Advances in Applied Mathematics 应用数学进展, 2017, 6(2), 114-126
Published Online March 2017 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2017.62013

Hans汉斯

A Finite Difference Parallel Scheme Based on MPI Implementation for Fourth Order Parabolic Equations

Yuyang Gao, Haiming Gu

Qingdao University of Science and Technology, Qingdao Shandong
Email: crazy3435@163.com, gum@qust.edu.cn

Received: Feb. 27th, 2017; accepted: Mar. 18th, 2017; published: Mar. 21st, 2017

Abstract

Parallel computing can save a lot of time in the field of large-scale scientific computing. In this paper, the main idea is that a finite difference parallel scheme for fourth order parabolic equations. The scheme is constructed by Saul'yev asymmetric difference schemes which called the four-point scheme. It's one explicit difference scheme, the computational domain can be divided into a number of large areas; each sub-region computes themselves, and the parallel scheme is unconditionally stable. Then, the paper focuses on the numerical calculation of the four-point scheme in MPI parallel environment. Two different MPI parallel algorithms are constructed, one is blocking communication (wait communication) mode, and the other is non-blocking communication (non-wait communication) mode. These two parallel algorithms both better than serial algorithm to calculate numerical solutions use four-point scheme, and the non-blocking communication mode is higher computational than the other, because the wait time in non-blocking communication mode is less than blocking communication mode.

Keywords

Fourth Order Parabolic Equations, Finite Difference Parallel Scheme, MPI (Message Passing Interface)

基于MPI的一种有限并行差分格式求解四阶抛物方程

高玉羊，顾海明

青岛科技大学，山东 青岛
Email: crazy3435@163.com, gum@qust.edu.cn

文章引用: 高玉羊, 顾海明. 基于 MPI 的一种有限并行差分格式求解四阶抛物方程[J]. 应用数学进展, 2017, 6(2): 114-126. https://doi.org/10.12677/aam.2017.62013

收稿日期：2017年2月27日；录用日期：2017年3月18日；发布日期：2017年3月21日

摘　要

在大规模的科学与工程计算问题中，并行计算能够节省大量的时间，本文针对一维四阶抛物方程给出了一类并行差分格式。利用Saul'yev非对称格式进行恰当的组合，形成求解抛物方程的四点格式。四点格式是显式求解的，因此可以将空间区域分为若干子区域，每个子区域独立计算。验证分析表明，该格式是绝对稳定的。随后本文着重介绍了在MPI并行环境下对该格式进行数值计算，构建了两种不同的MPI并行算法并与串行状态下的有限差分格式做出比较，即阻塞通信(等待通信)和非阻塞通信(非等待通信)模式。相对于串行算法运用四点格式求解四阶抛物方程，两种并行通信模式都表现出极好的效果，而且，非阻塞通信模式下的计算由于相对减少了一部分数据的通信等待时间，使得相对于阻塞通信模式，非阻塞通信模式表现出较好的并行效率。

关键词

四阶抛物方程，有限并行差分法，Message Passing Interface

1. 引言

在自然科学和工程技术领域中，尤其是以物理学、化学、生物学等为主的学科引领了现代科学技术的快速发展，而这些学科自身的精确化又是他们取得进展的重要保证。其中，很多问题可以描述为偏微分方程或方程组的形式，因此，运用有限差分方法数值求解偏微分方程问题具有很多重要的理论意义与应用价值。

随着科学计算问题的规模不断扩大，并行计算越来越受到人们的重视，其中，高性能并行计算机的快速发展也促进了偏微分方程并行数值算法的研究。MPI 是当今最重要的并行编程工具之一，MPI 是一个信息传递应用程序接口，包括协议和语义说明，MPI 的目标是高性能，大规模性，和可移植性。大部分的 MPI 实现由一些指定惯例集(API)组成，可由 C，C++，Fortran，或者有此类库的语言比如 C#, Java or Python 直接调用。MPI 优于老式信息传递库是因为他的可移植性和速度。MPI 为并行算法提供了多种通信模式，其中，一般的阻塞通讯(等待通讯)模式即可满足大多数并行算法，非阻塞通讯(非等待通讯)相对减少了各并行模块之间的通讯时间，从而使得并行计算模式的效率得到提升。

关于求解二阶抛物偏微分方程的并行差分格式，已经有了很多的研究(文献[1])，D. J. Evans 和 A. R. Abdullah 在文献[2]中巧妙的利用 Saul'yev 非对称格式(文献[3])构造了交替分组显式(AGE)方法，以及在近年来，Evans 等陆续基于 AGE 方法经过变换扩展到其他方面(见文献[4]，[5]，[6])。在文献[7]中，作者运用分组显式的方法求解了四阶椭圆方程，并应用于 MPI 并行计算。文献[8]，[9]中，作者改进了 AGE 方法，得到交替分段方法，使数值结果更加精确。实际上，四阶抛物偏微分方程相对于二阶方程的求解有更大的困难，一般的显式格式的稳定性条件比二阶抛物方程更加苛刻，一般四阶抛物方程的隐格式形成的代数方程组的系数矩阵比二阶方程占据更大运算空间，因此，四阶抛物方程在数值求解的规模上要

远远大于二阶方程。

本文在探究四阶抛物方程的并行有限差分格式时，主要借鉴于求解二阶抛物方程的差分方法，运用不同类型的 Saul’yev 非对称格式进行恰当的组合，对少部分点的求解形成代数方程组，从而构造出方程的显式差分格式，由于显式格式不必要求规模庞大的代数方程组，因而可以适用于并行计算。单独使用一种非对称差分格式会使误差偏上或偏下(文献[10])，因此在时间层上交替的使用不同的非对称格式，可以使每层产生的误差抵消，从而使方程数值解的精度得到提高。

2. 四阶抛物方程的非对称差分格式

考虑如下四阶抛物方程以及初边值问题，

$$\frac{\partial u}{\partial t}+\frac{\partial^4 u}{\partial x^4}=0,\ x\in(0,\pi),\ t\in(0,T] \tag{1}$$

$$u(x,0)=u^0(x),\ x\in(0,\pi) \tag{2}$$

$$u(0,t)=\frac{\partial^2 u(0,t)}{\partial x^2}=u(\pi,t)=\frac{\partial^2 u(\pi,t)}{\partial x^2}=0 \tag{3}$$

建立差分格式之前，不妨设 $U(x,t)$ 为精确解。对求解区域 $(0,\pi)\times(0,T]$ 进行网格剖分，空间步长为 h，时间步长为 Δt。$h=\frac{\pi}{N}$，$\Delta t=\frac{T}{M}$，其中 N，M 是自然数。

$$x_i=i\cdot h(i=0,1,\cdots,N)$$

$$t^n=n\cdot\Delta t(n=0,1,\cdots,M)$$

将求解区域分割成矩形网格，网格节点为 (x_i,t_n)，为方便起见，记 $U_i^n=U(x_i,t_n)$ 为节点处的值。将 $u_i^n=u(x_i,t_n)$ 记为初边值问题(1)，(2)，(3)在网格节点 (x_i,t_n) 的数值解，$r=\frac{\Delta t}{h^4}$ 为网格比。

2.1. 四阶抛物方程的显格式

易知，

$$\frac{\partial U_i^n}{\partial t}\approx\frac{U_i^{n+1}-U_i^n}{\Delta t}$$

$$\frac{\partial^2 U_i^n}{\partial x^2}\approx\frac{U_{i+1}^n-2U_i^n+U_{i-1}^n}{h^2}$$

方程(1)可以写成下面的形式

$$\frac{U_i^{n+1}-U_i^n}{\Delta t}+\frac{1}{h^2}\left[\frac{\partial^2 U_{i+1}^n}{\partial x^2}-2\frac{\partial^2 U_i^n}{\partial x^2}+\frac{\partial^2 U_{i-1}^n}{\partial x^2}\right]+O(\Delta t+h^2)=0 \tag{4}$$

将二阶差商化为一阶得

$$\frac{U_i^{n+1}-U_i^n}{\Delta t}+\frac{1}{h^2}\left[\frac{U_{i+2}^n-2U_{i+1}^n+U_i^n}{h^2}-2\frac{U_{i+1}^n-2U_i^n+U_{i-1}^n}{h^2}+\frac{U_i^n-2U_{i-1}^n+U_{i-2}^n}{h^2}\right]$$
$$+O(\Delta t+h^2)=0$$

用 u_i^n 代替 U_i^n，舍弃无穷小项化简后可以得到

$$\begin{aligned} & u_i^{n+1} = -ru_{i+2}^n + 4ru_{i+1}^n + (1-6r)u_i^n + 4ru_{i-1}^n - ru_{i-2}^n \\ & \qquad\qquad i = 1,2,\cdots,N-1, \quad n = 1,2,\cdots,M \\ & u_i^0 = u^0(x_i), \qquad\qquad i = 0,1,\cdots,N \\ & u_0^n = u_{-1}^n + u_1^n = u_N^n = u_{N-1}^n + u_{N+1}^n = 0,\ n = 1,2,\cdots,M \end{aligned} \tag{5}$$

运用 Fourier 方法可以证明四阶抛物方程的显式格式是条件稳定的，此处不再赘述，稳定性条件是 $r \le \frac{1}{8}$，截断误差为 $O(\Delta t + h^2)$。

2.2. 四阶抛物方程的隐格式

易知，

$$\frac{\partial U_i^{n+1}}{\partial t} \approx \frac{U_i^{n+1} - U_i^n}{\Delta t}$$
$$\frac{\partial^2 U_i^{n+1}}{\partial x^2} \approx \frac{U_{i+1}^{n+1} - 2U_i^{n+1} + U_{i-1}^{n+1}}{h^2}$$

方程(1)可以写成如下形式

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \frac{1}{h^2}\left[\frac{\partial^2 U_{i+1}^{n+1}}{\partial x^2} - 2\frac{\partial^2 U_i^{n+1}}{\partial x^2} + \frac{\partial^2 U_{i-1}^{n+1}}{\partial x^2}\right] + O(\Delta t + h^2) = 0 \tag{6}$$

舍弃无穷小，用 u_i^n 代替 U_i^n，化简后得到

$$\begin{aligned} & ru_{i+2}^{n+1} - 4ru_{i+1}^{n+1} + (1+6r)u_i^{n+1} - 4ru_{i-1}^{n+1} + ru_{i-2}^{n+1} = u_i^n \\ & \qquad\qquad i = 1,2,\cdots,N-1, n = 1,2,\cdots,M \\ & u_i^0 = u^0(x_i) \qquad\qquad i = 0,1,\cdots,N \\ & u_0^n = u_{-1}^n + u_1^n = u_N^n = u_{N-1}^n + u_{N+1}^n = 0 \qquad n = 1,2,\cdots,M \end{aligned} \tag{7}$$

可以证明四阶方程的隐式格式是绝对稳定的，且局部截断误差为 $O(\Delta t + h^2)$。

2.3. 四阶抛物方程的非对称格式

由中值定理可以得到，

$$\frac{\partial^2 U_{i+1}^n}{\partial x^2} = \frac{\partial^2 U_{i+1}^{n+1}}{\partial x^2} - \Delta t \frac{\partial^3 U_{i+1}^{(n+\theta_1)\Delta t}}{\partial x^2 \partial t} \tag{8}$$

$$\frac{\partial^2 U_{i+1}^{n+1}}{\partial x^2} = \frac{\partial^2 U_{i+1}^n}{\partial x^2} + \Delta t \frac{\partial^3 U_{i+1}^{(n+\theta_2)\Delta t}}{\partial x^2 \partial t} \tag{9}$$

$$\frac{\partial^2 U_{i-1}^n}{\partial x^2} = \frac{\partial^2 U_{i-1}^{n+1}}{\partial x^2} - \Delta t \frac{\partial^3 U_{i-1}^{(n+\theta_3)\Delta t}}{\partial x^2 \partial t} \tag{10}$$

$$\frac{\partial^2 U_{i-1}^n}{\partial x^2} = \frac{\partial^2 U_{i-1}^{n+1}}{\partial x^2} - \Delta t \frac{\partial^3 U_{i-1}^{(n+\theta_3)\Delta t}}{\partial x^2 \partial t} \tag{11}$$

其中，$0 \le \theta_i \le 1$，$i = 1,2,3,4$。

类似于二阶抛物方程的非对称差分格式的构造(文献[3])，尝试构造四阶抛物方程的非对称差分格式，将(8)代入方程(4)，可以得到

$$\frac{U_i^{n+1}-U_i^n}{\Delta t}+\frac{1}{h^4}\Big[U_{i+2}^{n+1}-2U_{i+1}^{n+1}+U_i^{n+1}-2U_{i+1}^n+4U_i^n-2U_{i-1}^n+U_i^n \\ -2U_{i-1}^n+U_{i-2}^n-\Delta t\frac{\partial^3 U_{i+1}^{(n+\theta_1)\Delta t}}{\partial x^2\partial t}\Big]+O\left(\Delta t+h^2\right)=0 \tag{12}$$

舍去无穷小项 $O\left(\Delta t+h^2\right)$ 和 $-\frac{\Delta t}{h^2}\frac{\partial^3 U_{i+1}^{(n+\theta_1)\Delta t}}{\partial x^2\partial t}$，得到网格方程

$$ru_{i+2}^{n+1}-2ru_{i+1}^{n+1}+(1+r)u_i^{n+1}=2ru_{i+1}^n+(1-5r)u_i^n+4ru_{i-1}^n-ru_{i-2}^n \tag{13}$$

即得到四阶抛物方程的一个非对称差分格式。

运用同样的方法，分别将(9)，(10)，(11)代入等式(6)，(4)，(6)，可以得到

$$ru_{i-2}^{n+1}-4ru_{i-1}^{n+1}+(1+5r)u_i^{n+1}-2ru_{i+1}^{n+1}=(1-r)u_i^n+2ru_{i+1}^n-ru_{i+2}^n \tag{14}$$

$$ru_{i-2}^{n+1}-2ru_{i-1}^{n+1}+(1+r)u_i^{n+1}=2ru_{i-1}^n+(1-5r)u_i^n+4ru_{i+1}^n-ru_{i+2}^n \tag{15}$$

$$\begin{gathered} ru_{i+2}^{n+1}-4ru_{i+1}^{n+1}+(1+5r)u_i^{n+1}-2ru_{i-1}^{n+1}=(1-r)u_i^n+2ru_{i-1}^n-ru_{i-2}^n \\ i=1,2,\cdots,N-1,\ n=1,2,\cdots,M \\ u_i^0=u^0(x_i) \qquad i=0,1,\cdots,N \\ u_0^n=u_{-1}^n+u_1^n=u_N^n=u_{N-1}^n+u_{N+1}^n=0\ \ n=1,2,\cdots,M \end{gathered} \tag{16}$$

上述四个非对称网格方程的局部截断误差为 $O\left(\frac{\Delta t}{h^2}+\Delta t+h^2\right)$。

利用方程(13)~(16)可以构造出求解方程(1)的显式四点格式，即将其联立，可以得到一组差分方程

$$\begin{cases} ru_{i+1}^{n+1}-2ru_i^{n+1}+(1+r)u_{i-1}^{n+1}=2ru_i^n+(1-5r)u_{i-1}^n+4ru_{i-2}^n-ru_{i-3}^n \\ ru_{i+2}^{n+1}-4ru_{i+1}^{n+1}+(1+5r)u_i^{n+1}-2ru_{i-1}^{n+1}=(1-r)u_i^n+2ru_{i-1}^n-ru_{i-2}^n \\ ru_{i-1}^{n+1}-4ru_i^{n+1}+(1+5r)u_{i+1}^{n+1}-2ru_{i+2}^{n+1}=(1-r)u_{i+1}^n+2ru_{i+2}^n-ru_{i+3}^n \\ ru_i^{n+1}-2ru_{i+1}^{n+1}+(1+r)u_{i+2}^{n+1}=2ru_{i+1}^n+(1-5r)u_{i+2}^n+4ru_{i+3}^n-ru_{i+4}^n \end{cases} \tag{17}$$

四点格式所涉及到的网格节点如图 1 所示。

可以将方程组(17)写成矩阵的形式，如下

$$\begin{bmatrix} 1+r & -2r & r & 0 \\ -2r & 1+5r & -4r & r \\ r & -4r & 1+5r & -2r \\ 0 & r & -2r & 1+r \end{bmatrix}\begin{bmatrix} u_{i-1}^{n+1} \\ u_i^{n+1} \\ u_{i+1}^{n+1} \\ u_{i+2}^{n+1} \end{bmatrix} \\ =\begin{bmatrix} 1-5r & 2r & 0 & 0 \\ 2r & 1-r & 0 & 0 \\ 0 & 0 & 1-r & 2r \\ 0 & 0 & 2r & 1-5r \end{bmatrix}\begin{bmatrix} u_{i-1}^n \\ u_i^n \\ u_{i+1}^n \\ u_{i+2}^n \end{bmatrix}+\begin{bmatrix} -ru_{i-3}^n+4ru_{i-2}^n \\ -ru_{i-2}^n \\ -ru_{i+3}^n \\ -ru_{i+4}^n+4ru_{i+3}^n \end{bmatrix} \tag{18}$$

为了求解初边值问题(1)，(2)，(3)，假设网格的剖分节点是 $N=4m+3$（m 是自然数)，则内部节点数为 $N=4m+2$。因此共有 $4m+2$ 个需要计算的内点，共有 m 个四点组使用方程组(17)计算，可以明显的看到，还有两个单独的内点无法使用四点方法计算，可以考虑将两个内点放在计算区域的最左端或者最右端，使用方程(13)和(16)处理右边界临近的内点，方程(14)，(15)处理左端边界临近的内点，这两种不同的方法与四点方法组合可以得到一种新的方法。

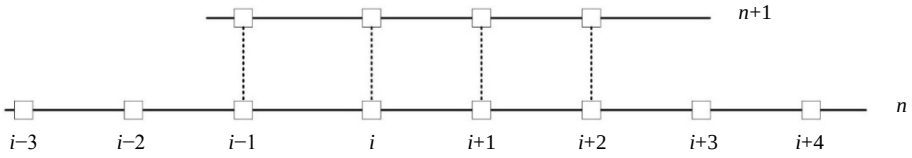

Figure 1. Points of four-point method
图 1. 四点格式涉及节点

考虑右边界的两点格式，

$$\begin{cases} ru_N^{n+1}-2ru_{N-1}^{n+1}+(1+r)u_{N-2}^{n+1}=2ru_{N-1}^{n}+(1-5r)u_{N-2}^{n}+4ru_{N-3}^{n}-ru_{N-4}^{n} \\ ru_{N+1}^{n+1}-4ru_N^{n+1}+(1+5r)u_{N-1}^{n+1}-2ru_{N-2}^{n+1}=(1-r)u_{N-1}^{n}+2ru_{N-2}^{n}-ru_{N-3}^{n} \end{cases} \tag{19}$$

将初边值条件，即 $u_N^n=0$， $u_{N-1}^n+u_{N+1}^n=0$ 代入上式，可得

$$\begin{cases} -2ru_{N-1}^{n+1}+(1+r)u_{N-2}^{n+1}=2ru_{N-1}^{n}+(1-5r)u_{N-2}^{n}+4ru_{N-3}^{n}-ru_{N-4}^{n} \\ (1+4r)u_{N-1}^{n+1}-2ru_{N-2}^{n+1}=(1-r)u_{N-1}^{n}+2ru_{N-2}^{n}-ru_{N-3}^{n} \end{cases} \tag{20}$$

同理，考虑左边界的两点格式，得到方程组

$$\begin{cases} -2ru_1^{n+1}+(1+r)u_2^{n+1}=2ru_1^{n}+(1-5r)u_2^{n}+4ru_3^{n}-ru_4^{n} \\ (1+4r)u_1^{n+1}-2ru_2^{n+1}=(1-r)u_1^{n}+2ru_2^{n}-ru_3^{n} \end{cases} \tag{21}$$

2.4. GER 和 GEL 格式

当临近右边界的两点采用两点格式(20)，其余内点采用四点格式，即可得到右端分组显式(GER)格式。写成矩阵的形式为

$$\begin{gathered} (I+rG_1)u^{n+1}=(I-rG_2)u^n \\ u^n=\left(u_1^n,u_2^n,\cdots,u_{N-1}^n\right)^{-1} \end{gathered} \tag{22}$$

$$G_1=\begin{bmatrix} G_{11} & & & \\ & G_{11} & & \\ & & \ddots & \\ & & & G_{12} \end{bmatrix},\ G_2=\begin{bmatrix} G_{21} & & & \\ & G_{11} & & \\ & & \ddots & \\ & & & G_{11} \end{bmatrix}$$

$$G_{11}=\begin{bmatrix} 1 & -2 & 1 & 0 \\ -2 & 5 & -4 & 1 \\ 1 & -4 & 5 & -2 \\ 0 & 1 & -2 & 1 \end{bmatrix},\ G_{12}=\begin{bmatrix} 1 & -2 \\ -2 & 4 \end{bmatrix},\ G_{21}=\begin{bmatrix} 4 & -2 \\ -2 & 1 \end{bmatrix}$$

与 GER 格式相似，在临近左边界的两个内点采用方程(21)，其余内点可以分成 m 个四点组，使用方程组(17)计算，可以得到左端分组显式(GEL)格式，方程如下，

$$(I+rG_2)u^{n+1}=(I-rG_1)u^n \tag{23}$$

下面对 GER 格式进行稳定性分析。

Kellogg 引理(文献[10])：设 $p\geq 0$，如果 G 为非负定实阵，即满足 $G+G^{\mathrm{T}}$ 非负定，则 $(\rho I+G)^{-1}$ 存在，且 $\left\|(\rho I+G)^{-1}\right\|_2\leq 1$。

在上述引理条件下，可以得到$\left\|(\rho I-G)(\rho I+G)^{-1}\right\|_2 \le 1$。

方程(22)，(23)可以写成$u^{n+1}=Tu^n$的形式，$T=(I+rG_1)^{-1}(I-rG_2)$。因G_1为非负定矩阵，可以得到$\left\|(I+rG_1)^{-1}\right\|_2 \le 1$。再由$G_2$的对称性以及特征值与范数的关系可以得到，

$$\left\|(I-rG_2)\right\|_2 \le \max\{|1-2r|,|1-5r|,|1-10r|,1\}$$

所以，只要$r \le \frac{1}{5}$，就能得到$\|T\|_2 \le 1$。因此，GER 或者 GEL 格式是条件稳定的，稳定条件为$r \le \frac{1}{5}$。

条件稳定的局限性并不能满足高效的求解微分方程，因此，在下面给出了分层交替方法，使稳定性得到本质的改善。

2.5. 分组交替方法(AGE 方法)

考虑上述的 GER 和 GEL 格式，在不同的时间层上分别交替的使用两种格式，写成方程组的形式为

$$\begin{cases}(I+rG_1)u^{n+1}=(I-rG_2)u^n \\ (I+rG_2)u^{n+2}=(I-rG_1)u^{n+1}\end{cases} \quad n=0,2,4,\cdots \tag{24}$$

其中G_1，G_2已在上文中定义，增长矩阵为

$$T=(I+rG_2)^{-1}(I-rG_1)(I+rG_1)^{-1}(I-rG_2)$$

利用 Kellogg 引理，容易得到

$$\begin{aligned}\left\|T^n\right\|_2 &\le \left\|(I+rG_2)^{-1}\right\|_2 \left\|(I-rG_1)(I+rG_1)^{-1}\right\|_2^n \left\|(I-rG_2)(I+rG_2)^{-1}\right\|_2^{n-1} \left\|(I-rG_2)\right\|_2 \\ &\le \left\|(I-rG_2)\right\|_2 \le 1+10r\end{aligned}$$

即$\left\|T^n\right\|_2 \le c$，$c=1+10r$，即可以得到，四阶抛物方程的分层交替方法是绝对稳定的。

3. MPI 并行算法

在本文中，所涉及到的并行算法都是基于上文中给出分组交替方法，GER 方法，GEL 方法实现的，在并行计算之前，首先，将求解区域$[0,\pi]$划分为若干大的区域，例如，若将抛物方程(1)进行 4 进程并行计算，则将求解区域分割成四条带状的区域，每个进程负责一条带状计算区域，并在时间层上逐次计算。为了实现高效计算，比较适宜的做法是为每一条进程分配大约相等的任务量，即保证在空间方向上分割的带状区域大小几乎是一致的。在进行每一时间层上的计算时，每两两相邻进程之间进行数据通信，当前时间层上的通信与计算完毕后，即进入下一时间层的计算。下文中，给出了两种不同的通信模式，即阻塞通信与非阻塞通信。一般情况下，这两种通讯模式即可满足大多数的并行计算任务。然后针对上文中论述的多种显示差分方法构造了相适应的并行处理框架，随后对其进行了并行运算。阻塞通信模式流程如表 1 所示。

考虑问题(1) (2) (3)，如上文，网格划分后，单一时间层上的内部节点数为$4m+2$ (即$N=4m+3$)，通过合适的划分网格，可以选择不同数目的进程进行计算，不失一般性，对上述三种差分格式考虑在 4 条进程下完成计算。首先，将整个求解区域大约的划分为四个部分，考虑 GEL 格式，在进程 0 中，前两个内部节点使用方程组(21)计算，进程 0 的其他内点以及其他进程的内部节点使用方程(17)计算。由于各相邻进程之间需要数据通信，此处以 0 号进程与 1 号进程之间的通信传递说明，不妨假设已知第n层上的数据，求解第$n+1$层上的数据，n层上的动作：进程 0 右端的两个点的数据发送至进程 1，进程 1 左

Table 1. The step of blocking communication mode
表 1. 阻塞通信模式下并行详细步骤

	进程 0	其他进程
步骤 1	获得本进程的所负责的计算区间	同 0 进程
步骤 2	读入本进程的计算数据	同 0 进程
步骤 3	发送数据到其他进程或从其他进程接收数据	同 0 进程
步骤 4	将数据带入差分格式并进行计算	同 0 进程
步骤 5	进入下一时间层的计算，转到步骤 3	同 0 进程

端的两内点数据发送至进程 0。$n+1$层上的动作：进程 0 和进程 1 接收来自对方的数据，并运用于计算。其他进程之间的通讯也是如此，通信模式简图见图 2。

在使用 GER 格式计算的情况下，与 GEL 格式类似，两点组的内部节点由最后一个进程计算，即最右端进程的最右端的两个内部节点由方程组(21)计算，其他内部节点由四点方法进行计算。与 GEL 格式一致，相邻进程的附近节点进行数据通信，见图 3。

在使用分组交替格式计算的情况下，与 GER，GEL 格式有些不同之处，首先考虑在时间层第 n 层使用 GEL 格式，第 $n+1$层使用 GER 格式，第 n 层上的通讯与计算与普通的 GEL 格式一样，在进行第 $n+1$ 时间层上的 GER 格式的计算时，在 n 层上，相邻两个进程的最初与最后四个内部节点的数据分别向前一个进程与后一个进程发送。通讯简图如图 4 所示。

在 MPI 并行环境下，阻塞通信模式下用到的计算与通讯过程中的命令，如下，

MPI_Send(void* buf, int count, MPI_Datatype, int destination, int tag, MPI_Comm comm)

MPI_Recv(void* buf, int count, MPI_Datatype, int source, int tag, MPI_Comm comm, MPI_Status*status)

非阻塞通信模式下三种差分格式的通信和计算与阻塞通信模式下的基本一致，差异之处在于各个进程之间的通讯可能尚未结束时，就已经在进行数据的计算，这样往往可以使 CPU 在各个核心在通讯的过程中，仍然满载运行，从而减少程序运行的时间，达到并行计算效率的提升。但非阻塞通讯同样带来程序实现成本的上升。非阻塞通信模式流程如表 2 所示。

在 MPI 并行环境下，非阻塞通信模式下用到的计算与通讯过程中的命令，如下

MPI_Isend(void* buf, int count, MPI_Datatype, int destination, int tag, MPI_Comm comm, MPI_Request*request)

MPI_Irecv(void* buf, int count, MPI_Datatype, int source, int tag, MPI_Comm comm, MPI_Request*request)

MPI_Waitall(int count, MPI_Request*request, MPI_Status*status)

4. 数值并行计算

为了验证分层交替格式的稳定性及误差情况，以及并行效率的提升情况。作如下数值运算。对方程(1)，考虑初始条件，$u^0(x)=\sin x$ 为初值，此时，问题(1)的精确解为 $u(x,t)=\mathrm{e}^{-t}\sin x$。

表 3 为 $h=\pi/403$ 所给出的近似解及误差情况，$\left|\mathrm{e}_i^n\right|$ 为绝对误差，$\left|E_i^n\right|$ 为相对误差，$t=0.1$ 为最终计算时间。

下文中给出的数值计算结果是在 ACER 双核四线程计算机，处理器 i3-2350M 进行并行计算，定义内部节点数为 40002，时间层为 2000，时间步长 $\Delta t=5\times10^{-8}$，针对分组交替格式，分别给出了在 1、2、4、

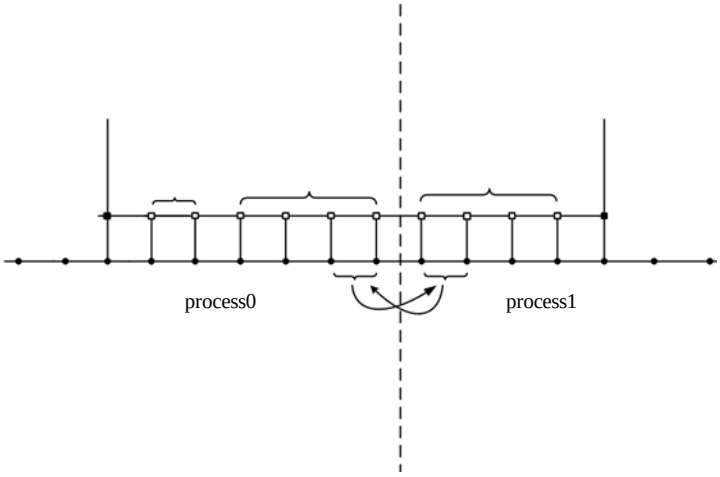

Figure 2. Communication mode of GEL
图 2. GEL 格式通讯简图

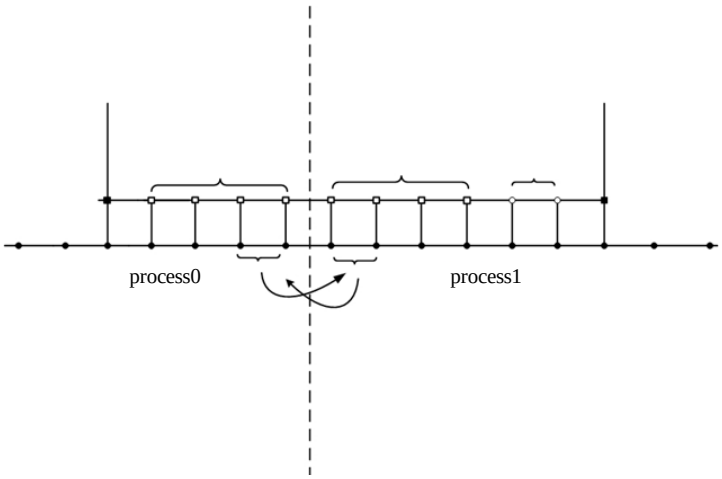

Figure 3. Communication mode of GER
图 3. GER 格式通讯简图

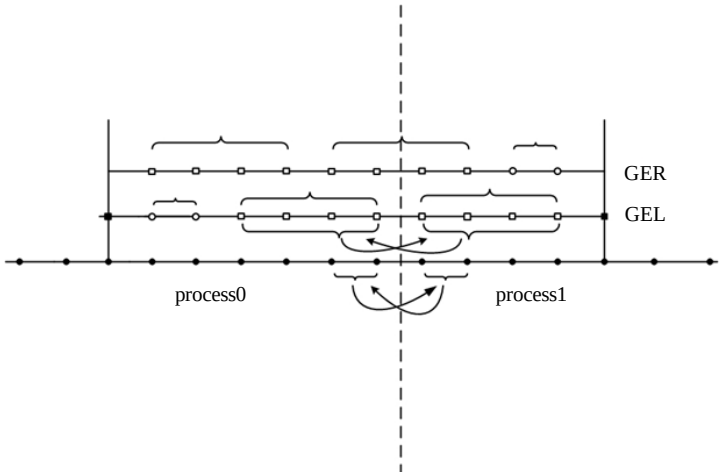

Figure 4. Communication mode of AGE
图 4. AGE 格式通讯简图

Table 2. The step of non-blocking communication mode
表 2. 非阻塞通信模式下并行详细步骤

	进程 0	其他进程
步骤 1	获得本进程的所负责的计算区间，并读取本进程的计算数据	同 0 进程
步骤 2	发送数据到其他进程或从其他进程接收数据	同 0 进程
步骤 3	非通信区域的数据计算	同 0 进程
步骤 4	等待通信完成，计算通信区域的数据	同 0 进程
步骤 5	进入下一时间层的计算，转到步骤 2	同 0 进程

Table 3. The comparison of exact solution and approximate solution
表 3. 数值解与解析解及误差对比

i	Exact solution (10^{-1})	Approximate solution (10^{-1})	$\lvert e_i^n \rvert (10^{-1})$	$\lvert e_i^n \rvert (10^{-1})$
1	0.07053	0.07023	0.00031	4.34677
40	2.77583	2.76178	0.01405	5.06082
80	5.28394	5.25725	0.02669	5.05077
123	7.40600	7.36869	0.03731	5.03791
163	8.64347	8.59988	0.04359	5.04310
203	9.04730	9.04019	0.04561	5.04158
243	8.57855	8.53531	0.04325	5.04139
283	7.28243	7.24570	0.03673	5.04308
323	5.28394	5.25727	0.02667	5.04770
363	2.77583	2.76185	0.01398	5.03600
402	0.07053	0.07018	0.00035	4.95515

8 条进程下的程序运行时间，在多核并行运算的程序中，也给出了阻塞通信模式与非阻塞通信模式的时间效率的对比，并以图表的方式给予了清晰的展示。非阻塞通信相对于阻塞通信来说，并未改变算法的逻辑结构，非阻塞通信只是降低了通信过程中消息传递的成本，因此阻塞通信与非阻塞通信在最终数值结果上是一致的。表 4~7 展现了不同进程数目下的并行运算时间。表 8 给出不同进程数目下的并行加速比。图 5~7 展示了表 4 中时间数据的柱状图，使并行效率得到了更清晰的展示。

5. 总结

在上述数值模拟计算的数据表格中，可以看出多条进程下，运用分组交替格式数值计算四阶抛物方程要比一个进程拥有更快的效率，多次试验数据表明，2 条进程运算时间约是 1 条进程的 1/2，4 条进程运算时间约是 1 条进程的 1/3，8 条进程运算的时间约是一条进程的 1/3，可以看到，随着进程数量的增多，程序运算的时间并没有随之线性下降，可以在表 7 中看到，在程序运行中，大量的时间用于各个进程之间的通信，从而导致计算效率相对于 4 条进程执行并未得到明显的提高。产生这种问题的原因一方面是由于进程总数过多，导致各个进程之间的通讯变得复杂。另一方面的原因是进程总数超过了计算机 CPU 的硬件核心总数，一个核心不再单独的只负责一个进程。所以，要达到完美的并行效果，就尽量要求进程总数与并行计算系统的计算核心总数相持平或小于 CPU 核心数。

Table 4. The executive time of 4 processes in blocking and non-blocking communication mode (sec.)
表 4. 4 进程阻塞通信与非阻塞通信模式的程序运算时间比较(单位/秒)

	进程 0	进程 1	进程 2	进程 3	单个进程
AGE 阻塞通信模式计算时间	58.505	58.968	61.701	65.237	
AGE 阻塞通信模式总时间	81.389	81.360	81.358	81.359	
AGE 单进程运算执行时间					220.363
AGE 非阻塞通信模式计算时间	64.990	63.319	66.774	63.720	
AGE 非阻塞通信模式总时间	76.650	76.709	76.741	76.675	
AGE 非阻塞通信模式等待通信时间	10.529	11.895	7.635	8.703	

Table 5. The executive time of 2 processes in blocking and non-blocking communication mode for AGE scheme (sec.)
表 5. AGE 格式下 2 进程阻塞通信与非阻塞通信模式的程序运算时间比较(单位/秒)

	非阻塞通信模式等待时间	计算时间	程序合计用时
阻塞通信 0 号进程	Nan	114.011	121.827
阻塞通信 1 号进程	Nan	119.580	121.828
非阻塞通信 0 号进程	0.4858	112.353	113.167
非阻塞通信 1 号进程	0.4295	112.446	113.206

Table 6. The executive time of 8 processes in blocking communication mode for AGE scheme (sec.)
表 6. AGE 格式下 8 进程阻塞通信程序运算时间(单位/秒)

	进程 0	进程 1	进程 2	进程 3	进程 4	进程 5	进程 6	进程 7
计算时间	35.182	31.825	29.418	33.193	35.810	35.380	37.449	34.668
程序合计用时	71.437	71.439	71.434	71.433	71.448	71.442	71.416	71.388

Table 7. The executive time of 8 processes in non-blocking communication mode for AGE scheme (sec.)
表 7. AGE 格式下 8 进程非阻塞通信程序运算时间(单位/秒)

	进程 0	进程 1	进程 2	进程 3	进程 4	进程 5	进程 6	进程 7
通信等待时间	28.775	25.150	32.126	18.025	32.614	19.151	36.485	32.107
计算时间	38.148	39.938	35.049	45.287	33.652	40.086	30.537	35.573
程序合计用时	68.250	68.235	68.217	68.226	68.261	68.276	68.261	68.245

Table 8. Comparison of the parallel efficiency in different processes
表 8. 不同进程数下并行效率比较

	总时间(单位/秒)	加速比	并行效率
非并行时	220.363	-----	-----
2 进程阻塞通信	121.828	1.81	0.905
2 进程非阻塞通信	113.206	1.95	0.975
4 进程阻塞通信	81.389	2.71	0.678
4 进程非阻塞通信	76.741	2.87	0.716
8 进程阻塞通信	71.448	3.08	0.385
8 进程非阻塞通信	68.276	3.23	0.404

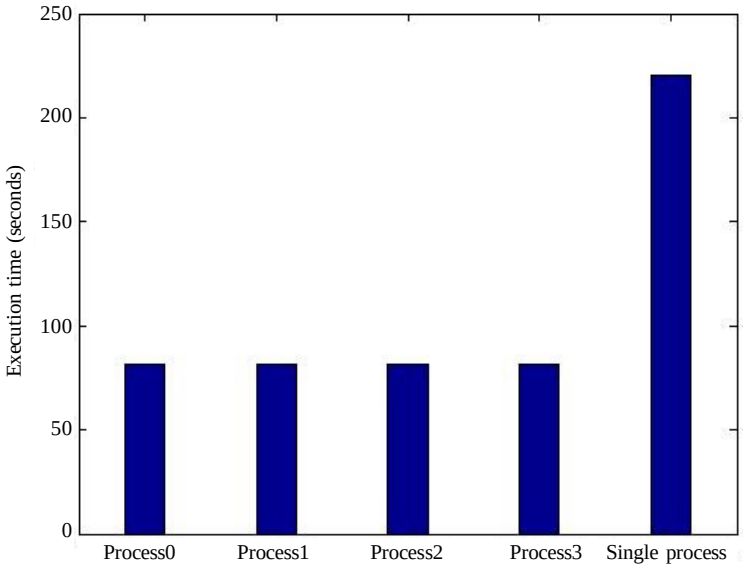

Figure 5. The executive time of single process and 4 processes of AGE blocking communication program
图 5. AGE 阻塞通信模式下四进程并行计算与单进程计算的计算时间柱状图

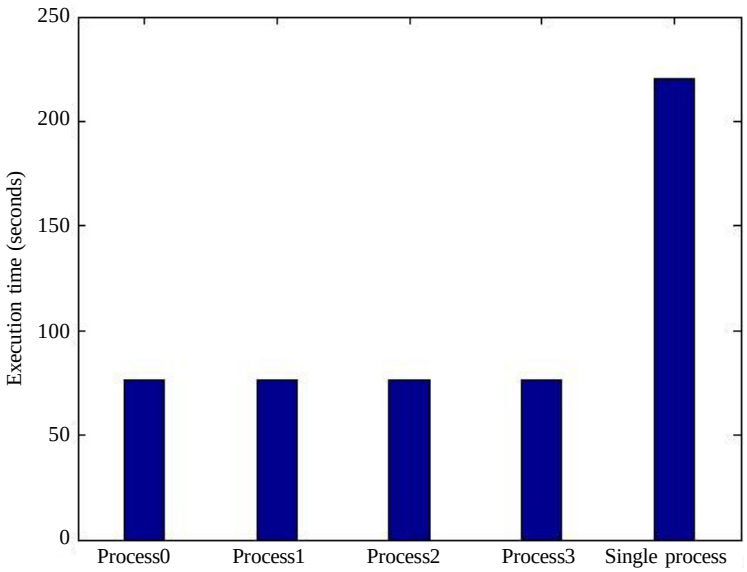

Figure 6. The executive time of single process and 4 processes of AGE non-blocking communication program
图 6. AGE 非阻塞通信模式下四进程并行计算与单进程计算的计算时间柱状图

在表 4~7 与图 7 中，可以看到非阻塞通信相对于阻塞通信模式的优势。以 4 条进程运算为例，非阻塞通信程序的总时间相对于阻塞通信程序的总时间是减少的。在信息传递过程中，当数据发送操作已完成，非阻塞通信模式即开始数据的计算，阻塞通信模式需要等待各个进程间的通信完成，因而，非阻塞通信降低了程序的运算时间。在表 8 和图 5，图 6 中，可以看到，多进程计算相当于单进程计算时是极有优势的。在表 8 中，随着进程总数的增加，加速比是不断提高的，但并行的效率却逐渐下降，这也从

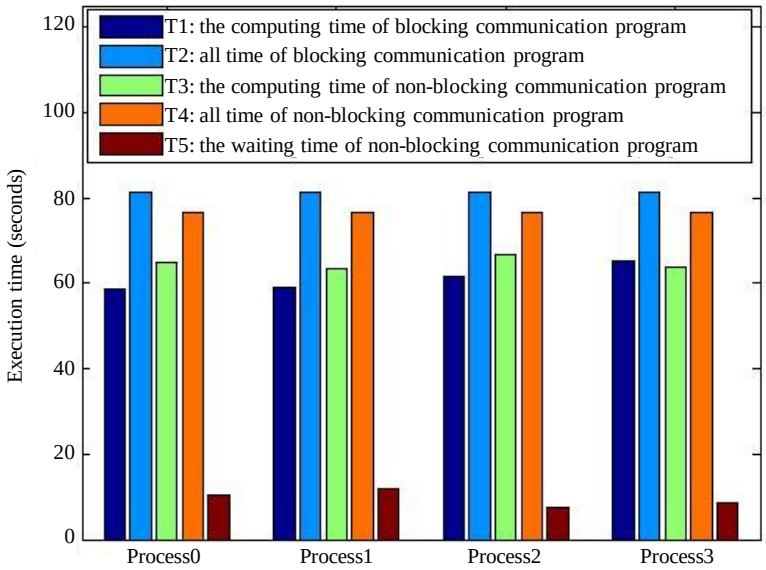

Figure 7. The time of AGE blocking and non-blocking communication program in 4 processes
图 7. 四进程下 AGE 阻塞通信模式与非阻塞通信模式下时间柱状图

侧面论证了进行多进程并行运算时，要选择恰当的进程总数。在表 8 中，还可以看到非阻塞通信相对于阻塞通信的优势，即在相同的进程数目下，无论是加速比还是并行效率，非阻塞通信都要高于阻塞通信。

参考文献 (References)

[1] Vabishchevich, P.N. (2015) Explicit Schemes for Parabolic and Hyperbolic Equations. *Applied Mathematics and Computation*, **250**, 424-431.

[2] Evans, D.J. and Abdullah, A.R. (1983) Group Explicit Methods for Parabolic Equations. *International Journal Computer Mathematics*, **14**, 73-105. https://doi.org/10.1080/00207168308803377

[3] Saul'yev, V.K. (1965) Integration of Equations of Parabolic Type by the Method of Nets. *Proceedings of the Edinburgh Mathematical Society*, **14**, 247-248. https://doi.org/10.1017/S0013091500008890

[4] Evans, D.J. and Abdullah, A.R. (1985) A New Explicit Method for the Diffusion-Convection Equation. *Computers and Mathematics with Applications*, **11**, 145-154.

[5] Abdullah, A.R. (1991) The Four Point Explicit Decoupled Group (EDG) Method: A Fast Poisson Solver. *International Journal of Computer Mathematics*, **38**, 61-70. https://doi.org/10.1080/00207169108803958

[6] Evans, D.J. (1985) Alternating Group Explicit Method for the Diffusion Equations. *Applied Mathematical Modelling*, **9**, 201-206.

[7] Ali, M., Hj, N., Teong, K. and Khoo (2010) Numerical Performance of Parallel Group Explicit Solvers for the Solution of Fourth Order Elliptic Equations. *Applied Mathematics and Computation*, **217**, 2737-2749.

[8] Zhang, B.L. and Li, W.Z. (1994) One Alternating Segment Crank-Nicolson Scheme. *Parallel Computing*, **20**, 897-902.

[9] 张宝琳. 求解扩散方程的交替分段显－隐式方法[J]. 数值计算与计算机应用, 1991(4): 245-253.

[10] Kellogg, R.B. (1964) An Alternating Direction Method for Operator Equations. *Journal of the Society of Industrial and Applied Mathematics*, **12**, 7. https://doi.org/10.1137/0112072