Advances in Applied Mathematics 应用数学进展, 2017, 6(3), 249-258
Published Online May 2017 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2017.63030

Hans汉斯

Higher Order Iterative Operator Splitting Method for the 2D Convection Diffusion Equation with Variable Coefficients

Lin Yao, Haiyan Su

College of Mathematics and System Sciences, Xinjiang University, Urumqi Xinjiang
Email: yaolin.wushi@foxmail.com, shymath@163.com

Received: May 3rd, 2017; accepted: May 18th, 2017; published: May 25th, 2017

Abstract

In this paper, a novel higher order iterative operator splitting method is presented for the 2D convection diffusion equation with the variable coefficient. The proposed scheme combines the classical iterative scheme and Zassenhaus product formula for the temporal discretization. And Fourier pseudo spectral method and dimensional splitting scheme are applied for the spatial operators. The numerical results verified that the proposed method can get second order accuracy by weighted iterative scheme. Besides, the new method not only can reduce numerical error but also save a lot of CPU time than the classical iterative method.

Keywords

Convection Diffusion, Zassenhaus Product, Higher Order Iterative, Fourier Pseudo Spectral

变系数2D对流扩散方程的高阶迭代算子分裂方法

姚　林，苏海燕

新疆大学数学与系统科学学院，新疆　乌鲁木齐
Email: yaolin.wushi@foxmail.com, shymath@163.com

收稿日期：2017年5月3日；录用日期：2017年5月18日；发布日期：2017年5月25日

摘　要

本文针对变系数2D对流扩散方程，呈现了一种新颖的高阶迭代算子分裂方法。该方法结合了经典迭代格式和Zassenhaus乘积公式。傅立叶谱方法和维数分裂格式用于空间算子。数值实验验证了所提出的方法

文章引用: 姚林, 苏海燕. 变系数 2D 对流扩散方程的高阶迭代算子分裂方法[J]. 应用数学进展, 2017, 6(3): 249-258.
https://doi.org/10.12677/aam.2017.63030

通过加权方法可以达到高阶精度。此外，新方法不仅可以减少误差而且能够节省大量的CPU时间。

关键词

对流扩散，Zassenhaus乘积，高阶迭代，傅立叶拟谱

1. 引言

众所周知，许多复杂有趣的现象是由于对流扩散方程中的对流项和扩散项的不同引起的。因此，它的数值解的研究在许多科学和工程领域中都具有重要的意义，呈现于[1] [2] [3]。因为扩散项和对流项的特点，很难找到一个精确有效的数值方法。因此，求解对流扩散问题的有效数值解方法是计算数学中的一个重要研究课题。已有的数值方法有限差分方法[4]，有限体积方法[5]和有限元方法[6]。然而，对于对流占优的问题，通常的有限差分法和有限元法解对流占优的问题可能会产生数值震荡。本文提出的新方法能够很好地消除震荡。

本文将考虑变系数对流扩散方程：

$$\begin{aligned} &u_t = \kappa\Delta u + v(x)\cdot\nabla u + f(x,t), (x,t)\in\Omega\times R^+, \\ &u(x,0) = u_0(x), x\in\Omega, \end{aligned} \tag{1}$$

这地方 Ω 是一个在 R^2 上的有界区域，R^+ 是在 $(0,T]$ 上的时间范围，$u(x,t)$ 表示浓度，$v(x)=[v_1(x),v_2(x)]^T$ 表示速度场，$\kappa(0<\kappa\le 1)$ 是扩散系数，f 是有界的连续函数。本文将使用周期性边界条件处理对流扩散方程。

Jürgen Geiser 研究了常系数对流扩散方程，使用改进的加权迭代格式，得到了很好的结果，呈现于[7] [8]。本文研究的高阶迭代格式也是改进的经典迭代格式基于 Zassenhaus 乘积公式。主要的思想是应用 Zassenhaus 乘积公式不断改进迭代格式的初始条件，从而提高经典迭代格式的精度和节省 CPU 时间。经典迭代格式的基本思想是把复杂的物理问题分解成简单的物理问题，从而能够简化运算和并行计算，看[9] [10]。Zassenhaus 乘积公式是李代数中的一个基本的重要的公式。

本文将应用加权方法求解变系数的二维对流扩散方程。空间离散应用傅立叶谱方法，时间离散使用加权迭代格式和高阶龙格库塔技术。更具体地，我们使用维数分裂的思想把高维问题分成几个简单的低维问题，使求解变得简单，本文的维数分裂格式是沿着 X 方向的算子和沿 Y 方向的算子。然后使用高阶迭代格式和高阶龙格库塔方法去解低维问题，可以减少分裂误差和促进解的过程。另外，提出的新方法要比经典的迭代方法省 CPU 时间和收敛精度高。

本文的框架如下：第二部分，介绍迭代算子分裂方法。第三部分，介绍傅立叶拟谱方法和维数分裂格式。第四部分，讨论高阶迭代格式联合经典的迭代格式和 Zassenhaus 乘积公式。第五部分，几个数值实例验证高阶格式的高效性和收敛率。第六部分，结论。

2. 经典的迭代算子分裂方法

我们知道经典的迭代格式广泛用于解决现实生活中的问题。本文考虑下列的非均匀的柯西问题

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = Au(t) + Bu(t) + f(t), t \in [0,T], u(0) = u_0,$$

其中 u_0 是初始条件，线性算子是在巴拿赫空间 X $(A,B: X \to X)$ 上的有界线性算子， $f(t)$ 是连续函数。

算法 2.1 给出了经典的迭代格式，本文考虑使用交替迭代格式去求解上面的非均匀的柯西问题：

$$\begin{aligned} \frac{\mathrm{d}u_i(t)}{dt} &= Au_i(t) + Bu_{i-1}(t) + f(t), u_i(t^n) = u^n, \\ \frac{\mathrm{d}u_{i+1}(t)}{\mathrm{d}t} &= Au_i(t) + Bu_{i+1}(t) + f(t), u_{i+1}(t^n) = u^n, \end{aligned} \tag{2}$$

这里初始迭代 $u_0(t) = 0$，其中 u^n 是初始条件， $t \in [t^n, t^{n+1}], i = 1,3,\cdots,2m+1$。这个迭代格式的数值解定义为 $u^{n+1} = u_{2m+2}(t^{n+1}), m = 0,1,2,\cdots$。在下面的章节中，将推导出高阶迭代格式基于交替迭代格式和 Zassenhaus 乘积公式。

3. 傅立叶谱离散和维数分裂格式

我们应用傅立叶谱方法[11]去解方程(3)和(4)。更具体地，给定一个正整数 N，使

$$x_l = \frac{2\pi l}{N}, l = 0,1,\cdots,N-1,$$

是在区间 $[0,2\pi)$ 上的等距网格，这些点称为傅立叶谱点。给定一套节点基函数，然后做这套基函数的线性组合，组合系数是节点函数值，有

$$u(x,t) = \sum_{l=0}^{N-1} u(x_l,t) h_l(x),$$

这地方基函数 $h_l(x) = \frac{1}{N}\sin\left[N\frac{x-x_l}{2}\right]\cot\left[\frac{x-x_l}{2}\right]$ 和 $h_l(x_k) = \delta_{lk}, l,k = 0,1,\cdots,N-1$， δ_{lk} 是克罗内克 δ 函数。对函数 $u(x,t)$ 求 d 阶导，导函数可以应用谱离散点的线性组合近似逼近，有

$$u^{(d)}(x,t) = \sum_{l=0}^{N-1} u(x_l,t) h_l^{(d)}(x),$$

我们可以把上述公式写成矩阵向量乘积的形式

$$[u(t)]^{(d)} = D^{(d)} u(t), d = 1,2,$$

这里

$$\begin{aligned} D^{(d)} &= \left(h_l^d(x)\right)_{k,l=0,1,\cdots,N-1}, \\ u(t) &= \left[u(x_0,t),\cdots,u(x_{N-1},t)\right]^{\mathrm{T}}, \\ [u(t)]^{(d)} &= \left[u^{(d)}(x_0,t),\cdots,u^{(d)}(x_{N-1},t)\right]^{\mathrm{T}}. \end{aligned}$$

一阶傅立叶微分矩阵 $D^{(1)}$ 可以写成形式

$$D_{kl}^{(1)} = h_l'(x_k) = \begin{cases} 0, k = l, \\ \frac{(-1)^{k+l}}{2}\cot\left[\frac{(k-l)\pi}{N}\right], k \neq l, \end{cases}$$

二阶傅立叶微分矩阵 $D^{(2)}$ 可以写成形式

$$D_{kl}^{(2)} = h_l''(x_k) = \begin{cases} -\dfrac{N^2+2}{12}, k = l, \\ -\dfrac{(-1)^{k+l}}{2}\sin^{-2}\left[\dfrac{(k-l)\pi}{N}\right], k \neq l. \end{cases}$$

特别地，定义 $D_1 = D^{(1)}, D_2 = D^{(2)}$。这样就推导了一维方程的一阶微分矩阵和二阶微分矩阵。下面讨论二维方程的微分矩阵形式。

对于本文要求解的二维对流扩散方程，首先应用维数分裂处理这个方程，能够得到下列的形式

$$\frac{\mathrm{d}u(x,y,t)}{\mathrm{d}t} = A_x u(x,y,t) + A_y u(x,y,t) + f(x,y,t),$$

这地方

$$A_x = \nu(x,y)\frac{\partial u(x,y,t)}{\partial x} + \kappa_x \frac{\partial^2 u(x,y,t)}{\partial x^2},$$
$$A_y = \nu(x,y)\frac{\partial u(x,y,t)}{\partial y} + \kappa_y \frac{\partial^2 u(x,y,t)}{\partial y^2},$$

然后使用傅立叶拟谱方法离散空间算子，前面介绍了一维微分矩阵 D_1 和 D_2，对于二维空间算子，使用克罗内克张量积通过一阶微分矩阵 D_1 和二阶微分矩阵 D_2 能够得到二维微分矩阵，把偏微分方程转化为常微分方程组，方程形式

$$\frac{\mathrm{d}U(t)}{\mathrm{d}t} = AU(t) + BU(t) + F(t), \tag{3}$$

这里

$$A = a(I \otimes D_1) + \kappa_x (I + D_2),$$
$$B = b(D_1 \otimes I) + \kappa_y (D_2 \otimes I),$$
$$U(t) = \left[u(x_0,y_0,t),\cdots,u(x_{N-1},y_0,t),\cdots,u(x_{N-1},y_{N-1},t)\right]^{\mathrm{T}},$$
$$F(t) = \left[f(x_0,y_0,t),\cdots,f(x_{N-1},y_0,t),\cdots,f(x_{N-1},y_{N-1},t)\right]^{\mathrm{T}},$$
$$a = diag\left(\left[\nu_1(x_0,y_0),\cdots,\nu_1(x_{N-1},y_0),\cdots,\nu_1(x_{N-1},y_{N-1})\right]^{\mathrm{T}}\right),$$
$$b = diag\left(\left[\nu_2(x_0,y_0),\cdots,\nu_2(x_{N-1},y_0),\cdots,\nu_2(x_{N-1},y_{N-1})\right]^{\mathrm{T}}\right),$$

$\otimes$表示克罗内克张量积符号，I 表示单位矩阵。

在下面的章节中，我们将推导高阶迭代格式和讨论常微分方程组(3)的数值解，理论能够证明本文提出的加权迭代格式可以增强收敛精度和节省 CPU 时间。

4. 高阶迭代格式用于时间离散

本节将讨论高阶迭代格式基于算法 2.1 和 Zassenhaus 乘积公式，构造的高阶方法可以提高收敛率和减少数值误差，以及减少迭代步数，从而节省时间。首先给出算法 2.1 的分析解

$$\begin{aligned} u_i(t^{n+1}) &= \exp(A\tau)u_i(t^n) + \exp(At^{n+1})\int_{t^n}^{t^{n+1}} \exp(-As)\left(Bu_{i-1}(s) + f(s)\right)\mathrm{d}s, \\ u_{i+1}(t^{n+1}) &= \exp(B\tau)u_{i+1}(t^n) + \exp(Bt^{n+1})\int_{t^n}^{t^{n+1}} \exp(-Bs)\left(Au_i(s) + f(s)\right)\mathrm{d}s. \end{aligned} \tag{4}$$

我们假设 $u_{i-1}(t) = 0$，因此对于 $i = 1$，有

$$u_1\left(t^{n+1}\right)=\exp(A\tau)u_1\left(t^n\right)+\exp\left(At^{n+1}\right)\int_{t^n}^{t^{n+1}}\exp(-As)\,f(s)\mathrm{d}s, \tag{5}$$

这时 $i+1=2$，得到第二步

$$u_2\left(t^{n+1}\right)=\exp(B\tau)u_2\left(t^n\right)+\exp\left(Bt^{n+1}\right)\int_{t^n}^{t^{n+1}}\exp(-Bs)\left(Au_1(s)+f(s)\right)\mathrm{d}s, \tag{6}$$

这地方 τ 是时间步长，$\exp(As)$ 是指数函数。下面推导加权迭代格式。

我们将应用 Zassenhaus 乘积公式[12]，这个公式具体形式

$$\exp\left((A+B)t\right)=\exp(At)\exp(Bt)\prod_{k=2}^{\infty}\exp\left(\tilde{u}_k t^k\right),$$

这里，将公式进行泰勒公式展开，能够推导出下列的权的误差阶[10]

$$\omega_j(t)=\exp(Bt)\exp(\tilde{u}_2)\cdots\exp(\tilde{u}_j)+o\left(t^{j+1}\right),$$

这地方 $\tilde{u}_j\ (j=2,3,\cdots,\infty)$ 是 Zassenhaus 指数。能够推导出 Zassenhaus 指数为

$$\tilde{u}_2=-\frac{1}{2}[A,B],$$

$$\tilde{u}_3=-\frac{1}{3}\big[B,[B,A]\big]+\frac{1}{6}\big[A,[A,B]\big],$$

这里[]是李代数括号，我们定义：$[A,B]=AB-BA$。因此权公式可以表示为

$$\omega_1(t)=I+Bt,$$

$$\omega_2(t)=I+Bt+\frac{B^2t^2}{2}-\frac{1}{2}[A,B]t^2,$$

$$\omega_3(t)=I+Bt+\frac{B^2t^2}{2}-\frac{1}{2}[A,B]t^2+\frac{B^3t^3}{6}+\left(-\frac{1}{3}\big[B,[B,A]\big]+\frac{1}{6}\big[A,[A,B]\big]\right)t^3-\frac{1}{2}B[A,B]t^3,$$

这样就推导出来了权公式。因此对于初始迭代(5)做如下改进

$$\hat{u}_i\left(t^{n+1}\right)=\exp(A\tau)\exp(B\tau)\prod_{j=2}^{i}\exp\left(\tilde{u}_j\tau^j\right)+\exp\left(At^{n+1}\right)\int_{t^n}^{t^{n+1}}\exp(-As)\,f(s)\mathrm{d}s,$$

因此迭代格式的误差精度提高到 $o\left(\tau^{i+j}\right)$ (参见[13])，这样每增加一步权，原来经典迭代格式的误差提高一阶。

一步加权和两步加权迭代格式为

$$\hat{u}_1\left(t^{n+1}\right)=\exp(A\tau)(I+B\tau)+\exp\left(At^{n+1}\right)\int_{t^n}^{t^{n+1}}\exp(-As)\,f(s)\mathrm{d}s,$$

$$\hat{u}_1\left(t^{n+1}\right)=\exp(A\tau)\left(I+B\tau+\frac{B^2\tau^2}{2}-\frac{1}{2}[A,B]\tau^2\right)+\exp\left(At^{n+1}\right)\int_{t^n}^{t^{n+1}}\exp(-As)\,f(s)\mathrm{d}s,$$

因此加权迭代的误差率能够提高到 $o\left(\tau^2\right)$ 和 $o\left(\tau^3\right)$。

这个 $\exp(At)$ 和 $\exp(Bt)$ 应用 Pade'方法近似，积分项使用四阶龙格库塔方法离散，这样能够获得高阶结果。加权迭代格式将通过一些数值实例进行验证。

5. 数值实验

本章节将验证提出的新格式的收敛阶和 CPU 时间，以及误差图。

首先给出方程(1)一个真解

$$u_{\text{exact}}(x,y,t)=\exp(-t)\cos(2\pi x)\cos(2\pi y),\ x,y\in[0,1]\times[0,1],$$

设置终止时间 $T=1$,初始条件 $u_0(x,y)=\cos(2\pi x)\cos(2\pi y)$。数值误差使用 L_∞ 范数形式，定义 L_∞ 范数

$$err_{L_\infty}:=\max\left(\max\left(\left|u\left(t^n\right)-u_{\text{exact}}\left(t^n\right)\right|\right)\right).$$

我们将选取不同的对流系数和不同的扩散系数来验证高阶格式的误差阶。

5.1. 数值实例 1

在这个数值例子中，考虑一般的对流扩散方程，扩散系数选取 $\kappa_x=\kappa_y=0.1$，表中将给出不同对流系数的情况。

表 1 中呈现的是加权迭代算法的误差，对流系数选取 $v_1(x,y)=-3+x, v_2(x,y)=-3+y$，选取空间谱点 $N=16$，时间步长选取不同的适合的步长。

表 2 将给出该算法误差，空间点选取 $N=16$，对流系数选取 $v_1(x,y)=1-x-y, v_2(x,y)=1-x-y$，挑选和表 1 中一样的时间步长。

表 3 说明的是加权迭代算法的误差阶，对流系数选择 $v_1(x,y)=1+x, v_2(x,y)=1+y$，空间点仍然选择 $N=16$，时间步长和表 2 同步。

表 4 呈现的权迭代算法的 CPU 时间比较，选择不同的空间点，时间步长 $\tau=\frac{1}{1600}$，对流系数 $v_1(x,y)=v_2(x,y)=1-x-y$，比较没有加权，一步加权和两步加权的时间。

图 1 呈现的是 2 步加权的误差图，对流系数选择表 1 中系数，时间步长 $\tau=\frac{1}{400}$，空间点 $N=20$。

图 2 呈现的也是 2 步加权的误差图，对流系数选择表 2 中系数，时间步长、空间点和图 1 中同步。

从表中能够看出，不同的对流系数对权格式的精度是有影响的，对流系数选择的不好可能会导致 2 步加权的误差增大，从时间的比较中也能看出，加权方法耗费的时间并不是很多相比于没有加权，但误差减小了很多，说明加权方法是一种花费时间少，且精度高的方法。从图中能够看出不同的对流系数呈现的误差图也不一样，选择对流系数函数值无零点的对流系数，误差图的跳跃性就不会很大，明显看出图 1 比图 2 好。

5.2. 数值实例 2

在这个例子中，将考虑对流占优的情形，选取扩散系数 $\kappa_x=\kappa_y=0.0001$，表中同样给出不同的对流系数来验证加权方法的误差阶情况。

表 5 呈现的是加权迭代的误差比较，对流系数选取表 1 中的对流系数，空间点选择 $N=16$，时间步长同样选择表 1 中的步长。

表 6 表明的是加权格式的误差阶比较，对流系数选择表 2 中的对流系数，空间谱离散点选择 $N=16$，时间步长和上表同步。

表 7 呈现的是高阶迭代方法的误差比较，对流系数选择表 3 中的系数，空间点和时间步长和表 6 同步。

图 3 呈现的是 2 步加权的误差图，对流系数、时间步长和空间点和图 1 同步。

图 4 表明的是 2 步加权的误差图，对流系数、时间步长和空间点和图 2 同步。

从表中可以看出，对于对角占优的对流扩散问题，加权迭代方法计算的比较好，同样体现了随着扩散系数的减小，误差会增大。同样也验证了加权方法的精度高，节省时间。从误差图中可以看出，图 4 的跳跃性相比图 2 更大，说明强对角占优凸显了双曲方程的特征，跳动性比较大，这样会影响高阶格式收敛性和稳定性。但不会影响加权方法的精度。

Table 1. Weighted iterative numerical error comparison. $v_1(x,y)=-3+x, v_2(x,y)=-3+y, \kappa_x=\kappa_y=0.1$

表 1. 加权迭代数值误差比较。$v_1(x,y)=-3+x, v_2(x,y)=-3+y, \kappa_x=\kappa_y=0.1$

1/τ	迭代次数	无加权	1 步加权	2 步加权
200	2	2.16E−02	9.50E−04	6.58E−04
400	2	1.11E−02	2.38E−04	1.61E−04
800	2	5.69E−03	5.93E−05	4.20E−05
1600	2	2.87E−03	1.48E−05	1.04E−05

Table 2. Comparison of weighted scheme error. $v_1(x,y)=v_2(x,y)=1-x-y, \kappa_x=\kappa_y=0.1$

表 2. 加权格式误差比较。$v_1(x,y)=v_2(x,y)=1-x-y, \kappa_x=\kappa_y=0.1$

1/τ	迭代次数	无加权	1 步加权	2 步加权
200	2	4.31E−03	5.20E−04	1.66E−04
400	2	2.18E−03	1.46E−04	3.44E−05
800	2	1.09E−03	4.75E−05	7.36E−06
1600	2	5.48E−04	9.48E−06	1.64E−06

Table 3. Weighted iterative numerical error comparison. $v_1(x,y)=1+x, v_2(x,y)=1+y, \kappa_x=\kappa_y=0.1$

表 3. 加权迭代数值误差比较。$v_1(x,y)=1+x, v_2(x,y)=1+y, \kappa_x=\kappa_y=0.1$

1/τ	迭代次数	无加权	1 步加权	2 步加权
200	2	1.04E−02	2.75E−04	2.90E−04
400	2	5.28E−03	6.88E−05	7.29E−05
800	2	2.65E−03	1.72E−05	1.83E−05
1600	2	1.33E−03	4.30E−06	4.58E−06

Table 4. CPU time comparison of weighted iterative. $v_1(x,y)=v_2(x,y)=1-x-y, \kappa_x=\kappa_y=0.1$

表 4. 加权迭代的 CPU 时间比较。$v_1(x,y)=v_2(x,y)=1-x-y, \kappa_x=\kappa_y=0.1$

N	8	10	16
无加权	9.0468s	18.0134s	152.5261s
1 步加权	9.4434s	18.2848s	164.1404s
2 步加权	10.0025s	19.1191s	168.0754s

Table 5. Comparison of weighted iterative error. $v_1(x,y)=-3+x, v_2(x,y)=-3+y, \kappa_x=\kappa_y=0.0001$

表 5. 加权迭代误差比较。$v_1(x,y)=-3+x, v_2(x,y)=-3+y, \kappa_x=\kappa_y=0.0001$

1/τ	迭代次数	无加权	1 步加权	2 步加权
200	2	5.01E+04	8.02E−03	1.52E−03
400	2	3.55E−01	2.01E−03	3.66E−04
800	2	4.71E−02	5.05E−04	8.80E−05
1600	2	2.46E−02	1.28E−04	2.12E−05

Table 6. Comparison of weighted scheme error. $v_1(x,y)=v_2(x,y)=1-x-y, \kappa_x=\kappa_y=0.0001$

表 6. 加权格式误差比较。 $v_1(x,y)=v_2(x,y)=1-x-y, \kappa_x=\kappa_y=0.0001$

1/τ	迭代次数	无加权	1 步加权	2 步加权
200	2	8.39E−02	9.14E−04	4.64E−04
400	2	4.28E−02	2.32E−04	1.12E−04
800	2	2.16E−02	5.72E−05	2.82E−05
1600	2	1.08E−02	1.45E−05	7.10E−06

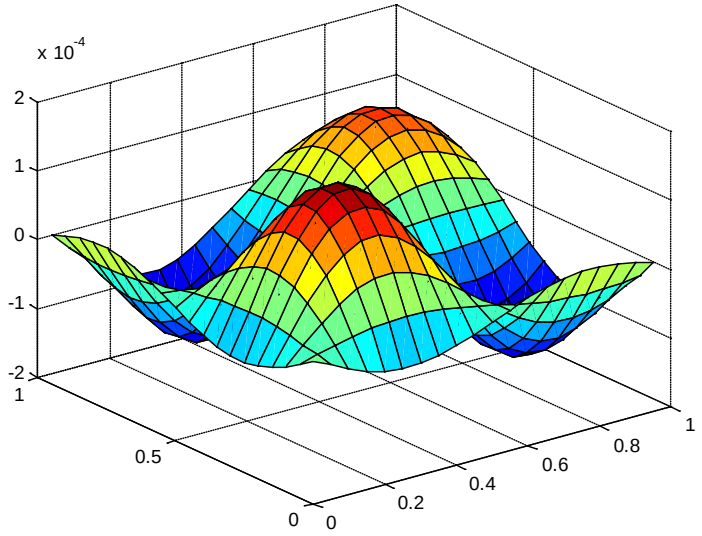

Figure 1. Error chart. $v_1(x,y)=-3+x, v_2(x,y)=-3+y, \kappa_x=\kappa_y=0.1, N=20, \tau=\frac{1}{400}$

图 1. 误差图。 $v_1(x,y)=-3+x, v_2(x,y)=-3+y, \kappa_x=\kappa_y=0.1, N=20, \tau=\frac{1}{400}$

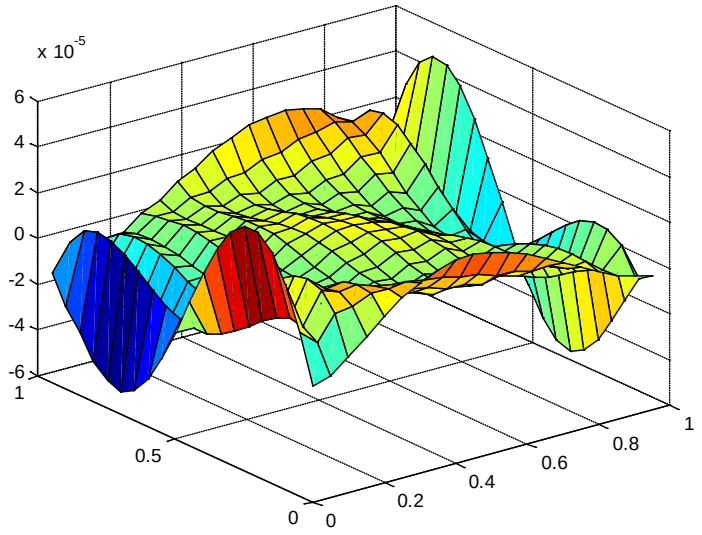

Figure 2. Error chart. $v_1(x,y)=v_2(x,y)=1-x-y, \kappa_x=\kappa_y=0.1$

图 2. 误差图。 $v_1(x,y)=v_2(x,y)=1-x-y, \kappa_x=\kappa_y=0.1$

Table 7. Comparison of weighted iterative error. $\nu_1(x,y)=1+x,\nu_2(x,y)=1+y,\kappa_x=\kappa_y=0.0001$

表 7. 加权迭代数值误差比较。$\nu_1(x,y)=1+x,\nu_2(x,y)=1+y,\kappa_x=\kappa_y=0.0001$

1/τ	迭代次数	无加权	1 步加权	2 步加权
200	2	7.20E−02	1.66E−03	6.17E−04
400	2	3.72E−02	4.15E−04	1.51E−04
800	2	1.90E−02	1.04E−04	3.76E−05
1600	2	9.64E−03	2.59E−05	9.36E−06

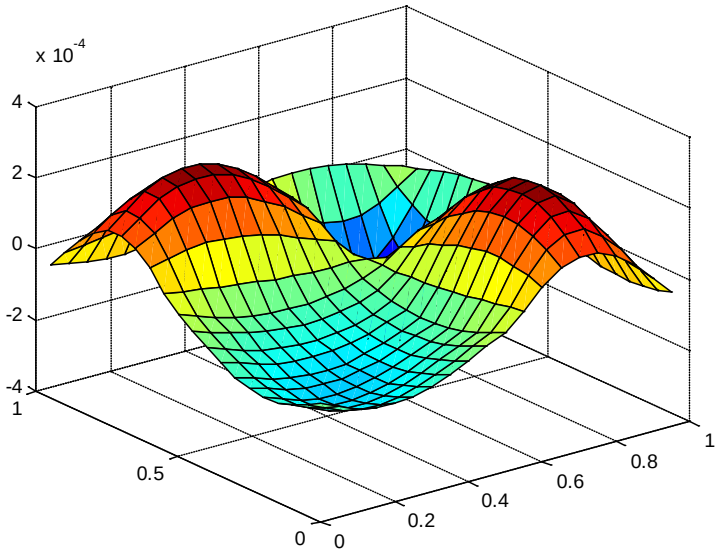

Figure 3. Error chart. $\nu_1(x,y)=-3+x,\nu_2(x,y)=-3+y,\kappa_x=\kappa_y=0.0001$

图 3. 误差图。$\nu_1(x,y)=-3+x,\nu_2(x,y)=-3+y,\kappa_x=\kappa_y=0.0001$

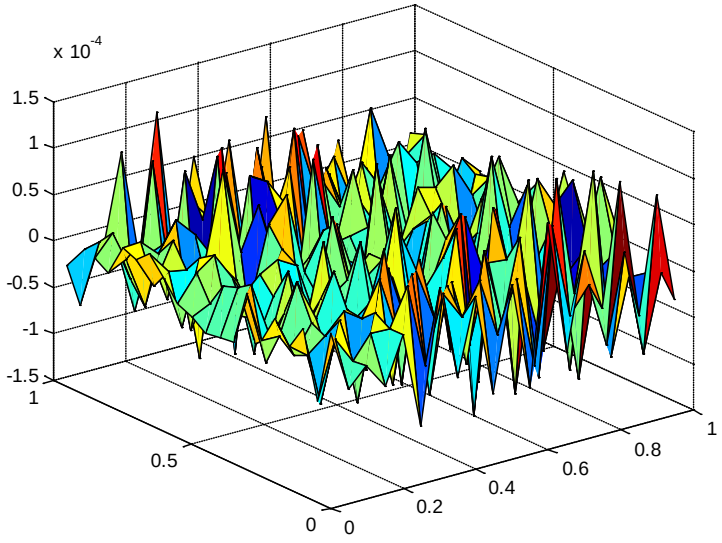

Figure 4. Error chart. $\nu_1(x,y)=\nu_2(x,y)=1-x-y,\kappa_x=\kappa_y=0.0001$

图 4. 误差图。$\nu_1(x,y)=\nu_2(x,y)=1-x-y,\kappa_x=\kappa_y=0.0001$

6. 结论

本文呈现了一种高阶迭代格式对于求解变系数2D对流扩散方程。提出的新方法结合了经典迭代格式和Zassenhaus乘积公式。傅立叶谱离散技术和维数分裂方法用于空间算子。数值算例呈现了新方法的高精度和高效性。

基金项目

国家自然科学基金资助项目(11271313)。

参考文献 (References)

[1] 章本照, 印建安, 张宏基. 流体力学数值方法[M]. 北京: 机械工业出版社, 2003

[2] 李荣华, 刘播. 微分方程数值解法[M]. 第四版. 北京: 高等教育出版社, 2009.

[3] Li, Q., Chai, Z. and Shi, B. (2014) An Efficient Lattice Boltzmann Model for Steady Convection Diffusion Equation. *Journal of Scientific Computing*, **61**, 308-326. https://doi.org/10.1007/s10915-014-9827-z

[4] Galligani, E. (2013) A Nonlinearity Lagging for the Solution of Nonlinear Steady State Reaction Diffusion Problems. *Communications in Nonlinear Science and Numerical Simulation*, **18**, 567-583.

[5] Angelini, O. and Brenner, K. (2013) A Finite Volume Method on General Meshes for a Degenerate Parabolic Convection Reaction Diffusion Equation. *Numerische Mathematik*, **123,** 219-257. https://doi.org/10.1007/s00211-012-0485-5

[6] Bause, M. and Schwegler, K. (2013) Higher Order Finite Element Approximation of Systems of the Convection Diffusion Reaction Equations with Small Diffusion. *Journal of Computational and Applied Mathematics*, **246**, 52-64.

[7] Geiser, J. and Tanoglu, G. (2011) Operator-Splitting Methods via the Zassenhaus Product Formula. *Applied Mathematics and Computation*, **217**, 4557-4575.

[8] Csomós, P. and Faragó, I. (2005) The Weighted Sequential Splitting and Their Analysis. *Computers and Mathematics with Applications*, **50**, 1017-1031.

[9] Geiser, J. (2008) Iterative Operator-Splitting Methods with Higher-Order Time Integration Methods and Applications for Parabolic Partial Differential Equations. *Journal of Computational Applied Mathematics*, **217**, 227-242. https://doi.org/10.1002/num.20568

[10] Geiser, J. (2011) Iterative Operator-Splitting Methods for the Nonlinear Differential Equations and Applications. *Numerical Method for Partial Differential Equation*, **27**, 1026-1054.

[11] Shen, J., Tang, T. and Wang, L. (2011) Spectral Methods. Springer, Berlin Heidelberg. https://doi.org/10.1007/978-3-540-71041-7

[12] Scholz, D. (2006) A Note on the Zassenhaus Product Formula. *Journal of Mathematical Physics*, **47**, 373-418. https://doi.org/10.1063/1.2178586

[13] Geiser, J. and Tano, G. (2011) Higher Order Operator Splitting Methods via the Zassenhaus Product Formula: Theory and Application. *Computer and Mathematic with Application*, **62**, 1994-2015.