $(\boldsymbol{u},p)\in X\times M$, 使得

$$a(\boldsymbol{u},\boldsymbol{v})-d(\boldsymbol{v},p)+d(\boldsymbol{u},q)=(\boldsymbol{f},\boldsymbol{v}), \qquad \forall(\boldsymbol{v},q)\in X\times M. \tag{4}$$

关于方程(4)的解的存在唯一性,我们有下列结果[5-6]:

定理 1.1 设 Ω 是 R^d 中 C^{t+1}- 型有界区域$(t\geqslant 1)$,或凸多边形或凸多面体有界区域$(t=1)$,对于给定的 $\boldsymbol{f}\in H^{t-1}(\Omega)^d$, 方程(4)有唯一解

$$(\boldsymbol{u},p)\in(H^{t+1}(\Omega)^d\cap H_0^1(\Omega)^d)\times(H^t(\Omega)\cap L_0^2(\Omega)),$$

满足

$$\nu\|\boldsymbol{u}\|_{s+1,\Omega}+\|p\|_{s,\Omega}\leqslant c\|\boldsymbol{f}\|_{s-1,\Omega}, \qquad 0\leqslant s\leqslant t. \tag{5}$$

1.2 混合有限元逼近

设 $T^h(\Omega)=\{K\}$ 是 Ω 的一个尺寸为 $h(0<h<1)$ 的正则网格剖分[5-6],$X_h^0(\Omega)\subset X$, $M_h^0(\Omega)\subset M$ 是相应于网格 $T^h(\Omega)$ 的两个有限元空间,满足:

1) 逼近性质:任给

$$(\boldsymbol{u},p) \in H^{t+1}(\Omega)^d \times H^t(\Omega) \qquad (t \geqslant 1),$$

存在

$$(\pi_h \boldsymbol{u}, \rho_h p) \in X_h^0(\Omega) \times M_h^0(\Omega),$$

使得

$$\begin{cases} \| \boldsymbol{u} - \pi_h \boldsymbol{u} \|_{1,\Omega} \leqslant ch^s \| \boldsymbol{u} \|_{1+s,\Omega}, \\ \| p - \rho_h p \|_{0,\Omega} \leqslant ch^s \| p \|_{s,\Omega}, \end{cases} \qquad 0 \leqslant s \leqslant t, \tag{6}$$

2) 反估计:

$$\begin{cases} \| \boldsymbol{v} \|_{1,\Omega} \leqslant ch^{-1} \| \boldsymbol{v} \|_{0,\Omega}, \\ \| q \|_{0,\Omega} \leqslant ch^{-1} \| q \|_{-1,\Omega}, \end{cases} \qquad \forall (\boldsymbol{v},q) \in X_h^0(\Omega) \times M_h^0(\Omega); \tag{7}$$

3) inf-sup(LBB)条件:存在 $\beta > 0$, 使得

$$\beta \| q \|_{0,\Omega} \leqslant \sup_{\boldsymbol{v} \in X_h^0(\Omega), \boldsymbol{v} \neq 0} \frac{(\operatorname{div} \boldsymbol{v}, q)}{\| \nabla \boldsymbol{v} \|_{0,\Omega}}, \qquad \forall q \in M_h^0(\Omega). \tag{8}$$

在上述假设条件下,方程(4)的有限元逼近如下:

求 $(\boldsymbol{u}_h, p_h) \in X_h^0(\Omega) \times M_h^0(\Omega)$, 使得

$$a(\boldsymbol{u}_h, \boldsymbol{v}) - d(\boldsymbol{v}, p_h) + d(\boldsymbol{u}_h, q) = (\boldsymbol{f}, \boldsymbol{v}), \qquad \forall (\boldsymbol{v},q) \in X_h^0(\Omega) \times M_h^0(\Omega). \tag{9}$$

对于方程(9),我们有以下结果[5-6]:

定理 1.2 假设定理 1.1 的条件和式(6)~(8)成立,则方程(9)有唯一解 $(\boldsymbol{u}_h, p_h)$, 满足

$$\begin{aligned} &\nu \| \nabla(\boldsymbol{u} - \boldsymbol{u}_h) \|_{0,\Omega} + \| p - p_h \|_{0,\Omega} \leqslant \\ &\quad ch^s(\nu \| \boldsymbol{u} \|_{s+1,\Omega} + \| p \|_{s,\Omega}) \leqslant \\ &\quad ch^s \| \boldsymbol{f} \|_{s-1,\Omega}, \qquad 1 \leqslant s \leqslant t. \end{aligned} \tag{10}$$

2 有限元并行算法

2.1 完全区域分解技巧

首先将求解区域 Ω 分解成互不重叠的子区域 $D_1, D_2, \cdots, D_J$,然后将每一子区域 D_j 向外扩展一定尺寸获得 Ω_j,满足 $D_j \subset\subset \Omega_j \subset \Omega (j = 1,2,\cdots, J)$;每台处理器负责一个子区域,各自对所负责的子区域生成尺寸为 h 的细网格,并对其余区域生成尺度为 $H \gg h$ 的粗网格,记这样的整个区域的多尺度网格为 $T_j^{H,h}(\Omega)$ $(j = 1,2,\cdots,J)$. 这些多尺度网格 $T_j^{H,h}(\Omega)$ $(j = 1,2,\cdots,J)$ 是 Ω 的不同剖分,它们可通过对整个区域的初始粗网格 $T^H(\Omega)$ 施行局部加密并通过自适应过程使之相容得到(即任两个单元的交集或者为空集,或者为一公共顶点,或者为一公共边或为一公共面[4-5]). 这些 $T_j^{H,h}(\Omega)$ $(j = 1,2,\cdots,J)$ 构成 Ω 的一个完全区域分解. 图 1 描绘了将求解区域分成 4 个子区域时的网格剖分情形.

2.2 基于完全区域分解的有限元并行算法

设 $X_{H,h,j}^0(\Omega) \subset X, M_{H,h,j}^0(\Omega) \subset M$ 是相应于 $T_j^{H,h}(\Omega)$ $(j = 1,2,\cdots,J)$ 的有限元空间.

算法 1 有限元并行算法.

1) 并行求 $(\boldsymbol{u}_j^{H,h}, p_j^{H,h}) \in X_{H,h,j}^0(\Omega) \times M_{H,h,j}^0(\Omega)$ $(j = 1,2,\cdots,J)$, 使得

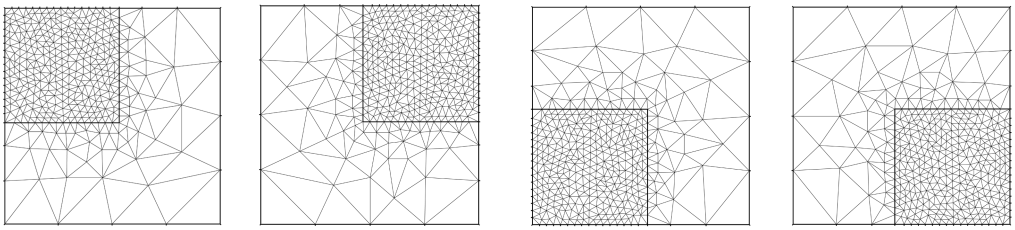


图1 4个子区域情形的完全区域分解

$$a(\boldsymbol{u}_j^{H,h},\boldsymbol{v}) - d(\boldsymbol{v},p_j^{H,h}) + d(\boldsymbol{u}_j^{H,h},q) = (\boldsymbol{f},\boldsymbol{v}),$$
$$\forall (\boldsymbol{v},q) \in X_{H,h,j}^0(\Omega) \times M_{H,h,j}^0(\Omega); \tag{11}$$

2) 在 D_j 中取$(\boldsymbol{u}^h,p^h) = (\boldsymbol{u}_j^{H,h},p_j^{H,h})$ $(j = 1,2,\cdots,J)$.

上述算法中所有子问题都是全局问题,它们定义在整个求解区域 Ω 上,但绝大部分自由度来自其负责的子区域,因而稍加修改现有的串行程序即可实现相应的并行计算.对于完全区域分解,则可由各处理器相互独立地使用某些串行自适应软件实现.

2.3 误差分析

定义分片范数

$$|||\ \boldsymbol{u} - \boldsymbol{u}^h\ |||_{1,\Omega} = \left(\sum_{j=1}^{J} \|\ \boldsymbol{u} - \boldsymbol{u}^h\ \|_{1,D_j}^2\right)^{1/2},$$
$$|||\ p - p^h\ |||_{0,\Omega} = \left(\sum_{j=1}^{J} \|\ p - p^h\ \|_{0,D_j}^2\right)^{1/2}.$$

在文献[7]中,使用两重网格方法和局部加密技巧,何银年教授等人提出了类似算法,并在某些假设条件(即逼近性、反估计、超收敛、稳定性)下,得到误差估计:

$$|||\ \boldsymbol{u} - \boldsymbol{u}^h\ |||_{1,\Omega} + |||\ p - p^h\ |||_{0,\Omega} \leqslant$$
$$c(h^s + H^{s+1})(\ \|\boldsymbol{u}\|_{s+1} + \|p\|_s), \qquad 1 \leqslant s \leqslant t, \tag{12}$$

其中 $c = c(\nu,\Omega)$,H 和 h 分别是粗细网格的尺寸.

仿文献[7]的证明思路并作简单改进,对于算法1,我们可得下列误差估计:

$$\nu\ |||\ \nabla(\boldsymbol{u} - \boldsymbol{u}^h)\ |||_{0,\Omega} + |||\ p - p^h\ |||_{0,\Omega} \leqslant$$
$$c(h^s + H^{s+1})\ \|\boldsymbol{f}\|_{s-1,\Omega}, \qquad 1 \leqslant s \leqslant t, \tag{13}$$

其中 c 与 ν 无关,

$$|||\ \nabla(\boldsymbol{u} - \boldsymbol{u}^h)\ |||_{0,\Omega} = \left(\sum_{j=1}^{J} \|\ \nabla(\boldsymbol{u} - \boldsymbol{u}^h)\ \|_{0,D_j}^2\right)^{1/2}.$$

式(13)与式(12)不同之处在于其明确地刻画了算法所得近似解的精度对参数 ν 的依赖性.式(13)表明:当我们选取粗网格的尺寸 $H = O(h^{s/(s+1)})$ 时,算法1可获得与标准有限元方法相同的收敛阶和最优渐进误差估计.

3 数值结果

本节我们给出两个数值算例以验证算法的有效性.数值试验平台为西安交通大学理学院智能信息处理与计算实验室的曙光集群并行机,其每个结点具有8核2.0 GHz CPU,2GB×8 内

存,各结点通过 20 Gbps InfiniBand 连在一起,信息传递软件为 MPICH.所使用的有限元为二阶 Taylor-Hood 元:

$$X^0_{H,h,j}(\Omega)=\{\boldsymbol{v}\in H^1_0(\Omega)^2:\boldsymbol{v}\mid_K\in P_2^2,\ \forall K\in T_j^{H,h}(\Omega)\},\qquad j=1,2,\cdots,J,$$

$$M^0_{H,h,j}(\Omega)=\{q\in L^2_0(\Omega)\cap C^0(\Omega):q\mid_K\in P_1,\ \forall K\in T_j^{H,h}(\Omega)\},\qquad j=1,2,\cdots,J,$$

其中 P_1 和 P_2 分别为一次和二次多项式函数空间.

3.1 解析解

本数值算例中,$\Omega=(0,1)\times(0,1)\subset R^2$,网格为非结构三角单元网格(参见图1),准确解为

$$\begin{cases}u_1=x^2(x-1)^2y(y-1)(2y-1),\\ u_2=-y^2(y-1)^2x(x-1)(2x-1),\\ p=3x^2+3y^2-2.\end{cases}\tag{14}$$

首先,为考察算法的渐进误差,我们将求解区域划分成 4 个互不重叠的子区域:

$$D_1=(0,1/2)\times(0,1/2),$$
$$D_2=(1/2,1)\times(0,1/2),$$
$$D_3=(0,1/2)\times(1/2,1),$$
$$D_4=(1/2,1)\times(1/2,1),$$

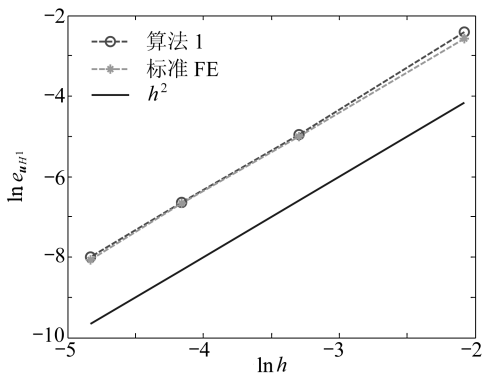
然后在 Ω 内将每一子区域向外扩展尺寸为 h 的区域获得 $\Omega_j(j=1,2,3,4)$,并分别就 $h=n^{-3}$,$H=h^{2/3}(n=2,3,4,5)$ 进行计算,其中 $\nu=1$.压力的零平均值由通常的标准有限元所采用的方法实现,相应的线性方程组使用 LU 分解求解.计算结果如表1(上半部分)所示,其中 T 是并行程序的运行时间(墙上时间)(单位:s),包括网格生成时间、方程求解时间和误差计算时间,N_t 为所有网格 $T_j^{H,h}(\Omega)$ $(j=1,2,\cdots,J)$ 中三角单元数的最小者,N_v 为相应的三角单元顶点数(即网格的节点数);而关于 h 的收敛率 R_u 和 R_p 由公式 $\ln(E_i/E_{i+1})/\ln(h_i/h_{i+1})$ 计算所得,其中 E_i 和 E_{i+1} 分别为网格尺寸为 h_i 和 h_{i+1} 时的相对误差.

表 1　近似解的误差:$\nu=1$,重叠尺寸为 $h\times 2$,P_2-P_1 元

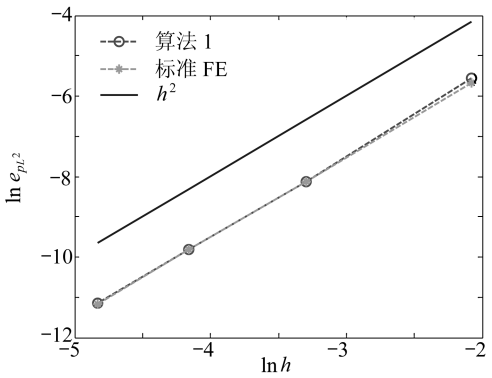
方法	h	H	N_t	N_v	T/s	$\frac{\|\|\|\nabla(\boldsymbol{u}-\boldsymbol{u}^h)\|\|\|_{0,\Omega}}{\|\nabla\boldsymbol{u}\|_{0,\Omega}}$	$\frac{\|\|\|p-p^h\|\|\|_{0,\Omega}}{\|p\|_{0,\Omega}}$	R_u	R_p
算法 1	1/8	1/4	92	58	1.29	0.089 304 1	0.003 863 58		
	1/27	1/9	826	442	1.73	0.007 001 3	0.000 293 416	2.093 03	2.119 18
	1/64	1/16	3 716	1 916	5.28	0.001 292 84	5.521 46E-05	1.957 32	1.935 43
	1/125	1/25	13 026	6 615	60.76	0.000 338 117	1.442 5E-05	2.003 5	2.005 08
标准 FE	1/8	-	152	93	1.36	0.076 146 4	0.003 473 7		
	1/27	-	1 720	915	2.64	0.006 723 29	0.000 295 054	1.995 3	2.027 15
	1/64	-	9 674	4 966	38.08	0.001 277 68	5.415 63E-05	1.924 03	1.964 3
	1/125	-	37 124	18 813	526.8	0.000 314 272	1.400 13E-05	2.095 13	2.020 71

为了与标准有限元方法进行比较,在表1(下半部分)中我们列出了使用非结构三角单元网格计算所得的标准有限元解的误差.由表1可以看出:我们的算法取得了与标准有限元方法

相同的收敛阶(参见图2).表1与图2表明:在所得近似解精度方面,我们的并行算法与标准有限元方法没有明显差别,但我们的并行算法大大节约了计算时间.



(a) 速度 $\boldsymbol{u}$ 的 H^1- 误差



(b) 压力 p 的 L^2- 误差

图2 近似解的误差

其次,为了考察参数 ν 对所得近似解精度的影响,我们令 $h = 1/125, H = 1/25$,分别就 $\nu = 10^{-n}(n = 0,1,2,3,4)$ 使用算法1和标准有限元方法计算相应的有限元解(见表2).结果表明:数值结果与理论结果完全一致,即:速度的精度随参数 ν 的变化而变化,而压力所受影响非常小,其中

$$K_{\mathrm{div}} = \max_{\substack{K \in T_j^{H,h}(D_j) \\ j=1,2,\cdots,J}} \left| \int_K \mathrm{div}\, \boldsymbol{u}^h \mathrm{d}x \right|,$$

它用来测量算法对流体不可压缩性质的逼近程度.一个有趣的发现是 K_{div} 的值也随参数 ν 的变化而变化;而且,在流体不可压缩性的逼近中,我们的并行算法取得了较标准有限元方法好的计算结果.

表2 近似解的误差:$h = 1/125, H = 1/25$, 重叠尺寸为 $h \times 2$, $P_2 - P_1$ 元

方法	ν	T/s	$\frac{\|\|\| \nabla(\boldsymbol{u} - \boldsymbol{u}^h) \|\|\|_{0,\Omega}}{\| \nabla \boldsymbol{u} \|_{0,\Omega}}$	$\frac{\|\|\| p - p^h \|\|\|_{0,\Omega}}{\| p \|_{0,\Omega}}$	K_{div}
算法 1	1	60.76	0.000 338 117	1.442 5E-05	1.066 72E-09
	0.1	60.75	0.003 015 65	1.409 61E-05	9.830 91E-09
	0.01	60.78	0.030 116 6	1.409 2E-05	9.747 28E-08
	0.001	60.76	0.301 16	1.409 19E-05	9.738 92E-07
	0.0001	60.66	3.011 6	1.409 23E-05	9.738 08E-06
标准 FE	1	526.8	0.000 314 272	1.400 13E-05	2.645 32E-09
	0.1	526.04	0.002 772 74	1.366 87E-05	2.687 63E-08
	0.01	526.31	0.027 687 9	1.366 54E-05	2.691 86E-07
	0.001	526.13	0.276 875	1.366 54E-05	2.692 28E-06
	0.0001	531.09	2.768 75	1.366 54E-05	2.692 32E-05

表3给出了不同的子区域重叠尺寸时算法的计算结果.其表明:当子区域的重叠程度加大时,并行算法所得近似解的精度有所提高.但与此同时,其计算时间也略有增加.

表 3　**近似解的误差：$\nu = 1, h = 1/125, H = 1/25, P_2 - P_1$ 元**

重叠尺寸	N_t	N_v	T/s	$\frac{\|\|\| \nabla(\boldsymbol{u} - \boldsymbol{u}^h) \|\|\|_{0,\Omega}}{\| \nabla \boldsymbol{u} \|_{0,\Omega}}$	$\frac{\|\|\| p - p^h \|\|\|_{0,\Omega}}{\| p \|_{0,\Omega}}$	K_{div}
$h \times 2$	13 026	6 615	60.76	0.000 338 117	1.442 5E-05	1.066 72E-09
$2h \times 2$	13 280	6 743	63.55	0.000 332 749	1.431 35E-05	7.577 18E-10
$3h \times 2$	13 722	6 965	67.06	0.000 324 617	1.411 52E-05	1.164 69E-09

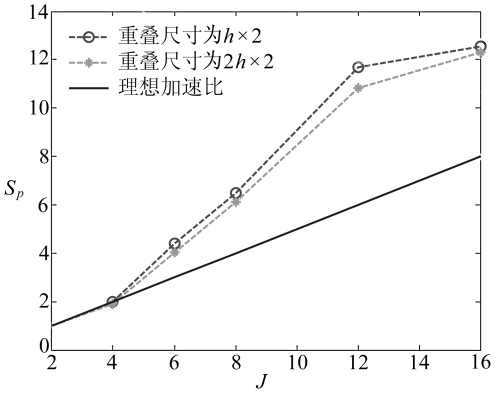
最后，为检验算法的并行性能，我们分别将求解区域分解成 2，4，6，8，12 和 16 个子区域，每个处理器负责一个子区域计算相应的有限元解，其中 $h = 1/125, H = 1/25, \nu = 1$. 表 4 列出了不同子区域数时并行程序的运行时间、由下列公式计算所得的加速比和并行效率：

$$S_p = \frac{T(2)}{T(J)}, \quad E_p = \frac{2 \times T(2)}{J \times T(J)}, \tag{15}$$

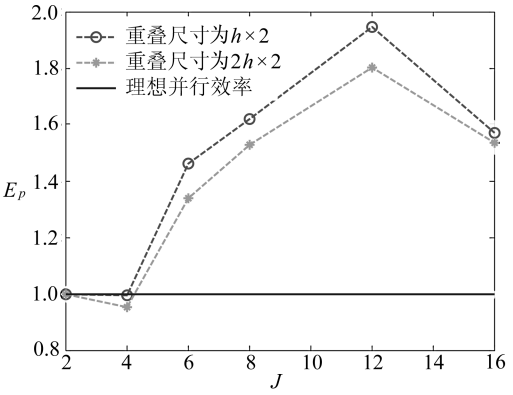
其中 J 为处理器（子区域）数，满足 $J \geqslant 2$，$T(2)$ 和 $T(J)$ 分别对应处理器数为 2 和 J 时并行程序的运行时间. 图 3 描绘了并行程序的加速比与并行效率随处理器数的变化情况. 由表 4 和图 3 可以看出：我们的并行算法具有良好的并行性能，取得了超线性的加速比和并行效率. 这种超线性加速比现象由所谓的 cache 效应所致.

表 4　**并行程序的运行时间 $T(J)$（单位：s）、加速比 $S_p = T(2)/T(J)$ 和并行效率 $E_p = 2 \times T(2)/(J \times T(J))$**

J	重叠尺寸 $= h \times 2$					重叠尺寸 $= 2h \times 2$				
	N_t	N_v	$T(J)$	S_p	E_p	N_t	N_v	$T(J)$	S_p	E_p
2	20 556	10 430	120.75	1.0	1.0	20 758	10 532	121.27	1.0	1.0
4	13 026	6 615	60.76	1.99	0.99	13 280	6 743	63.55	1.91	0.95
6	9 431	4 808	27.51	4.39	1.46	9 719	4 953	30.19	4.02	1.34
8	7 913	4 045	18.63	6.48	1.62	8 139	4 159	19.85	6.11	1.53
12	6 218	3 189	10.34	11.68	1.95	6 434	3 298	11.22	10.81	1.80
16	5 432	2 792	9.61	12.57	1.57	5 632	2 893	9.87	12.29	1.54



(a) 加速比

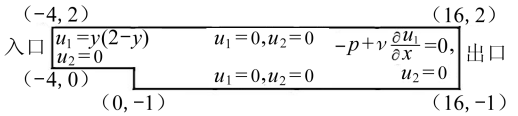


(b) 并行效率

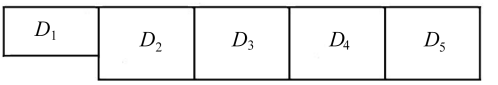
图 3　加速比与并行效率随处理器数的变化情况

3.2 后台阶问题

本算例考虑后台阶问题,它是检验一个算法有效性的很好的基准问题之一.其求解区域及边界条件如图4(a)所示,其中体积力 $\boldsymbol{f}=0$,$\nu=0.001$.我们将求解区域分解成5个互不重叠的子区域(见图4(b)),然后将每一子区域向外扩展尺寸为 H 的区域以获得相互重叠的区域分解,其中网格为 $h=1/8$,$H=1/4$ 的非结构三角单元网格,所使用的有限元为二阶 Taylor-Hood 元.

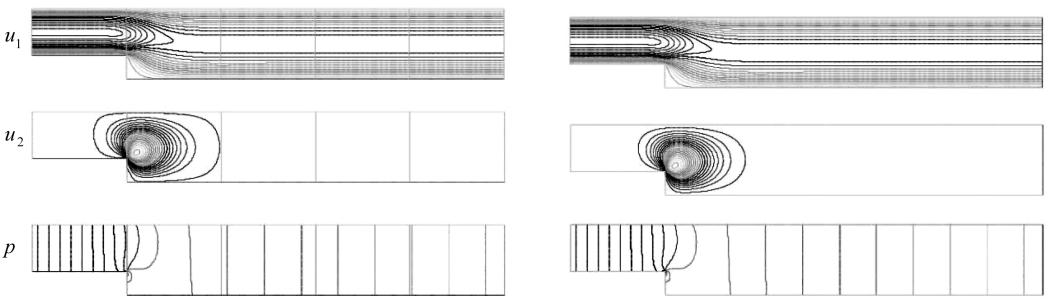


(a) 求解区域及边界条件



(b) 区域分解

图4 后台阶问题及其区域分解



(a) 算法1 (b) 标准有限元方法

图5 $\nu=0.001$ 时后台阶问题的等值线

图5描绘了算法1与标准有限元方法计算所得速度与压力的等值线.需注意的是图5的左边子图中 $x=0,4,8,12$ 处的垂直线是互不重叠的子区域 $D_j(j=1,\cdots,5)$ 的人工边界线.该算例进一步验证了并行算法的有效性.

4 结 束 语

基于完全区域分解技巧,本文提出了一种求解定常 Stokes 方程的有限元并行算法.该算法实现简单,具有良好的并行性能.数值结果验证了其高效性.

致谢 感谢西安交通大学理学院智能信息处理与计算实验室为本文提供了并行数值试验平台.

参考文献:

[1] Mitchell W F. The full domain partition approach to distributing adaptive grids[J]. *Appl Numer Math*, 1998, **26**(1/2):265-275.

[2] Mitchell W F. Parallel adaptive multilevel methods with full domain partitions[J]. *Appl Numer Anal Comput Math*, 2004, **1**(1/2):36-48.

[3] Adams R. *Sobolev Spaces*[M]. New York: Academic Press Inc,1975.

[4] Ciarlet P G, Lions J L. *Handbook of Numerical Analysis*[M]. Vol Ⅱ,Finite Element Methods (Part Ⅰ). Amsterdam: Elsevier Science Publisher, 1991.

[5] Girault V, Raviart P A. *Finite Element Methods for Navier-Stokes Equations: Theory and Al-*

gorithms[M]. Berlin Heidelberg: Springer-Verlag, 1986.

[6] Elman H C, Silvester D J, Wathen A J. *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*[M]. Oxford: Oxford University Press, 2005.

[7] HE Yin-nian, XU Jin-cao, ZHOU Ai-hui, *et al*. Local and parallel finite element algorithms for the Stokes problem[J]. *Numer Math*, 2008, **109**(3): 415-434.

A Parallel Finite Element Algorithm Based on Full Domain Partition for the Stationary Stokes Equations

SHANG Yue-qiang[1], HE Yin-nian[2]

(1. *School of Mathematics and Computer Science, Guizhou Normal University, Guiyang* 550001, *P. R. China*;
2. *Faculty of Science, Xi'an Jiaotong University, Xi'an* 710049, *P. R. China*)

Abstract: Based on full domain partition, a parallel finite element algorithm for the stationary Stokes equations was proposed and analyzed. In this algorithm, each subproblem was defined in the entire domain with the vast majority of the degrees of freedom associated with the particular subdomain that it was responsible for, and hence could be solved in parallel with other subproblems using an existing sequential solver without extensive recoding, allowing the algorithm to be implemented easily with low communication costs. Some numerical results are given which demonstrate the high efficiency of the parallel algorithm.

Key words: Stokes equations; finite element; parallel algorithm; full domain partition