Pure and Applied Mathematics Journal
2024; 3(2): 17-28
http://www.sciencepublishinggroup.com/j/pamj
doi: 10.11648/j.pamj.20241302.11
ISSN: 2326-9790 (Print); ISSN: 2326-9812 (Online)

An Integral-like Numerical Approach for Solving Burgers' Equation

Somrath Kanoksirirath

National Electronics and Computer Technology Center (NECTEC), National Science and Technology Development Agency (NSTDA), Pathum Thani, Thailand

Email address:
somrath.kan@ncr.nstda.or.th

To cite this article:
Somrath Kanoksirirath. (2024). An Integral-like Numerical Approach for Solving Burgers' Equation. *Pure and Applied Mathematics Journal, 3*(2), 17-28. https://doi.org/10.11648/j.pamj.20241302.11

Received: 20 April 2024; **Accepted:** 28 May 2024; **Published:** 12 June 2024

Abstract: The Burgers' equation, commonly appeared in the study of turbulence, fluid dynamics, shock waves, and continuum mechanics, is a crucial part of the dynamical core of any numerical weather model, influencing simulated meteorological phenomena. While past studies have suggested several robust numerical approaches for solving the equation, many are too complicated for practical adaptation and too computationally expensive for operational deployment. This paper introduces an unconventional approach based on spline polynomial interpolations and the Hopf-Cole transformation. Using Taylor expansion to approximate the exponential term in the Hopf-Cole transformation, the analytical solution of the simplified equation is discretized to form our proposed scheme. The scheme is explicit and adaptable for parallel computing, although certain types of boundary conditions need to be employed implicitly. Three distinct test cases were utilized to evaluate its accuracy, parallel scalability, and numerical stability. In the aspect of accuracy, the schemes employed cubic and quintic spline interpolation perform equally well, managing to reduce the ℓ_1, ℓ_2, and ℓ_∞ error norms down to the order of 10^{-4}. Parallel scalability observed in the weak-scaling experiment depends on time step size but is generally as good as any explicit scheme. The stability condition is $\nu\Delta t/\Delta x^2 > 0.02$, including the viscosity coefficient ν due to the Hopf-Cole transformation step. From the stability condition, the schemes can run at a large time step size Δt even when using a small grid spacing Δx, emphasizing its suitability for practical applications such as numerical weather prediction.

Keywords: Burgers' Equation, Hopf-Cole Transformation, Explicit Scheme, Parallel Scalability

1. Introduction

The partial differential equation (PDE) of the form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{1}$$

is called the one-dimensional Burgers' equation. It models physical transport phenomena in fluid flows, turbulence, traffic flows, and other areas [1]. It is a vital part of Navier-Stokes equations where u represents the velocity of the fluid and ν is a positive viscosity constant. The equation forms the core of computational fluid dynamics, weather models, ocean models, and hydrodynamic models.

The second term, uu_x, is non-linear, which hinders the development of simple, stable, and accurate numerical methods for solving various physics equations for practice uses. This study proposes an approach to address this non-linear term while keeping the scheme simple and explicit; since highly complex schemes often suffer from poor parallel scalability, limiting their practical use. Additionally, in contrast to most explicit schemes that have severe stability conditions resulting in higher computational costs, our scheme is stable even at large time step sizes.

Analytically, the solution of Burgers' equation can be obtained by converting it to a diffusion equation, $\phi_t = \nu\phi_{xx}$ using the Hopf-Cole transform, Eqs. (2) and (3). The diffusion equation is then solved using Fourier transform or other well-known approaches. This, however, is inconvenient and unsuitable for arbitrary initial conditions commonly encountered in practice. Therefore, several numerical schemes

have been developed.

$$\phi(x,t) = \exp\Big(\frac{-1}{2\nu}\int_{0'}^{x} u(\xi,t)\,\mathrm{d}\xi\Big) \tag{2}$$

$$u = -2\nu\frac{\phi_x}{\phi} \tag{3}$$

The standard numerical approach to tackle the non-linear term is to linearize it by assuming that u in uu_x is locally constant, as done in previous works such as [2–8]. However, this approach usually has limitations, particularly in terms of accuracy. Huang and Abduwali [6] successfully developed an explicit and unconditionally stable scheme using this assumption. Another approach is to rewrite the non-linear term as $(u^2)_x$ and solve it accordingly as in [9–14]. However, these methods often involve complicated, iterative, or implicit schemes that require solving a large set of linear equations, which can limit computational speed and parallel scalability of the programs.

On the other hand, some researchers, e.g., [15–24], have applied numerical Hopf-Cole transformation and solved the resulting diffusion equation instead. Advanced schemes have been employed to accurately diffuse exponential profiles generated by the Hopf-Cole transformation, while the transformation itself is generally approximated using a finite number of terms of its infinite series form or integrated numerically using Gaussian quadrature.

In the aspect of numerical procedures, the relatively expensive finite element method is widely employed [3, 14, 16, 24–28], as well as the Galerkin approach with cubic polynomials or other basis functions [2, 4, 5, 7, 10, 11, 17, 29–32]. Nevertheless, several studies have used the conventional finite difference method [6, 8, 12, 15, 18–20], but most of them are implicit schemes. Additionally, Gao et al. [33] applied a particle-based scheme to the Burgers' equation. To speed up implementation, Kumar et al. [34] developed a hybrid predictor-corrector scheme, while an artificial neural network was adopted to accelerate a prior Galerkin approach [35].

In this paper, we introduce an integral-like approach that mimics mathematical transformations and temporal integration to advance the numerical solution in time, while gridded data is represented as a continuous function by employing spline interpolation. The general idea of this approach is explained in Section 2. In Section 3, the time-stepping method used in solving the diffusion equation is described as an example and as an indispensable part of the scheme for solving Burgers' equation. In Section 4, the numerical Hopf-Cole transformation is explained and the entire scheme is composed. In Section 5, the results of several numerical experiments are shown and discussed, including an example of solving the Burgers'-Fisher equation. Conclusions are presented in Section 6.

2. Integral-like Approach

The proposed integral-like approach is different from conventional methods in both numerical differentiation in space and numerical forwarding scheme in time. To solve a PDE, our data grid contains not only the necessary field variables but also their derivatives, so that the variables can be represented as a continuous function using spline polynomial interpolation between adjacent grid points. To advance the variables in time, mathematical procedures along with additional approximations, based on the corresponding analytical solution, are emulated using the known continuous spline polynomial function. The derivatives also need to be updated in time. Initializing the derivatives using a finite difference method has been found to be sufficient. Since we can split terms in a PDE and solve more but simpler PDEs successively, as discussed in [36], the approach can be applied to any Burger-type equation. In Section 5, a Burgers'-Fisher equation, $u_t + uu_x = \nu u_{xx} - 3u(1-u)(1-2u)$, is split and solved as a demonstration.

In this paper, we investigate integral-like methods having variables represented by linear, cubic, and quintic spline interpolations. These methods will be referred to as linear grid scheme (LG), cubic grid scheme (CG), and quintic grid scheme (QG) respectively. For the cubic and quintic schemes, the second-order central finite difference is used to initialize u_x and u_{xx}, except at the two end points where the first-order forward and backward finite difference are employed instead.

To forward the numerical solution of the Burgers' equation for one time step, the Hopf-Cole transformation is applied numerically, the resulting diffusion equation is then solved in the Hopf-Cole space and converted back to the normal space. The application of the integral-like approach to the diffusion equation is discussed first to familiarize readers with the general idea of the method. It is worth noting that the integral-like approach is an explicit scheme that incorporates mathematical formulas and is equivalent to a Semi-Lagrangian method when applied to the linear advection equation.

3. Method for Linear Diffusion

The linear diffusion equation, $\phi_t = \nu\phi_{xx}$, where ν is a positive diffusion constant, can be solved analytically using the Fourier transform. This approach is discussed in textbooks such as [37] and [38]. The analytical solution is

$$\phi(x,t) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{4\pi\nu t}} \exp\Big(-\frac{(x-\xi)^2}{4\nu t}\Big)\,\phi(\xi,0)\,\mathrm{d}\xi \tag{4}$$

with analytical boundary conditions $\phi(\infty, t) = \phi(-\infty, t) = 0$.

The time-stepping scheme of the integral-like approach can be derived from Eq. (4) by substituting the local spline polynomial function $P_j(y,t)$ and changing the time interval. In the case of CG, $P_j(y,t) = a_j y^3 + b_j y^2 + c_j y + d_j$, with equally-spacing grid points, Eq. (4) becomes

$$\begin{aligned}\phi(x_i, t+\Delta t) &= \sum_{j=-\infty}^{\infty} \int_0^{\Delta x} \frac{1}{\sqrt{4\pi\nu\Delta t}} \exp\Big(-\frac{(x_i - (\xi_j + y))^2}{4\nu\Delta t}\Big) \quad P_j(y,t)\,\mathrm{d}(\xi_j + y) \\ &\approx \frac{1}{\sqrt{4\pi\nu\Delta t}} \sum_{|\ell_{i,j}|\le 5\sigma} \int_0^{\Delta x} \exp\Big(-\frac{(y+\ell_{i,j})^2}{4\nu\Delta t}\Big) \quad \Big(a_j y^3 + b_j y^2 + c_j y + d_j\Big)\,\mathrm{d}y \end{aligned} \tag{5}$$

Due to the Gaussian decay term, the summation range can be limited to j such that the relative distance $\ell_{i,j} \equiv \xi_j - x_i$ is within a margin of five standard deviations, i.e., $5\sigma = 5\sqrt{2\nu\Delta t}$, which is chosen for this paper.

It was found experimentally that all three methods are numerically stable as long as the marginal range of 5σ is larger than the grid spacing Δx. This is evidenced in Figure 7. Mathematically, the condition is equivalent to $d \ge 0.02$, where $d = \nu(\Delta t)/(\Delta x)^2$ is a non-dimensional diffusion number. Interestingly, the stability condition for these methods is reversed compared to the stability condition of explicit finite difference schemes. Since the 5σ length is an estimation, the value 0.02 is not exact. Notably, the stability condition is independent of u.

Due to the summation of j within the marginal range, the time complexity of the integral-like method is $\mathcal{O}((5\sigma/\Delta x)\, n_x n_t) \sim \mathcal{O}((\sqrt{\nu\Delta t}/\Delta x)\, n_x n_t) \sim \mathcal{O}(\sqrt{\nu}\, n_x^2 n_t^{1/2})$, where n_x is the number of grid points and n_t is the total number of time steps. Therefore, the complexity of the algorithm is not linear.

Another implication of the marginal range is that the boundary conditions may have to be specified by a small set of points, instead of a single point exactly at the boundary. For example, in the case of periodic boundary conditions, the required outside point $-j$ on the left of the considered domain corresponds to the inside point $n - j$. Similarly, the values at the outside point $n_x + j$ on the right are that of the j-th point. For Dirichlet and no-flux boundary conditions, reflected points or their mirror images are used in this study. Adding extra grid points to both ends can keep the implementation simple and is adopted here, as it also matches with the domain decomposition method for parallelization.

Next, to quantifies Eq. (5), the indefinite integral of the form, $\int y^m \exp(-(y+\ell)^2/\delta)\,\mathrm{d}y$, where m is a positive integer, is evaluated recursively using Eqs. (6) - (8), which is derived by applying integration by parts.

$$PG_m \equiv \int y^m \exp\Big(-\frac{(y+\ell)^2}{\delta}\Big)\mathrm{d}y = (m-1)\,\frac{\delta}{2}\,PG_{m-2} - \ell\,PG_{m-1} \quad - \frac{\delta}{2}\,y^{m-1}\exp\Big(-\frac{(y-\ell)^2}{\delta}\Big) \tag{6}$$

where

$$PG_0 = \frac{\sqrt{\pi\delta}}{2}\left[\mathrm{erf}\Big(\frac{y+\ell}{\sqrt{\delta}}\Big) - \mathrm{erf}\Big(\frac{\ell}{\sqrt{\delta}}\Big)\right] \tag{7}$$

$$PG_1 = -\frac{\ell\sqrt{\pi\delta}}{2}\left[\mathrm{erf}\Big(\frac{y+\ell}{\sqrt{\delta}}\Big) - \mathrm{erf}\Big(\frac{\ell}{\sqrt{\delta}}\Big)\right] \quad - \frac{\delta}{2}\left[\exp\Big(-\frac{(y+\ell)^2}{\delta}\Big) - \exp\Big(-\frac{\ell^2}{\delta}\Big)\right] \tag{8}$$

On the other hand, the coefficients of the cubic spline polynomial function $P_j(y)$, i.e., a_j, b_j, c_j and d_j, are found by solving Eqs. (9) - (12).

$$P_j(\Delta x, t) = \phi_{j+1} = a_j(\Delta x)^3 + b_j(\Delta x)^2 + c_j(\Delta x) + d_j \tag{9}$$

$$\partial_x P_j(\Delta x, t) = \partial_x \phi_{j+1} = 3a_j(\Delta x)^2 + 2b_j(\Delta x) + c_j \tag{10}$$

$$P_j(0,t) = \phi_j = d_j \tag{11}$$

$$\partial_x P_j(0,t) = \partial_x \phi_j = c_j \tag{12}$$

We denote $\partial_x \phi_j$ as the first derivative of ϕ at grid point j. These derivatives are stored in a data grid and are updated by using Eq. (13), which is derived by differentiating Eq. (5) with respect to x_i.

$$\begin{aligned}
\frac{\partial}{\partial x_i}\phi(x_i, t+\Delta t) &= \frac{1}{\sqrt{4\pi\nu\Delta t}} \sum_{|\ell_{i,j}|\le 5\sigma} \int_0^{\Delta x} \left[\frac{\partial(\xi_j - x_i)}{\partial x_i}\frac{\partial}{\partial \ell_{i,j}}\exp\Big(-\frac{(y+\ell_{i,j})^2}{4\nu\Delta t}\Big)\right] \Big(a_j y^3 + b_j y^2 + c_j y + d_j\Big)\,\mathrm{d}y \\
&= \frac{1}{\sqrt{4\pi\nu\Delta t}} \sum_{|\ell_{i,j}|\le 5\sigma} \int_0^{\Delta x} \left[(-1)\Big(\frac{-2(y+\ell_{i,j})}{4\nu\Delta t}\Big)\exp\Big(-\frac{(y+\ell_{i,j})^2}{4\nu\Delta t}\Big)\right] \Big(a_j y^3 + b_j y^2 + c_j y + d_j\Big)\,\mathrm{d}y \\
&= \frac{1}{(2\nu\Delta t)\sqrt{4\pi\nu\Delta t}} \sum_{|\ell_{i,j}|\le 5\sigma} \int_0^{\Delta x} \exp\Big(-\frac{(y+\ell_{i,j})^2}{4\nu\Delta t}\Big) \Big(a_j y^4 + (b_j + \ell_{i,j}a_j)y^3 + (c_j + \ell_{i,j}b_j)y^2 \\
&\quad + (d_j + \ell_{i,j}c_j)y + \ell_{i,j}d_j\Big)\,\mathrm{d}y
\end{aligned} \tag{13}$$

Hence, Eqs. (5) and (13) together form the complete integral-like scheme of CG for solving Burgers' equation. Their computation is facilitated by Eqs. (6) - (12). The derivation of LG and QG follows the same procedures as CG shown above but with a different order of spline polynomial $P_j(y)$ substituted.

4. Method for Hopf-Cole Transformation

The first step in performing the forward Hopf-Cole transformation, changing u to ϕ (Eq. 2), is to compute the integral of u in Eq. (14). For CG, it is

$$\begin{aligned}
\mathrm{Int}(u) &\equiv \int_{0'}^{x_i} u(\xi,t)\,\mathrm{d}\xi = \sum_{j=0}^{i-1}\int_0^{\Delta x} P_j(\xi,t)\,\mathrm{d}\xi \\
&= \sum_{j=0}^{i-1}\int_0^{\Delta x}\Big(a_j\xi^3 + b_j\xi^2 + c_j\xi + d_j\Big)\,\mathrm{d}\xi \\
&= \sum_{j=0}^{i-1} a_j\frac{(\Delta x)^4}{4} + b_j\frac{(\Delta x)^3}{3} \\
&\quad + c_j\frac{(\Delta x)^2}{2} + d_j(\Delta x)
\end{aligned} \tag{14}$$

Then, the exponent $-(1/2\nu)\,\mathrm{Int}(u)$ in Eq. (2) is represented as a quintic spline polynomial function Q_i using $\mathrm{Int}(u)$, u, and u_x at i and $i+1$. The quintic coefficients are found by standard approach, analogous to Eqs. (9) - (12).

Then, for a quintic polynomial Q_i of the i-th segment, the value of ϕ_i is approximated by the Taylor series of $\exp(Q_i)$. In our implementation, the number of terms in the Taylor series is varied adaptively, up to 16 terms, to ensure that the deviation is less than 0.01%. The resulting ϕ_i, which satisfies the diffusion equation, is then processed further using the method discussed in Section 3.

On the other hand, the backward Hopf-Cole transformation from ϕ to u in Eq. (3) is done by direct substitution. For example, u and u_x of CG are deduced from ϕ, ϕ_x, and ϕ_{xx} by using Eqs. (15) and (16).

$$u_i = -2\nu\left(\frac{\partial_x\phi_i}{\phi_i}\right) \tag{15}$$

$$\partial_x u_i = -2\nu\left(\frac{\phi_i(\partial_x^2\phi_i) - (\partial_x\phi_i)^2}{\phi_i^2}\right) \tag{16}$$

To sum up, an integral-like scheme for solving Burgers' equation consists of the following components: (1) Numerical Hopf-Cole integration scheme for transforming from u space to ϕ space, (2) Numerical diffusion scheme for advancing $\phi(x,t)$ and its required derivatives to $\phi(x, t+\Delta t)$, (3) Numerical Hopf-Cole differentiation scheme for transforming $\phi(x, t+\Delta t)$ and its derivatives back to u space, and (4) Spline interpolation method for representing discrete data points as a continuous function for the computation in (1)-(3). One complete time iteration step Δt for solving Burgers' equation is composed of (1), (2) and (3). Finite difference schemes are employed for computing the initial derivatives such as $u_x(x, t_0)$ from $u(x, t_0)$. Our source code is publicly available on GitHub (https://github.com/SKanoksi).

An implementation of an adaptive grid was explored, where grid points are redistributed based on the estimated path length of u, using Gaussian quadrature. However, the numerical solution was found to be less accurate due to numerical errors introduced when repeatedly rearranging the grids. Therefore, our unsuccessful implementation of the adaptive grid is briefly noted in this paper to inform the possibility that the adaptive grid may not improve integral-like schemes in general.

5. Numerical Experiment

In this section, four example cases are used to evaluate the integral-like methods for one-dimensional Burgers' equation, i.e., LG, CG, and QG. The accuracy of the methods is tested using Example 1 and 2, while numerical stability and parallel scalability are evaluated in Example 3. Example 4 investigates the viability of the split approach.

To quantify the accuracy of the schemes when comparing with exact/analytical solutions $f(x)$, the ℓ_1, ℓ_2, and ℓ_∞ error norms are adopted. These error norms are calculated as shown

below, for an equally-spacing grid.

$$\ell_1(u) = \frac{\sum_{j=0}^{n} |u_j - f(x_j)|}{\sum_{j=0}^{n} |f(x_j)|}, \tag{17}$$

$$\ell_2(u) = \frac{\sqrt{\sum_{j=0}^{n} (u_j - f(x_j))^2}}{\sqrt{\sum_{j=0}^{n} f(x_j)^2}}, \tag{18}$$

$$\ell_\infty(u) = \frac{\max|u_j - f(x_j)|}{\max|f(x_j)|} \tag{19}$$

Example 1. Burgers' equation (Eq. 1) with the initial condition

$$u(x, 0) = \sin(\pi x) \tag{20}$$

and the exact solution

$$u(x,t) = \frac{4\pi\nu \sum_{k=1}^{\infty} k\, A_k \sin(k\pi x) \exp(-k^2\pi^2\nu t)}{A_0 + 2\sum_{k=1}^{\infty} A_k \cos(k\pi x) \exp(-k^2\pi^2\nu t)} \tag{21}$$

where

$$A_k = \int_0^1 \cos(k\pi x) \exp\Big(\frac{\cos(\pi x) - 1}{2\pi\nu}\Big)\, \mathrm{d}x \tag{22}$$

are considered. Unlike in [33] and [39], periodic boundary conditions are employed.

The exact solution and numerical results are displayed in Figure 1 for $\nu = 0.01$, $\Delta x = 0.02$, and $\Delta t = 0.005$, while the numerical values at some grid points are given in Table 1. The error norms when using four different pairs of Δx and Δt are shown in Table 2. From the results, LG always performs the worst, while CG and QG are relatively comparable. For CG and QG, as the grid spacing and time step size are reduced, the error norms become saturated at around $\ell_2 = 10^{-3}$ as seen in Figure 4. This behavior may be caused by truncation errors introduced when approximating $\exp$ and erf functions, which are frequently used in the schemes. From Figure 4, although LG is approximately second-order accurate, the ℓ_2 error norm probably levels off at around 10^{-3} as well.

Table 1. *Numerical results and exact solution of Example 1 at $t = 1.0$ using $\nu = 0.1$ and $\Delta t = 0.01$.*

Position	Δx	LG	CG	QG	Exact
0.1	0.02	0.065558	0.066306	0.066305	0.066316
	0.01	0.066101	0.066284	0.066283	
0.2	0.02	0.129578	0.131189	0.131187	0.131209
	0.01	0.130750	0.131145	0.131144	
0.3	0.02	0.190039	0.192755	0.192752	0.192786
	0.01	0.192020	0.192689	0.192687	
0.4	0.02	0.243809	0.247999	0.247995	0.248041
	0.01	0.246874	0.247910	0.247908	
0.5	0.02	0.285749	0.291864	0.291859	0.291916
	0.01	0.290232	0.291752	0.291749	
0.6	0.02	0.307656	0.316005	0.316000	0.316068
	0.01	0.313784	0.315876	0.315872	
0.7	0.02	0.297804	0.308022	0.308017	0.308089
	0.01	0.305304	0.307884	0.307880	
0.8	0.02	0.243417	0.253657	0.253653	0.253718
	0.01	0.250927	0.253533	0.253529	
0.9	0.02	0.139279	0.146027	0.146025	0.146065
	0.01	0.144223	0.145951	0.145949	

Table 2. *Error norms of Example 1 at $t = 1.0$ using $\nu = 0.1$.*

Scheme	Δx	Δt	d	ℓ_1	ℓ_2	ℓ_∞
LG	0.02	0.0200	5.0	1.35e-02	1.46e-02	1.72e-02
	0.02	0.0100	2.5	2.64e-02	2.85e-02	3.35e-02
	0.01	0.0100	10.0	7.19e-03	7.75e-03	9.07e-03
	0.01	0.0025	2.5	2.70e-02	2.92e-02	3.42e-02
CG	0.02	0.0200	5.0	4.25e-04	4.33e-04	4.56e-04
	0.02	0.0100	2.5	1.97e-04	2.00e-04	2.11e-04
	0.01	0.0100	10.0	6.07e-04	6.16e-04	6.46e-04
	0.01	0.0025	2.5	8.27e-04	8.39e-04	8.76e-04
QG	0.02	0.0200	5.0	4.40e-04	4.48e-04	4.72e-04
	0.02	0.0100	2.5	2.12e-04	2.15e-04	2.26e-04
	0.01	0.0100	10.0	6.18e-04	6.28e-04	6.58e-04
	0.01	0.0025	2.5	8.97e-04	9.10e-04	9.50e-04

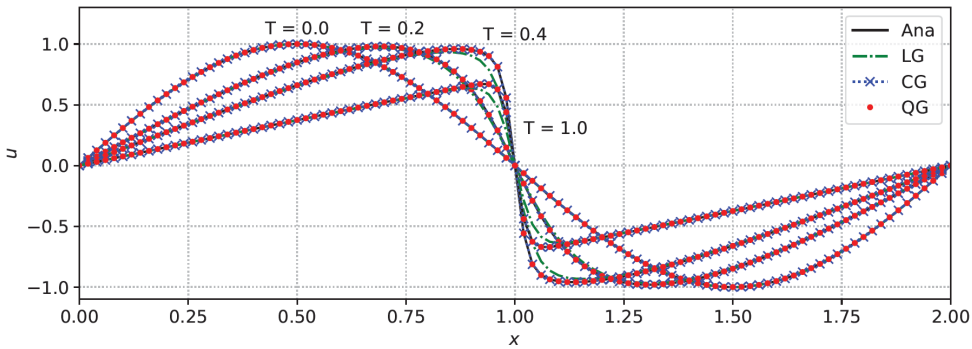

Figure 1. *Numerical solutions and exact solution of Example 1:* $\nu = 0.01$, $\Delta x = 0.02$, *and* $\Delta t = 0.005$.

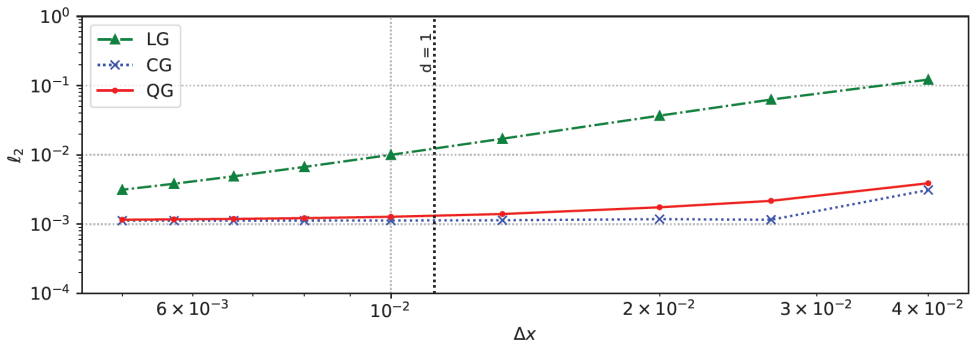

Figure 2. ℓ_2 *error norm of Example 1 as a function of* Δx *using* $\nu = 0.1$, $\Delta t = 0.0125$ *and* $n_t = 100$.

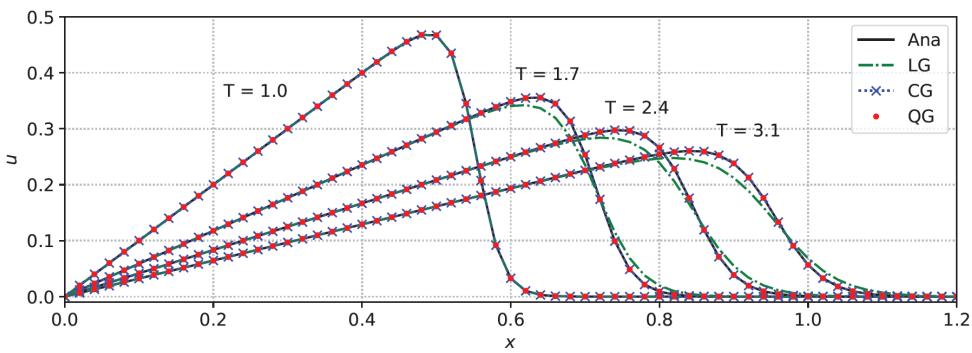

Figure 3. *Numerical solutions and exact solution of Example 2:* $\nu = 0.005$, $\Delta x = 0.02$ *and* $\Delta t = 0.02$.

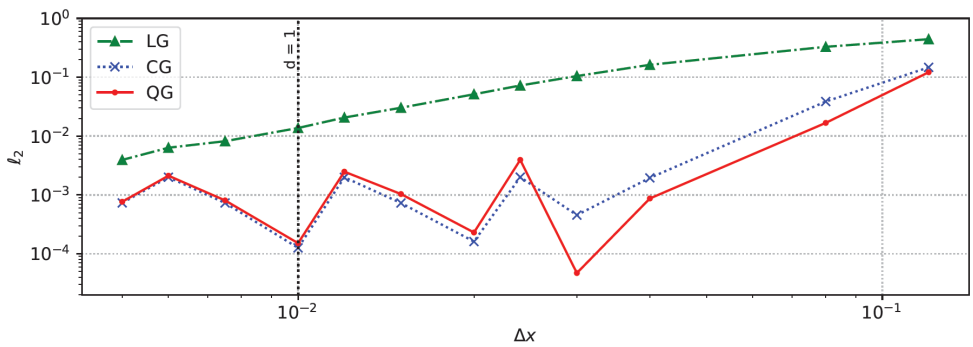

Figure 4. ℓ_2 *error norm of Example 2 as a function of* Δx *using* $\nu = 0.005$, $\Delta x = 0.02$, $\Delta t = 0.02$ *and* $n_t = 100$.

Table 3. *Numerical results and exact solution of Example 2 at $t = 2.4$ using $\nu = 0.005$ and $\Delta t = 0.02$.*

Position	Δx	LG	CG	QG	Exact
0.1	0.02	0.041664	0.041663	0.041662	0.041667
	0.01	0.041665	0.041665	0.041665	
0.2	0.02	0.083328	0.083328	0.083326	0.083333
	0.01	0.083331	0.083331	0.083330	
0.3	0.02	0.124995	0.124997	0.124994	0.125000
	0.01	0.124996	0.124996	0.124996	
0.4	0.02	0.166648	0.166661	0.166656	0.166665
	0.01	0.166659	0.166660	0.166659	
0.5	0.02	0.208207	0.208312	0.208305	0.208318
	0.01	0.208299	0.208311	0.208309	
0.6	0.02	0.248953	0.249808	0.249797	0.249816
	0.01	0.249681	0.249806	0.249804	
0.7	0.02	0.281531	0.288475	0.288445	0.288472
	0.01	0.287090	0.288458	0.288454	
0.8	0.02	0.239172	0.266132	0.266155	0.266228
	0.01	0.258598	0.266184	0.266179	
0.9	0.02	0.055092	0.038663	0.038622	0.038651
	0.01	0.043197	0.038638	0.038633	
1.0	0.02	0.003517	0.000902	0.000911	0.000912
	0.01	0.001385	0.000911	0.000912	
1.1	0.02	0.000137	0.000012	0.000013	0.000013
	0.01	0.000026	0.000013	0.000013	

Table 4. *Error norms of Example 2 at $t = 2.4$, using $\nu = 0.005$.*

Scheme	Δx	Δt	d	ℓ_1	ℓ_2	ℓ_∞
LG	0.02	0.035	0.4375	1.77e − 02	2.86e − 02	5.55e − 02
	0.02	0.020	0.2500	3.09e − 02	4.85e − 02	9.10e − 02
	0.01	0.020	1.0000	7.82e − 03	1.30e − 02	2.56e − 02
	0.01	0.005	0.2500	3.12e − 02	4.91e − 02	9.18e − 02
CG	0.02	0.035	0.4375	2.78e − 04	3.95e − 04	8.24e − 04
	0.02	0.020	0.2500	8.78e − 05	1.48e − 04	3.98e − 04
	0.01	0.020	1.0000	6.89e − 05	9.10e − 05	1.77e − 04
	0.01	0.005	0.2500	1.71e − 04	2.01e − 04	3.86e − 04
QG	0.02	0.035	0.4375	3.91e − 04	5.41e − 04	1.01e − 03
	0.02	0.020	0.2500	1.31e − 04	1.67e − 04	3.04e − 04
	0.01	0.020	1.0000	8.39e − 05	1.09e − 04	2.01e − 04
	0.01	0.005	0.2500	3.31e − 04	3.81e − 04	6.41e − 04

Example 2. Burgers' equation (Eq. 1) with the Dirichlet boundary condition

$$u(0,t) = u(1.2,t) = 0 \tag{23}$$

and the exact solution

$$u(x,t) = \frac{x/t}{1 + \sqrt{t/t_0}\exp(x^2/4\nu t)} \tag{24}$$

where $t \geq 1$ and $t_0 = \exp(0.125/\nu)$ are considered as in [40], [33], and [39]. In our case, the Dirichlet boundary conditions are modeled using the inverted reflection of the solution.

The results of this example, using $\nu = 0.005$, $\Delta x = 0.02$, and $\Delta t = 0.02$, are shown in Figure 3 and Table 3, while Table 4 and Figure 4 provide more insight into their order of accuracy. The results in Figure 4 exhibit similar features as in Figure 2 of the previous example, except for the oscillation of CG and QG at small Δx. The swing may be due to

the accumulation of errors, similar to Runge's and/or Gibbs' phenomenon near the steep slope. This behavior does not occur in Example 1, probably because the abrupt change is stationary at the grid point $x = 0$, while, in this example, the change is not always on a grid point as Δx is varied.

Example 3. Burgers' equation (Eq. 1) with a step initial condition at $x = 0$ is used to study both the stability and parallel scalability of the integral-like method. The exact solution for infinite boundary conditions $u(-\infty, t) = 1$ and $u(\infty, t) = 0$ is

$$u(x,t) = \frac{\operatorname{erfc}\left(\frac{x-t}{2\sqrt{\nu t}}\right)}{\operatorname{erfc}\left(-\frac{x}{2\sqrt{\nu t}}\right)\exp\left(\frac{x-t/2}{2\nu}\right) + \operatorname{erfc}\left(\frac{x-t}{2\sqrt{\nu t}}\right)} \tag{25}$$

which becomes a traveling-wave solution thereafter about $t = 10$. Using $\nu = 1$ and $\Delta t = 0.04$, the results are shown in Figures 5 and 6.

In Figure 7, the ℓ_2 error norm is plotted against the non-dimensional diffusion number $d = \nu\Delta t/\Delta x^2$. It can be seen that the stability condition of the integral-like method is not $d < 1$, as also indicated by Figures 2 and 4, but rather $d > 0.02$. This finding demonstrates that, unlike most simple explicit methods, the integral-like method is stable at large Δt. The condition $d > 0.02$ implies that the marginal range 5σ has to be larger than the grid spacing, as discussed in Section 3. The remaining stability issue encountered when ν is very small can be resolved by simply increasing Δt.

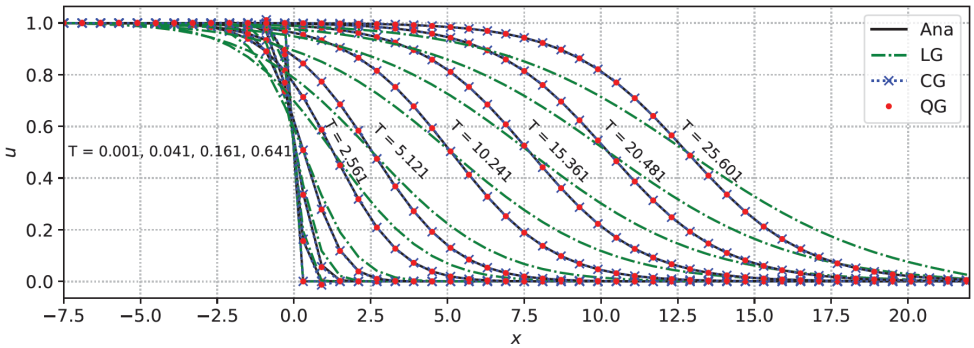

Figure 5. *Numerical solutions and exact solution of Example 3: $\nu = 1$, $\Delta x = 0.6$ and $\Delta t = 0.04$.*

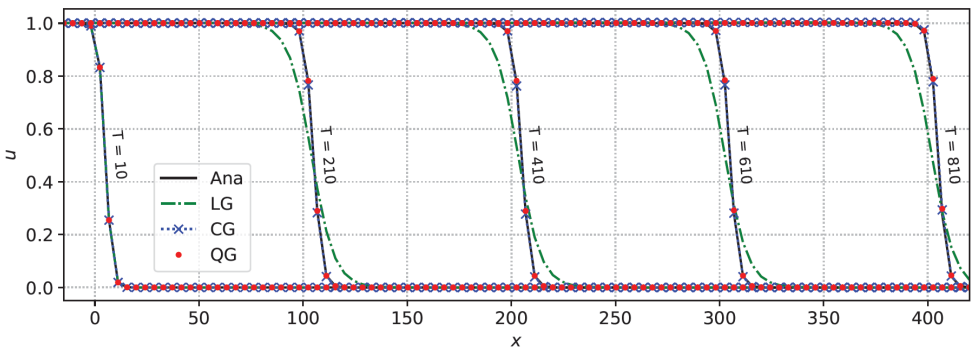

Figure 6. *Numerical solutions and exact solution of Example 3: $\nu = 1$, $\Delta x = 4.35$ and $\Delta t = 0.8$.*

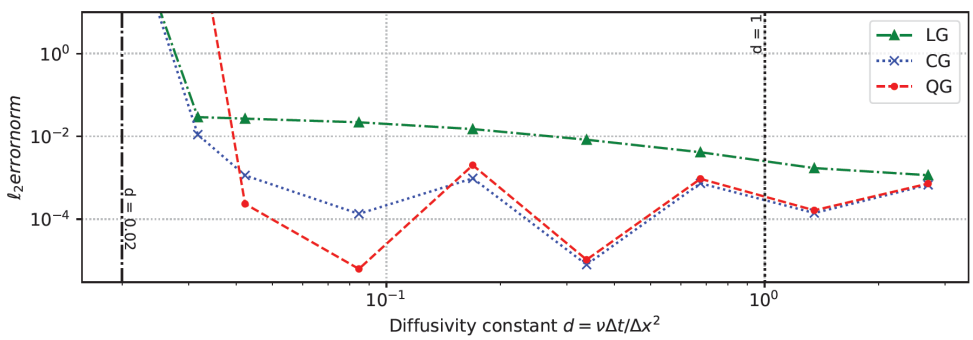

Figure 7. *ℓ_2 error norm of Example 3 as a function of ν using $\Delta x = 4.35$, $n_x = 401$, $\Delta t = 0.8$ and $n_t = 100$.*

A weak-scaling experiment was performed on the TARA cluster of the NSTDA supercomputer center (ThaiSC) to test the parallel efficiency of the integral-like method. Both the number of grid points and the number of time steps are adjusted to scale the computation workload while having the grid spacing Δx and the marginal range $5\sigma = 5\sqrt{2\nu\Delta t}$ approximately unaltered. In other words, roughly the same number of $u(x_j, t)$ is used in updating $u(x_i, t + \Delta t)$.

With N_s representing a running variable, the number of employed CPU cores is $N_\mathrm{core} = N_\mathrm{s}^2$ operating on the total number of grid points $n_x = 500\,N_\mathrm{s} + 1$ for domain $[-15, 400\,N_\mathrm{s} + 20]$ to simulate from $t_0 = 10$ to $t = 800\,N_\mathrm{s} + 10$. The time step size Δt is varied to explore the influence of the marginal range on parallel scalability. This is because the bigger the marginal range is, the larger the overlapped areas of the domain decomposition algorithm are. The outputs are shown in Figure 6.

TARA compute nodes, equipped with two Intel Xeon Gold 6148 CPU (2.40 GHz) and 192 GB of RAM, are employed. Our program was coded in Python and parallelized using mpi4py library, before being ported to C language by using Cython and compiled on TARA using foss-2021b toolchain. The weak-scaling parallel efficiency $R(1)/R(N_\mathrm{s})$, where $R(N_\mathrm{s})$ is the wall-clock runtime of the N_s scaled case, is plotted in Figure 8 from $N_\mathrm{s} = 1$ to $N_\mathrm{s} = 10$. The results of the serial runtime $R(1)$ are provided in Table 5.

From Figure 8, the parallel efficiency of the integral-like schemes decreases as a larger time step size Δt is chosen. However, from Table 5, the wall-clock runtime for a larger time step size is evidently lower. The parallel efficiency also declines as the problem size becomes larger and more CPU cores are used but remains roughly unchanged at moderate-to-large scale cases. Therefore, the time step size of operational, final applications may need to be experimentally found to match available computational resources.

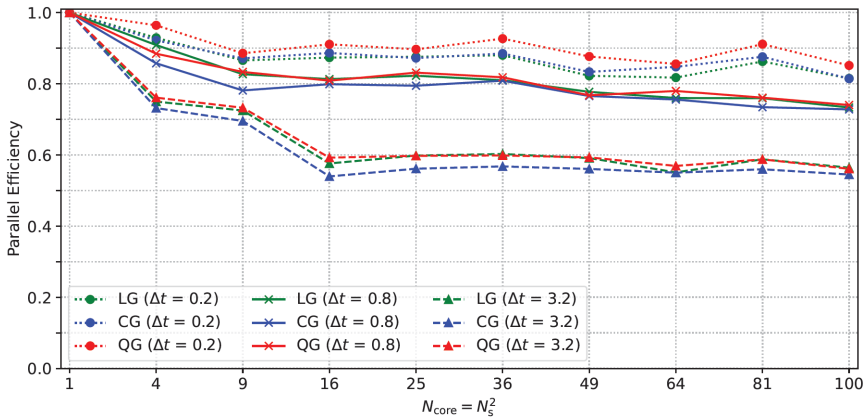

Figure 8. *Weak-scaling parallel efficiency of the integral-like schemes for Burgers' equation using time step size $\Delta t = 0.2$, 0.8 and 3.2.*

Table 5. *The wall-clock serial runtime $R(1)$ of each integral-like scheme for different time step size Δt, but ran for the same simulation time, i.e., from $t_0 = 10$ to $t = 810$.*

Δt	5σ	d	Runtime (sec)		
			LG	CG	QG
0.2	3.16	0.26-0.31	1,945	3,171	4,714
0.8	6.32	1.05-1.23	890	1,420	2,069
3.2	12.65	4.22-4.92	327	508	757

Example 4. Burgers'-Fisher equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} - 3u(1-u)(1-2u) \tag{26}$$

is considered to show a possible extension of the integral-like method. Applying the split approach, which separates terms in the equation into stages and successively solves them, to Eq. (26), the stage equations are

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{27}$$

$$\frac{\partial u}{\partial t} = -3u(1-u)(1-2u) \tag{28}$$

The first stage, Eq. (27), is to solve Burgers' equation; therefore, the procedures discussed in previous sections are directly employed. The second stage, Eq. (28), is a growth/decay equation. Its analytical solution is found by solving

$$-3\,\mathrm{d}t = \left(\frac{1}{u} - \frac{1}{(1-u)} + \frac{4}{(1-2u)}\right)\mathrm{d}u$$

$$\therefore \quad A\,\mathrm{e}^{-3t} = \frac{1}{4} - \frac{1}{4(2u-1)^2}$$

Therefore, the integral-like scheme for this second stage is

$$u(x_i, t+\Delta t) = \frac{1}{2}\left(1 \pm \frac{1}{\sqrt{1 - 4A\exp(-3\Delta t)}}\right)$$

where

$$A = \frac{1}{4}\left(1 - \frac{1}{(2u(x_i,t)-1)^2}\right)$$

From [41], an exact solution of Eq. (26) when $\nu = 1$ is

$$u(x,t) = \frac{1}{2}\left(1 - \tanh\left(x - \frac{t}{2}\right)\right) \tag{29}$$

Using the same setup as in Figure 5 of Example 3, the numerical results of CG and QG agree well with the exact solution as seen in Figure 9. The integral-like approach, therefore, has the potential for solving other Burger-type equations as well.

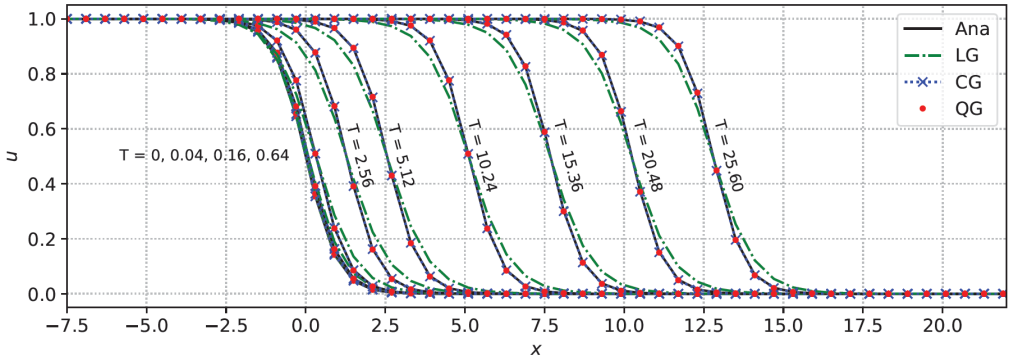

Figure 9. *Numerical solutions and exact solution of Example 4:* $\nu = 1$, $\Delta x = 0.6$ *and* $\Delta t = 0.04$.

6. Conclusions

In this study, a numerical approach based on the continuous representation of variables using local polynomial interpolation is explored. When applied to solve the one-dimensional Burgers' equation, the schemes derived using linear, cubic, and quintic interpolations are found to be numerically stable at large time step size Δt, with a stability condition of $\nu\Delta t > 0.02\Delta x^2$. From numerical experiments, the error norms decrease with smaller grid spacing and become constant around the order of $10^{-3} - 10^{-4}$. The weak-scaling parallel efficiency of the schemes is satisfactory among explicit scheme. This approach shows promise for operational applications that favor reliability, fast computation and good parallel scalability, such as numerical weather prediction.

Abbreviations

PDE	Partial Differential Equation
LG	Linear Grid scheme
CG	Cubic Grid scheme
QG	Quintic Grid scheme

Acknowledgments

I would like to thank NSTDA supercomputer center (ThaiSC) for providing computational resources for this work.

ORCID

0000-0002-1595-120X (Somrath Kanoksirirath)

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] M. P. Bonkile, A. Awasthi, C. Lakshmi, V. Mukundan, and V. S. Aswin, "A systematic literature review of burgers equation with recent advances, *Pramana*, vol. 90, pp. 1-21, 2018. https://doi.org/10.1007/s12043-018-1559-4

[2] M. Abdullah, M. Yaseen, and M. De la Sen, "An efficient collocation method based on Hermite formula and cubic B-splines for numerical solution of the Burgers equation, *Math. Comput. Simulation*, vol. 197, no. C, pp. 166-184, 2022. https://doi.org/10.1016/j.matcom.2022.02

[3] A. Dogan, "A galerkin finite element approach to burgers equation, *Appl. Math. Comput.*, vol. 157, no. 2, pp. 331-346, 2004. https://doi.org/10.1016/j.amc.2003.08.037

[4] I. Ganaie and V. Kukreja, "Numerical solution of burgers equation by cubic hermite collocation method, *Appl. Math. Comput.*, vol. 237, pp. 571-581, 2014. https://doi.org/10.1016/j.amc.2014.03.102

[5] Y. Hon and X. Mao, "An efficient numerical scheme for burgers equation, *Appl. Math. Comput.*, vol. 95, no. 1, pp. 37-50, 1998. https://doi.org/10.1016/S0096-3003(97)10060-1

[6] P. Huang and A. Abduwali, "The modified local cranknicolson method for one- and two- dimensional burgers equations, *Comput. Math. Appl.*, vol. 59, no. 8, pp. 2452-2463, 2010. https://doi.org/10.1016/j.camwa.2009.08.069

[7] A. Vs and A. Awasthi, "A differential quadrature based numerical method for highly accurate solutions of burgers equation: Dqm based numerical method for burgers equation, *Numer. Meth. Part. D. E.*, vol. 33, 07 2017. https://doi.org/10.1002/num.22178

[8] X. Yang, Y. Ge, and B. Lan, "A class of compact finite difference schemes for solving the 2d and 3d burgers equations, *Math. Comput. Simulation*, vol. 185, pp. 510-534, 2021. https://doi.org/10.1016/j.matcom.2021.01.009

[9] Y. Guo, Y. feng Shi, and Y. min Li, "A fifth-order finite volume weighted compact scheme for solving one-dimensional burgers equation, *Appl. Math. Comput.*, vol. 281, pp. 172-185, 2016. https://doi.org/10.1016/j.amc.2016.01.061

[10] S. Gupta and V. K. Kukreja, "An improvised collocation algorithm with specific end conditions for solving modified burgers equation, *Numer. Meth. Part. D. E.*, vol. 37, no. 1, pp. 874-896, 2021. https://doi.org/10.1002/num.22557

[11] S. R. Jena and G. S. Gebremedhin, "Decatic b-spline collocation scheme for approximate solution of burgers equation, *Numer. Meth. Part. D. E.*, vol. 39, no. 3, pp. 1851-1869, 2023. https://doi.org/10.1002/num.22747

[12] Y. Jiang, X. Chen, R. Fan, and X. Zhang, "High order semi-implicit weighted compact nonlinear scheme for viscous burgers equations, *Math. Comput. Simulation*, vol. 190, pp. 607-621, 2021. https://doi.org/10.1016/j.matcom.2021.06.006

[13] R. K. Mohanty and J. Talwar, "Anewcompactalternating group explicit iteration method for the solution of nonlinear time-dependent viscous burgers equation, *Numer. Anal. Appl.*, vol. 8, pp. 314-328, 2015. https://doi.org/10.1134/S1995423915040059

[14] R. Zhang, Y. Xi-Jun, and Z. Guo-Zhong, "Local discontinuous galerkin method for solving burgers and coupled burgers equations, *Chin. Phys. B*, vol. 20, no. 11, p. 110205, 11 2011. https://doi.org/10.1088/1674-1056/20/11/110205

[15] M. K. Kadalbajoo and A. Awasthi, "A numerical method based on crank-nicolson scheme for burgers equation, *Appl. Math. Comput.*, vol. 182, no. 2, pp. 1430-1442, 2006. https://doi.org/10.1016/j.amc.2006.05.030

[16] R. Kannan and Z. Wang, "A high order spectral volume solution to the burgers equation using the hopfcole transformation, *Internat. J. Numer. Methods Fluids*, vol. 69, no. 4, pp. 781-801, 2012. https://doi.org/10.1002/fld.2612

[17] S. S. Kumbhar and S. Thakar, "Galerkin finite element method for forced burgers equation, *J. Math. Model.*, vol. 7, no. 4, pp. 445-467, 2019. https://doi.org/10.22124/jmm.2019.13259.1265

[18] S. Kutluay, A. Bahadir, and A. zde, "Numerical solution of one-dimensional burgers equation: explicit and exact-explicit finite difference methods, *J. Comput. Appl. Math.*, vol. 103, no. 2, pp. 251-261, 1999. https://doi.org/10.1016/S0377-0427(98)00261-1

[19] W. Liao, "An implicit fourth-order compact finite difference scheme for one-dimensional burgers equation, *Appl. Math. Comput.*, vol. 206, no. 2, pp. 755-764, 2008. https://doi.org/10.1016/j.amc.2008.09.037

[20] V. Mukundan and A. Awasthi, "Efficient numerical techniques for burgers equation, *Appl. Math. Comput.*, vol. 262, pp. 282-297, 2015. https://doi.org/10.1016/j.amc.2015.03.122

[21] K. Pandey, L. Verma, and A. K. Verma, "On a finite difference scheme for burgers equation, *Appl. Math. Comput.*, vol. 215, no. 6, pp. 2206-2214, 2009. https://doi.org/10.1016/j.amc.2009.08.018

[22] K. Sakai and I. Kimura, "A numerical scheme based on a solution of nonlinear advectiondiffusion equations, *J. Comput. Appl. Math.*, vol. 173, no. 1, pp. 39-55, 2005. https://doi.org/10.1016/j.cam.2004.02.019

[23] S.-S. Xie, S. Heo, S. Kim, G. Woo, and S. Yi, "Numerical solution of one-dimensional burgers equation using reproducing kernel function, *J. Comput. Appl. Math.*, vol. 214, no. 2, pp. 417-434, 2008. https://doi.org/10.1016/j.cam.2007.03.010

[24] G. Zhao, X. Yu, and R. Zhang, "The new numerical method for solving the system of two-dimensional burgers equations, *Comput. Math. Appl.*, vol. 62, no. 8, pp. 3279-3291, 2011. https://doi.org/10.1016/j.camwa.2011.08.044

[25] E. Aksan, "A numerical solution of burgers equation by finite element method constructed on the method of discretization in time, *Appl. Math. Comput.*, vol. 170, no. 2, pp. 895-904, 2005. https://doi.org/10.1016/j.amc.2004.12.027

[26] J. Caldwell and P. Smith, "Solution of burgers equation with a large reynolds number, *Appl. Math. Model.*, vol. 6, no. 5, pp. 381-385, 1982. https://doi.org/10.1016/S0307-904X(82)80102-9

[27] J. Caldwell, P. Wanless, and A. Cook, "A finite element approach to burgers equation, *Appl. Math. Model.*, vol. 5, no. 3, pp. 189-193, 1981. https://doi.org/10.1016/0307-904X(81)90043-3

[28] Y. Chai and J. Ouyang, "Appropriate stabilized galerkin approaches for solving two-dimensional coupled burgers equations at high reynolds numbers, *Comput. Math. Appl.*, vol. 79, no. 5, pp. 1287-1301, 2020. https://doi.org/10.1016/j.camwa.2019.08.036

[29] G. Arora and B. K. Singh, "Numerical solution of burgersequationwithmodifiedcubicb-splinedifferential quadrature method, *Appl. Math. Comput.*, vol. 224, pp. 166-177, 2013. https://doi.org/10.1016/j.amc.2013.08.071

[30] M. Ghasemi, "An efficient algorithm based on extrapolation for the solution of nonlinear parabolic equations, *Int. J. Nonlinear Sci. Numer. Simul.*, vol. 19, no. 1, pp. 37-51, 2018. https://doi.org/10.1515/ijnsns-2017-0060

[31] B. K. Singh and M. Gupta, "A new efficient fourth order collocation scheme for solving burgers equation, *Appl. Math. Comput.*, vol. 399, p. 126011, 2021. https://doi.org/10.1016/j.amc.2021.126011

[32] M. Tamsir, N. Dhiman, and V. K. Srivastava, "Extended modified cubic b-spline algorithm for nonlinear burgers equation, *Beni-Suef Univ. J. Basic Appl. Sci.*, vol. 5, no. 3, pp. 244-254, 2016. https://doi.org/10.1016/j.bjbas.2016.09.001

[33] Y. Gao, L.-H. Le, and B.-C. Shi, "Numerical solution of burgers equation by lattice boltzmann method, *Appl. Math. Comput.*, vol. 219, no. 14, pp. 7685-7692, 2013. https://doi.org/10.1016/j.amc.2013.01.056

[34] N. Kumar, R. Majumdar, and S. Singh, "Predictorcorrector nodal integral method for simulation of high reynolds number fluid flow using larger time steps in burgers equation, *Comput. Math. Appl.*, vol. 79, no. 5, pp. 1362-1381, 2020. https://doi.org/10.1016/j.camwa.2019.09.001

[35] F. M. de Lara and E. Ferrer, "Accelerating high order discontinuous galerkin solvers using neural networks: 1d burgers equation, *Comput. & Fluids*, vol. 235, p. 105274, 2022. https://doi.org/10.1016/j.compfluid.2021.105274

[36] R. Bridson, Fluid Simulation for Computer Graphics, Second Edition. *Taylor & Francis*, 2015. ISBN 9781482232837.

[37] P. Olver, Introduction to Partial Differential Equations, ser. Undergraduate Texts in Mathematics. *Springer International Publishing*, 2013. ISBN 9783319020990.

[38] M. Stone and P. Goldbart, Mathematics for Physics: A Guided Tour for Graduate Students. *Cambridge University Press*, 2009. ISBN 9780521854030.

[39] M. Sarboland and A. Aminataei, "On the numerical solution of one-dimensional nonlinear nonhomogeneous burgers equation, *J. Appl. Math.*, vol. 2014, pp. 598432: 1-598432: 15, 2014. https://doi.org/10.1155/2014/598432

[40] E. R. Bentom and G. W. Platzman, "A table of solutions of the one-dimensional burgers equation, *Quart. Appl. Math.*, vol. 30, no. 2, pp. 195-212, 1972. https://doi.org/10.1090/qam/306736

[41] J. Ramos, "Picards iterative method for nonlinear advectionreactiondiffusion equations, *Appl. Math. Comput.*, vol. 215, no. 4, pp. 1526-1536, 2009. https://doi.org/10.1016/j.amc.2009.07.004