文章编号：1000-4750(2014)05-0020-07

基于多重网格法和 GPU 并行计算的大规模壳结构快速计算方法

蔡　勇，李光耀，王　琥

(湖南大学汽车车身先进设计制造国家重点实验室，长沙 410082)

摘　要：该文采用将 EBE 计算策略、多重网格法以及 GPU 并行计算方法三者相结合的计算策略，设计了一种新颖的迭代求解方法，可以有效的提高大规模壳结构的有限元分析效率。该方法中，EBE 计算策略将总体运算分解到单元上进行，可以节约计算内存，提高单机上问题的求解规模，并且可以有效地提高隐式有限元算法的并行性；多重网格法通过在疏密不同的网格层上进行迭代，平滑不同频率的误差分量，可以加快迭代收敛速度；GPU 并行计算方法可以在较低硬件成本的前提下实现高效的并行计算。该文采用统一计算架构(Compute Unified Device Architecture, CUDA)进行程序的编制，并在采用 GTX460 显卡的个人计算机执行。数值计算结果表明该方法在保证计算精度的同时可以取得较高的计算加速比。

关键词：有限元；多重网格法；壳单元；GPU；EBE

中图分类号：TP391　　**文献标志码：**A　　doi: 10.6052/j.issn.1000-4750.2012.12.0942

A FAST CALCULATION METHOD FOR LARGE-SCALE SHELL STRUCTURE BASED ON MULTIGIRD METHOD AND GPU PARALLEL COMPUTING

CAI Yong , LI Guang-yao , WANG Hu

(State Key Laboratory of Advanced Design and Manufacturing for Vehicle Body, Hunan University, Changsha 410082, China)

Abstract: A novel iterative solution method for implicit finite element equations based on EBE scheme, multigrid method and GPU parallel computing method is designed to speed up the finite element analysis of a large-scale shell structure. In this method, EBE calculation strategies disassemble a global computation to a local element, which can reduce memory consumption and significantly increase the solution scale, more importantly it can improve the parallelism of implicit finite element calculation. Multigrid method can accelerate the convergence of iteration by using different mesh densities to eliminate the different frequency components of errors. And, GPU parallel computing is a novel parallel approach to reduce the time of computation with lower cost. The program of this method is compiled by CUDA (Compute Unified Device Architecture) and then implemented in a personal computer with a GTX460 graphics card, the calculation results show that the method can achieved a high computing speed-up ratio with high calculation accuracy.

Key words: finite element; multigrid method; shell element; GPU; EBE

随着工程技术领域的不断拓展，出现了各种复杂壳结构件。这些复杂的壳结构不仅具有庞大的自由度，并且含有复杂的本构关系、动态载荷和多样的边界条件等元素。传统的串行算法，如迭代算法，

收稿日期：2012-12-12；修改日期：2013-05-31

基金项目：国家重点基础研究发展计划(973 计划)项目(2010CB328005)；国家自然科学基金重点项目(61232014)

通讯作者：蔡　勇(1986－)，男，湖南长沙人，博士生，从事并行计算方法在汽车 CAE 中应用的研究(E-mail: bsforever@126.com).

作者简介：李光耀(1963－)，男，湖南永州人，教授，博士，从事汽车 CAE 研究(E-mail: gyli@hnu.edu.cn)；

王　琥(1975－)，男，湖南长沙人，副教授，博士，从事工程优化研究(E-mail: wanghuenying@hotmail.com).

在计算此类问题时，往往存在收敛速度慢，单次迭代计算时间过长以及计算规模限制等问题。近年来，对于大规模有限元计算，并行计算已经成为主流[1−2]。

多重网格法是求解偏微分方程数值解的有效方法之一。其采用粗细网格交替迭代的策略，计算工作量仅与网格节点数的一次方成正比，并且收敛速度相对恒定，与网格规模大小无关，从而特别适合应用于大型、超大型工程数值计算问题中[3]。

计算硬件方面，为满足现代三维图形处理的高要求，图形处理器(Graphics processing unit, GPU)被设计为一种高度并行化、多线程、多核的处理器，具有强大的并行计算能力。可编程 GPU 的出现，使 GPU 不仅用于图形处理，还可以用于通用的科学计算任务(General purpose computation on graphics processing unit, GPGPU)[4]。统一计算架构(Compute unified device architecture, CUDA)是由 NVIDIA 推出的基于 GPU 的通用并行计算架构，该架构可以方便、高效地实现通用科学问题基于 GPU 的并行计算。至今，GPU 通用计算已经在多种应用领域有效地提高了计算效率[5−6]。

有限元的 EBE 策略是由 Hughes[7]提出，其基本原理是将总体运算分解到单元上进行，可以省略传统有限元计算过程中的整体刚度矩阵的组装过程，计算过程中只保存单元刚度矩阵[8]。与普通有限元方法相比，EBE 方法更节约计算内存，计算量也相对要小，计算都是在单元一级进行，具有很好的并行效率，可以很好的适应于 GPU 这类细粒度并行计算架构，如 Kiss[9]实现了基于 EBE 策略的共轭梯度法在 GPU 上的计算。

该文基于以上的研究成果，将多重网格法与 EBE 策略融合，并针对 GPU 并行计算架构进行调整和优化，设计并实现了基于 GPU 并行计算架构的壳单元有限元快速求解。该方法中，EBE 策略节约了计算内存，显著提高了问题的求解规模，多重网格法提高了收敛效率，而 GPU 并行计算极大地提升了整体计算速度。

1 GPU 并行计算方法

作为计算机中图像处理核心，随着人们对图像处理要求不断提高，GPU 的计算性能越来越强大，其浮点处理能力已经达到了同时期 CPU 的 10 倍以上，而其外部存储器带宽则是 CPU 的 5 倍以上[10]。现代 GPU 已经发展成为了一种高度并行化、多核、多线程的处理器。这些新的发展给 GPU 在通用计算领域上的应用注入了强劲的动力，特别适合于计算密集性、可高度并行化的计算。

CUDA 的提出简化了通用计算映射到 GPU 上执行的过程，大大缩短了计算程序的开发难度和周期。与传统的并行计算结构如 SMP、MPP 相比，CUDA 通过线程组层次结构、共享存储器、屏蔽同步等三个层次上的抽象提供了细粒度的数据和线程的并行模型。CUDA 采用单指令多线程(SIMT)执行模型，支持透明的可伸缩性，程序员不需要深入了解 GPU 的结构，便可以在任何可用处理器上处理各个子问题。另外，GPU 通用计算平台的计算机硬件成本也比传统并行计算架构所采用的超级计算机等计算平台低许多。

基于 GPU 通用计算平台上的通用计算主要有 3 个步骤：

1) 数据准备，在 CPU 端和 GPU 端分配计算所需的内存空间，并将计算所需数据从 CPU 中拷贝到 GPU 中。

2) GPU 端并行计算。

3) 将计算结果从 GPU 拷贝到 CPU 中。

2 多重网格法

迭代法求解线性方程组时，误差分量可分为两类：光滑分量和高频分量。对于传统迭代方法，高频分量衰减很快，而光滑分量衰减很慢。多重网格法为了克服固定网格的缺点，先在较细网格上进行迭代，把高频分量衰减掉，然后在较粗网格上迭代，把次高频分量衰减掉，逐层变粗直到最粗一层网格，进行方程的精确求解，把各种频率分量衰减掉。再由粗网格开始，依次返回到各级细网格，最后在最细网格上获得所求方程的解。整个过程保证了所有的误差分量的收敛速度都不会降低，从而可以更快地得到精确的解。

多重网格法有多种循环格式，本文采用 V 循环进行分析。假定 u^h 代表差分方程的精确解，v^h 为差分方程迭代解，f^h 为方程组的右端项。一个简单的二层网格 V 循环可用 4 个步骤来描述：

1) 以 $v^{h(0)}$ 为初值，在细网格上用松弛迭代法对

$$L_h u^h = f^h \tag{1}$$

做 v_1 次迭代，得近似值 v^h 及残余：

$$r^h = f^h - L_h v^h \tag{2}$$

2) 在粗网格上以 $v^{2h(0)}=0$ 为初值，求解误差方程：

$$L_{2h}u^{2h}=f^{2h}=I_h^{2h}r^h \tag{3}$$

得 v^{2h}。

3) 进行粗网格修正

$$v^h \leftarrow v^h + I_{2h}^h v^{2h} \tag{4}$$

4) 以新的 v^h 为初值，在细网格上再迭代 v_2 步，然后回到 1)，开始下一个 V 循环，直到达到一定的收敛标准为止。

上述循环中，L_h 代表细网格上的差分算子，L_{2h} 代表粗网格上差分算子；I_h^{2h} 是把细网格上的残余限制到粗网格上的算子，称为限制算子；I_{2h}^h 是把粗网格上的结果插值到细网格上的算子，称之为插值算子。

3 基于 GPU 的几何多重网格法

3.1 有限元 EBE 策略的并行化计算

对于一个线性边界的有限元问题，其弱形式的解等价于求解如下的线性方程：

$$Au=b \tag{5}$$

假设 $\hat{A}_e$ 和 $\hat{b}_e$ 为单元 e 对整体系统的贡献，则式(5)可以改写为：

$$(\sum_{e=1}^{E}\hat{A}_e)u=(\sum_{e=1}^{E}\hat{b}_e) \tag{6}$$

式中，单元对整体系统的贡献 $\hat{A}_e$ 和 $\hat{b}_e$ 可以单独计算，具有较好的并行性。在 GPU 上执行 EBE 策略时，可以采用 CUDA 线程与单元一一对应的并行计算方法，一个 CUDA 线程独立负责从全局显存中读取与一个单元的相关数据并计算该单元的单元刚度矩阵以及执行单元相关其它的计算任务。本文程序通过建立与单元数目相同的 CUDA 线程，同时的、无序的进行所有单元计算。与传统的有限元计算方法相比，采用 EBE 策略进行有限元计算时，最大的不同在于刚度矩阵的计算，而其它相关的计算与传统计算方法完全一致，由于是纯粹的向量计算，同样具有较高的可并行性。

EBE 策略中，尽管以单元的形式存储刚度矩阵可以有效的降低计算空间的使用量，但是对于大规模的计算问题，仍将占用绝大部分的内存空间。由于 GPU 计算所需要的显存空间是极其有限且不易扩充的，为了进一步减少显存使用量，本文对超出显存限制的计算问题，采用不存储任何刚度矩阵和载荷向量的方法，即在每一次迭代步中重复计算单元刚度矩阵。该方法虽然增加了庞大的计算量，但对于 GPU 计算来说是可行的，因为并行计算能够限制该部分计算的在很短的时间内完成(以本文的经验对于百万级自由度规模可以在 0.1s 内完成)。表 1 所示为采用 EBE 策略计算 $(\sum_{e=1}^{E}\hat{A}_e)u$ 的主要流程。

表 1 EBE 策略的 GPU 实现

Table 1 The EBE strategies on GPU

操作流程	操作内容
A 建立单元数据空间	1) 建立与单元数相同的线程数。 2) 对每一个单元，由一个独立的 CUDA 线程负责读取或计单元刚度矩阵 $\hat{\boldsymbol{A}}_e$，形成右端载荷向量 $\hat{\boldsymbol{u}}_e$，并施加边界条件；$\hat{\boldsymbol{A}}_e$ 和 $\hat{\boldsymbol{u}}_e$ 均存储在显卡的共享显存中，以提高计算效率。
B 单元级计算	1) 由 CUDA 线程在对应的单元数据空间内计算 $\hat{\boldsymbol{x}}_e=\hat{\boldsymbol{A}}_e\hat{\boldsymbol{u}}_e$。
C 单元数据离散	1) 按单元的节点自由度的关系，将单元向量 $\hat{\boldsymbol{x}}_e$ 离散到全局显存中的整体向量 $\boldsymbol{x}$ 中。

3.2 多重网格法的并行化计算

3.2.1 松弛方案

本文采用 Jacobi 迭代法[11]作为松弛方案，并引入松弛因子 w 保证迭代过程的收敛性。迭代公式为：

$$\boldsymbol{x}_i^{k+1}=\boldsymbol{x}_i^k+w(\boldsymbol{b}_i-\sum_{j=1}^{n}\boldsymbol{a}_{ij}\boldsymbol{x}_j^k) \tag{7}$$

其中：$\boldsymbol{x}$ 为解向量；$\boldsymbol{b}$ 为载荷向量；$\boldsymbol{a}$ 为刚度矩阵；k 为迭代步；$i=1,2,3,\cdots,n$ 为自由度下标。该迭代计算由多个矩阵与向量、向量与向量计算组成，适用于 CUDA 所提供的 CUBLAS 计算库，可以快速的实现整个迭代过程的 GPU 化计算。如：函数 cublas<*t*>axpy( ) 可以用于在 GPU 上实现式 $y[j]=\alpha\times x[k]+y[j]$ 并行计算，其中 t 代表计算数据的浮点类型。

同时，该迭代方法也适合于 EBE 策略。实际执行时，需要采用着色法[12]避免并行计算过程中的竞写错误。着色法的基本思想是对有限元网格进行着色，确保同一种颜色的单元没有共享的自由度，从而保证同一颜色的单元可以在同一时刻并行计算，并将计算结果离散到每个自由度上。本文制定的基于 EBE 策略 Jacobi 迭代法 GPU 计算流程如表 2 所示。

表 2　基于 EBE 策略的 Jacobi 迭代法在 GPU 上的计算流程

Table 2　EBE based Jacobi iterative method on GPU

计算流程	流程内容
A 预处理	1) 在 GPU 上进行单元的并行着色，记录第一色块内的单元编号。 2) 计算单元刚度矩阵所需空间，如果显存空间允许，在 GPU 上并行计算单元刚度矩阵并存储到显卡的全局显存中。
B 迭代过程(不同色块串行执行，色块内采用单元与 CUDA 线程一一对应的并行计算方法)	1) 采用表 1 中计算流程，计算 $y_{24}^e = k_{24\times24}^e \times x_{24}^e$ 。 2) 根据约束边界条件，对 y_{24}^e 进行修改，有约束的自由度处，令 $y_{\mathrm{dof}}^e = x_{\mathrm{dof}}^e$,下标 dof 为单元内自由度编号。 3) 组装 y_{24}^e 到结果向量 $\boldsymbol{y}^k$ 中。 4) 由 cublasDaxpy( )计算 $\boldsymbol{y}^k = b^k - \boldsymbol{y}^k$ 。 5) 由 cublasDaxpy( )计算 $\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k + w\boldsymbol{y}^k$ 。 6) 误差评估：由 cublasDsmax( )计算向量($\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k$)中的最大值下标，再由 cublasGetVector( )将该最大值也是当前的最大误差复制到 CPU 中，并在 CPU 上进行评估。 7) 如果收敛条件满足，则计算结束；收敛条件不满足，则返回到迭代过程的步 1)。

在 V 循环过程中，前后迭代次数 v_1 和 v_2 对程序的计算效率和收敛都有影响。以本文的经验，v_1 和 v_2 的值可根据计算模型的复杂度进行一定的修改，我们发现对于畸变程度较高的计算模型，适量提高 v_1 和 v_2 的值可以加快收敛速度，一般地，本文设定 v_1 和 v_2 的值分别为 5 和 10。

3.2.2　插值算子和约束算子

本文采用几何多层网格法进行计算，插值算子和限制算子分别采用线性插值算子和完全加权限制算子[13]。图 1 所示为粗细层网格的几何关系图，图中大黑点代表粗层网格节点，细黑点代表细层网格节点。

限制操作时，粗网格上节点的残余值 r 由以下等式求得，以节点 a 为例：

$$r_{\mathrm{c}}^a = r_{\mathrm{f}}^a + 1/2(r_{\mathrm{f}}^b + r_{\mathrm{f}}^d) + 1/4 r_{\mathrm{f}}^e \qquad (8)$$

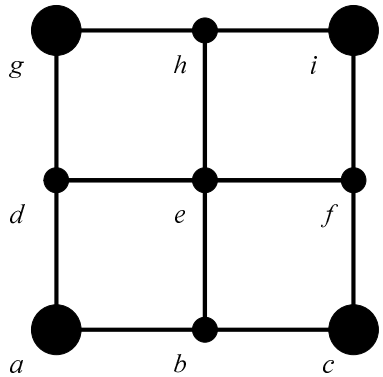


图 1　细网格与粗网格的关系图

Fig.1　Relation between finest and coarse mesh

插值操作时，细网格上节点的迭代值 v 由以下等式求得：

$$\begin{aligned} v_{\mathrm{f}}^a &= v_{\mathrm{f}}^a + v_{\mathrm{c}}^a , \\ v_{\mathrm{f}}^b &= v_{\mathrm{f}}^b + v_{\mathrm{c}}^b + 1/2(v_{\mathrm{c}}^a + v_{\mathrm{c}}^c) , \\ v_{\mathrm{f}}^e &= v_{\mathrm{f}}^e + v_{\mathrm{c}}^e + 1/4(v_{\mathrm{c}}^a + v_{\mathrm{c}}^c + v_{\mathrm{c}}^g + v_{\mathrm{c}}^i) 。 \end{aligned} \qquad (9)$$

式(8)和式(9)中，下标 c 代表粗网格层，下标 f 代表细网格层。

本文同样采用 CUDA 线程与对应层节点一一对应的计算方式，由一个线程分别查找当前节点不同权重下节点编号，并存储在显卡的全局显存上。由于每个节点对应的相关节点数并不相同，采用压缩存储的方式可以有效的节约显存空间，并保证计算效率。如图 2 所示，压缩存储的原理是在显存的全局显存上建立两个数组，数组 1 中按节点顺序依次存储该节点 1、1/2、1/3 权重的节点编号，数组 2 中按节点顺序存储该节点 1、1/2、1/3 权重节点在数组 1 中存储的起始位置。

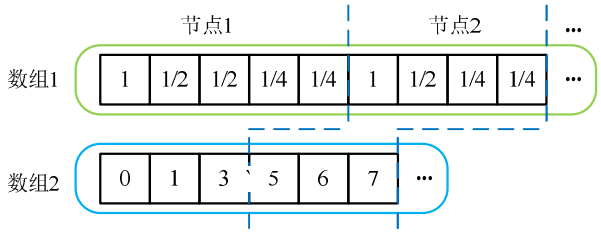


图 2　插值算子和限制算子的压缩存储

Fig.2　The compressed storage of interpolation and restriction operators

V 循环中执行限制和插值操作时，节点间相互独立，在上面两个数组的索引下，操作可以无序、并行的执行。对于限制操作，每一个粗网格上的节点将由一个 CUDA 线程依据索引数组按权重合并得到粗网格上节点的残余值。对于插值操作，每一个细网格上的节点由一个 CUDA 线程依据索引数组读取粗网格上节点的迭代解并按权重累加到细网格的节点上。两者均不存在细粒度并行中经常出现的竞写冲突，非常适合于 GPU 并行计算。

3.2.3　收敛标准

本文采用的收敛标准如下：

$$\mathrm{error} = \max | x_i^k - x_i^{k-1} | \qquad (10)$$

并行计算中，该收敛标准的计算难处在于寻找数组中的最大值。并行缩减算法可以并行寻找向量的极值，但是，为了编程难度，本文同样采用 CUBLAS 提供的库函数 cublasIdamax( )进行收敛误差的求解，程序代码如表 3 所示。

表 3 基于 CUBLAS 的收敛误差求解

Table 3 Convergence error computing based on CUBLAS

序号	代码
1	// Get the ERROR
2	int maxID = 0;
3	double max;
4	cublasDaxpy(FELevelGPU.sdof, −1.0, *x*+address, 1, *y*, 1);
5	maxID = cublasIdamax(FELevelGPU.sdof,*y*,1);
6	cublasGetVector(1, sizeof(double), *y*+maxID−1, 1, &max, 1);
7	error = fabs(max);

3.3 程序的整体架构

根据以上的分析，本文所编写程序的整体架构如表 4 所示。

表 4 程序的整体架构

Table 4 The overall structure of program

主流程	伪代码
1) 读取计算模型；	
2) 选择计算显卡设备，分配显存空间；	
3) 基于 GPU 的单元着色；	
4) 基于 GPU 的插值和限制算子搜寻；	

5) 求解	do{
	残余误差数组清零;
	// V 循环
	for(int c=0;c<numVCycles;c++){
	for(int L=0;L<numLeves-1;L++){
	for(int i=0;i<numPreSmoothSteps;i++){
	EBE 方法的 Jacobi 松弛; }
	计算当前层 L 的残余误差;
	单元当前层 L 的残余误差限制到层 l+1 上;}
	Jacobi 迭代法求解最粗层的解;
	for(int l=numLevels-1;l>=0;l--){
	将层 l-1 的迭代解插值到层 l 上，进行粗网格修正;
	for(int i=0;i<numPostSmoothSteps;i++){
	EBE 方法的 Jacobi 松弛;}}
	}
	}while(error < TOL)

6) 将计算结果从 GPU 复制到内存；	
7) 结果云图显示，写结果文件。	

4 数值验证

4.1 验证平台与方法

本文程序编写和测试所采用的硬件及软件环境如表 5 所示。其中，所采用的 GPU NVIDIA GTX460 拥有 348 个流处理器，1Gb 的显存，CUDA 计算能力为 2.1。

为了测试本文方法的优越性和稳定性，首先采用雅克比迭代法(Jacobi)、共轭梯度法(CG)和多重网格法(MG)三种计算方法，通过对比迭代次数和程序运行时间来确定多重网格法在收敛速度和计算时间上的优势。其中，共轭梯度法是目前一种公认的求解大型稀疏线性代数方程组最有用的方法，它适用于系数矩阵为对称正定的情况，并且理论上它能保证最多迭代方程组阶数次便可求得精确解。其次，对不同自由度数的模型分在 CPU 和 GPU 计算平台上采用对应的多重网格法版本进行计算，以验证 GPU 计算的优越性，并将 CPU 执行时间 t_{CPU} 和 GPU 执行时间 t_{GPU} 相除得到计算加速比 s_{p}：

$$s_{\mathrm{p}}=\frac{t_{\mathrm{CPU}}}{t_{\mathrm{GPU}}} \tag{11}$$

表 5 计算平台

Table 5 Computing platform

名称	型号
CPU	Intel Core i7 (2.8GHz)
GPU	NVIDIA GTX 460
操作系统	Windows 7
开发环境	Microsoft VC++ 2010 , CUDA 4.0

4.2 计算结果

已知一个二维平板如图 3 所示，长 200mm，宽 100mm，厚 1mm，一边固定，一边受垂直向下的载荷，大小 F=1.0N。平板材料的弹性模量为 E=2.10E5MPa，泊松比为 0.3，不计重力影响。计算该平板节点的位移和应力。多重网格法计算时有限元加密网格和稀疏网格如图 4 所示。

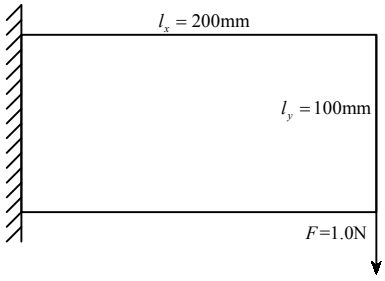
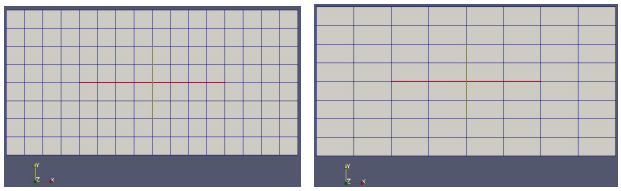


图 3 平板算例

Fig.3 Flat model



(a) 加密网格模型 (b) 稀疏网格模型

图 4 有限元网格模型

Fig.4 Mesh model of FEM

表 6 是针对不同的自由度数，给出三种计算方法的运行时间和迭代次数。可以看到，无论是在运行时间还是在迭代次数方面，多重网格法都比雅克比迭代法和共轭梯度法更具有优势，而纯粹使用雅

克比迭代法基本是不可行的。以 393216 个自由度的算例为例，多重网格法的计算迭代次数不到共轭梯度法的 1/70，程序运行计算不到共轭梯度法的 1/2。

表 6　不同规模下，程序运行时间及迭代次数

Table 6　Running time and iteration number for different scale

自由度数	迭代次数			程序运行时间/s		
	Jacobi	CG	MG	Jacobi	CG	MG
1536	3592	83	4	59.573	1.823	1.183
6144	10345	165	4	—	3.397	2.508
24576	—	325	6	—	17.385	10.88
98304	—	647	10	—	133.517	63.182
393216	—	1297	18	—	925.734	390.524

图 5 所示为不同规模自由度下多重网格法 CPU 和 GPU 版本程序的运行时间数据。可以得知，计算相同自由度数时，GPU 的计算时间要小于 CPU，并且单元数越多加速比越明显。同时，图 6 为 6144 个自由度时，CPU 和 GPU 的计算结果中 y 方向上位移云图。对比两图可知两者的计算结果基本一致，实际上，通过数值对比表明本文的 GPU 计算程序可以保证计算结果与 CPU 的误差小于 10^{-6}。这主要得益于本文程序编写过程中充分采用了“内建函数”来降低 GPU 计算时由于截断误差等引起的浮点计算误差。

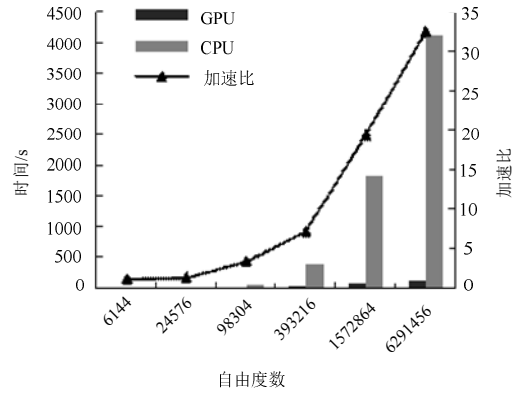
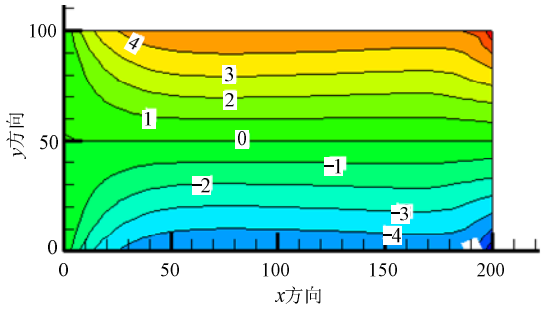
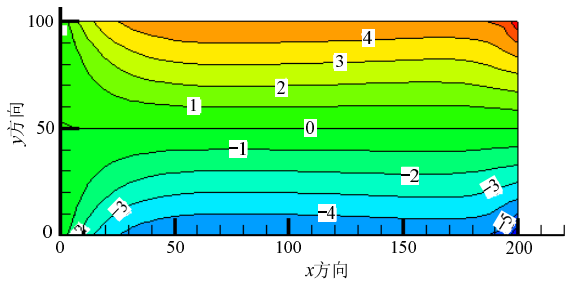


图 5　计算时间及加速比

Fig.5　Computing time and speed-up ration.



(a) CPU 计算结果



(b) GPU 计算结果

图 6　y 方向上的位移云图

Fig.6　Cloud picture of displacement in y direction

5　结论

该文针对大规模壳结构有限元分析中收敛效率低，计算速度慢等问题，提出了一种的基于多重网格法及 GPU 并行计算的壳单元有限元方程组高效计算方法。由数值算例可知，该并行计算方法在保证了计算结果正确性的同时具有较高的计算效率。同时，该方法实现了在个人计算机上利用 GPU 进行大规模的有限元计算。

参考文献：

[1] 雷霆, 姚振汉, 王海涛. 快速多极与常规边界元法机群并行计算的比较[J]. 工程力学, 2006, 23(11): 28－32.
Lei Ting, Yao Zhenhan, Wang Haitao. The comparison of parallel computation between fast multipole and conventional BEM on PC cluster [J]. Engineering Mechanics, 2006, 23(11): 28－32. (in Chinese)

[2] 付俊峰, 金生. 用 OpenMP 实现三维复杂渗流场的并行计算[J]. 工程力学, 2009, 26(12): 216－221.
Fu Junfeng, Jin Sheng. A parallel computation for 3D complex seepage flow using OpenMP [J]. Engineering Mechanics, 2009, 26(12): 216－221. (in Chinese)

[3] 谢学斌, 肖映雄, 潘长良, 舒适. 代数多重网格法在岩体力学有限元分析中的应用[J]. 工程力学, 2005, 22(5): 165－170.
Xie Xuebin, Xiao Yingxiong, Pan Changliang, Shu Shi. Application of algebraic multigrid method in finite element analysis of rock mechanics [J]. Engineering Mechanics, 2005, 22(5): 165－170. (in Chinese)

[4] 吴恩华, 柳有权. 基于图形处理器(GPU)的通用计算[J]. 计算机辅助设计与图形学学报, 2004, 16(5): 601－612.
Wu Enhua, Liu Youquan. General purpose computation on GPU [J]. Journal of Computer Aided Design & Computer Graphics, 2004, 16(5): 601－612. (in Chinese)

[5] 柳有权, 刘学慧, 吴恩华. 基于 GPU 带有复杂边界的三维实时流体模拟[J]. 软件学报, 2006, 17(3): 568－576.

Liu Youquan, Liu Xuehui, Wu Enhua. Real-time 3D fluid simulation of GPU with complex obstacles [J]. Journal of Software, 2006, 17(3): 568－576. (in Chinese)

[6] 蔡勇, 王琥, 李光耀, 崔向阳, 郑刚. 基于边光滑三角形壳元和统一计算架构的板料成形仿真并行计算方法[J]. 机械工程学报, 2012, 48(6): 32－38.
Cai Yong, Wang Hu, Li Guangyao, Cui Xiangyang, Zheng Gang. Parallel simulation of sheet metal forming based on EST element and compute unified device architecture [J]. Journal of Mechanical Engineering, 2012, 48(6): 32－38. (in Chinese)

[7] Thomas J R, Hughes, Itzhak Levit, James Winget. An element-by-element solution algorithm for problems of structural and solid mechanics [J]. Computer Methods in Applied Mechanics and Engineering, 1983, 36(2): 241－254.

[8] 刘耀儒, 周维垣, 杨强. 三维有限元并行 EBE 方法[J]. 工程力学, 2006, 23(3): 27－31.
Liu Yaoru, Zhou Weiyuan, Yang Qiang. Parallel 3-D finite element analysis based on EBE method [J]. Engineering Mechanics, 2006, 23(3): 27 － 31. (in Chinese)

[9] Kiss I, Szabolcs Gyimothy, Zsolt Badics, Jozsef Pavo. Parallel Realization of the Element-by-Element FEM Technique by CUDA [J]. IEEE Transactions on Magnetics, 2011, 48(2): 507－510.

[10] 周季夫, 钟诚文, 尹世群, 解建飞, 张勇. 基于 GPGPU 的 Lattice-Boltzmann 数值模拟算法[J]. 计算机辅助设计与图形学学报, 2008, 20(7): 912－918.
Zhou Jifu, Zhong Chengwen, Yin Shiqun, Xie Jianfei, Zhang Yong. Numerical simulation algorithm of lattice-boltzmann on GPGPU [J]. Journal of Computer Aided Design & Computer Graphics, 2008, 20(7): 912－918. (in Chinese)

[11] 王勖成, 邵敏. 有限单元法基本原理和数值方法[M]. 北京: 清华大学出版社, 1997: 216－222.
Wang Xucheng, Shao Min. Finite element method [M]. Beijing: Tsinghua University Press, 1997: 216－222. (in Chinese)

[12] Cris Cecka, Adrian J, Lew E, Darve. Assembly of finite element methods on graphics processors [J]. International Journal for Numerical Methods in Engineering, 2011, 85(5): 640－669.

[13] 刘超群. 多重网格法及其在计算流体力学中的应用[M]. 北京: 清华大学出版社, 1995: 13－20.
Liu Chaoqun. Multigrid method and application in CFD [M]. Beijing: Tsinghua University Press, 1995: 13－20. (in Chinese)

(上接第 14 页)

[46] 谭柱华, 盖秉政, 庞宝君, 等. 影响 Hopkinson 压杆实验结果因素的数值模拟分析[J]. 哈尔滨工业大学学报, 2007, 39(3): 363－366.
Tan Zhuhua, Gai Binzheng, Pang Baojun, et al. The investigation of factors effect on SHPB experiment results by using numerical simulation method [J]. Journal of Harbin Institute of Technology, 2007, 39(3): 363－366. (in Chinese)

[47] Holmquist T J, Johnson G R. A computational constitutive model for concrete subjected to large strains, high strain rates, and high pressures [C]. 14th International Symposium on Ballistics Quebec, 1993: 591－600.

[48] Malvar L J, Crawford J E, Wesevich J, Simons D. A plasticity concrete material model for DYNA3D [J]. International Journal of Impact Engineering, 1997, 19(9): 847－873.

[49] ABAQUS Incorporated. ABAQUS analysis user's manual [M]. Michigan, USA: ABAQUS Incorporated, 2005, 20.3.1.

[50] Riedel W, Thoma K, Hiermaier S, et al. Penetration of reinforced concrete by BETA-B-500 numerical analysis using a new macroscopic concrete model for hydrocodes [C]. Proceedings of the 9th International Symposium on the Effects of Munitions with Structures, 1999.

[51] 方秦, 还毅, 张亚栋, 等. ABAQUS 混凝土损伤塑性模型的静力性能分析[J]. 解放军理工大学学报(自然科学版), 2007, 8(3): 254－260.
Fang Qin, Huan Yi, Zhang Yadong, et al. Investigation into static properties of damaged plasticity model for concrete in ABAQUS [J]. Journal of PLA University of Science and Technology, 2007, 8(3): 254－260. (in Chinese)

[52] GB 50010-2002, 混凝土结构设计规范[S]. 北京: 中国建筑工业出版社, 2002.
GB 50010-2002, Code for design of concrete structures [S]. Beijing: China Architecture Industry Press, 2002. (in Chinese)