Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. *, No. *, *, pp. 1-18
DOI:10.22034/cmde.2021.46776.1966

Error analysis and Kronecker implementation of Chebyshev spectral collocation method for solving linear PDEs

Mehdi Razavi
Department of Applied Mathematics and Mahani Mathematical Research Center, Shahid Bahonar University of Kerman, Kerman, Iran.
E-mail: mrazavi@math.uk.ac.ir

Mohammad Mehdi Hosseini
Department of Applied Mathematics and Mahani Mathematical Research Center, Shahid Bahonar University of Kerman, Kerman, Iran.
E-mail: mhosseini@uk.ac.ir

Abbas Salemi *
Department of Applied Mathematics and Mahani Mathematical Research Center, Shahid Bahonar University of Kerman, Kerman, Iran.
E-mail: salemi@uk.ac.ir

Abstract Numerical methods have essential role to approximate the solutions of Partial Differential Equations (PDEs). Spectral method is one of the best numerical methods of exponential order with high convergence rate to solve PDEs. In recent decades the **C**hebyshev **S**pectral **C**ollocation (CSC) method has been used to approximate solutions of linear PDEs. In this paper, by using linear algebra operators, we implement **K**ronecker **C**hebyshev **S**pectral **C**ollocation (KCSC) method for n-order linear PDEs. By statistical tools, we obtain that the Run times of KCSC method has polynomial growth, but the Run times of CSC method has exponential growth. Moreover, error upper bounds of KCSC and CSC methods are compared.

Keywords. Error analysis, Chebyshev spectral collocation method, Kronecker product, linear Partial differential equations.

2010 Mathematics Subject Classification. 65M70, 65N15, 35G05, 15A06.

1. Introduction

Partial Differential Equations have many applications in modern mechanics and theoretical physics. For example, PDEs were used by James Clerk Maxwell to model electromagnetic fields interacting with electrical charges and currents, by Ludwig Boltzmann to describe the non-equilibrium dynamics of rarified gases, by Albert Einstein to phrase the laws of gravitation in the general theory of relativity and by Erwin Schrdinger and Werner Heisenberg to formulate quantum mechanics in mathematical-analytical terms.

Received: 02 July 2021 ; Accepted: 12 December 2021.
Abbas Salemi .

With a short look at the history of partial differential equations (PDEs) it can be found that analytic methods are very expensive and in many cases inefficient [8, 11, 22]. On the other hand, two nearly similar PDEs may have completely different solutions, so it would not be possible to offer a similar analytic method to solve all PDEs.

According to the most mathematicians to assessment numerical methods efficiency, the computations order and speed convergence should be considered. So far, in solving linear PDEs, several numerical methods has been applied and compared [7, 15, 26]. Among these methods, spectral method is one of the important numerical methods for solving these PDEs. Spectral methods try to approximate functions (solutions of differential equations, partial differential equations, etc.) by means of truncated series of orthogonal polynomials. The well known Fourier series (for periodic problems), as well as series made up by Chebyshev or Legendre polynomials (for non-periodic problems), are examples of such series of orthogonal functions. Spectral methods are geometrically less flexible than lower-order methods, and they are usually more complicated to implement[1, 2, 5, 9, 16, 17, 18, 19, 20, 21, 23, 25, 27, 28].

In this paper, by using Kronecker product, we implement Kronecker Chebyshev Spectral Collocation KCSC method for n-order linear PDEs and we show that , it has less computational cost than the CSC method, in addition to the benefit of generalising the method to n-order linear PDEs.

Also, the error upper bound of the CSC and KCSC methods are compared. The following are some of the benefits of the error upper bound offered for the KCSC method compared to earlier error upper bounds:
1-The error upper bound for n-order linear PDEs has been generalised.
2-For some matrix norms, the error upper bound of the KCSC approach is reduced.
Moreover, by using Matlab codes, we will compare the CPU and Run times of KCSC method with CSC method. The rest of this paper is organized as follows: In this section, we state some preliminary lemmas and Theorems. In Section 2, we offer an algorithmic structure of CSC method and we implement KCSC method for n-order linear PDEs. In Section 3, the error analysis of the mentioned methods CSC and KCSC are compared. In Section 4, the effectiveness of KCSC method is illustrated by numerical examples and we fit the Run times of the mentioned method by polynomial and exponential models. Also, by statistical tools, we compare these regression models and we show that the Run times of the KCSC method has polynomial growth and the Run times of the CSC method has exponential growth. The conclusion is stated in Section 5.

Theorem 1.1. [12, Theorem 5.14] *If the function $u(x)$ has $m+1$ continuous derivatives on $[-1,1]$ and $u_N(x) := \sum_{n=0}^{N} \alpha_n T_n(x)$ be the Chebyshev series expansion of $u(x)$, where*

$$\alpha_0 = \frac{1}{\pi}\int_{-1}^{1} \frac{u(x)}{\sqrt{1-x^2}}dx,\ \alpha_n = \frac{2}{\pi}\int_{-1}^{1} \frac{T_n(x)u(x)}{\sqrt{1-x^2}}dx,\ n \geq 1,$$

then

$$| u(x) - u_N(x) | = O(N^{-m}), \quad for\ all\ x \in [-1,1].$$

Theorem 1.2. [12, Theorem 5.7] *If $u(x)$ is continuous and either of bounded variation or satisfying a Dini-Lipschitz condition on $[-1,1]$, then its Chebyshev series expansion is uniformly convergent.*

In light of Theorems 1.1 and 1.2, it is worthwhile to mention that convergence rate of the spectral methods is $O(N^{-m})$.

Lemma 1.3 ([4]). *If $v(x)=\sum_{k=0}^{m}\alpha_k T_k(x)$, then $v'(x)=\sum_{k=0}^{m}\alpha_k T_k'(x)=\sum_{k=0}^{m}\alpha_k^{(1)}T_k(x)$ and $v''(x)=\sum_{k=0}^{m}\alpha_k T_k''(x)=\sum_{k=0}^{m}\alpha_k^{(2)}T_k(x)$. Assume that $\alpha:=[\alpha_0\ \cdots\ \alpha_m]^t$, $\alpha^{(1)}:=[\alpha_0^{(1)}\ \alpha_1^{(1)}\ \cdots\ \alpha_m^{(1)}]^t$ and $\alpha^{(2)}:=[\alpha_0^{(2)}\ \alpha_1^{(2)}\ \cdots\ \alpha_m^{(2)}]^t$. Therefore, there exists derivative matrix D such that $\alpha^{(1)}=D\alpha$ and $\alpha^{(2)}=D^2\alpha$, where*

$$D_{ij}=\begin{cases} j-1 & i+j \text{ is odd},\ j>i=1,\\ 2j-2 & i+j \text{ is odd},\ j>i>1,\\ 0 & \text{other wise,}\end{cases}$$

$$(D^2)_{ij}=\begin{cases} \frac{((j-1)^2-(i-1)^2)(j-1)}{2} & i+j \text{ is even},\ j>i=1,\\ ((j-1)^2-(i-1)^2)(j-1) & i+j \text{ is even},\ j>i>1,\\ 0 & \text{other wise.}\end{cases}$$

Similarly, we can define the Chebyshev derivative matrix of degree n and is denoted by D^n. Note that the general form of the matrices D and D^n are presented in [25].

2. Algorithm and implementation of Chebyshev Spectral Collocation (CSC) Method for n-order linear PDEs

In this section, we consider the following n-order linear PDEs in two dimensional variables $(x,y)\in[a_1,b_1]\times[a_2,b_2]$ with boundary conditions

$$\sum_{k,l=0}^{n_1,n_2}a_{k,l}(x,y)\frac{\partial^{k+l}}{\partial x^k\partial y^l}u(x,y)=f(x,y), \tag{2.1}$$

$$\begin{cases}\frac{\partial^{k-1}u}{\partial x^{k-1}}(a_1,y)=g_{k-1,1}(y),\ k=1,2,\ldots,\left[\frac{n_1+1}{2}\right].\\ \frac{\partial^{k-1}u}{\partial x^{k-1}}(b_1,y)=g_{k-1,2}(y),\ k=1,2,\ldots,\left[\frac{n_1}{2}\right].\\ \frac{\partial^{l-1}u}{\partial y^{l-1}}(x,a_2)=h_{l,1}(x),\ l=1,2,\ldots,\left[\frac{n_2+1}{2}\right].\\ \frac{\partial^{l-1}u}{\partial y^{l-1}}(x,b_2)=h_{l-1,2}(x),\ l=1,2,\ldots,\left[\frac{n_2}{2}\right].\end{cases} \tag{2.2}$$

where n_1 and n_2 are the order of the partial derivatives of x and y, respectively and $a_{k,l}(x,y),f(x,y)\in\mathcal{C}^n([a_1,b_1]\times[a_2,b_2])$, where $n=n_1+n_2$. In special case for $n_1=n_2=2$, these equations can be expressed in the following form:

$$\begin{aligned}a_{2,0}(x,y)\tfrac{\partial^2u}{\partial x^2}+a_{1,1}(x,y)\tfrac{\partial^2u}{\partial x\partial y}+a_{0,2}(x,y)\tfrac{\partial^2u}{\partial y^2}+a_{1,0}(x,y)\tfrac{\partial u}{\partial x}+\\ a_{0,1}(x,y)\tfrac{\partial u}{\partial y}+a_{0,0}(x,y)u=f(x,y),\end{aligned} \tag{2.3}$$

$$\begin{aligned}u(a_1,y)=g_{0,1}(y),\ u(b_1,y)=g_{0,2}(y),\ \forall y\in[a_2,b_2],\\ u(x,a_2)=h_{0,1}(x),\ u(x,b_2)=h_{0,2}(x),\ \forall x\in[a_1,b_1].\end{aligned} \tag{2.4}$$

2.1. **Chebyshev spectral collocation Method algorithm for n-order linear PDEs.** Without loss of generality, we assume that $a_1 = a_2 = -1$ and $b_1 = b_2 = 1$, because we can easily transform equation (2.1-2.2) into desirable domain $[-1, 1] \times [-1, 1]$.

Let $\mathbb{R}^n = \{(x_1, x_2, \ldots, x_n)^t |\ x_i \in \mathbb{R}\}$ and $v(x, y) := \sum_{i,j=0}^{m} \gamma_{i,j} T_i(x) T_j(y)$ be an approximate solution of the above PDEs equation. We fixed the following notations throughout this section:

$$\mathbf{T}(t) := [T_0(t)\ T_1(t)\ \cdots T_m(t)]^t \in \mathbb{R}^{m+1},\ \tilde{\boldsymbol{e}} := [(-1)^0\ -1\ \cdots (-1)^m]^t \in \mathbb{R}^{m+1},$$
$$\mathbf{e} := [1\ 1\ \cdots\ 1]^t \in \mathbb{R}^{m+1},\ \boldsymbol{\Gamma} := [\gamma_{0,0}\ \gamma_{0,1} \cdots \gamma_{0,m} \mid \cdots \mid \gamma_{m,0}\ \gamma_{m,1} \cdots \gamma_{m,m}]^t \in \mathbb{R}^{(m+1)^2}.$$

By the following steps, we approximate the solution of n-order linear PDEs (2.1-2.2) by using two-dimensional Chebyshev spectral collocation Method.

step 1: For a given $m \in \mathbb{N}$, let $v(x, y) = \sum_{i,j=0}^{m} \gamma_{i,j} T_i(x) T_j(y)$. We compute the following partial derivative of $v(x, y)$

$$\frac{\partial^{k+l} v(x,y)}{\partial x^k \partial y^l} = \sum_{i,j=0}^{m} \gamma_{i,j} T_i^{(k)}(x) T_j^{(l)}(y),\ k = 0, 1, \ldots, n_1,\ l = 0, 1, \ldots, n_2. \quad (2.5)$$

Putting the relations (2.5) in equation (2.1), the following equation is obtained

$$\sum_{i,j=0}^{m} \sum_{k,l=0}^{n_1,n_2} a_{k,l}(x, y) \gamma_{i,j} T_i^{(k)}(x) T_j^{(l)}(y) = f(x, y). \quad (2.6)$$

step 2: Let $\{p_1, \ldots, p_{m+1}\}$ be the roots of $T_{m+1}(x)$. By replacing (p_i, p_j), $1 \le i, j \le m + 1$ in equation (2.6), the linear system $\boldsymbol{W}\boldsymbol{\Gamma} = \boldsymbol{F}$ is obtained, where $\boldsymbol{W} \in M_{(m+1)^2}$ and $\boldsymbol{F} \in \mathbb{R}^{(m+1)^2}$.

step 3: By replacing $v(x, y) = \sum_{i,j=0}^{m} \gamma_{i,j} T_i(x) T_j(y)$ and putting the roots of $T_{m+1}(x)$ in equations (2.2), the linear system $\boldsymbol{V}\boldsymbol{\Gamma} = \boldsymbol{G}$ is obtained, where $\boldsymbol{V} \in M_{(n_1+n_2)(m+1),(m+1)^2}$ and $\boldsymbol{G} \in \mathbb{R}^{(n_1+n_2)(m+1)}$.

step 4: Let $rank(\boldsymbol{V}) = r$ and $\{\boldsymbol{V}_{i_1}, \ldots, \boldsymbol{V}_{i_r}\}$ be independent rows of $\boldsymbol{V}$. We define $\mathbf{A} := [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{(m+1)^2-r}, \boldsymbol{V}_{i_1}, \ldots, \boldsymbol{V}_{i_r}]^t \in M_{(m+1)^2}$ and $\mathbf{b} \in \mathbb{R}^{(m+1)^2}$. By solving the linear system $\mathbf{A}\boldsymbol{\Gamma} = \mathbf{b}$, the coefficients of $v(x, y)$ can be obtained.

2.2. **Kronecker Implementation of Chebyshev Spectral Collocation (KCSC) method for n-order linear PDEs.** In this subsection, we present a Kronecker implementation of Chebyshev spectral collocation Method for equations (2.1-2.2) in Theorems 2.1 and 2.2.

Theorem 2.1. *If* $v(x, y) = \sum_{i,j=0}^{m} \gamma_{i,j} T_i(x) T_j(y) = \boldsymbol{\Gamma}^t(\boldsymbol{T}(x) \otimes \boldsymbol{T}(y))$ *and* $p_i, i = 1, 2, \ldots, m + 1$ *are the roots of* $T_{m+1}(x)$ *and* $\boldsymbol{M}_{n_1,n_2}(x, y) := \sum_{k,l=0}^{n_1,n_2} a_{k,l}$ $(x, y)(D^k \otimes D^l)(\boldsymbol{T}(x) \otimes \boldsymbol{T}(y))$, *then Implementation of Kronecker Chebyshev Spectral Collocation (KCSC) method for equation (2.1) is*

$$\hat{\boldsymbol{A}}\boldsymbol{\Gamma} = \boldsymbol{F},$$

where

$$\hat{\boldsymbol{A}} = \left[\begin{array}{cccc} \boldsymbol{M}_{n_1,n_2}(p_1,p_1) \dots & \boldsymbol{M}_{n_1,n_2}(p_1,p_{m+1}) \dots & \boldsymbol{M}_{n_1,n_2}(p_{m+1},p_{m+1}) \end{array} \right]^t \in M_{(m+1)^2}$$

and

$$\boldsymbol{F} = \left[\begin{array}{cccccccc} f(p_1,p_1) & f(p_1,p_2) & \dots & f(p_1,p_{m+1}) & f(p_2,p_1) & \dots & f(p_{m+1},p_{m+1}) \end{array} \right]^t \in \mathbb{R}^{(m+1)^2}.$$

Proof. By replacing $v(x,y) = \sum_{i,j=0}^{m} \gamma_{i,j} T_i(x) T_j(y)$ in the equation (2.1), we have

$$\sum_{i,j=0}^{m} \sum_{k,l=0}^{n_1,n_2} a_{k,l}(x,y)\gamma_{i,j} T_i^{(k)}(x) T_j^{(l)}(y) = \sum_{k,l=0}^{n_1,n_2} a_{k,l}(x,y) \boldsymbol{\Gamma}^t (D^k \otimes D^l)(\mathbf{T}(x) \otimes \mathbf{T}(y))$$
$$= \boldsymbol{\Gamma}^t \sum_{k,l=0}^{n_1,n_2} a_{k,l}(x,y)(D^k \otimes D^l)(\mathbf{T}(x) \otimes \mathbf{T}(y)) = f(x,y).$$

Then

$$\boldsymbol{\Gamma}^t \mathbf{M}_{n_1,n_2}(x,y) = \mathbf{M}^t_{n_1,n_2}(x,y)\boldsymbol{\Gamma} = f(x,y). \tag{2.7}$$

By replacing the roots (p_i, p_j), $i,j = 1,2,...,m+1$, in the relation (2.7) the result holds. □

Theorem 2.2. *If* $v(x,y) = \sum_{i,j=0}^{m} \gamma_{i,j} T_i(x) T_j(y) = \boldsymbol{\Gamma}^t(\boldsymbol{T}(x) \otimes \boldsymbol{T}(y))$ *and* $p_i, i = 1,2,\dots,m+1$ *are the roots of* $T_{m+1}(x)$*, then the following system of linear equations is obtained by KCSC method of the boundary conditions in equations (2.2).*

$$\tilde{\boldsymbol{A}}\boldsymbol{\Gamma} = \boldsymbol{G}, \tag{2.8}$$

where

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \tilde{\boldsymbol{A}}_1 \\ \tilde{\boldsymbol{A}}_2 \\ \tilde{\boldsymbol{A}}_3 \\ \tilde{\boldsymbol{A}}_4 \end{bmatrix}^t \in M_{(n_1+n_2)(m+1),(m+1)^2}, \quad \boldsymbol{G} = \begin{bmatrix} \tilde{\boldsymbol{G}}_1 \\ \tilde{\boldsymbol{G}}_2 \\ \tilde{\boldsymbol{G}}_3 \\ \tilde{\boldsymbol{G}}_4 \end{bmatrix} \in \mathbb{R}^{(n_1+n_2)(m+1)},$$

$$\tilde{\boldsymbol{A}}_1 = \begin{bmatrix} \tilde{\boldsymbol{A}}_{11} \\ \tilde{\boldsymbol{A}}_{21} \\ \vdots \\ \tilde{\boldsymbol{A}}_{\left[\frac{n_1+1}{2}\right]1} \end{bmatrix}, \quad \tilde{\boldsymbol{A}}_{k1} = \begin{bmatrix} (D^{k-1} \otimes I)(\tilde{\boldsymbol{e}} \otimes \boldsymbol{T}(p_1)) \\ (D^{k-1} \otimes I)(\tilde{\boldsymbol{e}} \otimes \boldsymbol{T}(p_2)) \\ \vdots \\ (D^{k-1} \otimes I)(\tilde{\boldsymbol{e}} \otimes \boldsymbol{T}(p_{m+1})) \end{bmatrix}, \quad k = 1,2,\dots,\left[\frac{n_1+1}{2}\right].$$

$$\tilde{\boldsymbol{G}}_1 = \begin{bmatrix} \tilde{\boldsymbol{G}}_{11} \\ \tilde{\boldsymbol{G}}_{21} \\ \vdots \\ \tilde{\boldsymbol{G}}_{\left[\frac{n_1+1}{2}\right]1} \end{bmatrix}, \quad \tilde{\boldsymbol{G}}_{k1} = \begin{bmatrix} g_{k-1,1}(p_1) \\ g_{k-1,1}(p_2) \\ \vdots \\ g_{k-1,1}(p_{m+1}) \end{bmatrix}, \quad k = 1,2,\dots,\left[\frac{n_1+1}{2}\right].$$

$$\tilde{\boldsymbol{A}}_2 = \begin{bmatrix} \tilde{\boldsymbol{A}}_{12} \\ \tilde{\boldsymbol{A}}_{22} \\ \vdots \\ \tilde{\boldsymbol{A}}_{\left[\frac{n_1}{2}\right]2} \end{bmatrix}, \quad \tilde{\boldsymbol{A}}_{k2} = \begin{bmatrix} (D^{k-1} \otimes I)(\boldsymbol{e} \otimes \boldsymbol{T}(p_1)) \\ (D^{k-1} \otimes I)(\boldsymbol{e} \otimes \boldsymbol{T}(p_2)) \\ \vdots \\ (D^{k-1} \otimes I)(\boldsymbol{e} \otimes \boldsymbol{T}(p_{m+1})) \end{bmatrix}, \quad k = 1,2,\dots,\left[\frac{n_1}{2}\right].$$

$$\tilde{\boldsymbol{G}}_{\boldsymbol{2}} = \begin{bmatrix} \tilde{\boldsymbol{G}}_{12} \\ \tilde{\boldsymbol{G}}_{22} \\ \vdots \\ \tilde{\boldsymbol{G}}_{[\frac{n_1}{2}]2} \end{bmatrix}, \quad \tilde{\boldsymbol{G}}_{k2} = \begin{bmatrix} g_{k-1,2}(p_1) \\ g_{k-1,2}(p_2) \\ \vdots \\ g_{k-1,2}(p_{m+1}) \end{bmatrix}, \ k = 1, 2, \ldots, \left[\frac{n_1}{2}\right].$$

$$\tilde{\boldsymbol{A}}_{\boldsymbol{3}} = \begin{bmatrix} \tilde{\boldsymbol{A}}_{13} \\ \tilde{\boldsymbol{A}}_{23} \\ \vdots \\ \tilde{\boldsymbol{A}}_{[\frac{n_2+1}{2}]3} \end{bmatrix} \quad \tilde{\boldsymbol{A}}_{l3} = \begin{bmatrix} (I \otimes D^{l-1})(\boldsymbol{T}(p_1) \otimes \tilde{\boldsymbol{e}}) \\ (I \otimes D^{l-1})(\boldsymbol{T}(p_2) \otimes \tilde{\boldsymbol{e}}) \\ \vdots \\ (I \otimes D^{l-1})(\boldsymbol{T}(p_{m+1}) \otimes \tilde{\boldsymbol{e}}) \end{bmatrix}, \ l = 1, 2, \ldots, \left[\frac{n_2+1}{2}\right].$$

$$\tilde{\boldsymbol{G}}_{\boldsymbol{3}} = \begin{bmatrix} \tilde{\boldsymbol{G}}_{13} \\ \tilde{\boldsymbol{G}}_{23} \\ \vdots \\ \tilde{\boldsymbol{G}}_{[\frac{n_2+1}{2}]3} \end{bmatrix}, \ \tilde{\boldsymbol{G}}_{l3} = \begin{bmatrix} h_{l-1,1}(p_1) \\ h_{l-1,1}(p_2) \\ \vdots \\ h_{l-1,1}(p_{m+1}) \end{bmatrix}, \ l = 1, 2, \ldots, \left[\frac{n_2+1}{2}\right].$$

$$\tilde{\boldsymbol{A}}_{\boldsymbol{4}} = \begin{bmatrix} \tilde{\boldsymbol{A}}_{14} \\ \tilde{\boldsymbol{A}}_{24} \\ \vdots \\ \tilde{\boldsymbol{A}}_{[\frac{n_2}{2}]4} \end{bmatrix}, \ \tilde{\boldsymbol{A}}_{l4} = \begin{bmatrix} (I \otimes D^{l-1})(\boldsymbol{T}(p_1) \otimes \boldsymbol{e}) \\ (I \otimes D^{l-1})(\boldsymbol{T}(p_2) \otimes \boldsymbol{e}) \\ \vdots \\ (I \otimes D^{l-1})(\boldsymbol{T}(p_{m+1}) \otimes \boldsymbol{e}) \end{bmatrix}, \ l = 1, \ldots, \left[\frac{n_2}{2}\right].$$

$$\tilde{\boldsymbol{G}}_{\boldsymbol{4}} = \begin{bmatrix} \tilde{\boldsymbol{G}}_{14} \\ \tilde{\boldsymbol{G}}_{24} \\ \vdots \\ \tilde{\boldsymbol{G}}_{[\frac{n_2}{2}]4} \end{bmatrix}, \quad \tilde{\boldsymbol{G}}_{l4} = \begin{bmatrix} h_{l-1,2}(p_1) \\ h_{l-1,2}(p_2) \\ \vdots \\ h_{l-1,2}(p_{m+1}) \end{bmatrix}, \ l = 1, 2, \ldots, \left[\frac{n_2}{2}\right].$$

Proof. We will show that the rows of the left and right hand sides of the equation (2.8) are equal. First we prove $\tilde{\boldsymbol{A}}_{k1}\boldsymbol{\Gamma} = G_{k1}$. By replacing $v(x,y) = \sum_{i,j=0}^{m} \gamma_{i,j} T_i(x) T_j(y) = \boldsymbol{\Gamma}^t(\mathbf{T}(x) \otimes \mathbf{T}(y))$ in the first boundary conditions of equation (2.1), we obtain the following:

$\frac{\partial^{k-1} v}{\partial x^{k-1}}(-1, y) = \sum_{i,j=0}^{m} \gamma_{i,j}^{(k-1,0)} T_i(-1) T_j(y) = \sum_{i,j=0}^{m} \gamma_{i,j}^{(k-1,0)} (-1)^i T_j(y) = g_{k-1,1}(y).$

Therefore, $v(-1, y)$ is equal to the following

$$\begin{aligned} (\boldsymbol{\Gamma}^{(\boldsymbol{k-1,0})})^t [T_0(y) \cdots T_m(y) \ \mid -T_0(y) \cdots - T_m(y) \mid \cdots \mid (-1)^m T_0(y) \cdots (-1)^m T_m(y)]^t \\ = (\boldsymbol{\Gamma}^{(\boldsymbol{k-1,0})})^t [(-1)^0 \boldsymbol{T}(y) \mid (-1)^1 \boldsymbol{T}(y) \mid \cdots \mid (-1)^m \boldsymbol{T}(y)]^t \\ = \boldsymbol{\Gamma}([(-1)^0 \ (-1)^1 \ \cdots \ (-1)^m]^t \otimes \boldsymbol{T}(y))(D^{k-1} \otimes I) = g_{k-1,1}(y). \end{aligned}$$

Then

$$\boldsymbol{\Gamma}^t(\tilde{\boldsymbol{e}} \otimes \boldsymbol{T}(y))(D^{k-1} \otimes I) = g_{k-1,1}(y). \tag{2.9}$$

By replacing the roots p_i, $i = 1, 2, ..., m+1$, in equation (2.9) and transposed of last system , $\tilde{\boldsymbol{A}}_{k1}\boldsymbol{\Gamma} = G_{k1}$. By the same argument as above for the other block rows, the result holds. □

In the following remark, computational complexity of KCSC and CSC methods are compared.

Remark 2.3. Let $A \in M_{m\times n}$ and $B \in M_{p\times k}$. Then the number of operations to obtain $A \otimes B$ is denoted by $num(A \otimes B) = mnpk$. In special case if $n = p$, then the number of operations to obtain $A\times B$ is denoted by $num(A\times B) = mnk+mk(n-1) = mk(2n-1)$. According to Theorem 2.1 and [9, relation (17)], for KCSC method, the coefficient matrix $\hat{A} \in M_{(m+1)^2}$ is obtained by $(n_1+n_2)\big((m+1)^4+2(m+1)^2\big)+(m+1)^4-2(m+1)^2$ operations. But each row of the coefficient matrix $\hat{A} \in M_{(m+1)^2}$ in CSC method needs $3(n_1+n_2+1)(m+1)^2$ operations. Then the coefficient matrix $\hat{A} \in M_{(m+1)^2}$ in CSC method is obtained by $(m+1)^2 \times \big(3(n_1+n_2+1)(m+1)^2\big) = \big(3(n_1+n_2+1)(m+1)^4\big)$ operations. Also by Theorem 2.2 and [9, relation (22)], the coefficient matrix $\tilde{A} \in M_{(n_1+n_2)(m+1),(m+1)^2}$ is obtained by $12(m+1)^4-2(m+1)^2$ and $(n_1+n_2)\big(3(m+1)^4-(m+1)^3\big)$ operations in KCSC and CSC methods, respectively. As a result, the computational complexity in KCSC method is less than CSC method.

The algorithm for the KCSC method is described in the following:

Algorithm 1 Coding algorithm for the KCSC method

Input: $m \in \mathbb{N}$, $\{p_1,\dots,p_{m+1}\}$ be the roots of $T_{m+1}(x)$, the coefficients, the known functions of equation (2.1) and equations (2.2).
1: Production of matrix system $\hat{\boldsymbol{A}}\boldsymbol{\Gamma} = \boldsymbol{F}$ by using Theorem 2.1 .
2: Production of matrix system $\tilde{\boldsymbol{A}}\boldsymbol{\Gamma} = \boldsymbol{G}$ by using Theorem 2.2 .
3: Formation matrix system $\boldsymbol{A}\boldsymbol{\Gamma} = \boldsymbol{b}$ by using independent rows of systems $\hat{\boldsymbol{A}}\boldsymbol{\Gamma} = \boldsymbol{F}$ and $\tilde{\boldsymbol{A}}\boldsymbol{\Gamma} = \boldsymbol{G}$, in accordance with **step 4** in subsection **2.1** .
4: Calculate the coefficients Γ by solving the linear system $\boldsymbol{A}\boldsymbol{\Gamma} = \boldsymbol{b}$.
Output: Approximate solution of equation (2.1).

3. Error analysis

In this section, the error analysis of the mentioned methods CSC and KCSC are compared. First, we state some notations which will be used in the sequel. According to Theorems (2.1-2.2), if $rank(\tilde{\boldsymbol{A}}) = r$ and $\{\tilde{\boldsymbol{A}}_{i_1},\dots,\tilde{\boldsymbol{A}}_{i_r}\}$ are independent rows of $\tilde{\boldsymbol{A}}$. We define $\mathbf{A} := [\hat{\boldsymbol{A}}_1,\dots,\hat{\boldsymbol{A}}_{m+1-r},\tilde{\boldsymbol{A}}_{i_1},\dots,\tilde{\boldsymbol{A}}_{i_r}]^t \in M_{(m+1)^2}$ and $\mathbf{b} \in \mathbb{R}^{(m+1)^2}$. By solving the linear system $\mathbf{A}\boldsymbol{\Gamma} = \mathbf{b}$, the coefficients $\gamma_{i,j}$ of $v(x,y)$ can be obtained.

Theorem 3.1. [14, Theorem 5.1.1] *Let $R = [-1,1]\times[-1,1]$ and $I = \{(x_i,y_j) : 0 \le i \le N, 0 \le j \le M\} \in R$ is a set of interpolation nodes. $I_{N,M}u$ is the interpolating polynomial of u through the grid I. Assume that $\dfrac{\partial^{N+1}u}{\partial x^{N+1}}$ and $\dfrac{\partial^{M+1}u}{\partial y^{M+1}}$ exist and continuous for all $(x,y) \in R$. Then, for any $(x,y) \in R$,*

$$|u(x,y)-I_{N,M}u(x,y)| \le \frac{|\omega_N(x)|}{(N+1)!}max_{-1\le x,y\le 1}\left|\frac{\partial^{N+1}u(x,y)}{\partial x^{N+1}}\right| + \Lambda_N(x)\frac{|\omega_M(y)|}{(M+1)!}max_{-1\le x,y\le 1}\left|\frac{\partial^{M+1}u(x,y)}{\partial y^{M+1}}\right|,$$

where
$\Lambda_N(x) = \sum_{i=0}^{N} |L_{i,N}(x)|$, $\omega_N(x) = \prod_{i=0}^{N}(x - x_i)$ *and* $\omega_M(y) = \prod_{i=0}^{M}(y - y_i)$.

Theorem 3.2. *Let the exact solution $u(x, y)$ of Equations (2.1-2.2) be sufficiently smooth and let $v(x, y) = \sum_{i,j=0}^{m} \gamma_{i,j} T_i(x) T_j(y)$. Then for any matrix norm $\| \cdot \|$, the following holds. (For abbreviation, the upper bound in the right hans side is denoted by $\mathcal{U}_m^1$).*

$$|u(x, y) - v(x, y)| \leq \| K_{m,m}(x, y) \| + \| T(x) \| \| T(y) \| \| \triangle \mathbf{b} \| \| \mathbf{A^{-1}} \| = \mathcal{U}_{\mathbf{m}}^{\mathbf{1}}.$$

Proof. By Theorem 3.1, we can write $u(x, y) = I_{m,m}u(x, y) + K_{m,m}(x, y)$, where $I_{m,m}u = \tilde{\mathbf{\Gamma}}^{\boldsymbol{t}}(\mathbf{T}(x) \otimes \mathbf{T}(y))$ is the interpolating polynomial of u and

$$K_{m,m}u(x, y) = \frac{\omega_m(x)}{(m+1)!} \frac{\partial^{m+1} u(\xi, y)}{\partial x^{m+1}} + \Lambda_m(x) \frac{\omega_m(y)}{(m+1)!} \frac{\partial^{m+1} u(x, \eta)}{\partial y^{m+1}}, \ -1 \leq \xi, \eta \leq 1.$$

Since $u(x, y)$ is the exact solution to Equations (2.1-2.2), coefficients of Chebyshev expansion $u(x, y)$ apply to $\mathbf{A\Gamma} = \mathbf{b}$. Similarly, coefficients $v(x, y)$ apply to $\mathbf{A\tilde{\Gamma}} = \mathbf{b} + \triangle\mathbf{b}$. Thus

$$\mathbf{\Gamma} - \tilde{\mathbf{\Gamma}} = \mathbf{A}^{-1} \triangle \mathbf{b}.$$

On the other hand

$$|u(x, y) - v(x, y)| \leq |u(x, y) - I_{m,m}u(x, y)| + |v(x, y) - I_{m,m}u(x, y)|.$$

The first term on the right-hand side can be bounded by Theorem 3.1. So it is enough, we find an upper bound for the second term. Thus

$$\begin{aligned} |v(x, y) - I_{m,m}u(x, y)| &= | (\mathbf{T}(x) \otimes \mathbf{T}(y))\mathbf{\Gamma} - (\mathbf{T}(x) \otimes \mathbf{T}(y))\tilde{\mathbf{\Gamma}} | \\ &\leq \| \mathbf{T}(x) \otimes \mathbf{T}(y) \| \| \mathbf{\Gamma} - \tilde{\mathbf{\Gamma}} \| \leq \| \mathbf{T}(x) \otimes \mathbf{T}(y) \| \| \triangle \mathbf{b} \| \| \mathbf{A}^{-1} \|, \end{aligned}$$

According to relation $\| A \otimes B \| \leq \| A \| \| B \|$ for matrix norms, the proof is complete. □

In the following theorem, Yuksel presented an error upper bound for CSC method.

Theorem 3.3. [9, Theorem 5.1] *Let $v(x, y)$ be the Chebyshev series solution and $u(x, y)$ be the exact solution of Equations 2.3 -2.4, if $u(x, y)$ is sufficiently smooth, then*

$$|u(x, y) - v(x, y)| \leq \| K_{m,m}(x, y) \| + \| \mathbf{T}(x) \| \| \mathbf{Q}(y) \| \| \triangle \mathbf{b} \| \| \boldsymbol{A^{-1}} \| = \mathcal{U}_m^2,$$

where

$$\mathbf{Q}(y) = dig(\boldsymbol{T}(y), \boldsymbol{T}(y), \ldots, \boldsymbol{T}(y)) \in M_{(m+1),(m+1)^2}. \tag{3.1}$$

Remark 3.4. Let $X \in \mathbb{R}^n$ and $A = diag(X^t, X^t, ..., X^t) \in M_{n,n^2}$. Then easy computation shows that $\| A \|_1 = \| X \|_\infty$, $\| A \|_F = \sqrt{n} \| X \|_F$ and $\| A \|_\infty = \| X \|_1$,

where

$$\| A \|_\infty = max\{\sum_{j=1}^{n} | a_{i,j} |,\ i = 1, 2, ..., n\},\ \| X \|_\infty = max\{| x_i |,\ i = 1, 2, ..., n\},$$

$$\| A \|_1 = max\{\sum_{i=1}^{n} | a_{i,j} |,\ j = 1, 2, ..., n^2\}, \qquad \| X \|_1 = \sum_{i=1}^{n} | x_i |,$$

$$\| A \|_F = \sqrt{\sum_{i,j=1}^{n} | a_{i,j}^2 |},\ \| X \|_F = \sqrt{\sum_{i=1}^{n} | x_i^2 |}\ .$$

Moreover, if $X \in \mathbb{R}^n$ has at least two non-zero components, then $\| X \|_\infty < \| X \|_1 = \| A \|_\infty$.

Now, in the following we will compare the new upper bound in Theorem 3.2 with the upper bound presented in Theorem 3.3.

Theorem 3.5. *Let $\mathcal{U}_m^1$ and $\mathcal{U}_m^2$ be the error upper bounds in Theorem 3.2 and Theorem 3.3, respectively. If we consider the norm $\| \cdot \|$ in Theorems 3.2-3.3 as infinity norm $\| \cdot \|_\infty$ or Frobenius norm $\| \cdot \|_F$, then $\mathcal{U}_m^1 < \mathcal{U}_m^2$.*

Proof. Since in general $\boldsymbol{T}(y)$ has at least two nonzero components, by choosing $X = \boldsymbol{T}(y)$ and $A = \boldsymbol{Q}(y)$ in Remark 3.4, we obtain that $\| \boldsymbol{T}(y) \|_\infty < \| \boldsymbol{Q}(y) \|_\infty$ and $\| \boldsymbol{Q}(y) \|_F = \sqrt{n} \| \boldsymbol{T}(y) \|_F$. Therefore, the result holds. □

4. Numerical results

In this section, by calculating the CPU times and Run times for three examples, we show that the CPU times and Run times of KCSC method is significantly less than the CPU times and Run times of CSC method. Also, by statistical tools, we show that the Run times of the KCSC method has polynomial growth and the Run times of the CSC method has exponential growth. All computations in the following examples are performed using Matlab software "2014a" and a system with the specifications: Intel(R) Core(TM) i5 - 4200U and RAM: 8.00 GB.

4.1. **Numerical examples.** The following examples show that although the absolute errors of the KCSC and CSC methods are the same up to six digits, but the CPU times and Run times of KCSC method is significantly less than the CPU and Run times of CSC method. In Tables 1-3, in the first and second columns, the CPU times of the mentioned methods for $m = 4, \ldots, 15$ are computed, in the third and fourth columns, the Run times and in the fifth and sixth columns, the absolute errors of mentioned methods for $m = 4, \ldots, 15$ are presented. Also, we depict the absolute errors in Figures 1, 3, 5, and the CPU times of these implementations are compared in Figures 2(a), 4(a), 6(a). Moreover, the Run times of these implementations are compared in the Figures 2(b), 4(b) and 6(b). Note that, to compute the CPU times and Run times of Tables 1 - 3, the Matlab codes were compiled 10 times and the results were averaged.

Example 4.1. *We consider the following Poisson's equation with exact solution* $u(x,y) = sin(\pi x)sin(\pi y)$.

$$\begin{aligned} u_{xx} + u_{yy} &= -2\pi^2 sin(\pi x)sin(\pi y), \\ u(x,-1) = u(x,1) &= 0, \ -1 \le x \le 1, \\ u(-1,y) = u(1,y) &= 0, \ -1 \le y \le 1. \end{aligned}$$

TABLE 1. Absolute errors, CPU and Run times for Example 4.1

m	*CPU times of CSC method*	*CPU times of KCSC method*	*Run times of CSC method*	*Run times of KCSC method*	*Absolute error of CSC method*	*Absolute error of KCSC method*
4	2.1703×10^1	9.0313×10^0	5.4×10^1	4.2×10^1	7×10^{-1}	7×10^{-1}
5	5.3609×10^1	1.7516×10^1	1.68×10^2	1.33×10^2	2.1×10^{-1}	2.1×10^{-1}
6	2.0342×10^2	9.9953×10^1	2.22×10^2	1.41×10^2	3.2×10^{-2}	3.2×10^{-2}
7	2.4925×10^2	1.0927×10^2	4.43×10^2	2.41×10^2	1.25×10^{-2}	1.25×10^{-2}
8	5.6116×10^2	1.2503×10^2	7.51×10^2	4.28×10^2	6.2×10^{-4}	6.2×10^{-4}
9	1.0861×10^3	2.6033×10^2	1.45×10^3	8.71×10^2	3.75×10^{-5}	3.75×10^{-5}
10	1.9926×10^3	6.6877×10^2	1.846×10^3	8.81×10^2	3.8×10^{-7}	3.8×10^{-7}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	1.1634×10^4	2.6040×10^3	1.2202×10^4	4.733×10^3	3.2×10^{-9}	3.2×10^{-9}

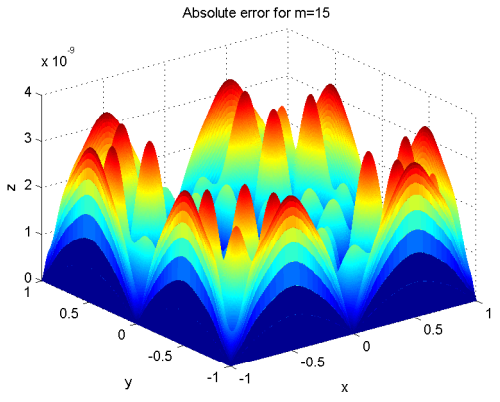
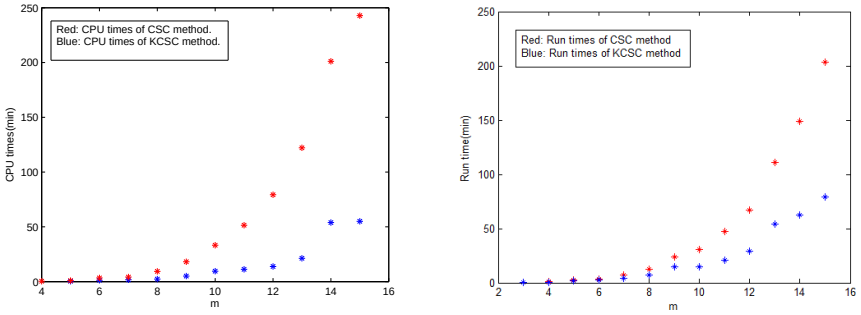


FIGURE 1. Absolute errors of Example 4.1



(a) (b)

FIGURE 2. CPU times (a) and Run times (b) of Example 4.1

Example 4.2. *We consider the following PDE with exact solution* $u(x,y) = e^{x+2y}$.

$$u_{xx} + u_{yy} + u_x + u_y + u = 9e^{x+2y},$$
$$u(x,-1) = e^{x-2},\ u(x,1) = e^{x+2},\ -1 \le x \le 1,$$
$$u(-1,y) = e^{-1+2y},\ u(1,y) = e^{1+2y},\ -1 \le y \le 1.$$

TABLE 2. Absolute errors, CPU and Run times for Example 4.2

m	*CPU times of CSC method*	*CPU times of KCSC method*	*Run times of CSC method*	*Run times of KCSC method*	*Absolute error of CSC method*	*Absolute error of KCSC method*
4	5.8890×10^1	1.8093×10^1	5.1×10^1	4.0×10^1	7.8×10^{-1}	7.8×10^{-1}
5	2.0034×10^2	6.3609×10^1	1.64×10^2	1.31×10^2	1.8×10^{-1}	1.8×10^{-1}
6	2.7384×10^2	9.7640×10^1	2.2×10^2	1.4×10^2	3.5×10^{-2}	3.5×10^{-2}
7	5.2176×10^2	1.1245×10^2	4.14×10^2	2.38×10^2	5.8×10^{-3}	5.8×10^{-3}
8	9.7975×10^2	2.3979×10^2	7.46×10^2	4.25×10^2	8×10^{-4}	8×10^{-4}
9	1.9235×10^3	6.5723×10^2	1.442×10^3	8.66×10^2	9.5×10^{-5}	9.5×10^{-5}
10	2.5267×10^3	4.9076×10^2	1.840×10^3	8.78×10^2	1.02×10^{-5}	1.02×10^{-5}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	1.2087×10^4	2.5949×10^3	1.2189×10^4	4.722×10^3	1.25×10^{-9}	1.25×10^{-9}

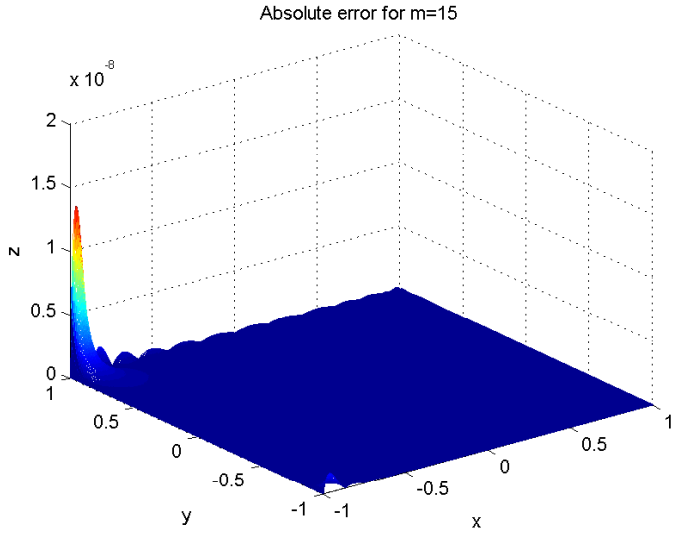
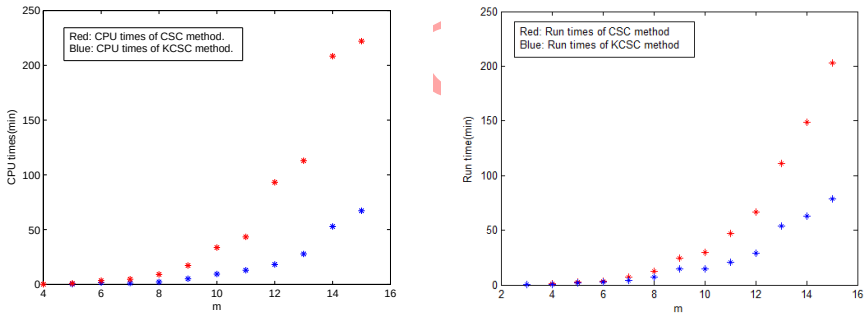


FIGURE 3. Absolute errors of Example 4.2



(a) (b)

FIGURE 4. CPU times (a) and Run times (b) of Example 4.2

Example 4.3. *We consider the following PDE with exact solution* $u(x,y) = e^x siny$.

$$u_{xx} - u_{yy} + (x+y)u_x - x^2 u_y + u = (3+x+y)e^x siny - x^2 e^x cosy,$$
$$u(x,-1) = e^x sin(-1),\ u(x,1) = e^x sin(1),\ -1 \le x \le 1,$$
$$u(-1,y) = e^{-1} siny,\ u(1,y) = e^1 siny,\ -1 \le y \le 1.$$

TABLE 3. Absolute errors, CPU and Run times for Example 4.3

m	*CPU times of CSC method*	*CPU times of KCSC method*	*Run times of CSC method*	*Run times of KCSC method*	*Absolute error of CSC method*	*Absolute error of KCSC method*
4	5.489×10^{1}	1.8109×10^{1}	6.5×10^{0}	5.4×10^{1}	2×10^{-2}	2×10^{-2}
5	2.0782×10^{2}	1.0203×10^{2}	1.89×10^{2}	1.52×10^{2}	1.2×10^{-2}	2.1×10^{-1}
6	2.8594×10^{2}	1.1175×10^{2}	2.56×10^{2}	1.69×10^{2}	2×10^{-3}	2×10^{-3}
7	5.4670×10^{2}	1.3178×10^{2}	4.40×10^{2}	2.75×10^{2}	4×10^{-5}	4×10^{-5}
8	1.0353×10^{3}	3.0967×10^{2}	8.1×10^{2}	4.79×10^{2}	4.2×10^{-6}	4.2×10^{-6}
9	2.016×10^{3}	5.9094×10^{2}	1.323×10^{3}	9.03×10^{2}	1.9×10^{-7}	3.75×10^{-5}
10	2.6×10^{3}	7.7761×10^{2}	1.94×10^{3}	9.09×10^{2}	1.8×10^{-7}	1.8×10^{-7}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
15	1.3383×10^{4}	3.173×10^{3}	1.2415×10^{4}	5.001×10^{3}	3.1×10^{-8}	3.1×10^{-8}

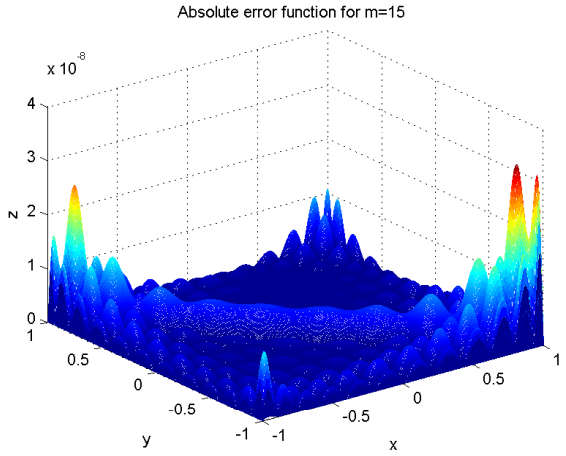
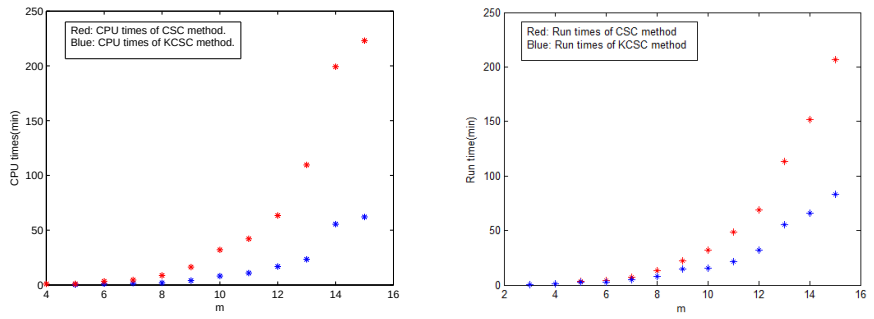


FIGURE 5. Absolute errors of Example 4.3



(a) (b)

FIGURE 6. CPU times (a) and Run times (b) of Example 4.3

Comparison of the Run times obtained in the above three examples show that the Run times of KCSC method has much lower than Run times of CSC method.

In the last part of this section, in the Tables 4 and 5, for examples 4.1-4.3, the error bounds $\mathcal{U}_m^1$ and $\mathcal{U}_m^2$ for infinity norm and Frobenius norm are computed , respectively.

TABLE 4. Error upper bounds $\mathcal{U}_m^1$ and $\mathcal{U}_m^2$(infinity norm)

m	$\mathcal{U}_m^1$ of Example 4.1	$\mathcal{U}_m^2$ of Example 4.1	$\mathcal{U}_m^1$ of Example 4.2	$\mathcal{U}_m^2$ Example 4.2	$\mathcal{U}_m^1$ of Example 4.3	$\mathcal{U}_m^2$ of Example 4.3
5	1.7213	10.3088	1.2571	7.4176	1.7865	10.6690
10	1.3244	10.5415	1.0234	11.0554	1.2357	13.4947
15	0.9885	15.8025	0.8868	13.9008	1.0160	16.1255

TABLE 5. Error upper bounds $\mathcal{U}_m^1$ and $\mathcal{U}_m^2$(Frobenius norm)

m	$\mathcal{U}_m^1$ of Example 4.1	$\mathcal{U}_m^2$ of Example 4.1	$\mathcal{U}_m^1$ of Example 4.2	$\mathcal{U}_m^2$ Example 4.2	$\mathcal{U}_m^1$ of Example 4.3	$\mathcal{U}_m^2$ of Example 4.3
5	16.2538	36.8880	14.1321	29.2573	15.1744	35.2773
10	5.3027	16.7658	4.1278	13.2728	4.8752	15.2119
15	0.3759	1.4533	0.2926	1.4125	0.3275	1.3912

Remark 4.4. Note that error upper bound $\mathcal{U}_m^1$ for KCSC method in Theorem 3.2 can be used for n-order linear PDEs, while error upper bound $\mathcal{U}_m^2$ for CSC method in Theorem 3.3 can only be used for second-order linear PDEs. Moreover, the numerical results of Table 4 show that, the sequence $\{\mathcal{U}_m^1\}$ is decreasing while the sequence $\{\mathcal{U}_m^2\}$ is an increasing sequence. But the numerical results of Table 5 show that, the sequences $\{\mathcal{U}_m^1\}$ and $\{\mathcal{U}_m^2\}$ are decreasing.

4.2. **Statistical results.** Regression is a technique for determining the statistical relationship between two or more variables. Regression is primarily used for prediction and causal inference[13].

Let $\hat{y}_i$ indicates a prediction of y_i on the basis of $x = x_i$ by regression model for $i = 1, 2, ..., n$, we define

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2,\ SST = \sum_{i=1}^{n}(y_i - \bar{y})^2,\ SSR = SST - SSE = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2,$$

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST},\ RMSE = \sqrt{\frac{SSE}{n}}\ .$$

Root Mean Square Error (RMSE) is the standard deviation of the residuals. Residuals are a measure of how far from the regression data points are, RMSE is a measure of how spread out these residuals are. R^2 is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). R^2 has the useful property that its scale is intuitive: it ranges from zero to one, with zero indicating that the proposed model does not improve prediction over the mean model and one indicating perfect prediction [3, 13]. In this section, by regression techniques

for the examples in the previous section, we fit the Run times for $m = 4, 5, ..., 15$ of the mentioned methods by polynomial and exponential models. Then by statistical tools, we compare these regression models and we show that the Run times of KCSC method is much less than the CSC method.

4.2.1. *Statistical results of Example 4.1.* In this subsection statistical results related to Example 4.1 are studied. We compute regression models for the Run times of the mentioned methods, then by using the R^2 and $RMSE$ tools, we find the best regression for the Run times. You can see the results in Tables 6-7.

TABLE 6. Regression equations of KCSC method for Example 4.1

Type of regression	Equation	Value RMSE for KCSC method	Value R^2 for KCSC method
poly1	$p_1(x) = 1002x - 6911$	3265.0	0.7117
poly2	$p_2(x) = 122.4x^2 - 1814x + 5985$	1162.0	0.9635
poly3	$p_3(x) = 9.971x^3 - 221.6x^2 + 1661x - 3647$	441.8	0.9947
poly4	$\mathbf{p_4(x) = .536x^4 - 14.68x^3 + 167x^2 - 755.2x + 1164}$	**360.5**	**0.9965**
poly6	$p_6(x) = -.01173x^6 + .7991x^5 - 20.88x^4 + 271.6x^3 - 1822x^2 + 5961x - 7350$	378.1	0.9961
exponential	$exp(x) = 63.3e^{.2914x}$	419.1	0.9953

TABLE 7. Regression equations of CSC method for Example 4.1

Type of regression	Equation	Value $RMSE$ for CSC method	Value R^2 for CSC method
poly1	$p_1(x) = 2963x - 21280$	11688.0	0.631
poly2	$p_2(x) = 416.7x^2 - 6620x + 22610$	5775.0	0.9097
poly3	$p_3(x) = 45.26x^3 - 1145x^2 + 9151x - 21110$	3089.0	0.9742
poly4	$p_4(x) = 5.003x^4 - 184.9x^3 + 2483x^2 - 13400x + 23800$	1962.0	0.9896
poly6	$p_6(x) = .01124x^6 - 7.063x^5 + 176.1x^4 - 2196x^3 + 14320x^2 - 45730x + 55380$	1928.5	0.9899
exponential	$\mathbf{exp(x) = 39.33e^{.3749x}}$	**1645.0**	**0.9927**

4.2.2. *Statistical results of Example 4.2.* In this subsection statistical results related to Example 4.2 are studied. We compute regression models for the Run times of the mentioned methods, then by using the R^2 and $RMSE$ tools, we find the best regression for the Run times. You can see the results in Tables 8-9.

TABLE 8. Regression equations of KCSC method for Example 4.2

Type of regression	Equation	Value RMSE for KCSC method	Value R^2 for KCSC method
poly1	$p_1(x) = 1001x - 6910$	3267.0	0.7112
poly2	$p_2(x) = 122.4x^2 - 1814x + 5984$	1163.0	0.9634
poly3	$p_3(x) = 9.974x^3 - 221.7x^2 + 1662x - 3650$	442.2	0.9947
poly4	$\mathbf{p_4(x) = .5362x^4 - 14.69x^3 + 167.1x^2 - 755.6x + 1163}$	**360.9**	**0.9965**
poly6	$p_6(x) = -.01173x^6 + .7988x^5 - 20.88x^4 + 271.6x^3 - 1822x^2 + 5964x - 7357$	380.9	0.9961
exponential	$exp(x) = 62.98e^{.2916x}$	419.8	0.9952

TABLE 9. Regression equations of CSC method for Example 4.2

Type of regression	Equation	Value RMSE for CSC method	Value R^2 for CSC method
poly1	$p_1(x) = 2963x - 21280$	11533.0	0.698
poly2	$p_2(x) = 416.6x^2 - 6620x + 22600$	5784.0	0.9094
poly3	$p_3(x) = 45{,}24x^3 - 11.14x^2 + 9145x - 21100$	3089.0	0.9742
poly4	$p_4(x) = 5.005x^4 - 185x^3 + 2484x^2 - 13420x + 23830$	1968.0	0.9895
poly6	$p_6(x) = .1122x^6 - 7.049x^5 + 175.7x^4 - 2193x^3 + 14280x^2 - 45610x + 552200$	4251	0.9511
exponential	$\mathbf{exp(x) = 39.19e^{.3751x}}$	**1645.0**	**0.9927**

4.2.3. *Statistical results of Example 4.3.* In this subsection statistical results related to Example 4.3 are studied. We compute regression models for the Run times of the mentioned methods, then by using the R^2 and $RMSE$ tools, we find the best regression for the Run times. You can see the results in Tables 10-11.

TABLE 10. Regression equations of KCSC method for Example 4.3

Type of regression	Equation	Value RMSE for KCSC method	Value R^2 for KCSC method
poly1	$p_1(x) = 1049x - 7270$	3488	0.7035
poly2	$p_2(x) = 129.9x^2 - 1938x + 6441$	1308.0	0.9583
poly3	$p_3(x) = 11.13x^3 - 254.1x^2 + 1941x - 4312$	519.3	0.9934
poly4	$\mathbf{p_4(x) = .7047x^4 - 21.28x^3 + 256.7x^2 - 1236x + 2013}$	**395.9**	**0.9962**
poly6	$p_6(x) = -.01303x^6 + .9079x^5 - 24.25x^4 + 321.6x^3 - 2197x^2 + 7309x - 9137$	432	0.995
exponential	$exp(x) = 59.68e^{.2972x}$	427.2	0.9956

TABLE 11. Regression equations of CSC method for Example 4.3

Type of regression	Equation	Value RMSE for CSC method	Value R^2 for CSC method
poly1	$p_1(x) = 2988x - 21420$	11700.0	0.6338
poly2	$p_2(x) = 418.1x^2 - 6629x + 22620$	5760.0	0.9113
poly3	$p_3(x) = 45.1x^3 - 1138x^2 + 9088x - 20950$	3086.0	0.9745
poly4	$p_4(x) = 4.989x^4 - 184.4x^3 + 2479x^2 - 13400x + 23840$	1965.0	0.9897
poly6	$p_6(x) = .1115x^6 - 6.995x^5 + 174.1x^4 - 2167x^3 + 14090x^2 - 44910x + 54290$	4239.5	0.9519
exponential	$\mathbf{exp(x) = 42e^{.3719x}}$	**1670.0**	**0.9925**

5. CONCLUSION

Spectral method is one of the numerical methods of exponential order with high convergence rate. But its implementation on PDEs is too complicated. In this note, we implement Kronecker Chebyshev Spectral Collocation (KCSC) method for n-order linear PDEs and we show that the computational complexity of KCSC method is less than of CSC method. By statistical tools on three examples, we show that the best regression model for Run times in CSC method is exponential and the best regression model for Run times in KCSC method is fourth-degree polynomial. Moreover, comparing the CPU and Run times of the proposed methods, show that the CPU and Run times of KCSC method is significantly less than CSC method. Also, for KCSC method, provide an error upper bound and for some matrix norm, we show that this error upper bound is better than Previous error upper bound. According to the structure and implementation of KCSC method, it is possible to implement the KCSC method for semi-linear and nonlinear equations. Additionally, combining KCSC method with other numerical methods (finite difference method or crank nicolson method) can greatly reduce computational complexity.

REFERENCES

[1] A. B. Orovio, V. M. P. Garcia, F. H. Fenton. *Spectral Methods for Partial Differential Equations in Irregular Domains: The Spectral Smoothed Boundary Method*, SIAM Journal on Scientific Computing, 28(3) (2006) 886-900.

[2] A. G . Atta, W. M. A. Elhameed, Y. H. Youssri, *Shifted fifth-kind Chebyshev Galerkin treatment for linear hyperbolic first-order partial differential equations*, Appl. Numerical Math. **167** (2021) 237-256.

[3] A. S. Hadi, *Regression Analysis By Example*, WILEY, New Jersey, 2012.

[4] C. Johnson, R. A. Horn *Matrix Analysis*, Cambridge Univ. Press, 2013.

[5] C. K. san, *Chebyshev polynomial solutions of second-order linear partial differential equations*, Appl. Math. Comput. **134** (2003), 109-124.

[6] D. Johnson, *Chebyshev polynomial in the Tau spectral methods and applications to eigenvalue problems*, National Aeronautics and Space Administration, 1996.

[7] D. V. Hutton, *FUNDAMENTALS OF FINITE ELEMENT ANALYSIS*, Elizabeth A .Jones (2004) 721-732.

[8] G. Evans, J. Blackledge, P. Yardley, *Analytic Methods for Partial Differential Equations*, Springer, 1999.

[9] G. Yuksel, O. R. Isik, M. Sezer, *Error analysis of the Chebyshev collocation method for linear second order partial differential equations, Internat. Journal of Comput. Math.* **43**, (2015) 2261-2268.

[10] I. Celik, *Collocation method and residual correction using Chebyshev series*, Appl. Math. Comput. **174** (2006) 910-920.

[11] J. Kevorkian, *Partial Differential Equations: Analytical Solution Techniques*, Springer, 1999.

[12] J. Mason, C. Handscomb, *Chebyshev polynomials*, CRC Press, 2013.

[13] J. O. Rawlings, S. G. Pantula, D. K. Dickey, *Applied Regression Analysis*, Springer, 1998.

[14] K. E. Atkinson, *The Numerical Solution of Integral Equations of the Second Kind*, Cambridge Univ. Press, New York, 1997.

[15] K. Goyal, M. Mehra, *Fast diffusion wavelet method for partial differential equations.* Appl. Math. Mod. **40** (2016) 5000-5025.

[16] M. Dehghan, M. Lakestani, *The use of Chebyshev cardinal functions for solution of the second-order one-dimensional telegraph equation*, Num. Methods for Partial Differ. Equations. **25(4)** (2009) 931-938.

[17] M. Izadi, M. Afshar, *Solving the Basset equation via Chebyshev collocation and LDG methods*, Journal of Math. Mod. **9** (2021) 61-79.

[18] M. Lakestani, M. Dehghan, *Numerical solution of fourth-order integro-differential equations using Chebyshev cardinal functions*, Inter. Journal of Com. Math. **87(6)**(2008) 1389-1394.

[19] M. Lakestani, M. Dehghan, *The use of Chebyshev cardinal functions for the solution of a partial differential equation with an unknown time-dependent coefficient subject to an extra measurement*, Journal of Comput. App. Math. **235(3)**(2020) 669-678.

[20] M. Pourbabaee, A. Saadatmandi, *Collocation method based on Chebyshev polynomials for solving distributed order fractional differential equations*, Comput. Methods for Differ. Equations. **9(3)** (2021) 858-873.

[21] S. Akbarpour, A. Shidfar, H. Saberinajafi, *A Shifted Chebyshev-Tau method for finding a time-dependent heat source in heat equation*, Comput. Methods for Differ. Equations. **8(1)**(2020) 1-13.

[22] P. Pedersen, *New Solutions for Singular Lane-Emden Equations Arising in Astrophysics Based on Shifted Ultraspherical Operational Matrices of Derivatives*, Differ. and Integral Equ. (1999) 721-732.

[23] W. M. A. Elhameed, Y. H. Youssri, *New formulas of the high-order derivatives of fifth-kind Chebyshev polynomials: Spectral solution of the convectiondiffusion equation*, Num. Methods for Partial. Differ. Equations. (2021) .

[24] W. M. A. Elhameed, J. A. Tenreiro Machado, Y. H. Youssri, *Hypergeometric fractional derivatives formula of shifted Chebyshev polynomials: tau algorithm for a type of fractional delay differential equations*, Comput. Methods for Differ. Equations. **2(3)** (2014) 171-185.

[25] W. M. A. Elhameed, Y. Youssri, E. H. Doha, *High-order finite element methods for time-fractional partial differential equations* , Comput. Methods for Differ. Equations. **2(3)** (2014) 171-185.

[26] Y. Jianga, J. Ma, *High-order finite element methods for time-fractional partial differential equations* , A Math. Journal (2004) 1-32.

[27] Y. H. Youssri, W. M. A. Elhameed, M. Abdelhakem, *A robust spectral treatment of a class of initial value problems using modified Chebyshev polynomials*, Math. Methods in the Appl. Sciences. **44(11)** (2021) 9224-9236.
[28] Y. H. Youssri, R. M. Hafez, *Chebyshev collocation treatment of VolterraFredholm integral equation with error analysis*, Arabian Journal of Math. **9** (2020) 471-480.