Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2016.m1229

Vol. **9**, No. 4, pp. 686-704
November 2016

Finite Element Simulations with Adaptively Moving Mesh for the Reaction Diffusion System

Congcong Xie[1] and Xianliang Hu[2,*]

[1] *College of Science, Zhejiang University of Technology, Hangzhou 310023, China*
[2] *School of Mathematical Science, Zhejiang University, Hangzhou 310027, China*

Received 12 September 2012; Accepted (in revised version) 20 July 2016

Abstract. A moving mesh method is proposed for solving reaction-diffusion equations. The finite element method is used to solving the partial different equation system, and an efficient numerical scheme is applied to implement mesh moving. In the practical calculations, the moving mesh step and the problem equation solver are performed alternatively. Several numerical examples are presented, including the Gray-Scott, the Activator-Inhibitor and a case with a growing domain. It is illustrated numerically that the moving mesh methods costs much lower, compared with the numerical schemes on a fixed mesh. Even in the case of complex pattern dynamics described by the reaction-diffusion systems, the adapted meshes can capture the details successfully.

AMS subject classifications: 65M60, 65N50
Key words: moving mesh; finite elements; reaction-diffusion; growing domain; pattern formation.

1. Introduction

The pattern formation modeled with the reaction-diffusion systems are interested in various research fields, such as chemistry, biology, ecology and biochemistry [16–19]. It describes the interplay between several "species" when reactions among them are considered. The reaction terms are always nonequilibrium and nonlinear, and the diffusion effects are modeled with the Laplacian. Numerical simulation is one of the main strategies for practical investigations, see for e.g., [2, 6, 9, 11]. However, some numerical schemes may lead to plausible but non-physical solutions due to the presence of the nonlinearity [24]. High resolution numerical methods, including either high order schemes or sufficiently refined meshes, are capable of avoiding such miscalculations.

The adaptive numerical methods are commonly used in numerical simulations, which contain two main topics, the discretization method and the adaptive method.

*Corresponding author. *Email addresses:* xlhu@zju.edu.cn (X. Hu), ccxie@zjut.edu.cn (C. Xie)

 ©2016 Global-Science Press

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

The Finite Difference Methods (FDM) and the Finite Element Methods (FEM) are both popular for discretization [10, 15], however, the FEM win a bit due to its better flexibility for complex topological geometries and the convenience in performing adaptive calculations. So that the FEM are preferred as the basic discretization method in this research. The mesh adaptivity is constantly used for better efficiency of the finite element calculations. The Adaptive Mesh Moving (AMM) and the Adaptive Mesh Refinement (AMR) are two different choices in the region of numerical simulations. While AMM plays an growing role in the last decade [7,13,14,20,24,25]. In this research, the concerned small scale structure for pattern formation always appears locally in a moving space region and/or periodically in time, then we prefer the moving mesh strategy for the purpose of lower costs.

The fundamental idea of the AMM is to find a minimization of a so-called mesh-energy functional. Then different methodology for finding the minimization yield different moving mesh models. Huang [8] proposed the so-called Moving Mesh Partial Differential Equations (MMPDE) based on the gradient flow of the functional, and also provided several different types of MMPDEs, which is suitable for different type of physical models. Li *et al.* [12] suggest to find the minimization of the mesh-energy functional directly, and rigorous proof for the existence and convergence are given based the harmonic mapping theory. It is robust and effective when applied in different finite element applications [7, 22]. The corresponding implementation is also kindly shared by the authors under an open source policy. There are several further different variants for the moving mesh equations by other authors [4, 20]. In general, the linear systems arising from the discretization of the moving mesh models is required to be solved accurately enough for archiving the convergence. However, the efficiency is the mainly concerned issue for the numerical methods for mesh moving in this research, so that a simpler but efficient version moving mesh equation solver is applied.

Moving mesh finite element calculations are investigated for the reaction-diffusion system in this paper. The rest of this paper is organized as following. In the next section, the finite element scheme on a fixed mesh are introduced for the reaction-diffusion systems. In Section 3, we explain our numerical scheme for the moving mesh equations. Numerical examples are illustrated, and comparisons are made between with and without the moving mesh approach. Some observation and conclusions are made at the last section.

2. The finite element formulations of model problem

2.1. The reaction-diffusion systems

There are different types of classical reaction-diffusion systems in the pattern formation theory: Brusselator, Gray-Scott, Activator-Inhibitor and Schnakenberg etc., which has only slight differences between them on some parameters. Considering two different species living in two dimensional fixed domain Ω, let scalar functions $u(t, x, y)$ and $v(t, x, y)$ be their densities respectively, where t is the temporal variable and x, y are

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

spatial variables for two space dimensions, then the evolution of two species, say u and v, can be describes with a two equation system

$$\begin{cases} u' = \epsilon_1 \triangle u + f(u, v), \\ v' = \epsilon_2 \triangle v + g(u, v), \end{cases} \tag{2.1}$$

where $'$ means the first order derivative with respect to t and $\triangle$ is the Laplacian operator with respect to the spatial dimensions. Two scalar parameters ϵ_1 and ϵ_2 mean the diffusion rates for different species, which are always small in their absolute values. It is known that the above reaction-diffusion system models different behaviors in different configurations of $f(u, v)$ and $g(u, v)$, which describe the interplay between the two species in a nonlinear fashion. In this research, the Gray-Scott and the Activator-Inhibitor are considered as the examples, and we leave their specification to the section of numerical examples.

Proper boundary conditions and initial conditions should be given to close the equation system (2.1). Generally, there are two different choices for different biological problems, namely, the periodic boundary and the homogeneous Neumann boundary. The periodic one is appropriate to describe closed boundaries, while the homogeneous Neumann boundary are for self-organization process. According to our test problems, the later one is used in the implementation, say

$$\frac{\partial u}{\partial \nu} = 0, \quad \frac{\partial v}{\partial \nu} = 0. \tag{2.2}$$

It is also worth to remark that our moving mesh framework is capable to accommodate the periodic boundary conditions as usual finite element calculations. The initial condition of u and v are given by $u_0(x, y)$ and $v_0(x, y)$ respectively, which will also be specified in different numerical examples.

The weak solution theory for the PDEs is fundamental for both the analysis and computation, and it is also the source of the finite element formulations for the considered problem. Let Ω be the prescribed domain, then the standard Sobolev spaces, $H^1(\Omega)$ and $H_0^1(\Omega)$, are then well-defined on Ω. The weak form for the above reaction-diffusion system is to find a pair of solution $u, v \in H^1(\Omega)$, which satisfying

$$\begin{cases} \langle u', p\rangle = -\epsilon_1 \langle \nabla u, \nabla p\rangle + \langle f(u, v), p\rangle, & \forall p \in H^1(\Omega), \\ \langle v', q\rangle = -\epsilon_2 \langle \nabla v, \nabla q\rangle + \langle g(u, v), q\rangle, & \forall q \in H^1(\Omega), \end{cases} \tag{2.3}$$

and proper boundary conditions, where $\langle \cdot, \cdot \rangle$ means the standard L_2 inner product. For more details on the weak solution theory for PDEs, people are referred to [1].

2.2. The finite element discretization

The finite element methods are not new for the partial differential equations, and the contributions on this subject are abundant. We just describe the main ideas briefly

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

for the ease of reference, people are referred to the classical FEM textbooks for a better understanding on this subject, such as [3, 5, 21]. Here let us introduce some basic notations. Denote $\triangle$ as the mesh for Ω, and $\mathcal{P}(\triangle) \in H^1(\Omega)$ as the standard bilinear finite element spaces defined on Ω. Following the standard technique of Galerkin finite element methods, it is desired to find a finite element approximation in a weak sense, that is, $u_h \in \mathcal{P}(\triangle)$ and $v_h \in \mathcal{P}(\triangle)$, which satisfy the discretized weak formulations

$$\begin{cases} \langle u_h', \phi_h \rangle = -\epsilon_1 \langle \nabla u_h, \nabla \phi_h \rangle + \langle f(u_h, v_h), \phi_h \rangle, \\ \langle v_h', \phi_h \rangle = -\epsilon_2 \langle \nabla v_h, \nabla \phi_h \rangle + \langle g(u_h, v_h), \phi_h \rangle, \end{cases} \quad \forall \phi_h \in \mathcal{P}(\triangle) \cap H^1(\Omega). \tag{2.4}$$

Assume that $\mathcal{P}(\triangle)$, sometimes referred as the finite element space, being expandable by $\{\phi_i(x, y)\}_{i=1}^N$, where N is the number of basis, then the approximation can be represented in the following expansion form

$$u_h = \sum_{j=1}^{N} u_j(t)\phi_j(x, y), \quad v_h = \sum_{j=1}^{N} v_j(t)\phi_j(x, y). \tag{2.5}$$

A finite dimensional system will be generated when taking $\{\phi_i(x, y)\}_{i=1}^N$ as the trial functions ϕ_h in (2.4). We would like to omit the dependent variables t, as well as x and y, for the clarity if there is no confusion arises. Further denote the vector forms $\mathbf{u} = (u_1, u_2, \cdots, u_N)^T$ and $\mathbf{v} = (v_1, v_2, \cdots, v_N)^T$, then (2.4) yields the Ordinary Differential Equations (ODEs)

$$\begin{cases} M\mathbf{u}' + \epsilon_1 K\mathbf{u} = \mathbf{f} \\ M\mathbf{v}' + \epsilon_2 K\mathbf{v} = \mathbf{g} \end{cases}, \tag{2.6}$$

where $M := \{M_{i,j}\}_{i,j=1}^N$ and $K := \{K_{i,j}\}_{i,j=1}^N$ are referred as the mass matrix and the stiff matrix for the corresponding finite element space $\mathcal{P}(\triangle)$, and the vectors $\mathbf{f} := (f_1, f_2, \cdots, f_N)^T$ and $\mathbf{g} := (g_1, g_2, \cdots, g_N)^T$ are finite element discretization for the nonlinear reaction terms in (4.1) or (4.4), depending on the problem specifications. More precisely

$$M_{i,j} = \int_{\triangle} \phi_i \phi_j dxdy, \qquad K_{i,j} = \int_{\triangle} \nabla\phi_i \cdot \nabla\phi_j dxdy,$$
$$f_i = \int_{\triangle} f(u_h, v_h)\phi_i dxdy, \qquad g_i = \int_{\triangle} g(u_h, v_h)\phi_i dxdy,$$

for all $i, j = 1, \cdots, N$. The ODE system (2.6) is obviously nonlinear because of the nonlinearity of functionals f and g, and the time stepping methods are interested for such kind of ODEs. We would like to emphasis the sparsity of M and K, which are extremely suitable for being treated with various efficient iterative methods.

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

2.3. The semi-implicit time discretization

We are considering the time stepping schemes for ODE system (2.6). It is well known that if any or both of the parameters ϵ_1 and ϵ_2 admit small values, the choice of the time stepping schemes is nontrivial but interesting topic. It is also well accepted that the explicit difference is the simplest scheme, however, it requires very small time step due to the restriction of the stability. From the computational point of view, people refer this as the stiffness of the ODE. To alleviate the difficulty arising from stiffness, the diffusion term is always treated implicitly and the nonlinear reaction term keep explicit for time integration, which result in the well known semi-implicit schemes.

It has been observed that a first order semi-implicit time discretization scheme may lead to plausible but not physical patterns [2, 24]. So that the second order Semi Implicit Backward Formula (2-SBDF) is strongly recommended for practical calculations, which make (2.6) into

$$\begin{cases} (\frac{3}{2}M + \epsilon_1 \delta t K)\mathbf{u}^{n+1} = 2(\delta t \mathbf{f}^n + M\mathbf{u}^n) - (\delta t \mathbf{f}^{n-1} + \frac{1}{2}M\mathbf{u}^{n-1}) \\ (\frac{3}{2}M + \epsilon_2 \delta t K)\mathbf{v}^{n+1} = 2(\delta t \mathbf{g}^n + M\mathbf{v}^n) - (\delta t \mathbf{g}^{n-1} + \frac{1}{2}M\mathbf{v}^{n-1}) \end{cases}, \tag{2.7}$$

where the superscription n and $(n+1)$ means with respectively the values corresponding to nth and $(n+1)$th time step. We then use the 2-SBDF to obtain numerical results for the purpose of comparisons on the a fine static meshes.

When the moving meshes are considered, the 2-SBDF is not convenient because the unknowns on three different levels need to be maintained. More seriously, the interpolations between the meshes may introduce additional errors and destroy its second order convergence rate. Recently, Madzvamuse [15] proposed a first order semi-implicit scheme named 1-SBEM, which use a little information about the nonlinear terms, which are both polynomials of the unknowns. For the ease of description, let us refer them with $f(u, u, v)$ and $g(u, v, v)$ from now on, and so does the discretized vector forms. Despite of the low order in the convergence rate, the stability region of 1-SBEM is declared to be almost 10 times larger than that of 2-SBDF, which shows good balance between the stability and the accuracy. So we prefer 1-SBEM when the moving mesh scheme is considered. It yields

$$\begin{cases} M\mathbf{u}^{n+1} + \epsilon_1 \delta t K \mathbf{u}^{n+1} = M\mathbf{u}^n + \delta t \mathbf{f}(\mathbf{u}^{n+1}, \mathbf{u}^n, \mathbf{v}^n) \\ M\mathbf{v}^{n+1} + \epsilon_2 \delta t K \mathbf{v}^{n+1} = M\mathbf{v}^n + \delta t \mathbf{g}(\mathbf{u}^n, \mathbf{v}^n, \mathbf{v}^{n+1}) \end{cases}, \tag{2.8}$$

where $\mathbf{f}(\mathbf{u}^{n+1}, \mathbf{u}^n, \mathbf{v}^n)$ and $\mathbf{g}(\mathbf{u}^n, \mathbf{v}^n, \mathbf{v}^{n+1})$ are also treated in a semi-implicit manner, for example, the entry for the Gray-Scott's reaction terms (4.1) are

$$\mathbf{f}_i(\mathbf{u}^{n+1}, \mathbf{u}^n, \mathbf{v}^n) = \int_{\triangle} \Big(- \mathbf{u}^{n+1}\mathbf{v}^n\mathbf{v}^n + A(1 - \mathbf{u}^{n+1}) \Big) \phi_i dxdy, \tag{2.9a}$$

$$\mathbf{g}_i(\mathbf{u}^n, \mathbf{v}^n, \mathbf{v}^{n+1}) = \int_{\triangle} \Big(\mathbf{u}^n\mathbf{v}^n\mathbf{v}^{n+1} - \alpha \mathbf{v}^{n+1} \Big) \phi_i dxdy, \ i = 1, \ldots, N, \tag{2.9b}$$

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

and those of Activator-Inhibitor's (4.4) are then turned into

$$\mathbf{f}_i(\mathbf{u}^{n+1},\mathbf{u}^n,\mathbf{v}^n)=\int_{\triangle}\gamma(\mathbf{u}^{n+1}(\mathbf{u}^n-\alpha)(1-\mathbf{u}^n)-\mathbf{v}^n)\phi_i dxdy, \tag{2.10a}$$

$$\mathbf{g}_i(\mathbf{u}^n,\mathbf{v}^n,\mathbf{v}^{n+1})=\int_{\triangle}(\mathbf{u}^n-\lambda\mathbf{v}^{n+1})\phi_i dxdy,\ i=1,\ldots,N. \tag{2.10b}$$

The above scheme avoid the calculation of the Jacobian of the nonlinear reaction terms, so that the costs of discretization are almost equivalent to the costs of 2-SBDF. However, the nonlinear term is treated implicitly in 1-SBEM, which is a compromise between the linear explicit scheme and the high order linearized one, for example, the Newton's methods.

Due to the compact support of the basis of the finite element spaces, the linear systems arising from (2.7) or (2.8) are both sparse. So that a traditional iteration method for sparse linear equations, such as the preconditioned conjugate gradient method, could be applied. We use a multi-grid solver for the linear equations in this research. The multi-grid methods are always served as the fast solver for the sparse linear algebraic equations arising in the finite element applications.

Remark 2.1. A black box algebraic multi-grid method named DMGD9V [23] is used in our computation, which has been extensively tested in various finite element applications. In this solver, the Cartesian structure of the mesh for discretization is assumed, which the operators of limitation and interpolation in multi-grid procedures is based on.

Remark 2.2. The unstructured triangular mesh is also suitable for the proposed moving mesh finite element method if a general linear equation solver is used, for example, the algebraic multi-grid solver supplied by an open source finite element library - the Adaptive Finite Element Package (AFEPack), which is undergoing for three dimensional cases.

3. The adaptively moving mesh scheme

In this section, we explains the details of the moving mesh adaptivity. Under the framework of moving mesh methods, the so called moving mesh equation and the monitor function are two basic elements, which have great freedoms in making a proper choice. As for the finite element simulations for the reaction-diffusion systems, the mainly concerned feature of the numerical methods is the efficiency. The moving mesh adaptivity can surely reduce the computational costs, however, the adaptive algorithm itself should be kept as cheap as possible.

3.1. The moving mesh equation

For the ease of representation, meshes in the finite element methods are always regard as mappings between two different coordinates, in two dimensional case,

$$x=x(\xi,\eta),\quad y=y(\xi,\eta),$$

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

where (ξ, η) and (x, y) are always referred as the computational mesh and the physical mesh respectively. It is remarkable that the two meshes have the same number of nodes and the same topological mesh structure. The computational mesh (ξ, η) is fixed and uniform in all time steps, while the physical mesh or the map (x, y) is nonuniform and exactly what we want to find out during the calculation. Fig. 1 illustrates a intuitive concept for such a moving mesh mapping, which can transfer the basic calculation from a highly non-uniform mesh (the right one) to certain uniform mesh (the left one).

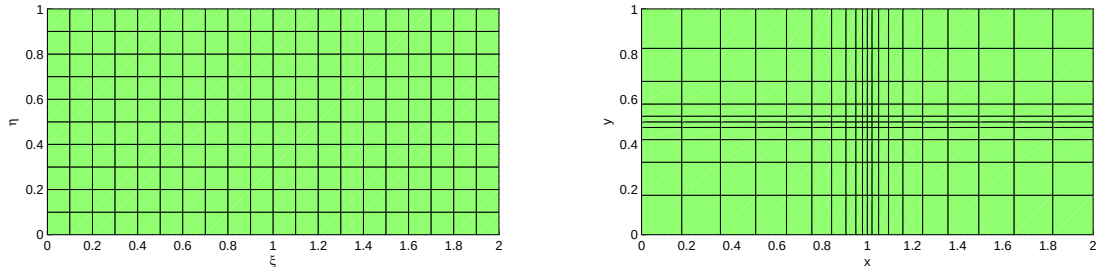


Figure 1: An example of the computational mesh (Left) and the physical mesh (Right).

The concept of mesh mapping make it possible to modify the position of the nodes in the physical mesh (x, y) at each time step for different requirements. In this sense, the nodes of the physical mesh change constantly, so that people refer the strategy as the moving mesh method. As have been stated in the introduction, the fundamental idea of the moving mesh methods is to find an optimal mesh to minimize the mesh-energy functional. The Winslow's equation is the exact Euler-Lagrange equation for the functional. Then the minimization problem is equal to solve

$$\nabla \cdot \left(\frac{1}{\omega} \nabla \xi(x, y) \right) = 0, \quad \nabla \cdot \left(\frac{1}{\omega} \nabla \eta(x, y) \right) = 0,$$

where $\xi(x, y), \eta(x, y)$ is actually the inverse mesh mapping, and $\nabla := (\nabla_x, \nabla_y)^T$ means taking derivatives with respective to coordinate system (x, y). In this model, ω is referred as the monitor function, which is responsible for adjusting the density of the mesh nodes locally.

Additional cost should be paid to find the optimal mesh mapping since the calculation of the inverse map is required in this model, which seems to be too complicated when a problem defined on simple domains, as well as unit square, are considered. A new but similar model is considered to adaptively moving mesh by Mackenzie [14], where ADI is used to solve the algebraic equation and the numerical tests illustrate the efficiency and robustness when application to numerical method for the reaction-diffusion equations. We consider the same elliptic type moving mesh equation in this paper, which is

$$\nabla \cdot (\omega \nabla \mathbf{x}) = 0, \quad \nabla \cdot (\omega \nabla \mathbf{y}) = 0, \tag{3.1}$$

where $\nabla := (\nabla_\xi, \nabla_\eta)$ means taking derivatives with respective to (ξ, η). Such elliptic

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

system are actually the formulation of equi-distribution theory based on the computational meshed. As is pointed out by Mackenzie [14] and Tan [20], the PDE system (3.1) is admissible when general convex domain are considered.

Another important issue is the schemes for boundary mesh movement, which is a interesting research topic as suggested as in [24]. Several standard strategies can be followed from the moving mesh literature, one can either solve a independent 1D moving mesh equation only for the boundary nodes [8,13,14], or minimize a functional with boundary restrictions for harmonic maps [12]. In this paper, we consider a rather simple but effective one, a homogeneous Neumann boundary condition for the both coordinates, which is

$$\frac{\partial x}{\partial n} = 0, \quad \frac{\partial y}{\partial n} = 0. \tag{3.2}$$

If further assume that the corner nodes of the domain keep fixed, such boundary settings keep the domain size fixed although the mesh is changed constantly. The partial differential equation (3.1) together with the boundary condition (3.2) are served as our moving mesh model in this paper.

3.2. A numerical scheme for the moving mesh equation

Since the mesh moving cause the problem of interpolation, a highly accurate solution for the moving mesh model (3.1)+(3.2) is always not necessary, as a matter of fact, people are always satisfied with a roughly adapted mesh mapping. So that a rough, however, cheap solver for the moving mesh equation is interested.

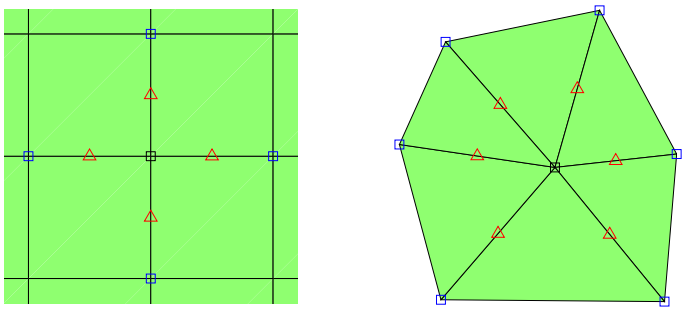


Figure 2: The concept of discrete Laplacian on Cartesian mesh (Left) as well as for triangle mesh (Right). Black square: updating node; Blue square: nodes neighboured; Red △:position to evaluate ω.

It is observed that a iterative solver, as simple as the Gauss-Sediel iteration, for moving mesh model is effective and sufficient if smoothed monitor function is presented. In the case of Cartesian mesh, as illustrated as in the left plot of Fig. 2, a standard finite difference scheme is efficient for computation, which yields

$$X_{i,j} = \frac{(\omega_{i-\frac{1}{2},j}X_{i-1,j} + \omega_{i+\frac{1}{2},j}X_{i+1,j} + \omega_{i,j-\frac{1}{2}}X_{i,j-1} + \omega_{i,j+\frac{1}{2}}X_{i,j+1})}{(\omega_{i,j-\frac{1}{2}} + \omega_{i,j+\frac{1}{2}} + \omega_{i,j-\frac{1}{2}} + \omega_{i,j+\frac{1}{2}})}, \tag{3.3}$$

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

where the $\frac{1}{2}$ in the subscription means to evaluate the monitor function at the half-point of the corresponding mesh edge. Practically it is evaluated by taking the average between two neighboring nodes, such as $\omega_{i,j-\frac{1}{2}} = \frac{1}{2}(\omega_{i,j-1} + \omega_{i,j})$.

Remark 3.1. The extension for the iteration scheme (3.3) to the case of unstructured triangle mesh is trivial. From the right plot in Fig. 2, we can find that it is easy to update position of the center nodes in a similar manner as well as those on Cartesian meshes, roughly,

$$X_i = \sum_j (\omega_i^j X_i^j) / \sum_j \omega_i^j,$$

where X_i^j is the j'th neighbor of node X_i and ω_i^j is the value of the monitor function defined on edge $X_i X_j$. Considerable time saving could be archived since only 3-5 iterations of (3.3) is enough to obtain satisfied adaptive meshes in practical computations, while the whole linear system should be solved in any other moving mesh methods.

3.3. Choice of the monitor function

The monitor function ω is responsible for adjusting the densities of the mesh locally, in a way that the mesh nodes are dense where the value of ω are large. In the framework of the moving mesh method, the choice of the monitor function is the key issue for a robust moving mesh algorithm. The most common choice of ω is related with the gradient of the solution

$$\omega(u) = \sqrt{\alpha + \beta |\nabla u|^2 + \gamma u^2}, \tag{3.4}$$

where α, β and γ are all parameters for different configurations. Practically, α is used to prevent the monitor function from too small value, which may destroy the smoothness of the generated mesh, and a larger value of β yields more mesh nodes near the region where the numerical solution varying rapidly, while γ is useful when you are intend to capture the position of singularity accurately for a potentially singular problems [4]. Proper values of these parameters are depend on the problems to generate satisfied nonuniform meshes, and only u part of the solution is used since u and v have almost the same rapid varying areas. A smoothed mesh is always preferred because it perform well in reducing the discretization errors for various numerical methods. So that the monitor function (3.4) is smoothed 2 or 3 times via

$$\begin{aligned} \omega_{i,j} = {} & \frac{1}{16}\Big(\omega_{i+1,j+1} + \omega_{i+1,j-1} + \omega_{i-1,j+1} + \omega_{i-1,j-1}\Big) \\ & + \frac{1}{8}\Big(\omega_{i,j-1} + \omega_{i,j+1} + \omega_{i-1,j} + \omega_{i+1,j}\Big) + \frac{1}{4}\omega_{i,j}, \end{aligned} \tag{3.5}$$

after it is calculated using (3.4). We remark here that it is a special form for smooth procedure defined the Cartesian mesh, and the case for the triangle mesh is trivial to obtain following the concept as illustrated as in Fig. 2. It is also known from the PDEs theory that a smooth monitor function ω leads to a smooth mesh mapping. More importantly, the smooth step accelerates the convergence of iterative solver for the moving mesh equation.

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

3.4. Interpolating between the meshes

Considering mesh moving during the finite element computation, the interpolation scheme from the old mesh to the new one is necessary for numerical solving time dependent partial differential equations. Then the numerical solutions on the current time step can be fed to the next time step as the initial values after executing the interpolation. For the ease of practical using, it is reasonable to decouple the moving mesh step totally from the evolution of physical models, so that we can handle the mesh movement in any prescribed way without affecting the finite element solver for the physical PDEs. Following the basic idea of continuation method as proposed as in [12], a simplified version is implemented in the current research, which is actually the first order approximation based on the grid system of (x, y):

$$u^* = u + \nabla_x u(x^{n+1} - x^n) + \nabla_y u(y^{n+1} - y^n), \tag{3.6}$$

where u,$\nabla_x u$ and $\nabla_y u$ denote the value and two partial derivatives of u with respect to coordinate x and y of the old mesh (x^n, y^n), and u^* is the updated value onto the new mesh (x^{n+1}, y^{n+1}). One can update point-by-point or following a finite element way as proposed in [12], with additional cost for solving a simple algebraic linear system. We remark here that when the time step is small enough, the meshes change continuously so that the update scheme (3.6) is accurate enough.

As an enclosure for the current section, let us describe the whole moving mesh finite element calculation as in Algorithm 3.1.

Algorithm 3.1. *Moving mesh computation for reaction-diffusion system.*

***Input**: mesh, parameters*
***Output**: u,v every 1000 time steps*
initialize (mesh, parameters)
calculate discrete operator M, K in (2.6)*;*
np = 1;
While** $t \leq TEND$ **do
  if** $mod\ (np,5) == 0$ **then
    calculate and smooth ω via (3.4) *and* (3.5) *respectively;*
    solve (3.1)+(3.2) *using Gauss-Sediel relaxation* (3.3)*;*
    update (u,v) via (3.6)*;*
    re-calculate (M, K) in (2.6)*;*
  end
  calculate right hand side of physical system with (2.9) *or* (2.10)*;*
  solve physical problem using approach (2.7) *or* (2.8)*;*
  if** $mod\ (np, 1000)$ **then
    output (u,v);
  end
end

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

4. Numerical experiments

In this section, we are intended to test the moving mesh algorithm in the framework of finite element method. Two different types of reaction diffusion systems are considered, which are the Gray-Scott model and the Activator-Inhibitor model. These two models have different kind of nonlinear reaction terms and are widely interested in pattern formation theory . The comparisons between the moving mesh results with those from 2-SBDF on a finer fixed mesh are made, which illustrate the convergence of the moving mesh computation. All the numerical simulations in this paper are all carried on a PC equipped with 4Gb memory and 3.0GHz intel core2 Duo CPU.

4.1. the Gray-Scott model

The first numerical experiment is about the Gray-Scott model [10, 24]. Let u and v be the concentration of two different chemical reactions, the Gray-Scott model has a consumer resource type nonlinear reaction term

$$\begin{aligned} f(u,v) &= -uv^2 + A(1-u), \\ g(u,v) &= uv^2 - \alpha v, \end{aligned} \tag{4.1}$$

where A and α are constant parameters for different configurations. It describes a variety of amazing patterns formations by autocatalytic chemical reactions. Their patterns exhibits in a great variation with respect to different parameters (refer to [19] for more details on the background for the model). In this simulation, the following choice for the model parameters

$$\epsilon_1 = 8 \times 10^{-5}, \quad \epsilon_2 = 4 \times 10^{-5}, \quad A = 0.024, \quad \alpha = 0.084,$$

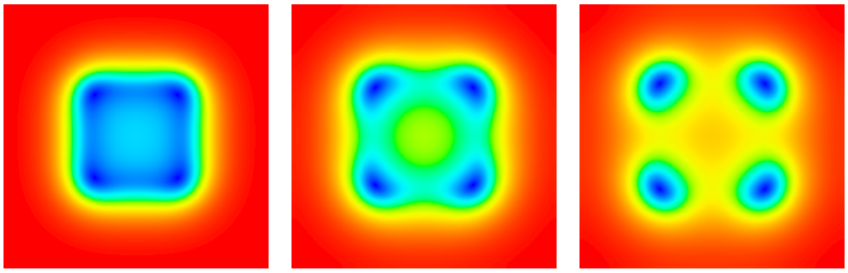
yield a classical pattern as considered in [24], where $A > 0$ means a supply of the resource u from the environment. The computational domain is selected as $[-0.5, 0.5] \times [-0.5, 0.5]$ for the symmetric property. The homogeneous Neumann boundary conditions (2.2) and the following initial condition are considered:

$$u(x,y,0) = \begin{cases} 0.5, & (x,y) \in [-0.3, 0.3] \times [-0.3, 0.3] \\ 1, & \text{otherwise,} \end{cases}, \tag{4.2}$$

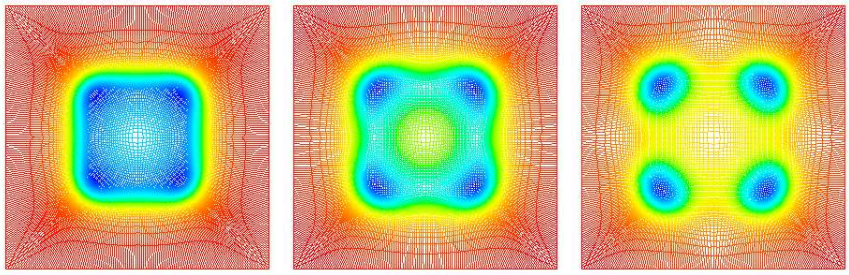
$$v(x,y,0) = \begin{cases} 0.25, & (x,y) \in [-0.3, 0.3] \times [-0.3, 0.3] \\ 0, & \text{otherwise.} \end{cases}, \tag{4.3}$$

We plot the pattern of u using 2-SBDF with a 1024×1024 fixed mesh and the adaptively generated 128×128 mesh using 1-SBEM in Fig. 3. Results for three different time $t = 50$, $t = 100$ and $t = 150$ are listed correspondingly from left to right. We are satisfied to find that the moving mesh catch the variation of u correctly and the smallest size of the element in the moving mesh is almost as the same that of a 1024×1024 fixed mesh, which illustrate the convergence of our moving mesh finite element scheme. The

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press



(a) Image of u from 2-SBDF on a 1024×1024 fixed uniform mesh.



(b) The mesh generated with 1-SBEM on a 128×128 moving mesh.

Figure 3: The Gray-Scott with $\epsilon_1 = 8 \times 10^{-5}$,$\epsilon_2 = 4 \times 10^{-5}$, $A = 0.024$ and $\alpha = 0.084$. We show the results at three different time level(from left to right): $t = 50$,$t = 100$ and $t = 150$. In both cases, the time step is 0.0025, and the moving meshes are updated every 10 steps.

moving mesh is colored according to the value of u. Since the value of the unknowns are uniformly bounded in all our numerical examples and the interfaces of different species need to be identified, we drop off the last term in the monitor function (3.4) and the other parameters are chosen as described as in the form of

$$\omega(u) = \sqrt{\alpha + \beta |\nabla u|^2},$$

where α and β are two parameters specified by users. In a traditional applications, $\alpha = 1$ always works and β is used to adjust the importance of the gradients. In this paper, $\alpha = 1$ and $\beta = 100$ works well.

Comparisons in quantities are also made. We record the CPU time costs both for the moving mesh step and the problem solving step, and the memory consuming are also tabulated in Table 1. It is also worth to mention that more CPU time could be

Table 1: The CPU time costs for the moving mesh step (CPU time I) and problem solving step (CPU time II), and memory consuming for both 2-SBDF and 1-SBEM for example 4.1.

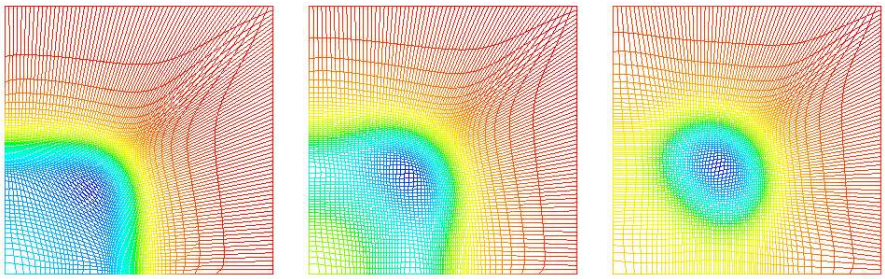
Scheme	Mesh	h_{min}	CPU time I(s)	CPU time II(s)	Memory(Mb)
2-SBDF	1024×1024	0.001	0	34,650	412.5
1-SBEM	128×128	0.0013	121	1,202	10.5

saved according to adjust the frequency of moving mesh. In this simulation, the mesh is adapted once every 10 time steps.

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

A fully adaptive strategy for adjusting the moving mesh frequency and more general form of the monitor functions, as well as those in [12, 14, 22], are favorable. The relationship between the time step length δt and the character grid size h, say, $\delta t \approx h^2$, could be used to determine δt in each time step, which may save the computational work furthermore. In this paper,a fixed time step length and a fixed adaptive moving mesh frequency are used. The reasons are in two folds. The first one is for the convenience of the comparison between 2-SBDF and 1-SBEM; and the second one is based on the numerical observations: it is observed that rough adjustments of the moving mesh frequency, such as 2,5 or 10 do not significantly affect the ultimate solutions. Simply speaking, a fixed time step is convenient and sufficient in these simulations, see Table. 1 for an illustration.

Noticing that u and v are symmetric both with respect to the x-axis and the y-axis, so the computation is actually performed within only one quarter, that is $[0, 0.5] \times [0, 0.5]$. In this case, we should consider moving the nodes along the line $x = 0$ and $y = 0$ because the pattern already grown to these boundaries. The homogeneous Neumann conditions (3.2) on these boundary and fixed condition on the right and top boundary are suitable for the moving mesh equations (3.1). One can find from the more closer look at the moving mesh in Fig. 4, that the meshes are dense where the values of u change rapidly both inside and on the boundaries.



(a) $t = 50$ (b) $t = 100$ (c) $t = 150$

Figure 4: A closer look at the moving mesh in Fig. 3

4.2. A complex pattern simulation

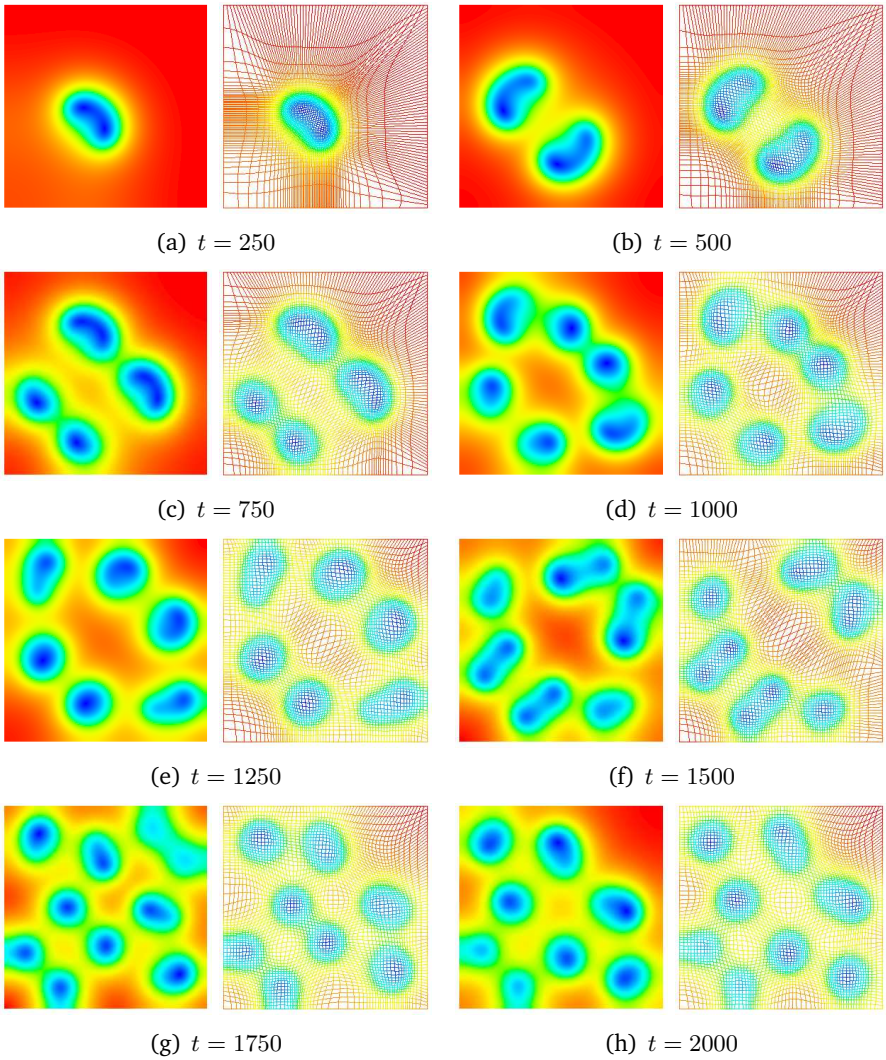
Let us consider a different parameters configuration as suggested as in [10], as well as [18], where it is figured out that the patterns could potentially be very complex in the case of $\epsilon_1/\epsilon_2 = 2$.

$$\epsilon_1 = 2 \times 10^{-5},\ \epsilon_2 = 1 \times 10^{-5},\ A = 0.02,\ \alpha = 0.077.$$

Other computational settings are same with the previous example. The time step size is also $dt = 0.01$ here and the simulation is not stopped until the time reaches to $t = 2000$. A self-replication dynamic procedure is presented and it leads to a relative complex pattern that fill almost all the domain as shown as in Fig. 5. Also the results

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

of 2-SBDF with a 512×512 fixed mesh and those of 1-SBEM with a 64×64 adaptive moving mesh are compared for different time levels.



(a) $t = 250$ (b) $t = 500$

(c) $t = 750$ (d) $t = 1000$

(e) $t = 1250$ (f) $t = 1500$

(g) $t = 1750$ (h) $t = 2000$

Figure 5: A complex self-replication procedure for Gray-Scott with $\epsilon_1 = 2 \times 10^{-5}, \epsilon_2 = 1 \times 10^{-5}, A = 0.02, \alpha = 0.077$, time step is $dt = 0.01$. In each subfigure, the left is the image using fixed uniform mesh, the right one is obtained with adaptively moving mesh, and the frequency for adapt mesh is 5.

Fig. 5 illustrates that the adaptive moving mesh can also catch up with the variation of u. The pattern almost fill all the domain after $t = 1000$, and they are just moving around continuously. In such a case, the moving mesh adaptivity is a better choice than an adaptive mesh refine scheme.

We also record the CPU time costs and the memory consuming in this simulation. They are tabulated in Table 2. The mesh adapt frequency is 5 in this example. It is also obvious from these data that the moving mesh shcme results into similar smallest mesh

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

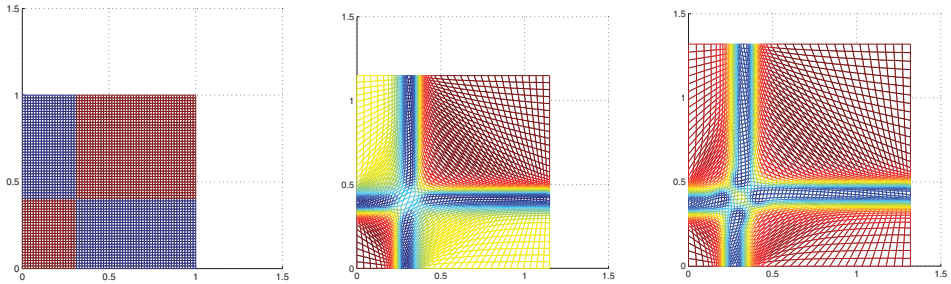
Table 2: The CPU time costs for the moving mesh step (CPU time I) and problem solving step (CPU time II), and memory consuming for both 2-SBDF and 1-SBEM for example 4.2.

Scheme	Mesh	h_{min}	CPU time I(s)	CPU time II(s)	Memory(Mb)
2-SBDF	512×512	0.002	0	19,680	112.9
1-SBEM	64×64	0.003	104	629	3.5

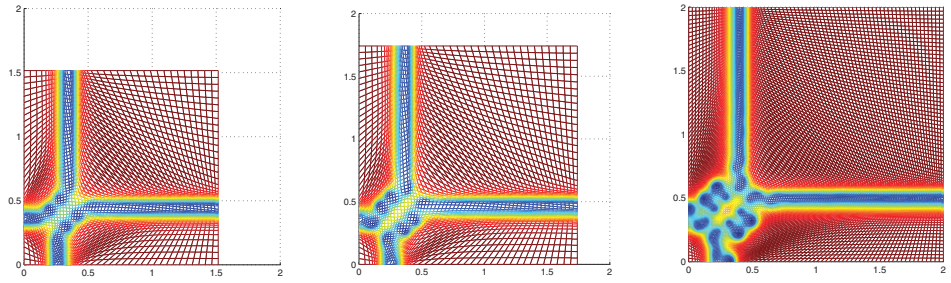
size, however, costs much more less than the SBDF2 on a fixed mesh.

4.3. An example with growing domain

Let us consider a more challenge case. The model problem is still the Gray-Scott (4.1), and the computational domain is a unit square, which will grow to twice of the original size until t = 500 seconds. An exponential growing rate $r(t) = e^{\rho t}$ is imposed, where r means the right end of x-axis or the top end of y-axis. In this setting, it is straightforward that $\rho = (\log 2)/500$. A similar reaction-diffusion system with growing domain has been considered in [15], where the domain growth is modeled by an additional "convection term". In the current moving mesh methods, the moving mesh step is responsible for the domain growth, no additional work is required to modify the reaction-diffusion system solver.



(a) $t = 0$ (b) $t = 100s$ (c) $t = 200s$



(d) $t = 300s$ (e) $t = 400s$ (f) $t = 500s$

Figure 6: The solutions for the Gray-Scott with growing domain are plotted in (a)-(f) for different time. The axis region in the first row are $[0, 1.5] \times [0, 1.5]$, and $[0, 2] \times [0, 2]$ for the second row. The mesh resolution for the plot of $t = 500$s is 128×128, and 64×64 for the other five plots.

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

All other physical parameters follow those in the previous example, and homogeneous Neumann boundary conditions are applied. The initial condition is chosen as plot (a) of Fig. 6, which will finally generate a complex pattern as shown in plot (f). It has been indicated in [15], that the time step of SBDF2 could be in the order of 10^{-6} due to the stability restriction. According to CPU time listed in Table 1 and Table 2, it seems impossible for doing a SBDF2 simulation within days. A moving mesh scheme based on SBEM1 seems to be a potential scheme.

As well as the previous example, the mesh is adaptively moved according to the monitor function $\omega(u) = \sqrt{1.0 + 10|\nabla u|^2}$. We take a 128×128 Cartesian mesh, and the minimum size of the mesh in (f) is 0.00131, which is better than a 1024×1024 uniform mesh. We take $\delta t = 0.0025$ in the moving mesh calculation, then totally 200,000 time steps are required for the domain to reach twice its original size. The results are saved every 40,000 time steps, and we show in Fig. 6 the meshes at six different time. The mesh resolution for plot (a)-(e) are cuted to half for the clarity. In summation, it totally costs about 3 minutes for adaptively moving mesh, and 31 minutes for solving the reaction-diffusion equation.

4.4. The Activator-Inhibitor nonlinearity

In this example, we are interested in a reaction-diffusion system with Activator-Inhibitor nonlinearity [10], which is

$$\begin{aligned} f(u,v) &= \gamma(u(u-\alpha)(1-u)-v), \\ g(u,v) &= u - \lambda v, \end{aligned} \tag{4.4}$$

where γ, α and λ are all constant parameters for different pattern configurations. The computational domain is $[0,1] \times [0,1]$, and the physical parameters are

$$\epsilon_1 = 0.001,\ \epsilon_2 = 1,\ \gamma = 1000,\ \alpha = 0.25,\ \lambda = 10.0.$$

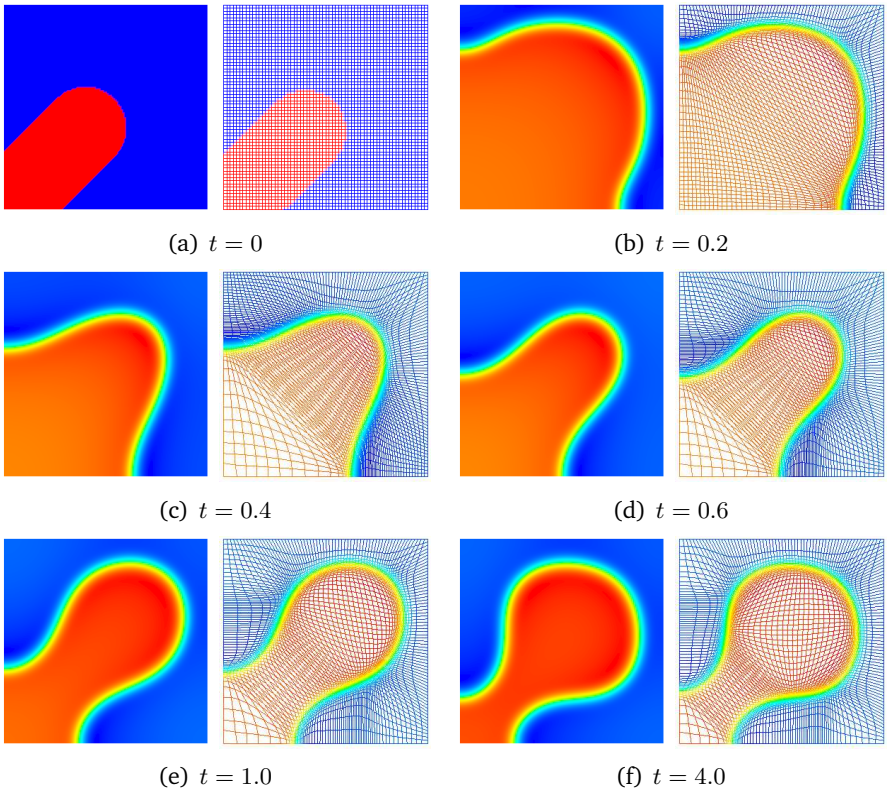
The initial condition $u_0(x,y)$ is shown in the first figure of Fig. 7, where the red region means $u_0(x,y) = 1.0$ and the blue one means $u_0(x,y) = 0.25$, and $v_0(x,y) = v_0 = 0.0$. As we find during the numerical simulation, the shape of $u_0(x,y)$ is not so important, even the choice like (4.2) is convergence to a similar result. The boundary condition for the physical model is still homogeneous Neumann condition.

Large ratio between the parameters are presented in such configuration, so in practical computation, we follow the original form for Activator-Inhibitor model as presented as in [10], where a factor of 0.001 is multiplied at both sides for the equation of u. In this case, the time step is selected to be $dt = 0.001$ and the simulation is end when $t = 4.0$. To generate a relative smoothed mesh, the monitor function is chosen as

$$\omega(u) = \sqrt{5 + 10|\nabla u|^2},$$

where we found through the numerical experiments that the numerical parameters are not very sensitive to the moving mesh results, if their values are chosen comparable to

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press



(a) $t = 0$ (b) $t = 0.2$

(c) $t = 0.4$ (d) $t = 0.6$

(e) $t = 1.0$ (f) $t = 4.0$

Figure 7: Snap shots for u with 2-SBDF on a 1024×1024 fixed mesh and the 1-SBEM computation on a 64×64 moving mesh.

those in the above form. This phenomena illustrates the stability of the moving mesh algorithm in some sense.

As a matter of fact, the pattern of u changes rapidly before $t = 1.0$ and come to be stable in the following time. We found that is not sensitive with respect to different choices of initial condition for u, so we illustrate only at some featured time in Fig. 7, where the pattern of u from 2-SBDF computation on a fixed 1024×1024 mesh versus the a 64×64 adaptively moving mesh with 1-SBEM, are presented at six different time levels. We are pleased to find that the moving mesh catches the large internal layer of u exactly, and also work with the 1-SBEM time stepping in Activator-Inhibitor type nonlinearity.

5. Conclusion

We present an efficient moving mesh finite element method for numerical simulations of general reaction-diffusion systems. The numerical experiments show that considerable savings both in time and storage consuming are archived. The movement of the mesh boundary is implemented by solving a moving mesh equations efficiently.

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

In all the simulations, the moving mesh step is decoupled from the solution of physical models, so that various combinations between the moving mesh step and the solve step can be tried for different type models. More importantly, the treatment of the cases with growing domain is straightforward with the moving mesh technique.

There are some phenomenon which may leads to more robust moving mesh applications. One significant problem is that some generated mesh elements are too "thin". It may caused by the regular assumption of the mesh in the current research. Actually, a combination of h-adaptive is under considering to improve the quality of the adaptive mesh. A potential method to avoid the problem is to do edge-swapping where a thin-element pair appears. Another problem is to push the two dimensional computation to the limit of the computers, such as extreme complex patterns, it should be considered in another research work.

Acknowledgments This work was supported by the Fundamental Research Funds for the Central Universities 2016XZZX005-01, the National Natural Science Foundation of China with grant NO. 10926087, and A Project Supported by Scientific Research Fund of Zhejiang Provincial Education Department with grant NO. Y201121891.

References

[1] R.A. ADAMS AND J.J.F. FOURNIER, *Sobolev Spaces*, Number 140 in Pure and Applied Mathematics, Elsevier Science, 2003.

[2] U. M. ASCHER, S. J. RUUTH, AND B. T. R. WETTON, *Implicit-explicit methods for time-dependent pdes*, SIAM J. Numer. Anal., 32 (1997), pp.797–823.

[3] S. C. BRENNER AND L. R. SCOTT, *The Mathematical Theory of Finite Element Methods*, Springer, 2nd edition, April 2002.

[4] H. D. CENICEROS AND T. Y. HOU, *An efficient dynamically adaptive mesh for potentially singular solutions*, J. Comput. Phys., 172 (2001), pp. 609–639.

[5] P. G. CIARLET, *The finite element method for elliptic problems*, SIAM: Society for Industrial and Applied Mathematics, 2nd edition, 2002.

[6] M. R. GARVIE, *Finite-difference schemes for reaction-diffusion equations modeling predator-prey interactions in matlab*, Bulletin of Mathematical Biology, 69 (2007), pp. 931–956.

[7] G. HU AND P. A. ZEGELING, *Simulating finger phenomena in porous media with a moving finite element method*, J. Comput. Phys., 230 (2011), pp. 3249–3263.

[8] W. HUANG, *Moving mesh strategy based on a gradient flow equation for two-dimensional problems*, SIAM J. Sci. Comp., 3 (1999), pp. 998–1015.

[9] A. KASSAM AND L. N. TREFETHEN, *Solving reaction-diffusion equations 10 times faster*, Oxford University, Numerical Analysis Group Research Report No.16, 2003.

[10] M. KIMURA, H. KOMURA, M. MIMURA, H. MIROSHI, T. TAKAISHI, AND D. UEYAMA, *Adaptive mesh finite element method for pattern dynamics in reaction-diffusion systems*, Proceedings of the Czech-Japanese Seminar in Applied Mathematics, 1 (2005), pp. 56–68.

[11] S. S. LEE AND E. A. GAFFNEY, *Aberrant behaviors of reaction diffusion self-organisation models on growing domains in the presence of gene expression time delays*, Bulletin of Mathematical Biology, 72 (2010), pp. 2161–2179.

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press

[12] R. Li, T. Tang, and P. Zhang, *Moving mesh methods in multiple dimensions based on harmonic maps*, J. Comput. Phys., 170 (2001), pp. 562–588.
[13] K. Liang, P. Lin, M. T. ONG, and R. C. E. Tan, *A splitting moving mesh method for reaction-diffusion equations of quenching type*, J. Comput. Phys., 215 (2006), pp. 757–777.
[14] J. A. Mackenzie, *The efficient generation of simple two-dimensional adaptive grids*, SIAM J. Sci. Comput., 19 (1998), pp. 1340–1365.
[15] A. Madzvamuse, *Time-stepping schemes for moving grid finite elements applied to reaction-diffusion systems on fixed and growing domains*, J. Comput. Phys., 214 (2006), pp. 239–263.
[16] M. Mimura, *Pattern formation in consumer-finite resource reaction-diffusion systems*, Publ. RIMS, Kyoto Univ., 40 (2004), pp. 1413–1431.
[17] J. D. Murray, *Mathematical Biology 2nd. ed.*, Springer-Verlag, 1993.
[18] K. M. Page, P. K. Maini, and N. A. M. Monk, *Complex pattern formation in reaction-diffusion systems with spatially varying parameters*, Physica D, 202 (2005), pp. 95–115.
[19] J. A. Pearson, *Complex patterns in a simple system*, Science, 261 (1993), pp. 189–192.
[20] Z. Tan, *Adaptive moving mesh methods for two-dimensional resistive magnetohydrodynamic PDE models*, Comput. & Fluids, 36 (2006), pp. 758–771.
[21] V. Thomée, *Galerkin Finite Element Methods for Parabolic Problems (Springer Series in Computational Mathematics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[22] H.-Y. Wang, R. Li, and T. Tang, *Efficient computation of dendritic growth with r-adaptive finite element methods*, J. Comput. Phys., 227 (2008), pp. 5984–6000.
[23] P.M. De Zeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multi-grid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.
[24] P. A. Zegeling and H. P. Kok, *Adaptive moving mesh computations for reaction-diffusion systems*, J. Comput. Appl. Math., 168 (2004), pp. 519–528.
[25] Z. Zhang and T. Tang, *An adaptive mesh redistribution algorithm for convection-dominated problems*, Comm. Pure Appl. Anal., 1 (2002), pp. 341–357.

https://doi.org/10.4208/nmtma.2016.m1229 Published online by Cambridge University Press