一种针对大波数 Helmholtz 方程的高性能并行预条件迭代求解算法

程东升[1,2] 刘志勇[3] 薛国伟[1] 高月芳[1]

(深圳信息职业技术学院软件学院 广东 深圳 518172)[1]

(中山大学广东省科学重点实验室 广州 510275)[2] (深圳职业技术学院工业中心 广东 深圳 518055)[3]

摘 要 针对传统串行迭代法求解大波数 Helmholtz 方程存在效率低下且受限于单机内存的问题,提出了一种基于消息传递接口(Message Passing Interface,MPI)的并行预条件迭代法。该算法利用复移位拉普拉斯算子对 Helmholtz 方程进行预条件处理,联合稳定双共轭梯度法和基于矩阵的多重网格法来求解预条件方程离散后的大规模线性系统,在 Linux 集群系统上基于 MPI 环境实现了求解算法的并行计算,重点解决了多重网格的并行划分、信息传递和多重网格组件的构建问题。数值实验表明,对于大波数问题,提出的算法具有良好的并行加速比,相较于串行算法极大地提高了计算效率。

关键词 Helmholtz 方程,并行,预条件子,稳定双共轭梯度法,多重网格

中图法分类号 O246 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.07.051

High-performance Parallel Preconditioned Iterative Solver for Helmholtz Equation with Large Wavenumbers

CHENG Dong-sheng[1,2] LIU Zhi-yong[3] XUE Guo-wei[1] GAO Yue-fang[1]

(Department of Software Engineering, Shenzhen Institute of Information and Technology, Shenzhen, Guangdong 518172, China)[1]

(Guangdong Province Key Laboratory of Computational Science, Sun Yat-sen University, Guangzhou 510275, China)[2]

(Industrial Center, Shenzhen Polytechnic, Shenzhen, Guangdong 518055, China)[3]

Abstract For solving the Helmholtz equation with a large wavenumber, the traditional sequential iterative solver is inefficient and limited to the memory of a single computer. To deal with these problems, a parallel preconditioned iterative solver was proposed based on the message passing interface(MPI). The complex shifted-Laplacian is used to precondition the Helmholtz equation, and the Krylov subspace method Bi-CGSTAB combined with the matrix-based multigrid method is employed to solve the large linear system resulted from discretization of the preconditioned equation. Parallelization of the preconditioned solver is achieved under the environment of MPI on the Linux cluster system, and the problems of parallel partition of the multigrid, information transfer and construction of the multigrid components are mainly tackled. Finally, numerical experiments were given. The results show that the proposed method contributes to an excellent parallel speedup, and improves the computing efficiency considerably compared with the sequential iterative solver.

Keywords Helmholtz equation, Parallel, Preconditioner, Bi-CGSTAB, Multigrid

1 引言

Helmholtz 方程常用来刻画波传播和散射现象,在地球物理、油气勘探、海洋技术、航空航天等科学技术领域有着广泛的应用。建立高效的 Helmholtz 方程数值模拟方法是计算科学领域中的一个研究热点,Helmholtz 方程的数值求解的难点在于处理大波数的"污染效应"[1],即随着波数的增加,若保持一定的求解精度,必须充分加细离散网格。在实际应用中,至少需要保持波数与离散步长的乘积为一个较小的常数。因此,当波数较大时,要求离散步长很小,从而导致方程离散后得到一个大规模的线性系统(方程组)。线性系统的求解方法总体有两类:直接法和迭代法。直接法在求解过程中带来

到稿日期:2017-05-18 返修日期:2017-08-14 本文受国家自然科学基金项目(11701389),广东省自然科学基金项目(2015A030313592),中山大学广东省计算科学重点实验室,深圳市科技计划项目(JCYJ20160527102119211,JCYJ20150630114140642),广东省优秀青年教师项目(YQ2014122),深圳信息职业技术学院科研培育项目(QN201710)资助。

程东升(1983-),男,博士,副教授,主要研究方向为偏微分方程数值解、并行计算、图像处理、大数据应用技术,E-mail:chengds@sziit.edu.cn;刘志勇(1975-),男,博士,副教授,主要研究方向为模式识别、计算机视觉,E-mail:liuzhiyong@szpt.edu.cn(通信作者);薛国伟(1982-),男,博士,高级工程师,主要研究方向为并行计算、图像处理;高月芳(1979-),女,博士,副教授,主要研究方向为智能控制。

了大量非零元素，增加了计算量和存储量，很难处理大规模问题，因此迭代法更受青睐。对于 Helmholtz 方程离散后的线性系统，其条件数很大，导致传统迭代法失效，因此需要对其进行预处理，发展预条件迭代法。Helmholtz 的预条件处理引起了许多研究者的关注[2-5]。Erlangga 等[2]提出使用复移位拉普拉斯算子来预处理 Helmholtz 方程，并使用多重网格法逼近预条件子的逆，然后利用 Krylov 子空间迭代法求解预条件系统。该方法效果良好且易于实施，因此被广泛采用。对于大规模问题，即使采用了预条件技术，传统的串行迭代法的计算效率依然低下，并且受到了计算机内存的限制，因此需要发展并行算法来进行快速计算。超大规模并行计算的实现方式主要有基于 CPU 的 MPI[6]和基于 GPU+CPU 的统一计算设备架构(Compute Unified Device Architecture，CUDA)[7]。近年来，关于 Helmholtz 方程并行计算的研究并不多[8-11]，且计算波数较小，计算规模(未知量个数)不大。文献[8]针对三维 Helmholtz 问题，提出一种基于三维半粗化多重网格的 MPI 并行预条件迭代法，其侧重于三维半粗化多重网格延拓算子的构建，最高计算波数为 50，使用的处理器数量至多为 25。文献[9]针对三维 Helmholtz 方程，提出一种基于代数多重网格和不完全 LU 分解的 MPI 并行预条件迭代法，其通过 PESTc 接口实现并行计算，最多使用了 16 个处理器，最大计算规模小于 20000。文献[10]针对二维 Helmholtz 方程，提出了一种基于复移位拉普拉斯算子的 GPU 并行预条件迭代法，其通过 CUDA 架构实现并行计算，最高波数为 640，最大计算规模为 1024^2(约为 100 万)。文献[11]则针对高频问题，提出了一种基于区域分解的并行算法。

本文针对二维大波数 Helmholtz 方程，提出了一种新的基于 MPI 的并行预条件迭代法。首先，把整个计算区域划分成若干个规模相当的子区域，实现计算任务(数据)的并行划分，并对子区域进行适当的拓展，以实现相邻子区域间的数据通信。然后，在子区域内，运用复移位拉普拉斯算子预处理 Helmholtz 方程，并利用优化的有限差分方案[12]进行离散。对于离散后得到的预条件线性系统，联合 Krylov 子空间方法 Bi-CGSTAB(稳定双共轭梯度法)和基于矩阵的多重网格法进行并行迭代求解。以矩阵向量积为特征的 Krylov 子空间法的并行执行比较容易[13]；而多重网格法[14]因涉及到多个层次的网格，进行并行计算比较困难，是算法需解决的重点。本文充分利用网格的拓展、数据的稀疏结构和 MPI 的消息传递机制来解决多重网格并行执行中的组件构建与信息传递等问题。最后，在 Linux 集群系统上对算法进行了测试，数值实验表明，对于大波数问题，算法具有良好的并行加速比与效率，相较于串行算法，大大节省了计算时间。本文计算的最高波数为 8000，相应的计算规模为16245^2(约为 2.6 亿)，最多用到 256 个处理器。

2 Helmholtz 方程的预条件迭代法

2.1 方程描述

与波传播相关的二维 Helmholtz 方程为：

$$-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}-k^2 u=g \tag{1}$$

其中，u 通常代表频率域内的压力场，g 表示震源，k 为波数(其通常与波的频率和传播速度有关)。为了在有限区域内对方程(1)进行数值求解，通常采用吸收边界条件的方式来消除边界处的非物理反射。本文采用完美匹配层[15](Perfectly Matched Layer，PML)边界条件，其具有良好的吸收效果。应用 PML 技巧把方程(1)的无限区域截断成一个有限计算区域，得到带 PML 边界条件的 Helmholtz 方程：

$$Au:=-\frac{\partial}{\partial x}\left(\frac{e_y}{e_x}\frac{\partial u}{\partial x}\right)-\frac{\partial}{\partial y}\left(\frac{e_x}{e_y}\frac{\partial u}{\partial y}\right)-k^2 u=g \tag{2}$$

其中，$e_x=1-i\sigma_x/\omega$，$e_y=1-i\sigma_y/\omega$，ω 为角频率，σ_x 是关于 x 的函数，定义为：

$$\sigma_x:=\begin{cases}2\pi a_0 f_0\dfrac{l_x}{L_{pml}}, & \text{在 PML 内}\\ 0, & \text{在 PML 外}\end{cases}$$

其中，f_0 为震源主频；a_0 为常数，通常取值为 1.79；L_{pml} 为 PML 的厚度，即在 x 轴方向所包含的网格点数；l_x 是点(x,y)到 PML 区域与内部区域界面的距离，$(x,y)\in\Omega_0\cup\Omega_1$ 时，$l_x=0$。σ_y 的定义与 σ_x 类似，$(x,y)\in\Omega_0\cup\Omega_2$ 时，$l_y=0$。当内部计算区域 Ω_0 为边长是 H 的正方形时，其可以规范化为$[0,1]\times[0,1]$，规范化后的波数为 kH，被称为无维波数。后文涉及的均为无维波数，仍记作 k。

2.2 基于复移位拉普拉斯算子和多重网格的预条件技术

方程(2)离散得到的线性系统是不定的，即其系数矩阵特征值的分布跨越复平面上左、右两个半平面，具有很大的条件数，特别对于大波数情形，将导致标准迭代法收敛速度慢甚至发散。为了提高迭代法的效率，需要应用预条件技术。对于 Helmholtz 方程，研究者们提出了许多预条件方案，其中基于复移位拉普拉斯算子的预条件具有很好的效果，得到了广泛的应用。在 PML 边界条件下，复移位拉普拉斯算子如下：

$$M:=-\frac{\partial}{\partial x}\left(\frac{e_y}{e_x}\frac{\partial}{\partial x}\right)-\frac{\partial}{\partial y}\left(\frac{e_x}{e_y}\frac{\partial}{\partial y}\right)-(1-\alpha i)k^2$$

其中，α 为常数，通常取值为 0.5；i 为虚数单位。利用算子 M 对方程(2)进行预处理，得到算子层次的预条件系统：

$$AM^{-1}v=g,\ v=Mu \tag{3}$$

对式(3)进行离散，从而得到离散预条件系统，其中复移位拉普拉斯算子 M 离散后得到的矩阵即为预条件子。对于离散后的预条件系统，利用 Krylov 子空间迭代法 BI-CGSTAB(稳定的双共轭梯度法)进行求解，称为预条件 BI-CGSTAB 法。Krylov 子空间迭代法是基于 Krylov 子空间投影而建立的动态迭代法，其主要操作在于矩阵向量积，在实际应用中，相对于雅克比、高斯赛德尔等静态迭代法具有良好的效果。对于求解 Helmholtz 方程离散后的线性系统，BI-CGSTAB 具有较快的收敛速度。

在预条件 BI-CGSTAB 法的执行过程中，需要求解预条件子(矩阵)的逆，这在技术上可以转化为求解另一个系数矩阵为预条件子的辅助线性系统。辅助线性系统不需要精确求解，只须近似求解即可。但近似解的精确度越高，预条件的效果就越好，预条件 BI-CGSTAB 法的收敛速度就越快。但得到高精确度的近似解需要付出高昂的代价。因此，需要在近似解的精确度和计算成本之间寻求效果极大化，而多重网格

法可以很好地满足这一需求，即可以用较小的代价得到较好的近似解。在预条件 BI-CGSTAB 法中，BI-CGSTAB 作为外迭代求解预条件系统，而多重网格法作为内迭代近似求解预条件子的逆。

多重网格的组件包括粗网格算子、细网格算子、限制算子和延拓算子。其中，延拓算子是多重网格最重要的组件。对于大波数 Helmholtz 方程，基于经典线性延拓算子的多重网格效果较差，即由内迭代求得的辅助线性系统的解的精确度不高，从而导致外迭代 BI-CGSTAB 法的收敛速度慢甚至发散。为了提升多重网格的性能，本文将采用基于矩阵的延拓算子，而相应的多重网格则称为基于矩阵的多重网格。基于矩阵的延拓算子是根据预条件子的代数信息构造的，可以很好地提升多重网格的性能。

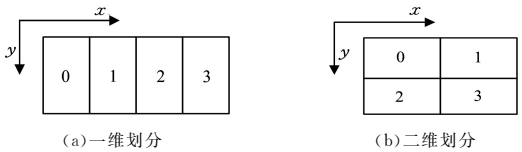
3　计算区域的并行划分与拓展

本节介绍了包含离散网格的计算区域的并行划分与拓展，这是并行预条件迭代算法的基础。

3.1　计算区域的并行划分

集群上基于 MPI 的并行计算是指利用多个处理器同时计算一个大的任务，每个处理器负责处理一个小的子任务，且处理器之间可以进行数据通信。因此，首先需要划分计算任务，即把大的任务分解为许多小的子任务。另一方面，任务的划分本质上也是计算数据的划分。本文求解的问题具有明显的几何背景（二维计算区域），因此可以通过计算区域的划分来实现计算任务的划分。包含离散网格的计算区域提供了计算所需的所有数据，故计算区域的划分决定了数据的划分。相对于直接划分数据，这种利用几何背景对数据进行划分的方法给后续的并行计算带来了极大的便利与效率。

包含所有离散网格点的计算区域对应一个大数据集，区域划分后，每个子区域包含一部分网格点，并对应一个小的数据集。总体上，计算区域的划分没有特别的规则，只考虑了负载平衡，尽量将其划分成多个大致相等的部分，从而使每个处理器承担的计算量和通信量类似。计算区域的划分通常有两种方式：一维划分和二维划分，如图 1 所示。



(a)一维划分　(b)二维划分

图 1　计算区域的两种划分方式

Fig. 1　Two partitions of computational domain

关于这两种划分方式，存在以下命题。

命题 1　对计算区域分别进行一维划分和二维划分，并得到相同数量的子区域。假定并行计算的数据通信发生在两个相邻子区域的交界线处，且每个交界线处有相同数量的数据需要通信，则二维划分产生的总数据通信量小于一维划分产生的总数据通信量。

证明：可以看出，总的数据通信量是由划分界线的数量决定的。令 n 为划分产生的子区域个数，则容易计算出一维划分和二维划分产生的总界线数分别为 $n-1$ 和 $2(\sqrt[2]{n}-1)$。当 $n\geqslant 3$ 时，有：

$$2(\sqrt[2]{n}-1)<n-1$$

上式表明，二维划分产生的总数据通信量小于一维划分产生的总数据通信量。

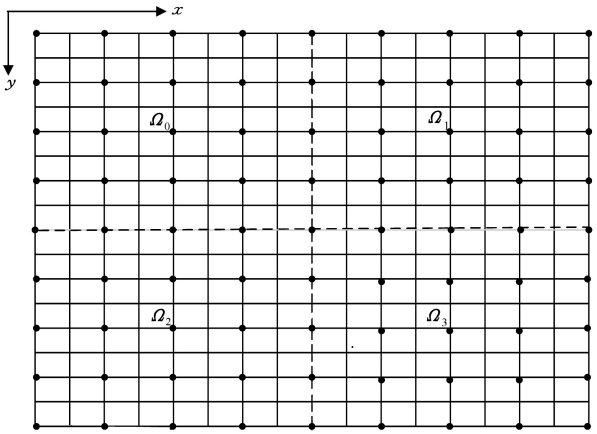
根据命题 1，在相同条件下，本文采用二维划分的方式产生的数据通信量较少，可以提高并行加速比和效率。

3.2　计算区域的拓展

并行划分将计算区域分解成许多个子区域，而相邻的子区域间的公共部分只有交界线，没有重叠的面积，这意味着两个相邻子区域对应的数据没有重叠，从而使得相邻子区域对应的进程无法相互从对方获取需要的数据信息，进而导致并行计算无法执行。为了解决这个问题，需要对子区域进行拓展，即把一个子区域从边界处拓展到其相邻的子区域，使得其与相邻子区域间有重叠的面积，从而使得它们对应的数据有重叠。一个拓展后的子区域称为原计算区域的一个覆盖（covering），由于有重叠的面积，两个相邻覆盖对应的进程之间可以相互获取对方的数据信息。

由于子区域包含了多重网格，其拓展需要根据网格的层次有序进行。子区域的拓展并不是随意的，必须满足两个原则：1）需要保证在拓展后的子区域内，数据信息可以通过限制算子和延拓算子在多重网格的各层网格之间流动，这意味着需要保证多重网格的嵌套性，即使得每个细网格点至少有两个相邻的粗网格点，否则粗网格点上的信息无法传递到细网格上；2）不能过度拓展子区域。在满足原则 1）的情况下，子区域拓展得越少越好，这样可以节省计算量。

图 2 显示了含有两层（粗和细）网格的计算区域的划分，计算区域被等分为 4 个子区域 Ω_0，Ω_1，Ω_2，Ω_3，其中的点线表示两个相邻子区域间的界线。



注：实心圆点为粗网格点，其余为细网格点

图 2　含有两层网格的计算区域的划分

Fig. 2　Partition of computational domain with two level grids

基于图 2 中的划分对子区域进行拓展，结果如图 3 所示。可以看出，图 2 中的子区域 Ω_0，Ω_1，Ω_2 被拓展为覆盖 Λ_0，Λ_1，Λ_2，而 $\Lambda_3=\Omega_3$。具体地，Ω_0 被拓展到其相邻子区域 Ω_1，Ω_2，Ω_3，而 Ω_1，Ω_2 分别都被拓展到 Ω_3，而 Ω_3 保持不变。图 3 中矩形阴影部分为相邻子区域拓展后的重叠部分，即相邻覆盖的重叠部分。矩形重叠部分，长边对应的边界线是由人为划分

和拓展而产生的，相对于计算区域的真实边界，将其称为人工边界。图 3 的拓展方法满足 3.2 节中提到的两个原则。首先，重叠部分的细网格和粗网格保持了很好的嵌套关系，即每个细网格点都有两个相邻的粗网格点。另外，也很容易看出这种拓展是最少的。对于含有多重网格的计算区域，由于多重网格可以看成多个两层网格嵌套而成，因此其拓展可以类似地分层次实现。

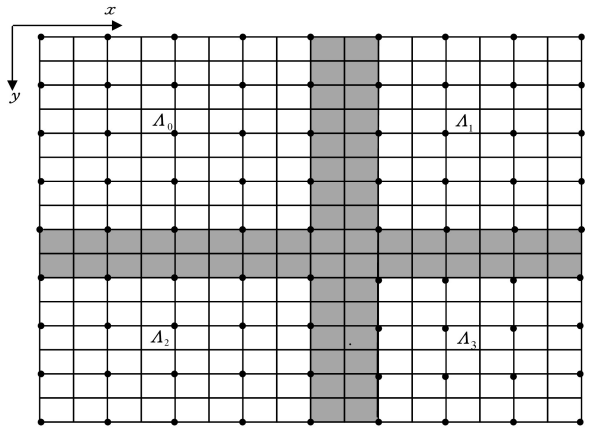


图 3 子区域拓展

Fig. 3 Extension of sub-domains

4 并行预条件迭代求解

对计算区域进行并行划分与拓展后，本节介绍了预条件系统(3)的并行迭代求解，其中 BI-CGSTAB 外迭代易于并行实现，因此重点介绍多重网格内迭代的并行执行。

4.1 预条件系统的离散

在计算区域的各个覆盖内分别离散预条件系统(3)。覆盖内有两类边界：1)原计算区域的边界；2)人工边界。对属于每个覆盖的人工边界，设其外方向相邻处存在一个虚拟边界，并在虚拟边界处添加 0 值边界条件，这样可以实现系统(3)在各覆盖内的单独离散。因为虚拟边界的 0 值边界条件，一个覆盖的人工边界上的数据值为假值，但由于这个覆盖的人工边界处于相邻覆盖的内部区域，因此，这个覆盖的人工边界上的真实数据值可以通过数据通讯从相邻覆盖处获取。

下面使用文献[12]提出的优化 9 点有限差分方案来离散预条件系统(3)。这种优化的 9 差分方案的构造简单且可以一定程度上抑制大波数导致的污染效应。离散后，在每个覆盖内得到离散预条件线性系统：

$$AM^{-1}v=g \tag{4}$$

$$Mu=v \tag{5}$$

其中，矩阵 $A,M\in C^{Np\times Np}$，向量 $v,g,u\in C^{Np}$，C 为复数集，Np 表示每个覆盖内的未知量个数(网格点数)。矩阵 M 为预条件子，由复移位的拉普拉斯算子 M 离散得到。分别联合 BI-CGSTAB 和多重网格法求解线性系统(4)和线性系统(5)，其中 BI-CGSTAB 作为外迭代来求解线性系统(4)，而多重网格作为内迭代来近似求解线性系统(5)，所求的向量 u 即为方程(2)对应的各个覆盖的数值解，合并所有的 u 即可得到 Helmholtz 方程(2)完整的数值解。

所有覆盖对应的计算任务由多个处理器并行协作完成，而一个覆盖对应的计算任务由一个处理器单独完成。由于计算数据来自于 Helmholtz 方程在覆盖内的离散，因此它们具有明显的几何结构。其中，矩阵数据 A 和 M 是稀疏 9 对角的，在计算机内存中采用稀疏压缩存储。数据的几何结构对建立有效的并行计算起到了非常重要的作用。所有的矩阵和向量数据都以一维数组的形式存储在本地内存。当各处理器之间需要相互使用数据时，通过 MPI 的通信函数(MPI_Isend，MPI_Irecv，MPI_Allreduce 等)进行数据获取。

在求解线性系统(4)和线性系统(5)时，BI-CGSTAB 外迭代的并行执行较容易实现，而多重网格内迭代由于涉及多个层次网格之间的纵横向信息的传递，其并行实现较为困难。在第 3 节解决了多重网格的并行划分，本节将解决并行多重网格的组件构建。多重网格的组件包括限制算子、延拓算子、粗(细)网格算子，其中细网格用来消除高频误差，粗网格用来消除低频误差，限制算子将细网格上的信息传递到粗网格上，而延拓算子将粗网格上的信息传递到细网格上。在多重网格迭代中，我们采用经典的全加权限制算子和基于矩阵的延拓算子，并利用 Galerkin 原则得到粗网格算子。在并行计算时，除限制算子外，还需要构建延拓算子和粗网格算子。

4.2 并行延拓算子的构建

基于预条件子 M 构建并行多重网格延拓算子。图 4 给出了 1 个粗网格和 4 个细网格单元，其中 p_2,p_4,p_5,p_6,p_8 为细网格点，而 p_1,p_3,p_7,p_9 既是粗网格点也是细网格点，即在这些点上粗网格点和细网格点重合。

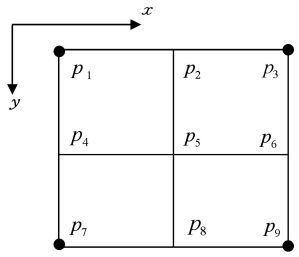


图 4 1 个粗网格单元和 4 个细网格单元

Fig. 4 One coarse-grid cell and four fine-grid cells

令 h 表示离散步长，并用上标 $\cdot^h$ 和 $\cdot^H$ 或下标 $\cdot_h$ 和 $\cdot_H$ 分别表示与细网格和粗网格相关的量，例如记 Λ_h 和 Λ_H 分别表示细网格点集和粗网格点集。对于 $j\in\Lambda_h$，令 $j=(j_x^h,j_y^h)$，$j_x^h,j_y^h\in\{h,2h,\cdots,Np\cdot h\}$ 分别为点 j 在 x 和 y 方向的坐标分量。对于点 $j,l\in\Lambda_h$，定义 l 相对于 j 在 x 和 y 方向的移动分别为：

$$\gamma_x(j,l)=\frac{l_x^h-j_x^h}{h},\ \gamma_y(j,l)=\frac{l_y^h-j_y^h}{h}$$

令 $E:=\{-1,0,1\}$，如果 $\gamma_x(j,l),\gamma_y(j,l)\in E$ 且 $j\neq l$，则称网格点 l 为 j 的邻居，当 $l\in\Lambda_h\cap\Lambda_H$ 时，l 是 j 的粗网格邻居。另外，当 $\gamma_x(j,l),\gamma_y(j,l)$ 有且仅有一个值非零，则称 l 为 j 的直接邻居。令 N_j 和 N_j^c 分别表示 j 的邻居集和粗网格邻居集，如对于点 p_2 有 $N_{p_2}=\{p_1,p_3,p_4,p_5,p_6\}$，$N_{p_2}^c=\{p_1,p_3\}$。同时，$p_1,p_3$ 也是 p_2 的直接邻居。根据粗网格邻居的个数，可以把 Λ_h 分成 3 个互不相交的子集，即 $\Lambda_h=\Lambda_H\cup\Lambda_1\cup\Lambda_2$，其中 Λ_1 中的点有 2 个粗网格邻居，如 p_1,p_3 是 p_2

的 2 个粗网格邻居；而 Λ_2 有 4 个粗网格邻居，如 $N^c_{p_5}=\{p_1,p_3,p_7,p_9\}$。

记 $S(j):=[S_{j_x,j_y}:j_x,j_y\in E]$为预条件子 M 在点 j 处的差分离散模板。对于点 j 与其直接邻居点 l，定义函数 σ,δ 如下：当 $\gamma_x(j,l)\neq 0$ 时，

$$\sigma(j,l):=|\sum_{j_y\in E}S_{\gamma_x(j,l),j_y}(j)|,\delta(j,l):=\max_{j_y\in E}\{|S_{\gamma_x(j,l),j_y}(j)|\}$$

当 $\gamma_y(j,l)\neq 0$ 时，

$$\sigma(j,l):=|\sum_{j_x\in E}S_{j_x,\gamma_y(j,l)}(j)|,\delta(j,l):=\max_{j_x\in E}\{|S_{j_x,\gamma_y(j,l)}(j)|\}$$

令 e_h 和 e_H 分别表示细网格和粗网格上的函数。若 e_H 已知，则 e_h 可以通过延拓算子基于 e_H 获得，即对于 $j\in\Lambda_H$，有 $e_h(j)=e_H(j)$，而对于 $j\in\Lambda_1\bigcup\Lambda_2$，有：

$$e_h(j)=\sum_{l\in N^c_j}w(j,l)e_H(l)$$

其中，$w(j,l)$是待求解的延拓系数。对于 $j\in\Lambda_1$，记 l_1 和 l_2 为其两个粗网格邻居，则系数 $w(j,l_1)$的计算式如下：

$$w(j,l_1)=\frac{\max\{\sigma(j,l_1),\delta(j,l_1)\}}{\max\{\sigma(j,l_1),\delta(j,l_1)\}+\max\{\sigma(j,l_2),\delta(j,l_2)\}}$$

为确保优良的性能，根据黑盒子多重网格延拓算子的构造要求[14]，系数 $w(j,l_2)$应满足 $w(j,l_1)+w(j,l_2)=1$，即 $w(j,l_2)=1-w(j,l_1)$。最后，对于 $j\in\Lambda_2$，$l\in N^c_j$，系数 $w(j,l)$的计算式如下：

$$w(j,l)=-\frac{S_{\gamma_x(j,l),\gamma_y(j,l)}(j)+\sum_{m\in N^c_j\cap N_l}S_{\gamma_x(j,l),\gamma_y(j,l)}(j)w(m,l)}{S_{0,0}(j)}$$

e_h 的计算可以表示成矩阵向量积的形式，即 $e_h=P^h_He_H$，其中矩阵 P^h_H 即为多重网格的延拓算子。

并行延拓算子 P^h_H 仍需要进一步校正，这是由计算区域的并行划分及拓展导致的。若 j 为人工边界上的网格点，则差分模板 $S(j)$的部分值为假值，从而导致以上两种延拓系数的计算出现偏差。随着迭代不断进行，这种偏差最终会导致并行预条件迭代失效。因此，P^h_H 对应人工边界及其邻接边界上的数据值需要校正更新。由于一个覆盖的人工边界处在相邻覆盖的内部区域，而相邻覆盖在此处对应的 P^h_H 数据值为真值，反之亦然。因此，可以通过 MPI 的通信函数 MPI_Isend 和 MPI_Ireceive 实现两个覆盖之间的数据交换，并计算更新人工边界及其邻近边界对应的延拓系数。

4.3　并行粗网格算子的构建

粗网格算子可以看作是移位拉普拉斯算子 M 在粗网格上的离散，记作 M_H。若令矩阵 M_h 和 R^H_h 分别表示已知的细网格算子和限制算子，则由 Galerkin 原则可得：

$$M_H=R^H_hM_hP^h_H$$

粗网格的计算可看作 3 个矩阵的乘积，计算过程较复杂，但每个矩阵数据都有自己的几何背景与稀疏结构，可以充分利用这些信息实现粗网格的高效计算。为此，首先在坐标轴方向对每个覆盖内的网格点进行编号。图 5 给出了 25 个细网格点和 9 个粗网格点的编号，其中网格外部和内部的数字分别为细网格点编号和粗网格点编号。记 N^x_h 和 N^y_h 分别表示 x,y 轴方向上的细网格点数，对应的细网格点编号集记为：

$$\Phi^\xi_h=\{0,1,\cdots,N^\xi_h\},\xi=x,y$$

类似地，定义 $N^\xi_H,\Phi^\xi_H,\xi=x,y$。细网格点和粗网格点的总数分别为 $N_h:=N^x_h\cdot N^y_h$，$N_H:=N^x_H\cdot N^y_H$，定义指标集 $\Phi_h=\{0,1,\cdots,N_h-1\},\Phi_H=\{0,1,\cdots,N_H-1\}$。可以看出，矩阵 R^H_h，M_h，P^h_H 的规模分别为 $N_H\times N_h$，$N_h\times N_h$，$N_h\times N_H$。

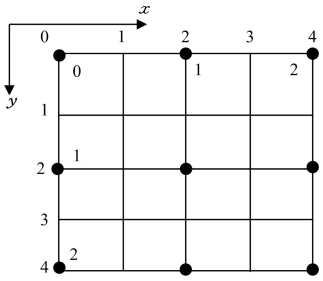


图 5　单个覆盖内的细网格和粗网格的标号

Fig. 5　Numbering of fine and coarse grid points in single covering

接着介绍矩阵数据 R^H_h,M_h,P^h_H 与几何背景相关的稀疏结构。为此，先定义一些符号。根据图 5，一个细网格点具有唯一的坐标值，记该网格点对应 x,y 轴的坐标分量分别为 i^x_h,i^y_h，则：

$$i_h=F(i^x_h,i^y_h):=i^x_h+i^y_h\cdot N^x_h$$

为细网格矩阵 M_h 的第 i_h 行标号（行列标号从 0 开始），$i_h\in\Phi_h$。反之，M_h 的第 j_h 行唯一对应一个坐标为(j^x_h,j^y_h)的细网格点。给定一个实数 ϑ，令$[\vartheta]$表示不大于 ϑ 的整数。对某个 $l_h\in\Phi_h$，定义函数 ν^x_h,ν^y_h 为：

$$\nu^y_h(l_h):=[l_h/N^x_h],\nu^x_h(l_h):=l_h-\nu^y_h(l_h)\cdot N^x_h$$

则 $j^y_h=\nu^y_h(j_h)$，$j^x_h=\nu^x_h(j_h)$。对应最细层网格（原始离散网格），有 $M_h=M$。根据 9 点差分离散方案和图 5 中的网格点标号，对于 $i_h,j_h\in\Phi_h$，有 $M_h(i_h,j_h)\neq 0$，当且仅当标号 j_h 对应的网格点是标号 i_h 对应的网格点的邻居。对于正整数 n，定义两个整数集合如下：

$$\Psi_1(n):=\{n-1,n,n+1\}$$

$$\Psi_2(n):=\{n-2,n-1,n,n+1,n+2\}$$

基于上述的整数集合定义两个与 i_h 相关的指标集：

$$K_1(i_h):=\{j_h:j_h=F(j^x_h,j^y_h),j^\xi_h\in\Phi^\xi_h\bigcap\Psi_1(\nu^\xi_h(i_h)),\xi=x,y\}$$

$$K_2(i_h):=\{j_h:j_h=F(j^x_h,j^y_h),j^\xi_h\in\Phi^\xi_h\bigcap\Psi_2(\nu^\xi_h(i_h)),\xi=x,y\}$$

那么对于行标记 i_h，仅当 $j_h\in K_1(i_h)$时，$M_h(i_h,j_h)\neq 0$；而对于列标记 j_h，仅当 $i_h\in K_1(j_h)$时，$M_h(i_h,j_h)\neq 0$。

延拓算子 R^H_h 和延拓算子 P^h_H 具有类似的稀疏结构。对于标号 $i_H\in\Phi_H$ 代表的粗网格点，其也是一个细网格点，其对应的细网格点标号记为 i_h，即 i_h 和 i_H 分别代表同一网格点在不同层次的标号。定义粗、细网格标号之间的映射为 η，即 $\eta(i_H)=i_h$。如果 i_H 对应的粗网格点在覆盖的边界上，则仅当 $j_h=\eta(i_H)$时，$R^H_h(i_H,j_h)\neq 0$。如果 i_H 对应的粗网格点在覆盖的内部区域，则仅当 $j_h\in K_1(\eta(i_H))$时，$R^H_h(i_H,j_h)\neq 0$。根据 4.2 节中延拓算子的构造可知，对于列标记 j_H，仅当其对应的粗网格点是 i_h 对应的细网格点的邻居时，即 $i_h\in K_1(\eta(j_H))$时，$P^h_H(i_h,j_H)\neq 0$。

明确了 R^H_h,M_h,P^h_H 的稀疏结构后，便可以高效地计算粗

网格算子 M_H。关于 M_H 的计算有以下命题。

命题 2 若 R_h^H 为全加权限制算子，P_H^h 为 4.2 节中构造的基于矩阵的延拓算子，M_h 为移位拉普拉斯算子 M 在细网格上通过 9 点差分离散得到的，对于行列标号 $i_H, j_H \in \Phi_H$，$k_h \in \Phi_h$，定义与它们相关的集合 G, L, Q 为：

$$G := K_1(\eta(i_H)) \cap K_1(k_h)$$

$$L := K_2(\eta(i_H)) \cap K_1(\eta(j_H))$$

$$Q := K_1(\eta(i_H)) \cap K_1(\eta(j_H))$$

则粗网格算子 M_H 的分量元素 $M_H(i_H, j_H)$ 的计算式如下。当 i_H 对应覆盖内部的网格点时，有：

$$M_H(i_H, j_H) = \sum_{k_h \in L} \sum_{l_h \in G} R_h^H(i_H, l_h) M_h(l_h, k_h) P_H^h(k_h, j_H)$$

当 i_H 对应覆盖边界上的网格点时，有：

$$M_H(i_H, j_H) = \sum_{k_h \in Q} M_h(\eta(i_H), k_h) P_H^h(k_h, j_H)$$

证明：计算 $M_H = R_h^H M_h P_H^h$ 时，先求解 $M_H^h := R_h^H M_h$，根据矩阵的乘法规则计算 M_H^h 的元素 $M_H^h(i_H, k_h)$，得：

$$M_H^h(i_H, k_h) = \sum_{l_h \in \Phi_h} R_h^H(i_H, l_h) M_h(l_h, k_h) \tag{6}$$

根据 R_h^H 和 M_h 的稀疏结构可知，当 i_H 对应边界点时，有 $M_H^h(i_H, k_h) = M_h(\eta(i_H), k_h)$。当 i_H 对应内部点时，式(6)中 l_h 的求和范围 Φ_h 可缩小到 G。进一步分析 M_H^h 的稀疏结构可知，当 i_H 对应边界点时，仅当 $k_h \in K_1(\eta(i_H))$ 时 $M_H^h(i_H, k_h)$ 取非零值；当 i_H 对应覆盖的内部点时，仅当 $k_h \in K_2(\eta(i_H))$ 时，$M_H^h(i_H, k_h)$ 取非零值。接着计算 $M_H = M_H^h P_H^h$，当 i_H 对应内部点时，据矩阵的乘法规则有：

$$\begin{aligned} M_H(i_H, k_h) &= \sum_{k_h \in \Phi_h} M_H^h(i_H, k_h) P_H^h(k_h, j_H) \\ &= \sum_{k_h \in L} \sum_{l_h \in G} R_h^H(i_H, l_h) M_h(l_h, k_h) P_H^h(k_h, j_H) \end{aligned}$$

当 i_H 对应边界点时，将式(6)简化为：

$$\begin{aligned} M_H(i_H, k_h) &= \sum_{k_h \in \Phi_h} M_H^h(i_H, k_h) P_H^h(k_h, j_H) \\ &= \sum_{k_h \in \Phi_h} M_h(\eta(i_H), k_h) P_H^h(k_h, j_H) \end{aligned} \tag{7}$$

而根据 M_h 和 P_H^h 的稀疏结构可知，$M_h(\eta(i_H), k_h)$ 和 $P_H^h(k_h, j_H)$ 仅当 $k_h \in Q$ 时取非零值，故式(7)可以转化为：

$$M_H(i_H, j_H) = \sum_{k_h \in Q} M_h(\eta(i_H), k_h) P_H^h(k_h, j_H)$$

同样，因为人工边界的问题，由命题 2 计算得到的 M_H 在并行计算过程中也需要进一步校正，与延拓算子类似，校正也是通过相邻覆盖对应进程之间的数据交换实现的。

本文并行预条件迭代求解的具体算法如下。

算法 1 基于多重网格的并行预条件 BI-CGSTAB

Step1 按照 3.1 节的内容进行计算区域(任务)划分。

Step2 按照 4.1 节中的离散算子方程式(3)，得到离散预条件系统式(4)和式(5)。记 $M_0 = M$，给定多重网格层数 L。

Step3 令 $l = 0$。

Step4 按照 3.2 节中的拓展计算区域(网格)，构建第 l 层网格的多重网格组件(限制算子 R_l、延拓算子 P_l 和网格算子 M_l)，其中，分别按照 4.2 节和 4.3 节中的内容构建 P_l 和 M_l。P_l 和 M_l 的计算需要用到 MPI_Isend，MPI_Ireceive 进行数据通信。

Step5 $l = l+1$，若 $l \leqslant L$，执行 Step4，否则继续执行 Step 6。

Step6 令 $i = 0$，给定初值 $u^{(0)}$，计算 $r^{(i)} = g - Au^{(i)}$。

Step7 $i = i+1$，给定任意值 $\tilde{r}$，计算 $\rho_{i-1} = \tilde{r} \cdot r^{(i-1)}$，如果 $\rho_{i-1} = 0$，重新选择初值 $u^{(0)}$ 或 $\tilde{r}$。ρ_{i-1} 的计算利用 MPI_AllReduce 进行规约数据通信。

Step8 如果 $i = 1$，则令 $p^{(i)} = r^{(i-1)}$；否则，计算 $\beta_{i-1} = \rho_{i-1}\alpha_{i-1}/(\rho_{i-2} w_{i-1})$，$p^{(i)} = r^{(i-1)} + \beta_{i-1}(p^{(i-1)} - w_{i-1} v^{(i-1)})$。

Step9 采用瀑布多重网格法 FMG(M, R_l, P_l, M_l)($l = 1, \cdots, L$)求解内部子线性系统 $M\tilde{p} = p^{(i)}$，并计算 $v^{(i)} = A\tilde{p}$，$\alpha_i = \rho_{i-1}/(\tilde{r}^T v^{(i)})$，$s = r^{(i-1)} - \alpha_{i-1} v^{(i-1)}$。其中，FMG 迭代利用 MPI_Isend，MPI_Ireceive 进行数据通信，参数 α_i 的计算需要用到利用 MPI_AllReduce 函数进行数据通信。

Step10 采用 FMG(M, R_l, P_l, M_l)求解 $M\tilde{s} = s$，并计算 $t = A\tilde{s}$，$w_i = t^T s/(t^T t)$，$u^{(i)} = u^{(i-1)} - \alpha_{i-1} v^{(i-1)}$。其中，FMG 迭代利用 MPI_Isend，MPI_Ireceive 进行数据通信，参数 w_i 的计算需要用到 MPI_AllReduce 进行数据通信。

Step11 计算 $r^{(i)} = s - w_i t$，如果 $\| r^{(i)} \| / \| r^{(0)} \| < \varepsilon$，则算法迭代终止，得到迭代至第 i 步的局部近似解 $u^{(i)}$；否则执行 Step7。其中，$\| \cdot \|$ 和 ε 分别表示欧几里得范数和计算精度。

在各子区域对应的进程(处理器)上同时执行算法 1，合并各个进程计算得到的局部近似解 $u^{(i)}$，即可得到方程的全局解。在算法的执行过程中，相邻子区域对应的进程间的数据通讯采用函数 MPI_Send 和 MPI_Receive 实现，而每个进程所用到的共同数据通过规约通信函数 MPI_AllReduce 得到。并行预条件迭代算法的执行流程如图 6 所示。

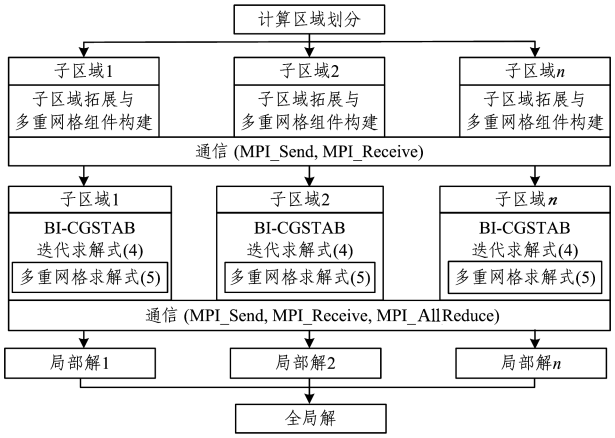


图 6 并行预条件迭代算法的执行流程

Fig. 6 Flowchart for implementation of parallel preconditioned Bi-CGSTAB method

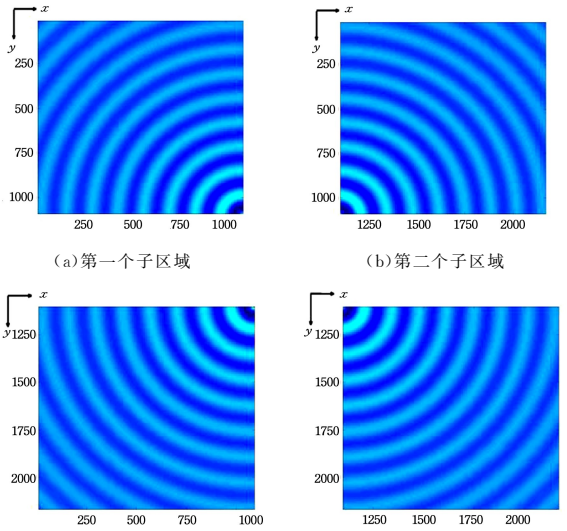
5 数值实验

本节测试了并行多重网格预条件 BI-CGSTAB 算法的性能。实验平台为中山大学“南方一号”曙光 Sugon Linux 计算集群，集群共有 54 个计算节点，每个节点有 2 个六核的 Intel Xeon E5620 CPU(工作频率 2.6 GHz)、6 个 NVIDIA C2050 GPU，以及 72 GB 的主内存。计算网络是 40Gbs 的 Infiniband。

在计算过程中，PML 的厚度为 20，即 PML 在每个方向含有 20 个网格点。多重网格迭代采用 1 次 FMG 循环，细网格上的光滑子采用点态的雅克比松弛(ω-JAC)。并行预条件

迭代的终止条件是近似解的相对残差欧氏范数小于10^{-6}。

首先，测试一个小波数情形($k=100$)，以验证并行预条件迭代结果的正确性。计算区域的规模为 2170 m×2170 m，点震源放置在区域中心，离散步长 $h=10$ m，网格规模为217^2(不含 PML 中的网格点)，计算区域划分为 4 个子区域，规模分别为 1090 m×1090 m，1080 m×1090 m，1090 m×1080 m 和 1080 m×1080 m。使用 4 个处理器(进程)进行并行计算，每个处理器负责一个子区域上的计算任务。图 7 给出了对应 4 个子区域上的计算结果(局部迭代解实部的可视化)，合并这 4 个局部解即可得到系统(3)的全局解，如图 8 所示。可见波在区域中心处开始传播，在边界处被吸收，未出现非物理性的反射。



(a)第一个子区域　(b)第二个子区域

(c)第三个子区域　(d)第四个子区域

图 7　波数为 100 时 4 个子区域上局部迭代解实部的可视化显示

Fig. 7　Visualization of local iterative solutions(real parts) for $k=100$ in four sub-domains

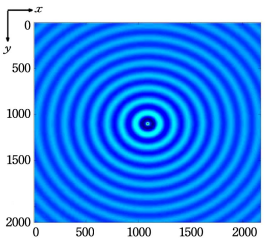


图 8　波数为 100 时全局迭代解实部的可视化

Fig. 8　Visualization of global iterative solution for $k=100$ in original domain

接着对大波数问题进行测试。测试的 3 个波数 k 分别取 5000，6500，8000，对应的计算自由度(网格点数、未知量个数)最多达16245^2(约 2.6 亿)，最少用到 2 个处理器，最多用到 256 个处理器。计算结果如表 1 和表 2 所列。当波数 k 取 5000，6500 时，串行算法(用 1 个处理器，对应 $p=1$)的计算时间分别为 95.4 min 和 281.8 min，而用 256 个处理器并行计算的耗时分别为 2.95 min 和 4.18 min，分别节省了 94.4% 和 98.5%。当 k 取 8000 时，由于计算规模大，单个结点的内存无法满足计算需求，因此使用 2 个节点上的 2 个处理器进行并行计算，耗时 402.6 min；而用 256 个处理器(分布在 22 个节点上)并行计算的耗时为 8.62 min，节省了 97.9% 的时间。可见，并行算法极大地提高了计算效率。

表 1　大波数情形预条件迭代的耗时($p=1,2,4,8,16$)

Table 1　Computational time of parallel preconditioned Bi-CGSTAB method with large wavenumbers ($p=1,2,4,8,16$)

(单位：min)

k	自由度	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$
5000	8153^2	95.42	52.43	29.13	16.64	9.35
6500	12249^2	281.8	149.9	81.02	45.52	25.01
8000	16245^2	—	402.6	211.9	115.2	61.25

表 2　大波数情形预条件迭代的耗时($p=32,64,128,256$)

Table 2　Computational time of parallel preconditioned Bi-CGSTA B method with large wavenumbers ($p=32,64,128,256$)

(单位：min)

k	自由度	$p=32$	$p=64$	$p=128$	$p=256$
5000	8153^2	5.44	3.26	2.18	2.95
6500	12249^2	14.4	8.46	5.35	4.18
8000	16245^2	34.0	19.56	11.6	8.62

令 $t(p)$表示用 p 个处理器并行计算的耗时，定义并行加速比函数 $Spd(p)$和并行效率函数 $Efc(p)$分别为：

$$Spd(p):=\frac{t(2)}{t(p)},p=2,4,8,\cdots,256 \tag{8}$$

$$Efc(p):=\frac{2Spd(p)}{p},p=2,4,8,\cdots,256 \tag{9}$$

这里采用两个处理器的并行计算时间作为基准，而未采用串行(一个处理器)计算时间，原因在于 k 取 8000 时的计算规模太大，单个集群单个计算节点的内存无法承受相应的大量数据。为了比较 3 个大波数对应的并行加速比，统一采用两个处理器的计算时间作为加速比的计算基准。根据表 1 和表 2 的数据，通过式(8)和式(9)的计算，即可得到并行加速比和并行效率。图 9 给出了 k 取 5000，6500，8000 时的并行加速比曲线，图 10 给出了对应的并行效率曲线。

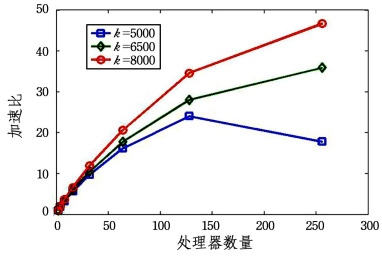


图 9　并行加速比曲线

Fig. 9　Speedup of parallel preconditioned Bi-CGSTAB method

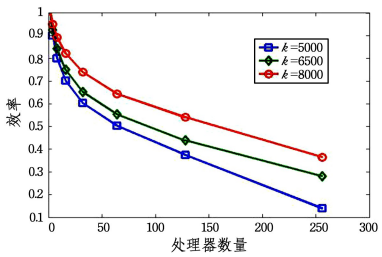


图 10　并行效率曲线

Fig. 10　Efficiency of parallel preconditioned Bi-CGSTAB method

可以看出，$k=5000$ 时的加速比曲线在 $p=128$ 处出现了

拐点，这是因为并行预条件迭代的耗时分为两部分：计算开销和通信开销。当用到 128 个处理器时，通信开销占据了总耗时的大部分，因此加速比下滑。然而，随着波数和计算规模的增加，加速比曲线却逐渐得到改善，即当利用相同数量的处理器进行计算时，$k=6500$ 时对应的加速比 $k=5000$ 的更高，而 $k=8000$ 对应的加速比比 $k=6500$ 的更高。同样，并行效率曲线也是随着波数和计算规模的增加呈增加趋势。最后，将本文算法与文献[16]中的算法进行比较，分别用这两种方法来测试具有广泛应用的变波数 Helmholtz 问题（基于 Marmousi 模型），计算自由度为 6000m×1600m，单个波长内至少包含 7.5 个网格点。由于不同平台计算时间的差异性，因此比较两种并行方法的并行加速比（以 1 个处理器的计算时间为基准），结果如表 3 所列。可见，本文算法具有较高的并行加速比。

表 3 本文算法与文献[16]中算法的并行加速比的对比

Table 3 Comparison of parallel speedup of proposed method and method in reference [16]

	$p=1$	$p=2$	$p=4$	$p=8$	$p=16$
文献[15]中的算法	—	1.61	2.98	5.59	7.38
本文算法	—	1.78	3.30	5.85	8.13

结束语 Helmholtz 方程在很多科学和技术领域有着重要的应用，建立其高效的数值模拟方法一直是计算科学的研究热点。本文提出一种求解 Helmholtz 方程的 MPI 并行预条件 BI-CGSTAB 迭代算法，采用基于复移位拉普拉斯算子的预条件技术，利用基于矩阵的多重网格来逼近预条件子的近似逆。在设计并行算法时，重点解决了具有多尺度结构的多重网格的并行计算。通过计算区域的并行划分与合理拓展、并行多重网格组件的构建等来保障数据信息的纵横向传递与交换，使并行计算顺利执行。数值实验结果显示了并行预条件迭代法的高效性和良好的并行加速比。

参 考 文 献

[1] IHLENBURG F, BABUŠKA I. Finite Element Solution of the Helmholtz Equation with High Wave Number, Part I: The h-version of the FEM[J]. Compute & Mathematics with Applications, 1995, 30(9): 9-37.

[2] ERLANGGA Y, OOSTERLEE C, VUIK C. A Novel Multigrid Based Preconditioner for Heterogeneous Helmholtz Problem [J]. SIAM Journal on Scientific Computing, 2006, 27(4): 1471-1492.

[3] CHENG D S, LIU Z Y, WU T T. A Multigrid Based Preconditioned Solver for the Helmholtz Equation with a Discretization By 25-point Difference Scheme[J]. Mathematics and Computers in Simulation, 2015, 117(11): 54-67.

[4] KE R H, LI W. Numerical Method for the Two-dimensional Helmholtz Equation Discretized By CCD Scheme[J]. Journal on Numerical Method and Computer Application, 2013, 34(3): 221-230. (in Chinese)
柯日焕，黎稳. 用 CCD 法离散求解二维 Helmholtz 方程的数值方法[J]. 数值计算与计算机应用，2013，34(3)：221-230.

[5] CAO J, CHEN J B, CAO S H. Studies on Iterative Algorithms for Modeling of Frequency-domain Wave Equation Based on Multi-grid Precondition [J]. Chinese Journal of Geophysics, 2015, 58(3): 1002-1012. (in Chinese)
曹健，陈景波，曹书红. 频率域波动方程正演基于多重网格预条件的迭代算法研究[J]. 地球物理学报，2015，58(3)：1002-1012.

[6] 张林波，迟学斌，莫则尧，等. 并行计算导论[M]. 北京：清华大学出版社，2006：1-520.

[7] COOK S. CUDA Programming: A Developer's Guide to Parallel Computing with GPUs[M]. San Francisco: Morgan Kaufmann publishers inc, 2012: 1-600.

[8] RIYANTI C D, KONONOV A, ERLANGGA Y A, et al. A parallel multigrid-based preconditioner for the 3D heterogeneous high-frequency Helmholtz equation[J]. Journal of Computational Physics, 2007, 224(1): 431-448.

[9] ZHANG L L, GONG X P, SONG J Q. Parallel Preconditioned GMRES Solvers for 3-D Helmholtz Equations in Regional Non-hydrostatic Atmosphere Model[C]// International Conference on Computer Science and Software Engineering. Washington, DC: IEEE Computer Society, 2008: 287-290.

[10] KNIBBE H, OOSTERLEE C W, VUIK C. GPU implementation of a Helmholtz Krylov solver preconditioned by a shifted Laplace multigrid method[J]. Journal of Computational and Applied Mathematics, 2011, 236(3): 281-293.

[11] LENG W. A Fast Propagation Method for the Helmholtz Equation[J]. Chinese Journal of Engineering Mathematics, 2015, 32(5): 726-742.

[12] CHEN Z Y, CHENG D S, WU T T. An Optimal 9-point Finite Difference Scheme for the Helmholtz Equation with PML[J]. International Journal of Numerical Analysis and Modeling, 2013, 10(2): 389-410.

[13] LI X M, WU J P. Krylov Subapce Methods and Parallel Computation[J]. Computer Science, 2005, 32(1): 19-20. (in Chinese)
李晓梅，吴建平. Krylov 子空间方法及其并行计算[J]. 计算机科学，2005，32(1)：19-20.

[14] BRIGGS W L, HENSON V E, MCCORMICK S F. A Multigrid Tutorial[M]. California: SIAM, 2000: 1-192.

[15] BÉRENGER J P. A Perfectly Matched Layer for the Absorption of Electromagnetic Waves[J]. Journal of Computational Physics, 1994, 114(2): 185-200.

[16] GORDON D, GORDON R. Robust and highly scalable parallel solution of the Helmholtz equation with large wave numbers [J]. Journal of Computational & Applied Mathematics, 2013, 237(1): 182-196.