doi:10.6041/j.issn.1000-1298.2015.04.050

基于稀疏存储的有限元结构分析高效缩聚并行计算方法*

苗新强[1]　金先龙[1]　丁峻宏[2]

(1. 上海交通大学机械与动力工程学院, 上海 200240; 2. 上海超级计算中心, 上海 201203)

摘要: 基于稀疏存储技术和直接稀疏求解器提出了一种有限元结构分析高效缩聚并行计算方法。该方法将缩聚过程转换为一系列线性方程组的求解过程,并通过直接稀疏求解器进行求解。它能够避免传统变带宽格式缩聚并行计算方法对带宽内大量零元素的存储和运算,从而大幅度节省内存空间和有效减少计算量。最后通过发动机曲轴的有限元数值仿真实验对算法的有效性进行了验证。结果表明:相对传统变带宽格式缩聚并行计算方法,稀疏存储格式缩聚并行计算方法能够大幅度节省内存空间和有效提高计算效率;各子区域规模越大,该方法对内存空间的节省和计算效率的提高效果就越明显。

关键词: 高性能计算　有限元分析　稀疏存储技术　直接稀疏求解器　缩聚

中图分类号: TP311; O246　　**文献标识码**: A　　**文章编号**: 1000-1298(2015)04-0338-06

Efficient Condensation Parallel Computing Method for Finite Element Structural Analysis Based on Sparse Storage Scheme

Miao Xinqiang[1]　Jin Xianlong[1]　Ding Junhong[2]

(1. *School of Mechanical Engineering*, *Shanghai Jiaotong University*, *Shanghai* 200240, *China*

2. *Shanghai Supercomputer Center*, *Shanghai* 201203, *China*)

Abstract: An efficient condensation parallel computing method for finite element structural analysis was proposed based on the sparse storage techniques and direct sparse solvers. In the proposed method, the process of condensation was converted to the process of solving a series of linear equations, and then the linear equations were solved with a direct sparse solver. It can avoid the storage and computation of many zero elements within the bandwidth in the traditional parallel computing method with variable bandwidth format condensation. Therefore, the memory space can be greatly saved and the amount of computation can be effectively reduced. Finally, the experiment of the finite element numerical simulation for an engine crankshaft was used to validate the proposed method. Test results showed that, compared with the conventional parallel computing method with variable bandwidth format condensations, the proposed parallel computing approach with sparse storage format condensation could considerably save memory space and significantly improve computational efficiency. The larger the size of each subdomain, the effect of the proposed method on aspects of saving memory space and improving computational efficiency was more obvious. The proposed method can be applied to many industrial areas such as aerospace, automobile, energy, civil and architecture to significantly improve the efficiency of engineering design and analysis.

Key words: High performance computing　Finite element analysis　Sparse storage technique　Direct sparse solver　Condensation

收稿日期: 2014-05-03　修回日期: 2014-06-26

*国家高技术研究发展计划(863 计划)资助项目(2012AA01A307)和国家自然科学基金资助项目(11272214、51475287)

作者简介: 苗新强,博士生,主要从事高性能计算和数值仿真研究,E-mail: newsturdy_miao@163.com

通讯作者: 金先龙,教授,博士生导师,主要从事高性能计算和复杂工程系统的计算机仿真研究,E-mail: jxlong@sjtu.edu.cn

引言

随着结构有限元模型规模的扩大、求解复杂性和求解精度的提高[1-3]，利用超级计算机的高效计算性能提高工程分析的效率成为一种行之有效的新途径。超级计算机又称并行计算机，它根据存储方式的不同大致可以分为共享存储并行计算机和分布式存储并行计算机两类[4]。共享存储并行计算机因内存访问瓶颈的问题导致系统可扩展性差，目前已逐步退出市场。分布式存储并行计算机则具有良好的可扩展性和优异的浮点运算性能，成为当前高性能计算领域超级计算机发展的主流。为充分发挥分布式存储并行计算机高效的计算性能，研究和开发相应的并行算法对于提高科学计算和工程分析的效率具有重要的现实意义。区域分解法因本身具备良好的并行计算和分布式存储特性成为分布式存储并行计算机上广泛采用的一种并行计算方法[5-7]。

在采用区域分解法并行求解的过程中，缩聚是其中非常关键的一个环节。缩聚算法不仅影响到系统内存需求，而且关系到系统计算量。由于结构有限元总体刚度矩阵具有大型、稀疏、对称带状分布的特点，传统存储方法一般采用变带宽格式只存储下三角部分带宽之内的元素从而避免了对带宽外大量零元素的存储。传统缩聚算法则通过对变带宽存储的刚度矩阵进行部分三角分解实现缩聚过程[8-9]。可以证明，三角分解刚度矩阵产生的非零元素总是落在带宽范围之内[8]。因此，缩聚过程中可直接利用原刚度矩阵的存储空间以节约内存资源。然而，对于三维复杂结构，随着有限元规模的扩大其刚度矩阵非零元素的分布将更加分散，从而导致系统带宽的急剧增加。此外，在带宽之内仍存在大量的零元素，对这些元素进行存储和运算会造成系统内存和 CPU 资源的严重浪费。

近十多年来，有限元领域逐渐采用更加高效的稀疏存储技术对刚度矩阵进行存储[10]。其中，应用最为广泛的一种是行压缩存储技术[11]。它只对刚度矩阵中的非零元素进行存储，从而避免了变带宽格式存储导致的一系列问题。基于行压缩存储格式的直接稀疏求解器通过填充 - 约化排序也确保了刚度矩阵在三角分解过程中产生的填充最少，有效减少了内存空间需求和计算量[12]。本文在吸收和利用稀疏存储技术和直接稀疏求解器优点的基础上，提出一种高效的缩聚并行计算方法。它将缩聚过程转换为一系列线性方程组的求解过程，并通过直接稀疏求解器进行求解。

1　区域分解法基本原理

区域分解法将结构有限元网格划分为若干个子区域，然后在每个子区域上单独运算，最后把各个子区域的结果收集汇总得到问题的完整解[13-16]。基于区域分解法进行并行计算的具体过程如下：

首先按照先内部节点后边界节点的编号原则同时独立形成各子区域的平衡方程

$$\begin{bmatrix} \boldsymbol{K}_{II} & \boldsymbol{K}_{IB} \\ \boldsymbol{K}_{BI} & \boldsymbol{K}_{BB} \end{bmatrix} \begin{bmatrix} x_I \\ x_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{P}_I \\ \boldsymbol{P}_B \end{bmatrix} \tag{1}$$

式中　x_I、x_B——内部节点和边界节点对应的位移

$\boldsymbol{P}_I$、$\boldsymbol{P}_B$——内部节点和边界节点对应的外部载荷向量

$\boldsymbol{K}_{**}$——刚度矩阵的分块矩阵

在式(1)的基础上各子区域通过缩聚同时独立消去内部自由度，得到只含边界自由度未知量的界面方程

$$\widetilde{\boldsymbol{K}} x_B = \widetilde{\boldsymbol{P}} \tag{2}$$

$$\widetilde{\boldsymbol{K}} = \boldsymbol{K}_{BB} - \boldsymbol{K}_{BI}\boldsymbol{K}_{II}^{-1}\boldsymbol{K}_{IB} \tag{3}$$

$$\widetilde{\boldsymbol{P}} = \boldsymbol{P}_B - \boldsymbol{K}_{BI}\boldsymbol{K}_{II}^{-1}\boldsymbol{P}_I \tag{4}$$

式中　$\widetilde{\boldsymbol{K}}$——缩聚刚度矩阵

$\widetilde{\boldsymbol{P}}$——缩聚载荷向量

系统总体界面方程的组集不仅需要大量的通信，而且需要消耗大量的内存资源。为减少通信量和节省内存空间，界面方程不宜采用直接法求解。而迭代法能够避免系统总体界面方程的组集，故本文采用迭代法中应用最为广泛的并行预条件共轭梯度算法[17]求解界面方程。

求得边界自由度后，各子区域内部自由度可根据

$$x_I = \boldsymbol{K}_{II}^{-1}(\boldsymbol{P}_I - \boldsymbol{K}_{IB}x_B) \tag{5}$$

同时独立回代求解。

以上是区域分解法并行求解的基本过程。其中，式(3)和式(4)表达了子区域缩聚的基本原理。实际操作时，为避免矩阵求逆消耗大量内存和计算资源，一般不采用矩阵求逆缩聚。采用传统变带宽格式缩聚的过程参照文献[8]。它采用变带宽格式存储刚度矩阵，并通过对带宽内的元素进行部分三角分解实现缩聚过程。

由于传统变带宽格式缩聚算法基于带宽内的元素进行操作，它需要对带宽内的所有元素进行存储和运算，故系统总带宽的大小也就直接决定了缩聚过程所需的内存空间和计算量。而在带宽之内仍存在大量的零元素，对这些元素进行存储和运算会造

成系统内存和 CPU 资源的严重浪费。特别是对于三维复杂结构,随着有限元规模的扩大其刚度矩阵非零元素的分布将更加分散,从而导致系统带宽的急剧增加。因此,采用传统变带宽格式缩聚时就会导致系统所需内存空间和计算量的大幅度增加。

2 高效缩聚并行计算方法

基于稀疏存储技术和直接稀疏求解器开发的缩聚算法,能够避免传统变带宽格式缩聚算法对带宽内大量零元素的存储和运算,从而有效节省内存空间和大幅度减少计算量。本文通过将缩聚过程转换为一系列线性方程组的求解过程,并借助直接稀疏求解器进行求解实现了一种高效的缩聚算法。

由式(3)可见,缩聚刚度矩阵的计算需要用到 $\boldsymbol{K}_{II}^{-1}\boldsymbol{K}_{IB}$ 的结果。可将 $\boldsymbol{K}_{IB}$ 的每一列元素依次提取出来并赋值给向量 $\boldsymbol{b}$,然后通过求解线性方程组 $\boldsymbol{K}_{II}\boldsymbol{t}_1=\boldsymbol{b}$ 依次得到 $\boldsymbol{K}_{II}^{-1}\boldsymbol{K}_{IB}$ 每一列的计算结果。而 $\boldsymbol{K}_{BI}\boldsymbol{K}_{II}^{-1}\boldsymbol{K}_{IB}$ 每一列的值则可通过计算 $\boldsymbol{t}_2=\boldsymbol{K}_{BI}\boldsymbol{t}_1$ 得到。最后,缩聚刚度矩阵每一列的结果则通过计算 $\boldsymbol{K}_{BB}$ 相应的列向量与 $\boldsymbol{t}_2$ 的差得到。由于缩聚刚度矩阵的列数与子区域内部自由度的个数 n 相等,因此上述操作共需进行 n 次。同理,缩聚载荷向量可采取同样的求解策略进行计算。因缩聚载荷向量只有一列,故相应的流程只需操作一次。采用直接稀疏求解器进行高效缩聚并行计算的伪代码如图 1 所示。图中,$\boldsymbol{K}_{IB}(:,i)$、$\widetilde{\boldsymbol{K}}(:,i)$ 和 $\boldsymbol{K}_{BB}(:,i)$ 分别表示相应矩阵的第 i 列元素构成的列向量;$\boldsymbol{t}_1$、$\boldsymbol{t}_2$ 为临时列向量。

1. // calculate the condensed stiffness matrix
2. do $i=1,n$
3. $\boldsymbol{b}=\boldsymbol{K}_{IB}(:,i)$
4. solve $\boldsymbol{K}_{II}\boldsymbol{t}_1=\boldsymbol{b}$ with a direct sparse solver
5. compute $\boldsymbol{t}_2=\boldsymbol{K}_{BI}\boldsymbol{t}_1$
6. compute $\widetilde{\boldsymbol{K}}(:,i)=\boldsymbol{K}_{BB}(:,i)-\boldsymbol{t}_2$
7. end do
8. // calculate the condensed load vector
9. $\boldsymbol{b}=\boldsymbol{P}_I$
10. solve $\boldsymbol{K}_{II}\boldsymbol{t}_1=\boldsymbol{b}$ with a direct sparse solver
11. compute $\boldsymbol{t}_2=\boldsymbol{K}_{BI}\boldsymbol{t}_1$
12. compute $\widetilde{\boldsymbol{P}}=\boldsymbol{P}_B-\boldsymbol{t}_2$
13. // solve interface equations
14. solve $\widetilde{\boldsymbol{K}}\boldsymbol{x}_B=\widetilde{\boldsymbol{P}}$ with the parallel PCGalgorithm
15. // calculate internal degrees of freedom
16. $\boldsymbol{b}=\boldsymbol{P}_I-\boldsymbol{K}_{IB}\boldsymbol{x}_B$
17. solve $\boldsymbol{K}_{II}\boldsymbol{x}_I=\boldsymbol{b}$ with a direct sparse solver

图 1 高效的缩聚并行计算方法

Fig. 1 Efficient condensation parallel computing method

由图 1 可见,在缩聚刚度矩阵和缩聚载荷向量的计算过程中需要多次求解线性方程组以及计算矩阵与向量的乘积。而结构有限元刚度矩阵采用稀疏存储格式只保存非零元素后,一方面有效节省了内存空间,另一方面可利用稀疏矩阵求解数学库来提高线性方程组求解和矩阵向量运算的效率。目前应用比较广泛的稀疏矩阵求解数学库有 SuperLU, MUMPS 以及 Intel MKL 数学库等。本文采用 Intel MKL 数学库来实现线性方程组的求解和矩阵向量的乘积运算。Intel MKL 数学库是面向科学和工程计算的高性能数学库[18],它提供了一套高度优化和线程安全的数学例程,其核心数学函数包括 BLAS, LAPACK, ScaLAPACK, 直接稀疏求解器、快速傅立叶变换、矢量数学及其他函数。以 Intel MKL 数学库为基础进行二次开发不但可以节省大量底层的编程工作,而且还可以借助其高度优化和线程安全的特点保证软件开发的质量和效率。

对于稀疏矩阵与向量的乘积,直接调用 Intel MKL 数学库里稀疏矩阵运算的函数即可实现。而稀疏线性方程组的求解则需利用数学库里的 PARDISO 求解器。PARDISO 是 Intel MKL 数学库中一款求解大规模对称和非对称线性方程组的高性能稳健直接稀疏求解器。它具有高效的内存管理机制,能够通过先进的填充-约化排序算法确保系统在求解过程中消耗的内存资源最少。为方便用户使用,PARDISO 将求解线性方程组的过程划分为 3 个阶段,并允许用户通过 phase 参数进行控制。各阶段 phase 参数的设置及实现的相关功能如下:

(1) phase = 11,对方程组的左端项进行填充-约化排序使系统在三角分解阶段产生的填充最少,并进行符号分解以确定下三角矩阵的稀疏结构。

(2) phase = 22,对方程组的左端项进行数值分解,具体计算经三角分解得到的下三角矩阵中各元素的数值。

(3) phase = 33,根据方程组的右端项进行回代求解,得到方程组的解。

由于在图 1 算法中所有要求解的线性方程组的左端项都相同,因此利用 PARDISO 求解时其前两个阶段的操作,即矩阵的填充 - 约化排序及符号分解和数值分解过程,只需进行一次。然后在具体求解某个线性方程组时,只要将 phase 的值设置为 33 并根据相应的右端项回代求解即可。

相对传统变带宽格式缩聚算法,本文提出的稀疏存储格式缩聚算法具有如下优点:首先可以避免对带宽内大量零元素的存储和运算,从而有效节省内存空间和大幅度减少计算量。其次,可以利用稀疏矩阵求解数学库来保证开发出的程序具有较低的

内存消耗和高效的计算性能。

3　算例验证

为验证本文并行算法的有效性，在上海超级计算中心的魔方超级计算机上进行了实例测试。魔方超级计算机的型号为 DAWNING－5000A，它属于典型的分布式存储大规模并行计算机。整个系统由多核节点通过 InfiniBand 网络连接而成。其具体的硬件配置如下：每个节点包括 4 颗 AMD Barcelona 1.9 GHz 四核处理器，64 GB 共享内存。节点机间采用 InfiniBand 光纤网络互联，理论带宽 20 Gbits/s。每个核拥有 512 KB 二级缓存，位于相同芯片上的 4 个核共享 2 MB 三级缓存。

魔方超级计算机上运行的操作系统为 Suse Linux Enterprise Server 10。本文并行计算程序采用 FORTRAN 语言编写，以 Intel MKL 数学库为基础实现相关求解算法，进程间通信根据 MPI 标准实现。并行算法评估的计算模型如图 2 所示。图 2 为某发动机曲轴的静力学分析有限元计算模型。

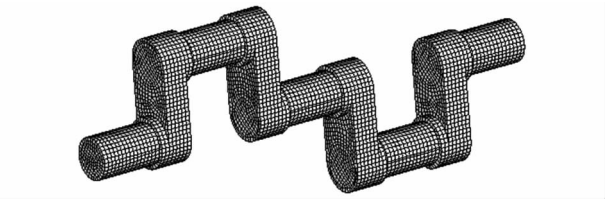

图 2　曲轴计算模型

Fig. 2　Computational model of crankshaft

采用六面体单元进行网格剖分后，该模型具有 147 312 单元，173 056 节点，519 168 自由度。并行计算时，将整体有限元模型分别剖分为 8 个子区域和 16 个子区域两种规格。然后针对每种规格，分别采用传统变带宽格式缩聚算法和本文稀疏存储格式缩聚算法进行并行计算。各子区域的单元、节点数以及采用不同方法进行缩聚并行计算的结果如表 1 和表 2 所示。变带宽格式和稀疏存储格式并行计算的总时间在 8 个子区域时分别为：1 086 s 和 828 s；在 16 个子区域时分别为 605 s 和 451 s。

表 1　8 个分区并行计算结果

Tab. 1　Results of parallel computing for eight partitions

子区域编号	单元数	内部节点数	边界节点数	变带宽格式缩聚		稀疏存储格式缩聚	
				最大内存需求/MB	缩聚时间/s	最大内存需求/MB	缩聚时间/s
1	18 414	21 459	334	510	1 052	214	772
2	18 415	21 619	165	376	627	213	398
3	18 413	21 455	339	481	960	215	792
4	18 414	21 453	339	482	950	215	796
5	18 414	21 617	166	378	600	213	393
6	18 414	21 455	335	507	1 041	214	781
7	18 414	21 457	338	482	914	215	791
8	18 414	21 457	338	500	1 050	215	790

表 2　16 个分区并行计算结果

Tab. 2　Results of parallel computing for sixteen partitions

子区域编号	单元数	内部节点数	边界节点数	变带宽格式缩聚		稀疏存储格式缩聚	
				最大内存需求/MB	缩聚时间/s	最大内存需求/MB	缩聚时间/s
1	9 208	10 655	332	246	553	115	415
2	9 206	10 650	336	241	521	115	420
3	9 207	10 813	163	187	324	114	218
4	9 208	10 656	328	240	537	116	391
5	9 206	10 646	339	241	551	115	422
6	9 207	10 650	339	242	550	116	404
7	9 207	10 651	338	246	576	115	405
8	9 207	10 651	338	247	577	116	423
9	9 207	10 651	338	248	575	115	442
10	9 207	10 651	338	241	544	115	404
11	9 207	10 650	339	249	581	116	422
12	9 207	10 650	339	241	522	115	406
13	9 207	10 654	332	248	568	116	416
14	9 207	10 649	336	253	596	115	403
15	9 207	10 813	163	190	345	114	206
16	9 207	10 655	328	249	570	115	409

由表 1 和表 2 可见,相对传统变带宽格式缩聚算法,本文稀疏存储格式缩聚算法能够大幅度节省内存空间和有效减少缩聚时间。在表 1 中子区域 1 采用变带宽格式缩聚时最大内存需求为 510 MB,缩聚时间为 1 052 s;而采用稀疏存储格式缩聚时最大内存需求仅为 214 MB,缩聚时间为 772 s。这就节省了 296 MB 内存空间,缩聚时间也减少了 280 s。这是由于采用稀疏存储格式缩聚可以避免变带宽格式缩聚对带宽内大量零元素的存储和运算,从而大幅度节省内存空间和有效减少计算量。

另外,由表 1 和表 2 也可见,各子区域有限元规模越大,稀疏存储格式缩聚对内存空间的节省和缩聚时间的减少效果就越明显。在表 2 中,各子区域有限元规模在 9 206 ~ 9 208 单元时,稀疏存储格式缩聚相对变带宽格式缩聚内存空间节省 130 MB 左右,缩聚时间减少 150 s 左右;而在表 1 中,各子区域有限元规模增大到 18 413 ~ 18 415 单元时,稀疏存储格式缩聚相对变带宽格式缩聚内存空间节省 280 MB左右,缩聚时间减少 260 s 左右。这是由于采用变带宽格式缩聚时随着有限元规模的扩大,系统带宽也急剧增加导致其所需内存空间和计算量大幅度增加。而稀疏存储格式缩聚则避免了对带宽内大量零元素的存储,并通过填充-约化排序有效减少了缩聚过程的内存空间需求和计算量。

为从计算效率的角度评价本文方法的有效性,将图 2 所示曲轴模型分别剖分为 8、16、24 和 32 个子区域,然后启动相应的核数分别采用不同算法进行并行求解,得到的计算效率统计如表 3 所示。

表 3 计算效率统计

Tab. 3 Statistics of computational efficiency

核数	变带宽格式求解总时间/s	稀疏存储格式求解总时间/s	计算效率提高率/%
8	1 086	828	23.76
16	605	451	25.45
24	338	269	20.41
32	229	187	18.34

由表 3 可见,在本案例中稀疏存储格式缩聚并行计算方法相对传统变带宽格式缩聚并行计算方法计算效率的提高 18.34% ~ 25.45%,从而验证了本文算法能够有效提高计算效率。

4 结束语

区域分解法是有限元结构分析中广泛采用的一种并行计算方法。缩聚是区域分解法中非常关键的一个环节。缩聚算法不仅影响到系统内存需求,而且关系到系统计算量。在吸收和利用稀疏存储技术和直接稀疏求解器优点的基础上,提出了一种基于稀疏存储格式的有限元结构分析高效缩聚并行计算方法。通过典型数值算例表明:相对传统变带宽格式缩聚并行计算方法,本文稀疏存储格式缩聚并行计算方法能够大幅度节省内存空间和有效提高计算效率;各子区域有限元规模越大,本文算法对内存空间的节省和缩聚时间的减少效果越明显。

参 考 文 献

1 Jacobsen D A, Senocak I. Multi-level parallelism for incompressible flow computations on GPU clusters[J]. Parallel Computing, 2013, 39(1):1 - 20.

2 王维军,王洋,刘瑞华,等. 离心泵空化流动数值计算[J]. 农业机械学报,2014,45(3):37 - 44.
Wang Weijun, Wang Yang, Liu Ruihua, et al. Numerical calculation of cavitation flow in a centrifugal pump[J]. Transactions of the Chinese Society for Agricultural Machinery,2014,45(3):37 - 44. (in Chinese)

3 Li Y Y, Jin X L, Li L J, et al. A parallel and integrated system for structural dynamic response analysis[J]. The International Journal of Advanced Manufacturing Technology, 2006, 30(1 - 2):40 - 44.

4 杨学军. 并行计算六十年[J]. 计算机工程与科学, 2012,34(8):1 - 10.
Yang Xuejun. Sixty years of parallel computing[J]. Computer Engineering & Science, 2012,34(8):1 - 10. (in Chinese)

5 王建炜,金先龙,曹源. 列车与结构动态耦合分析的并行计算方法[J]. 计算力学学报,2012,29(3):352 - 356.
Wang Jianwei, Jin Xianlong, Cao Yuan. Parallel computing for the train-structure dynamic coupling analysis[J]. Chinese Journal of Computational Mechanics, 2012,29(3):352 - 356. (in Chinese)

6 Vassilev D, Wang C, Yotov I. Domain decomposition for coupled Stokes and Darcy flows[J]. Computer Methods in Applied Mechanics and Engineering, 2014, 268: 264 - 283.

7 Wang Xicheng, Baggio P, Schrefler B A. A multi-level frontal algorithm for finite element analysis and its implementation on parallel computation[J]. Engineering Computations, 1999, 16(4): 405 - 427.

8 Han T Y, Abel J E. Substructure condensation using modified decomposition[J]. International Journal for Numerical Methods in Engineering, 1984, 20(11): 1959 - 1964.

9 Ondřej Medek, Jaroslav Kruis, Zdeněk Bittnar, et al. Static load balancing applied to Schur complement method[J]. Computers & Structures, 2007, 85(9): 489 - 498.

10 Montagne E, Ekambaram A. An optimal storage format for sparse matrices [J]. Information Processing Letters, 2004, 90(2):

87 – 92.
11 Rao A R M. MPI-based parallel finite element approaches for implicit nonlinear dynamic analysis employing sparse PCG solvers [J]. Advances in Engineering Software, 2005, 36(3):181 – 198.
12 Chen P, Zheng D, Sun S, et al. High performance sparse static solver in finite element analyses with loop-unrolling [J]. Advances in Engineering Software,2003, 34(4): 203 – 215.
13 Skogestad J O, Keilegavlen E, Nordbotten J M. Domain decomposition strategies for nonlinear flow problems in porous media[J]. Journal of Computational Physics, 2013, 234: 439 – 451.
14 Kurc O. Workload distribution framework for the parallel solution of large structural models on heterogeneous PC clusters[J]. Journal of Computing in Civil Engineering, 2010, 24(2):151 – 160.
15 Kraus J. Additive Schur complement approximation and application to multilevel preconditioning[J]. Siam Journal on Scientific Computing, 2012, 34(6):A2872 – A2895.
16 Kocak S, Akay H. Parallel Schur complement method for large-scale systems on distributed memory computers[J]. Applied Mathematical Modelling, 2001, 25(10):873 – 886.
17 Carter W T, Law K H. A parallel finite element methods and its prototype implementation on a hypercube[J]. Computer & Structures, 1989, 31(6): 921 – 934.
18 Intel Math Kernel Library—Fastest and most used math library for Intel and compatible processors [EB/OL]. [2014 – 05 – 03]. http://software. intel. com/en-us/intel-mkl/.

(上接第 364 页)

7 Rusinkiewicz S, Levoy M. Efficient variants of the ICP algorithm [C]// Proceedings of the Third International Conference on 3D Digital Imaging and Modeling, 2001: 145 – 152.
8 Zhu L, Barhak J, Srivatsan V, et al. Efficient registration for precision inspection of free-form surfaces[J]. The International Journal of Advanced Manufacturing Technology,2007,32(5 – 6): 505 – 515.
9 Maurer J R, Aboutanos C R, Dawant G B, et al. Registration of 3-D images using weighted geometrical features[J]. IEEE Transactions on Medical Imaging, 1996,15(6): 836 – 849.
10 Zinsser T, Schnidt H, Niermann J. A refined ICP algorithm for robust 3-D correspondences estimation[C]// Proceedings of International Conference on Image Processing, 2003:695 – 698.
11 Stewart C V, Tsai C L, Roysam B. The dual bootstrap iterative closest point algorithm with application to retinal image registration [J]. IEEE Transactions on Medical Imaging, 2003, 22(11): 1379 – 1394.
12 Chetverikov D, Stepanov D, Krsek P. Robust Euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm [J]. Image and Vision Computing, 2005, 23: 299 – 309.
13 程云勇, 张定华, 卜昆, 等. 涡轮叶片形状检测中的模型配准控制点集选取[J]. 机械工程学报, 2009, 45(11): 240 – 246.
Cheng Yunyong, Zhang Dinghua, Bu Kun, et al. Model registration control point set selection for turbine blade shape inspection [J]. Journal of Mechanical Engineering, 2009, 45(11): 240 – 246. (in Chinese)
14 Bing Jian, Baba C Vemuri. Robust point set registration using Gaussian mixture models[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(8): 1633 – 1646.
15 Tsin Y, Kanade T. A correlation-based approach to robust point set registration[C]// 8th European Conference on Computer Vision, 2004, 1: 558 – 569.
16 Per Bergstrom, Ove Edlund. Robust registration of point sets using iteratively reweighted least squares[J]. Computational Optimization and Applications, 2014,58(3):543 – 561.

(上接第 378 页)

17 苗虎, 周玉成, 盛振湘, 等. 连续平压机热压板升降系统控制算法[J]. 农业机械学报, 2014, 45(7): 333 – 339.
Miao Hu, Zhou Yucheng, Sheng Zhenxiang, et al. Design and application of a control algorithm for hydraulic lifting system of the hot platen in continuous flat press [J]. Transactions of the Chinese Society for Agricultural Machinery, 2014, 45(7): 333 – 339. (in Chinese)
18 张健, 肖兴明. 基于 LabVIEW 的螺旋给料机状态监测与变频调速反馈控制系统设计[J]. 矿山机械, 2013, 41(6): 103 – 106.
Zhang Jian, Xiao Xingming. Design of condition monitoring and frequency conversion speed feedback control system for screw feeder based on LabVIEW [J]. Mining & Processing Equipment, 2013, 41(6): 103 – 106. (in Chinese)
19 刘宝, 宗力, 张东兴. 锤片式粉碎机空载运行中锤片的受力及运动状态[J]. 农业工程学报, 2011, 27(7): 123 – 128.
Liu Bao, Zong Li, Zhang Dongxing. Force and motion states of hammer mill at unloaded running [J]. Transactions of the CSAE, 2011, 27(7): 123 – 128. (in Chinese)
20 陈君梅, 赵祚喜, 陈嘉琪, 等. 水田激光平地机非线性水平控制系统[J]. 农业机械学报, 2014, 45(7): 79 – 84.
Chen Junmei, Zhao Zuoxi, Chen Jiaqi, et al. Design of nonlinear leveling control system for paddy land leveler [J]. Transactions of the Chinese Society for Agricultural Machinery, 2014, 45(7): 79 – 84. (in Chinese)