Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 222 (2007) 176–193

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

An efficient adaptive mesh redistribution method for a non-linear Dirac equation

Han Wang, Huazhong Tang [*]

LMAM, School of Mathematical Sciences, Peking University, 5# YiHe Yuan Lu, Haidian District, Beijing 100871, PR China

Received 7 March 2006; received in revised form 9 July 2006; accepted 20 July 2006
Available online 30 August 2006

Abstract

This paper presents an efficient adaptive mesh redistribution method to solve a non-linear Dirac (NLD) equation. Our algorithm is formed by three parts: the NLD evolution, the iterative mesh redistribution of the coarse mesh and the local uniform refinement of the final coarse mesh. At each time level, the equidistribution principle is first employed to iteratively redistribute coarse mesh points, and the scalar monitor function is subsequently interpolated on the coarse mesh in order to do one new iteration and improve the grid adaptivity. After an adaptive coarse mesh is generated ideally and finally, each coarse mesh interval is equally divided into some fine cells to give an adaptive fine mesh of the physical domain, and then the solution vector is remapped on the resulting new fine mesh by an affine method. The NLD equation is finally solved by using a high resolution shock-capturing method on the (fixed) non-uniform fine mesh.

Extensive numerical experiments demonstrate that the proposed adaptive mesh method gives the third-order rate of convergence, and yields an efficient and fast NLD solver that tracks and resolves both small, local and large solution gradients automatically.

Keywords: Adaptive mesh redistribution; The Dirac equation; Local uniform refinement; Solitary wave; High resolution scheme

1. Introduction

Ever since its invention in 1929 the Dirac equation has played a fundamental role in various areas of modern physics and mathematics, and is important for the description of interacting particles and fields. In past three decades, several authors have committed themselves to analytically investigating the non-linear Dirac (NLD) model, see [4,6–8,31,32] and references therein. Some reliable, higher-order accurate numerical methods have also been constructed to solve the NLD model. They are Crank–Nicholson type schemes [3,5], split-step spectral schemes [17], Legendre rational spectral methods [46], multi-symplectic Runge–Kutta methods [22], and Runge–Kutta discontinuous Galerkin (RKDG) methods [35], etc. The interaction dynamics for the one-humped Dirac solitary waves were investigated in [5] by using a

[*] Corresponding author. Tel.: +86 10 62757018; fax: +86 10 62751801.
E-mail addresses: amcadmus@hotmail.com (H. Wang), hztang@pku.edu.cn (H. Tang).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.07.011

second-order accurate difference scheme. The weakly inelastic interaction in ternary collisions is first reported in [35]. In [36,37], the second author and his co-worker further observed the strong inelastic interaction in ternary collisions, and investigated interaction dynamics of two-humped Dirac solitary waves with or without an initial phase shift.

In studying the interaction dynamics of the Dirac solitary waves, the physical solutions are usually very singular in fairly localized regions. To resolve these large solution variations, numerical simulations require extremely fine meshes on those small localized portions of the physical domain. They will become very expensive if a uniform fine mesh is used. Otherwise, the numerical wave will incorrectly propagate due to under-resolution of the waves. Hence it is very necessary to develop an effective adaptive grid method for the NLD model. Successful implementation of an adaptive strategy can increase accuracy of numerical approximation and also decrease computational costs.

Adaptive moving mesh methods have important applications in solving partial differential equations (PDEs). Up to now, there have been important progresses, including the variational approach of Winslow [47], Brackbill and Saltzman [11,12], Dvinsky [18] and Li et al. [25,26]; moving finite element methods of Miller and Miller [29], Davis and Flaherty [16], and Beckett et al. [10]; and moving mesh PDEs of Cao et al. [13,14], Li and Petzold [27], Ceniceros and Hou [15] and Ren and Wang [33] as well as [45], etc. Harten and Hyman [21] began the earliest study of the self-adaptive moving mesh methods to improve resolution of discontinuous solutions of hyperbolic equations. After their work, many other moving mesh methods in this direction have been proposed based on combining the variational grid methods with high resolution shock capturing methods. They include works of Azarenok et al. [9], Fazio and LeVeque [19], Liu et al. [28], Saleri and Steinberg [34], Stockie et al. [39], Tang et al. [40–42], Zegeling et al. [48,49] and Zhang [50]. We refer the readers to a recent paper [43] for a detailed review. However, to our knowledge, there is no any research work on adaptive moving mesh methods for the NLD model in the literatures.

The aim of this paper is to present an efficient adaptive mesh redistribution method for the NLD model based on the adaptive moving mesh method for hyperbolic conservation laws, proposed by Tang and Tang in [41]. The present adaptive mesh algorithm will include three parts: the NLD evolution, an iterative redistribution of the coarse mesh, and the local uniform refinement of the final coarse mesh. The NLD evolution may be any appropriate high resolution finite volume scheme. While the coarse mesh redistribution is an iterative procedure. In each iteration, the coarse mesh points are first iteratively redistributed by the equidistribution principle, and then the scalar monitor function is updated on the resulting new coarse mesh to improve the grid adaptivity. After the final adaptive coarse mesh is generated ideally, each coarse mesh element is equally divided into some fine cells to give an adaptive fine mesh of the physical domain, and then the solution vector is remapped from the initial fine mesh to the new fine mesh by the affine method. These combinations will yield a powerful and fast NLD solver that tracks and resolves both small, local and large solution gradients automatically. It is worth mentioning that our proposed method may be considered as a combination of the r-refinement method and the h-refinement method and shares the same idea of the two-level mesh movement technique of Huang et al. [23,24] and Fiedler and Trapp [20] as well as the hr-refinement method in the literatures, see e.g. [1,2,30].

This paper is organized as follows. In Section 2 we briefly review the NLD model as well as its two exact solitary wave solutions. In Section 3, we begin to present an adaptive moving mesh method for the NLD equation. Section 4 conducts some numerical experiments to validate the accuracy and capability of the proposed approach. The numerical examples include binary, ternary, and quadruple collisions of the Dirac solitary waves. Emphatically, quadruple collisions of the Dirac solitary waves are investigated for the first time. We conclude the paper with a few remarks in Section 5.

2. Preliminaries

Consider a classical spinorial model with scalar self-interaction, described by the non-linear Lagrangian $L = \mathrm{i}\overline{\psi}\gamma^\mu\partial_\mu\psi - m\overline{\psi}\psi + \lambda(\overline{\psi}\psi)^2$ from which we may derive the non-linear Dirac equation (NLD)

$$\mathrm{i}\gamma^\mu\partial_\mu\psi - m\psi + 2\lambda(\overline{\psi}\psi)\psi = 0, \tag{1}$$

where $\mathrm{i} = \sqrt{-1}, \overline{\boldsymbol{\psi}}$ is the complex conjugate of $\boldsymbol{\psi}$, λ and m are two real constants, and the matrices γ^μ are defined by

$$\gamma^0 = \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix}, \quad \gamma^k = \begin{pmatrix} 0 & \sigma^k \\ -\sigma^k & 0 \end{pmatrix},$$

here σ^k with $k = 1, 2, 3$, denote the Pauli matrices. The non-linear self-coupling term $(\overline{\boldsymbol{\psi}}\boldsymbol{\psi})^2$ in the Lagrangian allows the existence of finite energy, localized solitary waves, or extended particle-like solutions, see e.g. [38].

We restrict our attention to the $(1+1)$-dimensional NLD model (1), and use the notations $\rho_E(x,t)$, $\rho_P(x,t)$ and $\rho_Q(x,t)$ to denote the energy density, the linear momentum density and the charge density, which are defined by

$$\rho_E(x,t) = \mathrm{Im}(\overline{\psi}_1 \partial_x \psi_2 + \overline{\psi}_2 \partial_x \psi_1) + m(|\psi_1|^2 - |\psi_2|^2) - \lambda(|\psi_1|^2 - |\psi_2|^2)^2, \tag{2}$$

$$\rho_P(x,t) = \mathrm{Im}(\overline{\psi}_1 \partial_x \psi_1 + \overline{\psi}_2 \partial_x \psi_2), \tag{3}$$

$$\rho_Q(x,t) = |\psi_1|^2 + |\psi_2|^2, \tag{4}$$

where ψ_1 and ψ_2 are two components of the spinor $\boldsymbol{\psi}(x,t)$. Then we have the (total) energy E, the linear momentum P and the charge Q as follows

$$E(t) = \int_{\mathbb{R}} \rho_E(x,t)\,\mathrm{d}x, \quad P(t) = \int_{\mathbb{R}} \rho_P(x,t)\,\mathrm{d}x, \quad Q(t) = \int_{\mathbb{R}} \rho_Q(x,t)\,\mathrm{d}x, \tag{5}$$

which are conservative, if $\lim_{|x|\to+\infty}|\boldsymbol{\psi}| = 0$ and $\lim_{|x|\to+\infty}|\partial_x \boldsymbol{\psi}| < +\infty$ hold uniformly for $t \geqslant 0$.

The $(1+1)$-dimensional NLD equation (1) has two exact solutions, which will be used in our numerical experiments. The first is the standing wave solution at $x = x_0$ defined by

$$\boldsymbol{\psi}^{\mathrm{sw}}(x - x_0, t) \equiv \begin{pmatrix} \psi_1^{\mathrm{sw}}(x - x_0, t) \\ \psi_2^{\mathrm{sw}}(x - x_0, t) \end{pmatrix} = \begin{pmatrix} A(x - x_0) \\ \mathrm{i}B(x - x_0) \end{pmatrix} \mathrm{e}^{-\mathrm{i}\Lambda t} \tag{6}$$

with

$$A(x) = \frac{\sqrt{\frac{1}{\lambda}(m^2 - \Lambda^2)(m + \Lambda)}\cosh\left(x\sqrt{(m^2 - \Lambda^2)}\right)}{m + \Lambda\cosh\left(2x\sqrt{(m^2 - \Lambda^2)}\right)}, \tag{7}$$

$$B(x) = \frac{\sqrt{\frac{1}{\lambda}(m^2 - \Lambda^2)(m - \Lambda)}\sinh\left(x\sqrt{(m^2 - \Lambda^2)}\right)}{m + \Lambda\cosh\left(2x\sqrt{(m^2 - \Lambda^2)}\right)}. \tag{8}$$

Here $0 < \Lambda \leqslant m$.

The second exact solution of the Dirac model (1) is the single solitary wave solution placed initially at x_0 with a velocity v:

$$\boldsymbol{\psi}^{\mathrm{ss}}(x - x_0, t) = (\psi_1^{\mathrm{ss}}(x - x_0, t), \psi_2^{\mathrm{ss}}(x - x_0, t))^T, \tag{9}$$

where

$$\psi_1^{\mathrm{ss}}(x - x_0, t) = \sqrt{\frac{\gamma + 1}{2}}\psi_1^{\mathrm{sw}}(\tilde{x}, \tilde{t}) + \mathrm{sign}(v)\sqrt{\frac{\gamma - 1}{2}}\psi_2^{\mathrm{sw}}(\tilde{x}, \tilde{t}), \tag{10}$$

$$\psi_2^{\mathrm{ss}}(x - x_0, t) = \sqrt{\frac{\gamma + 1}{2}}\psi_2^{\mathrm{sw}}(\tilde{x}, \tilde{t}) + \mathrm{sign}(v)\sqrt{\frac{\gamma - 1}{2}}\psi_1^{\mathrm{sw}}(\tilde{x}, \tilde{t}), \tag{11}$$

here $\gamma = 1/\sqrt{1 - v^2}$, $\tilde{x} = \gamma(x - x_0 - vt)$, $\tilde{t} = \gamma(t - v(x - x_0))$, ψ_1^{sw} and ψ_2^{sw} are defined in (6) and $\mathrm{sign}(x)$ is the sign function, which returns 1 if $x > 0$, 0 if $x = 0$ and -1 if $x < 0$. The solution $\boldsymbol{\psi}^{\mathrm{ss}}(x - x_0, t)$ represents a sol-

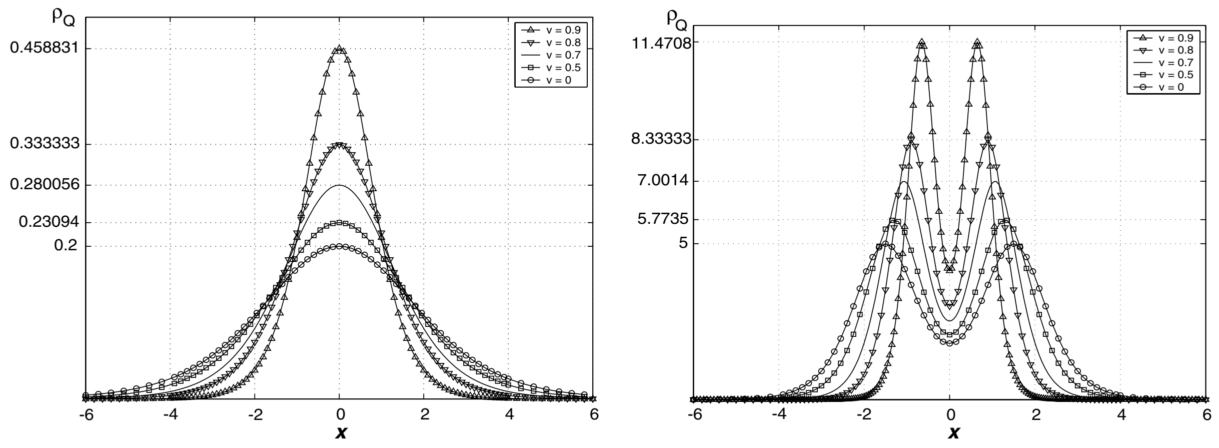


Fig. 1. Dependence of ρ_Q on Λ and v. Left: $\Lambda = 0.9$; right: $\Lambda = 0.1$.

itary wave travelling from left to right if $v > 0$, or travelling from right to left if $v < 0$, and the standing wave $\boldsymbol{\psi}^{\text{sw}}(x - x_0, t)$ is actually a solitary wave at rest placed at x_0 or identical to $\boldsymbol{\psi}^{\text{ss}}(x - x_0, t)$ with $v = 0$.

The profile of the solution (6) or (9) is strongly dependent on the parameter Λ:

- it is a two-humped solitary wave with two peaks whose locations are determined by $\cosh(2\sqrt{m^2 - \Lambda^2}\tilde{x}) = \frac{m^2-\Lambda^2}{m\Lambda}$ if $0 < \Lambda < \frac{m}{2}$;
- it becomes a one-humped solitary wave with one peak located at $\tilde{x} = 0$ if $\frac{m}{2} \leqslant \Lambda < m$; and
- $\boldsymbol{\psi}^{\text{ss}}(x - x_0, t) \equiv 0$ if $\Lambda = m$.

Moreover, amplitude of the solitary waves also depends strongly on the velocity v: $\rho_Q^{\text{ss}}(x - x_0, t) = \gamma\rho_Q^{\text{sw}}(\tilde{x}, \tilde{t})$. Fig. 1 shows that dependence, which will give different interaction dynamics. We also refer the readers to [35–37] for more detailed investigations. It is worth noting that $e^{i\theta}\boldsymbol{\psi}^{\text{ss}}(x - x_0, t)$ is still a solitary wave solution of the (1 + 1)-dimensional Dirac model (1), if θ is a constant.

For actual numerical computations, we decompose the complex function $\psi_1(x, t)$ and $\psi_2(x, t)$ into its real and imaginary parts by writing

$$\psi_i(x,t) = \psi_i^r(x,t) + i\psi_i^s(x,t), \quad i = 1, 2$$

and then rewrite the (1 + 1)-dimensional NLD model (1) in a conservative form of real variables

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{u})}{\partial x} = \boldsymbol{s}(\boldsymbol{u}), \tag{12}$$

where $\boldsymbol{u} = (\psi_1^r, \psi_1^s, \psi_2^r, \psi_2^s)^T$, $\boldsymbol{s}(\boldsymbol{u}) = g(x,t)(\psi_1^s, -\psi_1^r, -\psi_2^s, \psi_2^r)^T$, and

$$\boldsymbol{f}(\boldsymbol{u}) = \boldsymbol{A}\boldsymbol{u} \equiv \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \psi_1^r \\ \psi_1^s \\ \psi_2^r \\ \psi_2^s \end{pmatrix},$$

here $g(x,t) := m + 2\lambda(|\psi_2|^2 - |\psi_1|^2)$, and $|\psi_i|^2 = (\psi_i^r)^2 + (\psi_i^s)^2, i = 1, 2$.

3. An adaptive mesh method

This section presents an adaptive moving mesh finite volume approach for the (1 + 1)-dimensional NLD model, which is an extension of the method introduced by Tang and Tang [41]. To increase efficiency, we only apply the iterative grid redistribution technique to the coarse mesh, and then refine uniformly each final coarse mesh interval. That will yield a powerful and fast NLD solver that tracks and resolves both small, local and large solution gradients automatically.

3.1. Coarse mesh redistribution

Let x and ξ denote the physical and logical coordinates, respectively. A one-to-one coordinate transformation from logical domain $\Omega_l = [0,1]$ to the physical domain $\Omega_p = [a,b]$ is denoted by

$$x = x(\xi), \quad \xi \in \Omega_l.$$

Its inversion is denoted by

$$\xi = \xi(x), \quad x \in \Omega_p.$$

If giving a uniform partition of the logical domain Ω_l such as $0 = \xi_0 < \xi_1 < \cdots < \xi_N = 1$, ($\xi_j = j/N$, $j = 0,1,2,\ldots,N$), then we usually use the coordinate transformation $x = x(\xi)$ to give an "adaptive" mesh of the physical domain Ω_p: $a = x_0 < x_1 < \cdots < x_N = b$, where $x_j = x(\xi_j)$, $j = 0,1,2,\ldots,N$. To derive the coordinate transformation, we employ the well-known *equidistribution principle*, and then have

$$(\omega x_\xi)_\xi = 0, \quad \xi \in \Omega_l, \tag{13}$$

subject to boundary conditions $x(0) = a$ and $x(1) = b$. Here, ω is a positive weight function, i.e. the so-called *monitor function*. A widely used monitor function is defined by

$$\omega = \sqrt{1 + \alpha|\boldsymbol{u}|^2 + \beta|\boldsymbol{u}_x|^2}. \tag{14}$$

Solving (13) will end up with a desired mesh map $x = x(\xi)$. In this work, we use an iteration method, such as Gauss–Siedel iteration method, to solve the mesh redistribution equation (13), i.e.

$$\omega\left(x^{[\nu]}_{j+\frac{1}{2}}\right)\left(x^{[\nu]}_{j+1} - x^{[\nu+1]}_{j}\right) - \omega\left(x^{[\nu]}_{j-\frac{1}{2}}\right)\left(x^{[\nu+1]}_{j} - x^{[\nu+1]}_{j-1}\right) = 0 \tag{15}$$

for $j = 1,2,\ldots N-1$, where $\nu = 0,1,\ldots$

For most implementations of the mesh redistribution, the iteration number ν in (15) is usually controlled under a tolerance denoted by μ, in order to save CPU time. However, if the wave propagation speed becomes large, then the *equidistribution principle* cannot be perfectly preserved in numerical computations, unless we take a very large value of μ. To overcome the above disadvantage, we employ a local uniform refinement technique, that is to say, we only redistribute coarse mesh points by (15) iteratively, and then divide the final coarse cell (i.e. $\nu = \mu$) into some locally equal fine cells. Assume $N = N_0N_1$, where N_0 and N_1 are two positive integers, and use $\{x_j, j = 0, N_0, 2N_0, \ldots, N_1N_0\}$ to denote the coarse mesh. Then applying the above iterative mesh redistribution, e.g. (15), to the coarse mesh $\{x_j, j = 0, N_0, 2N_0, \ldots, N_1N_0\}$ gives

$$\omega\left(x^{[\nu]}_{j+\frac{N_0}{2}}\right)\left(x^{[\nu]}_{j+N_0} - x^{[\nu+1]}_{j}\right) - \omega\left(x^{[\nu]}_{j-\frac{N_0}{2}}\right)\left(x^{[\nu+1]}_{j} - x^{[\nu+1]}_{j-N_0}\right) = 0 \tag{16}$$

for $j = N_0, 2N_0, \ldots, (N_1 - 1)N_0$, where $\nu = 0,1,\ldots,\mu - 1$.

3.2. Interpolation of the monitor function on the coarse mesh

After each iteration of the mesh redistribution, the moving mesh finite volume approach of Tang and Tang [41] should remap the solution $\boldsymbol{u}$ on the new mesh $\left\{x^{[\nu+1]}_j\right\}_{j=0}^{N}$ by a high resolution conservative interpolation, according to the known data $\left\{x^{[\nu]}_j\right\}_{j=0}^{N}$ as well as $\left\{\boldsymbol{u}^{[\nu]}_{j+\frac{1}{2}}\right\}_{j=0}^{N-1}$, and then calculate $\omega^{[\nu+1]}_{j+\frac{1}{2}}$ for next iteration. It is proved that such remapping phase is successful and robust in capturing strong discontinuities (shock waves, etc.) in fluid flows.

However, since the iterative grid redistribution is solely implemented on the coarse mesh in the present algorithm, the remapping phase may be operated on the same coarse mesh. To further save the computational cost, we will directly remap the scalar monitor function ω on the coarse mesh $\{x_j^{[\nu+1]}, j = 0, N_0, 2N_0, \ldots, N_1N_0\}$ instead of the solution vector $\boldsymbol{u} = (u_1, u_2, u_3, u_4)^T$.

Assume that we have solved (16) to yield $\{x_j^{[\nu+1]}, j = 0, N_0, 2N_0, \ldots, N_1N_0\}$. Since it is not important whether the function ω is conservative, we may consider $\omega^{[\nu]}_{j+\frac{N_0}{2}}$ as an approximation of $\omega\left(x^{[\nu]}_{j+\frac{N_0}{2}}\right)$ and remap ω by a simple linear interpolation such as

$$\omega^{[\nu]}_{j+\frac{N_0}{2}} = \omega^{[0]}_{k+\frac{N_0}{2}} + \frac{\omega^{[0]}_{k+\frac{N_0}{2}} - \omega^{[0]}_{k-\frac{N_0}{2}}}{x^{[0]}_{k+\frac{N_0}{2}} - x^{[0]}_{k-\frac{N_0}{2}}}\left(x^{[\nu]}_{j+\frac{N_0}{2}} - x^{[0]}_{k+\frac{N_0}{2}}\right), \tag{17}$$

if $x^{[\nu]}_{j+\frac{N_0}{2}} \in \left[x^{[0]}_{k-\frac{N_0}{2}}, x^{[0]}_{k+\frac{N_0}{2}}\right]$, where $\nu \geqslant 1$. It is worth noting that we always interpolate the monitor function $\omega^{[\nu]}_{j+\frac{N_0}{2}}$ by using the "initial" data $\omega^{[0]}_{j+\frac{N_0}{2}}$.

Remark 3.1. The advantage of the interpolation (17) is that at each time level we only need to compute the "initial" first-order divided difference, see (17), during the iterative redistribution of the coarse mesh.

Remark 3.2. The initial value of the monitor function $\omega^{[0]}_{j+\frac{N_0}{2}}$ is computed by using a discrete form of (14) with the volume average of $\boldsymbol{u}^{[0]}_{j+\frac{1}{2}}$ over the fine cell, i.e.

$$\boldsymbol{u}^{[0]}_{j+\frac{N_0}{2}} = \frac{1}{\Delta x^{[0]}_{j+\frac{N_0}{2}}} \sum_{l=1}^{N_0} \Delta x^{[0]}_{j+\frac{l}{2}} \boldsymbol{u}^{[0]}_{j+\frac{l}{2}}. \tag{18}$$

Remark 3.3. We may also consider $\omega_{j+\frac{N_0}{2}}$ as cell averages of the monitor function ω over the cell $[x_j, x_{j+N_0}]$, and then employ the interpolation approach as introduced by Tang et al. [41] to remap the monitor function ω on the new mesh $\{x_j^{[\nu+1]}, j = 0, N_0, 2N_0, \ldots, (N_1-1)N_0\}$ as follows

$$\Delta x^{[\nu+1]}_{j+\frac{N_0}{2}} \omega^{[\nu+1]}_{j+\frac{N_0}{2}} = \Delta x^{[\nu]}_{j+\frac{N_0}{2}} \omega^{[\nu]}_{j+\frac{N_0}{2}} - \left((\widehat{c\omega})^{[\nu]}_{j+N_0} - (\widehat{c\omega})^{[\nu]}_{j}\right) \tag{19}$$

for $j = 0, N_0, 2N_0, \ldots, (N_1-1)N_0$, where $\Delta x_{j+\frac{N_0}{2}} = x_{j+N_0} - x_j$, and

$$(\widehat{c\omega})_j = \frac{c_j}{2}(\omega_{j,R} + \omega_{j,L}) - \frac{|c_j|}{2}(\omega_{j,R} - \omega_{j,L}), \tag{20}$$

here $c_j = x_j^{[\nu]} - x_j^{[\nu+1]}$, and

$$\omega_{j,L} = \omega_{j-\frac{N_0}{2}} + \frac{1}{2} S_{j-\frac{N_0}{2}}, \quad \omega_{j,R} = \omega_{j+\frac{N_0}{2}} - \frac{1}{2} S_{j+\frac{N_0}{2}}.$$

The slope limiter $S_{j\pm\frac{1}{2}}$ is an approximation of the derivative $\frac{\partial \omega}{\partial \xi}$ at $\xi = \xi_{j\pm\frac{1}{2}}$. In our computations, we employ the van Leer limiter [44]

$$S_{j+\frac{N_0}{2}} = \left(\mathrm{sign}\left(\Delta\omega_{j+\frac{N_0}{2}}\right) + \mathrm{sign}\left(\Delta\omega_{j-\frac{N_0}{2}}\right)\right) \frac{|\Delta\omega_{j+\frac{N_0}{2}} \Delta\omega_{j-\frac{N_0}{2}}|}{|\Delta\omega_{j+\frac{N_0}{2}}| + |\Delta\omega_{j-\frac{N_0}{2}}| + \varepsilon},$$

where $\Delta\omega_{j+\frac{N_0}{2}} = \omega_{j+\frac{3N_0}{2}} - \omega_{j+\frac{N_0}{2}}$, and $0 < \varepsilon \ll 1$ is used to avoid that the denominator becomes zero.

3.3. Local uniform refinement and remapping the solution

Assuming that the final adaptive coarse mesh $\{x_j^{[\mu]}, j = 0, N_0, 2N_0, \ldots, (N_1-1)N_0\}$ has been generated, we divide equally each coarse mesh interval or element $[x_j^{[\mu]}, x_{j+N_0}^{[\mu]}], j = 0, N_0, \ldots, (N_1-1)N_0$, into N_0 fine mesh cells and yield the fine grid points $\{x_i^{[\mu]}, i = j - N_0, j - N_0 + 1, \ldots, j-1\}, j = N_0, 2N_0, \ldots, N_1N_0$. As a result, we get an adaptive fine mesh of the physical domain Ω_p: $\{x_j^{[\mu]}\}_{j=0}^{N}$. Fig. 2 displays the procedure of the fine mesh

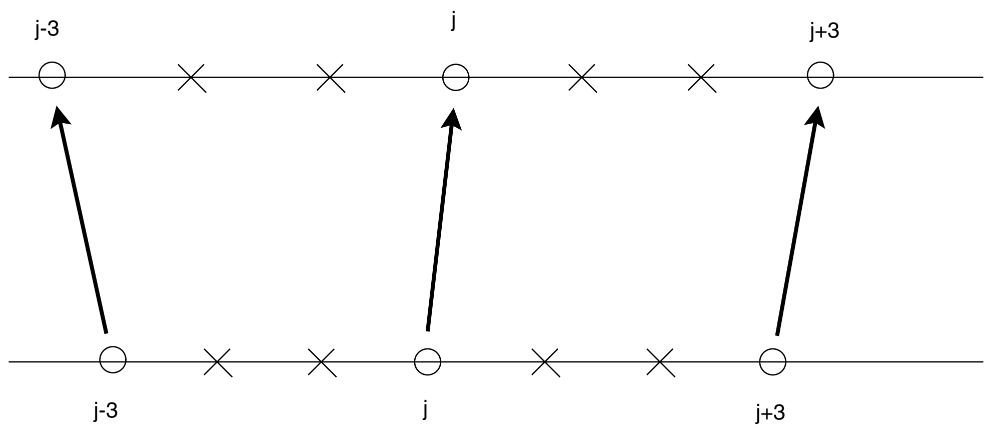


Fig. 2. The fine mesh redistribution with a local uniform refinement. $N_0 = 3$.

redistribution with a local uniform refinement, where $N_0 = 3$, symbol "○" denotes the coarse mesh point which is first redistributed by solving mesh redistribution equation iteratively, and symbol "×" is the mesh point obtained by a local uniform refinement. It is worth mentioning that our method may be considered as a combination of the *r*-refinement method and the *h*-refinement method and shares the same idea of the two-level mesh movement technique of Huang et al. [23,24] and Fiedler and Trapp [20] as well as the *hr*-refinement method in the literatures, see e.g. [1,2,30].

Now we need to remap the solution vector $\boldsymbol{u}$ from the old fine mesh $\{x_j^{[0]}\}_{j=0}^N := T^{[0]}$ to the new fine mesh $T^{[\mu]}$. Each cell of $T^{[\mu]}$ corresponds uniquely to a cell of $T^{[0]}$ by $x_j(\tau) = x_j^{[0]} + \tau\delta x_j^{[0]}$ where $\delta x_j^{[0]} := x_j^{[\mu]} - x_j^{[0]}$, $\tau \in [0,1]$. There is an affine map denoted by $x = x(\xi,\tau)$ between the two cells $[x_j^{[0]}, x_{j+1}^{[0]}]$ and $[x_j^{[\mu]}, x_{j+1}^{[\mu]}]$. The profile of $\boldsymbol{u}$ on Ω_p will not move, although the nodes of the mesh have been moved to new locations. Hence $\boldsymbol{u}$, as the function of x at a fixed time t, is independent on the parameter τ. That is

$$\frac{\partial \boldsymbol{u}}{\partial \tau} = 0, \quad \tau \in [0,1]. \tag{21}$$

During the movement of the mesh, $\boldsymbol{u}$ may be expressed as

$$\boldsymbol{u} = \boldsymbol{u}(x) = \boldsymbol{u}(x,\tau) = \boldsymbol{u}(x(\xi,\tau),\tau).$$

Integrating (21) over $[x_j(\tau), x_{j+1}(\tau)]$ gives

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\int_{x_j(\tau)}^{x_{j+1}(\tau)} \boldsymbol{u}\,\mathrm{d}x - (x_\tau \boldsymbol{u})_{j+1} + (x_\tau \boldsymbol{u})_j = 0, \quad \tau \in [0,1]. \tag{22}$$

This equation will be solved by a high-resolution shock-capturing scheme combined with a second-order Runge–Kutta time discretization, which is described in the subsequent section.

In Section 4, we will validate that the fine mesh redistribution with the local uniform refinement is more powerful and faster than the original adaptive mesh method introduced in [41].

3.4. The NLD solver on a fixed mesh

Assume that we have obtained the new mesh $T^{[\mu]} := \{x_j^{[\mu]}\}_{j=0}^N$. In the following, we solve (12) on the fixed non-uniform mesh $T^{[\mu]}$, that is to say, $x_j^{[\mu]}$ is independent on $t \in [t_n, t_n + \Delta t_n)$.

Integrating (12) over the control volume $[x_j^{[\mu]}, x_{j+1}^{[\mu]}]$ leads to the following semi-discrete finite volume method

$$\left(x_{j+1}^{[\mu]} - x_j^{[\mu]}\right)\frac{\mathrm{d}\boldsymbol{u}_{j+\frac{1}{2}}(t)}{\mathrm{d}t} = -(\hat{\boldsymbol{f}}_{j+1} - \hat{\boldsymbol{f}}_j) + \int_{x_j^{[\mu]}}^{x_{j+1}^{[\mu]}} \boldsymbol{s}(\boldsymbol{u})\,\mathrm{d}x, \tag{23}$$

where $\hat{\boldsymbol{f}}_j$ is some appropriate numerical flux satisfying

$$\hat{\boldsymbol{f}}_j = \hat{\boldsymbol{f}}(\boldsymbol{u}_{j,L}, \boldsymbol{u}_{j,R}), \quad \hat{\boldsymbol{f}}(\boldsymbol{u}, \boldsymbol{u}) = \boldsymbol{f}(\boldsymbol{u}). \tag{24}$$

An example of the numerical flux is the Lax–Friedrichs type flux:

$$\hat{\boldsymbol{f}}(\boldsymbol{v}, \boldsymbol{w}) = \frac{1}{2}[\boldsymbol{f}(\boldsymbol{w}) + \boldsymbol{f}(\boldsymbol{v}) - \sigma(\boldsymbol{w} - \boldsymbol{v})], \tag{25}$$

where $\sigma \geqslant \max_{\boldsymbol{u}}\left\{\left|\lambda\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)\right|\right\}$. In (24), $\boldsymbol{u}_{j,L}$ and $\boldsymbol{u}_{j,R}$ are defined by

$$\boldsymbol{u}_{j,L} = \boldsymbol{u}_{j-\frac{1}{2}} + \boldsymbol{S}_{j-\frac{1}{2}}\left(x_j^{[\mu]} - \frac{x_{j-1}^{[\mu]} + x_j^{[\mu]}}{2}\right), \tag{26}$$

$$\boldsymbol{u}_{j,R} = \boldsymbol{u}_{j+\frac{1}{2}} + \boldsymbol{S}_{j+\frac{1}{2}}\left(x_j^{[\mu]} - \frac{x_j^{[\mu]} + x_{j+1}^{[\mu]}}{2}\right). \tag{27}$$

The slope limiters $\boldsymbol{S}_{j+\frac{1}{2}} = \left(S^1_{j+\frac{1}{2}}, S^2_{j+\frac{1}{2}}, S^3_{j+\frac{1}{2}}, S^4_{j+\frac{1}{2}}\right)^T$ in (26) and (27) are defined by

$$S^i_{j+\frac{1}{2}} = \left(\operatorname{sign}\left(S^{i,+}_{j+\frac{1}{2}}\right) + \operatorname{sign}\left(S^{i,-}_{j+\frac{1}{2}}\right)\right)\frac{\left|S^{i,+}_{j+\frac{1}{2}}S^{i,-}_{j+\frac{1}{2}}\right|}{\left|S^{i,+}_{j+\frac{1}{2}}\right| + \left|S^{i,-}_{j+\frac{1}{2}}\right| + \varepsilon}, \quad i = 1, 2, 3, 4,$$

where $0 < \varepsilon \ll 1$ is used to avoid that the denominator becomes zero. Here,

$$\boldsymbol{S}^+_{j+\frac{1}{2}} = \frac{\boldsymbol{u}_{j+\frac{3}{2}} - \boldsymbol{u}_{j+\frac{1}{2}}}{x^{[\mu]}_{j+\frac{3}{2}} - x^{[\mu]}_{j+\frac{1}{2}}}, \quad \boldsymbol{S}^-_{j+\frac{1}{2}} = \frac{\boldsymbol{u}_{j+\frac{1}{2}} - \boldsymbol{u}_{j-\frac{1}{2}}}{x^{[\mu]}_{j+\frac{1}{2}} - x^{[\mu]}_{j-\frac{1}{2}}}.$$

Since

$$\int_{x_j^{[\mu]}}^{x_{j+1}^{[\mu]}} \boldsymbol{s}(\boldsymbol{u})\,\mathrm{d}x \approx \Delta x^{[\mu]}_{j+\frac{1}{2}}\boldsymbol{s}\left(\boldsymbol{u}_{j+\frac{1}{2}}\right). \tag{28}$$

Thus Eq. (23) can be further approximated as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{u}_{j+\frac{1}{2}} = -\frac{1}{\Delta x^{[\mu]}_{j+\frac{1}{2}}}\left(\hat{\boldsymbol{f}}_{j+1} - \hat{\boldsymbol{f}}_j\right) + \boldsymbol{s}\left(\boldsymbol{u}_{j+\frac{1}{2}}\right). \tag{29}$$

The semi-discrete scheme (29) gives a system of ordinary differential equations with respect to the unknown vector $\boldsymbol{u}$, which may be written in a matrix-operator form

$$\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} = \boldsymbol{L}(t, \boldsymbol{u}). \tag{30}$$

We apply an explicit second-order accurate Runge–Kutta method to discretization of the time derivative in (30) or (29). The Runge–Kutta method we consider is

$$\begin{cases} \boldsymbol{K}_1 = \Delta t_n \boldsymbol{L}(t_n, \boldsymbol{u}^n), \\ \boldsymbol{K}_2 = \Delta t_n \boldsymbol{L}(t_n + \frac{1}{2}\Delta t_n, \boldsymbol{u}^n + \frac{1}{2}\boldsymbol{K}_1), \\ \boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \boldsymbol{K}_2. \end{cases} \tag{31}$$

The semi-discrete MUSCL-type finite volume method (29) and (31), which is of second order accuracy in smooth regions in the sense of the truncation error, will be applied to (22) and the NLD Eq. (1) in our computations. When the above method is applied to (22), it is just marched forward in τ by one unit step size, i.e. $\Delta\tau = 1$. Note that $(x_\tau)_j$ and $x_j\left(\tau = \frac{1}{2}\right)$ are specified as

$$(x_\tau)_j = x_j^{[\mu]} - x_j^{[0]}, \quad x_j\left(\tau = \frac{1}{2}\right) = \frac{1}{2}\left(x_j^{[\mu]} + x_j^{[0]}\right).$$

The time step-size for the NLD equation (1) is determined by the CFL condition

$$\Delta t_n \leqslant cfl \frac{\min\left\{\Delta x^{[\mu]}_{j+\frac{1}{2}}\right\}}{\sigma}, \quad \max\left\{\left|\lambda\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right)\right|\right\} \leqslant \sigma, \tag{32}$$

where *cfl* denotes the CFL number.

3.5. Solution procedure

Our solution procedure is based on three parts: the NLD evolution, the iterative mesh redistribution of the coarse mesh, and the local uniform refinement of the final coarse mesh. That procedure can be illustrated by the following flowchart.

```
Algorithm 1

Step 1. Compute the cell average of the initial data $\left\{\boldsymbol{u}^0_{j+\frac{1}{2}}\right\}$ and give a uniform mesh $\{x^0_j\}_{j=0}^N$ at $t = 0$.
Step 2. For $j = 0, N_0, 2N_0, \ldots, N_1N_0$, define $x^{[0]}_j := x^n_j$, $\omega^{[0]}_{j+\frac{N_0}{2}} := \omega^n_{j+\frac{N_0}{2}}$, where $n \geqslant 0$. For $v = 0, 1, \ldots$, do the following:
    (a) Move the coarse mesh $\left\{x^{[v]}_j\right\}$ to $\left\{x^{[v+1]}_j\right\}$ by solving (16).
    (b) Compute $\left\{\omega^{[v+1]}_{j+\frac{N_0}{2}}\right\}$ by using (17) or (19).
    (c) Repeat the updating procedure (a) and (b) for a fixed number of iterations $\mu$ or until $\|x^{[v+1]} - x^{[v]}\| \leqslant \epsilon$ where $\|\cdot\|$ is the discrete norm on the coarse mesh.
Step 3. Divide equally the coarse mesh interval $\left[x^{[\mu]}_{j-N_0}, x^{[\mu]}_j\right]$ into $N_0$ fine cells, let $x^{n+1}_j := x^{[\mu]}_j$, and remap the solution $\boldsymbol{u}$ from the old fine mesh $\{x^n_j\}_{j=0}^N$ to the new fine mesh $\{x^{n+1}_j\}_{j=0}^N$ by the affine method (22).
Step 4. Evolve the Dirac equation using the high-resolution finite volume method on the fine mesh $\{x^{n+1}_j\}_{j=0}^N$.
Step 5. If$t_{n+1} < T$, then go to Step 2. Otherwise, output the result and stop.
```

4. Numerical experiments

This section will conduct numerical experiments to demonstrate the performance and efficiency of the adaptive mesh method proposed in the last section. All computations work in dimensionless units, or equivalently, take $m = 1$ and $\lambda = \frac{1}{2}$, and adopt the non-reflecting boundary conditions at artificial boundaries. The CFL number *cfl* and parameters N_1 and μ are taken as 0.5, 100 and 10, respectively, unless stated otherwise. Our codes are run on an IBM laptop (Pentium-M, 1.8 GHz) under the Linux environment. We define the grid density by $1/\Delta x_{j+1/2}$ to depict the grid quality.

For convenience, we will use the notations *Algorithm* 0 and *Algorithm* 1 to denote two different adaptive moving mesh methods, which are

- *Algorithm* 0 – the adaptive moving mesh method of Tang and Tang [41], in which the fine mesh is iteratively redistributed and the solution vector is remapped on the resulting new fine mesh $\left\{x^{[v+1]}_j\right\}_{j=1}^N$ from the old mesh $\left\{x^{[v]}_j\right\}_{j=1}^N$ in each iteration.

- *Algorithm* 1 – the present method with the local uniform refinement, in which the coarse mesh is just redistributed iteratively and the monitor function ω is remapped on the resulting new coarse mesh $\left\{x_j^{[v+1]}, j = 0, N_0, \ldots, N_1N_0\right\}$ from the old mesh $\{x_j^{[v]}, j = 0, N_0, \ldots, N_1N_0\}$ in each iteration. We yield the adaptive fine mesh $\left\{x_j^{[0]}\right\}_{j=1}^{N}$ of the physical domain Ω_p by the local uniform refinement.

Example 4.1 (*Single travelling solitary wave*). The first example is to simulate travel of a two-humped Dirac solitary wave. Since the exact single soliton solution (9) to the $(1+1)$-dimensional NLD equation (1) is known, we can compare our numerical solutions with the exact solution, and then evaluate the efficiency of our proposed algorithm. Here we take the monitor function (14) with $\alpha = 10$ and $\beta = 20$ and $\Lambda = 0.1$, $x_0 = -5$, $v = 0.1$. The physical domain Ω_p is considered as $[-25, 25]$.

Tables 1 and 2 give numerical errors at $t = 100$ and convergence rates for *Algorithm* 0 and *Algorithm* 1. Those estimated errors are defined by

$$l^1\text{-error} := \frac{1}{4}\sum_{i=1}^{4}\sum_{j}\left|(u_i^e)_{j+\frac{1}{2}} - (u_i^c)_{j+\frac{1}{2}}\right|(x_{j+1} - x_j),$$

$$l^2\text{-error} := \sqrt{\sum_{i=1}^{4}\sum_{j}\left|(u_i^e)_{j+\frac{1}{2}} - (u_i^c)_{j+\frac{1}{2}}\right|^2(x_{j+1} - x_j)},$$

$$l^\infty\text{-error} := \max_{i,j}\left\{\left|(u_i^e)_{j+\frac{1}{2}} - (u_i^c)_{j+\frac{1}{2}}\right|\right\},$$

where $(u_1^e, u_2^e, u_3^e, u_4^e)_{j+\frac{1}{2}}$ denote the cell averages of the exact solution $\boldsymbol{u}^e$ over $[x_j, x_{j+1}]$, while $(u_1^c, u_2^c, u_3^c, u_4^c)^T$ is the computed solution. We see that *Algorithm* 1 is more accurate and less time-consuming than *Algorithm* 0. Concretely, the accuracy of *Algorithm* 0 decreases as N increases, while *Algorithm* 1 gives a uniform third-order rate of convergence independent on N. The results show that *Algorithm* 1 may give a super-convergent solution and is not insensitive to smoothness and size of the mesh. By contraries, *Algorithm* 0 is strongly sensitive to smoothness and size of the mesh. Moreover, *Algorithm* 1 may save about 86% cost in the present computations, compared to *Algorithm* 0.

To further validate the performance and efficiency of *Algorithm* 1, we present the mesh densities (solid line) and the charge densities (black dot) in Figs. 3 and 4, obtained by using *Algorithm* 0 and *Algorithm* 1, where scale of the left axis is for the charge density and the right one is for the mesh density. From the left figure of Fig. 3, we see that the grid points at $t = 100$ redistributed by *Algorithm* 0 mainly cluster in the vicinity of the left peak of the Dirac solitary wave when the iterative tolerance μ equals to 10. Thus the mesh is not equally distributed. If we increase μ up to 30, the redistribution of the mesh points has been improved in the numerical computations, see the right figure of Fig. 3, but we have sacrificed quite a bit CPU time. The result given in the left figure of Fig. 4 shows that *Algorithm* 1 implements easily and fast the equidistribution principle, and the mesh density of *Algorithm* 1 preserves symmetry just as two peaks of the Dirac solitary wave are symmetric. So the quality of the adaptive mesh generated by *Algorithm* 1 is very perfect. The right plot of Fig. 4 gives the time

Table 1
Example 4.1. Numerical errors and convergence rates of *Algorithm* 0 at $t = 100$

N	100	200	400	800	1600	3200
l^1-error order	3.786	5.178e−1	7.565e−2	1.279e−2	3.465e−3	1.274e−03
	–	2.87	2.78	2.56	1.88	1.44
l^2-error order	2.992	4.089e−1	5.909e−2	9.840e−3	2.573e−3	9.477e−04
	–	2.87	2.79	2.59	1.94	1.44
l^∞-error order	1.004	1.309e−1	1.949e−2	3.670e−3	1.077e−3	3.979e−04
	–	2.94	2.75	2.41	1.77	1.44
CPU time (s)	6.11	21.46	83.76	335.69	1325.50	4940.00

Table 2
Example 4.1. Numerical errors and convergence rates of *Algorithm* 1 at $t = 100$

N	100	200	400	800	1600	3200
l^1-error order	3.782	4.725e−1	5.848e−2	7.331e−3	9.293e−4	1.184e−04
	–	3.00	3.01	3.00	2.98	2.97
l^2-error order	2.989	3.715e−1	4.504e−2	5.463e−3	6.597e−4	7.775e−05
	–	3.01	3.04	3.04	3.05	3.08
l^∞-error order	1.003	1.156e−1	1.435e−2	1.775e−3	2.177e−4	2.858e−05
	–	3.12	3.01	3.02	3.03	2.93
CPU time (s)	1.04	3.43	12.20	48.89	183.70	707.75

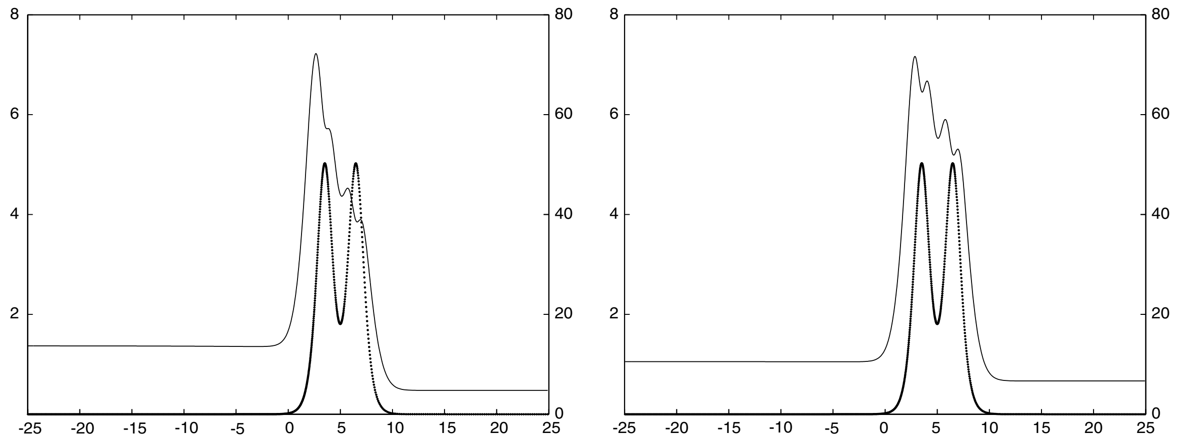


Fig. 3. Example 4.1. The mesh densities (solid line) and charge densities (black dot) at $t = 100$ obtained by using *Algorithm* 0. $\Lambda = 0.5$, $v = 0.1$, $x_0 = -5$ and $N = 800$. Left: $\mu = 10$; right: $\mu = 30$.

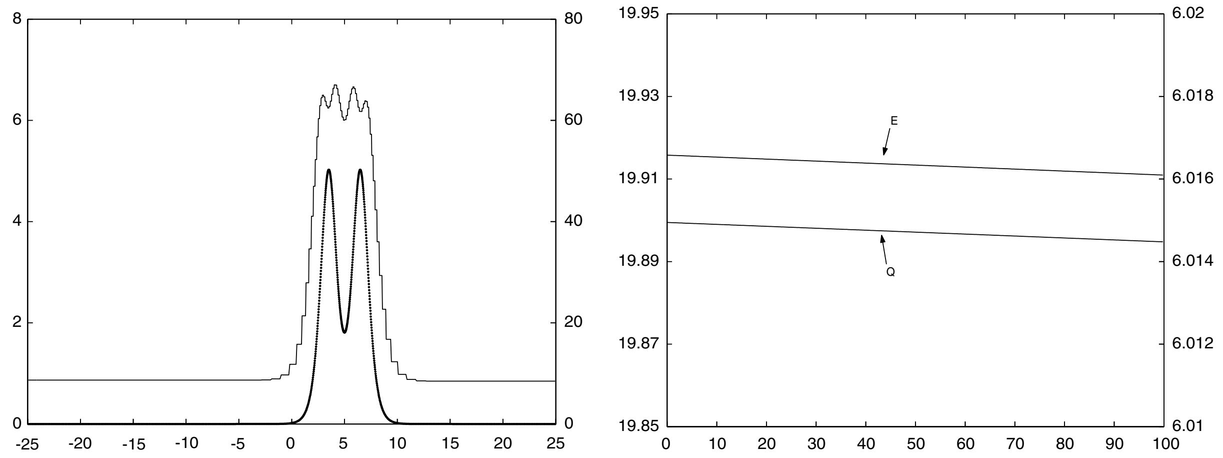


Fig. 4. Example 4.1. The results given by *Algorithm* 1. $\Lambda = 0.5$, $v = 0.1$, $x_0 = -5$, $N = 800$ and $\mu = 10$. Left: the charge density (block dot) and the mesh density (solid line) at $t = 100$; right: the time evolution of the charge Q and energy E.

evolution of the charge Q and energy E, where scale of the left axis is for Q and the right one is for E, and shows that they are approximately conservative, because their relative errors are 0.02% and 0.008%, respectively.

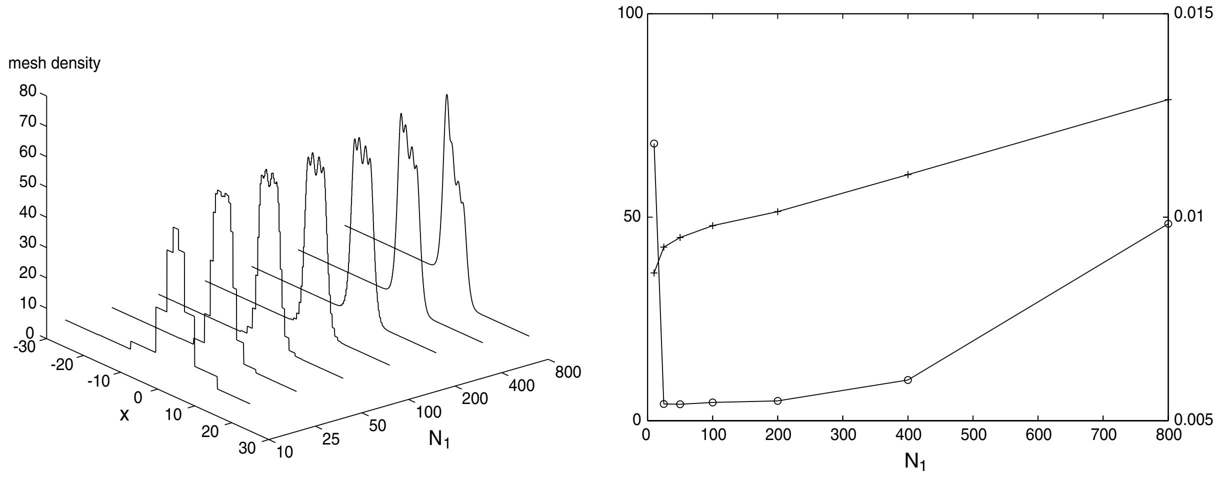


Fig. 5. Example 4.1. Dependence of *Algorithm* 1 on N_1. $\Lambda = 0.5$, $v = 0.1$, $x_0 = -5$, $N = 800$ and $t = 100$. Left: the mesh density; right: the CPU time (+) and the l^2-error (○).

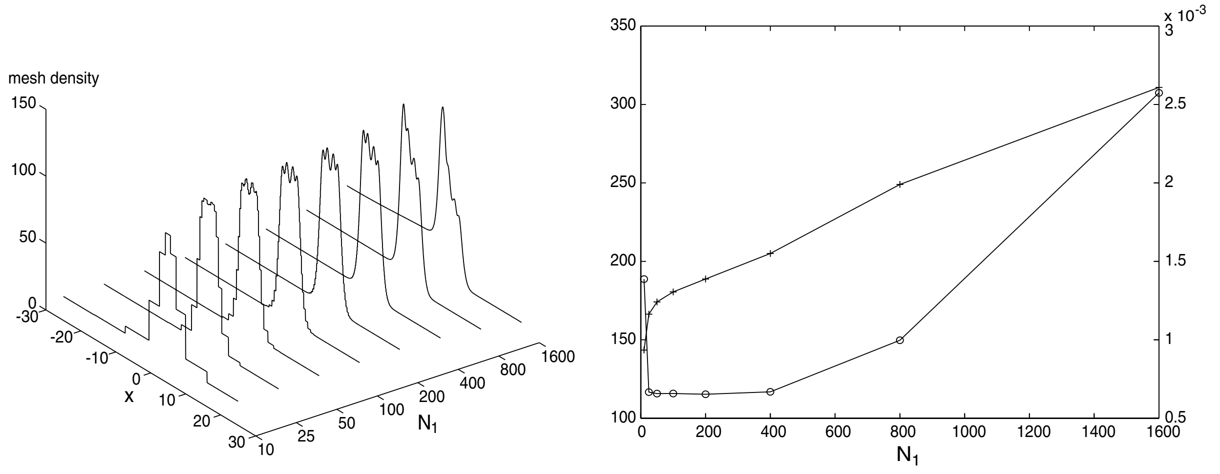


Fig. 6. Example 4.1. Same as Fig. 5, except for $N = 1600$.

Figs. 5 and 6 give dependence of the mesh density, the CPU time ("+", right axis) and the l^2-error ("○", left axis) of *Algorithm* 1 on the parameter N_1, where $t = 100$ and $N = 800$ and 1600, respectively. The results show that the recorded CPU time and l^2-error are monotonically increasing and convex functions with respect to N_1, respectively; the mesh quality may be improved very well when N_1 decreases properly. When $N_1 \in [25, 400]$, at least in the present example, *Algorithm* 1 is almost optimal, that is to say, its CPU time and error are the relatively lowest, and the mesh quality is the best. It is worth noting that the optimal choice of the parameter N_1 is not too sensitive with respect to N.

Example 4.2 (*Binary collisions*). The second example is to investigate collisions of two one-humped solitary waves with a phase shift of π, that is to say, we solve (1) subject to the initial data

$$\boldsymbol{\psi}(x,0) = \boldsymbol{\psi}^{\text{ss}}(x - x_l, 0) - \boldsymbol{\psi}^{\text{ss}}(x - x_r, 0) \tag{33}$$

with $\Lambda_l = \Lambda_r = 0.5$, $v_l = 0.1$, $v_r = -0.9$ and $x_r = -x_l = 10$. Extensive studies of binary collisions of Dirac solitary waves have been conducted in [36,37]. Here we take the physical domain $\Omega_p = [-40, 40]$, $\alpha = 10$, $\beta = 20$ and $N = 800$. For comparison, Fig. 7 gives the time evolution of the charge density ρ_Q and the total energy E

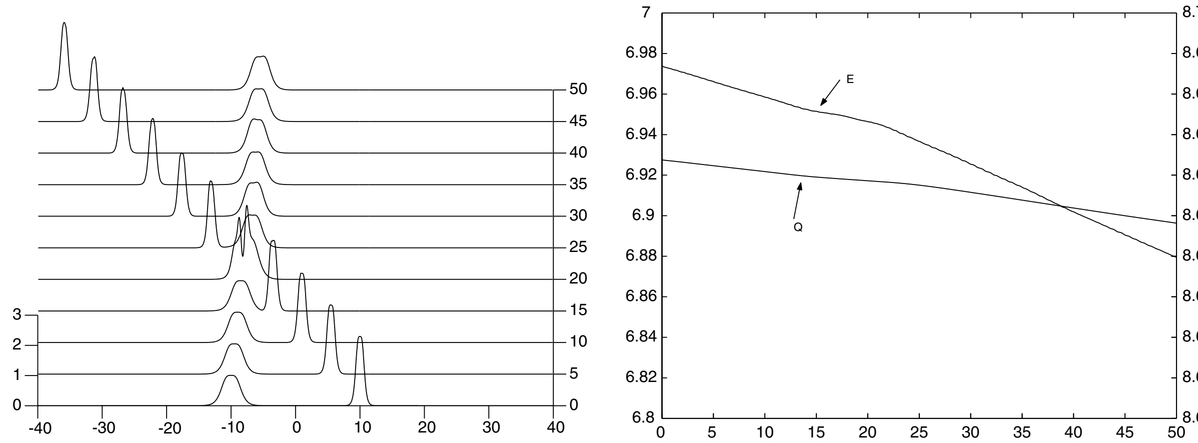


Fig. 7. Example 4.2. The time evolution of the charge density ρ_Q (left) and the total energy E and charge Q (right), obtained by *Algorithm* 1.

as well as the charge Q obtained by using *Algorithm* 1. The results show clearly the interaction dynamics of two Dirac solitary waves, and are comparable with ones obtained by using a higher-order RKDG method on a very fine uniform mesh, see Fig. 4 in [37]. For comparison, Fig. 8 gives charge densities (black dot) and grid densities (solid line) at $t = 50$ obtained by using *Algorithm* 0 (left) and *Algorithm* 1 (right). We see that the mesh quality of *Algorithm* 0 is much worse than that of *Algorithm* 1, because grid points clustering near the left peak (which is sharp) are not enough. As a result, the left peak, see left figure of Fig. 8, is lower and moving more slowly than one in right figure. The CPU times are 140 s for *Algorithm* 0 and 45 s for *Algorithm* 1, respectively. It means that about 67.86% cost is saved in the computation by using *Algorithm* 1.

Example 4.3 (*Ternary collisions*). The third example is to consider collisions of three in-phase Dirac solitary waves. The initial data are specified as follows:

$$\boldsymbol{\psi}(x,0) = \boldsymbol{\psi}^{\mathrm{ss}}(x - x_l, 0) + \boldsymbol{\psi}^{\mathrm{ss}}(x - x_m, 0) + \boldsymbol{\psi}^{\mathrm{ss}}(x - x_r, 0) \tag{34}$$

with $\Lambda_l = \Lambda_r = 0.9$, $\Lambda_m = 0.1$, $v_l = -v_r = 0.9$ and $v_m = 0$. That means that the initial waves are in phase, and the left and right waves are one-humped and the middle one is two-humped. We take the physical domain $\Omega_p = [-25, 25]$, $\alpha = \beta = 10$ and $N = 1600$.

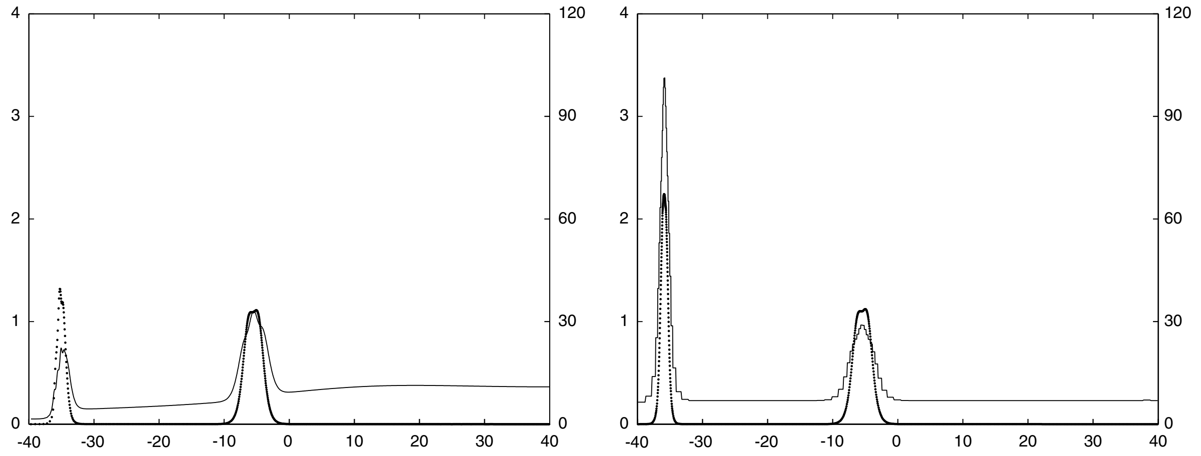
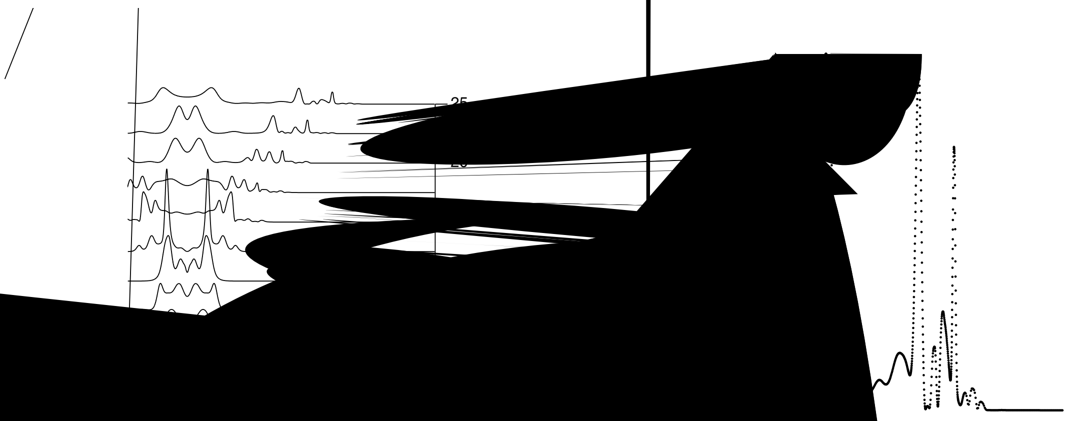


Fig. 8. Example 4.2. The charge densities (black dot) and the grid densities (solid line) at $t = 50$. Left: *Algorithm* 0; right: *Algorithm* 1.

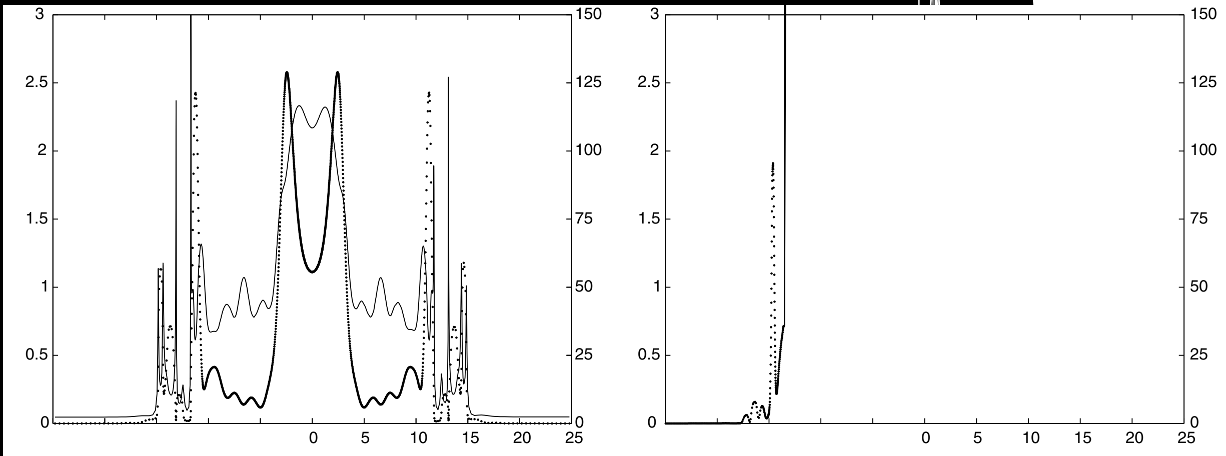
Fig. 9 shows the time evolution of the charge density ρ_Q obtained by *Algorithm* 1, and a comparison of the charge densities at $t = 25$ obtained by using *Algorithm* 1 with $N = 1600$ (black dots) and the uniform mesh method with $N = 10{,}000$ (solid line). We see that some small waves are generated after the main interaction around $t = 10$ so that the collapse phenomenon happens. But the solution of *Algorithm* 1 is still in accordance with the uniform fine mesh solution.

To compare *Algorithm* 0 with *Algorithm* 1, we present the mesh densities and the charge densities at $t = 25$ in Fig. 10, where small black dots stand for the charge density (left axis), and solid line presents mesh density (right axis). From the left figure of Fig. 10, we see that the mesh density of *Algorithm* 0 becomes quite a bit singular around $x = -13$, -11 and 13, and non-symmetric. The mesh points are clustered overfull in the interval $[-10, 10]$. The right figure of Fig. 10 shows that the mesh density of *Algorithm* 1 is approximately symmetric, and much smoother and better than that of *Algorithm* 0.

To further validate the proposed method, we present two close-ups of the charge densities in Fig. 11, in which symbol "×", block dot, and solid line stand for the results obtained by using *Algorithm* 0, *Algorithm* 1, and the uniform mesh method with $N = 10{,}000$, respectively. The left plot of Fig. 11 shows detailed wavelets moving to the left boundary of the physical domain. The right plot in Fig. 11 shows details of the peaks of the middle two-humped wave. From the left figure of Fig. 11, we see that *Algorithm* 0 cannot resolve small



Example 4.3 e density of *Algorithm* 1; 10,000 (solid line).



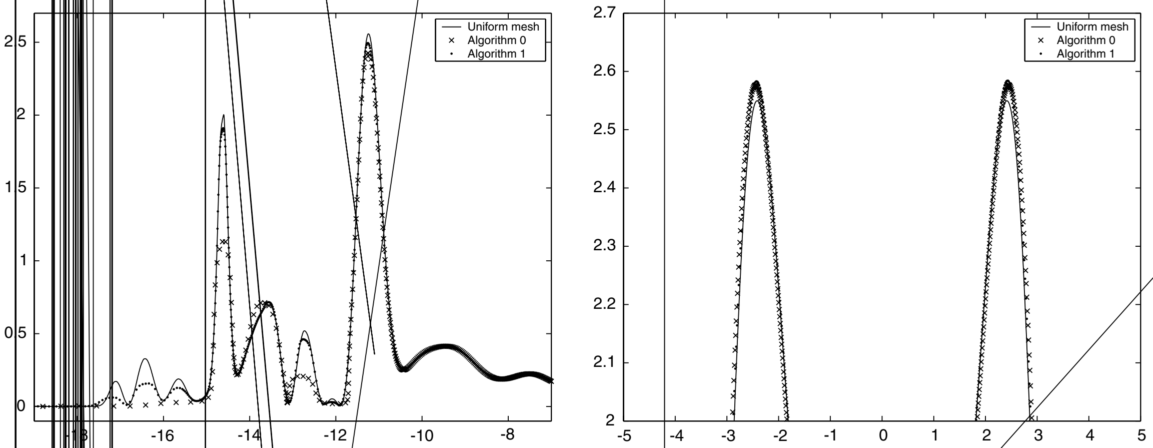


Fig. 11. Example 4.3. Close up of the charge densities obtained by using *Algorithm* 0 (×), *Algorithm* 1 (·), and the uniform mesh method with $N = 10{,}000$ (solid line).

oscillatory waves. Although the mesh of *Algorithm* 1 in the interval $[-10, 10]$ is sparser than that of *Algorithm* 0, see Fig. 10, but their solutions are accordant in that interval, see the right figure of Fig. 11. Thus it is unnecessary to cluster too many mesh points there. Even though *Algorithm* 1 damps the wavelets, its results are in accordance with the results obtained on a uniform mesh with $N = 10{,}000$. The CPU times are 328.45 s for *Algorithm* 0 and 80.60 s for *Algorithm* 1, respectively.

Example 4.4 (*Quadruple collisions*). The final example is to study quadruple collisions of the Dirac waves. The initial data are given as

$$\boldsymbol{\psi}(x,0) = -\boldsymbol{\psi}^{\rm ss}(x-x_l,0) + \boldsymbol{\psi}^{\rm ss}(x-x_{lm},0) + \boldsymbol{\psi}^{\rm ss}(x-x_{rm},0) - \boldsymbol{\psi}^{\rm ss}(x-x_r,0) \tag{35}$$

with $v_l = v_{lm} = v_{rm} = v_r = 0$, $\Lambda_l = \Lambda_r = \Lambda_{lm} = \Lambda_{rm} = 0.5$, $x_r = -x_l = 15$ and $x_{rm} = -x_{lm} = 5$.

We take the physical domain $\Omega_p = [-50, 50]$, $\alpha = \beta = 10$ and $N = 1200$, Fig. 12 gives the time evolution of the charge density ρ_Q obtained by *Algorithm* 1, and a comparison of the charge densities at $t = 25$ obtained by using *Algorithm* 1 with $N = 1200$ (black dots) and the uniform mesh method with $N = 10{,}000$ (solid line). The results show clearly the interaction dynamics of four Dirac solitary waves: four main interactions of the Dirac

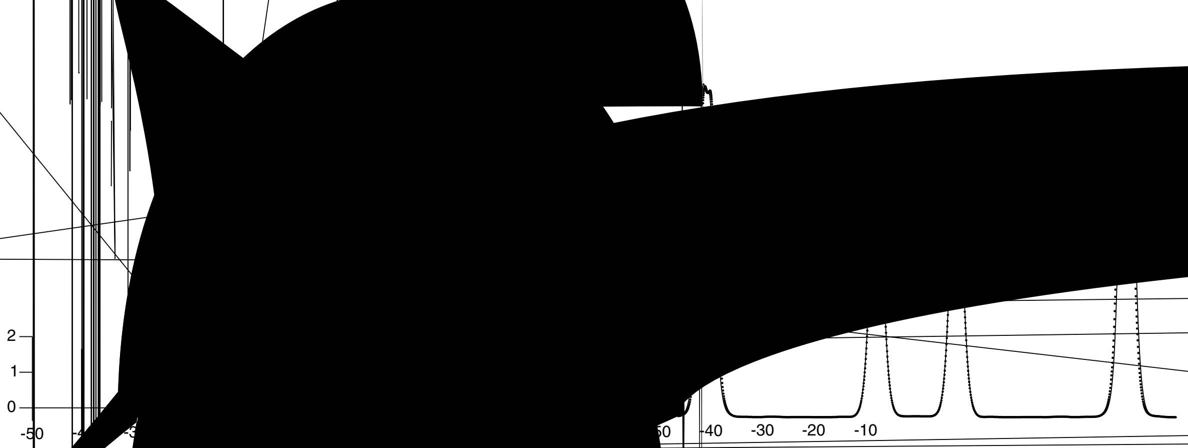


Fig. 12. Example 4.4 [illegible] Λ_{rm} [illegible] 5. Left: The time evolution of the charge density of *Algorithm* 1; right: The cha[illegible]1200 (black dots) and the uniform mesh method with $N = 10{,}000$ (solid line).

waves happen around $t = 50$, 120 and 180, respectively. In connection with each main interaction, the overlap happens, in other words, the interactions are all inelastic. The right figure of Fig. 12 shows that the solution of *Algorithm* 1 is in accord with the uniform fine mesh solution. The CPU times are 472.93 s for *Algorithm* 0 and 79.81 s for *Algorithm* 1, respectively. It means that about 85.65% cost is saved in the computation by using *Algorithm* 1.

5. Remarks and conclusions

In this paper, we have proposed a highly efficient adaptive mesh method for solving the $(1 + 1)$-dimensional non-linear Dirac (NLD) equation (1). The algorithm was formed by three main parts: the NLD evolution, the iterative redistribution of the coarse mesh, and the local uniform refinement of the final coarse mesh as well as the remapping phase of the solution vector.

At each time level, the equidistribution principle was first employed to iteratively redistribute coarse mesh points. In the iterative coarse mesh redistribution, the scalar monitor function was interpolated on the coarse mesh instead of the solution vector, which was considered in [41]. After the final adaptive coarse mesh was generated ideally, each coarse mesh interval was equally divided into some uniform fine cells to give the adaptive fine mesh of the physical domain. Then the solution vector was remapped from the old fine mesh to the new fine mesh by marching the remapping solver forward in the parameter or pseudo-time by one unit step size. Finally, the governing equations were solved by using a high resolution shock-capturing method over a fixed quadrate control volume in the space and time domain. Our proposed method may be considered as a combination of the *r*-refinement method and the *h*-refinement method and shares the same idea of the two-level mesh movement technique of Huang et al. [23,24] and Fiedler and Trapp [20] as well as the *hr*-refinement method in the literatures.

Extensive numerical experiments have been presented to demonstrate that the proposed adaptive mesh method is much more efficient and faster than the method of [41], may give third-order rates of convergence and yields a powerful and fast NLD solver that tracks and resolves both small, local and large solution gradients automatically. It is worth noting that quadruple collisions of the Dirac solitary waves are studied for the first time.

In future, we will extend the present method to multidimensional Dirac model and conduct research in theoretical and applied analysis of the adaptive mesh methods.

Acknowledgments

This research was partially supported by the National Basic Research Program under the Grant 2005CB321703, the National Natural Science Foundation of China (No. 10431050, 10576001), and Laboratory of Computational Physics. HZT would like to thank Mr. Sihong Shao for numerous discussions during the preparation of this work. We would also like to thank anonymous referees for many helpful suggestions.

References

[1] S. Adjerid, J.E. Flaherty, A moving-mesh finite element method with local refinement for parabolic partial differential equations, Comput. Methods Appl. Mech. Eng. 55 (1986) 3–26.

[2] S. Adjerid, J.E. Flaherty, A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations, SIAM J. Numer. Anal. 23 (1986) 778–796.

[3] A. Alvarez, Linearized Crank–Nicholson scheme for nonlinear Dirac equations, J. Comput. Phys. 99 (1992) 348–350.

[4] A. Alvarez, Spinorial solitary wave dynamics of a $(1 + 3)$-dimensional model, Phys. Rev. D 31 (1985) 2701–2703.

[5] A. Alvarez, B. Carreras, Interaction dynamics for the solitary waves of a nonlinear Dirac model, Phys. Lett. A 86 (1981) 327–332.

[6] A. Alvarez, A.F. Rañda, Blow-up in nonlinear models of extended particles with confined constituents, Phys. Rev. D 38 (1988) 3330–3333.

[7] A. Alvarez, M. Soler, Energetic stability criterion for a nonlinear spinorial model, Phys. Rev. Lett. 50 (1983) 1230–1233.

[8] A. Alvarez, M. Soler, Stability of the minimum solitary wave of a nonlinear spinorial model, Phys. Rev. D 34 (1986) 644–645.

[9] B.N. Azarenok, S.A. Ivanenko, T. Tang, Adaptive mesh redistribution method based on Godunov's scheme, Commun. Math. Sci. 1 (2003) 152–179.
[10] G. Beckett, J.A. Mackenzie, M.L. Robertson, An *r*-adaptive finite element method for the solution of the two-dimensional phase-field equations, Commun. Comput. Phys. 1 (2006) 805–826.
[11] J.U. Brackbill, An adaptive grid with directional control, J. Comput. Phys. 108 (1993) 38–50.
[12] J.U. Brackbill, J.S. Saltzman, Adaptive zoning for singular problems in two dimensions, J. Comput. Phys. 46 (1982) 342–368.
[13] W.M. Cao, W.Z. Huang, R.D. Russell, A study of monitor functions for two-dimensional adaptive mesh generation, SIAM J. Sci. Comput. 20 (1999) 1978–1999.
[14] W.M. Cao, W.Z. Huang, R.D. Russell, An *r*-adaptive finite element method based upon moving mesh PDEs, J. Comput. Phys. 149 (1999) 221–244.
[15] H.D. Ceniceros, T.Y. Hou, An efficient dynamically adaptive mesh for potentially singular solutions, J. Comput. Phys. 172 (2001) 609–639.
[16] S.F. Davis, J.E. Flaherty, An adaptive finite element method for initial-boundary value problems for partial differential equations, SIAM J. Sci. Stat. Comput. 3 (1982) 6–27.
[17] J. De Frutos, J.M. Sanz-serna, Split-step spectral schemes for nonlinear Dirac systems, J. Comput. Phys. 83 (1989) 407–423.
[18] A.S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, J. Comput. Phys. 95 (1991) 450–476.
[19] R. Fazio, R. LeVeque, Moving-mesh methods for one-dimensional hyperbolic problems using CLAWPACK, Comp. Math. Appl. 45 (2003) 273–298.
[20] B.H. Fiedler, R.J. Trapp, A fast dynamic grid adaption scheme for meteorological flows, Mon. Weather Rev. 121 (1993) 2879–2888.
[21] A. Harten, J.M. Hyman, Self-adjusting grid methods for one-dimensional hyperbolic conservation laws, J. Comput. Phys. 50 (1983) 235–269.
[22] J.L. Hong, C. Li, Multi-symplectic Runge–Kutta methods for nonlinear Dirac equations, J. Comput. Phys. 211 (2006) 448–472.
[23] W.Z. Huang, Practical aspects of formulation and solution of moving mesh partial differential equations, J. Comput. Phys. 171 (2001) 753–755.
[24] J. Lang, W.M. Cao, W.Z. Huang, R.D. Russell, A two-dimensional moving finite element method with local refinement based on a posteriori error estimates, Appl. Numer. Math. 46 (2003) 75–94.
[25] R. Li, T. Tang, P.W. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, J. Comput. Phys. 170 (2001) 562–588.
[26] R. Li, T. Tang, P.W. Zhang, A moving mesh finite element algorithm for singular problems in two and three space dimensions, J. Comput. Phys. 177 (2002) 365–393.
[27] S. Li, L. Petzold, Moving mesh methods with upwinding schemes for time-dependent PDEs, J. Comput. Phys. 131 (1997) 368–377.
[28] F. Liu, S. Ji, G. Liao, An adaptive grid method and its application to steady Euler flow calculations, SIAM J. Sci. Comput. 20 (1998) 811–825.
[29] K. Miller, R.N. Miller, Moving finite element. I, SIAM J. Numer. Anal. 18 (1981) 1019–1032.
[30] J.T. Oden, T. Strouboulis, P. Devloo, Adaptive finite element methods for the analysis of inviscid compressible flow. I. Fast refinement/unrefinement and moving mesh methods for unstructured meshes, Comput. Methods Appl. Mech. Eng. 59 (1986) 327–362.
[31] A.F. Rañada, M. Soler, Perturbation theory for an exactly soluble spinor model in interaction with its electromagnetic field, Phys. Rev. D 8 (1973) 3430–3433.
[32] A.F. Rañada, M.F. Rañada, M. Soler, L. Vázquez, Classical electrodynamics of a nonlinear Dirac field with anomalous magnetic moment, Phys. Rev. D 10 (1974) 517–525.
[33] W.Q. Ren, X.P. Wang, An iterative grid redistribution method for singular problems in multiple dimensions, J. Comput. Phys. 159 (2000) 246–273.
[34] K. Saleri, S. Steinberg, Flux-corrected transport in a moving grid, J. Comput. Phys. 111 (1994) 24–32.
[35] S.H. Shao, H.Z. Tang, Higher-order accurate Runge–Kutta discontinuous Galerkin methods for a nonlinear Dirac model, Discrete Cont. Dyn. Syst. Ser. B 6 (2006) 623–640.
[36] S.H. Shao, H.Z. Tang, Interaction for the solitary waves of a nonlinear Dirac model, Phys. Lett. A 345 (2005) 119–128.
[37] S.H. Shao, H.Z. Tang, Interaction for solitary waves with a phase difference in a nonlinear Dirac model, *ArXiv: nlin.SI/0604033v1*, 2006. Available from: <http://arxiv.org/pdf/nlin.SI/0604033>.
[38] M. Soler, Classical, stable, nonlinear spinor field with positive rest energy, Phys. Rev. D 1 (1970) 2766–2769.
[39] J.M. Stockie, J.A. Mackenzie, R.D. Russell, A moving mesh method for one-dimensional hyperbolic conservation laws, SIAM J. Sci. Comput. 22 (2001) 1791–1813.
[40] H.Z. Tang, T. Tang, Multi-dimensional moving mesh methods for shock computations, Contemp. Math. 330 (2003) 169–183.
[41] H.Z. Tang, T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, SIAM J. Numer. Anal. 41 (2003) 487–515.
[42] H.Z. Tang, T. Tang, P.W. Zhang, An adaptive mesh redistribution method for nonlinear Hamilton–Jacobi equations in two- and three-dimensions, J. Comput. Phys. 188 (2003) 534–572.
[43] T. Tang, Moving mesh methods for computational fluid dynamics, Contemp. Math. 383 (2005) 141–174.
[44] B. van Leer, Towards the ultimate conservative difference scheme, V. A second order sequel to Godunov's method, J. Comput. Phys. 32 (1979) 101–136.
[45] D.S. Wang, X.P. Wang, A three-dimensional adaptive method based on the iterative grid redistribution, J. Comput. Phys. 199 (2004) 423–436.

[46] Z.-Q. Wang, B.-Y. Guo, Modified Legendre rational spectral method for the whole line, J. Comput. Math. 22 (2004) 457–474.
[47] A. Winslow, Numerical solution of the quasi-linear Poisson equation, J. Comput. Phys. 1 (1967) 149–172.
[48] P.A. Zegeling, On resistive MHD models with adaptive moving meshes, J. Sci. Comput. 24 (2005) 263–284.
[49] P.A. Zegeling, W.D. de Boer, H.Z. Tang, Robust and efficient adaptive moving mesh solution of the 2-D Euler equations, Contemp. Math. 383 (2005) 375–386.
[50] Z.R. Zhang, Moving mesh method with conservative interpolation based on L2-projection, Commun. Comput. Phys. 1 (2006) 930–944.