mathematics

MDPI

Article

Numerical Solution of High-Dimensional Shockwave Equations by Bivariate Multi-Quadric Quasi-Interpolation

Shenggang Zhang [1,2], Chungang Zhu [1,*] and Qinjiao Gao [3,4]

1 School of Mathematical Sciences, Dalian University of Technology, Dalian 116024, China
2 School of Public Health, Dalian Medical University, Dalian 116044, China
3 School of Business, Dalian University of Foreign Languages, Dalian 116044, China
4 Department of Mathematics, Missouri State University, Springfield, MO 65897, USA
* Correspondence: cgzhu@dlut.edu.cn

Received: 21 July 2019; Accepted: 9 August 2019; Published: 12 August 2019

Abstract: Radial basis function-based quasi-interpolation performs efficiently in high-dimensional approximation and its applications, which can attain the approximant and its derivatives directly without solving any large-scale linear system. In this paper, the bivariate multi-quadrics (MQ) quasi-interpolation is used to simulate two-dimensional (2-D) Burgers' equation. Specifically, the spatial derivatives are approximated by using the quasi-interpolation, and the time derivatives are approximated by forward finite difference method. One advantage of the proposed scheme is its simplicity and easy implementation. More importantly, the proposed scheme opens the gate to meshless adaptive moving knots methods for the high-dimensional partial differential equations (PDEs) with shock or soliton waves. The scheme is also applicable to other non-linear high-dimensional PDEs. Two numerical examples of Burgers' equation (shock wave equation) and one example of the Sine–Gordon equation (soliton wave equation) are presented to verify the high accuracy and efficiency of this method.

Keywords: high-dimensional PDE; shockwave; quasi-interpolation; meshless method

MSC: 41A25; 65D05; 65M06; 65M12

1. Introduction

Studying the numerical solution of high-dimensional non-linear Burgers' equation is of great significance. One reason is that it is a fundamental shockwave PDE from fluid mechanics which often occurs in applied and computational mathematics, such as modeling of dynamics, heat conduction, and acoustic wave [1,2]. More importantly, it often plays the role of a testing equation to check the feasibility of the scheme, i.e., once the scheme is efficient for Burgers' equation, there is great possibility that it is applicable to other equations with shock or soliton waves, such as non-linear Schrodinger equation [3], Sine–Gordon equation [4], and Klein–Gordon equation [5].

In this paper, the proposed scheme is applicable to other non-linear high-dimensional PDEs. Burgers' equation has been studied by many researchers not only because its shockwave behavior when the coefficient R is large, but also because it is the simplest form of non-linear advection equation. Hence we take the 2-D coupled Burgers' equation as one example:

$$\begin{aligned} u_t + uu_x + vu_y &= \tfrac{1}{R}(u_{xx} + u_{yy}), \\ v_t + uv_x + vv_y &= \tfrac{1}{R}(v_{xx} + v_{yy}). \end{aligned} \tag{1}$$

The initial conditions are:

$$\begin{aligned} u(x,y,0) &= f_1(x,y),\ (x,y) \in D, \\ v(x,y,0) &= f_2(x,y),\ (x,y) \in D, \end{aligned} \tag{2}$$

and the boundary conditions are:

$$\begin{aligned} u(x,y,t) &= g_1(x,y,t),\ (x,y) \in \partial D, \\ v(x,y,t) &= g_2(x,y,t),\ (x,y) \in \partial D, \end{aligned} \tag{3}$$

where $t \in [0,T]$, $D \subset \mathbb{R}^2$ is a bounded domain and ∂D is the boundary. $u(x,y,t)$ and $v(x,y,t)$ are the coupled solutions to be determined, $f_1(x,y)$, $f_2(x,y)$, $g_1(x,y,t)$ and $g_2(x,y,t)$ are given functions, and R is the Reynolds number. Fletcher [6] obtained its exact solutions, so that one could evaluate the quality of numerical solutions.

During the past decades, various numerical techniques have been developed for the solution of 2-D Burgers' equation. Bahadir [7] introduced one fully implicit finite difference scheme, then Zhu et al. [8] proposed the discrete Adomian decomposition method (ADM), Yu et al. [9] applied the bivariate quintic spline to solve numerically.

Compared with these mesh-based methods, which depend on a topology shape of the mesh, the meshless method performs satisfactorily for problems with complicated and irregular geometries [10]. In recent years, the meshless collocation method has drawn considerable attention for solving PDEs [11–13]. Mittal and Tripathi [14] proposed a collocation method based on Modified bi-cubic B-Spline functions. However, it requires to solve the inverse of a large-scale matrix on each time step, which costs computational time and might be ill-conditioned.

Considering the problems mentioned above, we try to apply the MQ quasi-interpolation method for the numerical solution of PDEs because of the following reasons:

- Quasi-interpolation could give the approximant and its derivatives directly needing no solution of any large-scale system, hence it is easy to be implemented by the engineers in applications.
- Quasi-interpolation could filter the noise of the data which often occurs in applications because of various reasons, e.g., measure error, data pollution, and so forth.
- Ma [15] proved the superiority of MQ quasi-interpolation over finite difference and other interpolation methods in both stability and accuracy, especially when approximating scattered data.

MQ function was firstly introduced by Hardy [16], then it has been investigated extensively [17–19] and successfully applied to solve one dimensional (1-D) PDEs [12,20–23]. Due to the extraordinary performance in 1-D, Ling [24] extended the univariate MQ quasi-interpolation to 2-D. Feng and Zhou [25], Wu et al. [26] and Wu et al. [27,28] constructed high-dimensional quasi-interpolations.

The ambition of this paper is to construct a numerical scheme for solving 2-D Burgers' equation applying the bivariate MQ quasi-interpolation. Firstly, we develop a bivariate MQ quasi-interpolation and analyze its approximation order to a given function and its high order derivatives. Then the numerical scheme for solving the 2-D coupled Burgers' equation is proposed applying the bivariate MQ quasi-interpolation. In the numerical scheme, the spatial derivatives are approximated by using the derivatives of the quasi-interpolation, and the time derivatives of the dependent variables are approximated by the forward finite difference method. The truncation error and the total error of our scheme are estimated to show the accuracy of the proposed scheme. At last, two numerical examples of

Burgers' equation (shockwave equation) and one example of the Sine–Gordon equation (soliton wave equation) are presented to verify the efficiency of the proposed method.

One advantage of the proposed scheme is its simplicity, easy implementation, but high accuracy and stability. More importantly, since the proposed method requires no triangulation of the region, adaptive refinement can be added in the scheme directly. In other words, it allows more flexible knots movement and one need not to worry the problems of mesh over gathering or condition numbers. In addition, the proposed scheme is applicable to other high-dimensional PDEs with shock or even soliton waves.

The rest of the paper is organized as follows. In Section 2, the bivariate MQ quasi-interpolant is introduced. In Section 3, we present the numerical scheme to solve 2-D Burgers' equation and give the error estimations. In Section 4, three numerical examples are proposed to verify our method. Section 5 concludes the whole paper.

2. Bivariate MQ Quasi-Interpolations

In this section, we firstly introduce the univariate MQ quasi-interpolation. On the basis, we develop the bivariate MQ quasi-interpolation and show its approximation order to the function and its derivatives.

Given the data $\{x_j, f_j\}$, $f_j = f(x_j)$, the univariate MQ quasi-interpolation is

$$Qf(x) = \sum_j f_j \psi_j(x), \tag{4}$$

$$\psi_j(x) = \frac{\phi_{j+1}(x) - \phi_j(x)}{2(x_{j+1} - x_j)} - \frac{\phi_j(x) - \phi_{j-1}(x)}{2(x_j - x_{j-1})}, \tag{5}$$

$\phi_j(x) = \sqrt{(x - x_j)^2 + c_1^2}$ is named MQ function, c_1 is shape parameter, which is often a small constant. As c_1 goes to zero, $\phi_j(x)$ will converge to $|x - x_j|$, consequently the univariate MQ interpolation could be taken as piecewise linear interpolation but possesses smoothness property. Denoting by $h_1 = \max_j(x_{j+1} - x_j)$, the approximation error of $Qf(x)$ to the original function $f(x)$ and the k-th derivatives were given in [19]:

Theorem 1. *If $f(x) \in \mathbb{C}^{(k+2)}(\mathbb{R})$ and $f^{(j)}(x)$ is bounded by a polynomial of degree $k + 2 - j$, then*

$$|Qf^{(k)}(x) - f^{(k)}(x)| \leq \mathcal{O}(h_1^{\frac{2}{k+1}}) \tag{6}$$

holds, provided that $c_1 = \mathcal{O}(h_1^{\frac{1}{k+1}})$.

Besides the high accuracy, the MQ quasi-interpolation has been proven to possess many other favorable properties, such as monotonicity preservation, shape preservation, and variation diminishing [17,18]. Because of these positive properties, researchers began to extend the univariate quasi-interpolation to multi-dimension [24–26].

In this paper, we propose one kind of bivariate MQ quasi-interpolation using the tensor product technique. Specifically, given data (x_i, y_j, f_{ij}) and $f_{ij} = f(x_i, y_j)$, the bivariate MQ quasi-interpolation is defined as:

$$(Qf)(x, y) = \sum_i \sum_j f_{ij} \psi_i(x) \psi_j(y), \tag{7}$$

where $\psi_i(x)$ are described in (5) and $\psi_j(y)$ are represented as follows:

$$\phi_j(y) = \sqrt{(y-y_j)^2 + c_2^2},$$
$$\psi_j(y) = \frac{\phi_{j+1}(y) - \phi_j(y)}{2(y_{j+1} - y_j)} - \frac{\phi_j(y) - \phi_{j-1}(y)}{2(y_j - y_{j-1})},$$

c_2 is a small constant. Denoting $h_2 = \max_j(y_{j+1} - y_j)$, the error estimation of the proposed bivariate MQ quasi-interpolation (7) is given in Theorem 2.

Theorem 2. *If $f(x,y) \in \mathbb{C}^{(k+2)}(\mathbb{R}^2)$ and $\frac{\partial (Qf)^{i+j}(x,y)}{\partial x^i \partial y^j}$ is bounded by a polynomial $p_1(x)p_2(y)$, $p_1(x)$ and $p_2(y)$ are polynomials of degree $\alpha_1 + 2 - i$ and $\alpha_2 + 2 - j$ respectively ($\alpha_1, \alpha_2, \alpha_1 + 2 - i$, and $\alpha_2 + 2 - j$ are all nonnegative integers), then the error of bivariate MQ quasi-interpolation (7) satisfies*

$$\left\| \frac{\partial (Qf)^{\alpha_1+\alpha_2}(x,y)}{\partial x^{\alpha_1} \partial y^{\alpha_2}} - \frac{\partial f^{\alpha_1+\alpha_2}(x,y)}{\partial x^{\alpha_1} \partial y^{\alpha_2}}) \right\|_\infty \leqslant \mathcal{O}(h_1^{\frac{2}{\alpha_1+1}} + h_2^{\frac{2}{\alpha_2+1}}), \tag{8}$$

provided that $c_1 = \mathcal{O}(h_1^{\frac{1}{\alpha_1+1}})$ and $c_2 = \mathcal{O}(h_2^{\frac{1}{\alpha_2+1}})$.

Proof. By adding one middle term, the following inequalities are obtained:

$$\begin{array}{rl} & \| (Qf)(x,y) - f(x,y) \|_\infty \\ = & \| \sum_i \sum_j f_{ij}\psi_i(x)\psi_j(y) - \sum_i f(x_i,y)\psi_i(x) + \sum_i f(x_i,y)\psi_i(x) - f(x,y) \|_\infty \\ \leq & \| \sum_i \psi_i(x)(\sum_j f_{ij}\psi_j(y) - f(x_i,y)) \|_\infty + \| \sum_i f(x_i,y)\psi_i(x) - f(x,y) \|_\infty . \end{array}$$

Since $\sum_j f_{ij}\psi_j(y)$ and $\sum_i f(x_i,y)\psi_i(x)$ are the approximation of $f(x_i,y)$ and $f(x,y)$ separately, when $c_1 = \mathcal{O}(h_1)$ and $c_2 = \mathcal{O}(h_2)$, they are estimated as:

$$\| \sum_j f_{ij}\psi_j(y) - f(x_i,y) \|_\infty = \mathcal{O}(h_2^2),$$
$$\| \sum_i f(x_i,y)\psi_i(x) - f(x,y) \|_\infty = \mathcal{O}(h_1^2).$$

In addition, $| \sum_i \psi_i(x) | \leq 1$, the result are as follows:

$$\| Qf(x,y) - f(x,y) \|_\infty \leq | \sum_i \psi_i(x) | \, \mathcal{O}(h_1^2) + \mathcal{O}(h_2^2) \leq \mathcal{O}(h_1^2 + h_2^2).$$

Similarly, the derivative errors of $(Qf)^{(k)}$ to $f^{(k)}$ could be estimated based on Theorem 1. This ends the proof. □

3. Numerical Scheme Applying Bivariate MQ Quasi-Interpolant

In this section, we develop a numerical scheme for solving 2-D Burgers' Equations (1)–(3) based on bivariate MQ quasi-interpolation in Section 3.1. Then we provide the error estimations of Algorithm 1 in Section 3.2.

3.1. The Main Algorithm

We represent the approximations of $u(x,y,t)$ and $v(x,y,t)$ at the knots $(x_i, y_j, t_k) = (ih_1, jh_2, k\tau)$ by u_{ij}^k and v_{ij}^k respectively, where h_1 and h_2 are the mesh sizes in x and y direction separately, τ is the time step sizes, i.e., $u_{ij}^k \approx u(x_i,y_j,t_k)$, $v_{ij}^k \approx v(x_i,y_j,t_k)$. We use the similar expressions for the derivative

approximations, e.g., $(u_x)_{ij}^k \approx u_x(x_i, y_j, t_k)$, $(u_y)_{ij}^k \approx u_y(x_i, y_j, t_k)$ and so forth. The numerical solution could be realized in the following Algorithm 1.

3.2. Error Estimations of Algorithm 1

In Section 3.2, the truncation error of Algorithm 1 is proposed in Lemma 1. Then based on Lemma 1, the total error of Algorithm 1 is provided in Theorem 3.

Algorithm 1 Numerical scheme of 2-D Burgers' equation applying the bivariate MQ quasi-interpolation.

Input: u_{ij}^k, v_{ij}^k

Output: u_{ij}^{k+1}, v_{ij}^{k+1}

1: Approximate the solutions' (u and v) first and second space derivatives by using the bivariate MQ quasi-interpolation. For example, the first derivatives of function u could be approximated as follows,

$$(u_x)_{ij}^k = \sum_m \sum_n u_{ij}^k \psi_m'(x_i)\psi_n(y_j),$$
$$(u_y)_{ij}^k = \sum_m \sum_n u_{ij}^k \psi_m(x_i)\psi_n'(y_j).$$

the other derivatives can be approximated similarly based on the bivariate MQ quasi-interpolation.

2: Discrete the Burgers' equation in time and substitute the points (x_i, y_j) into the derivatives, compute u_{ij}^{k+1} and v_{ij}^{k+1}:

$$u_{ij}^{k+1} = u_{ij}^k - \tau u_{ij}^k (u_x)_{ij}^k - \tau v_{ij}^k (u_y)_{ij}^k + \frac{\tau}{R}((u_{xx})_{ij}^k + (u_{yy})_{ij}^k),$$
$$v_{ij}^{k+1} = v_{ij}^k - \tau u_{ij}^k (v_x)_{ij}^k - \tau v_{ij}^k (v_y)_{ij}^k + \frac{\tau}{R}((v_{xx})_{ij}^k + (v_{yy})_{ij}^k).$$

3: Return to Step 1.

For the error estimation, we denote the following operators $L_1, L_2, L_{1,d}, L_{2,d}$ as

$$L_1(u,v) = u_t + uu_x + vu_y - \tfrac{1}{R}(u_{xx} + u_{yy}),$$
$$L_2(u,v) = v_t + uv_x + vv_y - \tfrac{1}{R}(v_{xx} + v_{yy}),$$
$$L_{1,d}(u_{ij}^k, v_{ij}^k) = \tfrac{u_{ij}^{k+1} - u_{ij}^k}{\tau} + u_{ij}^k (u_x)_{ij}^k + v_{ij}^k (u_y)_{ij}^k - \tfrac{1}{R}((u_{xx})_{ij}^k + (u_{yy})_{ij}^k),$$
$$L_{2,d}(u_{ij}^k, v_{ij}^k) = \tfrac{v_{ij}^{k+1} - v_{ij}^k}{\tau} + u_{ij}^k (v_x)_{ij}^k + v_{ij}^k (v_y)_{ij}^k - \tfrac{1}{R}((v_{xx})_{ij}^k + (v_{yy})_{ij}^k).$$

Lemma 1. *For any $t \in [0, T]$, if $u(x,t) \in \mathbb{C}^4(\mathbb{R})$ and $v(x,t) \in \mathbb{C}^4(\mathbb{R})$, the truncation error of Algorithm 1 is*

$$(R_1)_{ij}^k = L_{1,d}(u_{ij}^k, v_{ij}^k) - L_1(u,v)(x_i, y_j, t_k) = \mathcal{O}(\tau + h_1^{2/3} + h_2^{2/3}),$$
$$(R_2)_{ij}^k = L_{2,d}(u_{ij}^k, v_{ij}^k) - L_2(u,v)(x_i, y_j, t_k) = \mathcal{O}(\tau + h_1^{2/3} + h_2^{2/3}).$$

Proof. Firstly, based on the Taylor expansion, we have:

$$\tfrac{u(x_i, y_j, t_{k+1}) - u(x_i, y_j, t_k)}{\tau} = u_t(x_i, y_j, t_k) + \mathcal{O}(\tau),$$
$$\tfrac{v(x_i, y_j, t_{k+1}) - v(x_i, y_j, t_k)}{\tau} = v_t(x_i, y_j, t_k) + \mathcal{O}(\tau).$$

Then based on Theorem 2, we get the following error estimations:

$$\begin{aligned}
&u(x_i,y_j,t_k)(u_x)_{ij}^k + v(x_i,y_j,t_k)(u_y)_{ij}^k \\
&= u(x_i,y_j,t_k)(u_x(x_i,y_j,t_k) + \mathcal{O}(h_1)) + v(x_i,y_j,t_k)(u_y(x_i,y_j,t_k) + \mathcal{O}(h_2)), \\
&u(x_i,y_j,t_k)(v_x)_{ij}^k + v(x_i,y_j,t_k)(v_y)_{ij}^k \\
&= u(x_i,y_j,t_k)(v_x(x_i,y_j,t_k) + \mathcal{O}(h_1)) + v(x_i,y_j,t_k)(v_y(x_i,y_j,t_k) + \mathcal{O}(h_2)),
\end{aligned}$$

and

$$\begin{aligned}
(u_{xx})_{ij}^k + (u_{yy})_{ij}^k &= u_{xx}(x_i,y_j,t_k) + u_{yy}(x_i,y_j,t_k) + \mathcal{O}(h_1^{\frac{2}{3}} + h_2^{\frac{2}{3}}), \\
(v_{xx})_{ij}^k + (v_{yy})_{ij}^k &= v_{xx}(x_i,y_j,t_k) + v_{yy}(x_i,y_j,t_k) + \mathcal{O}(h_1^{\frac{2}{3}} + h_2^{\frac{2}{3}}).
\end{aligned}$$

Finally, we get the truncation errors:

$$\begin{aligned}
(R_1)_{ij}^k &= L_{1,d}(u_{ij}^k, v_{ij}^k) - L_1(u,v)(x_i,y_j,t_k) \\
&= \mathcal{O}(\tau) + \mathcal{O}(h_1+h_2) - \tfrac{1}{R}\mathcal{O}(h_1^{2/3} + h_2^{2/3}) \\
&= \mathcal{O}(\tau + h_1^{2/3} + h_2^{2/3}), \\
(R_2)_{ij}^k &= L_{2,d}(u_{ij}^k, v_{ij}^k) - L_2(u,v)(x_i,y_j,t_k) \\
&= \mathcal{O}(\tau) + \mathcal{O}(h_1+h_2) - \tfrac{1}{R}\mathcal{O}(h_1^{2/3} + h_2^{2/3}) \\
&= \mathcal{O}(\tau + h_1^{2/3} + h_2^{2/3}).
\end{aligned}$$

This ends the proof. □

Suppose $u_k(x,y)$ and $v_k(x,y)$ are the simulations of the exact solutions $u(x,y,t_k)$ and $v(x,y,t_k)$ using Algorithm 1, the total error of Algorithm 1 is provided in Theorem 3:

Theorem 3. *Defining the total errors of Algorithm 1 as* $e_u^k(x,y) = u_k(x,y) - u(x,y,t_k)$, $e_v^k(x,y) = v_k(x,y) - v(x,y,t_k)$, *if* $u(x,y,t) \in \mathbb{C}^4(\mathbb{R})$, $v(x,y,t) \in \mathbb{C}^4(\mathbb{R})$, *the error estimation of Algorithm 1 satisfies:*

$$\begin{pmatrix} \|e_u^K\|_{2,\infty} \\ \|e_v^K\|_{2,\infty} \end{pmatrix} = \mathcal{O}(\tau + h_1^{2/3} + h_2^{2/3}), K = T/\tau, \tag{9}$$

where the Soblev norm for the function $g(x,y)$ *is defined as:*

$$\|g\|_{2,\infty} = \max_{0 \le \alpha_1+\alpha_2 \le 2} \|\frac{\partial^{\alpha_1+\alpha_2} g}{\partial^{\alpha_1} x \partial^{\alpha_2} y}\|_\infty.$$

Proof. Since $u(x,y,t), v(x,y,t)$ are the exact solutions of (1), they satisfy:

$$\begin{aligned}
\frac{u(x,y,t_{k+1}) - u(x,y,t_k)}{\tau} &= -u(x,y,t_k)u_x'(x,y,t_k) - v(x,y,t_k)u_y'(x,y,t_k) \\
&\quad + \frac{1}{R}(u_{xx}''(x,y,t_k) + u_{yy}''(x,y,t_k)) + \mathcal{O}(\tau), \\
\frac{v(x,y,t_{k+1}) - v(x,y,t_k)}{\tau} &= -u(x,y,t_k)v_x'(x,y,t_k) - v(x,y,t_k)v_y'(x,y,t_k) \\
&\quad + \frac{1}{R}(v_{xx}''(x,y,t_k) + v_{yy}''(x,y,t_k)) + \mathcal{O}(\tau).
\end{aligned} \tag{10}$$

According to Lemma 1, $u_k(x,y), v_k(x,y)$ satisfy

$$\begin{aligned}\frac{u_{k+1}(x,y)-u_k(x,y)}{\tau} =& -u_k(x,y)(u_k)'_x(x,y)-v_k(x,y)(u_k)'_y(x,y)\\ &+\frac{1}{R}((u_k)''_{xx}(x,y)+(u_k)''_{yy}(x,y))+\mathcal{O}(h_1^{2/3}+h_2^{2/3}),\\ \frac{v_{k+1}(x,y)-v_k(x,y)}{\tau} =& -u_k(x,y)(v_k)'_x(x,y)-v_k(x,y)(v_k)'_y(x,y)\\ &+\frac{1}{R}m((v_k)''_{xx}(x,y)+(v_k)''_{yy}(x,y))+\mathcal{O}(h_1^{2/3}+h_2^{2/3}).\end{aligned} \tag{11}$$

Subtracting (10) from (11), one can get the following equation by adding some middle terms:

$$\begin{aligned}\frac{e_u^{k+1}(x,y)-e_u^k(x,y)}{\tau} =& -e_u^k(x,y)(u_k)'_x(x,y)-u_k(x,y)(e_u^k)'_x(x,y)\\ &-e_v^k(x,y)(u_k)'_y(x,y)-v_k(x,y)(e_u^k)'_y(x,y)\\ &+\frac{1}{R}((e_u^k)''_{xx}(x,y)+(e_u^k)''_{yy}(x,y))+\mathcal{O}(\tau+h_1^{2/3}+h_2^{2/3}),\\ \frac{e_v^{k+1}(x,y)-e_v^k(x,y)}{\tau} =& -e_u^k(x,y)(v_k)'_x(x,y)-u_k(x,y)(e_v^k)'_x(x,y)\\ &-e_v^k(x,y)(v_k)'_x(x,y)-v_k(x,y)(e_v^k)'_x(x,y)\\ &+\frac{1}{R}((e_v^k)''_{xx}(x,y)+(e_v^k)''_{yy}(x,y))+\mathcal{O}(\tau+h_1^{2/3}+h_2^{2/3}).\end{aligned} \tag{12}$$

Therefore, the Soblev norm of the errors satisfy

$$\begin{aligned}\tfrac{\|e_u^{k+1}\|_{2,\infty}-\|e_u^k\|_{2,\infty}}{\tau} &\le C(\|e_u^k\|_{2,\infty}+\|e_v^k\|_{2,\infty})+\mathcal{O}(\tau+h_1^{2/3}+h_2^{2/3}),\\ \tfrac{\|e_v^{k+1}\|_{2,\infty}-\|e_v^k\|_{2,\infty}}{\tau} &\le C(\|e_u^k\|_{2,\infty}+\|e_v^k\|_{2,\infty})+\mathcal{O}(\tau+h_1^{2/3}+h_2^{2/3}),\end{aligned}$$

where $C=2max\{\|u\|_{1,\infty},\|v\|_{1,\infty},\frac{1}{R}\}$.

Hence we have

$$\begin{aligned}\begin{pmatrix}\|e_u^{k+1}\|_{2,\infty}\\ \|e_v^{k+1}\|_{2,\infty}\end{pmatrix} =& B\begin{pmatrix}\|e_u^k\|_{2,\infty}\\ \|e_v^k\|_{2,\infty}\end{pmatrix}+\mathcal{O}(\tau^2+\tau h_1^{2/3}+\tau h_2^{2/3})\\ &\cdots\\ =& B^{k+1}\begin{pmatrix}\|e_u^0\|_{2,\infty}\\ \|e_v^0\|_{2,\infty}\end{pmatrix}+(B^k+B^{k-1}+\cdots+B^0)\mathcal{O}(\tau^2+\tau h_1^{2/3}+\tau h_2^{2/3}),\end{aligned}$$

where $B=\begin{pmatrix}1+C\tau & C\tau\\ C\tau & 1+C\tau\end{pmatrix}$. B is a positive definite matrix, hence there exists a nonsingular matrix X satisfying

$$B=X^{-1}\begin{pmatrix}1 & 0\\ 0 & 1+2C\tau\end{pmatrix}X.$$

Since $\|e_u^0\|_{2,\infty} = 0$, $\|e_v^0\|_{2,\infty} = 0$, we can get

$$\begin{pmatrix} \|e_u^K\|_{2,\infty} \\ \|e_v^K\|_{2,\infty} \end{pmatrix} = X^{-1} \begin{pmatrix} K & 0 \\ 0 & \frac{1-(1+2C\tau)^K}{-2C\tau} \end{pmatrix} X\mathcal{O}(\tau^2 + \tau h_1^{2/3} + \tau h_2^{2/3}) = \mathcal{O}(\tau + h_1^{2/3} + h_2^{2/3}).$$

This completes the proof. □

Remark 1. *In this section, we apply the bivariate MQ quasi-interpolation for solving the 2-D coupled Burgers' equation. As is shown in Ma [15], the MQ method is more accurate, more stable, and simpler for simulating the high order derivatives than finite difference method, especially for scatted data or the data with noise. In addition, the proposed method is simple and easy to implement, since one need not to solve any large-scale linear system and could get the original function and derivatives' approximation directly.*

4. Numerical Examples

2-D Burgers' Equation

In this section, we select two presentative test examples to illustrate the performance of the method described in the previous section. There are several ways to process the boundary, here the popular bivariate operator which appears in Ling [24] is employed. All these experiments are carried out on a computer with an Intel (R) Core (TM) i7-7500 CPU, 3.20 GHZ processor and 8.00 GB RAM. To check the validity of the scheme, the root mean square (RMS) and L_∞ error norms are applied to make comparisons with the previous methods [8]. The error norms are defined as

$$\begin{aligned} \text{RMS} &= \sqrt{\frac{\sum_i \sum_j (u_{ij}^{exact} - u_{ij}^{num})^2}{(N_1+1)(N_2+1)}}, \\ L_\infty &= \max_{ij} |u_{ij}^{exact} - u_{ij}^{num}|, \end{aligned}$$

where u_{ij}^{exact} and u_{ij}^{num} are the exact and numerical results of u at the knot (x_i, y_j).

Example 1. *In this problem, we choose the computational domain as* $D = \{(x, y) : 0 \le x \le 0.5, 0 \le y \le 0.5\}$, *and the spatial knots distribute uniformly on the computational domain with a mesh width* $h_1 = h_2 = 0.025$. *For the Burgers' Equations* (1) *and* (2), *the initial conditions at* $t = 0$ *is*

$$\begin{aligned} f_1(x, y) &= x + y, \qquad (x, y) \in D, \\ f_2(x, y) &= x - y, \qquad (x, y) \in D. \end{aligned} \tag{13}$$

Under the above conditions, the exact solutions could be given [6]:

$$u(x, y, t) = \frac{x + y - 2xt}{1 - 2t^2}, \tag{14}$$

$$v(x, y, t) = \frac{x - y - 2yt}{1 - 2t^2}. \tag{15}$$

In this example, we observe the performance of our method with $R = 100$ for the convenience of comparison, and the numerical solutions are simulated using uniform knots, with a space width $h_1 = h_2 = 0.025$, the time step is $\tau = 10^{-4}$. The parameter is set as $c = 2.65 \times 10^{-6}$.

The study compares the errors of the presented method and Zhu et al.'s method [8]. The numerical errors at time $t = 0.1$ and $t = 0.4$ are listed in Tables 1 and 2 respectively. It is observed that at time $t = 0.1$, our errors are similar(even a little larger) than Zhu et al.'s method [8]. However, at time $t = 0.4$, our method is far smaller than Zhu et al.'s method [8]. That is to say, our method could maintain a longer time simulation.

Table 1. Example 1: Comparisons of numerical errors by Zhu et al. [8] and the present method with $R = 100$, $\tau = 10^{-4}$, at $t = 0.1$.

Mesh Grid	Error of u		Error of v	
	Zhu et al. [8]	**The Present**	**Zhu et al. [8]**	**The Present**
(0.1, 0.1)	3.3075×10^{-6}	3.3081×10^{-6}	1.0538×10^{-6}	1.0524×10^{-6}
(0.3, 0.1)	5.5616×10^{-6}	7.6686×10^{-6}	3.3077×10^{-5}	7.5178×10^{-6}
(0.2, 0.2)	6.6152×10^{-6}	6.6161×10^{-6}	2.1077×10^{-6}	2.1049×10^{-6}
(0.4, 0.2)	8.8694×10^{-6}	1.0977×10^{-5}	2.2540×10^{-6}	8.5703×10^{-6}
(0.1, 0.3)	7.6693×10^{-6}	5.8456×10^{-6}	7.5234×10^{-6}	3.3081×10^{-6}
(0.3, 0.3)	9.9233×10^{-6}	9.9242×10^{-6}	3.1615×10^{-6}	3.1573×10^{-6}
(0.2, 0.4)	1.0977×10^{-5}	1.0977×10^{-6}	8.5770×10^{-6}	2.2556×10^{-6}
(0.3, 0.4)	1.2104×10^{-5}	1.1052×10^{-5}	6.3960×10^{-6}	9.7708×10^{-7}
(0.5, 0.5)	1.6539×10^{-5}	1.6540×10^{-5}	5.2692×10^{-6}	5.2622×10^{-6}

Table 2. Example 1: Comparison of the numerical errors by Zhu et al. [8] and the present method with $R = 100$, $\tau = 10^{-4}$ at $t = 0.4$.

Mesh Grid	Error of u		Error of v	
	Zhu et al. [8]	**The Present**	**Zhu et al. [8]**	**The Present**
(0.1, 0.1)	1.0195×10^{-4}	2.2716×10^{-5}	3.5483×10^{-4}	4.2353×10^{-5}
(0.3, 0.1)	5.5872×10^{-4}	8.7786×10^{-5}	1.0195×10^{-4}	1.9213×10^{-4}
(0.2, 0.2)	2.0389×10^{-4}	4.5433×10^{-5}	7.0967×10^{-4}	8.4705×10^{-5}
(0.4, 0.2)	6.6067×10^{-4}	1.0502×10^{-5}	4.5678×10^{-4}	2.3448×10^{-4}
(0.1, 0.3)	1.5094×10^{-4}	3.0802×10^{-6}	1.3174×10^{-3}	2.2716×10^{-5}
(0.3, 0.3)	3.0584×10^{-4}	6.8149×10^{-6}	1.0645×10^{-3}	1.2706×10^{-4}
(0.2, 0.4)	4.8996×10^{-5}	2.5797×10^{-5}	1.6722×10^{-3}	1.9636×10^{-5}
(0.3, 0.4)	1.7939×10^{-4}	1.0068×10^{-4}	1.5458×10^{-3}	9.4524×10^{-5}
(0.5, 0.5)	5.0973×10^{-4}	1.1358×10^{-4}	1.7742×10^{-3}	2.1176×10^{-4}

In Table 3, the L_∞ errors and the CPU time of the present method at $t = 0.1$ and $t = 0.4$ are listed. One can conclude that our method is efficient in sense that it could get a satisfying error with rather little CPU time. Figure 1 describes the numerical solutions of $u(x, y, t), v(x, y, t)$ by the present method, as expected, the present method could simulate Example 1 quickly and rather precisely.

Table 3. Example 1: The L_∞ errors and CPU time of the present method at $t = 0.1$ and $t = 0.4$ with $\tau = 10^{-4}$.

	$L_\infty(u)$	$L_\infty(v)$	**CPU Time (s)**
$t = 0.1s$	1.6540	1.6163×10^{-5}	6.728
$t = 0.4s$	1.6267×10^{-4}	3.7444×10^{-4}	25.658

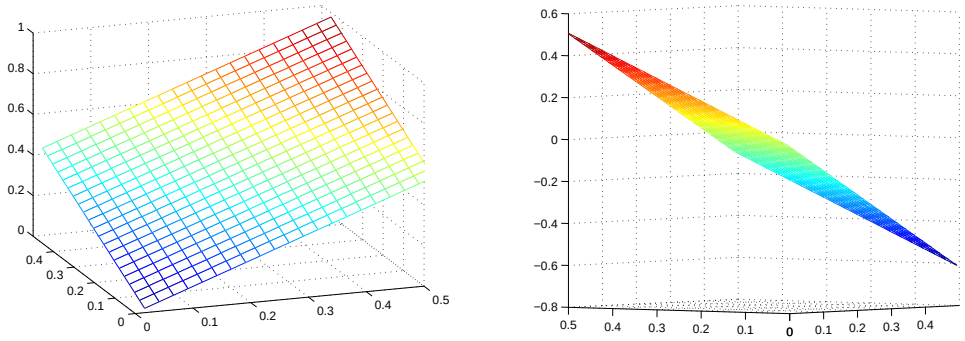


Figure 1. Example 1: Numerical solutions of $u(x,y,t)$, and $v(x,y,t)$ by the present method with $\tau = 10^{-4}$, $h_1 = h_2 = 0.025$, at $t = 0.1$.

Example 2. *When applying the Hopf-Cole transformation in [6], one could get the exact solutions of Burgers' Equations* (1) *and* (2)

$$u(x,y,t) = \frac{3}{4} - \frac{1}{4(1+exp((-4x+4y-t)R/32)}, \tag{16}$$

$$v(x,y,t) = \frac{3}{4} + \frac{1}{4(1+exp((-4x+4y-t)R/32)}. \tag{17}$$

In this example, we choose the computational domain as $D = \{(x,y) : 0 \leq x \leq 1, 0 \leq y \leq 1\}$, and we take the initial and boundary conditions from the above exact solution. The knots are distributed uniformly on the computational domain with a width $h_1 = h_2 = 0.05$, and $R = 100$. However the time step sizes of our method ($\tau = 10^{-3}$) in this example is chosen to be ten times of the method in Zhu et al. [8], and Bahadir [7] ($\tau = 10^{-4}$). The parameter c is chosen as $c = 1.53 \times 10^{-5}$. The numerical results will show that we could get the similar accuracy while with little CPU time.

In Table 4, the L_∞ numerical errors of the solutions at some typical mesh points at time $t = 0.01$, 0.05 and 0.2 with $\tau = 10^{-3}$. It is observed that our method could get satisfactory accuracy. In addition, the CPU time is rather little. The numerical results using the proposed algorithm are in good agreement with the exact solutions.

Table 4. Example 2: The L_∞ numerical errors and CPU time by the present method with $R = 100$, $\tau = 10^{-3}$ at different time t.

Mesh Grid	$t = 0.01$	$t = 0.05$	$t = 0.2$
(0, 1)	1.09714×10^{-8}	3.01044×10^{-7}	1.39170×10^{-5}
(0.1, 0.9)	5.21864×10^{-8}	2.88925×10^{-7}	1.12905×10^{-5}
(0.2, 0.8)	6.05494×10^{-7}	3.45039×10^{-6}	1.52312×10^{-5}
(0.3, 0.7)	6.52640×10^{-6}	3.53596×10^{-5}	1.70559×10^{-4}
(0.4, 0.6)	6.56867×10^{-6}	1.00372×10^{-6}	5.22325×10^{-4}
(0.5, 0.5)	5.55844×10^{-5}	1.67227×10^{-4}	6.56682×10^{-4}
(0.6, 0.4)	3.08342×10^{-5}	1.27268×10^{-4}	2.60973×10^{-4}
(0.7, 0.3)	2.12905×10^{-7}	1.01505×10^{-6}	7.31540×10^{-6}
(0.8, 0.2)	1.47614×10^{-8}	6.45096×10^{-8}	9.81073×10^{-6}
(0.9, 0.1)	2.52669×10^{-9}	4.25191×10^{-8}	1.28665×10^{-5}
(1, 0)	3.35256×10^{-8}	4.05295×10^{-7}	1.00646×10^{-5}
CPU time (s)	0.130	0.404	1.418

In Table 5, we compare the numerical errors of the present method with time step $\tau = 10^{-3}$, and that of Zhu et al. [8], and Bahadir [7] with time step size $\tau = 10^{-4}$. It is observed that even using the time step that is ten times of the method in Zhu et al. [8], and Bahadir [7], our method could get similar accuracy.

Table 5. Example 2: The error of numerical results by Zhu et al. [8], Bahadir[7] with $R = 100$, $\tau = 10^{-4}$, at $t = 0.01$, and the present method with $\tau = 10^{-3}$.

Mesh Grid	Error of u			Error of v		
	Zhu et al. [8]	**Bahadir** [7]	**The Present**	**Zhu et al.** [8]	**Bahadir** [7]	**The Present**
(0.1, 0.1)	5.91368×10^{-5}	7.24132×10^{-5}	5.44503×10^{-5}	5.91368×10^{-5}	8.35601×10^{-5}	5.44503×10^{-5}
(0.5, 0.1)	4.84030×10^{-6}	2.42869×10^{-5}	6.49632×10^{-6}	4.84030×10^{-6}	5.13642×10^{-5}	6.49632×10^{-6}
(0.9, 0.1)	3.41000×10^{-8}	8.39751×10^{-6}	5.21864×10^{-8}	3.41000×10^{-8}	7.03298×10^{-6}	5.21864×10^{-8}
(0.3, 0.3)	5.91368×10^{-5}	8.25331×10^{-5}	5.55844×10^{-5}	5.91368×10^{-5}	6.15201×10^{-5}	5.55844×10^{-5}
(0.7, 0.3)	4.84030×10^{-6}	3.43163×10^{-5}	6.52640×10^{-6}	4.84030×10^{-6}	5.41000×10^{-5}	6.52640×10^{-6}
(0.1, 0.5)	1.64290×10^{-6}	5.62014×10^{-5}	2.71889×10^{-7}	1.64290×10^{-6}	7.35192×10^{-5}	2.71889×10^{-7}
(0.5, 0.5)	5.91368×10^{-5}	7.32522×10^{-5}	5.55844×10^{-5}	5.91368×10^{-5}	8.51040×10^{-5}	5.55844×10^{-5}

The numerical solutions of $u(x, y, t), v(x, y, t)$ by the present method are shown in Figure 2. In conclusion, Example 2 shows that the present method is easy to implement, accurate, and rather time saving.

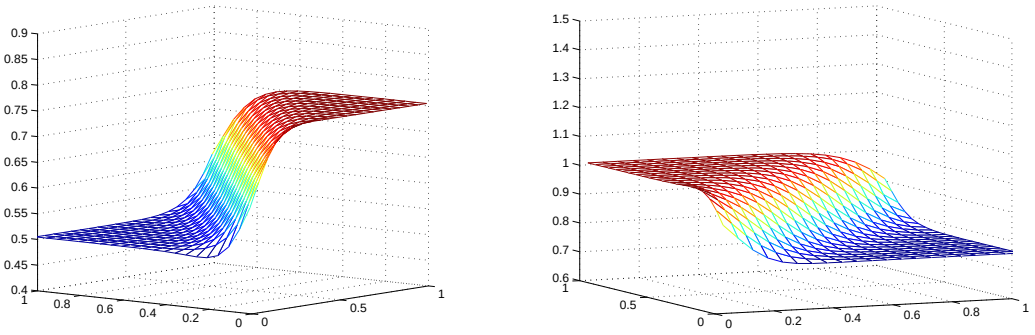


Figure 2. Example 2: The numerical solutions of $u(x, y, t)$, and $v(x, y, t)$ by the present method with $\tau = 10^{-3}$, $h_1 = h_2 = 0.05$ at $t = 0.1$

Example 3. *In this example, we will show that the proposed scheme is also applicable to other high-dimensional PDEs with shock waves or even soliton waves. We take the typical soliton equation Sine–Gordon equation as the example:*

$$\begin{aligned} u_t &= v, \\ v_t &= u_{xx} + u_{yy} - sinu, \end{aligned} \tag{18}$$

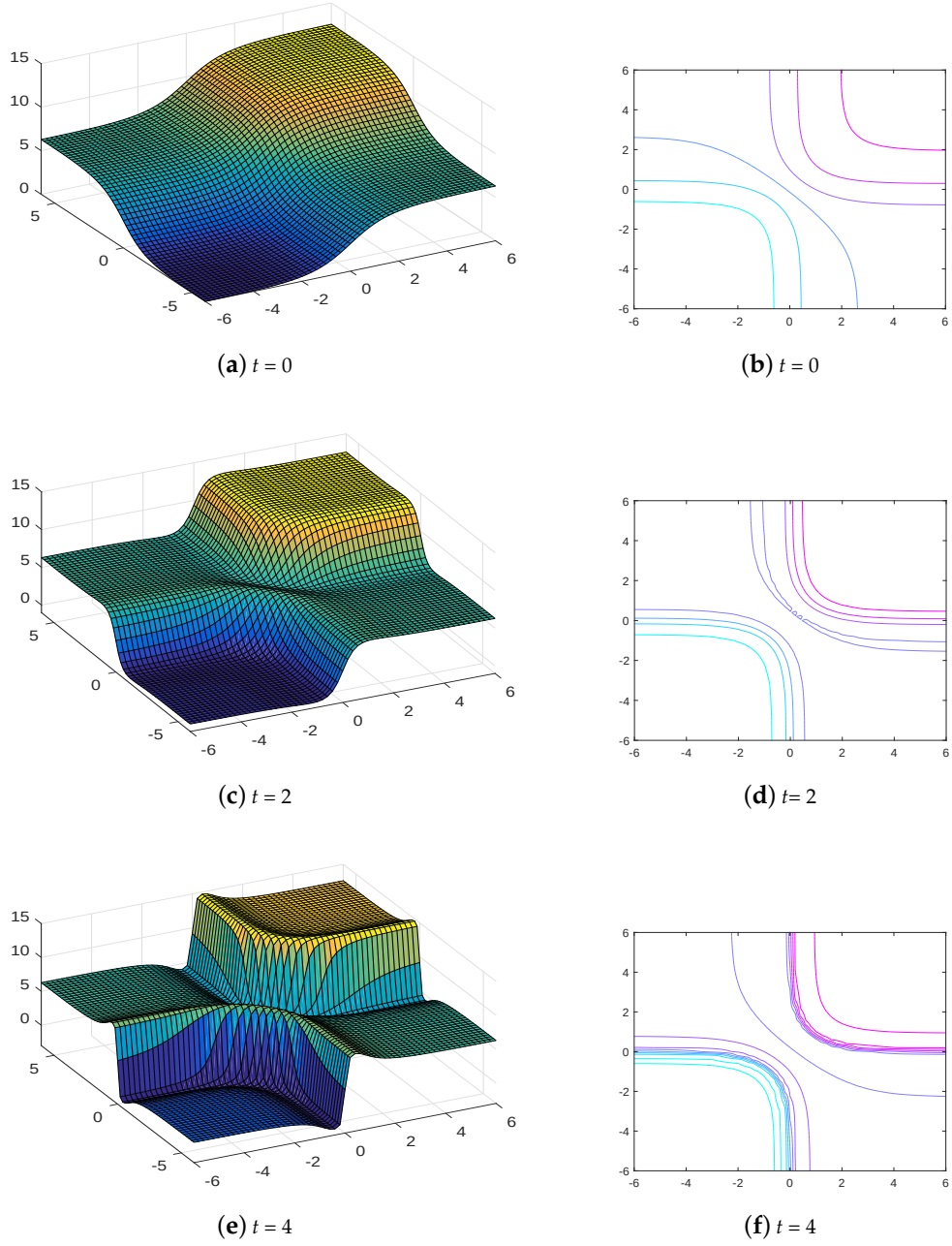
The initial conditions are:

$$\begin{aligned} u(x, y, 0) &= 4arctan(exp(x)) + 4arctan(exp(y)), \ -6 \le x, y \le 6, \\ v(x, y, 0) &= 0, \end{aligned}$$

and the boundary conditions are:

$$\begin{aligned} u_x &= 0, \ for\ x = -6, 6, -6 \le y \le 6, \\ u_y &= 0, \ for\ y = -6, 6, -6 \le x \le 6. \end{aligned}$$

The example depicts the superposition of two orthogonal line solitons, which is also used by many researchers to verify their schemes [4,29]. The numerical computations are simulated using equi-distributed knots, with a space width $h_1 = h_2 = 0.2$ and time steps $\tau = 0.01$. In this example, the parameter c is chosen as $c = 0.0075$. Figure 3 shows that our methods could simulate the solitons rather exactly. That is to say, our scheme can solve a wild range of high-dimensional PDEs efficiently.



(**a**) $t = 0$ (**b**) $t = 0$

(**c**) $t = 2$ (**d**) $t = 2$

(**e**) $t = 4$ (**f**) $t = 4$

Figure 3. Example 3: The numerical solutions and contours by the present method with $\tau = 10^{-2}$, $h_1 = h_2 = 0.2$ at $t = 0, 2, 4$.

5. Conclusions

In this paper, we present a numerical scheme for 2-D Burgers' equation by bivariate MQ quasi-interpolation. The error estimations of the algorithm are also provided. Compared with the previous method (such as Zhu et al. [8]), the proposed method owns similar accuracy, while using far less computational effort. The present algorithm owns the following advantages: Firstly, it is very simple and easy to be implemented by the engineers in the applications. Secondly, it is of satisfactory accuracy and efficient for the time dependent PDEs with shock waves. Thirdly, it uses less computational effort, because it need not to solve any large-scale linear system. The method is also applicable to other high-dimensional time dependent PDEs [22,30], such as the Burgers-Fisher, the Klein–Gordon, and so forth.

Remark 2. *Since the quasi-interpolation method have no requirement of the knots' topology structure and allows flexible knots moving, there are many works could be done to improve the efficiency and accuracy of the proposed algorithm with similar computational effort. One method is to introduce the moving knots strategy into the algorithm and let the nodes concentrated in the areas with shockwave, such as [31]. Hence better approximation accuracy could be achieved with the same computational effort. Another method is using the symmetric quasi-interpolation to construct the algorithm that preserve the energy or some important property of the original PDE, e.g., [12,22]. In this way, one could simulate the PDE for a longer time.*

Author Contributions: Software and formal analysis, S.Z.; methodology, C.Z.; writing, Q.G.

Funding: This research was funded by the National Natural Science Foundation of China (Nos. 11671068, 11601064) and Educational Commission of Liaoning Province (2017JYT07).

Acknowledgments: The authors wish to express our great gratitude to the referees for their valuable comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PDE Partial differential equation
MQ Multi-quadric

References

1. Basto, M.; Semiao, V.; Calheiros, F. Dynamics and synchronization of numerical solutions of the Burgers equation. *J. Comput. Appl. Math.* **2009**, *231*, 793–806. [CrossRef]
2. Hashimoto, I.; Matsumura, A. Asymptotic behavior toward nonlinear waves for radially symmetric solutions of the multi-dimensional Burgers equation. *J. Differ. Equ.* **2019**, *266*, 2805–2829. [CrossRef]
3. Cao, Y.; Malomed, B.A.; He, J. Two (2+1)-dimensional integrable nonlocal nonlinear Schrodinger equations: Breather, rational and semi-rational solutions. *Chaos Solitons Fractals* **2018**, *114*, 99–107. [CrossRef]
4. Hosseini, K.; Mayeli, P. ; Kumar, D. New exact solutions of the coupled sine-Gordon equations in nonlinear optics using the modified Kudryashov method. *J. Mod. Opt.* **2018**, *65*, 361–364. [CrossRef]
5. Wang, B.; Wu, X. The formulation and analysis of energy-preserving schemes for solving high-dimensional nonlinear Klein-Gordon equations. *IMA J. Numer. Anal.* **2018**. [CrossRef]
6. Fletcher. C.A.J. Generating exact solutions of the two-dimensional Burgers' equations. *Int. J. Numer. Meth. Fluids.* **1983**, *3*, 213–216. [CrossRef]
7. Bahadir, A.R. A fully implicit finite-difference scheme for two-dimensional Burgers' equations. *Appl. Math. Comput.* **2003**, *137*, 131–137. [CrossRef]

8. Zhu, H.Q.; Shu, H.Z.; Ding, M.Y. Numerical solutions of two-dimensional Burgers-equations by discrete Adomian decomposition method. *Appl. Math. Comput.* **2010**, *60*, 840–848. [CrossRef]
9. Yu, R.G.; Zhu, C.G.; Hou, M.M. Quasi-Interpolation Operators for Bivariate Quintic Spline Spaces and Their Applications. *Math. Comput. Appl.* **2017**, *22*, 10. [CrossRef]
10. Liu, G.R. *Meshfree Methods: Moving Beyond the Finite Element Method*; CRC Ppress: Boca Raton, FL, USA, 2009.
11. Gao, Q.; Wu, Z.; Zhang, S. Adaptive moving knots meshless method for simulating time dependent partial differential equations. *Eng. Anal. Bound. Elem.* **2018**, *96*, 115–122. [CrossRef]
12. Sun, Z.; Gao, W. An energy-momentum conserving scheme for Hamiltonian wave equation based on multiquadric trigonometric quasi-interpolation. *Appl. Math. Model.* **2018**, *57*, 179–191. [CrossRef]
13. Wang, B.; Iserles, A.; Wu, X. Arbitrary-order trigonometric Fourier collocation methods for multi-frequency oscillatory systems. *Found. Comput. Math.* **2016**, *16*, 151–181. [CrossRef]
14. Mittal, R.C.; Tripathi, A. Numerical solutions of two-dimensional Burgers-equations using modified Bi-cubic B-spline finite elements. *Engin. Comput.* **2015**, *32*, 1275–1306. [CrossRef]
15. Ma, L.; Wu, Z. Stability of multiquadric quasi-interpolation to approximate high order derivatives. *Sci. China Math.* **2010**, *53*, 985–992. [CrossRef]
16. Hardy, R.L. Theory and applications of the multiquadric-biharmonic method. *Appl. Math. Comput.* **1990**, *19*, 163–208. [CrossRef]
17. Beatson, R.K.; Powell, M.J.D. Univariate multiquadric approximation: quasi-interpolation to scattered data. *Constr. Approx.* **1992**, *8*, 275–288. [CrossRef]
18. Wu, Z.M.; Schaback, R. Shape preserving properties and convergence of univariate multiquadric quasi-interpolation. *Acta Math. Appl. Sin.* **1994**, *10*, 441–446. [CrossRef]
19. Ma, L.M.; Wu, Z.M. Approximation to the k-th derivatives by multiquadric quasi-interpolation method. *J. Comput. Appl. Math.* **2009**, *231*, 925–932. [CrossRef]
20. Hon, Y.C.; Wu, Z.M. A quasi-interpolation method for solving stiff ordinary differential equations. *Int. J. Numer. Methods Eng.* **2000**, *48*, 1187–1197. [CrossRef]
21. Gao, W.; Wu, Z. Solving time-dependent differential equations by multiquadric trigono- metric quasi-interpolation. *Appl. Math. Comput.* **2015**, *253*, 377–386.
22. Gao, Q.; Zhang, S.; Zhang, J. Adaptive moving knots meshless method for Degasperis-Procesi equation with conservation laws. *Appl. Numer. Math.* **2019**, *142*, 90–101. [CrossRef]
23. Ozdemir, N.; Secer, A.; Bayram, M. The Gegenbauer Wavelets-Based Computational Methods for the Coupled System of Burgers-Equations with Time-Fractional Derivative. *Mathematics* **2019**, *7*, 486. [CrossRef]
24. Ling, L. Multivariate quasi-interpolation schemes for dimension-splitting multiquadric. *Appl. Math. Comput.* **2005**, *161*, 195–209. [CrossRef]
25. Feng, R.Z.; Zhou, X. A multivariate multiquadric quasi-interpolation with quadric reproduction. *J. Comput. Math.* **2012**, *30*, 311–323.
26. Wu, R.F.; Wu, T.R.; Li, H.L. A family of multivariate multiquadric quasi-interpolation operators with higher degree polynomial reproduction. *J. Comput. Appl. Math.* **2015**, *274*, 88–108. [CrossRef]
27. Wu, Z.M.; Xiong, Z. Multivariate quasi-interpolation in $L_p(\mathbb{R}^d)$ with radial basis functions for scattered data. *Int. J. Comput. Math.* **2010**, *87*, 583–590. [CrossRef]
28. Gao, W.; Wu, Z. Constructing radial kernels with higher-order generalized Strang-Fix conditions. *Adv. Comput. Math.* **2017**, *43*, 1355–1375. [CrossRef]
29. Ram, J.; Pandit, S.; Mittal, R.C. Numerical simulation of two-dimensional sine-Gordon solitons by differential quadrature method. *Comput. Phys. Commu.* **2012**, *183*, 600–616.

30. Levesley, J.; Sun, X. Approximating probability measures on manifolds via radial basis functions. In *Approximation Algorithms for Complex Systems;* Springer: Berlin/Heidelberg, Germany, 2011.
31. Brown, D.; Ling, L.; Kansa, E.; Levesley, J. On approximate cardinal preconditioning methods for solving PDEs with radial basis functions. *Eng. Anal. Bound. Elem.* **2005**, *29*, 343–353. [CrossRef]