基于模型减缩的并行共轭梯度算法研究

聂雪媛，丁　桦

（中国科学院力学研究所　水动力学与海洋工程重点实验室，北京 100190）

摘要：有限元并行计算被应用于求解大规模工程问题。为进一步提高计算效率，将模型减缩方法引入到并行计算区域划分步骤中，采用共轭梯度法，推导了适用于分布式并行环境的模型减缩并行算法。该方法在模型减缩的基础上，将边界节点进一步分为“本地节点”和“共享节点”，各任务处理器仅交换“共享节点”相关数据，降低了数据交换量，同时仅保存降阶后的区域矩阵，减少了内存存储。编制了 MPI 并行计算程序，并通过算例对该方法和普通有限元并行方法进行了比较，验证了算法的有效性。

关键词：模型减缩；并行计算；区域划分；共轭梯度法
中图分类号：O342　　**文献标识码**：A　　**文章编号**：0254-0053(2012)02-275-6

Model Reduction-Based Parallel Conjugate Gradient Method

NIE Xue-yuan, *DING Hua*

(Key Laboratory for Hydrodynamics and Ocean Engineering, Institute of Mechanics,
Chinese Academy of Sciences, Beijing 100190, China)

Abstract: The finite element parallel computation is applied to analysis of large-scale engineering problems. To make further improvement of computation efficiency, the model reduction method was employed in the procedure of parallel computation domain decomposition, then the parallel computation method with conjugate gradient arithmetic was derived for distributed parallel environment. The method divides the boundary nodes of one domain into “local nodes” and “common nodes” and only the data about “common nodes” are exchanged among the parallel processors. In this way, it reduces communication traffic. Moreover, it saves the reduction model's matrices to reduce the requirement for storage. The MPI parallel computation program was developed in the numerical example. By comparison between the method mentioned above and the general finite element parallel computation, the results testify the validity of the method.

Key words: model reduction; parallel computation; domain decomposition; conjugate gradient

有限元法(FEM)是当今科学和工程领域中分析连续物理系统应用最广泛的数值分析方法，该方法的要点是将结构按照某种原则离散为若干子域继而进行单元分析，传统的串行有限元分析是按单元顺序逐个进行的，每分析完成一个单元，就将其叠加到总体刚度矩阵中。当遇到大型或超大型复杂工程结构时，为保证数值解的精度，需借助大量的多节点高阶单元，将结构细分，此时，串行有限元分析方法受求解此类问题所需的计算时间长和内存大的限制，难以实现这些结构的计算，使其应用受到束缚，因此在科学研究和工程结构分析中引入高性能并行计算势在必行[1]。有限元法的“化整为零、积零为整”的基本思想与并行处理技术的基本原则“分而治之”基本一致，因此有限元法存在高度的内在并行性[2]。

可见串行有限元在大型复杂结构中的应用之所以受到制约，根源于结构自由度的剧增，导致计算量的

收稿日期：2011-09-05

作者简介：聂雪媛(1978-)，女，湖北武汉人，助研. 研究方向：并行计算及应用研究. E-mail：nie_xueyuan@yahoo.com.cn

增大。若在使用并行算法之前，先对大规模系统进行模型减缩，使系统的自由度减小，必将能进一步提高系统的计算效率。在一般有限元分析计算过程中，单元分析，生成总体刚度矩阵和总荷载向量以及方程组求解，是有限元求解的关键，且占据了整个有限元分析计算量的80%以上[3]。基于此，本文针对这三个计算量大的步骤，实现了并行计算，即在有限元模型的基础上，基于并行迭代求解的共轭梯度法，采用文献[4]提出的模型简化方法，给出了适用于分布存储并行机的模型减缩并行计算算法(Model Reduction Parallel Conjugate Gradient 简写为 MR-PCG)。对于数据交换，采用按需收集、按需散发的数据交换技术，以减少数据交换量。该算法采用 MPICH2 和 Fortran 语言编程实现，程序的基本框架对于一大类问题(刚度矩阵对称正定)的有限元法并行求解都是适用的。

1 模型减缩方法介绍

文献[4]中的模型减缩的基本思想是：在有限元划分模型的基础上，根据运动同步性假设，将结构划分为若干同步性区域，区域边界节点的自由度保留，内部节点的自由度则利用已知位移模态(如刚体位移模态)，由区域质心超节点(即用来表征内部节点运动模态的节点，本文选用表征刚体位移模态平动和转动自由度的质心节点)凝聚，达到减小整体结构的自由度数。由于刚体运动可以看作是位移同步性假设的极值状态，因此下面以刚体模态为例来描述运动同步性区域的位移(以位移元为例)即

$$u=\begin{Bmatrix} u_s \\ u_{i1} \\ \vdots \\ u_{il} \\ \vdots \\ u_{in} \end{Bmatrix}=\begin{bmatrix} I & & & & & \\ & R_1 & & & & \\ & & \ddots & & & \\ & & & R_l & & \\ & & & & \ddots & \\ & & & & & R_n \end{bmatrix}\begin{Bmatrix} u_s \\ q_{i1} \\ \vdots \\ q_{il} \\ \vdots \\ q_{in} \end{Bmatrix}=Rq \tag{1}$$

其中 u 为结构的节点位移向量，n 为划分的同步性区域个数，u_s 为划分区域的边界节点位移向量，u_{il} 为划分的第 l 个区域的内部节点位移向量($l=1,\cdots,n$)，q_l($l=1,\cdots,n$)为同步性区域的内部超节点位移向量，R_l($l=1,\cdots,n$)为内部第 l 个同步性区域节点的位移变换矩阵。该矩阵具体计算步骤如下：设 u_j 为第 l 个区域的第 j 个内部节点($j=1,\cdots,m$，m 代表该区域内部节点数)，由刚体运动的基本理论可知

$$u_j=\begin{Bmatrix} u_{jx} \\ u_{jy} \\ u_{jz} \end{Bmatrix}=\begin{bmatrix} 1 & 0 & 0 & 0 & z_{lj} & -y_{lj} \\ 0 & 1 & 0 & -z_{lj} & 0 & x_{lj} \\ 0 & 0 & 1 & y_{lj} & -x_{lj} & 0 \end{bmatrix}\begin{Bmatrix} q_{lx} \\ q_{ly} \\ q_{lz} \\ q_{l\theta_x} \\ q_{l\theta_y} \\ q_{l\theta_z} \end{Bmatrix}=R_j q_l$$

其中 x_{lj}，y_{lj}，z_{lj} 表示该区域中任一内部节点 j 的相应坐标与该区域超节点相应坐标的差值，由此可得第 l 个区域位移变换矩阵 $R_l=\begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_m \end{pmatrix}$。

经修正后的模型简化节点位移向量与原系统节点位移向量间的关系为

$$u=(R^T)^+R^TKRq=K^{-1}((R^TR)^{-1}R^T)^TR^TKRq=Tq \tag{2}$$

其中 K 为同步性区域的刚度矩阵。

将(2)带入结构动力学方程

$$M\ddot{U}+C\dot{U}+KU=F \tag{3}$$

及

$$M_R\ddot{q}+C_R\dot{q}+K_Rq=F_R \tag{4}$$

式中 M,C,K 为原始有限元结构的质量、阻尼和刚度矩阵；$M_R=T^TMT$，$C_R=T^TCT$，$K_R=T^TKT$ 为修正后简化结构的质量、阻尼和刚度矩阵，$F_R=T^TF$ 为简化模型的载荷向量。

2　模型减缩并行算法(MR-PCG)

基于模型减缩的并行算法就是以上述划分的同步性区域为求解单元，将求解单元的边界自由度再分为“本地节点”自由度(即该节点不被其他区域共享)和“共享节点”自由度(与其他区域共有的节点)。将每个求解单元映射到相应的处理器，在各处理器上所对应单元的质量/刚度矩阵 M_R、K_R 以及载荷向量 F_R。按照“本地节点”自由度和“共享节点”自由度进行分块。“本地节点”自由度可直接在当前处理器进行处理，“共享节点”自由度对应的矩阵元素通过相邻单元节点间的数据交换来形成，可避免总刚度阵和总质量阵的生成。此后各处理器上仅包含本求解单元的信息而不再涉及其他任何求解单元，所以可以独立进行求解，从而达到完全并行的目的。

在式(4)中，质量矩阵为协调质量矩阵。由于质量矩阵只和位移有关，因此在相同精度要求下，为简化计算可采用集中质量阵，同时为描述问题方便，本文仅考虑无阻尼项的系统。此时，(4)式变为

$$\begin{bmatrix}M_L & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\ddot{q}\\ \ddot{u}_s\end{bmatrix}+\begin{bmatrix}K_{qq} & K_{qs}\\ K_{sq} & K_{ss}\end{bmatrix}\begin{bmatrix}q\\ u_s\end{bmatrix}=\begin{bmatrix}F_q\\ F_s\end{bmatrix} \tag{5}$$

针对上述求解模型，具体算法过程如下

(1) 划分同步性区域，分配并行任务：

对大型结构划分 n 个同步性区域，并将区域边界节点分为“共享节点”和“本地节点”，由(5)式可得

$$\begin{bmatrix}M_L & & \\ & 0 & \\ & & 0\end{bmatrix}\begin{bmatrix}\ddot{q}\\ \ddot{u}_{sl}\\ \ddot{u}_{sc}\end{bmatrix}+\begin{bmatrix}K_{qq} & K_{ql} & K_{qc}\\ K_{lq} & K_{ll} & K_{lc}\\ K_{cq} & K_{cl} & K_{cc}\end{bmatrix}\begin{bmatrix}q\\ u_{sl}\\ u_{sc}\end{bmatrix}=\begin{bmatrix}F_q\\ F_{sl}\\ F_{sc}\end{bmatrix} \tag{6}$$

划分的第 i 个同步性区域的动力学方程为

$$\begin{bmatrix}M^i_{qq} & & \\ & 0 & \\ & & 0\end{bmatrix}\begin{bmatrix}\ddot{q}^i\\ \ddot{u}^i_{sl}\\ \ddot{u}^i_{sc}\end{bmatrix}+\begin{bmatrix}K^i_{qq} & K^i_{ql} & K^i_{qc}\\ K^i_{lq} & K^i_{ll} & K^i_{lc}\\ K^i_{cq} & K^i_{cl} & K^i_{cc}\end{bmatrix}\begin{bmatrix}q^i\\ u^i_{sl}\\ u^i_{sc}\end{bmatrix}=\begin{bmatrix}F^i_q\\ F^i_{sl}\\ F^i_{sc}\end{bmatrix} \tag{7}$$

式中上标 i 代表划分的第 i 个同步性区域($i=1,2,\cdots,n$)，u^i_{sl} 为第 i 个区域“本地节点”位移向量，u^i_{sc} 为第 i 个区域“共享节点”位移向量，M^i_{qq} 为第 i 个区域的集中质量阵，$K_{cc}=\sum_{i=1}^{n}K^i_{cc}$，$F_{sc}=\sum_{i=1}^{n}F^i_{sc}$，$u_{sc}=\sum_{i=1}^{n}u^i_{sc}$，(注：本文中的 $\sum$ 不是简单的叠加而是将每个区域的矩阵元素根据节点编号“对号入座”的集成，其意义见[5])。

将划分的 n 个区域映射到 n 个处理器，在第 i 个处理器上求解的方程组即式(7)。

每个处理器上进行以下步骤的运算

(2) 初始化，给定阈值 ε，和初值：$q^i(0)=0$；$\begin{pmatrix}u^i_{sl}(0)\\ u^i_{sc}(0)\end{pmatrix}=0$

(3) 求解 $q^i(k+1)$，$u^i_s(k+1)$，$k=0,1,\cdots$

(3.1)采用中心差分法显式求解 $q^i(k+1)$。将(7)式第一行展开得

$$M^i_{qq}\ddot{q}^i = F^i_q - K^i_{qq}q^i - K^i_{ql}u^i_{sl} - K^i_{qc}u^i_{sc} \tag{8}$$

该式中各项仅与本域内的节点信息相关，在各自的处理器上即可完成计算。因此可直接计算出 $\ddot{q}^i(k)$，采用中心差分法，求解超节点在 $k+1$ 时刻的位移为

$$q^i(k+1) = \ddot{q}^i(k)\Delta t^2 + 2q^i(k) - q^i(k-1)$$

(3.2)求解区域边界节点位移 $u^i_s(k)$：将式(7)中的后两项展开整理得

$$\begin{bmatrix} K^i_{ll} & K^i_{lc} \\ K^i_{cl} & K^i_{cc} \end{bmatrix} \begin{bmatrix} u^i_{sl} \\ u^i_{sc} \end{bmatrix} = \begin{bmatrix} F^i_{sl} - K^i_{lq}q^i \\ F^i_{sc} - K^i_{cq}q^i \end{bmatrix} \tag{9}$$

由于 K^i_{cc}，F^i_{sc} 包含了与其他区域共享的边界节点，需要进行相邻区域间共享节点的通信。采用并行共轭梯度法实现。

(a) 计算
$$\begin{aligned} r^{i(0)}_{sl} &= K^i_{ll}u^i_{sl}(k) + K^i_{lc}u^i_{sc}(k) - F^i_{sl} - K^i_{lq}q^i(k) \\ r^{i(0)}_{sc} &= K^i_{cl}u^i_{sl}(k) + K^i_{cc}u^i_{sc}(k) - F^i_{sc} - K^i_{cq}q^i(k) \end{aligned}$$

将与第 i 个区域相邻的所有区域对该区域的“共享节点”的贡献叠加，即 $\hat{r}^{i(0)}_{sc} = \sum_{j=1}^{neighbour(i)} r^{j(0)}_{sc}$ (neighbour(i)为与第 i 个区域相邻的区域个数，包括第 i 个区域)，该累加通过分布式处理器上数据通讯实现。

(b) 令 $d^{i(l)}_{sl} = -r^{i(l)}_{sl}$，$d^{i(l)}_{sc} = -\hat{r}^{i(l)}_{sc}$，$\hat{d}^{i(l)} = (d^{i(l)}_{sl}, d^{i(l)}_{sc})^T$，$l := 0$

$\hat{r}^{i(l)} = (r^{i(l)}_{sl} \quad \hat{r}^{i(l)}_{sc})^T$

计算标量 $\gamma^{(l)} = \text{Innerprod}(\hat{r}^{i(l)}, \hat{r}^{i(l)})$

InnerProd 具体实现为[6]

Function Innerprod(a,b)

$$c = \sum_{j=1}^{size(u^i_{sl})} a(j) * b(j) + \sum_{j=u^i_{sc}+1}^{size(u^i_{sc}+u^i_{sc})} (a(j) * b(j)/\text{shared}(j))$$

MPIAllReduce (c,Innerprod,MPISUM)

End function

其中 shared(j)为本处理器所对应区域的各“共享节点”被其他区域共享的次数(包括本区域)，若 $\sqrt{\gamma^{(l)}} \leqslant \varepsilon$，则停止迭代，令 $k = k+1$ 转至(3)。

(c) 计算

$$w^{i(l)}_{sl} = K^i_{ll}d^{i(l)}_{sl} + K^i_{lc}d^{i(l)}_{sc}$$

$$w^{i(l)}_{sc} = K^i_{cl}d^{i(l)}_{sl} + K^i_{cc}d^{i(l)}_{sc}$$

$$\hat{w}^{i(l)}_{sc} = \sum_{j=1}^{neighbour(i)} w^{j(l)}_{sc}$$

$$w^{i(l)} = (w^{i(l)}_{sl}, \hat{w}^{i(l)}_{sc})^T$$

$$\alpha^{(l)} = \frac{\gamma^{(l)}}{\text{Innerprod}(d^{(l)}_s, w^{(l)}_s)}$$

(d) $u^{i(l+1)}_{sl} = u^{i(l)}_{sl} + \alpha^{(l)}d^{i(l)}_{sl}$

$$u^{i(l+1)}_{sc} = u^{i(l)}_{sc} + \alpha^{(l)}d^{i(l)}_{sc}$$

(e) $r^{i(l+1)}_{sl} = r^{i(l)}_{sl} + \alpha^{(l)}w^{i(l)}_{sl}$

$$\hat{r}^{i(l+1)}_{sc} = \hat{r}^{i(l)}_{sc} + \alpha^{(l)}\hat{w}^{i(l)}_{sc}$$

$$\hat{r}^{i(l+1)} = (r^{i(l)}_{sl} \quad \hat{r}^{i(l+1)}_{sc})^T$$

(f) $\gamma^{(l+1)}=\text{innerproduct}(\hat{r}^{i(l+1)},\ \hat{r}^{i(l+1)})$

(g) $\beta^{(l)}=\dfrac{\gamma^{(l+1)}}{\gamma^{(l)}}$

$$d_{sl}^{i(l+1)}=-r_{sl}^{i(l+1)}+\beta^{(l)}d_{sl}^{i(l)}$$

$$d_{sc}^{i(l+1)}=-\hat{r}_{sc}^{i(l+1)}+\beta^{(l)}d_{sc}^{i(l)}$$

若 $\sqrt{\gamma^{(l+1)}}\leqslant\varepsilon$ 则 $u_{sl}^{i}(k+1)=u_{sl}^{i(l+1)}$，$u_{sc}^{i}(k+1)=u_{sc}^{i(l+1}$，$k=k+1$，转至(3)，否则 $l=l+1$，转至(c)。迭代是以残差 $\|\gamma\|_2\leqslant\varepsilon$ 为收敛准则。

并行实现时，在每个处理器上只存储该区域的信息，包括在该处理器上生成的刚度阵 M_R^i，K_R^i 以及仅作用在该区域节点上的载荷。本算法中所涉及到的处理器间的数据交换仅仅只交换与“共享节点”相关的数据，以保证通信开销不会太大。此外还需要注意的是，在(a)中涉及到相邻区域对“共享节点”贡献的叠加，只有各相邻区域的“共享节点”必须按照一定的顺序(即都按照节点由小到大或者由大到小的顺序)进行编号，才能保证叠加的正确。

3　算例

对本文所介绍的算法，采用 Fortran 语言，编写了 MPI 程序。并在银河高性能计算机系统中实现。

图 1 为由两种材料构成的悬臂梁模型。左部分几何尺寸：0.5×0.5m，材料参数：$E=1.0\times10^{10}$ Pa，$\mu=0.25$，$\rho=4500\text{kg/m}^3$，右部分几何尺寸 0.3×0.3m 材料参数：$E=1.0\times10^{9}$ Pa，$\mu=0.3$，$\rho=4000\text{kg/m}^3$，自由端受图 2 所示冲击载荷的作用。以 MR-PCG 法实现并行，和有限元法求解进行比较，来对程序进行验证。

以采用有限元法直接求解的结果作为标准，将模型(见图 1)用本文方法划分为 2 个同步性区域进行计算，如图 3 所示，图中●为“本地节点”，○为“共享节点”，＋为“超节点”。

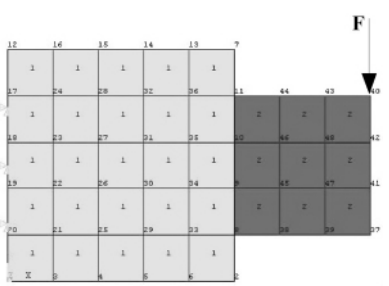

图 1　有限元模型

Fig. 1　Finite elements model

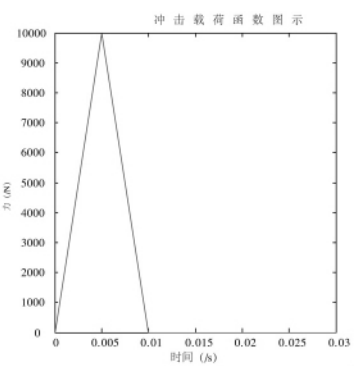

图 2　载荷模型

Fig. 2　Load model

模型施加冲击载荷后加载点的位移时程与有限元求解的对比如图 4 所示。

并行加速比是并行计算中常用的性能指标[7]。它是指完成同样的计算量时，串行耗时与并行耗时之比。对算例采用有限元直接并行化和用本方法并行化所得的加速比分别为 1.89，1.96，应用本方法的加速比大于进程数是由于同时使用了模型减缩和并行计算而产生的双重功效。

4　结论

基于模型减缩的共轭梯度并行算法将模型节点自由度减缩和分布式并行算法结合，使并行计算达到了事半功倍的效果，进一步提高了计算效率。从本文的分析中可以得到以下结论：

(1) 在实现并行计算之前，采用模型减缩对所划分区域进行预处理，能使进行并行计算的任务模型维

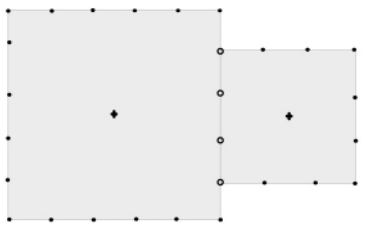

图 3 划分 2 个同步性区域

Fig. 3 Two domains with movement synchronization

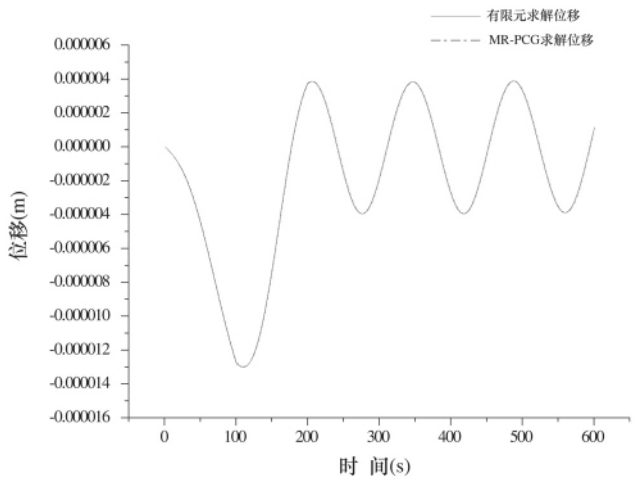

图 4 加载点位移比较

Fig. 4 Comparison of displacement of load point

数降低，节省内存需求和计算开支；

（2）采用按需收集、按需发送的数据交换形式，在每个并行任务间仅交换“共享节点”相关信息，尽可能减少通信开销；

（3）计算实例表明，与传统并行有限元计算相比，在加速比性能方面得到了改善，对于大型复杂结构的计算，该性能指标的提高将更加显著。

参考文献：

[1] 周树荃，梁维泰，邓绍忠. 有限元结构分析并行计算[M]. 北京：科学出版社，1999.

[2] Horie T, Kuramae H. Possibilities of workstation cluster for parallel finite element analysis [J]. Microcomputers in Civil Engineering, 1997, 12(2): 129－139.

[3] 刘耀儒. 三维有限元并行计算及其在水利工程中的应用[D]. 北京：清华大学，2003.

[4] 丁桦，郑淑飞，聂雪媛. 基于变形修正的动力减缩算法及超单元构造方法 [P]. 中国专利：200810102136.8，2011－05－11.

[5] 王勖成. 有限单元法[M]. 北京：清华大学出版社，2003.

[6] 付朝江. 集群 MPI 环境下有限元结构分析并行计算[D]. 上海：上海大学，2006.

[7] Lauria M, Chien A. MPI-FM: High performance MPI on workstation clusters[J]. Journal of Parallel and Distributed Computing, 1997, 40(1): 4－18.