On the meshless quasi-interpolation methods for solving 2D sine-Gordon equations

Li Shanshan[1], Yong DUAN[1], and Libing Bai[1]

[1]University of Electronic Science and Technology of China

March 07, 2024

Abstract

This paper is devoted to applying a numerical method for solution of the 2D sine-Gordon equation. The bivariate multiple quadratic quasi-interpolation (MQQI) method is adopted to simulate this equations, which the first order spatial derivative is approximated by MQQI, the second spatial and time derivative are approximated by forward difference. One of the merit of this scheme is its simple structure and easy implementation. In the meanwhile, we present truncation and total error of this scheme, high accuracy and efficiency of the method are verified by numerical experiments. In addition, the optimal value of parameters are investigated in this article based on Luh [10, 11].

On the meshless quasi-interpolation methods for solving 2D sine-Gordon equations

Shan shan Lii [*] · Yong Duan [*] · Li bing Bai [†]

Abstract: This paper is devoted to applying a numerical method for solution of the 2D sine-Gordon equation. The bivariate multiple quadratic quasi-interpolation (MQQI) method is adopted to simulate this equations, which the first order spatial derivative is approximated by MQQI, the second spatial and time derivative are approximated by forward difference. One of the merit of this scheme is its simple structure and easy implementation. In the meanwhile, we present truncation and total error of this scheme, high accuracy and efficiency of the method are verified by numerical experiments. In addition, the optimal value of parameters are investigated in this article based on Luh [4, 5].

Keywords: 2D sine-Gordon equations · MQ function · Radial basis functions (RBFs) · Meshless quasi-interpolation method

Mathematics Subject Classification (2010): 65D05 · 65D15 · 65F10· 65M12 · 78M25

1 Introduction

A famous reads: "with a suitable initial boundary conditions, called solition equation, also, sine-Gordon equations". It is an important class of nonlinear soliton equations. As mentioned in [7], solitons essentially represent special wave-like solutions of nonlinear dynamic equations. In fact, these waves advance in the medium without any deformation owing to dispersion. Moreover, it will not deform after interacting with other solitons. Solitons have been proved to play a key role in the theory of nonlinear differential equations. Soliton solutions of various equations have been found, including the Korteweg-de Vries equation, the nonlinear Schrödinger equation and the sine-Gordon equation [12].

This work was partly supported by NSFC 12071062, NSAFU 2030205 and Science Strength Promotion Programme of of UESTC.

✉ Yong Duan
Corresponding author: duanyong@uestc.edu.cn
✉ Shan shan Lii
slsaccount@163.com
✉ Libing Bai
libing.bai@uestc.edu.cn

[*]School of Mathematical Science, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, P. R. China.
[†]School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, P. R. China

This paper is devoted to numerical computation of two dimensional time-dependent nonlinear sine-Gordon equation. The two dimensional sine-Gordon equation is given by

$$\frac{\partial^2 u}{\partial^2 t} + \beta \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} + \phi(x,y) sin(u), \tag{1.1}$$

with $u(x,y,t)$ in the region $\Omega = \{(x,y), L_x^0 < x < L_x^1, L_y^0 < y < L_y^1\}$ for $t > 0$ and u is a sufficiently differentiable function.

Eq. (1.1) is defined in some continuous domain with suitable initial and Neumann's boundary conditions. Here, initial conditions associated with Eq. (1.1) will be assumed to be of the form

$$u(x,y,0) = f(x,y), (x,y) \in \Omega, \tag{1.2}$$

with initial velocity

$$\frac{\partial u}{\partial t}(x,y,t) = g(x,y), (x,y) \in \Omega. \tag{1.3}$$

And the boundary conditions are supposed to be in the following form

$$\frac{\partial u}{\partial x}(x,y,t) = p(x,y,t), (x,y) \in \partial\Omega,\ t > 0, \tag{1.4}$$

and

$$\frac{\partial u}{\partial y}(x,y,t) = q(x,y,t), (x,y) \in \partial\Omega,\ t > 0, \tag{1.5}$$

where $p(x,y,t)$ and $q(x,y,t)$ are normal gradients along the boundary of the region Ω.

In Eq. (1.1), the parameter β is the so-called dissipative term, which is supposed to be a real number with $\beta \geqslant 0$. When $\beta = 0$, Eq. (1.1) calls the undamped sine-Gordon equation with two space variables, while $\beta > 0$, it calls the damped one. The function $\phi(x,y)$ may be interpreted as the Josephson current density, while in Eqs. $(1.2) - (1.3)$, the functions $f(x,y)$ and $g(x,y)$ represent wave modes or kinks and velocity, respectively, as said in [8]. In experiments the kink profiles will be considered known from the relevant one-dimensional problem. They will move on straight lines or closed curves in the $xy-$ plane. These curves will be called line and ring soliton waves respectively.

Radial basis function (RBF) methods have been widely developed since 1988, and breakthroughs had been made in properties of the coefficient matrices, establishment of simplified algorithm, and selection of the best interpolation points and so on [1]. Although most of the work to data on RBFs is related to the approximation of scattered data and interpolation theory, there has been more and more interested in its application in solving partial differential equations (PDEs), see [3]. This method approximates the global solution of PDE by the translation of radial basis function. Because it is a real meshless method and independent of space dimension, it can be easily extended to solve high-dimensional problems. In addition, due to the smoothness of RBF, it can be easily applied to solve higher order differential equations. The advantages of RBF based meshless method also include that there is no need to explicitly connect the volumes, surfaces and nodes compared with the method based on conformal hexahedron or tetrahedron mesh. Flexibility in node distribution allows shape preserving and multi-scale modeling.

Meanwhile, there are some shortcomings in the numerical calculation. It is found that the conditions of the generated linear system are too harsh to guarantee the accuracy of the numerical solution, especially for large-scale problems. With the intensive study of RBFs interpolation theory, a new quasi-interpolation (QI) method has attracted people's attention. This new interpolation method does not

need to solve large scale linear systems, but maintains the optimal efficiency in terms of energy norm and can obtain the solution directly, mentioned in [3]. One of the advantages of QI is that it is related to the smoothness, simplicity, good shape and exponential decay of the generating function. Therefore, compared with other meshless techniques (such as RBF interpolation), this method can reduce the computation time and even approximate functions in high-dimensions. QI has been successfully applied to scattered data approximation and interpolation, numerical solution and quadrature of PDEs, see [14].

Some quasi-interpolations can even keep the positive, monotone and convexity of approximations, such as spline quasi-interpolation [7, 13], multiple-quadratic quasi-interpolation [3, 10]. Because of those excellent properties, more and more scholars have devoted themselves to the study of multiple quadratic quasi-interpolation (MQQI), following Refs. [3, 6, 8, 9, 10, 13]. To improve the approximation behaviors near the boundary, Beaston and Powell [15] put forward three univariate MQQI to deal with scattered data, that is ℓ_A, ℓ_B, ℓ_C. Wu and Schaback [16] constructed a new MQQI scheme, which is named ℓ_D and improved Beatson's and Powell's results. Meanwhile, they also proved convergence rates and shape preserving properties of these MQQI schemes. Ma and Wu [17] studied the approximate properties of k-th derivatives by MQQI. What's more, a new MQQI scheme designed by Ma and Wu [18], which was successfully applied to solve sine-Gordon equation, stiff ordinary differential equation and shock wave. By introducing multiple quadratic trigonometric kernel, Gao and Wu [2] extended MQQI to single variable periodic data with irregular center spacing.

The MQQI method is suitable for the initial conditions of the scattered data and any boundary conditions so that it can approximate almost all functions. This paper takes the lead in utilizing the multivariate meshless MQQI method for two-dimensional sine-Gordon equation systematically. In this proposed numerical scheme, the first derivative of space is approximated by the derivative of MQQI, the second-order spatial and temporal derivatives are approximated by forward difference. Compared with Luh's aritcle [4, 5], this paper investigate the optimal selection of parameters friendly. The experimental results demonstrate that the errors of root mean squart (RMS) and L_∞ are 1e-05 and 1e-04 respectively, within a short time, which is much better than the RBF method used in 2008 [8].

This article is arranged as follows. Section 2 introduce the univariate MQQI firstly, on this basis, the bivariate MQQI is developed. In section 3, the numerical scheme of bivariate MQQI is introduced and the corresponding programs are appeared. In section 4, we use the algorithm to approximate five kinds of two-dimensional sine-Gordon equations and their partial derivatives to fit the equations. The results present that the method has high accuracy and can simulate the soliton equations accurately, in a short time. The choice of the parameters friendly approaching are also showed in section 5. Some conclusion are made at the end of this paper.

2 Bivariate multiple quadratic quasi-interpolation

In this section, the univariate MQQI is introduced firstly. Based on this, we present the bivariate MQQI and show its approximation order to the function and its derivatives by using tensor product technique ($\oplus$), see [2].

Suppose $f(x)$ has values on a set of different nodes x_j, given a group of scattered points $\{(x_j, f(x_j)\}$, then the unknown function $f(x)$ can be constructed by

$$(Qf)(x) = \sum_j f(x_j)\psi_j(x), \tag{2.1}$$

where,

$$\psi_j(x) = \frac{\phi_{j+1}(x) - \phi_j(x)}{2(x_{j+1} - x_j)} - \frac{\phi_j(x) - \phi_{j-1}(x)}{2(x_j - x_{j-1})}, \tag{2.2}$$

$\phi(x) = \sqrt{c_1^2 + x^2}$ is the MQ function, the transform of MQ function is $\phi_j(x) = \sqrt{c_1^2 + (x - x_j)^2}$. Where, c_1 is a positive constant (also called shape parameter). As c_1 goes to zero, $\phi_j(x)$ will converge to $|x - x_j|$. Therefore, the univariate MQQI could be used as piecewise linear interpolation, but possesses smoothness property.

The n-th order derivative of the MQQI can be directly calculated:

$$(Qf)^{(n)}(x) = \sum_j f(x_j)\psi_j^{(n)}(x). \tag{2.3}$$

It is also a better approximation of the derivative of an unknown function. The approximate error of $Qf(x)$ to the original function $f(x)$ and the k-th derivative were given in [17], by defining $h_1 = max_j(x_{j+1} - x_j)$.

In this paper, we presents a bivariate MQQI using tensor product technique ($\oplus$). Specifically, given the date (x_i, y_j, f_{ij}) and $f_{ij} = f(x_i, y_j)$, the bivariate MQ quasi-interpolation is defined as:

$$(Qf)(x, y) = \sum_i \sum_j f_{ij}\psi_j(x)\psi_j(y), \tag{2.4}$$

where, $\psi_i(x)$ are described in (2.2) and $\psi_j(y)$ are represented as follows:

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

$$\phi_j(y) = \sqrt{c_2^2 + (y - y_j)^2}.$$

c_2 is a positive constant. In this paper, we only discuss the case of $c_1 = c_2$. Step size in y direction is defined as $h_2 = max_j(y_{j+1} - y_j)$.

3 Solving 2D sine-Gordon equation with MQQI scheme

Here, a numerical scheme is applied for solving two-dimensional sine-Gordon equations (1.1) – (1.5) based on bivariate MQQI mentioned in Section 2.

3.1 Numerical scheme

At the points $(x_i, y_j, t_k) = (ih_1, jh_2, k\tau)$, the approximation of $u(x, y, t)$ is replaced by u_{ij}^k. Where h_1 and h_2 are the grid sizes in x and y direction separately, τ is the time step sizes. For the sake of recall, we define $u_{ij}^k \approx u(x_i, y_j, t_k)$, then the first derivative is given by $(u_x)_{ij}^k \approx u_x(x_i, y_j, t_k)$, other definitions of derivatives are similar to this.

The first partial derivative of u with respect to spatial variables is approximated by MQQI as following,

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

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

and the second derivative is approximated by the forward difference of the first derivative

$$(u_{xx})_{ij}^k = ((u_x)_{i+1,j}^k - (u_x)_{i,j}^k)/h_1,$$

$$(u_{yy})_{ij}^k = ((u_y)_{i,j+1}^k - (u_y)_{i,j}^k)/h_2.$$

In this case, the boundary conditions belong to the second kind (also called Riemann boundary conditions). Although there are some methods to dispose of boundary conditions, following Refs. [7, 19], they generally require complex calculation or are beyond comprehension for non mathematical people. On the right boundary, we let the value of the second derivative to the nearest value hear. That is to say, $(u_{xx})_{ij}^k = (u_{xx})_{i-1,j}^k$ and $(u_{yy})_{ij}^k = (u_{yy})_{i,j-1}^k$. In the sequel, we certificate the effectiveness and feasibility of this method. Now, we infer the interpolation expression of $u(x, y, t)$.

Discretization of time derivatives by forward difference as follows,

$$(u_t)_{ij}^k = \frac{u_{ij}^{k+1} - u_{ij}^k}{\tau},$$

$$(u_{tt})_{ij}^k = \frac{(u_t)_{ij}^{k+1} - (u_t)_{ij}^k}{\tau} = \frac{u_{ij}^{k+2} - 2u_{ij}^{k+1} + u_{ij}^k}{(\tau)^2},$$

then, put it in (1.1), obtain

$$\frac{u_{ij}^{k+2} - 2u_{ij}^{k+1} + u_{ij}^k}{(\tau)^2} + \beta\frac{u_{ij}^{k+1} - u_{ij}^k}{\tau} = (u_{xx})_{ij}^k + (u_{yy})_{ij}^k + \phi sin((u_t)_{ij}^k),$$

after simplification, the representation of $u(x, y, t)$ is gained

$$u_{ij}^{k+2} = (2 - \beta\tau)u_{ij}^{k+1} + (\beta\tau - 1)u_{ij}^k + \tau^2\{(u_{xx})_{ij}^k + (u_{yy})_{ij}^k + \phi sin((u_t)_{ij}^k)\},$$

where, τ is time step.

3.2 Implementation of the numerical method

Define some global variables X Y M N T τ β t h_1 h_2 c_1 c_2. Let $h_1 = X/M$, $h_2 = Y/N$, $\tau = t/T$. Let X is the matrix of $M \times 1$, Y is the matrix of $N \times 1$. t is the matrix of $T \times 1$. M and N are the number of nodes obtained in space. T is the running times. τ is the time interval. h_1 and h_2 are the spatial interval in x and y directions respectively. c_1 and c_2 are the shape parameters ($c_i > 0, i = 1, 2$). Note that f and g is the initial conditions defined in section 1. ϕ' is the derivative of ϕ. ψ and ψ' are shows in section 2.

---

1. Let $u(X(i), Y(j), 1) = f(X(i), Y(j))$, and $u_t(X(i), Y(j), 1) = g(X(i), Y(j))$, for i = 1, 2, ..., M, j = 1, 2, ..., N.

---

2. Let $u(X(i), Y(j), 2) = u(X(i), Y(j), 1) + \tau * u_t(X(i), Y(j), 1)$, ($i$ = 1, 2, ..., M, j = 1, 2, ..., N).

---

3. Define $\psi(X(i), Y(j)) = (\phi(X(i), Y(j) + h, c) - \phi(X(i), Y(j), c))/(2h) - (\phi(X(i), Y(j), c) - \phi(X(i), Y(j) - h, c))/(2h)$, then let $\psi = \psi(:, 2 : end)$, for i = 1, ..., M, j = 2, ..., $N - 1$,
where, matrix ψ removes the first column.

---

4. Define $\psi'(X(i),Y(j)) = (\phi'(X(i),Y(j)+h,c) - \phi'(X(i),Y(j),c))/(2h) - (\phi'(X(i),Y(j),c) - \phi'(X(i),Y(j)-h,c))/(2h)$, let $\psi' = \psi'(:,2:end)$, for i = 1, ..., M, j = 2, ..., $N-1$.

---

5. Define $\psi = phi(X,Y,h)$, $\psi' = dex(X,Y,h)$, where, ψ and ψ' are showed in step 3 and 4 respectively.

---

6. Let $p = dex(X(i),X(j),h_1)$, $q = phi(Y(i),Y(j),h_2)$, $q1 = dex(Y(i),Y(j),h_1)$,
$p1 = phi(X(i),X(j),h_2)$, $s_1(X(i),Y(j)) = sum(p(X(i),:)) * sum(q(Y(j),:))$,
$s_2(X(i),Y(j)) = sum(p1(X(i),:)) * sum(q1(Y(j),:))$, for i = 1, ..., M, j = 1, ..., N.

---

7. Solving u_x and u_y, (i = 1, ..., M, j = 1, ..., N, k = 1, ..., T-2),
if $i < M$ and $j \le N$, $u_x(X(i+1),Y(j),t(k)) = u(X(i+1),Y(j),t(k)) * s_1(X(i+1),Y(j))$,
if $i = M$ and $j \le N$, $u_x(X(i),Y(j),t(k)) = u(X(i),Y(j),t(k)) * s_1(X(i),Y(j))$,
if $i \le M$ and $j < N$, $u_y(X(i),Y(j+1),t(k)) = u(X(i),Y(j+1),t(k)) * s_2(X(i),Y(j+1))$,
if $i \le M$ and $j = N$, $u_y(X(i),Y(j),t(k)) = u(X(i),Y(j),t(k)) * s_2(X(i),Y(j))$.

---

8. Solving u_{xx} and u_{yy}, (i = 1, ..., M, j = 1, ..., N, k = 1, ..., T-2),
if $i < M, j \le N$, $u_{xx}(X(i),Y(j),t(k)) = (u_x(X(i+1),Y(j),t(k)) - u_x(X(i),Y(j),t(k))/h_1$,
if $i = M, j \le N$, $u_{xx}(X(i),Y(j),t(k)) = u_{xx}(X(i-1),Y(j),t(k))$,
if $i \le M, j < N$, $u_{yy}(X(i),Y(j),t(k)) = (u_y(X(i),Y(j+1),t(k)) - u_y(X(i),Y(j),t(k))/h_2$,
if $i \le M, j = N$, $u_{yy}(X(i),Y(j),t(k)) = u_{yy}(X(i),Y(j-1),t(k))$.

---

9. Let $u(X(i),Y(j),t(k+2)) = (2-\beta\tau)u(X(i),Y(j),t(k+1)) - (1-\beta\tau)u(X(i),Y(j),t(k))$
$+ \tau^2\{u_{xx}(X(i),Y(j),t(k)) + u_{yy}(X(i),Y(j),t(k)) - sin(u(X(i),Y(j),t(k)))\}$,
for i = 1, ..., M, j = 1, ..., N, k = 1, ..., T-2.

---

4 Numerical Examples

In this section, we chose the sine-Gordon equation to illustrate the performance of the method described in the previous section. All these experiments are executed on a computer with AMD Ryzen 7 4800H with Radeon Graphics, 2.90 GHz processor and 16.00 GB RAM (random access memory). In additon, we show that the proposed scheme is applicable to other two-dimensional partial differential equations containing shock waves and even soliton waves.

In order to verify the effectiveness of the scheme, the root mean square (RMS) and L_∞ error norms are employed. The error norms are defined as

$$L_2 = \sqrt{\frac{\sum_i \sum_j (u_{ij}^{exact} - u_{ij}^{num})^2}{(N_1+1)(N_2+1)}},$$

$$L_\infty = max_{ij}|u_{ij}^{exact} - u_{ij}^{num}|,$$

where u_{ij}^{exact} and u_{ij}^{num} are the analysis and numerical results of u at the knot (x_i, y_j), L_2 is RMS error.

4.1 2-D sine-Gordon Equations

To observe the behavior of the numerical method, it was tested on the sine-Gordon obtained for $\phi(x, y) = -1$ with initial conditions,

$$f(x, y) = 4tan^{-1} \exp(x + y), (x, y) \in \Omega,$$
$$g(x, y) = -\frac{4 \exp(x + y)}{1 + \exp(2x + 2y)}, (x, y) \in \Omega,$$

and boundary conditions

$$p(x, y, t) = \frac{4 \exp(x + y + t)}{\exp(2t) + \exp(2x + 2y)}, x = -7 \; and \; 7, -7 \leq y \leq 7, t > 0,$$

$$q(x, y, t) = \frac{4 \exp(x + y + t)}{\exp(2t) + \exp(2x + 2y)}, y = -7 \; and \; 7, -7 \leq x \leq 7, t > 0.$$

The numerical solution of this problem, in which the parameter $\beta = 0$ is given by

$$u(x, y, t) = 4tan^{-1} \exp(x + y - t), (x, y) \in \Omega, t > 0,$$

where, $\Omega = \{-7 \leq x \leq 7, -7 \leq y \leq 7\}$.

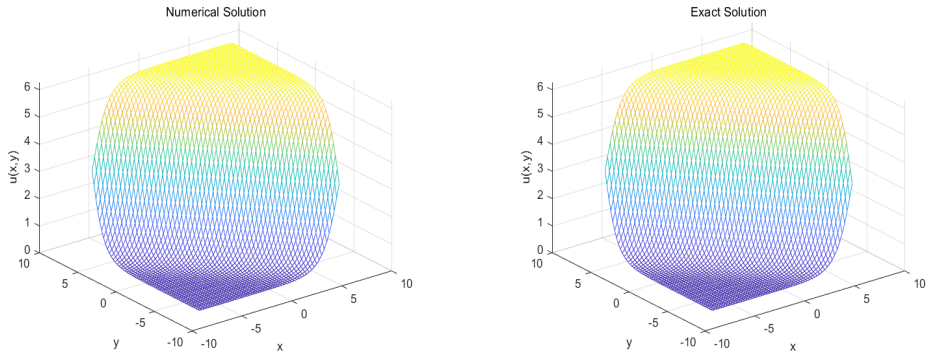


Figure 1: Estimated and analytical solutions in $t = 0.01$, with τ = 1e-04, $c = 1.7$, $h_1 = h_2 = 0.25$ for problem 4.1.

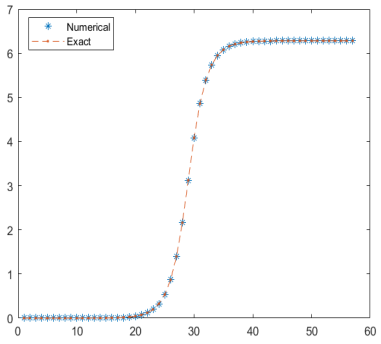


Figure 2: Analytical and estimated solutions in $t = 0.01$, with τ = 1e-04, $c = 1.7$ and $h_1 = h_2 = 0.25$ for problem 4.1.

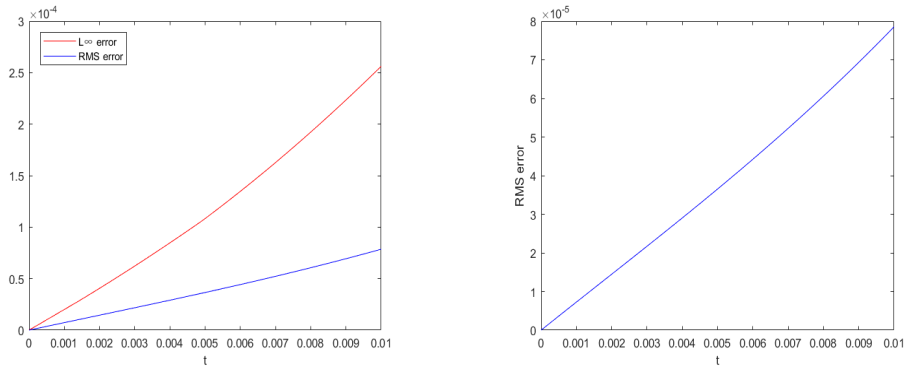


Figure 3: L_2 and L_∞ errors using MQIQ method, with τ = 1e-04, $c = 1.7$, $h_1 = h_2 = 0.25$ for problem 4.1.

The graph of estimated and analytical solutions for $t = 0.01$ is given in Fig. 1. From this picture, we can find that the image obtained by MQQI method is almost identical with the image of numerical solution. Fig. 2 presents the value of analysis and numerical solution for knots which are accruing in line $y = x$. This also directly confirms the above conclusion. Fig. 3 shows the L_2 error, L_∞ norms of error and comparison between L_2 and L_∞ norms of errors at time $t = 0.01$. We can draw a such conclusion immediately from the Figure 3, in a extraordinary time, the two kinds of error precision are excellent, the L_2 error even reaches 1e-05. This is more accurate than the traditional RBFs method proposed in reference [8], and the error accuracy in this paper is only 1e-02.

The MQ function has such excellent properties [10], it has a round and smooth waveform, not a shock wave. Therefore, MQQI technique is better than finite difference method (FDM), see [8], finite element method (FEM), see [11], boundary element method (BEM), and is better than the one obtained by the collocation solution of traditional RBFs proposed in [7].

4.2 The Typical Soliton Equation

In this example, we will show the proposed scheme which is applicable to other 2 dimensional PDEs with shock waves or even soliton waves. We take the typical soliton equation as the example

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

The initial conditions are

$$\begin{aligned} u(x, y, 0) &= 4tan^{-1}\exp(x) + 4tan^{-1}\exp(y), (x, y) \in \Omega, \\ v(x, y, 0) &= 0, (x, y) \in \Omega, \end{aligned}$$

and the boundary conditions are

$$\begin{aligned} u_x &= 0, x = -6 \; and \; 6, -6 \le y \le 6, t > 0, \\ u_y &= 0, y = -6 \; and \; 6, -6 \le x \le 6, t > 0, \end{aligned}$$

where, $\Omega = \{-6 \le x \le 6, -6 \le y \le 6\}$.

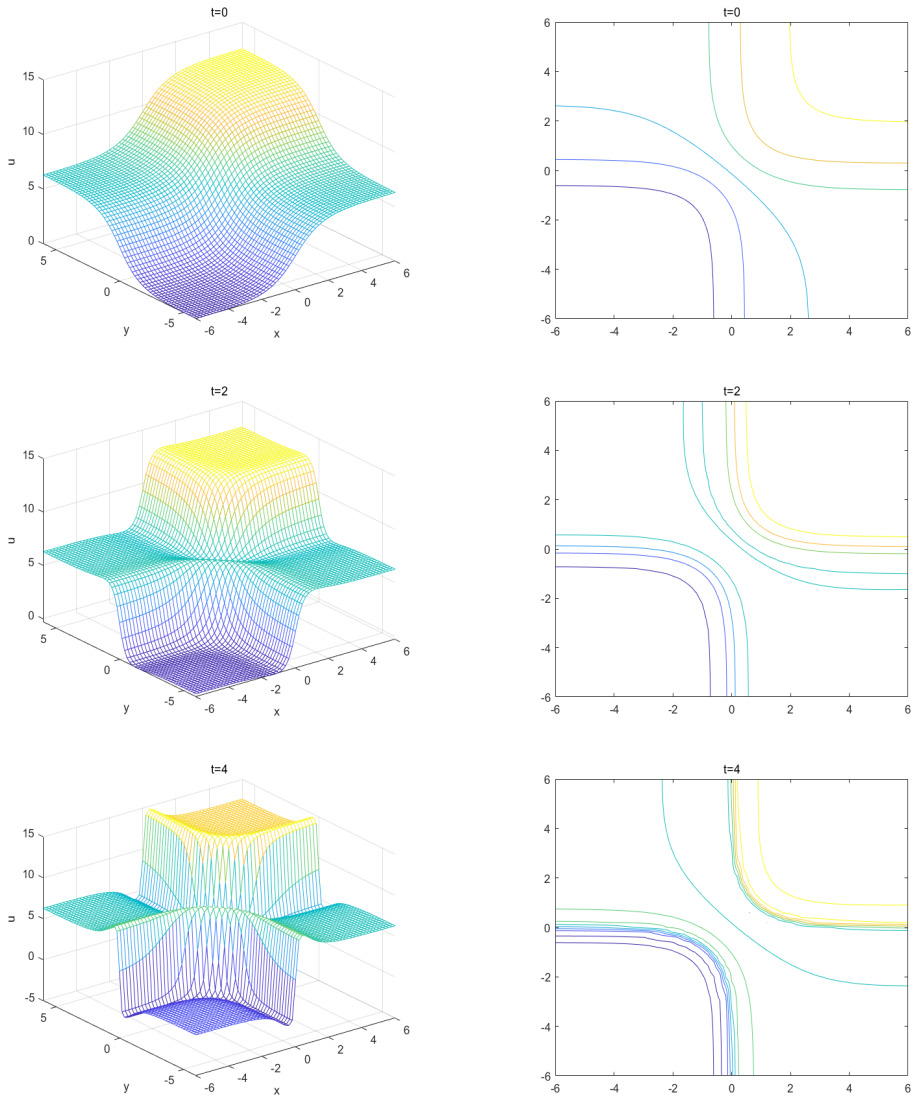


Figure 4: The numerical solutions and contours with $\tau = 0.01$, $h_1 = h_2 = 0.2$, $t = 0$, $t = 2$, and $t = 4$ for problem 4.2, in terms of $sin(u/2)$.

The example depicts the superposition of two orthogonal line silitons, the numerical results are gained with a space width $h_1 = h_2 = 0.2$ and time step $\tau = 0.01$. The images and contours of the function $sin(u/2)$ are shows in Fig. 4, at $t = 0$, $t = 2$ and $t = 4$ with shape parameter $c = 13$. Figure 4 shows that the solitons are simulated exactly by adopting MQQI scheme. That is to say, the scheme can solve a wild range of two dimensional PDEs efficiently.

4.3 Circular Ring Soliton

Circular ring solitons is obtained for $\phi(x, y) = -1$ and initial conditions

$$f(x, y) = 4tan^{-1}(\exp(3 - \sqrt{(x^2 + y^2)})), (x, y) \in \Omega,$$
$$g(x, y) = 0, (x, y) \in \Omega,$$

and the boundary conditions

$$p(x, y, t) = 0, x = -7 \; and \; 7, -7 \leq y \leq 7, t > 0,$$

$$q(x, y, t) = 0, y = -7 \; and \; 7, -7 \leq x \leq 7, t > 0.$$

where, the solution domain of the equation is $\Omega = \{-7 \leq x \leq 7, -7 \leq y \leq 7\}$.

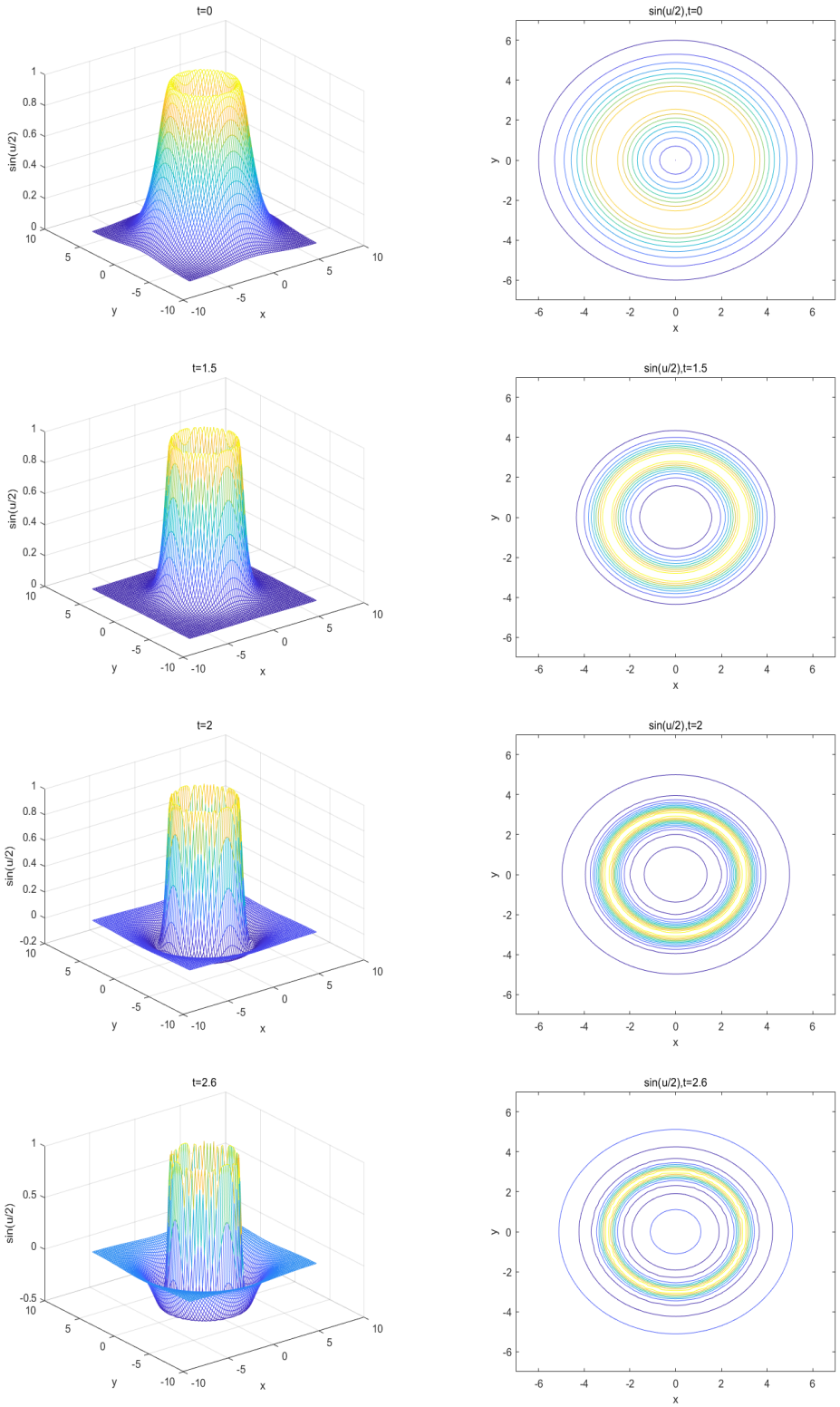


Figure 5: Numerical solutions at times $t = 0$, $t = 1.5$, $t = 2$, $t = 2.6$, with $\tau = 0.01$ and $h_1 = h_2 = 0.2$, for problem 4.3.

Figure 5 presents the solution of $t = 0$, $t = 1.5$, $t = 2$, $t = 2.6$ and contour map, in terms of $sin(u/2)$ respectively. The solution is obtained for $\beta = 0$, $\tau = 0.01$, $h_1 = h_2 = 0.2$ and $c = 1.5$. The

simulated ring solitons shrink at the initial stage, as time goes by, the oscillation and radiation begin to form and continue slowly. The fact is consistent with the results of Refs. [7, 19] and the motion state of the ring soliton can be clearly observed by contour maps.

Figure 5 only shows the oscillation of the soliton from the initial time to 2.6 seconds. This is because the error of MQQI method increases a little fast with the increase of time. Therefore, the motion of the soliton is not demonstrated later. When the space step h is reduced to 0.1, we find that the soliton oscillation can last up to 3.3 seconds. It is worth noting that as long as shape parameter c is greater than or equal to 1.5, this method can accurately simulate solitons in a short time.

4.4 Collision of Two Circular Ring Solitons

The collision between circular solitons is obtained for $\phi(x, y) = -1$ and initial conditions

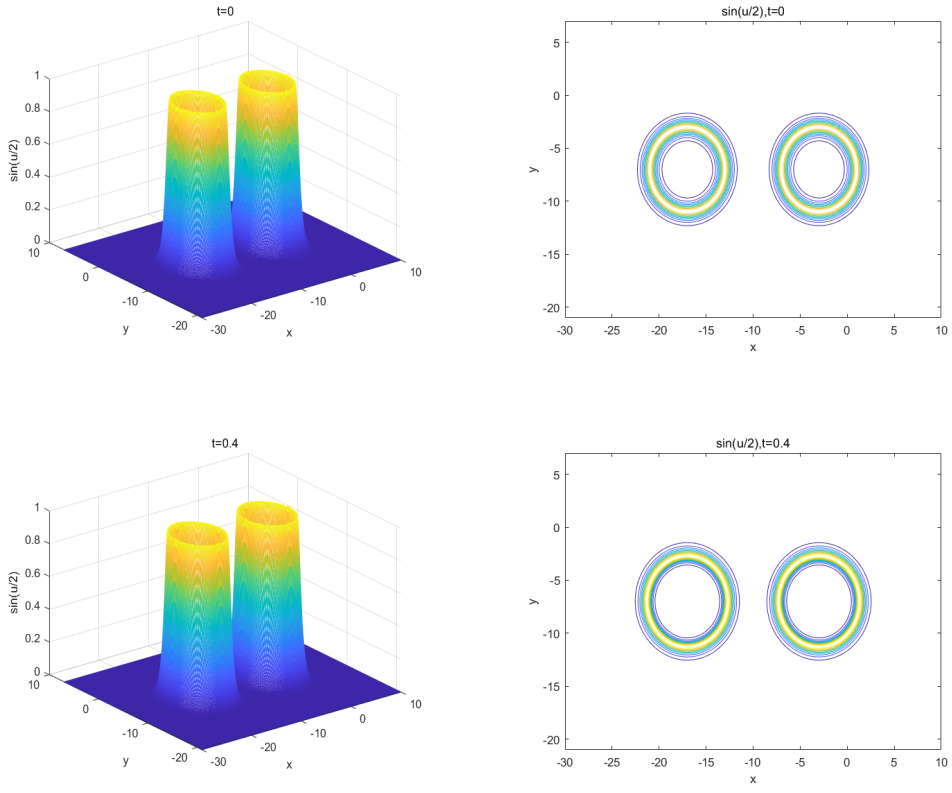
$$f(x,y) = 4tan^{-1}(\exp(\frac{4-\sqrt{(x+3)^2+(y+7)^2}}{0.436})), (x,y) \in \Omega,$$

$$g(x,y) = 4.13sech(\exp(\frac{4-\sqrt{(x+3)^2+(y+7)^2}}{0.436})), (x,y) \in \Omega,$$

and the boundary conditions

$$p(x,y) = 0, x = -30 \; and \; 10, \; -21 \leq y \leq 7,$$
$$q(x,y) = 0, y = -21 \; and \; 7, \; -30 \leq x \leq 10.$$

The image and the contour map of the equation with respect to $t = 0$, $t = 0.4$, $t = 0.8$ and $t = 2$ in region $\Omega = \{-30 \leq x \leq 10, -21 \leq y \leq 7\}$ are shown in Fig. 6, in terms of $sin(u/2)$. The numerical solution is obtained when the parameters are defined as $c = 5$, $\tau = 0.01$ and $h_1 = h_2 = 0.1$ separately.



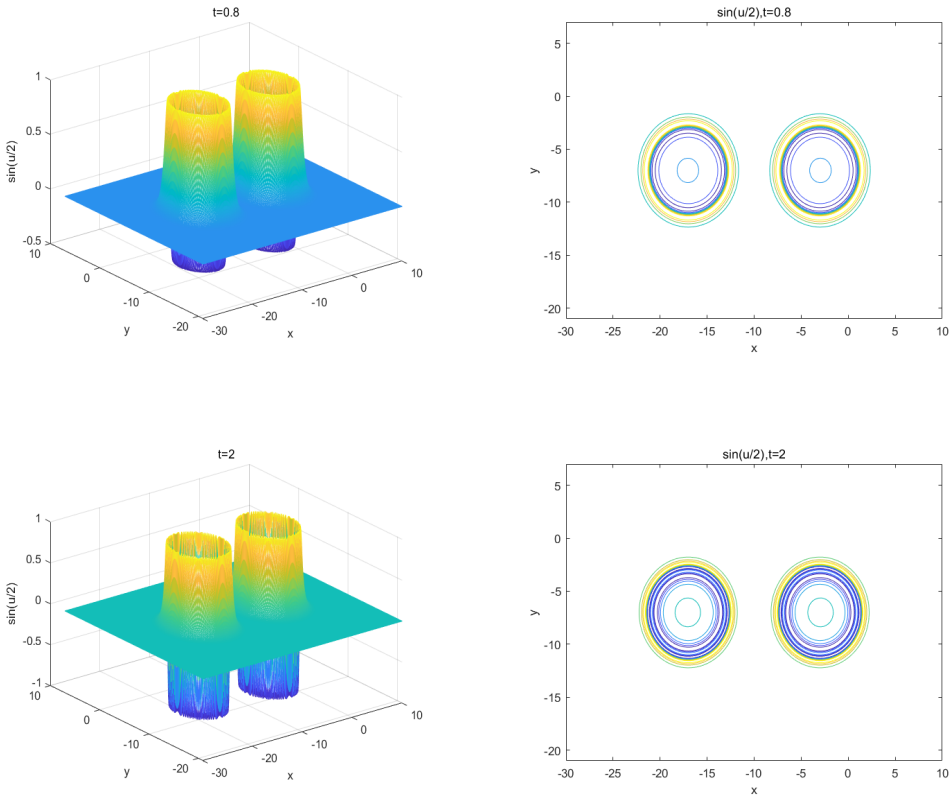


Figure 6: Numerical solutions at times $t = 0$, $t = 0.4$, $t = 0.8$, $t = 2$, with $\tau = 0.01$ and $h_1 = h_2 = 0.1$, for problem 4.4.

The solution is extended across $x = -10$ and $y = -7$ by symmetry relations. As illustrated in the Figure 6, we can survey the expansion and oscillation of the two ring solitons, and the solitons appear obvious oscillation in a short time. Contour maps can reflect the trajectory of solitons more directly. In this example, as long as shape parameter c is greater than or equal to 1, the soliton can be accurately simulated in a short time.

4.5 Collision of Four Circular Ring Solitons

A collision of four expanding circular ring solitons is is obtained for $\phi(x, y) = -1$ with initial conditions

$$f(x,y) = 4tan^{-1}(\exp(\frac{4-\sqrt{(x+3)^2+(y+3)^2}}{0.436})), (x,y) \in \Omega,$$
$$g(x,y) = \frac{4.13}{cosh(\exp(4-\sqrt{\frac{(x+3)^2+(y+3)^2}{0.436}}))}, (x,y) \in \Omega,$$

and the boundary conditions

$$p(x,y) = 0, x = -30 \; and \; 10, \; -30 \leq y \leq 10,$$
$$q(x,y) = 0, y = -30 \; and \; 10, \; -30 \leq x \leq 10.$$

Over the region $\Omega = \{-30 \leq x \leq 10, -30 \leq y \leq 10\}$ are presented in Fig. 7 for $\beta = 0$, $\tau = 0.01$ and $h_1 = h_2 = 0.1$ at $t = 0$, $t = 0.4$, $t = 0.8$ and $t = 1$, in terms of $sin(u/2)$. From the diagram, we can observe the phenomenon similar to the collision of four expanding ring solitons.

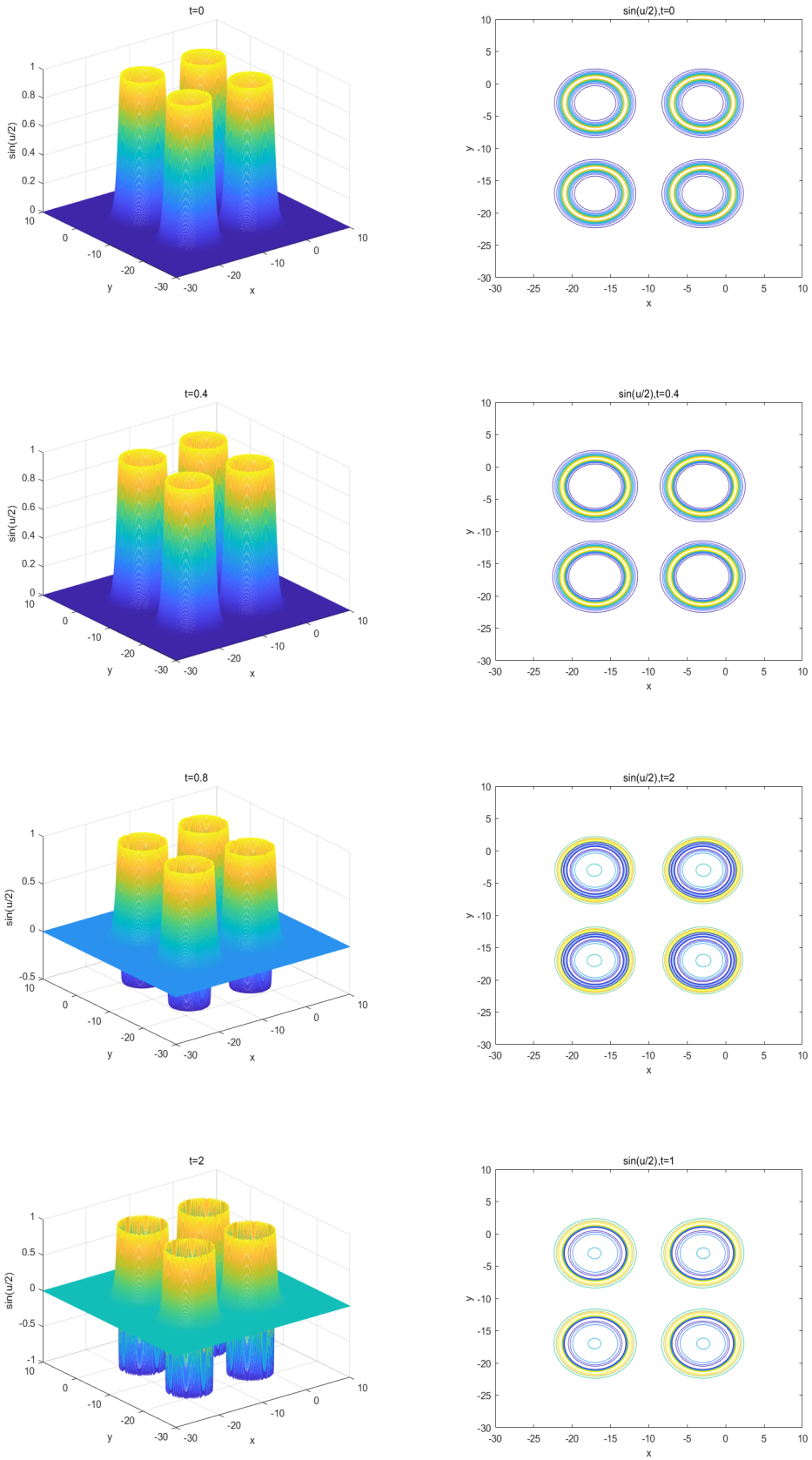


Figure 7: Numerical solutions at times $t = 0$, $t = 0.4$, $t = 0.8$ and $t = 1$, with $\tau = 0.01$ and $h_1 = h_2 = 0.1$, for problem 4.5.

Over the domain $-30 \leq x \leq 10$, $-30 \leq y \leq 10$, the solution is received in a quarter of the domain, and then symmetrically about $x = -10$ and $y = -10$ to extend the solution to the whole domain. The

solitons expansion from the initial time to 2 seconds is shown in Fig. 7, and the shape parameter c is selected as 5. In addition, it is found that the shape parameter c have little influence on the results, as long as c is greater than or equal to 1.

In the above five numerical examples, the shape parameters c are slightly different, which may be related to the fitting equation, interpolation region and spatial step. How to choose the value of shape parameter is further investigated in the sequel.

5 The choice of parameters

The sine-Gordon equation which is mentioned in section 1 is chosen to consider the value of parameters. In order to verify the effectiveness of the scheme, RMS and L_∞ error norms are used. The MQQI method in this paper takes the MQ function with shape parameter c as the radial basis function, therefore, it is very essential to find the influence of shape parameter on the superiority of fitting effect. Besides, we also discuss the influence of the selection of time step and space step on the approximation effect.

For any set $\Omega \subseteq R^2$, any set $X = \{x_1, ..., x_M\}$ and $Y = \{y_1, ..., y_N\}$ included in Ω, the fill distance is defined by

$$h_1 = \underset{x_i \in \Omega}{max}(x_{i+1} - x_i),\ i = 1, 2, ..., M-1,$$
$$h_2 = \underset{y_j \in \Omega}{max}(y_{j+1} - y_j),\ j = 1, 2, ..., N-1,$$

which measure the space interval of Ω. M and N define the number of interpolation points in $x - axis$ and $y - axis$ respectively. The smaller h_1 and h_2 are, the more sampling points are needs. As long as h_i $(i = 1, 2)$ is controlled within a certain range, the interpolation points in the region can be randomly distributed, or even not a simple shape.

For given h_1 and h_2, Ω is divided into $(14/h_1) \times (14/h_2)$ small rectangles with an area of $h_1 \times h_2$. There are $14/h_1+1$ horizontal lines in Ω. For each horizontal lines, $14/h_1+1$ interpolation points are set, when the points are equidistant. In this case, the number of interpolation points is $M \times N = \frac{14}{h_1+1} \times \frac{14}{h_2+1}$. The points $x = L_x^1$ and $y = L_y^1$ are always chosen as the last interpolation points in x and y direction separately. In the following Table 1, N_d denotes the numbers of the interpolation spots.

5.1 The optimal value of shape parameter

In this experiment, $\Omega = \{-7 \le x \le 7, -7 \le y \le 7\}$ is the interpolation domain, which space step is $h_1 = h_2$ and time interval is τ = 1e-04. Suppose that the shape parameter c is equal to c_2, in this case, it is denoted as c. For the following Table, the value of h is fixed, and then change the value of c, so that the correlation between errors and c can be observed, so as to obtain the optimal value of c.

Now the interpolation points evenly spaced in the domain. The data in all tables in this subsection are obtained at $t = 0.01$. Empirical results show that when c changes from 1e-05 to 1e+04, the errors of L_2 and L_∞ keep within 1e-05 and 1e-04 orders of magnitude respectively, which shows that MQQI method has very good stability in a short time and is not sensitive to the value of shape parameters, as presented in Table 1.

Combined with Tables 2 to 7, it is not difficult to find that when the filling distance h decreases gradually, c must be larger to retain the error in the same order of magnitude as Table 1. The optimal

Table 1

c	1e-05	1e-04	1e-03	1e-02	1e-01	1	1.45
N_d	225	225	225	225	225	225	225
L_2	8.52377e-05	8.52375e-05	8.52356e-05	8.52145e-05	8.48247e-05	7.6485e-05	7.56725e-05
L_∞	3.04754e-04	3.04754e-04	3.04754e-04	3.04724e-04	3.031728e-04	2.25628e-04	2.24467e-04
c	1.46	1.47	1.48	1e+01	1e+02	1e+03	1e+04
N_d	225	225	225	225	225	225	225
L_2	7.56721e-05	7.56721e-05	7.56726e-05	8.07334e-05	8.14634e-05	8.14637e-05	8.14637e-05
L_∞	2.24439e-04	2.24412e-04	2.24384e-04	2.26018e-04	2.26916e-04	2.26917e-04	2.26917e-04

$h = 1$, CPU time: 0.03s

Table 2

c	1e-05	1e-04	1e-03	1e-02	1e-01	1	1.55
N_d	441	441	441	441	441	441	441
L_2	1.11186e-04	1.11185e-04	1.11181e-04	1.11122e-04	1.09543e-04	7.88396e-05	7.68623e-05
L_∞	6.22012e-04	6.220124e-04	6.22011e-04	6.21885e-04	6.09535e-04	2.53142e-04	2.52066e-04
c	1.56	1.57	1.58	1e+01	1e+02	1e+03	1e+04
N_d	441	441	441	441	441	441	441
L_2	7.68615e-05	7.68613e-05	7.68616e-05	8.17509e-05	8.248e-05	8.24803e-05	8.24803e-05
L_∞	2.52045e-04	2.52024e-04	2.52002e-04	2.53356e-04	2.54065e-04	2.54066e-04	2.54066e-04

$h = 0.7$, CPU time: 0.04s

Table 3

c	1e-05	1e-04	1e-03	1e-02	1e-01	1	1.61
N_d	841	841	841	841	841	841	841
L_2	1.63512e-04	1.63511e-04	1.63502e-04	1.63366e-04	1.58092e-04	8.04161e-05	7.75625e-05
L_∞	1.21915e-03	1.21915e-03	1.21915e-03	1.21867e-03	1.17205e-03	2.57034e-04	256051e-04
c	1.62	1.63	1.64	1e+01	1e+02	1e+03	1e+04
N_d	841	841	841	841	841	841	841
L_2	7.75619e-05	7.75619e-05	7.75624e-05	8.22622e-05	8.29875e-05	8.29878e-05	8.29878e-05
L_∞	2.56033e-04	2.56015e-04	2.55997e-04	2.57212e-04	2.57774e-04	2.57775e-04	2.57775e-04

$h = 0.5$, CPU time: 0.06s

Table 4

c	1e-04	1e-03	1e-02	1e-01	1	1.64	1.65
N_d	1296	1296	1296	1296	1296	1296	1296
L_2	2.21343e-04	2.21329e-04	2.21092e-04	2.09733e-04	8.12211e-05	7.79411e-05	7.79405e-05
L_∞	1.90492e-03	1.9049e-03	1.90373e-03	1.79212e-03	2.53166e-04	2.52245e-04	2.52229e-04
c	1.66	1.67	1.68	1e+01	1e+02	1e+03	1e+04
N_d	1296	1296	1296	1296	1296	1296	1296
L_2	7.79405e-05	7.7941e-05	7.7942e-05	8.25192e-05	8.32413e-05	8.32416e-05	8.32416e-05
L_∞	2.52213e-04	2.52197e-04	2.52181e-04	2.53334e-04	2.53817e-04	2.53818e-04	2.53818e-04

$h = 0.4$, CPU time: 0.09s

Table 5

c	1e-04	1e-03	1e-02	1e-01	1	1.68	1.69
N_d	3249	3249	3249	3249	3249	3249	3249
L_2	4.4049e-04	4.40457e-04	4.39599e-04	3.85529e-04	8.24496e-05	7.85521e-05	7.85512e-05
L_∞	4.87641e-03	4.87633e-03	4.86862e-03	4.19361e-03	2.57154e-04	2.56213e-04	2.56197e-04
c	1.70	1.71	1.72	1e+01	1e+02	1e+03	1e+04
N_d	3249	3249	3249	3249	3249	3249	3249
L_2	7.85507e-05	7.85508e-05	7.85514e-05	8.29172e-05	8.36331e-05	8.36334e-05	8.36334e-05
L_∞	2.56182e-04	2.56166e-04	2.56151e-04	2.57294e-04	2.57774e-04	2.57775e-04	2.57775e-04

$h = 0.25$, CPU time: 0.18s

Table 6

c	1e-04	1e-03	1e-02	1e-01	1	1.7	1.71
N_d	5041	5041	5041	5041	5041	5041	5041
L_2	6.15884e-04	6.15833e-04	6.14154e-04	5.04184e-04	8.28577e-05	7.87654e-05	7.87646e-05
L_∞	7.61912e-03	7.61894e-03	7.60014e-03	6.06187e-03	2.57101e-04	2.5611e-04	2.56095e-04
c	1.72	1.73	1.75	1e+01	1e+02	1e+03	1e+04
N_d	5041	5041	5041	5041	5041	5041	5041
L_2	7.87643e-05	7.87645e-05	7.87663e-05	8.30534e-05	8.37669e-05	8.37672e-05	8.37672e-05
L_∞	2.56079e-04	2.56063e-04	2.56032e-04	2.57224e-04	2.57732e-04	2.57733e-04	2.57733e-04

$h = 0.2$, CPU time: 0.28s

Table 7

c	1e-04	1e-03	1e-02	1e-01	1	1.74	1.75
N_d	19881	19881	19881	19881	19881	19881	19881
L_2	1.75203e-03	1.75177e-03	1.73579e-03	9.39713e-04	8.36552e-05	7.92055e-05	7.9205e-05
L_∞	3.04676e-03	3.04647e-03	3.01664e-03	1.44306e-03	2.57213e-04	2.56272e-04	2.56257e-04
c	1.75	1.77	1.78	1e+01	1e+02	1e+03	1e+04
N_d	19881	19881	19881	19881	19881	19881	19881
L_2	7.92049e-05	7.92053e-05	7.92061e-05	8.33311e-05	8.40393e-05	8.40396e-05	8.40396e-05
L_∞	2.56243e-04	2.56229e-04	2.56215e-04	2.57342e-04	2.57774e-04	2.57775e-04	2.57775e-04

$h = 0.1$, CPU time: 0.65s

value of c is between 1 and 2, which holds for any filling distance h, ($h \in [0.1, 1]$). It can be seen that there is a negative correlation between h and c, in terms of L_2 error.

In Tables 1 to 7, the value of c slightly greater than 1 is to obtain the best L_2 error. The value of c when L_2 error is the best is shown in the upper part of table 8, which can be directly observed in the previous tables. The same method is used to get the c value when the L_∞ error is optimal, as shown in the lower part of Table 8. For each filling distance h, the L_2 and L_∞ errors corresponding to each optimal value c are given simultaneously in Table 8.

Table 8

h	0.1	0.2	0.25	0.4	0.5	0.7	1
c	1.75	1.72	1.70	1.66	1.63	1.57	1.47
L_2	7.9205e-05	7.87643e-05	7.85507e-05	7.79405e-05	7.75619e-05	7.68613e-05	7.56721e-05
L_∞	2.56257e-04	2.56079e-04	2.56182e-04	2.52213e-04	2.56015e-04	2.52024e-04	2.24412e-04
c	3.34	3.44	3.32	3.15	3.29	3.50	3.52
L_2	8.0623e-05	8.03856e-05	8.01127e-05	7.94742e-05	7.93241e-05	7.89709e-05	7.79341e-05
L_∞	2.54404e-04	2.53707e-04	2.53952e-04	2.5007e-04	2.53401e-04	2.48664e-04	2.20265e-04

On the whole, we also find that the error accuracy of this method is the highest near $c = 1$, so the more accurate value of c are discussed. In Tables 1 to 7, the value of c slightly greater than 1 is to gain the optimal L_2 error. The L_∞ error first decreases until it reaches the minimum value near $c = 1$, and then increases gently until it remains unchanged. The value of L_2 error has the same changing trend as well. From the overall point of view, as c is a number greater than 1, the error accuracy is better. When c is greater than 100, the error almost does not change, which indicates that the MQQI algorithm converges. The error precision when c is a large positive number is the same as that c is the optimal value, which seems to be consistent with the conclusion of Luh [4] roughly. Luh points out that $c = 12b_0 = 120$ is the optimal value, where $r/2 \leq b_0 \leq r$, r is the the diameter of Ω, the value of b_0 is defined as 10.

CPU time refers to the time required for each command execution, which is presented in the table comments. Although when the number of interpolation points is a large value (the number in Table 7

is close to 20000), the program can still get the result within one second, which greatly improves the efficiency of the program.

5.2 The choice of time and space step

In this experiment, we first fix h, and define c as its optimal value (which can be obtained directly from Table 8), then research the relevance between time and space step and approximation effect from Table 9 to Table 14. The values of other parameters are consistent with Table 1. In this text, it is worth noting that the value of τ should not be too small, because it will lead to a sharp increase in the amount of calculation.

Table 9

τ	1e-06	5e-06	1e-05	5e-05	1e-04	5e-04	1e-03
h	1	1	1	1	1	1	1
c	1.47	1.47	1.47	1.47	1.47	1.47	1.47
L_2	4.00474e-05	3.96768e-05	3.95028e-05	4.87799e-05	7.56721e-05	3.54122e-04	7.12154e-04
L_∞	1.21065e-04	1.15793e-04	1.09203e-04	1.60302e-04	2.24412e-04	9.99431e-04	1.9995e-03
c	3.52	3.52	3.52	3.52	3.52	3.52	3.52
L_2	3.7905e-05	3.77966e-05	3.79667e-05	4.97088e-05	7.79341e-05	3.57103e-04	7.14742e-04
L_∞	1.02178e-04	9.83394e-05	1.04756e-04	1.56092e-04	2.20265e-04	9.99373e-04	1.99945e-03
CPU time	0.37s	0.14s	0.09s	0.04s	0.03s	0.02s	0.02s

Table 10

τ	1e-06	5e-06	1e-05	5e-05	1e-04	5e-04	1e-03
h	0.5	0.5	0.5	0.5	0.5	0.5	0.5
c	1.63	1.63	1.63	1.63	1.63	1.63	1.63
L_2	4.12709e-05	4.09459e-05	4.08275e-05	5.04156e-05	7.75619e-05	3.59185e-04	7.21589e-04
L_∞	1.10171e-04	1.04913e-04	1.09465e-04	1.6791e-04	2.56015e-04	9.99425e-04	1.9995e-03
c	3.29	3.29	3.29	3.29	3.29	3.29	3.29
L_2	3.92286e-05	3.91206e-05	3.92887e-05	5.09786e-05	7.93241e-05	3.61676e-04	7.23765e-04
L_∞	1.00721e-04	9.99725e-05	1.06387e-04	1.65255e-04	2.53401e-04	9.99381e-04	1.99946e-03
CPU time	1.07s	0.29s	0.19s	0.08s	0.06s	0.03s	0.03s

Table 11

τ	1e-06	5e-06	1e-05	5e-05	1e-04	5e-04	1e-03
h	0.4	0.4	0.4	0.4	0.4	0.4	0.4
c	1.66	1.66	1.66	1.66	1.66	1.66	1.66
L_2	4.13186e-05	4.10056e-05	4.0905e-05	5.0648e-05	7.79405e-05	3.60436e-04	7.23907e-04
L_∞	.113434e-04	1.07373e-04	1.12083e-04	1.71329e-04	2.52213e-04	9.99422e-04	1.9995e-03
c	3.15	3.15	3.15	3.15	3.15	3.15	3.15
L_2	3.93098e-05	3.91949e-05	3.93563e-05	5.10484e-05	7.94742e-05	3.62693e-04	7.25884e-04
L_∞	1.02469e-04	1.02061e-04	1.09471e-04	1.68749e-04	2.5007e-04	9.99384e-04	1.99946e-03
CPU time	1.51s	0.60s	0.50s	0.14s	0.08s	0.04s	0.03s

Table 9 to Table 14 research the influence of time step on error accuracy under different optimal value c, and give the running time of each order. From a global perspective, when the value of τ is less than 1e-04, the errors precision are kept at 1e-05 and 1e-04 order of magnitude respectively. But the value of τ is not the smaller, the better. Such as, the error precision of τ = 1e-05 is higher than that of τ = 1e-06, and the error precision at this moment is the highest. In order to reduce the calculation, the value of τ can be selected from 1e-05 to 1e-04. Therefore, τ = 1e-04 is a good choice in the previous

section. From Table 9 to 14, we can also find that when the time step τ is less than or equal to 1e-05, we can choose a larger value of c to get higher accuracy. When τ is greater than 1e-05, we need to choose a smaller value of c.

Table 12

τ	1e-06	5e-06	1e-05	5e-05	1e-04	5e-04	1e-03
h	0.25	0.25	0.25	0.25	0.25	0.25	0.25
c	1.70	1.70	1.70	1.70	1.70	1.70	1.70
L_2	4.14038e-05	4.11127e-05	4.10415e-05	5.10312e-05	7.85507e-05	3.62402e-04	7.27533e-04
L_∞	1.10666e-04	1.04411e-04	1.11485e-04	1.72713e-04	2.56182e-04	1.01188e-03	1.99949e-03
c	3.32	3.32	3.32	3.32	3.32	3.32	3.32
L_2	3.95056e-05	3.94126e-05	3.96019e-05	5.15005e-05	8.01127e-05	3.64645e-04	729495e-04
L_∞	1.00511e-04	1.01285e-04	1.08942e-04	1.70201e-04	2.53952e-04	1.01141e-04	1.99946e-03
CPU time	3.24s	0.95s	0.67s	0.32s	0.18s	0.06s	0.05s

Table 13

τ	1e-06	5e-06	1e-05	5e-05	1e-04	5e-04	1e-03
h	0.2	0.2	0.2	0.2	0.2	0.2	0.2
c	1.72	1.72	1.72	1.72	1.72	1.72	1.72
L_2	4.13941e-05	4.11116e-05	4.1053e-05	5.115e-05	7.87643e-05	3.63096e-04	7.28796e-04
L_∞	1.09546e-04	1.04796e-04	1.12201e-04	1.72368e-04	2.56079e-04	1.01454e-03	1.99949e-03
c	3.44	3.44	3.44	3.44	3.44	3.44	3.44
L_2	3.95897e-05	3.9508e-05	3.97116e-05	5.17039e-05	8.03856e-05	3.65362e-04	7.30774e-04
L_∞	1.00951e-04	1.217e-04	1.0958e-04	1.69959e-04	2.53707e-04	1.01449e-03	1.99946e-03
CPU time	5.11s	1.37s	0.92s	0.47s	0.27s	0.11s	0.06s

Table 14

τ	1e-06	5e-06	1e-05	5e-05	1e-04	5e-04	1e-03
h	0.1	0.1	0.1	0.1	0.1	0.1	0.1
c	1.75	1.75	1.75	1.75	1.75	1.75	1.75
L_2	4.1429e-05	4.1164e-05	4.11286e-05	5.14189e-05	7.9205e-05	3.64498e-04	7.31361e-04
L_∞	1.07753e-04	1.05015e-04	1.12253e-04	1.73409e-04	2.56257e-04	1.0146e-03	2.00575e-03
c	3.34	3.34	3.34	3.34	3.34	3.34	3.34
L_2	3.96868e-05	3.9604e-05	3.98074e-05	5.18409e-05	8.0623e-05	3.66542e-04	7.3315e-04
L_∞	1.00893e-04	1.025673e-04	1.09887e-04	1.71157e-04	2.54404e-04	1.01453e-03	2.00611e-03
CPU time	19.43s	4.27s	2.35s	0.80s	0.66s	0.27s	0.18s

For now, we control the value of h and only consider the influence of a single variable τ on the fitting effect of the equation. Next, we determine the grid ratio (fixed ratio of τ to h) to discuss the influence on the approximation effect, as shown in Table 15 and 16.

Table 15 and 16 explain that when h and τ are reduced at the same time, the error accuracy is higher. Table 15 shows when $\frac{\tau}{h} = \frac{1}{1000}$, more interpolation points (that is, smaller filling distance h) are needed to improve the accuracy of MQQI algorithm. As $\frac{\tau}{h} = \frac{1}{10000}$, for any filling distance h, the algorithm has high accuracy.

Table 16 demonstrates the trend of errors accuracy when the grid ratio is $\frac{1}{2000}$ and $\frac{1}{20000}$ respectively. Good error accuracy can be obtained when fill distance $h = 0.25$ and time step τ = 1.25e-04. The experimental results present that the accuracy of the algorithm can be improved by reducing the time step and space step at the same time. The MQQI method may be unstable when the grid is large, which is a pity of this method.

Table 15

c	1.48	1.64	1.68	1.72	1.75	1.78
h	1	0.5	0.4	0.25	0.2	0.1
τ	1e-03	5e-04	4e-04	2.5e-04	2e-04	1e-04
L_2	7.12175e-04	3.59209e-04	2.88135e-04	1.81628e-04	1.46528e-04	7.92061e-05
L_∞	1.9995e-03	9.99425e-04	8.04311e-04	5.286357e-04	4.36312e-04	2.56215e-04
CPU time	0.02s	0.03s	0.04s	0.09s	0.15s	0.61s
h	1	0.5	0.4	0.25	0.2	0.1
τ	1e-04	5e-05	4e-05	2.5e-05	2e-05	1e-05
L_2	7.56726e-05	5.0392e-05	4.66591e-05	4.26753e-05	4.17874e-05	4.09928e-05
L_∞	2.24384e-04	1.67892e-04	1.56479e-04	1.3441e-04	1.2696e-04	1.12203e-04
CPU time	0.03s	0.08s	0.17s	0.49s	0.61s	2.22s

Table 16

c	1.48	1.64	1.68	1.72	1.75	1.78
h	1	0.5	0.4	0.25	0.2	0.1
τ	5e-04	2.5e-04	2e-04	1.25e-04	1e-04	5e-05
L_2	3.54145e-04	1.79761e-04	1.45204e-04	9.46695e-05	7.87663e-05	5.13601e-05
L_∞	9.9943e-04	5.20337e-04	4.36212e-04	3.00204e-04	2.56032e-04	1.73362e-04
CPU time	0.04s	0.04s	0.05s	0.15s	0.27s	0.78s
τ	5e-05	2.5e-05	2e-05	1.25e-05	1e-05	5e-06
L_2	4.87573e-05	4.23432e-05	4.15826e-05	4.10317e-05	4.09105e-05	4.10205e-05
L_∞	1.60274e-04	1.28673e-04	1.26855e-04	1.15277e-04	1.12149e-04	1.04963e-04
CPU time	0.04s	0.12s	0.31s	0.60s	0.84s	4.16s

5.3 The number of interpolation points

5.3.1 Point set of uniform distribution

Here, we alter the number of interpolation points with fixing the value of c = 1.75 and time step τ = 1e-04. Then, we control $h_1 = h_2$, which makes the sampling points equally spaced and uniformly distributed in Ω. All data are recorded at t = 0.01.

The Table 17 present that the L_2 and L_∞ errors increase slightly as the number of interpolation points increase on the whole. N_d denote the numbers of the interpolation points. When we use only 225 interpolation points to approximating the equation in the domain, the errors still perform very good. This demonstrates that we only need a few points to simulate the equation, and can accurately describe the equation, which immensely reduces the amount of calculation and avoids taking up too much RAM.

Table 17

h	1	0.7	0.5	0.4	0.25	0.2	0.1
c	1.75	1.75	1.75	1.75	1.75	1.75	1.75
N_d	225	441	841	1296	3249	5041	19881
L_2	7.58223e-05	7.69344e-05	7.75979e-05	7.79614e-05	7.85557e-05	7.87663e-05	7.9205e-05
L_∞	2.23636e-04	2.51636e-04	2.558e-04	2.52069e-04	2.56105e-04	2.56032e-04	2.56257e-04
CPU time	0.02s	0.04s	0.06s	0.08s	0.18s	0.26s	0.61s

It is not difficult to find that when fill distance h is smaller, the error accuracy of the equation is almost not change, as shown in Table 17. Generally, the number of interpolation points has tiny effect on errors. Therefore, it is not necessary to select too many interpolation points when using MQQI method

to simulate sine-Gordon equations.

5.3.2 Point set of random distribution

In this example, the number of fixed interpolation points is equal to 225. Ω is divided into 14×14 small squares with a length of 1 on each side. Then, we put each interpolation point in a square and let it move randomly, but not beyond the boundary of the square. In this way, a series of stochastic point sets are obtained. In the horizontal direction, the interval between two points is represented by h_1, and the maximum value of h_1 will not exceed 2, because we specify that the side length of each small square is 1. The maximum value of h_2 is not exceed 2 as well, in the vertical direction.

Table 18

c	1.75	1.75	1.75	1.75	1.75	1.75	1.75
$h1$	1.47501	1.51796	1.62331	1.688	1.70996	1.75952	1.80836
$h2$	1.50519	1.71993	1.90052	1.88006	1.65441	1.83736	1.49901
L_2	7.79181e-05	8.01394e-05	7.82469e-05	7.7535e-05	8.11389e-05	7.70429e-05	7.8127e-05
L_∞	272383e-04	301378e-04	267596e-04	265808e-04	313885e-04	268151e-04	2967397e-04

According to Table 18, it shows that although the total number of interpolation points in the domain is stochastically, the L_2 and L_∞ errors remain keeping the order of 1e-05 and 1e-04, which is consistent with the result when the interpolation points are evenly distributed. The above experimental results verify that the MQQI method does not need to divide the mesh. The method can simulate 2D sine-Gordon equation accurately as long as the time step τ and shape parameter c are set appropriately. There is one thing we need to note, the value of shape parameter c depends on the size of interpolation range as well, which can be found from the numerical examples in Section 4.

6 Conclusions

In this paper, the MQQI algorithm in two-dimensional was presented, and used it to simulate the numerical solution of sine-Gordon equation, which proved that the method can accurately and effectively simulate the soliton equation. In addition, we testified that MQQI algorithm performs better than RBF method [7] in approximation of sine-Gordon equations. We also carefully discussed the correlation between parameters and approximation effect, in the fifth part of the paper. Due to the complexity of the influence of shape parameter on the equation, we only considered the selection of shape parameter friendly. We can draw the following conclusions through example verification,

- The choice of shape parameter is not controlled by a single variable, but depends on the dimension of the equation, the size of the interpolation range, etc.
- The value of shape parameter has a negative correlation with spatial step. As the increase of the spatial step, the value of shape parameter also decreases.
- The L_∞ error between the analysis solution and the numerical solution can reach 1e-04, and L_2 error can up to 1e-05. The precision has been improved at least 1000 times, compared with the RBF method of Dehghan [7].
- When the time step size is selected as a smaller value, the error accuracy will be improved, but the amount of calculation will addition. Therefore, in the experiment, the time step size can be selected as 1e-04.

- The number of interpolation points determines the size of space interval. In the numerical example, we verify that the number of interpolation points is not the more the better, but needs reasonable arrangement. Even when the interpolation points are randomly distributed, the fitting effect of the equation is excellent.

All the numerical results in this paper can be completed within several seconds, which proves that the MQQI method is very easy to implement and efficient.

References

1. R. Z. Feng and S. Peng, Quasi-interpolation scheme for arbitrary dimensional scattered data approximation based on natural neighbors and RBF interpolation, J. Comput. Appl. Math., 329 (2018), 95-105.
2. W. W. Gao, G. E. Fasshauer, X. P. Sun and X. Zhou, Optimality and Regularization Properties of Quasi-Interpolation: Deterministic and Stochastic Approaches, SIAM. J. Numer. Anal., 58 (2020), 2059-2078.
3. Y. Duan and F. Rong, A numerical scheme for nonlinear Schrödinger equation by MQ quasi-interpolation, Eng. Anal. Bound. Elem., 37 (2013), 89-94.
4. L. T. Luh, The mystery of the shape parameter III, Appl. Comput. Harmon. Anal., 40 (2016), 186-199.
5. L. T. Luh, The choice of the shape parameter-A friendly approach, Eng. Anal. Bound. Elem., 98 (2019), 103-109.
6. S. S. Lii, Y. Duan and A. Naji, Convergence estimate of Cauchy problems for the shallow water equations with MQ quasi-interpolation, Preprint.
7. M. Dehghan and A. Shokri, A numerical method for solution of the two-dimensional sine-Gordon equation using the radical basis functions, Math. Comput. Simulat., 79 (2008), 700-715.
8. H. Y. Wu and Y. Duan, Multi-quadric quasi-interpolation method coupled with FDM for the Degasperis-Procesi equation, Appl. Math. Comput., 274 (2016), 83-92.
9. R. H. Chen and Z. M. Wu, Solving hyperbolic conservation laws using multiquadric quasi-interpolation, Numer. Meth. Part. D. E., 22 (2006), 776-796.
10. Z. M. Wu and R. Schaback, Shape preserving properties and convergence of univariate multiquadric quasi-interpolation, Acta. Math. Appl. Sin., 10 (1994), 441-446.
11. J. Argyris, M. Haase and J. C. Heinrich, Finite element approximation to two-dimensional sine-Gordon solitons, Comput. Meth. Appl. Mech. Eng., 86 (1991), 1-26.
12. W. J. Cai, C. L. Jiang and Y. Z. Song, Structure-preserving algorithms for the two-dimensional sine-Gordon equation with Neumann boundary conditions, J. Comput. Phys. 395 (2019), 166-185.
13. Z. J. Sun, Z. M. Wu and W. W. Gao, An iterated quasi-interpolation approach for derivative approximation, Numer. Algorithms., 85 (2020), 255-276.
14. F. Usta and J. Levesley, Quasi-interpolation on a sparse grid with Gaussian, Numer. Algorithms., 77 (2018), 793 – 808.
15. R. Beatson and M. Powell, Univariate multiquadric approximation: Quasi-interpolation to scattered data, Constr. Approx., 8 (1992), 275 – 288.
16. Z. M. Wu and R. Schaback, Shape preserving properties and convergence of univariate multiquadric quasi-interpolation, Acta Math. Appl. Sin., 10 (1994), 441 – 446.
17. L. M. Ma, Z. M. Wu, Approximation to the k-th derivatives by multiquadric quasi-interpolation method, J. Comput. Appl. Math., 231 (2), (2009), 925 – 32.
18. L. M. Ma, Z. M. Wu, A numerical method for one-dimensional nonlinear Sine-Gordon equation using multiquadric quasi-interpolation, Chinese. Phys. B. 18 (8), (2009), 3099 – 4005.
19. C. L. Jiang, Y. S. Wang and W. J. Cai, A linearly implicit energy-preserving exponential integrator for the nonlinear Klein-Gordon equation, J. Comput. Phys., 419 (2020), 109690.