Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 8, No. 4, 2020, pp. 708-732
DOI:10.22034/cmde.2020.32615.1513

Applications of the Newton-Raphson method in a SDFEM for inviscid Burgers equation

Mohammad Izadi
Department of Applied Mathematics,
Faculty of Mathematics and Computer,
Shahid Bahonar University of Kerman, Kerman, Iran.
E-mail: izadi@uk.ac.ir

Abstract In this paper we investigate in detail the applications of the classical Newton-Raphson method in connection with a space-time finite element discretization scheme for the inviscid Burgers equation in one dimensional space. The underlying discretization method is the so-called *streamline diffusion* method, which combines good stability properties with high accuracy. The coupled nonlinear algebraic equations thus obtained in each space-time slab are solved by the generalized Newton-Raphson method. Exploiting the band-structured properties of the Jacobian matrix, two different algorithms based on the Newton-Raphson linearization are proposed. In a series of examples, we show that in each time-step a quadratic convergence order is attained when the Newton-Raphson procedure applied to the corresponding system of nonlinear equations.

Keywords. Finite element method, Burgers equation, Newton-Raphson method.
2010 Mathematics Subject Classification. 76M10, 35Q35, 65H10.

1. Introduction

The Burgers equation as a fundamental and simplest nonlinear scalar hyperbolic partial differential equation is a canonical model in various fields such as turbulence, acoustics, shock wave theory, hydrodynamic, and traffic flow [4, 9].

The streamline diffusion method (the SD-method for short) is a general finite-element method (FEM) and was first introduced by Hughes et al. [10] and [11], in order to cope with hyperbolic type problems specifically to treat the particular difficulties, like shocks and boundary layers, numerically. It is based on a least-squares modification of the standard Galerkin FEM on a general space-time mesh giving added stability without sacrificing accuracy; the error is of order $\mathcal{O}(h^{k+\frac{1}{2}})$ for smooth solutions when polynomials of degree k are used. The mathematical analysis of this method for linear problems, together with extensions to time-dependent problems using space-time elements, was started by Johnson et al. [15]. This methodology together with a solid mathematical analysis has now been successfully applied to various types of practical problems due to its good stability and high accuracy. Among others, we emphasize in the case of convection-diffusion problems [7], incompressible and compressible Euler and Navier-Stokes equations [16, 21], second-order wave equations [14], Vlasov-Poisson-Fokker-Planck system [1] as well as inverse scattering

Received: 19 March 2019 ; Accepted: 10 June 2019.

problems [2] and (viscous) conservation laws [13, 22] (see also references therein for more information).

The study of SD-method for hyperbolic conservation laws has been initialized by Johnson and Szepessy in [17, 18]. Afterwards, the SD-method and its modification, i.e., the shock-capturing SD-method has been developed rapidly in theory and practice in [19, 22]. In the shock-capturing variant artificial viscosity is added with the viscosity coefficient depending locally on the residual and the mesh parameter h.

Two main theoretical results for the basic SD-method applied to Burgers equation was proven in [17]:

A) If a sequence of finite element solutions converges (as the mesh size h tends to zero) boundedly a.e. to a function u, then u is an entropy solution of Burgers equation.

B) If the finite element solution stays uniformly bounded, then a subsequence will converge a.e. to a function u.

Subsequently, in [19] proved that the hypothesis of B) holds for the shock-capturing SD-method in the case $k = 1$. Combining the latter result with A), they obtained that a subsequence of the finite element solutions given by the shock-capturing SD-method with $k = 1$ converges to an entropy solution of Burgers equation.

However, in this study as an extension of the work in the above references, we are interested in the computational aspects behind the basic SD-method for the Burgers equation that has not yet been considered so far (to the best of our knowledge), rather than the theoretical counterparts. We will first review some of the theatrical results on this subject and will then look more closely at the ways one may solve the obtained nonlinear systems of equations arising from the discretization of Burgers equation by the SD-method. We will mainly be concerned with the application of the classical Newton-Raphson method in connection with a space-time finite element basis functions.

More specifically, we shall consider the Burgers equation in one dimensional space, i.e., finding a scalar function $u(x,t)$ such that

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \quad (x,t) \in \Omega := \mathbb{R} \times (0,\infty), \tag{1.1a}$$

with initial condition

$$u(x,0) = u_0(x), \quad x \in \mathbb{R}, \tag{1.1b}$$

where $u_0(x)$ is a given real valued function with compact support. Equation (1.1a) is a nonlinear hyperbolic conservation law $u_t + f(u)_x = 0$ with flux function $f(u) = u^2/2$.

It is well-known that even for smooth initial condition u_0, the initial boundary value problem (1.1a)-(1.1b) does not always have a solution in the classical sense, which means a solution that is continuously differentiable satisfying (1.1a)-(1.1b). Instead, solutions in the sense of distributions, called integral solutions, or weak solutions, must be considered (see for instance [6]). To be more precise, let us recall that a function $u \in L_\infty(\Omega)$, is a weak solution of (1.1) if for all test functions $\varphi \in C_0^\infty(\bar{\Omega}), \Omega := \mathbb{R} \times [0,\infty)$, we have

$$\int_\Omega \left(u\frac{\partial \varphi}{\partial t} + \frac{u^2}{2}\frac{\partial \varphi}{\partial x} \right) dxdt = -\int_{\mathbb{R}} u_0(x)\varphi(x,0)dx. \tag{1.2}$$

Unfortunately, in the class of weak solutions uniqueness does not hold in general. Therefore an additional criteria, the so-called entropy admissibility condition has to be satisfied for

the problem (1.1). We further recall that a weak solution u of $u_t + f(u)_x = 0$ is an entropy solution if the following entropy inequality holds

$$\int_\Omega \left(\eta(u)\frac{\partial \varphi}{\partial t} + q(u)\frac{\partial \varphi}{\partial x} \right) dxdt \geq 0, \tag{1.3}$$

for every test function $\varphi \in C_0^\infty(\Omega)$ with $\varphi \geq 0$ and for all entropy pairs (η, q) with convex entropy function η and the associated entropy flux q that satisfies the compatibility condition $q' = \eta' \cdot f'$. In the scalar case, every convex function η leads to an entropy pair by putting $q(u) := \int \eta'(u) f'(u) du$. This specific explicit formulation for the entropy pairs was used by Kruzhkov [20] to obtain existence, uniqueness and stability of solutions for scalar conservation laws.

This implies that for the numerical treatments of conservation laws, one needs to devise suitable algorithms not only to deal with discontinuous solutions but also select those solution that satisfy the entropy condition. As demonstrated in [17], the approximated SD solution for the Burgers equation has the satisfactory properties, including not only converging to a weak solution in the sense (1.2) but also capturing those weak solutions that satisfies the entropy condition (1.3). In the other words, they prove that a subsequence of solutions U_h of a SD-method for the Burgers equation converges almost every where to an entropy solution u of (1.1) corresponding to the entropy function $\eta = u^2/2$.

The content of this paper is organized as follows: In Section 2, we construct a space-time discretization and formulate the streamline diffusion method for the Burgers equation (1.1). In addition, we introduce some notations that will be used later on. Section 3 is devoted to the statement and proof of the basic stability estimates, the convergence theorem towards entropy solutions as well as a priori error estimates for the SD-method for the Burgers equation in a triple norm. In Section 4, we employ a space-time basis function and describe the algorithm formulation and practical implementation of the SD-method in detail. Hence, we briefly discuss the Newton-Raphson procedure to linearize the resulting coupled system of nonlinear equations in each space-time slab. Based on this linearization, two different algorithms then are proposed to solve the obtained linear systems. In computational Section 5, numerical results obtained by the SD-method, with emphasis on performance of two proposed algorithms are compared with the exact solutions. We provide conclusions in Section 7.

2. Space-time discretization

We shall formulate a finite element method for the Burgers equation in (1.1). We shall use the Galerkin method with piecewise linear basis functions which are continuous in space and discontinuous in time, i.e. the SD-method. To do this we first introduce some basic notation.

Let $\Delta := \{0 = t_0 < t_1 < \ldots < t_N = T\}$ be a subdivision of the time interval $I := (0, T]$ into an increasing sequence of discrete time-levels and set $I_n := (t_n, t_{n+1})$ to be the corresponding subintervals with time steps $k_n = t_{n+1} - t_n$ for $n = 0, 1, \ldots, N-1$. Next, we define the corresponding space-time strips or "slabs" as

$$\mathbb{S}_n := \big\{(x,t) : x \in \mathbb{R} \text{ and } t \in I_n\big\}, \quad n = 0, \ldots, N-1.$$

Further, for $h > 0$ and for $n = 1, \ldots, N-1$, let T_h^n be quasi-uniform partitions of the space-time slab $\mathbb{S}_n$ into space-time elements K of diameter h_K. Note that the partitionings of two consequent slab $\mathbb{S}_n$ and $\mathbb{S}_{n+1}$ may be chosen differently, but they must satisfy the standard quasi-uniformity conditions for finite element meshes (cf. [5]). This means that for each $K \in T_h^n$ there is an inscribed sphere in K such that the ratio of the diameter of this sphere and the h_K is bounded below, independently of K and h.

Let now q be a positive integer. By $\mathcal{P}_q(K)$ we denote the space of polynomials of degree less than or equal to q on $K \in T_h^n$. Associated with the slab $\mathbb{S}_n$, we introduce the finite element space

$$\mathcal{U}_h^n = \left\{ u \in H^1(\mathbb{S}_n) \ : \ u\big|_K \in \mathcal{P}_q(K), \ \forall K \in T_h^n \right\},$$

where H^1 denotes the usual Sobolev space of order one. Due to the fact that u_0 has compact support, the solution u has also compact support in $\mathbb{R} \times [0, t]$ for any $t > 0$. Thus, we may define the trial and test function spaces as the subspaces of $\mathcal{U}_h^n$ by

$$\mathcal{V}_h^n = \left\{ v \in \mathcal{U}_h^n \ : \ v(x,t) \to 0 \text{ as } |x| \to \pm\infty \right\},$$

Summing over n, taking all the slabs together we get the function spaces

$$\mathcal{V}_h = \prod_{n=0}^{N-1} \mathcal{V}_h^n.$$

Thus, we shall seek an approximate solution $U_h \in \mathcal{V}_h$ such that for $n = 0, 1, ..., N-1$ we will have that $U_h\big|_{\mathbb{S}_n} = U_h^n$.

We emphasize that a function in $\mathcal{V}_h$ is continuous within each $\mathbb{S}_n$, and in particular continuous in x everywhere. It may be discontinuous in t at discrete time levels t_n, i.e., across the lines $\mathbb{L}_n := \mathbb{R} \times \{t_n\}$. Hence a function $U_h \in \mathcal{V}_h$ has always two values on both sides of $\mathbb{L}_n$ denoted by $U_{h,+}^n$ and $U_{h,-}^n$, where

$$U_{h,+}^n(\cdot) = \lim_{s \longrightarrow 0+} U_h(\cdot, t_n + s), \qquad U_{h,-}^n(\cdot) = \lim_{s \longrightarrow 0-} U_h(\cdot, t_n + s).$$

Therefore, it is natural to introduce the jump terms $[U_h]$ across each time level by defining, for $x > 0$ and $n = 0, 1, \ldots, N-1$,

$$[U_h](\cdot, t_n) = \begin{cases} U_{h,+}^0(\cdot) & \text{if } n = 0 \\ U_{h,+}^n(\cdot) - U_{h,-}^n(\cdot) & \text{if } n \neq 0. \end{cases}$$

We shall use the following notation: Given a domain Q, we denote by $(\cdot, \cdot)_Q$ the usual $L_2(Q)$ scalar product, $\|\cdot\| = \|\cdot\|_{L_2(Q)}$ the corresponding L_2 norm, and for a positive integer s, $H^s(Q)$ will denote the usual Sobolev space of functions with square integrable derivative of order less than or equal s and with the norm $\|\cdot\|_{s,Q}$. We also write $\|\cdot\|_{\infty,Q} = \|\cdot\|_{L_\infty(Q)}$. We also write

$$(u,v)_n := \int_{\mathbb{S}_n} uv dx dt, \qquad \|v\|_n := \sqrt{(v,v)_n},$$

$$\langle u, v \rangle_n := \int_{\mathbb{L}_n} u(x,t) v(x,t) dx, \qquad |v|_n := \sqrt{\langle v, v \rangle_n}.$$

Furthermore, we use the convention that

$$(u,v)_{\Omega_N} = \sum_{n=0}^{N-1} (u,v)_n,$$

where $\Omega_N := \mathbb{R}\times(0,t_N)$. In what follows, C and c will denote positive constants, independent of h, and not necessarily the same at each occurrence, unless explicitly stated otherwise. Normally if C and c appear in the same chain of estimates, then we shall mean $Cc = \mathcal{O}(1)$.

3. The SD-method and the basic stability estimate

Now the SD-method as an discrete approximation to (1.1) can be successively formulated as follows: Given $U^n_{h,-}$, for some $n \in \{0,1,\dots,N-1\}$, find $U^n_h \in \mathcal{V}^n_h$, such that

$$\Big(U^n_{h,t} + U^n_h\, U^n_{h,x}, w^n_h + \delta(w^n_{h,t} + U^n_h\, w^n_{h,x})\Big)_n + \langle U^n_{h,+}, w^n_{h,+}\rangle_n = \langle U^n_{h,-}, w^n_{h,+}\rangle_n, \quad (3.1)$$

where $U^0_{h,-} = u^0$ is the initial data and $\delta = ch$ for some appropriately chosen positive constant c. The system (3.1) is nonlinear in U^n_h, which then requires a certain linearization. For this there are many possibilities, a simple fixed-point iteration or the more sophisticated Newton method (see Section 5, below).

In order to write the above SD formulation in a compact form suitable for analysis, after summing over $n = 0,\dots,N-1$ we may rewrite (3.1) as follows: Find $U \equiv U_h \in \mathcal{V}_h$ such that

$$\mathcal{B}(U,w) = \mathcal{L}(w), \quad \forall w \in \mathcal{V}_h, \quad (3.2)$$

where

$$\mathcal{B}(U,w) := \big(U_t + UU_x, w + \delta(w_t + Uw_x)\big)_{\Omega_N} + \sum_{n=0}^{N-1} \langle [U], w_+\rangle_n,$$

and

$$\mathcal{L}(w) := \langle u_0, w_+\rangle_0.$$

The stability of the SD-method is a consequence of the coercivity properties of the bilinear form $\mathcal{B}$ in (3.2). The next lemma gives us a basic stability estimate for (3.2) (see also [17] and [12]):

Lemma 3.1. *For $U \in \mathcal{V}_h$, and with compactly supported boundary conditions we have*

$$\mathcal{B}(U,U) \geq |||U|||^2,$$

where

$$|||U|||^2 := \frac{1}{2}\left[|U_-|^2_N + |U_+|^2_0 + \sum_{n=1}^{N-1} |[U]|^2_n\right] + \delta\|U_t + UU_x\|^2_{\Omega_N}.$$

Proof: By setting $w = U$ in the definition of $\mathcal{B}$ in (3.2) we get

$$B(U,U) = (U_t + UU_x, U)_{\Omega_N} + \delta\|U_t + UU_x\|^2_{\Omega_N} + \sum_{n=0}^{N-1} \langle [U], U_+\rangle_n.$$

Applying the integrating by parts for the first term and using the compactly supported boundary conditions yields

$$(U_t + UU_x, U)_{\Omega_N} = (U_t, U)_{\Omega_N}.$$

On the other hand, an easy calculation shows that

$$(U_t, U)_{\Omega_N} = \frac{1}{2}\left[\sum_{n=1}^{N} |U_-|_n^2 - |U_+|_0^2 - \sum_{n=1}^{N-1} |U_+|_n^2\right].$$

Collecting the terms as

$$(U_t, U)_{\Omega_N} + \sum_{n=1}^{N-1} \langle [U], U_+\rangle_n + \langle U_+, U_+\rangle_0 = \frac{1}{2}\left[|U_-|_N^2 + |U_+|_0^2 + \sum_{n=1}^{N-1} |[U]|_n^2\right],$$

and putting into the above formula will complete the proof. ■

Taking $w = U$ in (3.2), utilizing the triple norm defined in Lemma 3.1, and applying the following form of Young's inequality

$$\alpha\beta \le \epsilon\alpha^2 + \frac{\beta^2}{\epsilon},$$

which holds for any real numbers $\alpha, \beta \in \mathbb{R}$ and $\epsilon > 0$, one can easily conclude the following estimate:

Corollary 3.2. *Under the assumptions of Lemma 3.1, the following estimate holds*

$$\frac{1}{2}\left[|U_-|_N^2 + \sum_{n=1}^{N-1} |[U]|_n^2\right] + \delta\|U_t + UU_x\|_{\Omega_N}^2 \le \frac{1}{2}|u_0|_0^2. \tag{3.3}$$

To proceed we also use the following estimate for $\|U(t)\|_{\Omega_N}$ for all $t > 0$, a proof of which can be found in [17] or [12]:

Lemma 3.3. *For any $c > 0$ and with the compactly supported boundary conditions, we have for $U \in \prod_{n=0}^{N-1} H^1(\mathbb{S}_n)$ that*

$$\|U(t)\|_{\Omega_N}^2 \le \left[\sum_{n=1}^{N} |U_-|_n^2 + \frac{1}{C}\|U_t + UU_x\|_{\Omega_N}^2\right] \exp(Ch). \tag{3.4}$$

With a comparison between equations (3.3) and (3.4), it is not a difficult task to show that the approximated solutions $U \equiv U_h$ bounded by

$$\|U(t)\|_{\Omega_N} \le C\|u_0\|_{\Omega_N}, \quad t > 0. \tag{3.5}$$

It remains to show that the method (3.1) cannot produce discrete solutions converging to a nonphysical solution of (1.1). The following theorem proves that if the solutions U_h of (3.1) converge boundedly almost everywhere to a function u, then u satisfies (1.2) and (1.3) with $\eta = u^2/2$; a proof of which can be found in [17]:

Theorem 3.4. *Let the solutions U_h of (3.1) converge boundedly a.e. in Ω to a function u as h tends to zero, so that in particular for $h > 0$,*

$$\|U_h\|_{\infty,\Omega} \le C.$$

Then u satisfies (1.2) and (1.3), and thus u is an entropy solution of (1.1).

3.1. **An a priori error estimate for the SD-method.** Let us now state without proof a basic a priori error estimate for the SD-method (3.2); for the exact solution in the Sobolev space H^{k+1}, we show that the error is of order $\mathcal{O}(h^{k+1/2})$. The foundation of the proof is relied on the concepts of *Galerkin orthogonality* and introducing the linear nodal interpolant of the exact solution (see [12]).

Theorem 3.5. *If $U_h \in \mathcal{V}_h$ satisfies (3.2) and the exact solution u satisfies (1.1), and further*

$$\|u\|_{\infty,\Omega} \leq c,$$

then there is a constant C such that

$$|||u - U_h||| \leq Ch^{k+\frac{1}{2}} \|u\|_{k+1,\Omega}. \tag{3.6}$$

4. The coupled systems of nonlinear equations

To implement the space-time SD-method (3.1) in practice, let $\{\phi_1, \phi_2, \cdots, \phi_m\}$ be a spatial basis for $\mathcal{V}_h^n$. Take $\{\theta_1, \theta_2\}$ as the basis for the space $\mathcal{P}_1$ and defined on the interval $I_n = (t_n, t_{n+1})$ as

$$\theta_1(t) = \frac{t_{n+1} - t}{k}, \quad \theta_2(t) = \frac{t - t_n}{k}.$$

Now, the space-time basis $\{\Psi_{11}, \Psi_{12}, \ldots, \Psi_{1m}, \Psi_{21}, \Psi_{22}, \ldots, \Psi_{2m}\}$ of $\mathcal{V}_h$ can be constructed as follows:

$$\Psi_{11}(x,t) = \phi_1(x)\theta_1(t), \Psi_{12}(x,t) = \phi_2(x)\theta_1(t), \ldots, \Psi_{1m}(x,t) = \phi_m(x)\theta_1(t),$$
$$\Psi_{21}(x,t) = \phi_1(x)\theta_2(t), \Psi_{22}(x,t) = \phi_2(x)\theta_2(t), \ldots, \Psi_{2m}(x,t) = \phi_m(x)\theta_2(t).$$

This means that we use finite element approximation on a space-time slab with the trial functions which are piecewise polynomials in space and linear in time. Therefore, for $(x,t) \in \mathbb{S}_n$, we let

$$U_h^n(x,t) = \sum_{i=1}^{m} \left(\Psi_{1i}(x,t)\tilde{U}_i^n + \Psi_{2i}(x,t)U_i^{n+1}\right), \tag{4.1}$$

where, the nodal values of u for node i at $(t_n)^+$ and $(t_{n+1})^-$ are denoted by $\tilde{U}_i^n$ and U_i^{n+1}, respectively. The test functions for each space-time slab are defined as $\Psi_{1j}(x,t)$ and $\Psi_{2j}(x,t)$ for $j = 1, \ldots, m$.

Being the approximate solution on $\mathbb{S}_n$ by U_h^n as in (4.1), substituting in (3.1), we arrive at the following systems of nonlinear equations of order $2m$:

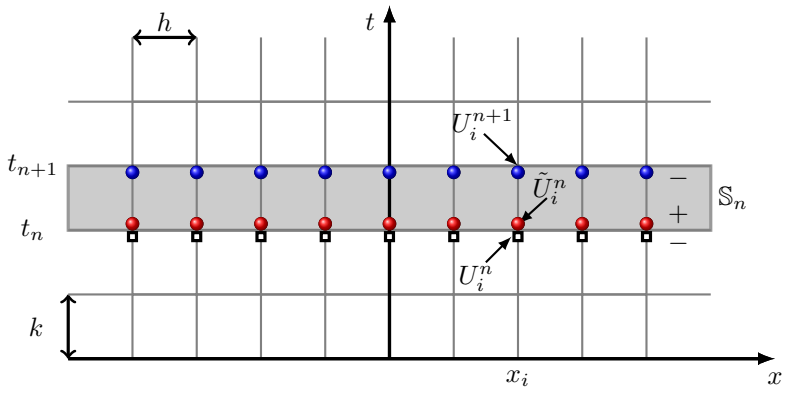
$$\sum_{i=1}^{m} \int_{\mathbb{S}_n} \left[\frac{\partial \Psi_{1i}}{\partial t}\tilde{U}_i^n + \frac{\partial \Psi_{2i}}{\partial t}U_i^{n+1} + \left\{\sum_{l=1}^{m}(\Psi_{1l}\tilde{U}_l^n + \Psi_{2l}U_l^{n+1})\right\}\left(\frac{\partial \Psi_{1i}}{\partial x}\tilde{U}_i^n + \frac{\partial \Psi_{2i}}{\partial x}U_i^{n+1}\right)\right] \tag{4.2a}$$
$$\times \left[\Psi_{1j} + \delta\left(\frac{\partial \Psi_{1j}}{\partial t} + \left\{\sum_{l=1}^{m}(\Psi_{1l}\tilde{U}_l^n + \Psi_{2l}U_l^{n+1})\right\}\frac{\partial \Psi_{1j}}{\partial x}\right)\right] dxdt = 0,$$

and

$$\sum_{i=1}^{m}\int_{\mathbb{S}_n}\left[\frac{\partial\Psi_{1i}}{\partial t}\tilde{U}_i^n+\frac{\partial\Psi_{2i}}{\partial t}U_i^{n+1}+\left\{\sum_{l=1}^{m}(\Psi_{1l}\tilde{U}_l^n+\Psi_{2l}U_l^{n+1})\right\}\left(\frac{\partial\Psi_{1i}}{\partial x}\tilde{U}_i^n+\frac{\partial\Psi_{2i}}{\partial x}U_i^{n+1}\right)\right]$$
$$\times\left[\Psi_{2j}+\delta\left(\frac{\partial\Psi_{2j}}{\partial t}+\left\{\sum_{l=1}^{m}(\Psi_{1l}\tilde{U}_l^n+\Psi_{2l}U_l^{n+1})\right\}\frac{\partial\Psi_{2j}}{\partial x}\right)\right]dxdt \tag{4.2b}$$
$$+\sum_{i=1}^{m}\int_{\mathbb{R}}\phi_i(x)\phi_j(x)\left(\tilde{U}_i^n-U_i^n\right)dx=0,$$

for all $j=1,\ldots,m$. We emphasize that in the two above equations (4.2), the unknown nodal values $\tilde{U}_i^n$ and U_i^{n+1} at the points marked by solid circles in Fig. 1 while the known nodal values U_i^n as given data are indicated by open rectangles in Fig. 1. Let the spatial basis

FIGURE 1. A uniform mesh of size h and k.



functions ϕ_i for $i=1,2,\ldots,m$ be the piecewise linear finite elements satisfying $\phi_i(x_j)=\delta_{ij}$ for $1\le i,j\le m$, i.e.,

$$\phi_i(x)=\begin{cases}\frac{1}{h}(x-x_{i-1}) & x\in[x_{i-1},x_i],\\ \frac{1}{h}(x_{i+1}-x) & x\in[x_i,x_{i+1}],\\ 0 & x\notin[x_{i-1},x_{i+1}].\end{cases}$$

After substituting into equations (4.2), the entries of different kind of matrices $\boldsymbol{M},\boldsymbol{N},\boldsymbol{O}$, $\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R}$, $\boldsymbol{S},\boldsymbol{T},\boldsymbol{U}$, $\boldsymbol{V},\boldsymbol{X},\boldsymbol{Y}$ need to be evaluated. Due to the fact that the selected basis

functions have finite compact support, most elements of these matrices become zero.

$$\boldsymbol{m}_{ij} := \int_{\mathbb{R}} \phi_i \phi_j dx = \begin{cases} \pm\frac{h}{6}, & j = i \pm 1, \\ \frac{2h}{3}, & j = i, \\ 0, & \text{elsewhere}, \end{cases}$$

$$\boldsymbol{n}_{ij} := \int_{\mathbb{R}} \phi_i \phi_{i-1} \phi_j' dx = \begin{cases} -\frac{1}{6}, & j = i-1, \\ \frac{1}{6}, & j = i, \\ 0, & \text{elsewhere}, \end{cases}$$

$$\boldsymbol{o}_{ij} := \int_{\mathbb{R}} \phi_i^2 \phi_j' dx = \begin{cases} \pm\frac{1}{3}, & j = i \pm 1, \\ 0, & \text{elsewhere}, \end{cases}$$

$$\boldsymbol{p}_{ij} := \int_{\mathbb{R}} \phi_i \phi_{i+1} \phi_j' dx = \begin{cases} 0, & \text{elsewhere}, \\ -\frac{1}{6}, & j = i, \\ \frac{1}{6}, & j = i+1. \end{cases}$$

Also we have that

$$\boldsymbol{q}_{ij} := \textstyle\int_{\mathbb{R}} \phi_{i-1} \phi_i' \phi_j dx = \begin{cases} \frac{1}{3}, & j = i-1, \\ \frac{1}{6}, & j = i, \\ 0, & \text{elsewhere}, \end{cases} \qquad \boldsymbol{r}_{ij} := \textstyle\int_{\mathbb{R}} \phi_{i-1}^2 \phi_i' \phi_j' dx = \frac{2}{h} \boldsymbol{n}_{ij},$$

$$\boldsymbol{s}_{ij} := \textstyle\int_{\mathbb{R}} \phi_{i-1} \phi_i \phi_i' \phi_j' dx = \frac{1}{h} \boldsymbol{n}_{ij}, \qquad \boldsymbol{t}_{ij} := \textstyle\int_{\mathbb{R}} \phi_i \phi_i' \phi_j dx = -\frac{1}{h} \boldsymbol{o}_{ij},$$

$$\boldsymbol{u}_{ij} := \textstyle\int_{\mathbb{R}} \phi_i^2 \phi_i' \phi_j' dx = \begin{cases} -\frac{1}{3h}, & j = i \pm 1, \\ \frac{2}{3h}, & j = i, \\ 0, & \text{elsewhere}, \end{cases} \qquad \boldsymbol{v}_{ij} := \textstyle\int_{\mathbb{R}} \phi_i \phi_i' \phi_{i+1} \phi_j' dx = -\frac{1}{h} \boldsymbol{p}_{ij},$$

$$\boldsymbol{x}_{ij} := \textstyle\int_{\mathbb{R}} \phi_{i+1} \phi_i' \phi_j dx = \begin{cases} 0, & \text{elsewhere}, \\ -\frac{1}{6}, & j = i, \\ -\frac{1}{3}, & j = i+1, \end{cases} \qquad \boldsymbol{y}_{ij} := 2\boldsymbol{v}_{ij}.$$

Further using the fact that we have for $i, j = 1, 2$

$$\int_{I_n} \theta_i(t) dt = \frac{k}{2}, \qquad \int_{I_n} \theta_i(t) \theta_j(t) dt = \frac{k}{6}, \qquad \int_{I_n} \theta_i^2(t) dt = \frac{k}{3},$$

$$\int_{I_n} \theta_i^3(t) dt = \frac{k}{4}, \qquad \int_{I_n} \theta_i^2(t) \theta_j^2(t) dt = \frac{k}{30}, \qquad \int_{I_n} \theta_i^4(t) dt = \frac{k}{5},$$

$$\int_{I_n} \theta_i^2(t) \theta_j(t) dt = \frac{k}{12}, \qquad \int_{I_n} \theta_i^3(t) \theta_j(t) = \frac{k}{20},$$

we can rewrite equations (4.2a)-(4.2b) as a $2m \times 2m$ system of nonlinear equations:

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}) = \boldsymbol{0}, \tag{4.3}$$

where

$$\boldsymbol{\mathcal{F}} := \begin{bmatrix} \boldsymbol{\mathcal{F}}^1 \\ \boldsymbol{\mathcal{F}}^2 \end{bmatrix}, \quad \boldsymbol{\mathcal{U}} := \begin{bmatrix} \widetilde{\boldsymbol{\mathcal{U}}^n} \\ \boldsymbol{\mathcal{U}}^{n+1} \end{bmatrix}.$$

The unknown solution vectors $\boldsymbol{\mathcal{U}}^{n+1}$ and $\widetilde{\boldsymbol{\mathcal{U}}^n}$ at the discrete time levels $(t_{n+1})^-$ and $(t_n)^+$ are defined as follows (see Fig. (1))

$$\boldsymbol{\mathcal{U}}^{n+1} := (U_1^{n+1}, U_2^{n+1}, \ldots, U_m^{n+1})^T, \quad \widetilde{\boldsymbol{\mathcal{U}}^n} := (\tilde{U}_1^n, \tilde{U}_2^n, \ldots, \tilde{U}_m^n)^T.$$

Similarly, at each slab $\mathbb{S}_n$ the known solution vector $\boldsymbol{\mathcal{U}}^n$ at the time level $(t_n)^-$ is denoted by $\boldsymbol{\mathcal{U}}^n := (U_1^n,\ U_2^n, \ldots, U_m^n)^T$. Looking at equation(4.2b), it is clear that for computing $\boldsymbol{\mathcal{F}}^2$ at the slab $\mathbb{S}_n$ one needs to have $\boldsymbol{\mathcal{U}}^n$, which is already known from the preceding slab $\mathbb{S}_{n-1}$. Obviously this vector at the first time level $(t_0)^-$ is computed as $\boldsymbol{\mathcal{U}}^0 := (u_1^0,\ u_2^0, \ldots, u_m^0)^T$, where $u_j^0 = u(x_j, t_0) = u_0(x_j)$ for $j = 1, \ldots, m$.

Remark 4.1. Alternatively, one can combine $\boldsymbol{\mathcal{U}}^{n+1}$ and $\widetilde{\boldsymbol{\mathcal{U}}^n}$ to construct the components of the single vector solution as

$$\boldsymbol{\mathcal{U}}_i = \begin{cases} (\widetilde{\boldsymbol{\mathcal{U}}^n})_i & \text{if i odd,} \\ (\boldsymbol{\mathcal{U}}^{n+1})_{\frac{i}{2}} & \text{if i even,} \end{cases}$$

or $\boldsymbol{\mathcal{U}} := (\tilde{U}_1^n, U_1^{n+1}, \tilde{U}_2^n, U_2^{n+1}, \ldots, \tilde{U}_m^n, U_m^{n+1})^T$, so that we have again (4.3). In this case, the Jacobian matrix will be utilized in the Newton-Raphson procedure (see 5.4 below) has no longer a block-wise structure with four tridiagonal submatrices (see Fig. 2), but will be a matrix with two pentadiagonal subblocks.

To proceed , by defining the following constants

$$c_1^{\pm} := \frac{1}{2} \pm \frac{\delta}{k},\ c_2^{\pm} := \frac{k}{12} \pm \frac{\delta}{6},\ c_3^{\pm} := \frac{k}{12} \pm \frac{\delta}{3},\ c_4^{\pm} := \frac{k}{6} \pm \frac{\delta}{3},\ c_5^{\pm} := \frac{k}{4} \pm \frac{\delta}{3}, \tag{4.4}$$

the elements of vectors $\boldsymbol{\mathcal{F}}^1 = \boldsymbol{\mathcal{F}}^1(\widetilde{\boldsymbol{\mathcal{U}}^n}, \boldsymbol{\mathcal{U}}^{n+1})$ and $\boldsymbol{\mathcal{F}}^2 = \boldsymbol{\mathcal{F}}^2(\boldsymbol{\mathcal{U}}^n, \widetilde{\boldsymbol{\mathcal{U}}^n}, \boldsymbol{\mathcal{U}}^{n+1})$ can more compactly be expressed. For $j = 1, 2, \ldots, m$ we have

$$\begin{aligned}
\boldsymbol{\mathcal{F}}_j^1 = \sum_{i=1}^{m} \Big\{ & c_1^- \boldsymbol{m}_{ij}(U_i^{n+1} - \tilde{U}_i^n) + (\frac{\delta}{3}\boldsymbol{n}_{ij} + c_2^- \boldsymbol{q}_{ij})\tilde{U}_{i-1}^n U_i^{n+1} + \frac{2\delta k}{15}\boldsymbol{s}_{ij}\tilde{U}_i^n \tilde{U}_{i-1}^{n+1} U_i^{n+1} + \\
& (\frac{\delta}{6}\boldsymbol{n}_{ij} + c_3^- \boldsymbol{q}_{ij})U_{i-1}^{n+1} U_i^{n+1} + (\frac{\delta}{6}\boldsymbol{o}_{ij} + c_4^- \boldsymbol{t}_{ij})\tilde{U}_i^n U_i^{n+1} + \frac{\delta k}{5}\boldsymbol{u}_{ij}\big((\tilde{U}_i^n)^3 + \frac{3}{4}(\tilde{U}_i^n)^2 U_i^{n+1}\big) \\
& + (\frac{\delta}{6}\boldsymbol{o}_{ij} + c_3^- \boldsymbol{t}_{ij})(U_i^{n+1})^2 + (\frac{\delta}{3}\boldsymbol{p}_{ij} + c_2^- \boldsymbol{x}_{ij})\tilde{U}_{i+1}^n U_i^{n+1} + (\frac{\delta}{6}\boldsymbol{p}_{ij} + c_3^- \boldsymbol{x}_{ij})U_i^{n+1} U_{i+1}^{n+1} - \\
& (\frac{\delta}{3}\boldsymbol{n}_{ij} - c_5^- \boldsymbol{q}_{ij})\tilde{U}_{i-1}^n \tilde{U}_i^n - (\frac{\delta}{6}\boldsymbol{n}_{ij} - c_2^- \boldsymbol{q}_{ij})\tilde{U}_i^n U_{i-1}^{n+1} - (\frac{\delta}{6}\boldsymbol{p}_{ij} - c_2^- \boldsymbol{x}_{ij})\tilde{U}_i^n U_{i+1}^{n+1} - \\
& (\frac{\delta}{3}\boldsymbol{o}_{ij} - c_5^- \boldsymbol{t}_{ij})(\tilde{U}_i^n)^2 - (\frac{\delta}{3}\boldsymbol{p}_{ij} - c_5^- \boldsymbol{x}_{ij})\tilde{U}_i^n \tilde{U}_{i+1}^n + \frac{\delta k}{5}\boldsymbol{s}_{ij}\tilde{U}_{i-1}^n\big(2(\tilde{U}_i^n)^2 + \frac{1}{3}(U_i^{n+1})^2\big) + \\
& \frac{\delta k}{30}\boldsymbol{r}_{ij}\tilde{U}_i^n (U_{i-1}^{n+1})^2 + \frac{\delta k}{5}\boldsymbol{r}_{ij}(\tilde{U}_{i-1}^n)^2 \tilde{U}_i^n + \frac{\delta k}{10}\boldsymbol{r}_{ij}\tilde{U}_{i-1}^n \tilde{U}_i^n U_{i-1}^{n+1} + \frac{\delta k}{20}\boldsymbol{r}_{ij}(\tilde{U}_{i-1}^n)^2 U_i^{n+1} + \\
& \frac{\delta k}{5}\boldsymbol{s}_{ij}\big(\tilde{U}_{i-1}^n \tilde{U}_i^n U_i^{n+1} + \frac{1}{2}(\tilde{U}_i^n)^2 U_{i-1}^{n+1} + \frac{1}{2}U_{i-1}^{n+1}(U_i^{n+1})^2\big) + \frac{\delta k}{15}\boldsymbol{r}_{ij}\tilde{U}_{i-1}^n U_{i-1}^{n+1} U_i^{n+1} + \\
& \frac{\delta k}{20}\boldsymbol{r}_{ij}(U_{i-1}^{n+1})^2 U_i^{n+1} + \frac{2\delta k}{5}\boldsymbol{v}_{ij}(\tilde{U}_i^n)^2 \tilde{U}_{i+1}^n + \frac{\delta k}{10}\boldsymbol{v}_{ij}(\tilde{U}_i^n)^2 U_{i+1}^{n+1} + \frac{\delta k}{5}\boldsymbol{v}_{ij}\tilde{U}_i^n \tilde{U}_{i+1}^n U_i^{n+1} + \\
& \frac{\delta k}{10}\boldsymbol{u}_{ij}\tilde{U}_i^n (U_i^{n+1})^2 + + \frac{2\delta k}{15}\boldsymbol{v}_{ij}\tilde{U}_i^n U_i^{n+1} U_{i+1}^{n+1} + \frac{\delta k}{20}\boldsymbol{u}_{ij}(U_i^{n+1})^3 + \frac{\delta k}{15}\boldsymbol{v}_{ij}\tilde{U}_{i+1}^n (U_i^{n+1})^2 + \\
& \frac{\delta k}{10}\boldsymbol{v}_{ij}(U_i^{n+1})^2 U_{i+1}^{n+1} + \frac{\delta k}{5}\boldsymbol{y}_{ij}\tilde{U}_i^n (\tilde{U}_{i+1}^n)^2 + \frac{\delta k}{10}\boldsymbol{y}_{ij}\tilde{U}_i^n \tilde{U}_{i+1}^n U_{i+1}^{n+1} + \frac{\delta k}{30}\boldsymbol{y}_{ij}\tilde{U}_i^n (U_{i+1}^{n+1})^2 + \\
& \frac{\delta k}{20}\boldsymbol{y}_{ij}(\tilde{U}_{i+1}^n)^2 U_i^{n+1} + \frac{\delta k}{15}\boldsymbol{y}_{ij}\tilde{U}_{i+1}^n U_i^{n+1} U_{i+1}^{n+1} + \frac{\delta k}{20}\boldsymbol{y}_{ij}U_i^{n+1}(U_{i+1}^{n+1})^2 \Big\} = 0.
\end{aligned}$$

(4.5)

Similarly, for $\boldsymbol{\mathcal{F}}_j^2$ we have

$$\begin{aligned}
\boldsymbol{\mathcal{F}}_j^2 = \sum_{i=1}^{m} \Big\{ & \boldsymbol{m}_{ij}(c_1^+ U_i^{n+1} + c_1^- \tilde{U}_i^n) + (\frac{\delta}{6}\boldsymbol{n}_{ij} + c_2^+ \boldsymbol{q}_{ij})\tilde{U}_{i-1}^n U_i^{n+1} + \frac{\delta k}{15}\boldsymbol{r}_{ij}\tilde{U}_{i-1}^n \tilde{U}_i^n U_{i-1}^{n+1} + \\
& (\frac{\delta}{3}\boldsymbol{n}_{ij} + c_5^+ \boldsymbol{q}_{ij})U_{i-1}^{n+1} U_i^{n+1} - (\frac{\delta}{6}\boldsymbol{o}_{ij} - c_4^+ \boldsymbol{t}_{ij})\tilde{U}_i^n U_i^{n+1} + \frac{\delta k}{10}\boldsymbol{r}_{ij}(\tilde{U}_{i-1}^n)^2\big(\frac{\tilde{U}_i^n}{2} + \frac{U_i^{n+1}}{3}\big) + \\
& (\frac{\delta}{3}\boldsymbol{o}_{ij} + c_5^+ \boldsymbol{t}_{ij})(U_i^{n+1})^2 + (\frac{\delta}{6}\boldsymbol{p}_{ij} + c_2^+ \boldsymbol{x}_{ij})\tilde{U}_{i+1}^n U_i^{n+1} + \frac{\delta k}{5}\boldsymbol{s}_{ij}\tilde{U}_{i-1}^n \tilde{U}_i^n\big(\frac{\tilde{U}_i^n}{2} + \frac{2U_i^{n+1}}{3}\big) + \\
& (\frac{\delta}{3}\boldsymbol{p}_{ij} + c_5^+ \boldsymbol{x}_{ij})U_i^{n+1} U_{i+1}^{n+1} - (\frac{\delta}{6}\boldsymbol{n}_{ij} - c_3^+ \boldsymbol{q}_{ij})\tilde{U}_{i-1}^n \tilde{U}_i^n + \frac{\delta k}{5}\boldsymbol{r}_{ij}(U_{i-1}^{n+1})^2\big(\frac{\tilde{U}_i^n}{4} + U_i^{n+1}\big) - \\
& (\frac{\delta}{3}\boldsymbol{n}_{ij} - c_2^+ \boldsymbol{q}_{ij})\tilde{U}_i^n U_{i-1}^{n+1} - (\frac{\delta}{6}\boldsymbol{o}_{ij} - c_3^+ \boldsymbol{t}_{ij})(\tilde{U}_i^n)^2 + \frac{\delta k}{5}\boldsymbol{s}_{ij}(U_i^{n+1})^2\big(2U_{i-1}^{n+1} + \frac{\tilde{U}_{i-1}^n}{2}\big) - \\
& (\frac{\delta}{6}\boldsymbol{p}_{ij} - c_3^+ \boldsymbol{x}_{ij})\tilde{U}_i^n \tilde{U}_{i+1}^n - (\frac{\delta}{3}\boldsymbol{p}_{ij} - c_2^+ \boldsymbol{x}_{ij})\tilde{U}_i^n U_{i+1}^{n+1} + \frac{\delta k}{5}\boldsymbol{s}_{ij}\tilde{U}_i^n U_{i-1}^{n+1}\big(U_i^{n+1} + \frac{\tilde{U}_i^n}{3}\big) + \\
& \frac{\delta k}{10}\boldsymbol{r}_{ij}\tilde{U}_{i-1}^n U_{i-1}^{n+1} U_i^{n+1} + \frac{\delta k}{10}\boldsymbol{u}_{ij}(\tilde{U}_i^n)^2\big(\frac{\tilde{U}_i^n}{2} + U_i^{n+1}\big) + \frac{\delta k}{5}\boldsymbol{v}_{ij}(\tilde{U}_i^n)^2\big(\frac{\tilde{U}_{i+1}^n}{2} + \frac{2U_{i+1}^{n+1}}{3}\big) + \\
& \frac{\delta k}{5}\boldsymbol{u}_{ij}(U_i^{n+1})^2\big(U_i^{n+1} + \frac{3\tilde{U}_i^n}{4}\big) + \frac{\delta k}{5}\boldsymbol{v}_{ij}\tilde{U}_i^n U_i^{n+1}\big(\frac{2\tilde{U}_{i+1}^n}{3} + U_{i+1}^{n+1}\big) + \frac{\delta k}{20}\boldsymbol{y}_{ij}\tilde{U}_i^n(\tilde{U}_{i+1}^n)^2 + \\
& \frac{\delta k}{5}\boldsymbol{v}_{ij}(U_i^{n+1})^2\big(\frac{\tilde{U}_{i+1}^n}{2} + 2U_{i+1}^{n+1}\big) + \frac{\delta k}{5}\boldsymbol{y}_{ij}\tilde{U}_i^n U_{i+1}^{n+1}\big(\frac{\tilde{U}_{i+1}^n}{3} + \frac{U_{i+1}^{n+1}}{4}\big) + \frac{\delta k}{30}\boldsymbol{y}_{ij}(\tilde{U}_{i+1}^n)^2 U_i^{n+1} \\
& + \frac{\delta k}{10}\boldsymbol{y}_{ij}\tilde{U}_{i+1}^n U_i^{n+1} U_{i+1}^{n+1} + \frac{\delta k}{5}\boldsymbol{y}_{ij}U_i^{n+1}(U_{i+1}^{n+1})^2 - \boldsymbol{m}_{ij}U_i^n \Big\} = 0.
\end{aligned} \tag{4.6}$$

In the following, the main task is in the computation of the nonlinear system (4.3). This process relied on the Newton-Raphson method to linearize the coupled nonlinear equations (4.3) so that the computed solution is obtained by iteration in each time-step.

5. Numerical algorithms: Newton-Raphson linearization

To show the advantages of the SD-method (3.1) for the Burgers equation (1.1) numerically, one needs to solve the nonlinear systems of equations (4.3) in each space-time slab $\mathbb{S}_n$. As there several schemes for solving systems of nonlinear equations such as (4.3), the methods based on Newton-Raphson method which we also utilize here, are used much more often in practice (see [3] for a historical note on the notions, Newton or Newton-Raphson method). It is known that this method converges quadratically once the approximation is close to the actual solution of the given nonlinear system.

Let us briefly describe the main idea of the Newton-Raphson method. Given a (continuously differentiable) function $\boldsymbol{\mathcal{F}} : \mathbb{R}^s \longrightarrow \mathbb{R}^s$, the goal is to find the root $\boldsymbol{\mathcal{U}}_\star \in \mathbb{R}^s$ such that $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}_\star) = \mathbf{0}$. Suppose $\boldsymbol{\mathcal{U}}_0$ be some known initial guess for the solution vector $\boldsymbol{\mathcal{U}}_\star$. Approximating $\boldsymbol{\mathcal{F}}$ by the linear part of the Taylor series we have

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}_\star) \approx \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}_0) + J\boldsymbol{\mathcal{F}}\big|_{\boldsymbol{\mathcal{U}}=\boldsymbol{\mathcal{U}}_0}\delta\boldsymbol{\mathcal{U}},$$

where $J\boldsymbol{\mathcal{F}} := \frac{d\boldsymbol{\mathcal{F}}}{d\boldsymbol{\mathcal{U}}}$ denotes the Jacobian matrix and $\delta\boldsymbol{\mathcal{U}}$ is a small displacement between $\boldsymbol{\mathcal{U}}_\star$ and $\boldsymbol{\mathcal{U}}_0$, i.e., $\delta\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{U}}_\star - \boldsymbol{\mathcal{U}}_0$, which can be found by solving

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}_0) + J\boldsymbol{\mathcal{F}}\big|_{\boldsymbol{\mathcal{U}}=\boldsymbol{\mathcal{U}}_0}\delta\boldsymbol{\mathcal{U}} = \mathbf{0}.$$

By applying the linearization process repeatedly, one can devise the following iterative scheme to get closer to the root $\boldsymbol{\mathcal{U}}_\star$, which may be not known in most practical problems:

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{(l)}) + J\boldsymbol{\mathcal{F}}\big|_{\boldsymbol{\mathcal{U}}=\boldsymbol{\mathcal{U}}^{(l)}}\Delta\boldsymbol{\mathcal{U}} = 0, \tag{5.1}$$

where superscript l is an iteration index and $\Delta\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{U}}^{(l+1)} - \boldsymbol{\mathcal{U}}^{(l)}$ is the increment. Obviously, equation (5.1) represents a linear system of equations needs to be solved in each iteration step l in order to get the increment $\Delta\boldsymbol{\mathcal{U}}$:

$$\big[J\boldsymbol{\mathcal{F}}\big]\big|_{\boldsymbol{\mathcal{U}}^{(l)}}\Delta\boldsymbol{\mathcal{U}} = -\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{(l)}). \tag{5.2}$$

As an iterative procedure, a stopping criterion is required for the convergence. Chosen an appropriate norm $\|\cdot\|$ and given the predefined tolerance parameter $0 < \mathtt{tol}_f \ll 1$, or $0 < \mathtt{tol}_u \ll 1$, one may use either the condition $\|\Delta\boldsymbol{\mathcal{U}}\| < \mathtt{tol}_u$, or $\|\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{(l)})\| < \mathtt{tol}_f$ to get the desired accuracy.

Let now come back to our nonlinear system (4.3) and rewrite it as (see Figure. 1)

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}) = \begin{bmatrix} \boldsymbol{\mathcal{F}}^1(\boldsymbol{\mathcal{U}}_\oplus, \boldsymbol{\mathcal{U}}_\ominus) \\ \boldsymbol{\mathcal{F}}^2(\boldsymbol{\mathcal{U}}_\boxminus, \boldsymbol{\mathcal{U}}_\oplus, \boldsymbol{\mathcal{U}}_\ominus) \end{bmatrix} = \mathbf{0} \quad \text{with} \quad \boldsymbol{\mathcal{U}} = \begin{bmatrix} \boldsymbol{\mathcal{U}}_\oplus \\ \boldsymbol{\mathcal{U}}_\ominus \end{bmatrix} := \begin{bmatrix} \widetilde{\boldsymbol{\mathcal{U}}^n} \\ \boldsymbol{\mathcal{U}}^{n+1} \end{bmatrix}, \quad \boldsymbol{\mathcal{U}}_\boxminus := \boldsymbol{\mathcal{U}}^n. \tag{5.3}$$

Using Eq. (5.2) we arrive at the representation

$$\begin{bmatrix} \frac{\partial\boldsymbol{\mathcal{F}}^1}{\partial\boldsymbol{\mathcal{U}}_\oplus} & \frac{\partial\boldsymbol{\mathcal{F}}^1}{\partial\boldsymbol{\mathcal{U}}_\ominus} \\ \frac{\partial\boldsymbol{\mathcal{F}}^2}{\partial\boldsymbol{\mathcal{U}}_\oplus} & \frac{\partial\boldsymbol{\mathcal{F}}^2}{\partial\boldsymbol{\mathcal{U}}_\ominus} \end{bmatrix}\Bigg|_{\boldsymbol{\mathcal{U}}^{(l)}} \begin{bmatrix} \Delta\boldsymbol{\mathcal{U}}_\oplus \\ \Delta\boldsymbol{\mathcal{U}}_\ominus \end{bmatrix} = - \begin{bmatrix} \boldsymbol{\mathcal{F}}^1(\boldsymbol{\mathcal{U}}^{(l)}) \\ \boldsymbol{\mathcal{F}}^2(\boldsymbol{\mathcal{U}}_\boxminus, \boldsymbol{\mathcal{U}}^{(l)}) \end{bmatrix}, \tag{5.4}$$

with

$$\boldsymbol{\mathcal{U}}^{(l)} := \begin{bmatrix} \boldsymbol{\mathcal{U}}_\oplus^{(l)} \\ \boldsymbol{\mathcal{U}}_\ominus^{(l)} \end{bmatrix}, \quad \Delta\boldsymbol{\mathcal{U}}_\oplus := \boldsymbol{\mathcal{U}}_\oplus^{(l+1)} - \boldsymbol{\mathcal{U}}_\oplus^{(l)}, \quad \Delta\boldsymbol{\mathcal{U}}_\ominus := \boldsymbol{\mathcal{U}}_\ominus^{(l+1)} - \boldsymbol{\mathcal{U}}_\ominus^{(l)}.$$

The Jacobian matrix in the algorithm of the Newton-Raphson method is a block-tridiagonal matrix due to choosing the basis function in the finite element method as linear functions. This implies that all $m \times m$ matrices $\frac{\partial\boldsymbol{\mathcal{F}}^i}{\partial\boldsymbol{\mathcal{U}}_\oplus}$ and $\frac{\partial\boldsymbol{\mathcal{F}}^i}{\partial\boldsymbol{\mathcal{U}}_\ominus}$ for $i = 1, 2$ are tridiagonal matrices. For $m = 19$, the sparsity pattern of the Jacobian matrix is plotted in Figure. 2.

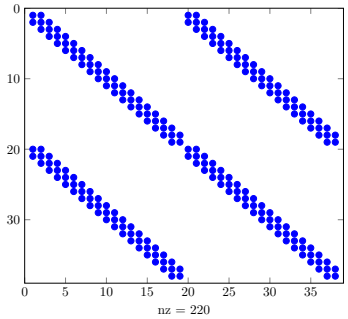
The system (5.4) can be solved in one single step or by taking into account the band-structured properties of the matrices $\frac{\partial\boldsymbol{\mathcal{F}}^i}{\partial\boldsymbol{\mathcal{U}}_\oplus}$, $\frac{\partial\boldsymbol{\mathcal{F}}^i}{\partial\boldsymbol{\mathcal{U}}_\ominus}$ (see [8]). Solving the first equation in (5.4) for $\Delta\boldsymbol{\mathcal{U}}_\oplus$ we obtain that

$$\Delta\boldsymbol{\mathcal{U}}_\oplus = -\left[\frac{\partial\boldsymbol{\mathcal{F}}^1}{\partial\boldsymbol{\mathcal{U}}_\oplus}\Bigg|_{\boldsymbol{\mathcal{U}}^{(l)}}\right]^{-1}\left(\boldsymbol{\mathcal{F}}^1(\boldsymbol{\mathcal{U}}^{(l)}) + \left[\frac{\partial\boldsymbol{\mathcal{F}}^1}{\partial\boldsymbol{\mathcal{U}}_\ominus}\Bigg|_{\boldsymbol{\mathcal{U}}^{(l)}}\right]\Delta\boldsymbol{\mathcal{U}}_\ominus\right) =: \boldsymbol{\mathcal{G}}_1(\Delta\boldsymbol{\mathcal{U}}_\ominus, \boldsymbol{\mathcal{U}}^{(l)}). \tag{5.5}$$

In order to obtain the increment $\Delta\boldsymbol{\mathcal{U}}_\ominus$, after inserting the former relation into the second equation in (5.4), one needs to solve

$$\boldsymbol{\mathcal{G}}_2(\boldsymbol{\mathcal{U}}^{(l)})\Delta\boldsymbol{\mathcal{U}}_\oplus = \boldsymbol{\mathcal{G}}_3(\boldsymbol{\mathcal{U}}_\boxminus, \boldsymbol{\mathcal{U}}^{(l)}), \tag{5.6}$$

FIGURE 2. The sparsity structure of Jacobian matrix for $m = 19$.



where,

$$\mathcal{G}_2(\mathcal{U}^{(l)}) := \left(\frac{\partial \mathcal{F}^2}{\partial \mathcal{U}_\ominus}\Big|_{\mathcal{U}^{(l)}} - \frac{\partial \mathcal{F}^2}{\partial \mathcal{U}_\oplus}\Big|_{\mathcal{U}^{(l)}} \left[\frac{\partial \mathcal{F}^1}{\partial \mathcal{U}_\oplus}\Big|_{\mathcal{U}^{(l)}} \right]^{-1} \frac{\partial \mathcal{F}^2}{\partial \mathcal{U}_\ominus}\Big|_{\mathcal{U}^{(l)}} \right),$$

$$\mathcal{G}_3(\mathcal{U}_\boxminus, \mathcal{U}^{(l)}) := -\mathcal{F}^2(\mathcal{U}_\boxminus, \mathcal{U}^{(l)}) + \frac{\partial \mathcal{F}^2}{\partial \mathcal{U}_\oplus}\Big|_{\mathcal{U}^{(l)}} \left[\frac{\partial \mathcal{F}^1}{\partial \mathcal{U}_\oplus}\Big|_{\mathcal{U}^{(l)}} \right]^{-1} \mathcal{F}^1(\mathcal{U}^{(l)}).$$

To summarize, the first Algorithm 1 will use the above described Newton-Raphson procedure in each time step t_{n+1} for $n = 0, \ldots, N-1$ to solve the nonlinear system (5.2). Algorithm start at the slab $\mathbb{S}_0$ with the solution vector $\mathcal{U}^0$ computed as the initial condition at grid points. Hence the solution vectors $\mathcal{U}^1, \mathcal{U}^2, \ldots, \mathcal{U}^N$ are computed iteratively by means of the Newton-Raphson algorithm.

We emphasize, the difference between two vectors $\mathcal{U}_\boxminus$ and $\mathcal{U}^{(0)}$ as the initial guesses in Algorithm 1. The vector $\mathcal{U}^{(0)}$ (in line 4) is taken as zero in all time levels t_n for $n = 1, \ldots, N$ before the computations involved in the Newton-Raphson procedure (lines 5-9). On the contrary, the vector $\mathcal{U}_\boxminus$ in each time step t_n, is initialized by the approximated solution at the previous time step t_{n-1} and mainly utilized in the evaluation of the vector $\mathcal{F}^2$ (see (4.6)). In the first initialization, i.e., at $t_0 = 0$, it is exact and is taken as $\mathcal{U}^0$ while will be updated in the all subsequent initializations.

The computation of the inverse matrix $[\frac{\partial \mathcal{F}^1}{\partial \mathcal{U}_\oplus}|_{\mathcal{U}^{(l)}}]^{-1}$ in the formula (5.5) and (5.6) may be inefficient. Instead by defining the quantities [8]

$$\mathcal{Z}_1 := \left[\frac{\partial \mathcal{F}^1}{\partial \mathcal{U}_\oplus}\Big|_{\mathcal{U}^{(l)}} \right]^{-1} \cdot \frac{\partial \mathcal{F}^1}{\partial \mathcal{U}_\ominus}\Big|_{\mathcal{U}^{(l)}}, \quad \mathcal{Z}_2 := \left[\frac{\partial \mathcal{F}^1}{\partial \mathcal{U}_\oplus}\Big|_{\mathcal{U}^{(l)}} \right]^{-1} \cdot \mathcal{F}^1(\mathcal{U}^{(l)}),$$

one can solve the the system of linear equations (5.4) with several right hand sides

$$\frac{\partial \mathcal{F}^1}{\partial \mathcal{U}_\oplus}\Big|_{\mathcal{U}^{(l)}} \cdot [\mathcal{Z}_1 \mid \mathcal{Z}_2] = \left[\frac{\partial \mathcal{F}^1}{\partial \mathcal{U}_\ominus}\Big|_{\mathcal{U}^{(l)}} \mid \mathcal{F}^1(\mathcal{U}^{(1)}) \right]. \tag{5.7}$$

Therefore, Algorithm 2 as an modification of the previous Algorithm 1, will compute the solution of nonlinear system 5.2 without using any direct matrix inversion. In the following

function NewtonI $(h,\delta,k,\boldsymbol{\mathcal{U}}^0,\boldsymbol{M},\boldsymbol{N},\boldsymbol{O},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R},\boldsymbol{S},\boldsymbol{T},\boldsymbol{X},\boldsymbol{Y})$;
Input : The mesh parameters h,δ,k, the initial solution vector $\boldsymbol{\mathcal{U}}^0$, and the nonzero entries of tridiagonal matrices $\boldsymbol{M},\boldsymbol{N},\boldsymbol{O},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R},\boldsymbol{S},\boldsymbol{T},\boldsymbol{X},\boldsymbol{Y}$.
Output: Discrete solution vectors $\boldsymbol{\mathcal{U}}^{n+1}$, $0 \le n \le N-1$.
$\boldsymbol{\mathcal{U}}_{\boxminus} \leftarrow \boldsymbol{\mathcal{U}}^0 = (u_1^0, u_2^0, \ldots, u_m^0)^T$;
for $i \leftarrow 1$ ***to*** $N-1$ **do**
  $\boldsymbol{\mathcal{U}}^{(0)} = \begin{bmatrix} \boldsymbol{\mathcal{U}}_{\ominus}^{(0)} \\ \boldsymbol{\mathcal{U}}_{\oplus}^{(0)} \end{bmatrix} \leftarrow \mathbf{0}$;
  repeat $l = 0, 1, \ldots$
    Solve $\boldsymbol{\mathcal{G}}_2(\boldsymbol{\mathcal{U}}^{(l)})\Delta\boldsymbol{\mathcal{U}}_{\ominus} = \boldsymbol{\mathcal{G}}_3(\boldsymbol{\mathcal{U}}_{\boxminus},\boldsymbol{\mathcal{U}}^{(l)})$; $\rightsquigarrow \Delta\boldsymbol{\mathcal{U}}_{\ominus}$
    $\Delta\boldsymbol{\mathcal{U}}_{\oplus} \leftarrow \boldsymbol{\mathcal{G}}_1(\Delta\boldsymbol{\mathcal{U}}_{\ominus},\boldsymbol{\mathcal{U}}^{(l)})$;
    Update $\boldsymbol{\mathcal{U}}^{(l+1)} \leftarrow \boldsymbol{\mathcal{U}}^{(l)} + \begin{bmatrix} \Delta\boldsymbol{\mathcal{U}}_{\oplus} \\ \Delta\boldsymbol{\mathcal{U}}_{\ominus} \end{bmatrix}$;
  until *convergence criteria is fulfilled*;
  $\boldsymbol{\mathcal{U}}_{\boxminus} \leftarrow \boldsymbol{\mathcal{U}}_{\ominus}^{(l+1)}$;
end

Algorithm 1: Newton-Raphson scheme for solving the coupled nonlinear system of equations.

numerical part, we will test and compare the performance and efficiency of Algorithm 1 and Algorithm 2.

6. Numerical experiments

In this section, we present some results of computations using the SD-method described in the preceding sections to test its accuracy and efficiency when applied to the Burgers equation with emphasizing on the performance of two proposed Algorithms 1 and 2. We solve this problem using the SD-method on uniform meshes with the mesh width $h := \Delta x = 2a/N$ obtained by partitioning the domain $[-a,a], a \in \mathbb{R}$ into N subintervals with $N = 25, 50, 100, 200, 400, 800, 1600$, and using the space $\mathcal{P}_1 \times \mathcal{P}_1$, i.e., with the basis functions continuous piecewise linear in space and discontinuous piecewise constant in time defined in (4.1). The time interval $[0,T]$ is divided into $nt = \lceil \frac{T}{k} \rceil$ small time-step $k := \Delta t$. Here, to ensure the CFL condition, the time-step is taken as $k/h = 0.5$. Moreover, for the inviscid Burgers equation, by T_s we denote the time at which the characteristics cross and a shock forms. This "breaking" time, can be determined exactly as $T_s = \frac{-1}{\min u_0'(x)}$.

To measure the accuracy of the numerical algorithms, we compute the difference between the analytic and numerical solutions. For this purpose, we calculate the discrete L_*-norm error, i.e., $\|e_h\|_* = \|U_h - u_{exact}\|_*$, where $*$ stands for L_1, L_2 or L_∞ norm. The relative error norms of numerical solutions U_h are defined by

$$E_{*,h} = \frac{\|e_h\|_*}{\|u_{exact}\|_*}.$$

function NewtonII $(h,\delta,k,\boldsymbol{\mathcal{U}}^0,\boldsymbol{M},\boldsymbol{N},\boldsymbol{O},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R},\boldsymbol{S},\boldsymbol{T},\boldsymbol{X},\boldsymbol{Y})$;

Input : The mesh parameters h,δ,k, the initial solution vector $\boldsymbol{\mathcal{U}}^0$, and the nonzero entries of tridiagonal matrices $\boldsymbol{M},\boldsymbol{N},\boldsymbol{O},\boldsymbol{P},\boldsymbol{Q},\boldsymbol{R},\boldsymbol{S},\boldsymbol{T},\boldsymbol{X},\boldsymbol{Y}$.

Output: Discrete solution vectors $\boldsymbol{\mathcal{U}}^{n+1}$, $0 \le n \le N-1$.

$\boldsymbol{\mathcal{U}}_{\boxminus} \leftarrow \boldsymbol{\mathcal{U}}^0 = (u_1^0, u_2^0, \ldots, u_m^0)^T$;

for $i \leftarrow 1$ ***to*** $N-1$ **do**

  $\boldsymbol{\mathcal{U}}^{(0)} = \begin{bmatrix} \boldsymbol{\mathcal{U}}_{\ominus}^{(0)} \\ \boldsymbol{\mathcal{U}}_{\oplus}^{(0)} \end{bmatrix} \leftarrow \boldsymbol{0}$;

  repeat $l = 0, 1, \ldots$

    Solve $\frac{\partial \boldsymbol{\mathcal{F}}^1}{\partial \boldsymbol{\mathcal{U}}_{\oplus}}\big|_{\boldsymbol{\mathcal{U}}^{(l)}} \cdot [\boldsymbol{Z}_1 \mid \boldsymbol{Z}_2] = \left[\frac{\partial \boldsymbol{\mathcal{F}}^1}{\partial \boldsymbol{\mathcal{U}}_{\ominus}}\big|_{\boldsymbol{\mathcal{U}}^{(l)}} \mid \boldsymbol{\mathcal{F}}^1(\boldsymbol{\mathcal{U}}^{(1)})\right]$; $\rightsquigarrow \boldsymbol{Z}_1, \boldsymbol{Z}_2$

    $\boldsymbol{\mathcal{G}}_2(\boldsymbol{\mathcal{U}}^{(l)}) \leftarrow \frac{\partial \boldsymbol{\mathcal{F}}^2}{\partial \boldsymbol{\mathcal{U}}_{\ominus}}\big|_{\boldsymbol{\mathcal{U}}^{(l)}} - \frac{\partial \boldsymbol{\mathcal{F}}^2}{\partial \boldsymbol{\mathcal{U}}_{\oplus}}\big|_{\boldsymbol{\mathcal{U}}^{(l)}} \cdot \boldsymbol{Z}_1$;

    $\boldsymbol{\mathcal{G}}_3(\boldsymbol{\mathcal{U}}_{\boxminus},\boldsymbol{\mathcal{U}}^{(l)}) \leftarrow -\boldsymbol{\mathcal{F}}^2(\boldsymbol{\mathcal{U}}_{\boxminus},\boldsymbol{\mathcal{U}}^{(l)}) + \frac{\partial \boldsymbol{\mathcal{F}}^2}{\partial \boldsymbol{\mathcal{U}}_{\oplus}}\big|_{\boldsymbol{\mathcal{U}}^{(l)}} \cdot \boldsymbol{Z}_2 \cdot \boldsymbol{\mathcal{F}}^1(\boldsymbol{\mathcal{U}}^{(l)})$;

    Solve $\boldsymbol{\mathcal{G}}_2(\boldsymbol{\mathcal{U}}^{(l)})\Delta\boldsymbol{\mathcal{U}}_{\ominus} = \boldsymbol{\mathcal{G}}_3(\boldsymbol{\mathcal{U}}_{\boxminus},\boldsymbol{\mathcal{U}}^{(l)})$; $\rightsquigarrow \Delta\boldsymbol{\mathcal{U}}_{\ominus}$

    $\Delta\boldsymbol{\mathcal{U}}_{\oplus} \leftarrow -\boldsymbol{Z}_1 - \boldsymbol{Z}_2 \cdot \Delta\boldsymbol{\mathcal{U}}_{\ominus}$;

    Update $\boldsymbol{\mathcal{U}}^{(l+1)} \leftarrow \boldsymbol{\mathcal{U}}^{(l)} + \begin{bmatrix} \Delta\boldsymbol{\mathcal{U}}_{\oplus} \\ \Delta\boldsymbol{\mathcal{U}}_{\ominus} \end{bmatrix}$;

  until *convergence criteria is fulfilled*;

  $\boldsymbol{\mathcal{U}}_{\boxminus} \leftarrow \boldsymbol{\mathcal{U}}_{\ominus}^{(l+1)}$;

end

Algorithm 2: A modified Newton-Raphson scheme for solving the coupled nonlinear system of equations.

Moreover, we compute the order of convergence rate of the SD-scheme through defining the convergence ratio $r = \frac{\|e_h\|_*}{\|e_{h/2}\|_*}$ and then calculating $\log_2(r)$.

The convergence (stopping) criterion used in Algorithms 1 and 2 is in the form

$$\left\|\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{(l)})\right\|_2 < \texttt{tol}_f,$$

where the predefined tolerance $\texttt{tol}_f$ is taken by default as 10^{-4}. However, for comparison we may consider different values for $\texttt{tol}_f$ ranging from 10^{-4} to ϵ_M as the machine precision. In addition, the starting point $\boldsymbol{\mathcal{U}}^{(0)}$ in all time-steps is taken as a zero vector. All methods were implemented in Matlab and run on a personal laptop computer. On our system $\epsilon_M = 2.22 \times 10^{-16}$ in double precision.

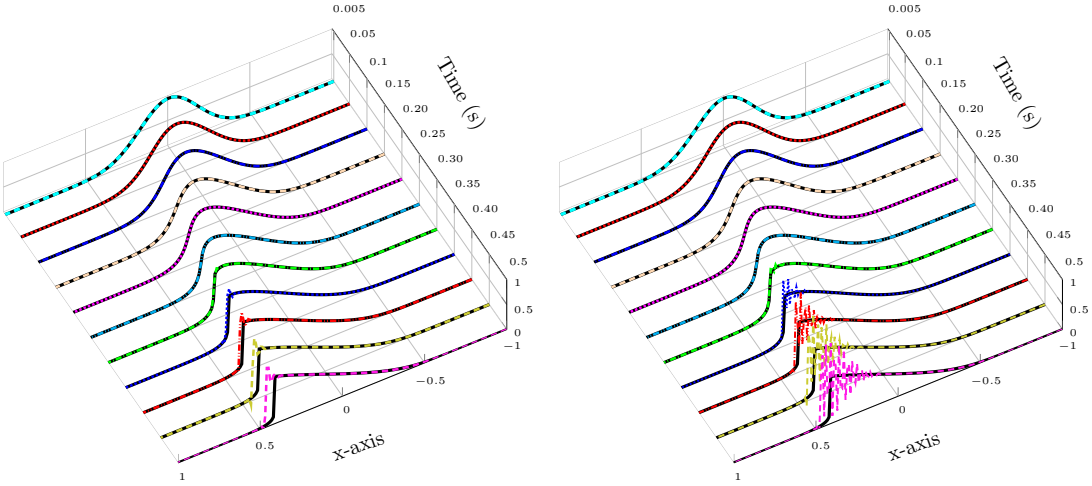
Example 6.1. We first consider the inviscid Burgers equation (1.1) with the continuous initial profile given by

$$u_0(x) = \exp(-16x^2),$$

which is a smooth Gaussian pulse function centered at $x = 0$. The computational domain is $[-1, 1]$ and final time is $T = 0.5$. The corresponding exact solution develops a shock at time $T_s = \sqrt{exp(1)/32} \approx 0.2915$.

First, for a fixed mesh size $\Delta x = 0.01$ we plot the numerical solutions as well as the exact solutions at different times $t = \Delta t, 10\Delta t, 20\Delta t, \ldots, 90\Delta t$, and $t = 100\Delta t$ using the SD-method with $\delta = h$ and $\delta = 0$ in Fig. 3. Note that choosing the SD parameter $\delta = 0$ corresponds to the standard Galerkin method. From Fig. 3 one can observe that both the SD and Galerkin mehtods work well and the computed results are in excellent agreement with the exact solutions before the shock formation time T_s. However, at the presence of the discontinuity it is evident that for $\delta = 0$, the Galerkin method does not work well and often produces widely oscillatory solutions. On the other hand, for $\delta = h$, the SD-method adds a numerical diffusion in the direction of the streamline to suppress oscillation. Of course, one must take care of selecting the parameter δ as an appropriate multiple of mesh size h to have more accurate results (see also [17]).

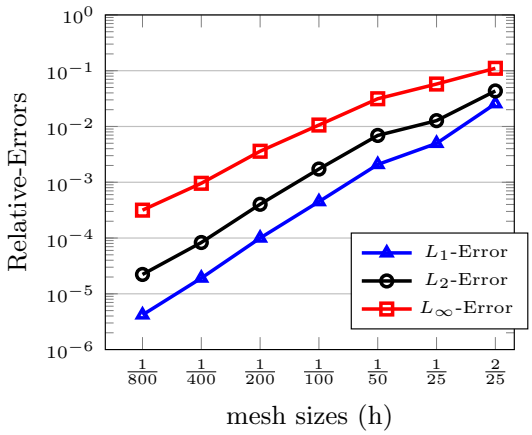
FIGURE 3. Numerical (dashed lines) and exact (solid lines) solutions of the Burgers equation correspond to Example 6.1 with the SD parameters $\delta = h$ (left) and $\delta = 0$ (right) at different times $t = s\Delta t, s = 1, 10, 20, \ldots, 100$.



In the next figure, we use different number of spatial elements $N = 25, 50, \ldots, 1600$, and measure the relative errors in the various norms L_1, L_2, and L_∞. These results that evaluated at time $t = 0.25$ $(< T_s)$ are visualized in Figure. 4, left. Furthermore, the corresponding convergence rates are reported in the same Figure. 4, right, for a fixed degree of polynomials $(k = 1)$ while the mesh size h is decreased. The numerical experiments shown in Figure. 4 indicate that achieving an order $(k + 1/2)$ or even an (optimal) order $(k + 1)$ of accuracy is possible, if one uses the SD-method with polynomials of degrees of k, here $k = 1$.

In order to compare the performance of Algorithm 1 with Algorithm 2, we investigate the number of Newton-Raphson iterations required by them with respect to the attainable accuracy. Let us fix a mesh size $h = 0.02$. Thus our time-step becomes $k = 0.01$ and therefore we have $nt = 50$ number of time-step. The two next figures show not only the required total number of iterations (left plot), but also present the corresponding L_2-norm of $\|\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{(l)})\|_2 < \texttt{tol}_f$ in the log scale (right plot), in each time-step number $1 \leq l \leq nt$.

FIGURE 4. Relative L_1, L_2, and L_∞-errors (left) and the corresponding convergence rates (right) evaluated at time $t = 0.25$ for different N.



N	L_1-order	L_2-order	L_∞-order
25	-	-	-
50	2.36	1.77	0.94
100	1.26	0.88	0.88
200	2.21	2.00	1.57
400	2.17	2.10	1.56
800	2.39	2.28	1.91
1600	2.19	1.90	1.60

Three values of tolerance are taken as $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$. A comparison between

FIGURE 5. Number of iterations (left) and the corresponding reached accuracies (right) versus the number of time-steps in Algorithm 1 for different tolerance $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$ for Example 6.1 with $\delta = h = 0.02$.

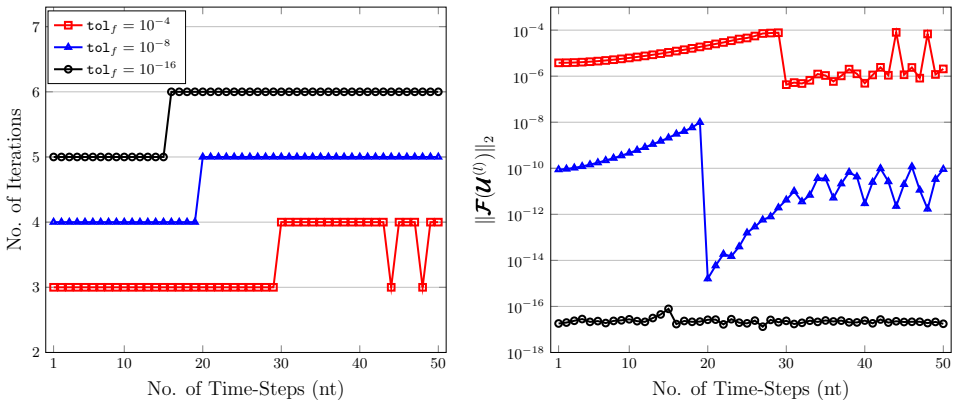
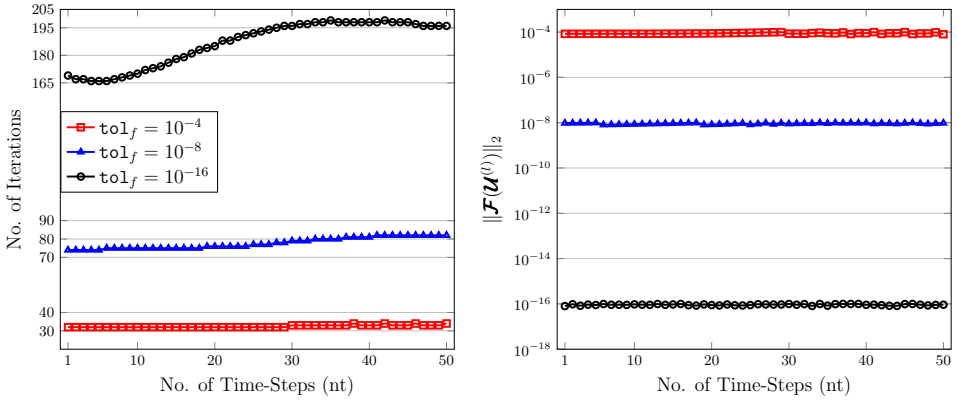


Figure. 5 and Figure. 6, it may well turn out that Algorithm 1 is superior to Algorithm 2 for the SD-method applied to the Burgers equation with the initial data given in Example 6.1. In average, the number of iterations obtained by Algorithm 1 during $nt = 50$ time-steps correspond to $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$ are $3.38, 4.62$, and 5.70, while in the same situation by Algorithm 2 these values are $32.50, 77.78$, and 186.68 respectively. On the other hand, the iteration sequence obtained by Algorithm 1, shows a quadratic convergence as expected in the Newton-Raphson procedure. This means that the norm of $\|\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{(l)})\|_2$ goes down

FIGURE 6. Number of iterations (left) and the corresponding reached accuracies (right) versus the number of time-steps in Algorithm 2 for different tolerance $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$ for Example 6.1 with $\delta = h = 0.02$.



from 10^{-4} in iteration numbers $3, 4$ to 10^{-8} in iteration numbers $4, 5$ and 10^{-16} in iteration numbers $5, 6$. In other words, the number of significant digits of accuracy doubles with every iteration. The former fact is not hold for the results achieved by Algorithm 2 and one observes an exponential growth in the number of iterations once we reach the desired accuracies, as indicated in Figure. 6. In the case of Galerkin method ($\delta = 0$), a moderate number of iterations will be needed by Algorithm 2 as illustrated in Figure. 7, but by exploiting Algorithm 1 the number of iterations remains the same as in the case $\delta = h$.

Example 6.2. As the second test case, we consider the inviscid Burgers equation (1.1) with piecewise continuous initial data $u_0(x)$ defined as

$$u_0(x) = \begin{cases} 0, & |x| \geq 1/4, \\ 1/2 + 2x, & -1/4 < x \leq 0, \\ 1/2 - 2x, & 0 < x \leq 1/4. \end{cases}$$

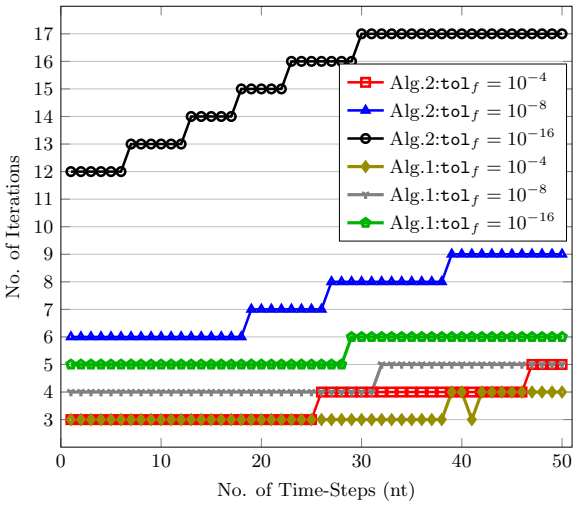
The exact solution which develops a shock at $x = 1/4$ and $T_s = 1/2$ is given by

$$u(x, t < T_s) = \begin{cases} 0, & |x| \geq \frac{1}{4}, \\ \dfrac{\frac{1}{2} + 2x}{1 + 2t}, & -\frac{1}{4} < x < \frac{t}{2}, \\ \dfrac{\frac{1}{2} - 2x}{1 - 2t}, & \frac{t}{2} < x < \frac{1}{4}, \end{cases}$$

$$u(x, t \geq T_s) = \begin{cases} 0, & x < -\frac{1}{4} \text{ or } x > x_s(t), \\ \dfrac{\frac{1}{2} + 2x}{1 + 2t}, & -\frac{1}{4} < x < \frac{t}{2}, \end{cases}$$

where $x_s(t) = \frac{-1+\sqrt{2+4t}}{4}$. Our spatial domain is restricted to $[-0.5, 0.5]$ and $T = 0.6$.

FIGURE 7. Number of iterations versus the number of time-steps in Algorithms 1 and 2 for different tolerance $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$ for Example 6.1 with $h = 0.02$ and $\delta = 0$.



In this example, we take $\Delta x = 0.02$ and $\texttt{tol}_f = 10^{-8}$. Based on these assumptions, we depict the numerical and exact solutions at different times $t = \Delta t, 6\Delta t, 12\Delta t, \ldots, 54\Delta t$, and $t = T$ in Figure. 8. We use the SD-method via Algorithm 1 with parameter $\delta = h$.

Looking at Figure. 8, one can observe that the approximated solutions obtained with the SD-method is almost perfectly aligned with the exact solutions before the shock formation time. However, after time T_s a slightly oscillation can be seen in the proximity of steep gradients.

Similarly, we will investigate in detail the behavior of Algorithm 1 for the second test case. For $\delta = h = 0.02$ and using $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$, the number of iterations l needed in each time-step as well as the corresponding L_2-norm of $\|\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{(l)})\|_2$ are visualized in Figure. 9. Under the same assumptions, the minimum and maximum number of iterations required by Algorithm 2 correspond to $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$ are $26 - 27$, $69 - 71$, and $153 - 157$ respectively. Obviously, the experiments in Figure. 9 show that the order of convergence of the Newton-Raphson in Algorithm 1 is equal to 2, whereas the modified version of Algorithm 1, i.e., Algorithm 2 fails to provide a quadratic convergence. Finally for the this example, we examine the behavior of relative errors in the L_1, L_2, and L_∞ norms with respect to varying the spatial mesh size h. The corresponding convergence rates are also obtained and represented in Figure. 10. These results that are evaluated at time $t = 0.4$ show that the order of convergence of the SD-method in the L_1-norm is at least $\frac{3}{2}$, which justified the theatrical results (see Theorem 3.5).

Example 6.3. The third test case, we propose for Burgers equation, is the propagation of initial square wave in different directions. This test case contains both expansion of

FIGURE 8. Numerical (dashed lines) and exact (solid lines) solutions of the Burgers equation correspond to Example 6.2 with the SD parameters $\delta = h = 0.02$ at different times $t = s\Delta t, s = 1, 6, 12, \ldots, 60$.

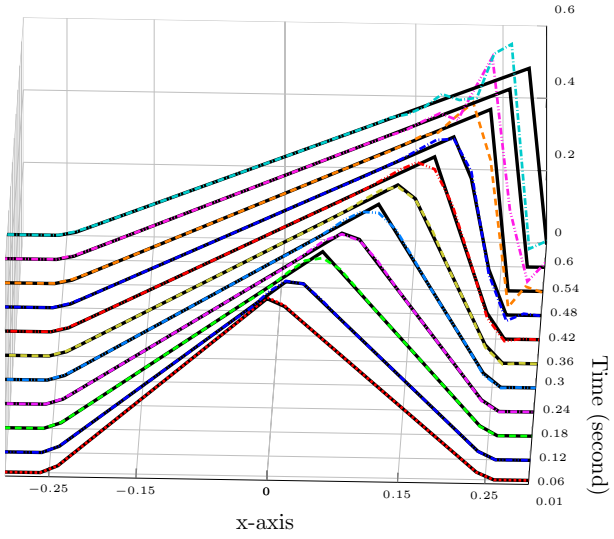
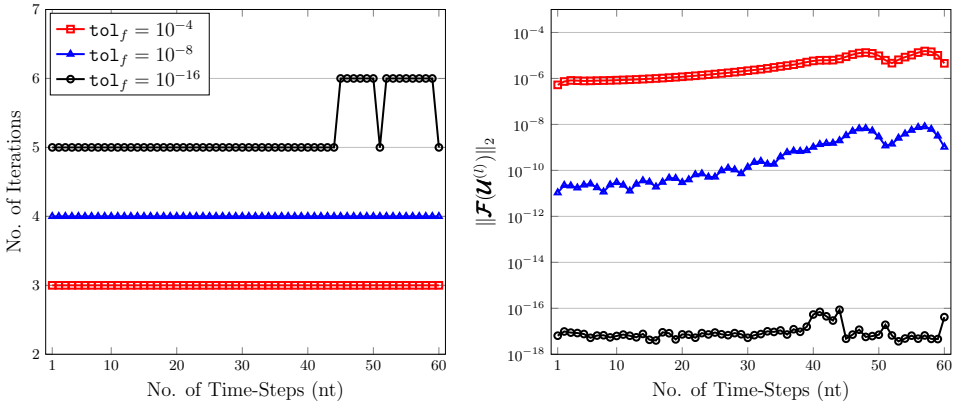


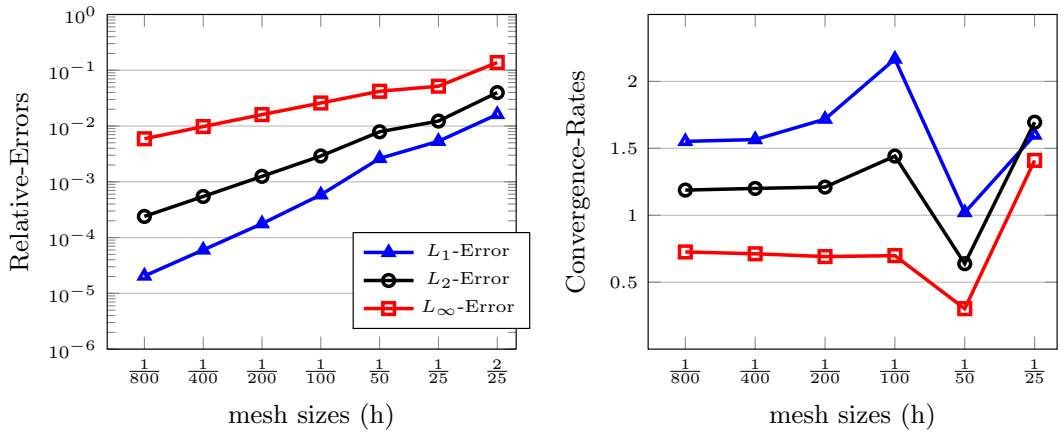
FIGURE 9. Number of iterations (left) and the corresponding reached accuracies (right) versus the number of time-steps in Algorithm 1 for different tolerance $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$ for Example 6.2 with $\delta = h = 0.02$.



discontinuities and collision of shocks. The initial state is given as

$$u_0(x) = \begin{cases} -1, & |x| \leq 1/2, \\ 0, & \text{otherwise}, \end{cases}$$

FIGURE 10. Relative L_1, L_2, and L_∞-errors (left) and the corresponding convergence rates (right) evaluated at time $t = 0.4$ for different N.



which is obviously discontinuous. The corresponding exact (weak) solution has a rarefaction wave centered at $x = 1/2$ and develops a shock first at $x = -1/2$ and $t = 0$, is given by

$$u(x, t < T_c) = \begin{cases} 0, & x < -\frac{t}{2} - \frac{1}{2}, \\ -1, & -\frac{t}{2} - \frac{1}{2} < x < -t + \frac{1}{2}, \\ \dfrac{x - \frac{1}{2}}{t}, & -t + \frac{1}{2} < x < \frac{1}{2}, \\ 0, & x > \frac{1}{2}, \end{cases}$$

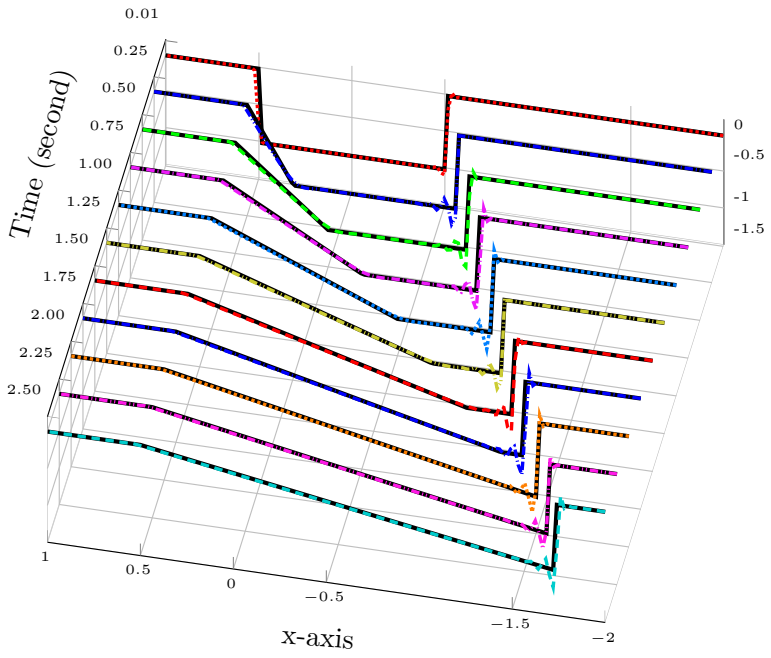
where $T_c = 2$ is the time at which the left side of the rarefaction wave collides the shock at $x = -3/2$. After this time, the second shock will be produced and therefore the solution takes the form

$$u(x, t \geq T_c) = \begin{cases} 0, & x < x_s(t) \text{ or } x > \frac{1}{2}, \\ \dfrac{x - \frac{1}{2}}{t}, & x_s(t) < x < \frac{1}{2}, \end{cases}$$

where $x_s(t) = \frac{1}{2} - \sqrt{2t}$. The computational domain is taken as $[-2, 1]$ and the final time for computations is $T = 2.5$.

Below we give the numerical experiments performed by Algorithm 1 using again with $\Delta x = 0.02$ and $\texttt{tol}_f = 10^{-4}$. The results are represented in Figure. 11 computed after $1, 25, 50, \ldots, 225$, and 250 time-steps with $\delta = h$. The numerical solutions are expressed as the dashed lines while the exact counterparts by the solid lines. The empirical results in Figure. 11 show that the SD-method exhibits a solution with small oscillations at the leading and trailing edges of the discontinuities, but not elsewhere. Furthermore, for a comparison we illustrate the SD-method associated with $\delta = 0$ after $1, 75, 175$, and 250 time-steps in Figure. 12. From this figure, it is evident that the Galerkin method is unstable for a discontinuous problem.

FIGURE 11. Numerical (dashed lines) and exact (solid lines) solutions of the Burgers equation correspond to Example 6.3 with the SD parameters $\delta = h = 0.02$ at different times $t = s\Delta t, s = 1, 25, 50 \ldots, 225, 250$.



To proceed, we investigate also the performance of Algorithm 1 for computing the approximated SD solutions for the Burgers equation subject to the initial condition in Example 6.3. We take $\delta = h = 0.02$ with tolerance ranges from $\texttt{tol}_f = 10^{-4}$ to $\texttt{tol}_f = 10^{-16}$. Since $\Delta t = 0.001$, we need to calculate the solution in $nt = 250$ time-steps. The number of iterations in each time level is plotted in Figure. 13, left. The right plot in Figure. 13 exhibits the norm $\|\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{U}}^{(l)})\|_2$ corresponds to each time level at iteration number l in which the stopping convergence criteria is fulfilled. The obtained results in the Figure. 13 indicate that convergence are obtained within 6 iterations. Analogous calculations by means of Algorithm 2 for different $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$ reveal that the number of iterations needed is significantly large rather than Algorithm 1. For $\texttt{tol}_f = 10^{-4}$, it ranges from 33 to 36 while from 78 up to 101 iterations required for the case $\texttt{tol}_f = 10^{-8}$. The corresponding values for $\texttt{tol}_f = 10^{-16}$ are 190 and 252.

We conclude our numerical section with a comparison between the SD scheme and Lax-Friedrichs finite difference method as well as Godunov scheme. For the last example, we give the results at the corresponding time levels obtained by these methods with $\Delta x = 0.02$ and for the value 0.5 of the CFL number. Figure. 14 displays the numerical solutions at times $t = 1$ and $t = 2$, at which the second shock forms. It seems that the Godunov scheme works better near the discontinuity and gives very smeared solutions, however, it is a first-order accurate procedure.

FIGURE 12. Numerical and exact solutions of the Burgers equation correspond to Example 6.3 with the SD parameters $\delta = 0$ and $h = 0.02$ at times $t = \Delta t, 75\Delta t$ (top), $t = 175\Delta t, T$ (bottom).

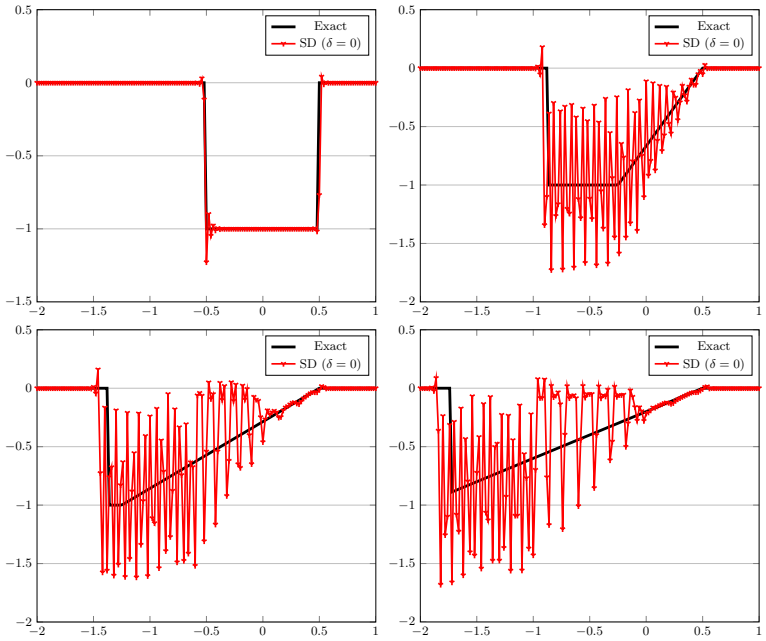


FIGURE 13. Number of iterations (left) and the corresponding reached accuracies (right) versus the number of time-steps in Algorithm 1 for different tolerance $\texttt{tol}_f = 10^{-4}, 10^{-8}, 10^{-16}$ for Example 6.3 with $\delta = h = 0.02$.

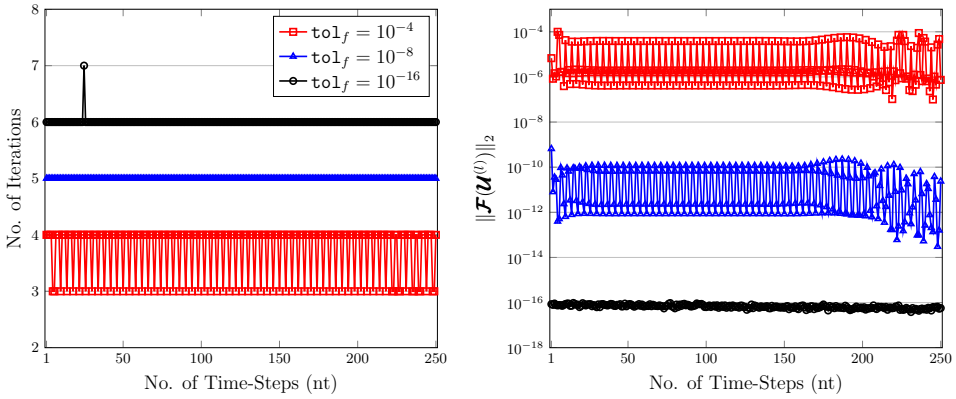
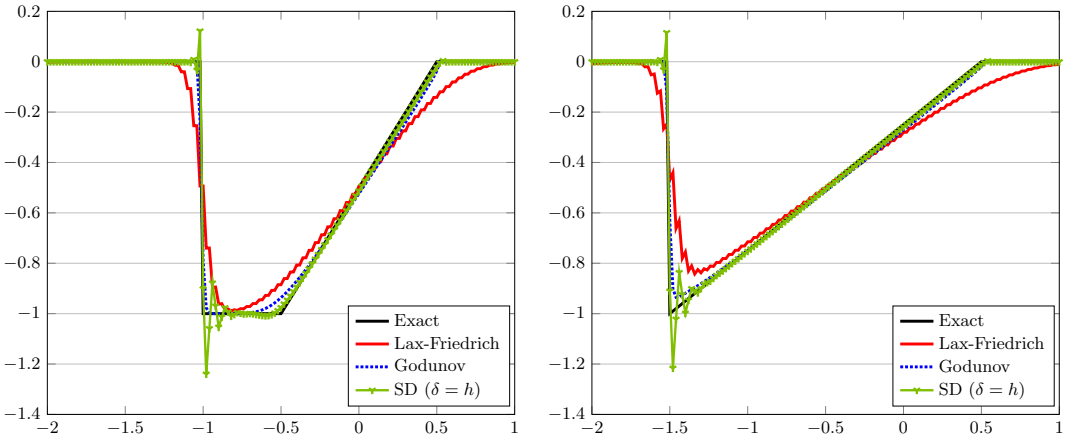


FIGURE 14. Numerical and exact solutions of the Burgers equation correspond to Example 6.3 with the Lax-Friedrichs, Godunov, and SD methods after 100 (left) and 200 (right) time-steps.



7. CONCLUSIONS

In this work, we have reviewed some fundamental aspects of the SD-method for the inviscid Burgers equation. In particular, we have investigated with a detailed description the performance of the Newton-Raphson method in connection with a space-time finite element basis functions. Based on the block-tridiagonal property of the Jacobian matrix, two different algorithms NewtonI and NewtonII are devised to iteratively solve the corresponding coupled system of nonlinear equations in each time level. While the first algorithm exploits the inversion procedure, the second one is free of inversion. Our numerical experiments show that the first Algorithm 1 is superior in terms of the number of iterations (and consequently execution time) for different mesh sizes. This means that by utilizing Algorithm 2 which is based on the Newton-Raphson scheme, a good accuracy is obtained in only few iterations. To summarize, it can be concluded that the approximated solutions obtained with the SD-method ($\delta = h$) through Algorithm 1 is almost in good agreement with the exact solutions whereas the Galerkin method ($\delta = 0$) fails to provide reasonably accurate results, particularly in the proximity of the discontinuity.

ACKNOWLEDGMENT

The author would like to express his thank to the anonymous referee whose constructive comments improved the quality of this paper.

References

[1] M. Asadzadeh and P. Kowalczyk, *Convergence of streamline diffusion methods for the Vlasov-Poisson-Fokker-Planck system*, Numer. Methods Partial Differential Equations, *21*(3) (2005), 472-495.

[2] L. Beilina and C. Johnson, *A posteriori error estimation in computational inverse scattering*, Math. Models Methods Appl. Sci, *15*(1) (2005), 23-36.

[3] N. Bicanic and K. Johnson, *Who was Raphson?*, Int. J. Numer. Methods. Eng., *14*(1) (1979), 148-152.

[4] J. M. Burgers, *A mathematical model illustrating the theory of turbulence*, Adv. Appl. Mech., (1948), 171-199.

[5] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, Amesterdam, North Holland, 1987.

[6] E. Godlewski and P. A. Raviart, *Numerical Approximation of Hyperbolic Systems of Conservation laws*, Vol. 118. Springer Science and Business Media, 1996.

[7] P. Hansbo, *The characteristic streamline diffusion method for convection-diffusion problems*, Comput. Methods Appl. Mech. Engrg, *96*(2) (1992), 239-253.

[8] H. Hartmann, *A remark on the application of the Newton-Raphson method in non-linear finite element analysis*, Comput. Mech, *36*(2) (2005), 100-116.

[9] E. Hopf, *The partial differential equation* $u_t + uu_x = \mu u_{xx}$, Comm. Pure Appl. Math, *3*(3) (1950), 201-230.

[10] T. J. R. Hughes and A. Brooks, *In: T.J.R. Hughes (eds.), Finite Element methods for Convection Dominated flows*, AMD-vol-34 (1979), 19-35.

[11] T. J. R. Hughes and A. Brooks, *Streamline upwind/Petrov-Galerkin formulation for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., *32*(1-3) (1982), 199-259.

[12] M. Izadi, Streamline diffusion methods for treating the coupling equations of two hyperbolic conservation laws, Math. Comput. Model, *45*(1-2) (2007), 201-214.

[13] M. Izadi, *A posteriori error estimates for the coupling equations of scalar conservation laws*, BIT. Numer. Math., *49*(4) (2009), 697-720.

[14] C. Johnson, *Discontinuous Galerkin finite element methods for second order hyperbolic problems*, Comput. Methods Appl. Mech. Engrg, *107*(1-2) (1993), 117-129.

[15] C. Johnson, U. Nävert, and J. Pitkäranta, *Finite element method for linear hyperbolic problems*, Comput. Methods Appl. Mech. Engrg, *45* (1984), 285-312.

[16] C. Johnson and J. Saranen, *Streamline diffusion methods for the incompressible Euler and Navier-Stokes equations*, Math. Comp., *47*(175) (1986), 1-18.

[17] C. Johnson and A. Szepessy, *On the convergence of a finite element method for a nonlinear hyperbolic conservation law*, Math. Comp., *49*(180) (1987), 427-444.

[18] C. Johnson and A. Szepessy, In: Tezduyar, T.E., Hughes, T.J.R., (eds.), *Numerical Methods for Compressible Flow-Finite Difference*, Element and Volume Techniques, AMD-vol-78, 1986.

[19] C. Johnson, A. Szepessy, and P. Hansbo, *On the convergence of shock-capturing streamline diffusion finite element methods for hyperbolic conservation laws*, Math. Comp, *54*(189) (1990), 107-129.

[20] S. N. Kruzhkov, *First order quasilinear equations in several independent variables*, USSR Math. Sbornik. *10*(2) (1970), 217-243.

[21] R. Sandboge, *Adaptive Finite Element Methods for Reactive Flow Problems*, Ph.D. Thesis, Department of Mathematics, Chalmers University of Technology, Göteborg, 1996.

[22] A. Szepessy, *Convergence of Streamline Diffusion Finite Element Method for Conservation Law*, Ph.D. Thesis, Department of Mathematics, Chalmers University of Technology, Göteborg, 1989.