基于CUDA的二维泊松方程快速直接求解

岳小宁[1,2] 肖炳甲[2] 罗正平[2]
(中国科学技术大学核科学技术学院 合肥 230027)[1]
(中国科学院等离子体物理研究所计算机应用研究室 合肥 230031)[2]

摘 要 二维泊松方程离散化之后可以转化为一个具有特殊格式的块三对角方程的求解问题,通过对这一结构化线性方程组的研究,提出了一个适用于统一计算架构(CUDA)的泊松方程并行算法。该算法通过离散正弦变化,可以将计算任务划分为若干相互独立的部分进行求解,各部分求解完成后再通过一次离散正弦变换即可获得最终解,整个求解过程只需要两次全局通信。结合GPU的硬件特征进行优化之后,该算法相比CPU上的串行算法可以获得10倍以上的加速比。

关键词 泊松方程,统一计算架构,并行计算,块三对角方程
中图法分类号 TP301.6 **文献标识码** A

Fast 2-dimension Poisson Direct Solver Based on CUDA

YUE Xiao-ning[1,2] XIAO Bing-jia[2] LUO Zheng-ping[2]
(School of Nuclear Science and Technology, University of Science and Technology of China, Hefei 230027, China)[1]
(Department of Applied Computer Science, Institute of Plasma Physics, Chinese Academy of Sciences, Hefei 230031, China)[2]

Abstract The finite difference approximation of two-dimension poisson equation would create a block-tridiagonal equation system. An algorithm which is suitable with the Compute Unified Device Architecture (CUDA) was proposed. Through a discrete sine transform, the computation task could be divided into several completely independent parts, after solving these parts in parallel, the final result could be obtained through another discrete sine transform. Only two global synchronizing is needed during the computation. After carefully optimized, an acceleration rate of more than 10 times is obtained.

Keywords Poisson equation, Compute unified device architecture, Parallel computing, Block-tridiagonal equation

1 概述

许多物理或者工程问题,如静电学中的电磁场求解,以及等离子体物理中的磁流体平衡求解等,最终都可以转化为对泊松方程的求解。二维泊松方程的基本形式如式(1)所示:

$$\frac{\partial^2 u(x,y)}{\partial x^2}+\frac{\partial^2 u(x,y)}{\partial y^2}=f(x,y) \tag{1}$$

方程的右侧称为源项。绝大多数的二维泊松方程是没有解析解的,因此需要通过离散的方式来进行数值求解。利用有限差分法进行离散化之后,泊松方程的求解可以转化为一个稀疏矩阵的求解问题,这一矩阵可以应用迭代法,如SOR方法,或者直接法,如Cholesky分解方法(离散后的稀疏矩阵为正定对称矩阵)等方法求解[1]。本文介绍的并行算法是基于一种特殊的线性方程直接解法,这一算法的串行时间复杂度为O($N\log N$),其中N为离散后的格点总数。

某些应用需要在很短时间以内完成泊松方程的求解,例如聚变实验装置托卡马克的位型控制,往往需要在1ms以内提供一次控制命令,而每次的控制量都需要通过求解Grad-Shafranov方程(类二维泊松方程)来获得[2]。为了保证较高的控制精度,理想的离散化网格规模为65×65或更高[3]。然而以65×65的离散规模为例,在现代计算机上仅G-S方程的串行求解就需要1ms以上的时间。因此有必要探寻求解泊松方程的并行算法。

基于传统架构的泊松方程并行化研究已有很多[4-6],这些研究大多基于迭代法以及其他的直接分解算法,取得的加速比依然较小(<2)。本文介绍了一种基于CUDA架构和离散正弦傅里叶变化方法(DST)的泊松方程并行求解方法。

2 CUDA架构

图形处理器(GPU)最初是为了提高计算机图形图像处理能力而设计的。基于这一设计目标,GPU的硬件结构非常适合处理高并发的计算任务,GPU往往比同时代的CPU拥有更高的浮点运算能力。

统一计算架构(CUDA)是英伟达提出的一种并行计算架构,通过CUDA可以利用熟悉的高级语言(如C语言)编写在

到稿日期:2012-12-05 返修日期:2013-03-11 本文受国家科技部"973"项目ITER计划专项(国内配套研究)(2012GB105000),国家自然科学基金(10835009)资助。

岳小宁(1988—),男,硕士生,主要研究方向为并行计算,E-mail:yxn@mail.ustc.edu.cn;肖炳甲(1966—),男,博士,研究员,主要研究方向为计算机应用、等离子体控制;罗正平 男,博士,助理研究员,主要研究方向为等离子体控制。

GPU 上运行的程序。GPU 上运行的程序与传统架构上的程序相比会拥有更多的限制。为了充分发挥 GPU 的运算能力，必须注意一定的优化原则。其中最重要的几点包括：

1)尽量增强算法的并行性，减少不必要的通信；

2)尽量减少并行对全局存储器(Global Memory)的读写操作；

3)尽量减少内存与显存之间的数据传送。

除此之外，还有一些次要的优化原则也必须注意，具体的内容在文献[7]中有详细描述。

3 算法描述

3.1 算法原理描述

该算法的基本思想最早在文献[8,9]中提出，在 FFT 算法出现后，该算法的应用价值进一步提高。其基本思想为，在一个矩形区域内对式(1)进行离散化求解，设 x 方向和 y 方向的格点数目分别为 $m+2$ 和 $n+2$，则 x 方向和 y 方向的步长可以表达为：

$$\Delta x=\frac{x_{\max}-x_{\min}}{m+1}$$
$$\Delta y=\frac{y_{\max}-y_{\min}}{n+1} \tag{2}$$

对式(1)进行五点差分，如式(3)所示：

$$f(x_i,y_j)=\frac{1}{\Delta x^2}(u_{i-1,j}-2u_{i,j}+u_{i+1,j})+\frac{1}{\Delta y^2}(u_{i,j-1}-2u_{i,j}+u_{i,j+1}) \tag{3}$$

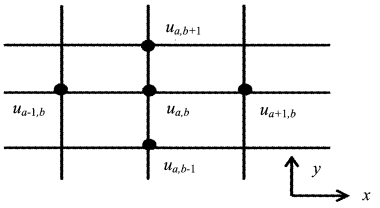

图 1 五点差分示意图

采用狄拉克边界条件，假设所有边界点处的值均已知。设：

$$c=\frac{\Delta y^2}{\Delta x^2}$$

$$\boldsymbol{u}_j=(u_{1,j}\quad u_{2,j}\quad\cdots\quad u_{m,j})^{\mathrm{T}}$$

则差分形成的线性方程组可以表示为：

$$\begin{pmatrix}\boldsymbol{A}&\boldsymbol{I}&&\\\boldsymbol{I}&\boldsymbol{A}&\ddots&\\&\ddots&\ddots&\boldsymbol{I}\\&&\boldsymbol{I}&\boldsymbol{A}\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_1\\\boldsymbol{u}_2\\\vdots\\\boldsymbol{u}_n\end{pmatrix}=\begin{pmatrix}\boldsymbol{g}_1\\\boldsymbol{g}_2\\\vdots\\\boldsymbol{g}_n\end{pmatrix} \tag{4}$$

其中，$\boldsymbol{I}$ 是 M 阶单位矩阵，

$$\boldsymbol{A}=\begin{pmatrix}-2(c+1)&c&&\\c&-2(c+1)&\ddots&\\&\ddots&\ddots&c\\&&c&-2(c+1)\end{pmatrix}$$

$$\boldsymbol{g}_j=(g_{1,j}\quad g_{2,j}\quad\cdots\quad g_{m,j})$$

$g_{i,j}$ 是源项 $f(x_i,y_j)$ 与边界值的线性组合，在边界值为零的情况下 $g_{i,j}=f(x_i,y_j)$。可以看出式(4)的系数矩阵是一个特殊的块三对角矩阵，矩阵 $\boldsymbol{A}$ 的特征向量和对应特征值分别为 $q_{ij}=\sqrt{\frac{2}{m+1}}\sin\frac{ij\pi}{m+1}$ 和 $\lambda_i=c(2-2\cos\frac{i\pi}{m+1})+2$，因此对矩阵 $\boldsymbol{A}$ 作特征值分解，可以得到：

$$\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{\mathrm{T}}$$

其中：

$$\boldsymbol{Q}=\sqrt{\frac{2}{m+1}}\begin{bmatrix}\sin\frac{\pi}{m+1}&\sin\frac{2\pi}{m+1}&\cdots&\sin\frac{m\pi}{m+1}\\\sin\frac{2\pi}{m+1}&\sin\frac{4\pi}{m+1}&\cdots&\sin\frac{2m\pi}{m+1}\\\vdots&\vdots&\vdots&\vdots\\\sin\frac{m\pi}{m+1}&\sin\frac{2m\pi}{m+1}&\cdots&\sin\frac{m^2\pi}{m+1}\end{bmatrix}$$

$\boldsymbol{\Lambda}$ 是由 λ_i 构成的对角矩阵，于是式(4)可以表示为：

$$\boldsymbol{u}_{j-1}+\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}_j+\boldsymbol{u}_{j+1}=\boldsymbol{g}_j,j=2,\cdots,n-1 \tag{5}$$

在式(5)的左右同乘以 $\boldsymbol{Q}^{\mathrm{T}}$，因为 $\boldsymbol{Q}$ 为正交矩阵，因此可以得到：

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}_{j-1}+\Lambda\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}_j+\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}_{j+1}=\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{g}_j,j=2,\cdots,n-1$$

设 $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}_j=\hat{\boldsymbol{u}}_j$，$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{g}_j=\hat{\boldsymbol{g}}_j$，则上式可化为：

$$\hat{\boldsymbol{u}}_{j-1}+\boldsymbol{\Lambda}_j\hat{\boldsymbol{u}}_j+\hat{\boldsymbol{u}}_{j+1}=\hat{\boldsymbol{g}}_j,j=2,\cdots,n-1 \tag{6}$$

对于 $j=1$ 和 n 的情况，只要相应地将式(4)－式(6)中的第一项或最后一项去掉即可，其他完全相同。

将 $\hat{\boldsymbol{u}}_j$ 中的元素进行重组，取每个 $\hat{\boldsymbol{u}}_j$ 中的第 i 个元素，重组得到：

$$\boldsymbol{u}_i'=(u_{i,1}'\quad u_{i,2}'\cdots u_{i,n}')^{\mathrm{T}},i=1,\cdots,m$$

相应地，对 $\hat{\boldsymbol{g}}_j$ 重组得到 $\tilde{\boldsymbol{g}}_i$，于是可以得到新的线性系统：

$$\begin{pmatrix}\boldsymbol{T}_1&&&\\&\boldsymbol{T}_2&\ddots&\\&\ddots&\ddots&\\&&&\boldsymbol{T}_m\end{pmatrix}\begin{pmatrix}\boldsymbol{u}_1'\\\boldsymbol{u}_2'\\\vdots\\\boldsymbol{u}_m'\end{pmatrix}=\begin{pmatrix}\boldsymbol{g}_1'\\\boldsymbol{g}_2'\\\vdots\\\boldsymbol{g}_m'\end{pmatrix} \tag{7}$$

其中，$\boldsymbol{T}_i=\begin{pmatrix}\lambda_i&1&&\\1&\lambda_i&\ddots&\\&\ddots&\ddots&1\\&&1&\lambda_i\end{pmatrix}$，$i=1,\cdots,m$。

从式(7)可以看出，整个线性系统被划分为了 m 个完全独立的子线性方程组。

求出 $\boldsymbol{u}_i'$ 之后，再进行一次重组，得到 $\hat{\boldsymbol{u}}_j$，因为 $\hat{\boldsymbol{u}}_j=\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{u}_j$，因此通过 $\boldsymbol{u}_j=\boldsymbol{Q}\hat{\boldsymbol{u}}_j$ 便可以得到最终的计算结果。此外，观察矩阵 $\boldsymbol{Q}$ 可以发现，该矩阵是属于一类离散正弦变换(DST)矩阵，因此 $\boldsymbol{g}_j$ 和 $\hat{\boldsymbol{g}}_j$ 以及 $\hat{\boldsymbol{u}}_i$ 和 $\boldsymbol{u}_i$ 之间的相互转化是可以通过快速傅里叶变换(FFT)来进行的。综上所述，整个算法的基本流程可以概括为：

```
输入  源项 G(i)
      for i=1 to n do
          G*(i)=DST(G(i))// 进行离散正弦变换
      end do
      exchange(g(i,j),g(j,i))//进行数据重组
      for i=1 to m do
          solve T(i)×U*(i)=G'(i)//求解三对角方程
      end do
      exchange(u(i,j),u(j,i))//进行数据重组
      for i=1 to n do
          u(i)=DST(U*(i))//进行离散正弦变换
      end do
输出  计算结果 U(i)
```

3.2 基于 CUDA 的算法并行化

在 CUDA 架构下，GPU 上运行的程序都会由以线程网格(Grid)组织的线程执行，每个线程网格由若干个线程块(block)组成，每个线程块会包含若干个线程，而同一线程块中的线程之间可以通过共享存储器(shared memory)和 barrier进行较为快速的通信，而不同的线程块之间则只能通过速度很慢的全局存储器通信。因此基于 CUDA 的这一特点，算法应该既可以在线程块之间进行不需要通信的粗粒度并行，又可以在线程块中进行局部通信的细粒度并行。

3.1 节中描述的直接求解泊松方程算法，本身已经具有很高的并行性。对 $\boldsymbol{g}_j$ 的 DST 变换可以在不同的 block 中同时对 $j=1,\cdots,n$ 进行完全并发的计算，而在线程块中，又可以利用多个线程间的细粒度并行，以加速每个 DST 的执行过程。对于 DST 的计算，可以使用 CUDA 自带的 CUFFT 库来进行。简略代码如下：

```
cufftPlan1d(&Plan,2*(m+1),CUFFT_D2Z,n);
cufftExecD2Z(Plan,source_data,aft_result);
```

m 个独立线性方程组也可以在 m 个线程块中同时并发求解。同时，每个线程块在解三对角方程时，可以利用文献[10]中的数据并行思想来进行并行加速。

整个算法中并行度较低的部分是两次重组过程，可以将 $g_{i,j}$ 以行优先矩阵的形式存储，如式(8)所示，则依照 3.1 节中的规则进行数据重组的过程就是对这一二维矩阵进行转置的过程。CUDA 架构下影响二维矩阵转置性能的主要是全局存储器的访问次数。为了减少全局存储器访问次数，可以将矩阵 $\boldsymbol{G}$ 划分为若干个块，每个块内利用共享存储器进行细粒度并行转置，同时利用线程块进行粗粒度的块转置，这样就实现了对全局存储器的合并访问，充分发挥了 GPU 的高带宽优势。

$$\boldsymbol{G}=\begin{pmatrix} g_{1,1} & g_{2,1} & \cdots & g_{m,1} \\ g_{1,2} & g_{2,2} & \cdots & g_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ g_{1,n} & g_{2,n} & \cdots & g_{m,n} \end{pmatrix} \tag{8}$$

简略代码如下：

```
__shared__ double cache[Blocksize][Blocksize+1];
cache[threadIdx.y][threadIdx.x]=G[index_in];
GT[index_out]=cache[threadIdx.x][threadIdx.y];
```

4 实验结果与分析

实验平台为一台华硕工作站，CPU 为 Intel Xeon E3-1230，GPU 为 NVIDIA Tesla c2050，操作系统为 Ubuntu10.10，CUDA Toolkit 版本为 4.0。

DST 算法求解泊松方程过程中由于会用到 FFT，因此考虑到 FFT 的效率问题，在进行有限差分的网格划分时令 m 和 n 为 2^k-1，即网格规模为 2^k+1，这样可以保证在执行 zero-padding 之后待变换的序列是 2 的指数倍，从而获得最佳的 FFT 效率。图 1 是在不同网格规模下在 CPU 和 GPU 上执行泊松方程 DST 解法所消耗的时间对比。除此之外，还加入了离散求解泊松方程常用的 Cholesky 分解算法(调用 CHOLMOD 库[11])作为对比。其中的加速比是指在 CPU 上串行与 GPU 上并行执行泊松方程 DST 解法的时间对比。可以看出，在网格规模较小时，GPU 上的并行运行的 DST 泊松解法与 CPU 上串行运行的相同算法相比，快 6 倍左右，而网格密度达到一定程度(>256×256)之后，加速比可以达到 10 以上，该算法与 Cholesky 分解算法相比，具有明显的速度优势。

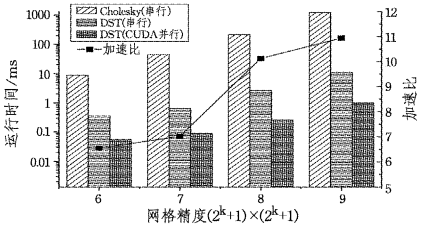

图 1 并行与串行算法的运行时间对比

图 2 列出了算法中主要耗时模块在不同网格数量运行时间占总运行时间的比例，表 1 列出了这些模块相对串行版本的加速比，可以看出在网格数量较少时，两次 DST 变换的时间消耗占主要部分，求解独立方程的加速比达到了 17 以上。然而随着网格密度的增加，求解独立方程的加速比迅速下降，而 DST 变换和数据重组的加速比则快速上升，相应的求解独立方程的运算时间开始成为主要部分。导致这一转变的主要原因为：随着网格数量的增加，在求解独立方程时的累加操作需要大量的共享存储器，因此在规模增大之后，每个流处理器上可以同时运行的 Block 数量开始急剧减少，从而导致 GPU 的利用率下降。而 DST 计算则随着 CuFFT 库的效率提高，其加速比迅速增加，数据重组则因 GPU 的高带宽而显现优势。

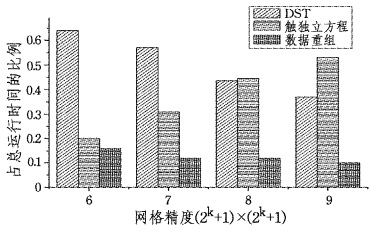

图 2 算法中主要模块运行时间占总时间的比例

表 1 不同模块加速比对比

模块	DST	解独立方程	数据重组
65×65	4.05	17.82	11.95
129×129	4.24	12.54	15.85
257×257	8.76	11.39	22.36
513×513	11.71	8.95	26.06
1025×1025	13.51	8.07	28.19

直接求解线性方程组时，截断误差对最终计算结果精度影响很大，因此该算法在实现时均采用了 double 类型。图 3 显示了各个算法计算结果的相对误差。计算公式为：

$$error=\max\left(\frac{(\boldsymbol{Ku})_i-\boldsymbol{g}_i}{\boldsymbol{g}_i},i=1,2\cdots\dim(\boldsymbol{K})\right) \tag{9}$$

式中，$\boldsymbol{K}$ 是式(4)中的系数矩阵，$\boldsymbol{u}$ 为计算结果，可以看出式(9)的意义是计算结果与精确结果的最大相对误差。

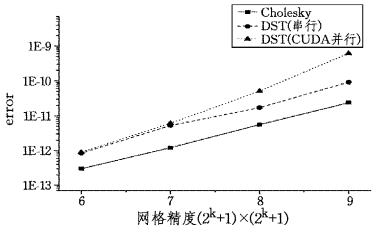

图 3 并行与串行算法的计算精度比较

(下转第 38 页)

明，该优化算法能有效地减少可逆电路的门数和控制位数，取得了较理想的综合结果。今后将尝试将其用于输入线数更多的可逆电路优化，同时也将考虑加入 Fredkin 等新的门来进一步降低可逆电路的代价。

参 考 文 献

[1] Nielsen M, Chuang I. Quantum Computation and Quantum Information[M]. Cambridge Univ. Press, 2000

[2] De Vos A, Desoete B, Janiak F, et al. Control Gates as Building Blocks for Reversible Computers[A]// Proceedings of 11th International Workshop on Power and Timing Modeling, Optimization and Simulation[C]. Yverdon, Switzerland, Sep. 2001: 9201- 9210

[3] Merkle R C. Two types of mechanical reversible logoc[J]. Nanotechnology, 1993(4): 114-131

[4] Wille R, Grobe D. Fast exact Toffoli network synthesis of reversible logic[A]// Proceedings of International Conference on Computer-Aided Design[C]. San Jose, USA, Nov. 2007: 60-64

[5] 倪丽慧，管致锦，聂志浪. 基于可逆函数复杂性的正反控制门可逆网络综合[J]. 计算机科学，2010，37(11)：117-121

[6] Miller D M, Maslov D, Dueck G W. A transformation based algorithm for reversible logic synthesis[A]// Proceedings of DAC[C]. Anahem, California, USA, 2003: 318-323

[7] Wan Si-shuang, Chen Han-wu, Cao Ru-jin. A Novel Transformation-Based Algorithm for Reversible Logic Synthesis[A]// Proceedings of the 4th International Symposium on Intelligence Computation and Applications (ISICA)[C]. Wuhan, PRC, Oct. 2009: 70-81

[8] Zheng Y, Huang C. A novel Toffoli network synthesis algorithm for reversible logic[A]// Proceedings of the 2009 Asia and South Pacific Design Automation Conference[C]. Los Alamitos: IEEE Computer Society Press, 2009: 739-744

[9] Arabzadeh M, Saeedi M, Zamani M S. Rule-based Optimization of Reversible Circuits[A]// Proceedings of ASP-DAC' 2010[C]. Taipei, Taiwan, Jan. 2010: 849-854

[10] Maslov D, Dueck G W, Miller D M. Toffoli network synthesis with templates[J]. Computer Aided Design of Integrated Circuits and Systems, 2005, 24(6): 807-817

[11] Maslov D, Dueck G, Miller D. Techniques for the synthesis of reversible Toffoli networks[J]. ACM Transactions on Design Automation of Electronic Systems, 2007, 12(4): 42

[12] 陈汉武，李志强，李文骞. 量子可逆逻辑综合的关键技术及其算法的研究[J]. 软件学报，2009，12：570-583

[13] 管致锦，秦小麟，施佺，等. 基于正反控制模型的可逆逻辑综合[J]. 计算机学报，2008，31(5)：835-844

[14] Maslov D, Dueck G W, Miller D M. Simplification of Toffoli networks via templates[A]// Proceedings of Integrated Circuits and Systems Design[C]. San Paulo: IEEE Computer Society, 2003: 53-58

(上接第 23 页)

从图 3 可以看出，随着网格密度指数的增加，3 种解法产生的误差都近似指数增长。基于 DST 的泊松方程串行直接解法误差要比基于 Cholesky 方法的串行解法大一个量级左右，这主要是由于 DST 方法中两次 FFT 引入的累积误差所致。而对于 DST 解法，在网格密度较大时，基于 CUDA 的并行版本误差明显大于串行版本，这主要是由于在并行版本中引入了一些累加运算，因此在维数较大的情况下，截断误差导致的影响会更突出。

但是，在大多数应用中，这些误差均处于可以接受的水平，例如，在网格密度为(513×513)的情况下，整个线性系统的方程个数已经达到了 261121 个，而此时基于 CUDA 的并行算法产生的相对误差也小于 10^{-9}，在绝大多数的应用场景下，这一误差是完全可以忽略的。

结束语 泊松方程差分离散后产生的特殊格式的块三对角方程组利用本文描述的基于 DST 的直接求解算法求解与利用针对一般实对称稀疏矩阵的求解算法求解相比具有明显优势，在利用 GPU 加速后，即使在 512×512 的高精度网格划分下，也可以在 1ms 左右得到基本精确的结果，比传统的使用 Cholesky 分解的求解速度快了 1000 倍以上，这使得基于泊松方程求解的实时应用成为可能，同时，这一快速算法也可以大大加速基于泊松方程的科学计算应用。而且，该算法并不局限于泊松方程的求解，很多形式类似于式(1)的偏微分方程均可以离散成为类似于式(2)的形式，此时，只要其对角线矩阵与 **A** 矩阵具有相同的特征向量，便可以采用本文所描述的算法进行求解，因此，该算法具有十分广泛的应用范围。

参 考 文 献

[1] Kincaid D, Cheney W. 数值分析(第三版)[M]. 王国荣，等译. 北京：机械工业出版社，2005

[2] 罗正平. 托卡马克中等离子体平衡计算[D]. 合肥：合肥工业大学，2007

[3] Ferron J R, Walker M L, Lag L L, et al. Real time equilibrium reconstruction for tokamak discharge control[J]. Nucl Fusion, 1998, 38(7): 1055-1066

[4] 廖臣，祝大军，刘盛纲. 五点差分格式求解泊松方程并行算法的研究[J]. 电子科技大学学报，2008，37(01)：81

[5] 许秋燕. 二维泊松方程和扩散方程的一类显式并行算法[D]. 济南：山东大学，2010

[6] 苑野，杨东华. 基于 MPI 的二维泊松方程差分并行实现与测试[J]. 哈尔滨商业大学学报：自然科学版，2011(06)：854

[7] Ryoo S, Rodrigues C I, Baghsorkhi S S, et al. Optimization Principles and Application Performance Evaluation of a Multithreaded GPU Using CUDA[C]// Ppopp' 08: Proceedings of the 2008 Acm Sigplan Symposium on Principles and Practice of Parallel Programming. 2008: 73-82

[8] Hockney R W. A Fast Direct Solution of Poissons Equation Using Fourier Analysis[J]. J Acm, 1965, 12(1): 95

[9] Buzbee B L, Golub G H, Nielson C W. Direct Methods for Solving Poissons Equations[J]. Siam J Numer Anal, 1970, 7(4): 627

[10] Hillis W D, Steele G L. Data Parallel Algorithms[J]. Commun Acm, 1986, 29(12): 1170-1183

[11] Chen Y Q, Davis T A, Hager W W, et al. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate[J]. Acm T Math Software, 2008, 35(3)