Bol. Soc. Paran. Mat.
©SPM –ISSN-2175-1188 ON LINE
SPM: www.spm.uem.br/bspm

(3s.) **v. 33** 2 (2015): 111–121.
ISSN-00378712 IN PRESS
doi:10.5269/bspm.v33i2.21544

# Numerical solution of Burger's equation based on cubic B-splines quasi-interpolants and matrix arguments

Omar Chakrone, Okacha Diyer and Driss Sbibih

ABSTRACT: In this paper, we give an efficient method for solving Burger's equation. The numerical scheme equation is based on cubic B-spline quasi-interpolants and some techniques of matrix arguments. We find an iterative expression which is easy to implement and we give also the error iterative scheme. Then we compare the obtained approximate solution with that given by the methods introduced in [22] and [7].

Key Words: Burger's equation, numerical solution, B-spline, quasi- interpolation.

## Contents



## 1. Introduction

We consider the following Burger's equation

$$U_t + UU_x = \nu U_{xx}, \tag{1.1}$$

where $\nu > 0$ is the kinematic viscosity of a liquid, and the subscripts x and t denote differentiation. The initial and the boundary conditions are

$$U(x,0) = f(x),\ 0 \le x \le 1 \tag{1.2}$$

$$U(0,t) = \beta_1,\ U(1,t) = \beta_2,\ 0 \le t. \tag{1.3}$$

2000 *Mathematics Subject Classification:* 41A15, 65D07, 65D25, 65D32.

This model arises in many physical applications such as gas dynamics, heat conduction, propagation of waves in shallow water or in elastic tube filled with a viscous fluid [11]. Burger's equation is solved exactly for an arbitrary initial and boundary conditions in [2,13]. The exact solutions are impractical for the small values of viscosity constant due to slow convergence of serious solutions. Many researchers have proposed various kinds of numerical methods for solving Burger's equation for small values of viscosity constant which corresponds to steep front in the propagation of dynamic wave forms, we cite for example meshfree method which is called element-free characteristic Galerkin method [21], in general there are many methods that belong to one of the following categories: finite difference method [10,12,16,18], finite element method [1,8,17], boundary element method [5], spectral methods [9]. S. Haq, SU M. Uddin and Islam have given in [11], a method that uses radial basis functions to approximate the solution of Burger's equation. For solving Burger's equation more researchers have been attracted by this meshless scattered data approximation scheme. Hon and Wu [15], Chen and Wu [4,20] provided the methods using multiquadric (MQ) quasi-interpolation for solving differential equations. Moreover, Hon and Mao [14] developed an efficient numerical scheme for Burger's equation applying the MQ as a spatial approximation scheme and a low order explicit finite difference approximation to the time derivative. Chen and Wu [3] presented the numerical scheme for solving Burger's equation, by using the derivative of the quasi-interpolant to approximate the spatial derivative of the dependent variable and a low order forward difference to approximate the time derivative of the dependent variable. C.G. Zhu and R.H. Wang [22] have used quasi-interpolants based on B-splines but they have introduced a function to dump dispersion of scheme. In this article, we present a numerical scheme for solving Burger's equation based on some techniques using matrix arguments and a cubic B-spline quasi-interpolant. We apply the derivative of the cubic B-spline quasi-interpolant to approximate the spacial derivative of the differential equations and employ a first order accurate forward difference for the approach of the temporal derivative [3,14]. So we do not have a system where we have to invert a matrix but an iterative relationship easy to implement.

This paper is organized as follows. In Section 2, we give a brief introduction of cubic B-spline quasi-interpolation. In Section 3, we develop the numerical techniques using matrix arguments and cubic B-spline quasi-interpolation (MBSQI) to solve Burger's equation. In Section 4, we study the error analysis. In Section 5, we give the numerical examples, the results obtained by MBSQI, are compared with those using a cubic B-spline quasi-interpolation (BSQI) [22] and Dag [7]. Finally, in Section 6, we derive a conclusion.

## 2. A brief introduction of cubic B-spline quasi-interpolation

In this section, we give a construction of a cubic B-spline quasi-interpolation. Univariate spline quasi-interpolants (abbr. QIs) can be defined as operators of the form

$$Q_d f = \sum_{j \in J} \mu_j(f) B_j,$$

where $\{B_j, j \in J\}$ is the B-spline basis of some space of splines $S_d(X_n)$ of degree $d$ and $C^{d-1}$ on a bounded interval $I = [a, b]$, endowed with the uniform partition $X_n = \{x_i = a + ih, i = 0, ..., n\}$ with meshlength $h = \frac{b-a}{n}$, where $n$ is a strictly positive integer and a,b are real numbers. Let $y_i = f(x_i),\ i = 0, 1, ..., n$. A quasi-interpolant based on cubic B-splines is given by

$$Q_3(f) = \sum_{j=1}^{n+3} \mu_j(f) B_j,$$

where the coefficients $\mu_j(f)$ are defined as follows, see [22],

$\mu_1(f) = y_0,\ \mu_2(f) = \frac{1}{18}(7y_0 + 18y_1 - 9y_2 + 2y_3),$

$\mu_j(f) = \frac{1}{6}(-y_{j-3} + 8y_{j-2} - y_{j-1}),\ j = 3, ..., n+1,$

$\mu_{n+2}(f) = \frac{1}{18}(2y_{n-3} - 9y_{n-2} + 18y_{n-1} + 7y_n),$

$\mu_{n+3}(f) = y_n.$

For $f \in C^4(I)$, we have the error estimate, see [19],

$$\|f - Q_3 f\|_\infty = O(h^4).$$

Let

$$Y = (y_0, y_1, ..., y_n)^T,\ Y' = (y'_0, y'_1, ..., y'_n)^T,\ Y'' = (y''_0, y''_1, ..., y''_n)^T,$$

where

$$y'_j = (Q_3 f)'(x_j) \text{ and } y''_j = (Q_3 f)''(x_j),\ j = 0, ..., n,$$

with

$$(Q_3(f))' = \sum_{j=1}^{n+3} \mu_j(f) B'_j \text{ and } (Q_3(f))'' = \sum_{j=1}^{n+3} \mu_j(f) B''_j.$$

According to the differentiel formulas for cubic B-splines, $Y'$ and $Y''$ can be expressed in terms of Y as follows

$$Y' = \frac{1}{h} D_1 Y \text{ and } Y'' = \frac{1}{h^2} D_2 Y,$$

where $D_1$ and $D_2$ are matrices of order $n+1$. There expressions are given in [22],

$$D_1 = \begin{pmatrix} \frac{-11}{6} & 3 & \frac{-3}{2} & \frac{1}{3} & 0 & 0 & ... & 0 & 0 \\ \frac{-1}{3} & \frac{-1}{2} & 1 & \frac{-1}{6} & 0 & 0 & ... & 0 & 0 \\ \frac{1}{12} & \frac{-2}{3} & 0 & \frac{2}{3} & \frac{-1}{12} & 0 & ... & 0 & 0 \\ 0 & \frac{1}{12} & \frac{-2}{3} & 0 & \frac{2}{3} & \frac{-1}{12} & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & ... & \frac{1}{12} & \frac{-2}{3} & 0 & \frac{2}{3} & \frac{-1}{12} & 0 \\ 0 & 0 & ... & 0 & \frac{1}{12} & \frac{-2}{3} & 0 & \frac{2}{3} & \frac{-1}{12} \\ 0 & 0 & ... & 0 & 0 & \frac{1}{6} & -1 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & ... & 0 & 0 & \frac{-1}{3} & \frac{3}{2} & -3 & \frac{11}{6} \end{pmatrix},$$

$$D_2 = \begin{pmatrix} 2 & -5 & 4 & -1 & 0 & 0 & ... & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & ... & 0 & 0 \\ \frac{-1}{6} & \frac{5}{3} & -3 & \frac{5}{3} & \frac{-1}{6} & 0 & ... & 0 & 0 \\ 0 & \frac{-1}{6} & \frac{5}{3} & -3 & \frac{5}{3} & \frac{-1}{6} & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & ... & \frac{-1}{6} & \frac{5}{3} & -3 & \frac{5}{3} & \frac{-1}{6} & 0 \\ 0 & 0 & ... & 0 & \frac{-1}{6} & \frac{5}{3} & -3 & \frac{5}{3} & \frac{-1}{6} \\ 0 & 0 & ... & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & ... & 0 & 0 & -1 & 4 & -5 & 2 \end{pmatrix}.$$

## 3. Description of the method

In this section, we present the numerical scheme for solving Burger's equation based on the techniques using matrix arguments and cubic B-spline quasi-interpolant.
We have

$$U_t = -UU_x + \nu U_{xx}.$$

The solution domain $[0,1] \times [t > 0]$ is divided into mesh with the spatial step size $h = \frac{1}{n}$ in x-direction and the time step size $\tau > 0$ in t-direction respectively, where $n$ is strictly positive integer. Grid points are defined by $(x_j, t_k) = (jh, k\tau)$,

$j = 0, 1, 2, ..., n$ and $k = 0, 1, 2, ....$ So for all $j = 0, 1, ..., n$, for all non-negative integer $k$,

$$U_t(x_j, t_k) = -U(x_j, t_k)U_x(x_j, t_k) + \nu U_{xx}(x_j, t_k), \tag{3.1}$$

In order to discretize Burger's equation (1.1) in time with meshlength $\tau$, we use a first order accurate forward difference for the approach of the temporal derivative,

$$U(x_j, t_{k+1}) = U(x_j, t_k) + \tau U_t(x_j, t_k) + O(\tau^2).$$

Thus

$$\frac{U(x_j, t_{k+1}) - U(x_j, t_k)}{\tau} = U_t(x_j, t_k) + O(\tau). \tag{3.2}$$

According to (3.1) and (3.2), we have

$$U(x_j, t_{k+1}) = U(x_j, t_k) - \tau U(x_j, t_k)U_x(x_j, t_k) + \nu\tau U_{xx}(x_j, t_k) + O(\tau^2). \tag{3.3}$$

We approximate the exact solution, its first and second derivatives by using quasi-interpolant based on cubic B-splines, so we put $Q_3U(x_j, t_k) = U_j^k$, $(Q_3U)'(x_j, t_k) = (U_x)_j^k$ and $(Q_3U)''(x_j, t_k) = (U_{xx})_j^k$. We have the following estimates, see [23],

$$\max_{x \in X_n} |U(x) - Q_3U(x)| = O(h^4) \tag{3.4}$$

$$\max_{x \in X_n} |U'(x) - (Q_3U)'(x)| = O(h^3) \tag{3.5}$$

$$\max_{x \in X_n} |U''(x) - (Q_3U)''(x)| = O(h^2) \tag{3.6}$$

Using (3.3), we find the scheme

$$U_j^{k+1} = U_j^k - \tau U_j^k (U_x)_j^k + \tau\nu (U_{xx})_j^k + O(\tau^2). \tag{3.7}$$

We put respectively

$$\mathbf{U}^k = (U_0^k, U_1^k, ...U_n^k)^T,$$
$$\mathbf{U}_x^{k+1} = ((U_x)_0^k, (U_x)_1^k, ..., (U_x)_n^k),$$
$$\mathbf{U}_{xx}^{k+1} = ((U_{xx})_0^k, (U_{xx})_1^k, ..., (U_{xx})_n^k)^T,$$

so

$$\mathbf{U}_x^{k+1} = \frac{1}{h} D_1 \mathbf{U}^k, \mathbf{U}_{xx}^{k+1} = \frac{1}{h^2} D_2 \mathbf{U}^k.$$

Thus

$$\mathbf{U}^{k+1} = \mathbf{U}^k - \tau \mathbf{U}^k * \mathbf{U}_x^{k+1} + \tau\nu \mathbf{U}_{xx}^{k+1}, \tag{3.8}$$

where $*$ is defined as follows: let $M = (M_0, M_1, ..., M_n)^T$ and $N = (N_0, N_1, ..., N_n)^T$ be two vectors in $\mathbb{R}^{n+1}$, then $M * N = (M_0N_0, M_1N_1, ..., M_nN_n)^T$.
Thus

$$\mathbf{U}^{k+1} = \mathbf{U}^k - \frac{\tau}{h} \mathbf{U}^k * D_1 \mathbf{U}^k + \frac{\tau}{h^2} \nu D_2 \mathbf{U}^k.$$

Let $A_k$ be the diagonal matrix whose diagonal is formed by the coordinates of the vector $\mathbf{U}^k$.
So

$$\mathbf{U}^k * D_1\mathbf{U}^k = A_k(D_1\mathbf{U}^k) = (A_kD_1)\mathbf{U}^k.$$

Hence

$$\mathbf{U}^{k+1} = \mathbf{U}^k - \frac{\tau}{h}A_kD_1\mathbf{U}^k + \frac{\tau}{h^2}\nu D_2\mathbf{U}^k.$$

Finally, we deduce an iterative formula

$$\mathbf{U}^{k+1} = (I_{n+1} - \frac{\tau}{h}A_kD_1 + \frac{\tau}{h^2}\nu D_2)\mathbf{U}^k.$$

## 4. Error analysis

We cite the work of Ziwu Ziang et al. [23], where they gave their numerical scheme by introducing a dispersion function, and they are interested only in time approximation without considering the spatial approximation. In this section, we give the error of the iterative scheme (3.7), in which we introduce the spatial and temporal error.

**Theorem 4.1.** *For all $j = 0, 1, ..., n$, for all non-negative integer $k$, let*

$$\begin{aligned}\varphi_{j,k}(h,\tau) = &\left(U(x_j,t_{k+1}) - U_j^{k+1}\right) - \left(U(x_j,t_k) - U_j^k\right)\\ &+ \tau\left(U(x_j,t_k)U_x(x_j,t_k) - U_j^k(U_x)_j^k\right) - \nu\tau\left(U_{xx}(x_j,t_k) - (U_{xx})_j^k\right),\end{aligned}$$

*we have*

$$\varphi_{j,k}(h,\tau) = O(h^4 + \tau h^2 + \tau^2).$$

**Proof:** According to (3.4)-(3.6), we deduce that for all $j = 0, 1, ..., n$, for all non-negative integer $k$, we have the following approximations:

$$U(x_j,t_{k+1}) - U_j^{k+1} = O(h^4), \tag{4.1}$$

$$U(x_j,t_k) - U_j^k = O(h^4), \tag{4.2}$$

$$U_{xx}(x_j,t_k) - (U_{xx})_j^k = O(h^2). \tag{4.3}$$

On the other hand, we put $\psi_{j,k}(h) = U(x_j,t_k)U_x(x_j,t_k) - U_j^k(U_x)_j^k$,
then

$$\begin{aligned}\psi_{j,k}(h) =&U(x_j,t_k)U_x(x_j,t_k) - U(x_j,t_k)(U_x)_j^k + U(x_j,t_k)(U_x)_j^k - U_j^k(U_x)_j^k\\ =&U(x_j,t_k)(U_x(x_j,t_k) - (U_x)_j^k) + (U(x_j,t_k) - U_j^k)(U_x)_j^k\\ =&U(x_j,t_k)(U_x(x_j,t_k) - (U_x)_j^k) + (U(x_j,t_k) - U_j^k)((U_x)_j^k - U_x(x_j,t_k)\\ &+ U_x(x_j,t_k))\\ =&U(x_j,t_k)(U_x(x_j,t_k) - (U_x)_j^k) + (U(x_j,t_k) - U_j^k)((U_x)_j^k - U_x(x_j,t_k))\\ &+ (U(x_j,t_k) - U_j^k)U_x(x_j,t_k).\end{aligned}$$

According to the approximations (4.1)-(4.3), we have

$$\psi_{j,k}(h) = O(h^3) + O(h^7) + O(h^4) = O(h^3). \tag{4.4}$$

So, it's easy to find that

$$\varphi_{j,k}(h,\tau) = O(h^4) + O(\tau h^3) + O(\tau h^2) + O(\tau^2).$$

Finally

$$\varphi_{j,k}(h,\tau) = O(h^4 + \tau h^2 + \tau^2).$$

□

## 5. Numericals examples

### 5.1. Comparison of a numericals results

In order to compare our results with that given in [22] and [7], we take the same data:
$U(x,0) = f(x) = \sin \pi x,\ \forall\ 0 \leq x \leq 1,$
$U(0,t) = U(1,t) = 0\ \forall\ t > 0.$
We denote the present scheme by MBSQI, that given in [22] by BSQI and the scheme in [7] by Dag. We note $re = \frac{\mid \text{numerical solution - exact solution} \mid}{\mid \text{exact solution} \mid}$, the relative error.
The exact solution was determined in terms of the infinite serie by Cole in [6] as

$$U(x,t) = 2\pi\nu \frac{\sum_{j=1}^{\infty} j a_j \sin(j\pi x) exp(-j^2\pi^2\nu t)}{a_0 + 2\sum_{j=1}^{\infty} a_j \cos(j\pi x) exp(-j^2\pi^2\nu t)},$$

where

$$a_j = \int_0^1 exp[-(2\pi\nu)^{-1}(1-\cos(\pi x))]\cos(j\pi x)dx, \text{ for all } j = 0\ldots .$$

Firstly, we compare the numerical results using MBSQI with exact solution and BSQI with $\nu = 1, \tau = 0.00001$ and various meshlength h at time $t = 0.1$, see Tables 1-3. Moreover a comparison of MBSQI with exact solutions, BSQI and Dag's method [7] when $t = 0.1$ for $\nu = 1$, $\tau = 0.00001$ and $h = 0.0125$ are given in Table 4. We will be interested also to the behavior of the relative error with respect to the decreasing of mesh length.

According to Tables 1-3, we remark that the numerical solution obtained by MBSQI provides better accuracy than the method given by BSQI, with various meshlength $h = 0.1, 0.05, 0.025$. In table 4, we find that the numerical solution obtained by MBSQI provides also better accuracy than those given by BSQI and Dag, for the meshlength $h = 0.0125$. On the other hand, when $h \to 0$, there is a significant improvement of the relative error.

Table 1: Comparison of results at $t = 0.1$ for $\nu = 1$, $\tau = 0.00001$ and $h = 0.1$.

| $x$ | BSQI | MBSQI | Exact | re$*10^2$(BSQI) | re$*10^2$(MBSQI) |
|---|---|---|---|---|---|
| 0.10 | 0.10831 | 0.1089868754 | 0.10954 | 1.122877488 | 0.5049521636 |
| 0.20 | 0.20724 | 0.2086239100 | 0.20979 | 1.215501215 | 0.5558367891 |
| 0.30 | 0.28799 | 0.2901321890 | 0.29190 | 1.339499829 | 0.6056221309 |
| 0.40 | 0.34273 | 0.3456451643 | 0.34792 | 1.491722235 | 0.6538387274 |
| 0.50 | 0.36531 | 0.3689505210 | 0.37158 | 1.687388987 | 0.7076481510 |
| 0.60 | 0.35223 | 0.3563399372 | 0.35905 | 1.899456900 | 0.7547870214 |
| 0.70 | 0.30400 | 0.3074800502 | 0.30991 | 1.907005260 | 0.7840824110 |
| 0.80 | 0.22358 | 0.2260611718 | 0.22782 | 1.861118427 | 0.7720253709 |
| 0.90 | 0.11860 | 0.1198889655 | 0.12069 | 1.731709338 | 0.6637124037 |

Table 2: Comparison of results at $t = 0.1$ for $\nu = 1$, $\tau = 0.00001$ and $h = 0.05$.

| $x$ | BSQI | MBSQI | Exact | re$*10^3$(BSQI) | re$*10^3$(MBSQI) |
|---|---|---|---|---|---|
| 0.10 | 0.10920 | 0.1093742235 | 0.10954 | 3.103888990 | 1.513387804 |
| 0.20 | 0.20912 | 0.2094618772 | 0.20979 | 3.193669860 | 1.564053577 |
| 0.30 | 0.29088 | 0.2914044197 | 0.29190 | 3.494347379 | 1.697774238 |
| 0.40 | 0.34658 | 0.3472874457 | 0.34792 | 3.851460106 | 1.818102725 |
| 0.50 | 0.36997 | 0.3708352747 | 0.37158 | 4.332848915 | 2.004212551 |
| 0.60 | 0.35740 | 0.3582647982 | 0.35905 | 4.595460243 | 2.186887063 |
| 0.70 | 0.30847 | 0.3091798822 | 0.30991 | 4.646510277 | 2.355902681 |
| 0.80 | 0.22676 | 0.2272558006 | 0.22782 | 4.652796067 | 2.476513915 |
| 0.90 | 0.12012 | 0.1203848204 | 0.12069 | 4.722843649 | 2.528623747 |

Table 3: Comparison of results at $t = 0.1$ for $\nu = 1$, $\tau = 0.00001$ and $h = 0.025$.

| $x$ | BSQI | MBSQI | Exact | re$*10^4$(BSQI) | re$*10^4$(MBSQI) |
|---|---|---|---|---|---|
| 0.10 | 0.10947 | 0.1094923708 | 0.10954 | 6.390359686 | 4.348110280 |
| 0.20 | 0.20965 | 0.2097008148 | 0.20979 | 6.673340006 | 4.251165451 |
| 0.30 | 0.29168 | 0.2917611570 | 0.29190 | 7.536827681 | 4.756526208 |
| 0.40 | 0.34764 | 0.3477498979 | 0.34792 | 8.047827088 | 4.889115315 |
| 0.50 | 0.37125 | 0.3713753480 | 0.37158 | 8.880994670 | 5.507616125 |
| 0.60 | 0.35871 | 0.3588333942 | 0.35905 | 9.469433228 | 6.032747529 |
| 0.70 | 0.30961 | 0.3097078378 | 0.30991 | 9.680229744 | 6.523255139 |
| 0.80 | 0.22759 | 0.2276638962 | 0.22782 | 10.09568958 | 6.852067422 |
| 0.90 | 0.12057 | 0.1206025191 | 0.12069 | 9.942828734 | 7.248396719 |

Table 4: Comparison of results at $t = 0.1$ for $\nu = 1$, $\tau = 0.00001$ and $h = 0.0125$.

| $x$ | BSQI | Dag | MBSQI | Exact | re$*10^4$(BSQI) | re$*10^4$(Dag) | re$*10^4$(MBSQI) |
|---|---|---|---|---|---|---|---|
| 0.10 | 0.10951 | 0.10952 | 0.1095226593 | 0.10954 | 2.738725580 | 1.825817053 | 1.583047289 |
| 0.20 | 0.20974 | 0.20975 | 0.2097615923 | 0.20979 | 2.383335716 | 1.906668573 | 1.354101721 |
| 0.30 | 0.29128 | 0.29184 | 0.2918518334 | 0.29190 | 2.124015074 | 2.055498458 | 1.650106201 |
| 0.40 | 0.34783 | 0.34785 | 0.3478676576 | 0.34792 | 2.586801564 | 2.011956772 | 1.504437802 |
| 0.50 | 0.37147 | 0.37149 | 0.3715133440 | 0.37158 | 2.960331557 | 2.422089455 | 1.793853275 |
| 0.60 | 0.35894 | 0.35896 | 0.3589793938 | 0.35905 | 3.063640162 | 2.506614678 | 1.966472636 |
| 0.70 | 0.30981 | 0.30983 | 0.3098443508 | 0.30991 | 3.226743248 | 2.581394598 | 2.118331128 |
| 0.80 | 0.22775 | 0.22776 | 0.2277706409 | 0.22782 | 3.072601176 | 2.633658151 | 2.166583268 |
| 0.90 | 0.12065 | 0.12065 | 0.1206611482 | 0.12069 | 3.314276245 | 3.314276245 | 2.390570884 |

### 5.2. Examples of passages states of the numerical solution to obtain the final solution

To move from the initial time $t = 0$ to the final time $t = 0, 1$, the approximate solution passes through intermediate states. The Figures 1 and 2, show the passages states of the numerical solution to obtain the final result at $t = 0.1$, with $h = 0.1$ for $\mu = 1, 0.1$ respectively.

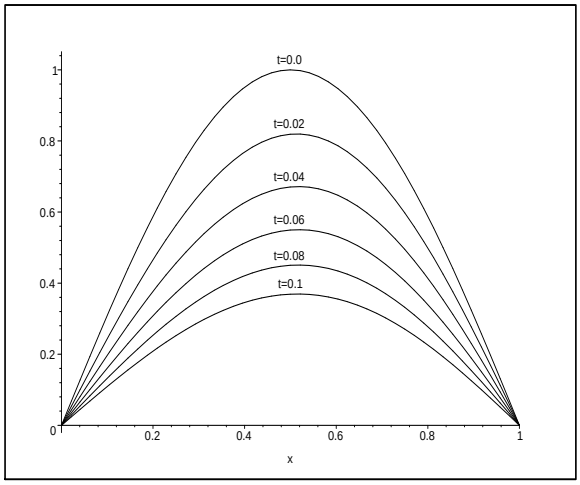


Figure 1: The passages states of the numerical solution at t=0.1, with h=0.1, $\tau = 0.00001$ and $\nu = 1$.

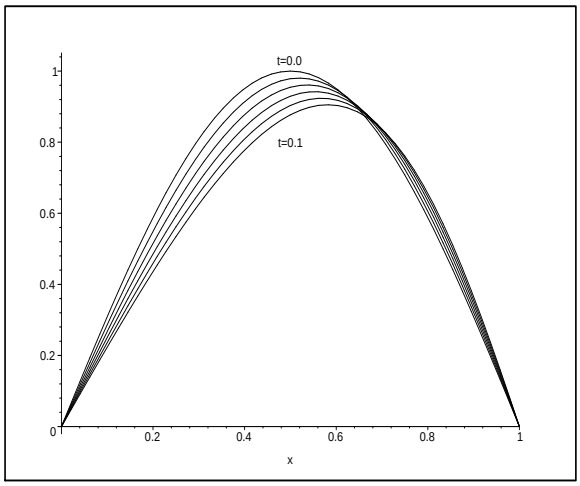


Figure 2: The passages states of the numerical solution at t=0.1, with h=0.1, $\tau = 0.00001$ and $\nu = 0.1$.

We note that the approximate solution to arrive at time $t = 0.1$, passes through of the steps which corresponds to the instants $t = 0.002, t = 0.004, t = 0.06, t = 0.08$, in a manner dependent on the viscosity of the liquid, see Figures 1 and 2.

## 6. Conclusion

The present technique (MBSQI) of discretization for solving numerically Burger's equation is based on matrix arguments and cubic B-spline quasi- interpolants. We have deduce that the MBSQI scheme gives the accuracy better than those given by BSQI and Dag. Furthermore we have find an iterative expression which is easy to implement. We also gave the corresponding error iterative scheme to our discritization.

## References

1. EN. Aksan, A numerical solution of Burger's equation by finite element method constructed on the method of discretization in time. *Applied Mathematical Computer*. (2005), 895-904.

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

3. R. Chen and Z. Wu, Applying multiquadric quasi-interpolation to solve Burgers' equation. *Applied Mathematical and Computation*. 172 (2006), 472-484.

4. R. Chen, Z. Wu, Solving partial differential equation by using multiquadric quasi-interpolation. *Appl. Math. Comput*. (2007), 1502-1510.

5. E. Chino, N. Tosaka, Dual reciprocity boundary element analysis of time-independent Burger's equation. *Eng Anal Boundary Elem*. (1998), 261-270.

6. J.D. Cole, On a quasi-linear parabolic equation occuring in aerodynamics. *Quart Math.* (1951), 225-236.

7. İ. Dag, D. Irk, B. Saka, A numerical solution of the Burgers'equation using cubic B-splines. *Appl. Math. Comput.* (2005) 199-211.

8. A. Dogan, A Galerkin finite element approach to Burger's equation. *Applied Mathematical Computer.* (2004), 331-346.

9. HM. El-Hawary, EO. Abdel-Rahman, Numerical solution of yhe generalized Burger's equation via spectral/spline methods. *Applied Mathematical Computer.* (2005), 267-279.

10. M. Gülsu, A finite difference approach for solution of Burger's equation . *Applied Mathematical Computer.* (2006), 1245-1255.

11. S. Haq, S.U. Islam and M. Uddin, A mesh-free method for the numerical solution of the Kdv-Burgers equation. *Applied Mathematical Modelling.* (2009), 3442-3449.

12. IA. Hassanien, AA. Salama, HA. Hosham, Fourth-order finite difference method for solving Burger's equation. *Applied Mathematical Computer.* (2005), 781-800.

13. B.M. Herbst, S.W. Schoombie, A.R. Mitchell, A moving Petrov-Galerkin method for transport equations. *Int. J. Numer. Methods Eng.* (1982), 1321-1336.

14. Y.C. Hon, X.Z. Mao, An efficient numerical scheme for Burgers' equation. *Appl. Math. Comput.* (1998), 37-50.

15. Y.C. Hon, Z. Wu, A quasi-interpolation method for solving ordinary differential equations. *Int. J. Numer. Methods Eng.* (2000), 1187-1197.

16. MK. Kadalbajoo, A. Awasthi, A numerical method based on Crank-Nicolson scheme for Burger's equation. *Applied Mathematical Computer.* (2006), 1430-1442.

17. T. Öziş, EN. Aksan, A. Özdeş, A finite element approach for solution of Burger's equation. *Applied Mathematical Computer.* (2003), 417-428.

18. SF. Radwan, Comparison of heigher-order accurate shemes for solving the two-dimensional unsteady Burger's equation. *Applied Mathematical Computer.* (2005), 383-397.

19. P. Sablonnière, univariate spline quasi-interpolants and applications to numerical analysis. *Rend. Sem. Mat. Univ. Pol. Torino.* (2005), 211-223.

20. Z. Wu, Dynamically knots setting in meshless method for solving time dependent propagations equation. *Comput. Methods Appl. Mech. Eng.* (2004), 1221-1229.

21. X.H. Zhang, J. Ouyang, L. Zhang, Element-free characteristic Galerkin method for Burger's equation. *engineering Analysis with Boundary Elements.* (2009), 356-362.

22. C.G. Zhu and R.H. Wang, Numerical solution of Burgers'equation by cubic B-spline quasi-interpolation. *Applied Mathematical and Computation.* 208 (2009), 260-272.

23. Z. Ziang, R. Wang, An improved numerical solution of Burgers'equation by cubic B-spline quasi-interpolation. *Journal of Information and Computational Science* . 5 (2010), 1013-1021.

*Omar Chakrone*
*Université Mohammed I, Faculté des Sciences*
*Laboratoire LANOL, Oujda, Maroc.*
*E-mail address:* chakrone@yahoo.fr

*and*

*Okacha Diyer and Driss Sbibih*
*Université Mohammed I, Ecole Supérieure de Technologie*
*Laboratoire MATSI, Oujda, Maroc.*
*E-mail address:* odiyer@yahoo.fr
*E-mail address:* sbibih@yahoo.fr