CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 39 卷第 8 期 2017 年 8 月
Vol. 39, No. 8, Aug. 2017

文章编号:1007-130X(2017)08-1419-06

一种基于正弦变换的三维泊松方程并行求解算法*

林士伟[1,2,3],张卫民[1,2],方民权[1,2],李　松[1,2]

(1. 国防科技大学计算机学院,湖南 长沙 410073;2. 海军海洋水文气象中心,北京 100161;
3. 国防科技大学海洋科学与工程研究院,湖南 长沙 410073)

摘　要:泊松方程的数值解法在许多物理或者工程问题上得到广泛应用,但是由于大部分三维泊松方程的离散化格式不具有明显的并行性,实际中使用整体迭代的思想,这使得计算效率和稳定性受到了限制。摒弃了传统数值解法中整体迭代的思想,结合离散正弦变换理论(DST),基于 27 点四阶差分格式,将三维泊松方程求解算法在算法级进行修改和并行优化,把整个求解问题转化成多个独立的问题进行求解,稳定性和并行性能得到大幅提升。对于确定的离散化形式,可以使用同一套参数解决不同的泊松方程,大大提高了编程效率。基于共享存储并行模型实现了该算法,实验结果显示,对于给出的实例,新算法具有较好的加速效果,计算结果精度误差约为 10e−5,在可接受范围内,并且计算精度随着维数的升高具有一定提升。

关键词:三维泊松方程;离散正弦变换;并行;OpenMP

中图分类号:TP391　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2017.08.005

A parallel 3D poisson equation solver based on discrete sine transform

LIN Shi-wei[1,2,3], ZHANG Wei-min[1,2], FANG Min-quan[1,2], LI Song[1,2]

(1. College of Computer, National University of Defense Technology, Changsha 410073;
2. Naval Hydrologic Meteorological Center, Beijing 100161;
3. Academy of Ocean Science and Engineering, National University of Defense Technology, Changsha 410073, China)

Abstract: Poisson equations are widely applied in physical and engineering problems. Most of numerical methods for solving 3D Poisson equations have no remarkable parallelism, and use the global iteration methods which limit the computational efficiency and stability. We abandon the idea of global iteration and combine the discrete sine transform theory (DST) with 27-point four-order difference scheme to modify and parallelize the 3D Poisson equation solver at the algorithm level, which separates the whole problem to several smaller independent ones and greatly improves the stability and parallel performance. For a given discrete form, common parameters can be used to solve different Possion equations so that the programming efficiency is greatly improved. The algorithm is implemented on the basis of the shared memory parallel model. Experimental results show that for the given instance, the new algorithm has a good acceleration effect and the final error of the computing result is about 10e−5, within an acceptable rang. And the result accuracy gets improved with the increase of problem scale.

Key words: 3D poisson equation; discrete sine transform; parallel; OpenMP

* **收稿日期**:2017-01-11;**修回日期**:2017-03-26
基金项目:国家自然科学基金(41375113)
通信地址:100161 北京市丰台区六里桥北里 4 号院 1307 信箱
Address: Mail Box 1307, Yard 4, Liuliqiao North, Fengtai District, Beijing 100161, P. R. China

1 引言

泊松方程是数学中一种常用于静电学、机械工程和理论物理的偏微分方程,其求解具有很重要的工程意义。地球物理重、磁、电勘探方法中涉及的地球物理场,如重力位、磁位、电位均满足泊松方程,快速精确地求解三维泊松方程在计算地球物理中具有重要意义。在数学物理方法理论体系中,对于泊松方程的解析求解方法已经有很多成熟的研究[1]。但是,实际中往往使用数值方法来更大程度地运用计算机资源。并且很多泊松方程没有解析解,这时候数值方法将会有更大的优势。对于不同边界下的泊松问题,采用合适的数值方法,可以很好地实现对重力场、磁场、电场的正演模拟。

在现代数值方法中,有限差分方法是最早且很完美的求解方法,在泊松方程这类椭圆型问题上备受关注。但是,这往往需要求解维数较大的稀疏矩阵问题[2],常采用的迭代方法有 Jacobi、Gauss-Seidel 和 SOR 方法[3,4]等。

二维泊松方程较三维有更成熟的研究成果,并行求解常使用 Jacobi 并行迭代算法,因其具有明显的并行性。冯慧等[5]通过不同点的隐式差分格式之间的相互约化来建立新型迭代(Stencil)方法,此方法和 Jacobi 方法同样具有并行性,却比 Jacobi 收敛快。并行求解也有基于并行计算、区域分解和交替分组显式思想有限差分并行迭代算法,在空间上接近 2 阶的收敛速率,它是无条件稳定的,也适用于二维变系数扩散问题[6]。

对于三维泊松方程,首先需要关注的是离散化方法。具有代表性的离散方法是 Spotz 和 Carey[7] 基于中心差分余项修正的方法以及 Gupta 和 Kouatchou[8]采用 Mathmatica 数学软件包得到的 7 点、15 点、19 点、21 点和 27 点差分方法。一般情况下,这些常用的离散化方法阶数越高精度越高,但是稳定性也相对下降。葛永斌等[9]从泰勒展开式入手,通过引进对称和的方法,消去中间的导数项,得到数值求解三维泊松方程的四阶和六阶精度的紧致差分格式,并采用多重等距网格方法进行求解。Wang[10]和 Ge[11]采用不同的途径得到三维泊松方程非等距网格上的四阶精度格式,其中 Wang 采用预条件的共轭梯度法进行求解,Ge 采用沿主控方向的面迭代和部分半粗化的多重网格方法进行求解。

这些三维泊松方程求解算法不具备明显的并行性,并且都是基于整体迭代的思想,当面临较大计算量的时候耗时较长。对于高阶离散化方案,算法实现难度也会大大提高。本文提出一种基于离散正弦变换的方法,在离散化的基础上对系数矩阵进行解相关,不仅并行性具有了极大的提升,算法实现也大大简化,对于工程应用具有重要作用。

2 基于离散正弦变换求解泊松方程的基本原理

2.1 泊松方程离散化方案

考虑在长方体区域 Ω_3 上,$\Omega_3=\{(x,y,z)\mid 0\leqslant x\leqslant M,0\leqslant y\leqslant N,0\leqslant z\leqslant T\}$,其中 M、N、T 为常数,三维泊松方程的一般形式:

$$\partial^2 u(x,y,z)/\partial x^2+\partial^2 u(x,y,z)/\partial y^2+\partial^2 u(x,y,z)/\partial z^2=f(x,y,z)$$

其边界条件为:$u(x,y,z)=g(x,y,z),(x,y,z)\in\partial\Omega_3$,其中 $\partial\Omega_3$ 为 Ω_3 的边界。

x,y,z 方向等间距分别取 m,n,t 个点,使用 27 点 4 阶差分格式:

$$\delta_x=u_{i-1,j,k}-2u_{i,j,k}+u_{i+1,j,k}$$

$$\delta_y=u_{i,j-1,k}-2u_{i,j,k}+u_{i,j+1,k}$$

$$\delta_z=u_{i,j,k-1}-2u_{i,j,k}+u_{i,j,k+1}$$

$$\frac{\partial^2 u}{\partial x^2}=\delta_x^2/(\Delta x^2(1+\frac{1}{12}\delta_x^2))+O(\Delta x^4)$$

$$\frac{\partial^2 u}{\partial y^2}=\delta_y^2)/(\Delta y^2(1+\frac{1}{12}\delta_y^2))+O(\Delta y^4)$$

$$\frac{\partial^2 u}{\partial z^2}=\delta_z^2)/(\Delta z^2(1+\frac{1}{12}\delta_z^2))+O(\Delta z^4))$$

将其代入三维泊松方程的一般形式中,并将其按列排列可得到 $m*n*t$ 阶的线性方程组,表示为:

$$\boldsymbol{H}_1\boldsymbol{U}=\boldsymbol{H}_2\boldsymbol{F}-\boldsymbol{U}_{\partial\Omega_3}=\boldsymbol{C}$$

其中 $\boldsymbol{H}_1$ 和 $\boldsymbol{H}_2$ 是线性方程组的系数矩阵[11],$U_{\partial\Omega_3}$ 是求解区域边界点初始值组成的矩阵,$\boldsymbol{U}$ 是求解变量在离散格点上排列组成的矩阵,如下所示:

$$\boldsymbol{U}=(\boldsymbol{U}_1\quad\boldsymbol{U}_2\quad\cdots\quad\boldsymbol{U}_t)$$

$$\boldsymbol{U}_k=(u_{11k}\quad u_{21k}\quad\cdots\quad u_{m1k}\quad u_{12k}\quad u_{22k}\quad\cdots\quad u_{m2k}\quad\cdots\quad\cdots\quad u_{1nk}\quad\cdots\quad u_{mnk})^{\mathrm{T}}$$

$\boldsymbol{H}_1$ 和 $\boldsymbol{H}_2$ 均是一个 $t*t$ 阶的三对角方块阵。

$$\boldsymbol{H}_1=\begin{bmatrix}\boldsymbol{R}&\boldsymbol{S}&&&&&\\\boldsymbol{S}&\boldsymbol{R}&\boldsymbol{S}&&&&\\&\boldsymbol{S}&\boldsymbol{R}&\boldsymbol{S}&&&\\&&\vdots&\vdots&\vdots&&\\&&&\boldsymbol{S}&\boldsymbol{R}&\boldsymbol{S}&\\&&&&\boldsymbol{S}&\boldsymbol{R}\end{bmatrix}$$

$$\boldsymbol{H}_2=\begin{pmatrix}\boldsymbol{RC} & \boldsymbol{SC} & & & & & \\ \boldsymbol{SC} & \boldsymbol{RC} & \boldsymbol{SC} & & & & \\ & \boldsymbol{SC} & \boldsymbol{RC} & \boldsymbol{SC} & & & \\ & & \vdots & \vdots & \vdots & & \\ & & & \boldsymbol{SC} & \boldsymbol{RC} & \boldsymbol{SC} & \\ & & & & \boldsymbol{SC} & \boldsymbol{RC} & \end{pmatrix}$$

其中，$\boldsymbol{R}$、$\boldsymbol{S}$、$\boldsymbol{RC}$、$\boldsymbol{SC}$ 是 $m*n$ 阶的 9 对角方阵，同时也是一个 3 对角块方阵。这样的线性离散化方程组，可以通过迭代方法进行求解，但是由于变量之间存在过多的依赖关系，很难实现并行计算，在进行高维问题求解时会面临很多问题。

2.2 离散正弦变换并行化原理

使用二维离散正弦变换公式：

$$A(k,l)=4\sum_{i=1}^{m}\sum_{j=1}^{m}x(i,j)\sin\frac{\pi kj}{n}\sin\frac{\pi li}{n}$$

$$k=1,2,\cdots,m-1;l=1,2,\cdots,n-1$$

得到 $m*n$ 阶的二维离散正弦变换矩阵 $\boldsymbol{Q}_2$，常数系数是 $\sqrt{2/(m+1)}\cdot\sqrt{2/(n+1)}$。

记 $Mi=\sin\frac{i\pi}{m+1}$ 和 $Ni=\sin\frac{i\pi}{n+1}$，则：

$$\boldsymbol{Q}_2=\sqrt{2/(m+1)}\cdot\sqrt{2/(n+1)}$$

$$\begin{pmatrix}M1N1 & M2N1 & \cdots & MmN1 & \cdots & M1Nn & M2Nn & \cdots & MmNn\\ M2N1 & M4N1 & \cdots & M2mN1 & \cdots & M2Nn & M4Nn & \cdots & M2mNn\\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots\\ MmN1 & M2mN1 & \cdots & Mm^2N1 & \cdots & MmNn & M2mNn & \cdots & Mm^2Nn\\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots\\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots\\ M1Nn & M2Nn & \cdots & MmNn & \cdots & M1Nn^2 & M2Nn^2 & \cdots & MmNn^2\\ M2Nn & M4Nn & \cdots & M2mNn & \cdots & M2Nn^2 & M4Nn^2 & \cdots & M2mNn^2\\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & & \vdots\\ MmNn & M2mNn & \cdots & Mm^2Nn & \cdots & MmNn^2 & M2mNn^2 & \cdots & Mm^2Nn^2\end{pmatrix}$$

$\boldsymbol{Q}_2$ 是一个正交的对称阵，具有性质 $\boldsymbol{Q}_2=\boldsymbol{Q}_2^{\mathrm{T}}=\boldsymbol{Q}_2^{-1}$。接下来的变换以矩阵 $\boldsymbol{H}_1$ 为基准展开。使用 $\boldsymbol{Q}_2$ 分别对矩阵 $\boldsymbol{R}$、$\boldsymbol{S}$ 进行正交变换，可以推导出下列公式：

$$\boldsymbol{Q}_2^{\mathrm{T}}\boldsymbol{R}\boldsymbol{Q}_2=diag(\boldsymbol{\Psi}_1\quad\boldsymbol{\Psi}_2\quad\cdots\quad\boldsymbol{\Psi}_n)=\boldsymbol{\Pi}$$

$$\boldsymbol{\Psi}_i=diag(\lambda_{i1}\quad\lambda_{i2}\quad\cdots\quad\lambda_{im});i=1,2,\cdots,n$$

$$\lambda_{ij}=\eta_j+2\tau_j\cos\frac{i\pi}{n+1};$$

$$j=1,2,\cdots,m;i=1,2,\cdots,n$$

$$\boldsymbol{Q}_2^{\mathrm{T}}\boldsymbol{S}\boldsymbol{Q}_2=diag(\boldsymbol{T}_1\quad\boldsymbol{T}_2\quad\cdots\quad\boldsymbol{T}_n)=\boldsymbol{\Theta}$$

$$\boldsymbol{T}_i=diag(\mu_{i1}\quad\mu_{i2}\quad\cdots\quad\mu_{im});i=1,2,\cdots,n$$

$$\mu_{ij}=\alpha_j+2\beta_j\cos\frac{i\pi}{n+1};$$

$$j=1,2,\cdots,m;i=1,2,\cdots,n$$

其中，α_j、β_j、η_j 分别是 $\boldsymbol{S}$ 和 $\boldsymbol{R}$ 中两个不同块对角阵的第 j 个特征值，对于相同的离散化方案，我们得到的这四组参数是完全一样的，所以可以用这四组参数来求解不同的泊松方程。

令 $\boldsymbol{Q}_3=diag(\underbrace{\boldsymbol{Q}_2\quad\cdots\quad\boldsymbol{Q}_2}_{t个})$，可以得到：

$$\boldsymbol{Q}_3\boldsymbol{Q}_3^{\mathrm{T}}\begin{pmatrix}\boldsymbol{\Pi} & \boldsymbol{\Theta} & & & & & \\ \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & & & & \\ & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & & & \\ & & \vdots & \vdots & \vdots & & \\ & & & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & \\ & & & & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \end{pmatrix}\boldsymbol{Q}_3\boldsymbol{U}=$$

$$\begin{pmatrix}\boldsymbol{\Pi} & \boldsymbol{\Theta} & & & & & \\ \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & & & & \\ & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & & & \\ & & \vdots & \vdots & \vdots & & \\ & & & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & \\ & & & & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \end{pmatrix}\boldsymbol{Q}_3\boldsymbol{U}=$$

$$\boldsymbol{Q}_3\boldsymbol{H}_2\boldsymbol{F}-\boldsymbol{Q}_3\boldsymbol{U}_{\partial\Omega_3}=\boldsymbol{Q}_3\boldsymbol{C}$$

令 $\boldsymbol{Q}_3\boldsymbol{U}=\hat{\boldsymbol{U}}$ 和 $\boldsymbol{Q}_3\boldsymbol{C}=\hat{\boldsymbol{C}}$，可以得到：

$$\begin{pmatrix}\boldsymbol{\Pi} & \boldsymbol{\Theta} & & & & & \\ \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & & & & \\ & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & & & \\ & & \vdots & \vdots & \vdots & & \\ & & & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \boldsymbol{\Theta} & \\ & & & & \boldsymbol{\Theta} & \boldsymbol{\Pi} & \end{pmatrix}\begin{pmatrix}\hat{\boldsymbol{U}}_1\\ \hat{\boldsymbol{U}}_2\\ \hat{\boldsymbol{U}}_3\\ \vdots\\ \hat{\boldsymbol{U}}_{t-1}\\ \hat{\boldsymbol{U}}_t\end{pmatrix}=\begin{pmatrix}\hat{\boldsymbol{C}}_1\\ \hat{\boldsymbol{C}}_2\\ \hat{\boldsymbol{C}}_3\\ \vdots\\ \hat{\boldsymbol{C}}_{t-1}\\ \hat{\boldsymbol{C}}_t\end{pmatrix}$$

其中 $\hat{\boldsymbol{U}}=(\hat{\boldsymbol{U}}_1\quad\hat{\boldsymbol{U}}_2\quad\cdots\quad\hat{\boldsymbol{U}}_t)^{\mathrm{T}}$，$\hat{\boldsymbol{C}}=(\hat{\boldsymbol{C}}_1\quad\hat{\boldsymbol{C}}_2\quad\cdots\quad\hat{\boldsymbol{C}}_t)^{\mathrm{T}}$。

每一行的表达式可表示为：$\boldsymbol{\Theta}\hat{\boldsymbol{U}}_{k-1}+\boldsymbol{\Pi}\hat{\boldsymbol{U}}_k+\boldsymbol{\Theta}\hat{\boldsymbol{U}}_{k+1}=\hat{\boldsymbol{C}}_k,k=2,3,\cdots,t-1$；对于 $k=1$ 和 t 的情况，只要相应地把等式左侧第一项和最后一项去掉即可。

这样我们得到 t 个 $m*n$ 阶的线性方程组，但是方程组之间还是有相互依赖关系。对于第 k 个线性方程组可重组得到 n 个 m 阶线性方程组：

$$\boldsymbol{T}_j\hat{\boldsymbol{U}}_{jk-1}+\boldsymbol{\Psi}_j\hat{\boldsymbol{U}}_{jk}+\boldsymbol{T}_j\hat{\boldsymbol{U}}_{jk+1}=\hat{\boldsymbol{C}}_{jk},$$

$$j=1,2\cdots,n;k=1,2\cdots,t$$

其中，$\boldsymbol{C}_{jk}=(c_{1jk}\quad c_{2jk}\quad\cdots\quad c_{mjk})^{\mathrm{T}}$；$\hat{\boldsymbol{U}}=(\hat{u}_{1jk}\quad\hat{u}_{2jk}\quad\cdots\quad\hat{u}_{mjk})^{\mathrm{T}}$。

对于第 j 个 m 阶方程组，重组可以得到 m 个线性方程：

$$\mu_{ij}\hat{u}_{ijk-1}+\lambda_{ij}\hat{u}_{ijk}+\mu_{ij}\hat{u}_{ijk+1}=\hat{c}_{ijk},$$

$$i=1,2\cdots,m;j=1,2\cdots,n;k=1,2\cdots,t$$

最终通过重组可以得到 $m*n$ 个 t 阶的线性方程组，矩阵形式表示为：

$$\begin{pmatrix} \lambda_{ij} & \mu_{ij} & & \\ \mu_{ij} & \lambda_{ij} & \mu_{ij} & \\ & \vdots & & \\ & & \mu_{ij} & \lambda_{ij} \end{pmatrix} \begin{pmatrix} \hat{u}_{ij1} \\ \hat{u}_{ij2} \\ \vdots \\ \hat{u}_{ijt} \end{pmatrix} = \begin{pmatrix} \hat{c}_{ij1} \\ \hat{c}_{ij2} \\ \vdots \\ \hat{c}_{ijt} \end{pmatrix},$$

$$i = 1,2\cdots,m, j = 1,2\cdots,n$$

求解得到 $\hat{U}$ 之后，通过离散正弦变换逆变换可以得到 U 。

整个方程组最终重新整理成了 $m*n$ 个独立的系数矩阵是三对角阵的 t 阶线性方程组，大大提高了并行性。

线性方程组系数矩阵维数大幅降低，独立的计算使得累积误差也被分成多个独立的部分，避免了误差的整体积累。同时，变换的过程相当于将多个三角对称阵的特征值进行线性相加，由原来的 α_j、β_j、η_j 变成了 λ 和 μ，将 H_1 主对角线元素的优势进一步加强，重组之后依然保留了主对角线元素的优势，使得累积误差进一步减少，一定程度上提高了数值计算的稳定性。

3 基于离散正弦变换求解算法描述

3.1 算法组成

整个求解算法可以分为初始化、离散变换、求解线性系统、离散逆变换等四大部分。其中对源项的处理包含在初始化中，逆变换和正变换其实是同一种运算。

初始化部分。在进行计算之前，需要进行数据准备。这一部分，我们计算正弦变换矩阵 Q、源项 C 以及矩阵 R、S、RC、SC 的特征值，需要注意的是对于一个确定的离散化方案，这些对角阵的特征值是确定的并且可以直接计算得到，不需要太大的计算量。

离散正弦变换部分直接使用矩阵乘的方式进行矩阵变换，也是算法的核心。总共使用两次，一次是在初始化的时候对源项 C 进行变换，一次是在算法结束时对 $\hat{U}$ 进行的逆变换。

求解线性系统是对独立出来的 $m*n$ 个线性方程组进行求解的部分。由于系数矩阵是三对角阵，采用追赶法可以很容易进行求解。每一次的计算结果都是 $\hat{U}$ 的一部分。

3.2 算法流程

可以将计算流程简单地概述如下：

(1)求得 C，并通过二维正弦变换求得 $\hat{C}_k$；

(2)对 $\hat{C}$ 进行重组，以向量形式进行存储，得到 $m*n$ 个独立的源项向量 $\hat{C}_{ij}$；

(3)指定一对 (i,j) 的值，求得 $t*t$ 阶的系数矩阵，计算出 z 轴方向 t 个元素；

(4)对 i 的值进行遍历，进行 m 次(3)的运算，计算出 y 轴方向 $m*t$ 个元素；

(5)对 j 的值进行遍历，进行 n 次(4)的运算，计算出全局的 $m*n*t$ 个元素；

(6)整理排列 $m*n*t$ 个元素，通过二维正弦逆变换求得最终的结果。

算法流程展示如图 1 所示。

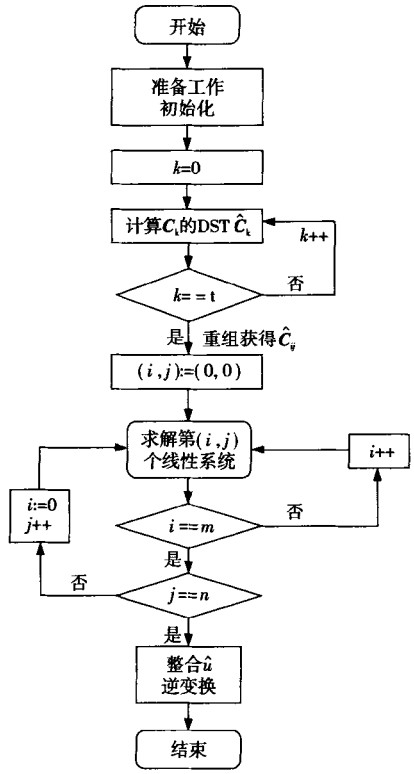


Figure 1 Algorithm implementation flow

图 1 基于 DST 的算法流程

每个 $C_{ij} = (c_{ij1} \quad c_{ij2} \quad \cdots \quad c_{ijt})^T$ 在计算 DST (Discrete Sine Transform)的时候是独立不影响的。计算结束之后重组为 $\hat{C}_{ij}$ 。每一个 $\hat{C}_{ij}$ 对应的 $m*n$ 个独立的线性方程组可以进行并行计算。每个 $\hat{C}_{ij}$ 之间互不相关，可以进行并行计算。所以，对于并行实现的依赖关系可以用图 2 表示。

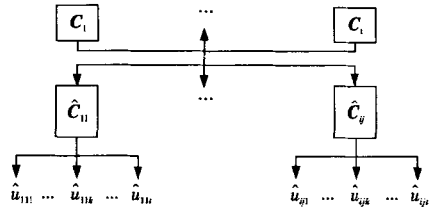


Figure 2 Data dependence relationship

图 2 并行实现依赖关系

4 实验结果与分析

4.1 实验平台

本文在两个平台上进行了测试,第一个平台包含一个 4 核龙芯 3A 芯片,第二个测试平台包含两个 8 核 Intel(R) Xeon(R) CPU E5-2670。分别记为测试平台 a 和测试平台 b。均采用 OpenMP 的 C 语言扩展来实现并行算法,采用 Intel C Compiler 13.0.0 编译器编译。

4.2 求解目标

泊松方程:

$$\nabla^2 u = -3\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z)$$

边界条件:

$$u(0,y,z)=u(1,y,z)=u(x,0,z)=$$
$$u(x,1,z)=u(x,y,0)=u(x,y,1)=0$$

解析解:

$$u(x,y)=\sin(\pi x)\sin(\pi y)\sin(\pi z)$$

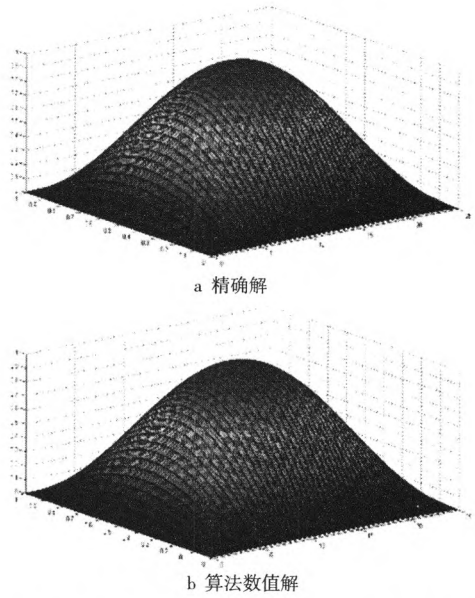
4.3 实验结果

为了能够更加直观地表示出运算结果,我们将 x 方向和 y 方向的值排成一列作为矩阵的行,z 方向的值作为矩阵的列,在三维空间将结果表示出来。结果如图 3 所示。

在测试平台 a 上,对 50 * 50 * 50 和 100 * 100 * 100 维数的数值解的误差进行统计,结果显示绝对误差分别为 1.11e−5 和 4.41e−6,可见随着维数的增加,模拟精度有所提高,同时文献[12]中也有类似的结果。

4.4 加速性能分析

对不同维数的加速性能进行统计,如图 4 和图 5 所示(括号中的数值是网格规模)。



a 精确解

b 算法数值解

Figure 3 Analytical solution and numerical solution

图 3 精确解和算法数值解

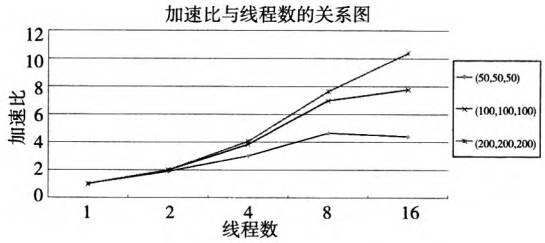


Figure 5 Relationship between speedup and threads on platform b.

图 5 测试平台 b 上加速比与线程数的关系图

从测试平台 a 上的结果可以看出,当网格规模较小的时候,随着线程数的增加,加速比先增加后减小,最后趋于稳定。在线程数等于 4 的时候加速比最大。当网格规模较大的时候,加速比随着线程数先线性增长,后趋于稳定。线程数固定时,随着网格数的增加,整体的加速比也在增加。从趋势来看,当网格数达到一定数量的时候,具有线性加速比。

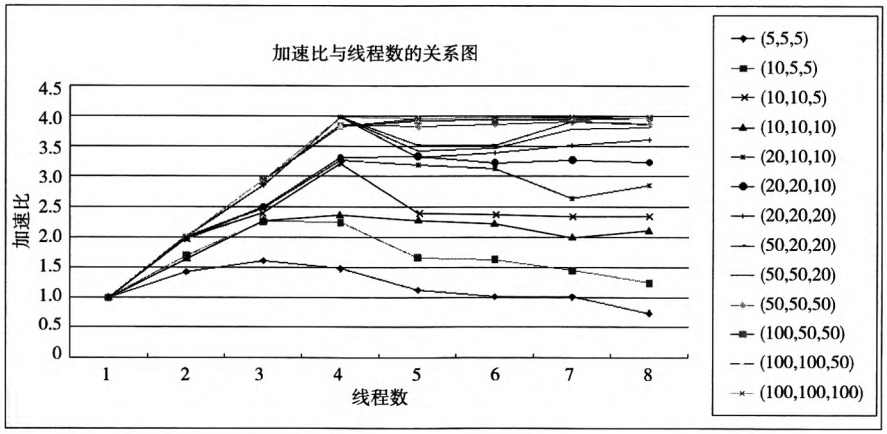


Figure 4 Relationship between speedup and threads on platform a.

图 4 测试平台 a 上加速比与线程数的关系图

从测试平台 b 上的结果可以看到,算法随着进程数的增加依旧具有很好的加速性能。

对于 200 * 200 * 200 维度的计算在 16 个核心的时候增加 OpenMP 的线程数为 64 可以得到 13.83158的加速比,进一步验证了算法本身具有更好的加速性能。

5 结束语

本文使用四阶 27 点差分格式对偏微分方程离散化,通过对系数矩阵进行二维正弦变换,将整个线性方程转化成有限个低维的独立的线性方程组进行求解。分别进行了串行和并行实验测试,验证了方案的正确性。通过实验结果分析比较得出了网格规模、线程数与加速比之间的相互关系,达到了预期的效果,但并行实现的内存优化还有待改进。

此外,根据串行中间实验结果发现 DST 使用了 80%的计算时间,所以 DST 的实现方法可以在算法级进行改进(例如改为 FFT 实现[14,15]),以获得更好的并行效率。线性方程组求解也可采用其他方法(例如 SOR、Cholesky 分解等),以获得更高的计算效率。计算机高性能计算也向多核和众核方向发展,将算法基于 CPU/MIC(Many Integrated Cores)异构系统进行实现,将会有更高的加速性能[16,17]。

参考文献:

[1] Yan Zhen-jun. Method of mathematical physics[M]. Hefei: Press of University of Science and Technology of China, 1999. (in Chinese)

[2] Hockney R W. A fast direct solution of Poisson's equation using Fourier analysis[J]. Journal of ACM,1965,12(1):95-113.

[3] Xu Shu-fang,Gao Li,Zhang Ping-wen. Numerical linear algebra[M]. Beijing:Peking University Press,2000. (in Chinese)

[4] Varga R S. Matrx iterative analysis[M]. NJ:Prentice-Hall, Englewood Cliffs,1962.

[5] Feng Hui, Zhang Bao-lin, Liu Yang. The application of finite defference approximation of Poisson equation[J]. Computational Mathematics,2005,35(8):901-909. (in Chinese)

[6] Xu Qiu-yan. An explicit parallel algorithm for 2D Poisson equation and diffusion equation[D]. Jinan:Shandong University,2010. (in Chinese)

[7] Spotz W F, Carey G F. A high order compact formulation for the 3D poisson equation[J]. Numerical Methods for Partial Differential Equations,1996,12:235-243.

[8] Gupta M M, Kouatchou J. Symbolic derivation of finite difference approximations for the three dimensional poisson equation[J]. Numerical Methods for Partial Differential Equation,1998,14:593-606.

[9] Ge Yong-bin, Tian Zhen-fu, Ma Hong-lei. High precision multigrid method for 3D Poission equation[J]. Applied Mathematics,2006,19(2):313-318. (in Chinese)

[10] Wang J, Zhong W J, Zhang J. A general meshsize fourth-order compact difference discretization scheme for 3D Poisson equation[J]. Applied Mathematics and Computation, 2006,183:804-812.

[11] Ge Y B. Multigrid method and fourth-order compact difference discretization scheme with unequal meshsizes for 3D Poisson equation[J]. J Comput Phys,2010,229:6381-6391.

[12] Kyei Y,Roop J P,Tang G. A family of sixth-order compact finite-difference schemes for the three-dimensional Poisson equation[J]. Advances in Numerical Analysis,2010,2010.

[13] Shiferaw A,Chand Mittal R. An efficient direct method to solve the three dimensional Poisson's equation[J]. American Journal of Computational Mathematics,2011,1(4):285-293.

[14] Guo Li-cai,Liu Yan-jun. Efficient implementation of FFT on Loongson 3A CPU[J]. Journal of Chinese Computer System,2012,33(3):594-597. (in Chinese)

[15] Wu J, Jaja J. High performance FFT based Poisson solver on a CPU-GPU heterogeneous platform[C]// Proc of 2013 IEEE 27th International Symposium on Parallel & Distributed Processing (IPDPS),2013:115-125.

[16] Yue Xiao-ning,Xiao Bing-jia, Luo Zheng-ping. Fast direct solver for 2D poission equation on CUDA[J]. Computer Science,2013,40(10):21-23. (in Chinese)

[17] Fang Min-quan,Zhang Wei-min, Gao Chang, et al. Parallelizing and optimizing maximum noise fraction rotation on multi-cores and many-cores[J]. Journal of Software,2015, 26(Suppl(2)):247-256. (in Chinese)

附中文参考文献:

[1] 严镇军. 数学物理方法[M]. 合肥:中国科学技术大学出版社,1999.

[3] 徐树方,高立,张平文. 数值线性代数[M]. 北京:北京大学出版社,2000.

[5] 冯慧,张宝琳,刘扬. Poisson 方程有限差分逼近的数学 stencil 及其应用[J]. 计算数学,2005,35(8):901-909.

[6] 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.

[9] 葛永斌,田振夫,马红磊. 三维泊松方程的高精度多重网格解法 [J]. 应用数学,2006,19(2):313-318.

[14] 郭利财,刘燕君. 龙芯 3A 处理器上 FFT 的高效实现[J]. 小型微型计算机系统,2012,33(3):594-597.

[16] 岳小宁,肖炳甲,罗正平. 基于 CUDA 的二维泊松方程快速直接求解[J]. 计算机科学,2013,40(10):21-23.

[17] 方民权,张卫民,高畅,等. 多核与众核上 MNF 并行算法与性能优化[J]. 软件学报,2015,26(Suppl(2)):247-256.

作者简介:

林士伟(1991 -),男,河南南阳人,硕士生,研究方向为并行算法和数据同化。**E-mail**:mylifeyear@163.com

LIN Shi-wei, born in 1991, MS candidate, his research interests include parallel algorithm, and data assimilation.