Highly accurate multi-domain multivariate spectral collocation method for (2+1) dimensional Burgers' equation

M. P. Mkhatshwa[1] M. Khumalo [1] & P. G. Dlamini[2]

[1]University of South Africa, South Africa
[2]University of Johannesburg, South Africa

06 - 08 December (SAMS 2022), Stellenbosch University, Cape Town, South Africa

Presentation Outline

Aim

- To describe the development of multi-domain multivariate spectral collocation (SC) method for (2+1) dimensional nonlinear parabolic PDEs defined on large time domains.
- To highlight accuracy and efficiency of the multi-domain multivariate SC method on single Burger's equation.

Why developing of numerical method? Why Burgers equation?

- The need to develop sufficiently accurate, convergent and computationally efficient numerical techniques for solving nonlinear problems is an ever-recurring theme in numerical mathematics.
- The difficulty in solving DEs rises with the number of variables the DE has. Thus, solving multi-dimensional PDEs is more involving.
- Burgers equation finds many applications in engineering, science and industry - turbulence theory [1], modeling traffic flow [2], cosmology [3], gas dynamics [4], shock wave theory [5], etc
- Burgers equation contains a coefficient $\frac{1}{Re}$, which has a significant impact on the accuracy of any method. Hence, to study the effect of Re gives a convincing argument on the performance of the method.

Advantages - SC methods

- SC methods have been shown in literature to be more accurate and efficient than common numerical methods (when solutions are smooth).
- SC method algorithm are easy to learn and implement in scientific computing software.

Limitations - single domain SC methods

- Accuracy of single-domain SC methods increases with an increase in grid points but beyond a certain number of grid points, the accuracy deteriorates rapidly.
- Accuracy also deteriorate as the computational domain (particularly time) increases.

This work reports new improvement that incorporates the non-overlapping multi-domain approach in the time variable when descretizing the solution domain for Chebyshev SC method.

Numerical experimentation

General single nonlinear PDE

We consider a second-order 2D nonlinear PDEs of the form:

$$\frac{\partial u}{\partial t} = G\left(\frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, u\right), \quad (x, y) \in [a, b] \times [c, d], \ t \in [0, T], \tag{1}$$

subject to the initial and Dirichlet boundary conditions

$$\begin{aligned} &u(x, y, 0) = h(x, y), \quad u(a, y, t) = g_a(y, t), \quad u(b, y, t) = g_b(y, t), \\ &u(x, c, t) = g_c(x, t), \quad u(x, d, t) = g_d(x, t), \end{aligned} \tag{2}$$

where G is a nonlinear operator, $h(x, y), g_a(y, t), g_b(y, t), g_c(x, t)$ and $g_d(x, t)$ are known functions.

Numerical experimentation

Specific example

We consider the two-dimensional nonlinear Burger's PDE [6, 7, 8, 9, 10] :

$$\frac{\partial u}{\partial t} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - u\frac{\partial u}{\partial x} - u\frac{\partial u}{\partial y}, \ (x, y) \in [a, b] \times [c, d], \ t > 0 \tag{3}$$

where Re is the Reynold number. The equation is solved subject to the initial and boundary conditions

$$u(x, y, 0) = \frac{1}{1 + e^{Re(x+y)/2}}, \quad u(0, y, t) = \frac{1}{1 + e^{Re(y-t)/2}}, \quad u(2, y, t) = \frac{1}{1 + e^{Re(2+y-t)/2}},$$
$$u(x, 0, t) = \frac{1}{1 + e^{Re(x-t)/2}}, \quad u(x, 2, t) = \frac{1}{1 + e^{Re(2+x-t)/2}}. \tag{4}$$

The exact solution is given by $u(x, y, t) = \frac{1}{1+e^{Re(x+y-t)/2}}$.

Linearisation

Applying quasilinearisation method (QLM) [12] on the general nonlinear PDE (1) gives the iterative scheme:

$$\varpi_{4,k}\frac{\partial^2 u_{k+1}}{\partial x^2} + \varpi_{3,k}\frac{\partial^2 u_{k+1}}{\partial y^2} + \varpi_{2,k}\frac{\partial u_{k+1}}{\partial x} + \varpi_{1,k}\frac{\partial u_{k+1}}{\partial y} + \varpi_{0,k}u_{k+1} - \frac{\partial u_{k+1}}{\partial t} = R_k, , \quad (5)$$

where the variable coefficients are defined as

$$\varpi_{4,k} = \frac{\partial G}{\partial (u_{xx})_k}, \ \varpi_{3,k} = \frac{\partial G}{\partial (u_{yy})_k}, \ \varpi_{2,k} = \frac{\partial G}{\partial (u_x)_k}, \ \varpi_{1,k} = \frac{\partial G}{\partial (u_y)_k}, \ \varpi_{0,k} = \frac{\partial G}{\partial (u)_k} \quad (6)$$

$$R_k = \varpi_{4,k}\frac{\partial^2 u_k}{\partial x^2} + \varpi_{3,k}\frac{\partial^2 u_k}{\partial y^2} + \varpi_{2,k}\frac{\partial^2 u_k}{\partial x} + \varpi_{1,k}\frac{\partial^2 u_k}{\partial y} + \varpi_{0,k}u_k - G_k, \quad (7)$$

with $(u_{xx})_k = \frac{\partial^2 u_k}{\partial x^2}$, $(u_{yy})_k = \frac{\partial^2 u_k}{\partial y^2}$, $(u_x)_k = \frac{\partial u_k}{\partial x}$, $(u_y)_k = \frac{\partial u_k}{\partial y}$, k and $k+1$ stands for previous and current iterations, respectively.

Domain decomposition

Let $t \in I$, where $I \in [0, T]$ is the time interval. Then, the computational domain I is split into p non-overlapping subintervals given by

$$I_\iota = [t_{\iota-1}, t_\iota],\ t_{\iota-1} < t_\iota,\ \iota = 1, 2, 3, \ldots, p,\ \text{with},\ 0 = t_0 < t_1 < t_2 < \ldots < t_p = T, \tag{8}$$

where each sub-interval is discretized using $N_t + 1$ Chebyshev-Gauss-Lobatto (C-G-L) collocation points.

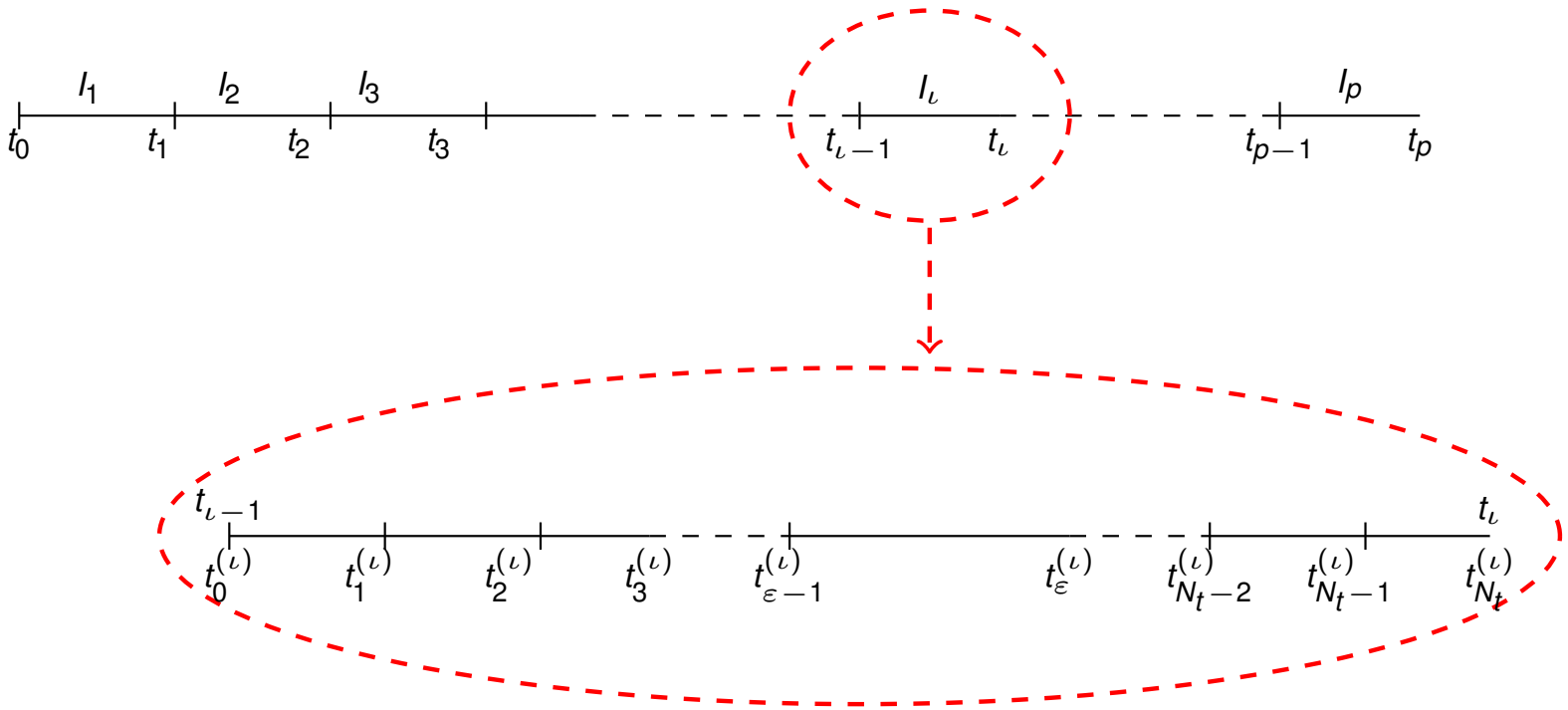


Figure: Non-overlapping grid ($t-$ domain)

Domain transformation

The time domain $t \in [t_{\iota-1}, t_\iota]$ in each ιth subinterval is transformed into $\hat{t} \in [-1, 1]$ using the linear mapping:

$$t_l^\iota = \frac{1}{2}\left(t_\iota - t_{\iota-1}\right)\hat{t}_l + \frac{1}{2}\left(t_\iota + t_{\iota-1}\right), \quad t \in [t_{\iota-1}, t_\iota], \tag{9}$$

where $\left\{\hat{t}_l\right\}_{l=0}^{N_t} = \cos\left(\frac{\pi l}{N_t}\right)$ signifies the collocation points in the ιth subinterval in t variable. The spatial domains $x \in [a, b]$ and $y \in [c, d]$ are transformed into $\hat{x} \in [-1, 1]$ and $\hat{y} \in [-1, 1]$, respectively, by using the linear mappings:

$$x_i = \frac{1}{2}(b-a)\hat{x} + \frac{1}{2}(b+a), \quad y_j = \frac{1}{2}(d-c)\hat{y} + \frac{1}{2}(d+c), \tag{10}$$

where $\{\hat{x}_i\}_{i=0}^{N_x} = \cos\left(\frac{\pi i}{N_x}\right)$ and $\{\hat{y}_j\}_{j=0}^{N_y} = \cos\left(\frac{\pi j}{N_y}\right)$ denote the collocation points which are considered to be symmetrically distributed CGL grid points on the interval $[-1, 1]$ [13, 14].

Continuity conditions

Let $\overset{(\iota)}{u}(x, y, t), \iota = 1, 2, 3, ..., p$, to denote the solutions at various time intervals. Making use of the continuity conditions

$$\overset{(\iota)}{u}(x, y, t_{\iota-1}) = \overset{(\iota-1)}{u}(x, y, t_{\iota-1}) \tag{11}$$

to yield the initial conditions for the subsequent subintervals ($\iota = 2, 3, 4, ..., p$) in t, the multi-domain is applied on the QLM iterative scheme (5). Consequently, we must solve:

$$\varpi_{4,k}\frac{\partial^2 \overset{(\iota)}{u}_{k+1}}{\partial x^2} + \varpi_{3,k}\frac{\partial^2 \overset{(\iota)}{u}_{k+1}}{\partial y^2} + \varpi_{2,k}\frac{\partial \overset{(\iota)}{u}_{k+1}}{\partial x} + \varpi_{1,k}\frac{\partial \overset{(\iota)}{u}_{k+1}}{\partial y} + \varpi_{0,k}\overset{(\iota)}{u}_{k+1} - \frac{\partial \overset{(\iota)}{u}_{k+1}}{\partial t} = \overset{(\iota)}{R}_k, \tag{12}$$

subject to the boundary conditions

$$\overset{(\iota)}{u}_{k+1}(a, y, t) = g_a(y, t), \qquad \overset{(\iota)}{u}_{k+1}(b, y, t) = g_b(y, t), \tag{13}$$

$$\overset{(\iota)}{u}_{k+1}(x, c, t) = g_c(x, t), \qquad \overset{(\iota)}{u}_{k+1}(x, d, t) = g_d(x, t), \tag{14}$$

and initial conditions

$$\overset{(1)}{u}_{k+1}u(x, y, 0) = h(x, y), \quad \overset{(k)}{u}_{k+1}u(x, y, t_{\iota-1}) = \overset{(k-1)}{u}_{k+1}u(x, y, t_{\iota-1}), \text{ for } \iota = 2, 3, 4, \ldots, p. \tag{15}$$

Trivariate Interpolation

The approximate solution is obtained by expanding the unknown $u(x, y, t)$ using the triple Lagrange interpolating polynomial. Thus, the trivariate Lagrange interpolation of $u(x, y, t)$ at each sub-interval I_ι is given by

$$\overset{(\iota)}{u}(x, y, t) \approx \overset{(\iota)}{u}_N(x, y, t) = \sum_{l=0}^{N_t} \sum_{j=0}^{N_y} \sum_{i=0}^{N_x} \overset{(\iota)}{u}_{ljl} L_l(t) L_j(y) L_i(x). \tag{16}$$

The space derivative matrices (first and n^{th}) in x and y and differentiation matrix (first and n^{th}) in time t at the collocation points $(\hat{x}, \hat{y}, \hat{t})$ in each sub-interval I_ι, $\iota = 1, 2, \ldots, p$, are approximated as:

$$\frac{\partial \overset{(\iota)}{u}}{\partial x}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \sum_{\tau=0}^{N_t} \sum_{\varrho=0}^{N_y} \sum_{\varsigma=0}^{N_x} \overset{(\iota)}{u}_{ljl} L_\tau(\hat{t}_l) L_\varrho(\hat{y}_j) D_{\varsigma i}, \quad \frac{\partial^n \overset{(\iota)}{u}}{\partial x^n}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \sum_{\tau=0}^{N_t} \sum_{\varrho=0}^{N_y} \sum_{\varsigma=0}^{N_x} \overset{(\iota)}{u}_{ljl} L_\tau(\hat{t}_l) L_\varrho(\hat{y}_j) D^{(n)}_{\varsigma i}, \tag{17}$$

$$\frac{\partial \overset{(\iota)}{u}}{\partial y}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \sum_{\tau=0}^{N_t} \sum_{\varrho=0}^{N_y} \sum_{\varsigma=0}^{N_x} \overset{(\iota)}{u}_{ljl} L_\tau(\hat{t}_l) D_{\varrho j} L_\varsigma(\hat{x}_i), \quad \frac{\partial^n \overset{(\iota)}{u}}{\partial y^n}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \sum_{\tau=0}^{N_t} \sum_{\varrho=0}^{N_y} \sum_{\varsigma=0}^{N_x} \overset{(\iota)}{u}_{ljl} L_\tau(\hat{t}_l) D^{(n)}_{\varrho j} L_\varsigma(\hat{x}_i), \tag{18}$$

$$\frac{\partial \overset{(\iota)}{u}}{\partial t}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \sum_{\tau=0}^{N_t} \sum_{\varrho=0}^{N_y} \sum_{\varsigma=0}^{N_x} \overset{(\iota)}{u}_{ljl} D_{\tau l} L_\varrho(\hat{y}_j) L_\varsigma(\hat{x}_i), \quad \frac{\partial^n \overset{(\iota)}{u}}{\partial t^n}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \sum_{\tau=0}^{N_t} \sum_{\varrho=0}^{N_y} \sum_{\varsigma=0}^{N_x} \overset{(\iota)}{u}_{ljl} D^{(n)}_{\tau l} L_\varrho(\hat{y}_j) L_\varsigma(\hat{x}_i), \tag{19}$$

where D represent the Chebyshev differentiation matrix associated with the Gauss-Lobatto nodes, and

$D_{\varsigma i} = \frac{dL_\varsigma}{d\hat{x}}(\hat{x}_i)$ for $i, \varsigma = 0, 1, 2, \ldots, N_x$, $D_{\varrho j} = \frac{dL_\varrho}{d\hat{y}}(\hat{y}_j)$ for $j, \varrho = 0, 1, 2, \ldots, N_y$, and $D_{\tau k} = \frac{dL_\tau}{d\hat{z}}(\hat{z}_l)$ for $l, \tau = 0, 1, 2, \ldots, N_t$, are entries of the standard first order Chebyshev differentiation matrices of sizes $(N_x + 1)(N_x + 1)$, $(N_y + 1)(N_y + 1)$ and $(N_t + 1)(N_t + 1)$, respectively.

Kronecker product

For simplicity and efficiency, the Kronecker product [13, 15] (operation that transforms two matrices into a matrix that contains all possible products of the entries of the two matrices) is used to represent the approximate solution in matrix form. Let **A** and **B** be matrices of order $m \times n$, respectively, then the Kronecker product between matrices **A** and **B** is the $(mn \times mn)$ block matrix

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \cdots & A_{1n}\mathbf{B} \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \cdots & A_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}\mathbf{B} & A_{m2}\mathbf{B} & \cdots & A_{mn}\mathbf{B} \end{bmatrix}, \text{ where } A_{mn} \text{ denotes the } (m, n) - \text{th entry of } \mathbf{A}. \tag{20}$$

In MATLAB, the Kronecker product of two matrices **A** and **B** is directly calculated with command *kron*(**A**, **B**). The Kronecker product of three matrices **A**, **B** and **C**, each of orders $m \times n$ is the block matrix

$$\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C} = \begin{bmatrix} A_{11}\mathbf{H} & A_{12}\mathbf{H} & \cdots & A_{1n}\mathbf{H} \\ A_{21}\mathbf{H} & A_{22}\mathbf{H} & \cdots & A_{2n}\mathbf{H} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}\mathbf{H} & A_{m2}\mathbf{H} & \cdots & A_{mn}\mathbf{H} \end{bmatrix}, \tag{21}$$

$$\text{where } \mathbf{H} \text{ is defined as } \mathbf{B} \otimes \mathbf{C} = \begin{bmatrix} B_{11}\mathbf{C} & B_{12}\mathbf{C} & \cdots & B_{1n}\mathbf{C} \\ B_{21}\mathbf{C} & B_{22}\mathbf{C} & \cdots & B_{2n}\mathbf{C} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1}\mathbf{C} & B_{m2}\mathbf{C} & \cdots & B_{mn}\mathbf{C} \end{bmatrix}. \tag{22}$$

In MATLAB, the Kronecker product of three matrices **A**, **B** and **C** is directly evaluated using the command *superkron*(**A**, **B**, **C**), which is a generalization of MATLAB's *kron* function. Superkron allows computation of Kronecker product of three matrices in one shot.

Approximate function & derivative matrices via Kronecker product

It follows that (16) becomes

$$\overset{(\iota)}{u}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \left(L(\hat{t}_l) \otimes L(\hat{y}_j) \otimes L(\hat{x}_i)\right) \overset{(\iota)}{U}, \tag{23}$$

where

$$L(\hat{t}_l) = \left[L_0(\hat{t}_l), L_1(\hat{t}_l), \ldots, L_{N_t}(\hat{t}_l)\right], \; L(\hat{y}_j) = \left[L_0(\hat{y}_j), L_1(\hat{y}_j), \ldots, L_{N_y}(\hat{y}_j)\right],$$

$$L(\hat{x}_i) = \left[L_0(\hat{x}_i), L_1(\hat{x}_i), \ldots, L_{N_x}(\hat{x}_i)\right], \tag{24}$$

$$\overset{(\iota)}{U} = [\overset{(\iota)}{u}_{000}, \ldots, \overset{(\iota)}{u}_{00N_x}, \overset{(\iota)}{u}_{010}, \ldots, \overset{(\iota)}{u}_{01N_x}, \ldots, \overset{(\iota)}{u}_{0N_yN_x}, \overset{(\iota)}{u}_{100}, \ldots, \overset{(\iota)}{u}_{1,N_yN_x}, \ldots, \overset{(\iota)}{u}_{N_tN_yN_x}]^T \tag{25}$$

$$L(\hat{t}_l) \otimes L(\hat{y}_j) \otimes L(\hat{x}_i) = [L_0(\hat{t}_l) \otimes L_0(\hat{y}_j) \otimes L(\hat{x}_i), \ldots, L_0(\hat{t}_l) \otimes L_{N_y}(\hat{y}_j) \otimes L(\hat{x}_l),$$

$$L_1(\hat{t}_l) \otimes L_0(\hat{y}_j) \otimes L(\hat{x}_i), \ldots, L_1(\hat{t}_l) \otimes L_{N_y}(\hat{y}_j) \otimes L(\hat{x}_i), \ldots, L_{N_t}(\hat{t}_l) \otimes L_{N_y}(\hat{y}_j) \otimes L(\hat{x}_i)]. \tag{26}$$

The first and n-th derivative of the approximate solution in each sub-interval ι-th are approximated at the collocation points $(\hat{x}, \hat{y}, \hat{t})$ as follows:

$$\frac{\partial \overset{(\iota)}{u}}{\partial x}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \left(L(\hat{t}_l) \otimes L(\hat{y}_j) \otimes \mathbf{D}_x\right) \overset{(\iota)}{U}, \; \frac{\partial^n \overset{(\iota)}{u}}{\partial x^n}(\hat{x}_i, \hat{y}_j, \hat{z}_k) = \left(L(\hat{t}_l) \otimes L(\hat{y}_j) \otimes \mathbf{D}_x^{(n)}\right) \overset{(\iota)}{U},$$

$$\frac{\partial \overset{(\iota)}{u}}{\partial y}(\hat{x}_i, \hat{y}_j, \hat{z}_k) = \left(L(\hat{t}_l) \otimes \mathbf{D}_y \otimes L(\hat{x}_i)\right) \overset{(\iota)}{U}, \; \frac{\partial^n \overset{(\iota)}{u}}{\partial y^n}(\hat{x}_i, \hat{y}_j, \hat{z}_k) = \left(L(\hat{t}_l) \otimes \mathbf{D}_y^{(n)} \otimes L(\hat{x}_i)\right) \overset{(\iota)}{U},$$

$$\frac{\partial \overset{(\iota)}{u}}{\partial t}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \left(\mathbf{D}_t \otimes L(\hat{y}_j) \otimes L(\hat{x}_i)\right) \overset{(\iota)}{U}, \; \frac{\partial^n \overset{(\iota)}{u}}{\partial t^n}(\hat{x}_i, \hat{y}_j, \hat{t}_l) = \left(\mathbf{D}_t^{(n)} \otimes L(\hat{y}_j) \otimes L(\hat{x}_i)\right) \overset{(\iota)}{U},$$

where $\mathbf{D}_x$, $\mathbf{D}_y$ and $\mathbf{D}_t$ are the differentiation matrices with respect to x, y and t, respectively.

Spectral collocation

Replacing the continuous derivatives with discrete derivatives at the collocation points, the QLM scheme (5) becomes a linear system of algebraic equations of size $(N_x+1)(N_y+1)(N_t+1)$, and given by

$$\Big[\mathrm{diag}[\varpi_{4,k}]\left(\mathbf{I}_{N_t+1}\otimes\mathbf{I}_{N_y+1}\otimes\Omega_x^2\mathbf{D}_x^{(2)}\right)+\mathrm{diag}[\varpi_{3,k}]\left(\mathbf{I}_{N_t+1}\otimes\Omega_y^2\mathbf{D}_y^{(2)}\otimes\mathbf{I}_{N_x+1}\right)$$
$$+\mathrm{diag}[\varpi_{2,k}]\left(\mathbf{I}_{N_t+1}\otimes\mathbf{I}_{N_y+1}\otimes\Omega_x\mathbf{D}_x\right)+\mathrm{diag}[\varpi_{1,k}]\left(\mathbf{I}_{N_t+1}\otimes\Omega_y\mathbf{D}_y\otimes\mathbf{I}_{N_x+1}+\mathrm{diag}[\varpi_{0,k}]\right)$$
$$-\left(\Omega_t\mathbf{D}_t\otimes\mathbf{I}_{N_y+1}\otimes\mathbf{I}_{N_x+1}\right)\Big]\overset{(\iota)}{U}_{k+1}=\overset{(\iota)}{R}_k, \tag{27}$$

where I_{N_x+1}, I_{N_y+1} and I_{N_t+1} are identity matrices of sizes $(N_x+1)\times(N_x+1)$, $(N_y+1)\times(N_y+1)$ and $(N_t+1)\times(N_t+1)$, $\Omega_x=\frac{2}{b-a}$, $\Omega_y=\frac{2}{d-c}$, $\Omega_t=\frac{2}{t_\iota-t_{\iota-1}}$ are used for scaling the derivative matrices.
From the initial and boundary conditions, numerical values of the unknown U are known at the boundaries. Thus, we have to solve an $N_t(N_x-1)(N_y-1)$ equation in $N_t(N_x-1)(N_y-1)$ unknowns for the interior points as:

$$\Big[\mathrm{diag}[\hat{\varpi}_{4,k}]\left(\mathbf{I}_{N_t}\otimes\mathbf{I}_{N_y-1}\otimes\Omega_x^2\left[\mathbf{D}_x^{(2)}\right]_{2:N_x}\right)+\mathrm{diag}[\hat{\varpi}_{3,k}]\left(\mathbf{I}_{N_t}\otimes\Omega_y^2\left[\mathbf{D}_y^{(2)}\right]_{2:N_y}\otimes\mathbf{I}_{N_x-1}\right) \tag{28}$$
$$+\mathrm{diag}[\hat{\varpi}_{2,k}]\left(\mathbf{I}_{N_t}\otimes\mathbf{I}_{N_y-1}\otimes\Omega_x\left[\mathbf{D}_x\right]_{2:N_x}\right)+\mathrm{diag}[\hat{\varpi}_{1,k}]\left(\mathbf{I}_{N_t}\otimes\Omega_y\left[\mathbf{D}_y\right]_{2:N_{yy}}\otimes\mathbf{I}_{N_x-1}\right)+\mathrm{diag}[\hat{\varpi}_{0,k}]$$
$$-\left(\Omega_t\left[\mathbf{D}_t\right]_{1:N_t}\otimes\mathbf{I}_{N_y-1}\otimes\mathbf{I}_{N_x-1}\right)\Big]\overset{(\iota)}{\hat{U}}_{k+1}=\left[\overset{(\iota)}{R}_k\right]_{2:N_x,2:N_y,1:N_t}-\left(\phi_{ijl}+\hat{\phi}_{ijl}\right)-\left(\varphi_{ijl}+\hat{\varphi}_{ijl}\right)-\hat{\vartheta}_{ijl}, \tag{29}$$

Spectral collocation

In equation (29),

$\overset{(\iota)}{\hat{U}} = \overset{(\iota)}{U}_{2:N_x,2:N_y,1:N_t}$, $\hat{\varpi}_{r,k}(r = 1, 2, 3, 4) = \left[\varpi_{r,k}\right]_{2:N_x,2:N_y,1:N_t}$,

$\hat{\vartheta}_{ijl} = -\left[D_t\right]_{1:N_t,N_t+1} \otimes \overset{(\iota)}{U}_{2:N_x,2:N_y,N_t+1}$,

$\phi_{ijl} = \varpi_{4,k}\left[D_x^{(2)}\right]_{2:N_x,1} \otimes \overset{(\iota)}{U}_{1,2:N_y,1:N_t} + \varpi_{2,k}\left[D_x\right]_{2:N_x,1} \otimes \overset{(\iota)}{U}_{1,2:N_y,1:N_t}$,

$\hat{\phi}_{ijl} = \varpi_{4,k}\left[D_x^{(2)}\right]_{2:N_x,N_x+1} \otimes \overset{(\iota)}{U}_{N_x+1,2:N_y,1:N_t} + \varpi_{2,k}\left[D_x\right]_{2:N_x,N_x+1} \otimes \overset{(\iota)}{U}_{N_x+1,2:N_y,1:N_t}$,

$\varphi_{ijl} = \varpi_{3,k}\left[D_y^{(2)}\right]_{2:N_y,1} \otimes \overset{(\iota)}{U}_{2:N_x,1,1:N_t} + \varpi_{1,k}\left[D_y\right]_{2:N_y,1} \otimes \overset{(\iota)}{U}_{2:N_x,1,1:N_t}$,

$\hat{\varphi}_{ijl} = \varpi_{3,k}\left[D_y^{(2)}\right]_{2:N_y,N_y+1} \otimes \overset{(\iota)}{U}_{2:N_x,N_y+1,1:N_t} + \varpi_{1,k}\left[D_y\right]_{2:N_y,N_y+1} \otimes \overset{(\iota)}{U}_{2:N_x,N_y+1,1:N_t}$,

Equation (29) can be expressed compactly as a matrix system of size $N_t(N_x - 1)(N_y - 1) \times N_t(N_x - 1)(N_y - 1)$

$$\mathbf{A}\,\overset{(\iota)}{\hat{\mathbf{U}}} = \overset{(\iota)}{\hat{\mathbf{R}}}, \quad \iota = 1, 2, 3, \ldots, p. \tag{30}$$

Error analysis

Maximum error norm

The accuracy of the numerical scheme has been determined using the maximum error norm evaluated by

$$L_\infty = ||u^a - u^e||_\infty \simeq \max_{i,j} \left\{ \left| u^a(x_i, y_j, t) - u^e(x_i, y_j, t) \right| , \ : 0 \leq i \leq N_x, 0 \leq j \leq N_y \right\}, \quad (31)$$

and the L_2 error norm computed as

$$L_2 = ||u^a - u^e||_\infty \simeq \left[\sum_{i=0}^{N_x} \sum_{j=0}^{N_y} \left| u^a(x_i, y_j, t) - u^e(x_i, y_j, t) \right|^2 \right]^{1/2}, \quad (32)$$

where u^a and u^e are numerical and exact solutions, respectively.

Solution error

To determine the convergence of the iterative scheme, we monitor the infinity error norm at various iterations. The error at each iteration level is evaluated as follows:

$$||E_{k+1}||_\infty = \max_{i,j} \left\{ \left| u^a_{k+1}(x_i, y_j, t) - u^e(x_i, y_j, t) \right| , \ : 0 \leq i \leq N_x, 0 \leq j \leq N_y \right\}, \quad (33)$$

where E is called the solution error.

Stability analysis

Condition number

The numerical scheme is assessed for stability by evaluating the condition number of the coefficient matrix of the resultant linear system of equations. The definition of a condition number of a square matrix say A, which is non-singular, is given by

$$\kappa_p(A) = ||A||_p.||A^{-1}||_p, \ ||A||_2 = \sqrt{\rho\left(A^T A\right)}. \tag{34}$$

This value depends on the choice of a matrix norm and MATLAB utilizes the matrix norm of L_2. Thus, equation (34) becomes

$$\kappa(A) = ||A||_2.||A^{-1}||_2. \tag{35}$$

In MATLAB, the condition number of the matrix **A** is directly evaluated by using the built-in function cond(**A**).

Table: Error norms when Re is varied, $N_x = N_y = 15, N_t = 10$ in different intervals $[0, T]$

		MV-SQLM	(p=1)				MDMV-SQLM	(p=10)		
$[0, T] \backslash Re$	1	5	10	15	20	1	5	10	15	20
$[0, 10]$	7.958e-06	1.793e-02	8.253e-02	1.552e-01	8.165e-01	1.421e-14	4.319e-14	2.343e-14	5.995e-14	4.191e-12
$[0, 12]$	1.633e-05	1.710e-02	6.038e-02	9.628e-02	2.776e+01	1.632e-14	1.321e-14	2.043e-14	3.242e-14	6.894e-14
$[0, 14]$	4.423e-05	2.542e-02	7.835e-02	1.169e-01	2.405e+00	1.954e-14	1.543e-14	1.976e-14	1.776e-14	7.327e-15
$[0, 16]$	9.896e-05	2.139e-02	7.178e-02	1.127e-01	1.954e+00	2.276e-14	1.910e-14	2.032e-14	1.887e-14	7.327e-15
$[0, 18]$	1.142e-04	1.920e-02	5.357e-02	8.104e-02	2.077e-01	1.799e-14	1.754e-14	2.043e-14	2.243e-14	8.105e-15
$[0, 20]$	1.485e-04	2.555e-02	6.223e-02	8.269e-02	1.219e+01	1.832e-14	1.621e-14	2.243e-14	2.509e-14	7.327e-15
Cond (**A**)	1.1060e+03	9.2821e+02	8.0314e+02	7.3647e+02	1.2602e+06	1.0554e+03	7.0374e+02	5.3362e+02	4.5957e+02	4.1998e+02
CPU Time	18.971757	19.319832	30.694018	29.466441	34.784713	16.054362	16.025947	16.609866	16.105091	16.595988

Table: Comparison of error norms for various T when $Re = 1$

		Liu et al. [9]		Mohamed et al. [10]		MDMV-SQLM	$(Nt = 10, p = 10)$
T	$N_x = N_y$	L_2 error norms	L_∞ error norms	L_2 error norms	L_∞ error norms	L_2 error norms	L_∞ error norms
0.05	5	4.969e-08	4.651e-08	4.375e-07	5.855e-07	3.753e-08	4.503e-09
0.05	10	6.217e-09	5.907e-09	4.775e-09	4.492e-09	1.984e-14	2.831e-15
0.05	15	2.532e-09	2.188e-09	2.407e-10	1.887e-10	3.615e-14	3.497e-15
0.25	5	9.98993e-09	9.80769e-09	2.909e-07	4.057e-07	3.632e-08	4.599e-09
0.25	10	8.12715e-09	7.04501e-09	2.379e-10	2.160e-10	1.664e-14	1.665e-15
0.25	15	7.23104e-09	6.05481e-09	1.207e-11	8.888e-12	5.945e-14	6.661e-15

Table: Comparison of error norms for various T when $Re = 10$

	Mohamed et al. [10]			MDMV-SQLM ($N_t = 10, p = 15$)		
T	$Nx = N_y$	L_2 error norms	L_∞ error norms	$N_x = N_y$	L_2 error norms	L_∞ error norms
3	16	3.18e-09	3.52e-09	10	7.252e-09	1.817e-09
5	16	1.29e-13	1.31e-13	10	5.526e-13	9.137e-14
10	16	3.08e-13	3.86e-13	10	3.125e-14	3.997e-15

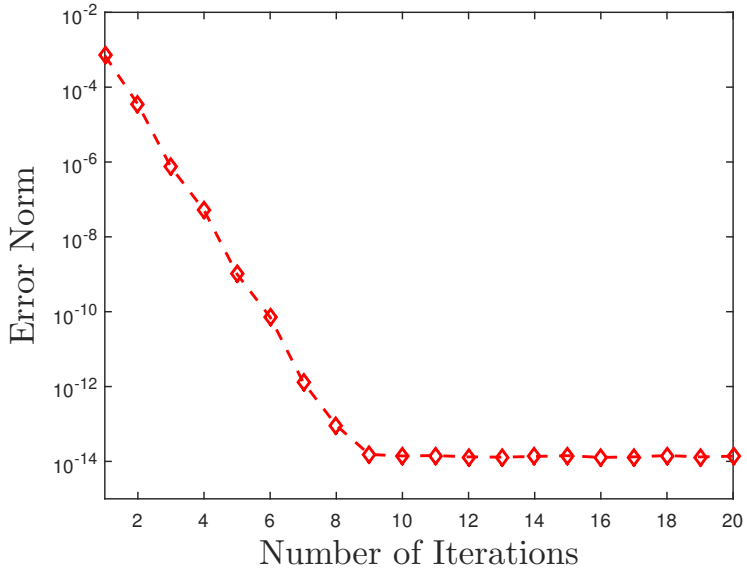
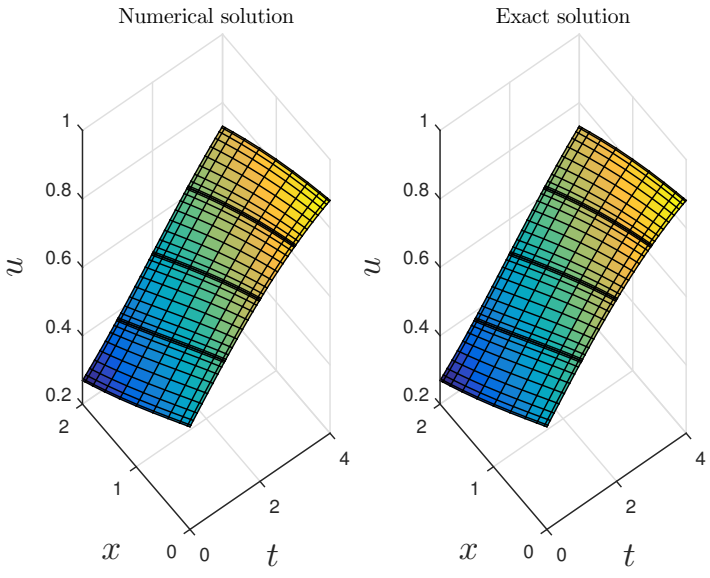
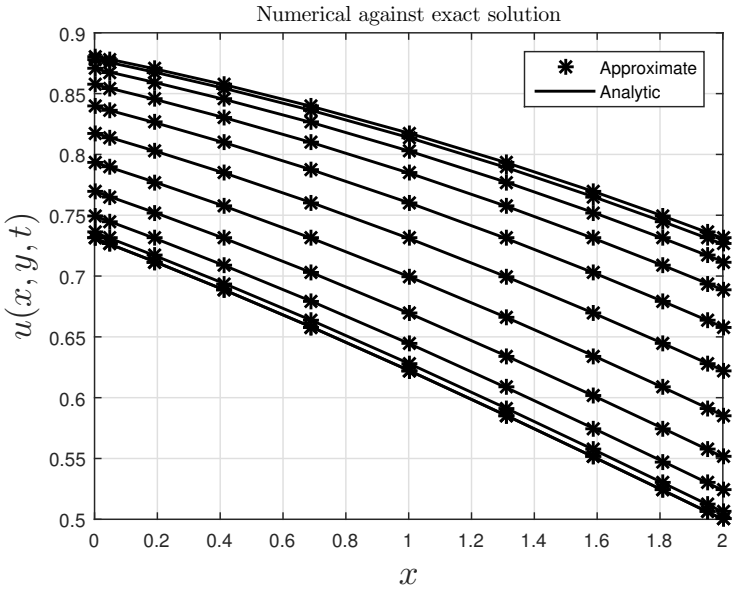


Figure: Convergence graph of when $Re = 1, N_x = N_y = N_t = 10$

Exact vs numerical solutions



(a) 3D plot when $Re = 1, y = 2$



(b) 2D plot when $Re = 1$, y is varied at $t = 4$

Figure: Numerical vs exact solutions when $N_x = N_y = N_t = 10$

Concluding remarks

- Reducing the size of time computational domain at each sub-interval assures mitigation of numerical challenges that are linked to large matrices and ill-conditioned nature of the coefficient matrix.
- Numerical scheme is computationally cheap, convergent and yield extremely accurate and stable results using minimal number of grid points.
- Proposed method can be efficient and accurate in obtaining better approximate solutions for nonlinear DEs since it can overcome the challenge of dealing with the multiplication of derivative matrices by very small parameters.

Future direction

- Modifying the method using overlapping grid idea in space to accommodate problems with large computational domains.
- Extending the method for adaptability in $(3 + 1)$ dimensional parabolic PDEs, multi-dimensional hyperbolic PDEs, time-fractional problems (e.g Burgers' PDEs) , PDEs with complex geometries and different type of boundary conditions.
- Various flow problems such as cavity flow problems can also be solved using the method.

END END —— THANK YOU !!!! —— END END

References


Burgers J M 1995 Mathematical Examples Illustrating Relations Occurring in the Theory of Turbulent Fluid Motion Selected Papers of J. M. Burgers (Berlin: Springer) 281?334


Greenshields B D. A study in highway capacity Highway Research Board Proc. 1935; 448?77


Gurbatov S and Saichev A. Probability distributions and spectra of potential hydrodynamic turbulence Radio?zika 1984; 27: 456?68


Hirsh R S. Higher order accurate difference solutions of ?uid mechanics problems by a compact differencing technique J. Comput. Phys. 1975; 19: 90-109


Kreiss G and Kreiss H-O. Convergence to steady state solutions of Burgers' equation. Appl. Numer. Math. 1986; 2: 161-179.


Arora G, Joshi V. A computational approach using modified trigonometric cubic B-spline for numerical solution of Burger's equation in one and two dimensions. Alex Eng J. 2018; 57: 1087–1098.


Samuel FM, Motsa SS. A highly accurate trivariate spectral collocation method of solution for two-dimensional nonlinear initial-boundary value. Appl Math Comput. 2019; 360: 221–235.

References


Dlamini PG, Magagula MV, A multivariate spectral quasi-linearisation method for the solution of the (2+1) dimensional Burger's equations. Int J Nonlinear Sci Numer Simul. 2020; 21: 683–691.


Liu X, Wang J, Zhou Y. A space-time fully decoupled wavelet Galerkin method for solving two-dimensional Burgers equations. Comput Math Appl. 2016; 72 : 2908–2919.


Mohamed NA, Rashed AS, Melaibari A, Sedighi HM, Eltaher MA. Effective numerical technique applied for Burgers' equation of (1 + 1)-, (2 + 1)-dimensional, and coupled forms. Math Methods Appl Sci. 2021; 44 : 10135–10153.


Mkhatshwa M.P., Khumalo M., Dlamini P.G. Multi-domain multivariate spectral collocation method for (2+1) dimensional nonlinear partial differential equations. Partial Differential Equations in Applied Mathematics. 2022; 6: 100440.


Bellman, R.E., Kalaba, R.E. *Quasilinearization and Nonlinear Boundary Value Problems*; RAND Corporation: Santa Monica, CA, USA, 1965.


Trefethen L. N., Spectral Methods in Matlab; SIAM: Philadelphia, PA, USA, 2000.


Canuto C., Hussaini M. Y., Quarteroni A., Zang T. A., Spectral Methods in Fluid Dynamics; Springer: Berlin, Germany, 1988.


Singh I.; Kumar, S. Wavelet methods for solving three-dimensional partial differential equations. *Math. Sci.* **2017,** *11*, 145–154.