Research Article

A Meshfree Quasi-Interpolation Method for Solving Burgers’ Equation

Mingzhu Li,¹,² Lijuan Chen,¹ and Qiang Ma²

¹ School of Science, Qingdao Technological University, Qingdao 266033, China
² Department of Mathematics, Harbin Institute of Technology at Weihai, Weihai 264209, China

Correspondence should be addressed to Qiang Ma; hitmaqiang@hotmail.com

Received 26 April 2014; Revised 29 June 2014; Accepted 1 July 2014; Published 16 July 2014

Academic Editor: Kim M. Liew

Copyright © 2014 Mingzhu Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The main aim of this work is to consider a meshfree algorithm for solving Burgers’ equation with the quartic B-spline quasi-interpolation. Quasi-interpolation is very useful in the study of approximation theory and its applications, since it can yield solutions directly without the need to solve any linear system of equations and overcome the ill-conditioning problem resulting from using the B-spline as a global interpolant. The numerical scheme is presented, by using the derivative of the quasi-interpolation to approximate the spatial derivative of the dependent variable and a low order forward difference to approximate the time derivative of the dependent variable. Compared to other numerical methods, the main advantages of our scheme are higher accuracy and lower computational complexity. Meanwhile, the algorithm is very simple and easy to implement and the numerical experiments show that it is feasible and valid.

1. Introduction

Burgers’ equation plays a significant role in various fields, such as turbulence problems, heat conduction, shock waves, continuous stochastic processes, number theory, gas dynamics, and propagation of elastic waves [1–5]. The one-dimensional Burgers’ equation first suggested by Bateman [6] and later treated by Burgers [1] has the form

\[ U_t + U U_x - \lambda U_{xx} = 0, \tag{1} \]

where \( \lambda > 0 \) is the coefficient of kinematic viscosity and the subscripts \( x \) and \( t \) denote space and time derivatives. Initial and boundary conditions are

\[ U(x,0) = f(x), \quad a \leq x \leq b, \]

\[ U(a,t) = \beta_1, \quad U(b,t) = \beta_2, \quad t \geq 0, \tag{2} \]

where \( \beta_1, \beta_2, \) and \( f(x) \) will be chosen in a later section.

Burgers’ equation is a quasi-linear parabolic partial differential equation, whose analytic solutions can be constructed from a linear partial differential equation by using Hopf-Cole transformation [1, 2, 7]. But some analytic solutions consist of infinite series, converging very slowly for small viscosity coefficient \( \lambda \). Thus, many researchers have spent a great deal of effort to compute the solution of Burgers’ equation using various numerical methods. Finite difference methods were presented to solve the numerical solution of Burgers’ equation in [8–11]. Finite element methods for the solution of Burgers’ equation were introduced in [12–15]. Recently, various powerful mathematical methods such as Galerkin finite element method [16, 17], spectral collocation method [18, 19], sinc differential quadrature method [20], factorized diagonal padé approximation [21], B-spline collocation method [22], and reproducing kernel function method [23] have also been used in attempting to solve the equation.

In 1968 Hardy proposed the multiquadric (MQ) which is a kind of radial basis function (RBF). In Franke’s review paper, the MQ was rated as one of the best methods among 29 scattered data interpolation and ease of implementation. Since Kansa successfully applied MQ for solving partial differential equation, more and more reasearchers have been attracted...
by this meshfree, scattered data approximation scheme [24]. The meshfree method uses a set of scattered nodes, instead of meshing the domain of the problem. It has been successfully applied to solve many physical and engineering problems with only a minimum of meshing or no meshing at all [25–30]. In recent years, many meshfree methods have been developed, such as the element-free Galerkin method [31], the smooth particle hydrodynamics method [32], the element-free kp-Ritz method [33–36], the meshless local Petrov-Galerkin method [37], and the reproducing kernel particle method [38].

With the use of univariate multiquadric (MQ) quasi-interpolation, solution of Burgers’ equations was obtained by Chen and Wu [39]. Moreover, Hon and Mao [40] developed interpolation, solution of Burgers’ equations was obtained by the derivative of the element-free Galerkin method [37], and the reproducing kernel particle method [38]. In [24], univariate quartic B-spline quasi-interpolants have been presented and compared with those obtained with some previous results. At last, we conclude the paper in Section 5.

### 2. Univariate Quartic B-Spline Quasi-Interpolant

For an interval I = [a, b], we introduce a set of equally-spaced knots of partition Ω = {x0, x1, ..., xn}. We assume that n ≥ 5, xi = a + ih (i = 0, 1, ..., n), x0 = a, and xn = b. Let S4[π] be the space of continuously-differentiable, piecewise, quartic-degree polynomials on π. A detailed description of B-spline functions generated by subdivision regarding the B-splines basis in S4[π] can be found in [45].

The zero degree B-spline is defined as

\[ N_{i,0}(x) = \begin{cases} 1, & x \in [x_i, x_{i+1}] \\ 0, & \text{otherwise} \end{cases} \]

and, for positive constant p, it is defined in the following recursive form:

\[ N_{i,p}(x) = \begin{cases} \frac{x-x_i}{x_{i+p}-x_i} N_{i+p-1}(x) + \frac{x_{i+p+1}-x}{x_{i+p+1}-x_{i+1}} N_{i+1,p-1}, & \text{if } p \geq 1. \end{cases} \]

We apply this recursion to get the quartic B-spline \( N_{i,4}(x) \), which is defined in \( S_4[\pi] \) as follows:

\[ N_{i,4}(x) = \begin{cases} 
\left(\begin{array}{l}
(x-x_i)^4, \\
(x-x_i)^4 - 5(x-x_i)^4, \\
(x-x_i)^4 - 5(x-x_i)^4 + 10(x-x_i)^4, \\
(x-x_i)^4 - 5(x-x_i)^4, \\
0,
\end{array}\right)
\end{cases} \]

As usual, we add multiple knots at the endpoints: a = x_{-4} = x_{-3} = \cdots = x_0 and b = x_n = x_{n+1} = \cdots = x_{n+4}.

In [24], univariate quartic B-spline quasi-interpolants (abbr. QI s) can be defined as operators of the form

\[ Q_4(f) = \sum_{j=1}^{n+4} \mu_j N_{j,4}(x) \]

The coefficients are listed as follows:

\[ \mu_1(f) = f_1, \]
\[ \mu_2(f) = \frac{17}{105} f_1 + \frac{35}{32} f_2 - \frac{35}{96} f_3 + \frac{21}{160} f_4 - \frac{5}{244} f_5, \]
\[ \mu_3(f) = -\frac{19}{45} f_1 + \frac{377}{288} f_2 + \frac{61}{288} f_3 - \frac{59}{480} f_4 + \frac{7}{288} f_5, \]
\[ \mu_4(f) = \frac{47}{315} f_1 - \frac{77}{144} f_2 + \frac{251}{144} f_3 - \frac{97}{240} f_4 + \frac{47}{1008} f_5, \]
\[ \mu_5(f) = \frac{47}{1152} (f_{j-4} + f_{j-1}) - \frac{107}{288} (f_{j-3} + f_{j-1}) + \frac{319}{192} f_{j-2}, \quad j = 5, \ldots, n, \]
\[ \mu_{n+1}(f) = \frac{47}{315} f_{n+2} - \frac{77}{144} f_{n+1} + \frac{251}{144} f_n - \frac{97}{240} f_{n-1} + \frac{47}{1008} f_{n-2}, \]

Section 3, we mainly propose the numerical techniques using quartic B-spline interpolation to solve Burgers’ equation. In Section 4, numerical examples of Burgers’ equation are presented and compared with those obtained with some previous results.
\[ \mu_{n+2}(f) = -\frac{19}{45} f_{n+2} + \frac{377}{288} f_{n+1} + \frac{61}{288} f_n - \frac{59}{480} f_{n-1} + \frac{7}{288} f_{n-2}, \]
\[ \mu_{n+3}(f) = \frac{17}{105} f_{n+2} + \frac{35}{324} f_{n+1} - \frac{35}{96} f_n + \frac{21}{160} f_{n-1} - \frac{5}{244} f_{n-2}, \]
\[ \mu_{n+4}(f) = f_{n+2}, \]  
(7)

and \( f_i = f(t_i), t_i = (1/2)(x_{i-2} + x_{i+1}), i = 1, \ldots, n + 2. \) For \( f \in C^3(I) \), we have the error estimate
\[ \|f - Q_4(f)\|_{\infty} = O(h^5). \]  
(8)

We use \( \prod_k \) to denote the space of polynomials of the total degree at most 4. In general, we impose that \( Q_4 \) is exact on the space \( \prod_4 \); that is, \( Q_4(p) = p \) for all \( p \in \prod_4 \). As a consequence of this property, the approximation order of \( Q_4 \) is \( O(h^5) \) on smooth functions. In this paper, the coefficient \( \mu_j \) is a linear combination of discrete values of \( f \) at some points. The main advantage of \( QIs \) is that they have a direct construction without solving any system of linear equations. Moreover, they are local in the sense that values of \( Q_4f(x) \) depend only on values of \( f \) in a neighborhood of \( x \). Finally, they have a rather small infinity norm and, therefore, are nearly optimal approximant.

Differentiating interpolation polynomials leads to the classic finite difference for the approximate computation of derivatives. Therefore, we can draw a conclusion of approximating derivatives of \( f \) by derivatives of \( Q_4f \). The general theory will be developed elsewhere. We can evaluate the value of \( f \) at \( x_i \) by \( (Q_4f)'' = \sum_{j=1}^{m+4} \mu_j(f)N_{j,4}' \) and \( (Q_4f)''' = \sum_{j=1}^{m+4} \mu_j(f)N_{j,4}'' \). \( N_{j,4}' \) and \( N_{j,4}'' \) can be computed by the formula of B-spline’s derivatives as follows:
\[ N_{j,4}^{(k)} = \frac{4!}{(4-k)!} \sum_{j=1}^{n} \alpha_{k,j} N_{i,j+k}, \]  
(9)

where
\[ \alpha_{0,0} = 1, \]
\[ \alpha_{k,0} = \frac{\alpha_{k-1,0}}{x_{i+3-k} - x_i}, \]
\[ \alpha_{k,k} = \frac{-\alpha_{k-1,k}}{x_{i+5-k} - x_{i+k}}, \]
\[ \alpha_{k,j} = \frac{\alpha_{k-1,j} - \alpha_{k-1,j-1}}{x_{i+j-5+k} - x_{i+j}}. \]  
(10)

By some trivial computations, we can obtain the value of \( N_{j,4}^{(k)} \) \((k = 0, 1, 2, 3)\) at the knots, which are illustrated in Table 1. Then, we get the differential formulas for quartic B-spline QIs as
\[ f' = \sum_{j=1}^{m+4} \mu_j(f) N_{j,4}', \]  
(11)
\[ f'' = \sum_{j=1}^{m+4} \mu_j(f) N_{j,4}''. \]  
(12)

3. Numerical Scheme Using the Meshfree Quasi-Interpolation

In this section, we present the numerical scheme for solving Burgers’ equation based on the quartic B-spline quasi-interpolation.

Discretizing the Burgers’ equation
\[ U_t + UU_x - \lambda U_{xx} = 0, \]  
(12)
in time with meshlength \( \tau \), we get
\[ \frac{U_j^{k+1} - U_j^k}{\tau} + U_j^k(U_x)_j^k - \lambda (U_{xx})_j^k = 0. \]  
(13)

We can get
\[ U_j^{k+1} = U_j^k + \tau U_j^k(U_x)_j^k - \tau \lambda (U_{xx})_j^k, \]  
(14)
where \( U_j^k \) is the approximation of the value of \( U(x,t) \) at the point \((x_j,t_k)\). Then, we can use the derivatives of the quartic B-spline quasi-interpolant \( Q_4U(x_j,t_k) \) to approximate \((U_x)_j^k \) and \((U_{xx})_j^k \). To damp the dispersion of the scheme, we define a switch function \( g(x,t) \), whose values are 0 and 1 at the discrete points \((x_j,t_k)\), as follows:
\[ g(x_j,t_k) = \max \left\{0, 1 + \min \left\{0, \frac{\min \left\{(U_x)_j^k, (U_x)_j^k\right\}}{2 \tau \lambda (U_{xx})_j^k} \right\} \right\}, \]  
(15)

where \( l = j - \text{sign}(U_j^k) \). Thus, the resulting numerical scheme is
\[ U_j^{k+1} = U_j^k + \tau U_j^k(U_x)_j^k g(x_j,t_k) - \tau \lambda (U_{xx})_j^k, \]  
(16)

Starting from the initial condition, we can compute the numerical solution of Burgers’ equation step by step using the B-spline quasi-interpolation scheme (16) and formulas (11).
4. Numerical Results

To investigate the applicability of the quasi-interpolation method to Burgers’ equation, four selected example problems are studied. To show the efficiency of the present method for our problem in comparison with the exact solution, we use the following norms to assess the performance of our scheme:

\[
L_\infty = \max_j |U_j^{\text{exact}} - U_j^{\text{num}}|, \\
L_2 = \sqrt{\frac{h}{n} \sum_{j=1}^{n} (U_j^{\text{exact}} - U_j^{\text{num}})^2}.
\]

(17)

Example 1. Burgers’ equation is solved over the region \([0, 1]\) and the initial and boundary conditions are given in Asaithambi [42]:

\[
U(x, 0) = \frac{2\lambda \pi \sin \pi x}{\alpha \cos \pi x} \quad (\alpha > 1),
\]

(18)

\[
U(0, t) = 0, \quad U(1, t) = 0, \quad t > 0,
\]

and the exact solution of this problem has the following nice compact closed-form, as given by Wood [46]:

\[
U(x, t) = \frac{2\lambda \pi e^{-\pi^2 t_\lambda}}{\alpha + e^{-\pi^2 t_\lambda}} \sin \pi x \quad (\alpha > 1).
\]

(19)

In this computational study, we set \(\alpha = 2, \ h = 0.025, \ \Delta t = 0.0001\). The comparison of the numerical solutions obtained by the present method, at the different coefficient of kinematic viscosity \(\lambda\), are presented with the solutions obtained by Asaithambi [42] and the exact solution in Table 2.

Example 2. In this example, we consider the exact solution of Burgers’ equation [47]:

\[
U(x, t) = \frac{\alpha + \mu + (\mu - \alpha) \exp \eta}{1 + \exp \eta}, \quad 0 \leq x \leq 1, \ t \geq 0,
\]

(20)

where \(\eta = (\alpha(x - \mu t - \gamma))/\lambda, \ \alpha, \ \mu, \ \text{and} \ \gamma \ \text{are constants. The boundary conditions are}

\[
U(0, t) = 1, \quad U(1, t) = 0.2, \quad t \geq 0,
\]

(21)

and initial condition is used for the exact solution at \(t = 0\).

We solve the problem with \(\alpha = 0.4, \ \mu = 0.6, \ \text{and} \ \gamma = 0.125\) by our method. In Table 3, \(L_2\) and \(L_\infty\) errors at the time level \(t = 0.5\) are compared with the error obtained by Chen and Wu [39], Zhu and Wang [41], Dağ et al. [17], and Saka and Dağ [43]. For comparison, the parameters are adopted as time step \(\tau = 0.01\), space step \(h = 1/36\), and viscosity coefficient \(\lambda = 0.01\). From Table 3, we can find that our method provides better accuracy than most methods through the \(L_2\) and \(L_\infty\) error norms. The profiles of initial wave and its propagation are depicted at some times in Figure 1.

### Table 2: Comparison of exact and numerical solution at \(t = 0.001\).

<table>
<thead>
<tr>
<th>(x)</th>
<th>(\lambda = 1) Our method</th>
<th>(\lambda = 1) Exact</th>
<th>(\lambda = 0.5) Asai [42]</th>
<th>(\lambda = 0.5) Exact</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.653563</td>
<td>0.653589</td>
<td>0.327870</td>
<td>0.327870</td>
</tr>
<tr>
<td>0.2</td>
<td>1.306559</td>
<td>1.305534</td>
<td>0.65028</td>
<td>0.650278</td>
</tr>
<tr>
<td>0.3</td>
<td>1.949321</td>
<td>1.949364</td>
<td>0.978449</td>
<td>0.978427</td>
</tr>
<tr>
<td>0.4</td>
<td>2.565877</td>
<td>2.565925</td>
<td>1.288417</td>
<td>1.288463</td>
</tr>
<tr>
<td>0.5</td>
<td>3.110769</td>
<td>3.110992</td>
<td>1.563014</td>
<td>1.563096</td>
</tr>
<tr>
<td>0.6</td>
<td>3.492902</td>
<td>3.493222</td>
<td>1.756653</td>
<td>1.756695</td>
</tr>
<tr>
<td>0.7</td>
<td>3.549538</td>
<td>3.550134</td>
<td>1.787184</td>
<td>1.787206</td>
</tr>
<tr>
<td>0.8</td>
<td>3.050059</td>
<td>3.050702</td>
<td>1.537658</td>
<td>1.537694</td>
</tr>
<tr>
<td>0.9</td>
<td>1.816492</td>
<td>1.817077</td>
<td>0.916795</td>
<td>0.916860</td>
</tr>
</tbody>
</table>

### Table 3: The computational results at \(t = 0.5\) for \(\lambda = 0.01\) with \(h = 1/36\) and \(\tau = 0.01\).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(L_\infty \times 10^3)</td>
<td>3.43253</td>
<td>5.77786</td>
<td>0.77033</td>
<td>1.92558</td>
</tr>
<tr>
<td>(L_2 \times 10^3)</td>
<td>9.26698</td>
<td>20.8467</td>
<td>3.05179</td>
<td>6.35489</td>
</tr>
</tbody>
</table>

![Figure 1](image-url)
Table 4: Comparison of results at different time for $\lambda = 0.1$ with $h = 0.025$ and $\tau = 0.0001$.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>0.3089</td>
<td>0.3083</td>
<td>0.3143</td>
<td>0.3089</td>
<td>0.3089</td>
</tr>
<tr>
<td>0.6</td>
<td>0.2407</td>
<td>0.2404</td>
<td>0.2437</td>
<td>0.2407</td>
<td>0.2407</td>
</tr>
<tr>
<td>0.8</td>
<td>0.1957</td>
<td>0.1954</td>
<td>0.1976</td>
<td>0.1957</td>
<td>0.1957</td>
</tr>
<tr>
<td>1.0</td>
<td>0.1626</td>
<td>0.1624</td>
<td>0.1639</td>
<td>0.1626</td>
<td>0.1626</td>
</tr>
<tr>
<td>3.0</td>
<td>0.0272</td>
<td>0.0272</td>
<td>0.0274</td>
<td>0.0272</td>
<td>0.0272</td>
</tr>
<tr>
<td>0.4</td>
<td>0.5696</td>
<td>0.5691</td>
<td>0.5764</td>
<td>0.5696</td>
<td>0.5696</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4472</td>
<td>0.4468</td>
<td>0.4517</td>
<td>0.4472</td>
<td>0.4472</td>
</tr>
<tr>
<td>0.8</td>
<td>0.3592</td>
<td>0.3589</td>
<td>0.3625</td>
<td>0.3592</td>
<td>0.3592</td>
</tr>
<tr>
<td>1.0</td>
<td>0.2919</td>
<td>0.2916</td>
<td>0.2944</td>
<td>0.2919</td>
<td>0.2919</td>
</tr>
<tr>
<td>3.0</td>
<td>0.0402</td>
<td>0.0402</td>
<td>0.0406</td>
<td>0.0402</td>
<td>0.0402</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6254</td>
<td>0.6256</td>
<td>0.6259</td>
<td>0.6257</td>
<td>0.6254</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4872</td>
<td>0.4570</td>
<td>0.4903</td>
<td>0.4872</td>
<td>0.4872</td>
</tr>
<tr>
<td>0.8</td>
<td>0.3739</td>
<td>0.3737</td>
<td>0.3771</td>
<td>0.3739</td>
<td>0.3739</td>
</tr>
<tr>
<td>1.0</td>
<td>0.2875</td>
<td>0.2872</td>
<td>0.2902</td>
<td>0.2875</td>
<td>0.2875</td>
</tr>
<tr>
<td>3.0</td>
<td>0.0298</td>
<td>0.0297</td>
<td>0.0133</td>
<td>0.0298</td>
<td>0.0298</td>
</tr>
</tbody>
</table>

Example 3. Consider Burgers’ equation with the initial condition
\[
U(x, 0) = \sin(\pi x), \quad 0 \leq x \leq 1 \tag{22}
\]
and the boundary conditions
\[
U(0, t) = U(1, t) = 0. \tag{23}
\]

The analytical solution of this problem was given by Cole [2] in the term of an infinite series as
\[
U(x, t) = \frac{2\pi \lambda \sum_{k=1}^{\infty} k A_k \sin(k \pi x)}{A_0 + \sum_{k=1}^{\infty} A_k \cos(k \pi x)} \exp\left(-k^2 \pi^2 \lambda t\right) \tag{24}
\]
with the Fourier coefficients
\[
A_0 = \int_{0}^{1} \exp\left\{-(2\pi \lambda)^{-1}(1 - \cos(\pi x))\right\} \, dx,
\]
\[
A_k = 2 \int_{0}^{1} \exp\left\{-(2\pi \lambda)^{-1}(1 - \cos(\pi x))\right\} \cos(k \pi x) \, dx,
\]
\[k \geq 1. \tag{25}\]

In Table 4, we have computed the numerical solutions of this example at differential time levels with parameter values $\lambda = 0.1$, $h = 0.025$, and $\tau = 0.0001$. The comparison of our results with the exact solutions as well as the solutions obtained in [11, 15, 44] is reported in Table 4. From Table 4, we can find that the presented scheme provides better accuracy. Moreover, in Tables 5, 6 and 7, we compare our method with Hon and Mao’s scheme, Chen and Wu’s MQQI method, and Zhu’s BSQI method at $t = 1$ with $\tau = 0.001$, $h = 0.01$ for $\lambda = 0.1, 0.01, 0.0001$, respectively. For the MQQI method, the shape parameter $c = 7.2 \times 10^{-3}, 2.9 \times 10^{-3}, 1.43 \times 10^{-4}$ for Table 5, respectively, as [39]. Solutions found with the present method are in good agreement with the result and better than other methods. These show that the method works well.

Example 4. We consider particular solution of Burgers’ equation:
\[
U(x, t) = \frac{x/t}{1 + \sqrt{t/t_0} \exp(x^2/4\lambda t)}, \quad t \geq 1, \quad 0 \leq x \leq 1,
\]
where $t_0 = \exp(1/8\lambda)$. Initial condition is obtained from when $t = 1$ is used. Boundary conditions are $U(0, t) = U(1.2, t) = 0$. Analytical solution represents shock-like solution of the one-dimensional Burgers’ equation. Parameters $h = 0.02, 0.005$ and $\lambda = 0.005, 0.01$ are selected for comparison over the domain $[0, 1]$. Accuracy of our method is shown by calculating the error norms. These together with some previous results are given in Table 8. Table 8 shows that our method provides better accuracy than MQQI method and BSQI method. Although the accuracy is not higher than that of QBCM method, we know that, at each time step, the complexity of our method is lower than theirs. The numerical solutions are depicted with $h = 0.02, \tau = 0.001$, and $\lambda = 0.005$ for $t \leq 4$ in Figure 2.

5. Conclusion

Following the recent development of the quasi-interpolation method for scattered data interpolation and the meshfree method for solving partial differential equations, this paper combines these ideas and proposes a new meshfree quasi-interpolation method for Burgers’ equation. The method does not require solving a large size matrix equation and, hence, the ill-conditioning problem from using B-spline functions as global interpolants can be avoided. We have made comparison studies between the present results and the exact solutions. The agreement of our numerical results with those exact solutions is excellent. For the high-dimensional
Table 5: Comparison of results at $t = 1$ for $\lambda = 0.1$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>Hon and Mao [40]</th>
<th>MQQI [39]</th>
<th>BSQI [41]</th>
<th>Our method</th>
<th>Exact</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.0664</td>
<td>0.07124</td>
<td>0.06628</td>
<td>0.06630</td>
<td>0.06632</td>
</tr>
<tr>
<td>0.2</td>
<td>0.1313</td>
<td>0.13431</td>
<td>0.1315</td>
<td>0.13119</td>
<td>0.13121</td>
</tr>
<tr>
<td>0.3</td>
<td>0.1928</td>
<td>0.19339</td>
<td>0.19269</td>
<td>0.19271</td>
<td>0.19279</td>
</tr>
<tr>
<td>0.4</td>
<td>0.2481</td>
<td>0.24538</td>
<td>0.24792</td>
<td>0.24797</td>
<td>0.24803</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2919</td>
<td>0.28517</td>
<td>0.29175</td>
<td>0.29185</td>
<td>0.29191</td>
</tr>
<tr>
<td>0.6</td>
<td>0.3159</td>
<td>0.30473</td>
<td>0.31580</td>
<td>0.31598</td>
<td>0.31607</td>
</tr>
<tr>
<td>0.7</td>
<td>0.3079</td>
<td>0.29288</td>
<td>0.30791</td>
<td>0.30800</td>
<td>0.30810</td>
</tr>
<tr>
<td>0.8</td>
<td>0.2534</td>
<td>0.23784</td>
<td>0.25337</td>
<td>0.25344</td>
<td>0.25372</td>
</tr>
<tr>
<td>0.9</td>
<td>0.1459</td>
<td>0.13542</td>
<td>0.14583</td>
<td>0.14587</td>
<td>0.14606</td>
</tr>
</tbody>
</table>

Table 6: Comparison of results at $t = 1$ for $\lambda = 0.01$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>Hon and Mao [40]</th>
<th>MQQI [39]</th>
<th>BSQI [41]</th>
<th>Our method</th>
<th>Exact</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.0755</td>
<td>0.07868</td>
<td>0.07530</td>
<td>0.07538</td>
<td>0.0754</td>
</tr>
<tr>
<td>0.2</td>
<td>0.1507</td>
<td>0.15202</td>
<td>0.15049</td>
<td>0.15066</td>
<td>0.1506</td>
</tr>
<tr>
<td>0.3</td>
<td>0.2257</td>
<td>0.22554</td>
<td>0.22544</td>
<td>0.22573</td>
<td>0.2257</td>
</tr>
<tr>
<td>0.4</td>
<td>0.3003</td>
<td>0.29904</td>
<td>0.30002</td>
<td>0.30028</td>
<td>0.3003</td>
</tr>
<tr>
<td>0.5</td>
<td>0.3744</td>
<td>0.37226</td>
<td>0.37407</td>
<td>0.37473</td>
<td>0.3744</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4478</td>
<td>0.44484</td>
<td>0.44742</td>
<td>0.44778</td>
<td>0.4478</td>
</tr>
<tr>
<td>0.7</td>
<td>0.5202</td>
<td>0.51643</td>
<td>0.51985</td>
<td>0.52038</td>
<td>0.5203</td>
</tr>
<tr>
<td>0.8</td>
<td>0.5913</td>
<td>0.58622</td>
<td>0.59106</td>
<td>0.59151</td>
<td>0.5915</td>
</tr>
<tr>
<td>0.9</td>
<td>0.6607</td>
<td>0.62956</td>
<td>0.65964</td>
<td>0.66007</td>
<td>0.6600</td>
</tr>
</tbody>
</table>

Table 7: Comparison of results at $t = 1$ for $\lambda = 0.0001$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>Hon and Mao [40]</th>
<th>MQQI [39]</th>
<th>BSQI [41]</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.0422</td>
<td>0.0422</td>
<td>0.0422</td>
<td>0.0422</td>
</tr>
<tr>
<td>0.16</td>
<td>0.1263</td>
<td>0.1263</td>
<td>0.1262</td>
<td>0.1262</td>
</tr>
<tr>
<td>0.27</td>
<td>0.2103</td>
<td>0.2103</td>
<td>0.2096</td>
<td>0.2103</td>
</tr>
<tr>
<td>0.38</td>
<td>0.2939</td>
<td>0.2939</td>
<td>0.2928</td>
<td>0.2939</td>
</tr>
<tr>
<td>0.50</td>
<td>0.3769</td>
<td>0.3769</td>
<td>0.3754</td>
<td>0.3769</td>
</tr>
<tr>
<td>0.61</td>
<td>0.4592</td>
<td>0.4592</td>
<td>0.4573</td>
<td>0.4592</td>
</tr>
<tr>
<td>0.72</td>
<td>0.5404</td>
<td>0.5404</td>
<td>0.5381</td>
<td>0.5404</td>
</tr>
<tr>
<td>0.83</td>
<td>0.6203</td>
<td>0.6201</td>
<td>0.6174</td>
<td>0.6203</td>
</tr>
<tr>
<td>0.94</td>
<td>0.6983</td>
<td>0.6957</td>
<td>0.6947</td>
<td>0.6983</td>
</tr>
</tbody>
</table>

Table 8: Comparison of results at different times for $\tau = 0.01$.

<table>
<thead>
<tr>
<th>$h = 0.02, \lambda = 0.005$</th>
<th>$L_2 \times 10^3$</th>
<th>$L_\infty \times 10^3$</th>
<th>$L_2 \times 10^3$</th>
<th>$L_\infty \times 10^3$</th>
<th>$L_2 \times 10^3$</th>
<th>$L_\infty \times 10^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSQI [41]</td>
<td>1.66464</td>
<td>5.12020</td>
<td>2.06695</td>
<td>6.31491</td>
<td>2.36889</td>
<td>6.85425</td>
</tr>
<tr>
<td>QBCM I [17]</td>
<td>0.54058</td>
<td>1.16930</td>
<td>0.41864</td>
<td>1.93664</td>
<td>1.28863</td>
<td>5.54899</td>
</tr>
<tr>
<td>QBCM II [17]</td>
<td>0.49130</td>
<td>1.69300</td>
<td>0.41864</td>
<td>1.93664</td>
<td>1.28863</td>
<td>5.54899</td>
</tr>
<tr>
<td>Our method</td>
<td>0.6642</td>
<td>0.9125</td>
<td>0.7573</td>
<td>1.1465</td>
<td>0.8592</td>
<td>1.2103</td>
</tr>
<tr>
<td>$h = 0.02, \lambda = 0.01$</td>
<td>0.82751</td>
<td>2.59444</td>
<td>0.98595</td>
<td>2.35031</td>
<td>1.58264</td>
<td>5.73827</td>
</tr>
<tr>
<td>BSQI [41]</td>
<td>0.82751</td>
<td>2.59444</td>
<td>0.98595</td>
<td>2.35031</td>
<td>1.58264</td>
<td>5.73827</td>
</tr>
<tr>
<td>QBCM I [17]</td>
<td>0.7014</td>
<td>0.40431</td>
<td>0.20476</td>
<td>0.86363</td>
<td>1.29951</td>
<td>6.69425</td>
</tr>
<tr>
<td>QBCM II [17]</td>
<td>0.24003</td>
<td>0.88000</td>
<td>0.30849</td>
<td>1.14760</td>
<td>1.57548</td>
<td>8.06798</td>
</tr>
<tr>
<td>Our method</td>
<td>0.82751</td>
<td>0.50367</td>
<td>0.46281</td>
<td>1.05625</td>
<td>0.88261</td>
<td>4.73827</td>
</tr>
</tbody>
</table>
Burgers’ equations, we believe our scheme can also be applicable. In this case, we would use multivariate spline quasi-interpolation instead of univariate spline quasi-interpolation. We will consider these problems in our future work.

Conflict of Interests
The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments
This work is supported by the Disciplinary Construction Guide Foundation of Harbin Institute of Technology at Weihai (no. WH20140206) and the Scientific Research Foundation of Harbin Institute of Technology at Weihai (no. HIT(WH)201319).

References


