Analysis and Computation of Adaptive Moving Grids by Deformation

Pavel Bochev, Guojun Liao, Gary dela Pena

Department of Mathematics
University of Texas at Arlington
Box 19408, Arlington, TX 76019-0408

Abstract

We develop and analyze a numerical method for creating an adaptive moving grid in one, two and three-dimensional regions. The method distributes grid nodes according to a given analytic or discrete weight function of the spatial and time variables, which reflects the fine structure of the solution. The weight function defines a vector field which is used to construct a transformation of the computational domain into the physical domain. We prove that the resulting grid has the prescribed cell sizes and that no "mesh tangling" occurs. Numerical implementation of the method utilizes efficient and robust least-squares solver to compute the vector field and a fourth order Runge-Kutta scheme to determine the transformation. Results of several numerical experiments in one and two-dimensions are also presented. These results indicate, among other things, that the method accurately redistributes the nodes and does not tangle the mesh.

1 Introduction

In many practical applications one has to solve numerically differential equations whose solutions exhibit boundary layers, singularities or shocks. An obvious disadvantage in using fixed grids for such problems is that the grid points are distributed in the physical domain before any details on the nature of the solution are known. As a result, the generated numerical data fails to reflect the true nature of the solution. One approach which leads to improved accuracy and efficiency is to consider adaptive methods. A characteristic trait of such methods is the use of grids which are generated according to the salient features of the solution. In general one may distinguish two common situations in which need for adaptivity arises. In the first case, which is typical for steady state problems, location of the regions where more resolution is needed is fixed and adaptivity can be efficiently implemented using local refinement techniques; see, e.g., [4], [11], and the comprehensive collection [3]. The second case is typical for time dependent problems. Solutions of such problems may not only involve large variations due to shock waves and contact surfaces, but the location of these variations may also change in time. The additional resolution needed in corresponding regions can again be provided by means of local refinement. However, since these regions change with time, grids generated at a particular time step may not be suitable for the subsequent time steps, and have to be replaced by new grids. As a result, the use of local refinement may become computationally demanding, in particular for complex geometries in three dimensions where the cost for grid generation becomes a significant factor in the overall performance of the method.

In this paper we pursue an alternative, moving grid approach in which instead of adding new nodes, the grid points are moved where and when the need arises. Methods based on moving grids have several potentially valuable computational properties, in particular for complex geometries in three-dimensions. Such methods do not require complicated data structures since the reference domain is not changed. Moreover, cost of moving the grid is limited to computation of the new physical coordinates of the grid points. As a result, overall efficiency of a moving grid adaptive method can be significantly improved.

Among the moving grid methods that have been introduced are the Moving Finite Element method (MFE) of Miller [20]-[21]. This method controls the movement of the grid by a process based on the equidistribution of the residual of the original equations written in finite element form. Recently Huang et. al. [14] developed an approach based on continuous moving meshes equations which they call Moving Mesh Partial Differential Equations (MMPDE), This approach is based on the equidistribution of some measure of error using finite difference schemes. More adaptive moving grid methods can be found in Hawken et. al. [13], and in the recent monograph by Zegeling [26].

In general, moving grid methods are well received for one-dimensional problems; see, e.g., [14] and [26]. At the same time, moving grid methods in two and three-dimensions may introduce "mesh tangling" and have been subject to a just criticism. Thus, the main goal of this paper is to devise efficient grid moving algorithms for one, two and three space dimensions which can be shown to be "mesh tangling" free in a rigorous mathematical way. To accomplish this we introduce a new approach based on the principle of equidistribution of a continuous moving grid method. Given a weight function $f(\mathbf{x},t), \mathbf{x} \in \mathbf{R}^n, t \in \mathbf{R}$, the method simultaneously redistributes the grid points so that the cell size is equal to value of the weight function for all $\mathbf{x}$ and t.

Our method generates the moving grid in two stages. On the first stage a given analytic or discrete weight function is used to construct a vector field with vanishing normal component on the boundary which satisfies a div-curl system. On the second stage, this vector field is used to generate a transformation which updates the grid. Our numerical implementation of the grid moving method incorporates several novel approaches. In particular, we use a least-squares finite element method to construct the vector field. This offers significant advantages in the algorithmic design of the moving grid method. For example, stable approximation of the vector field does not demand finite element spaces which satisfy inf-sup condition. Computation of this field requires solution of linear system of algebraic equations with symmetric and positive definite matrix which can be solved by efficient iterative methods, such as conjugate gradients. As a result, assembly of the discretization matrix can be avoided, even at the element level. Second, the essential boundary condition can be enforced in a weak, variational sense, simplifying further the required finite element spaces. As a result, the least-squares solver for the div-curl system is highly efficient in two and three-dimensions.

The paper is organized as follows. In Section 2 we introduce the equidistribution principle upon which the moving grid method is based. In the same section we present theoretical analysis of our algorithm. Then, in Section 3, we consider numerical implementation of this algorithm. Finally, in Section 4 we present results of numerical experiments with the grid moving method involving grids on one and two-dimensional regions and various analytic weight functions.

Let us now establish the notation used throughout this paper. We shall use bold face symbols to denote vectors. Ω will denote bounded region in $\mathbf{R}^n$, $n = 2, 3$ with Lipschitz continuous boundary

Γ. With C we shall denote a generic, positive constant which may depend on the particular domain Ω, but is independent from any grid parameters. As usual, $L^2(\Omega)$ will denote the Hilbert space of all square integrable functions on Ω with inner product $(\cdot,\cdot)$ and norm $\|\cdot\|_0$. The Sobolev space $H^1(\Omega)$ is defined as the space of all $L^2(\Omega)$ functions whose derivatives are also in $L^2(\Omega)$, and $\mathbf{H}^1_n(\Omega)$ will denote the space of all vector functions in $\mathbf{R}^n$, $n = 2, 3$ with vanishing normal component on the boundary Γ, i.e.,

$$\mathbf{H}^1_n(\Omega) = \{\mathbf{u} \in [H^1(\Omega)]^N, | \, \mathbf{u} \cdot \mathbf{n} = 0 \quad \text{on } \Gamma\} \tag{1}$$

We also agree to use $\|\cdot\|_1$ for the norm on both $H^1(\Omega)$ and $\mathbf{H}^1_n(\Omega)$. Finally, we recall that in two-dimensions there are two curl operators. If $\mathbf{v} \in \mathbf{R}^2$, then the operator curl maps $\mathbf{v}$ into the scalar function $\operatorname{curl} \mathbf{v}$. If $\mathbf{v}$ is three-dimensional vector field, then $\mathbf{curl}\, \mathbf{v}$ is again vector field in $\mathbf{R}^3$. However, to avoid multiplicity of notation we agree to use **curl** for both curl operators.

2 Solution-adaptive moving grid method

The goal of this section is to formulate the grid moving method and to present its theoretical analysis. Our main result is that constructed grids satisfy the equdistribution principle without mesh tangling.

The equidistribution principle involves selecting spatial grid points in such a manner that some measure of the solution error is equalized over each subinterval. Thus, in regions where solution error is large, equidistribution allows greater concentration of grid points. Similarly, equidistribution allows for a lower concentration of grid points in regions where solutions are smooth. In one-dimension, this principle can be formulated in the following manner:

> Consider the transformation $x(\xi)$, from a uniform grid in ξ-space, where the coordinate ξ identify the gridpoints. The equidistribution principle then is (see, e.g., [1], [24])
>
> $$x_\xi \; f = \text{constant},$$
>
> where x_ξ represents the variation in x between gridpoints and f is the weight function. In such a case we say that the weight function f is equidistributed over the grid.

In the general case let $f(\mathbf{x}, t) : \Omega \mapsto \mathbf{R}$ be a given weight function. Such function must satisfy several conditions. First, $f(\mathbf{x}, t)$ must be strictly positive, and it must have nonzero lower bound. Second, $f(\mathbf{x}, t)$ must be differentiable, and lastly, the weight function must have the following normalization property

$$\int_\Omega \Big(\frac{1}{f} - 1\Big)\, d\mathbf{x} = 0.$$

With the given weight function we associate a one-to-one mapping $\boldsymbol{\varphi}(\mathbf{x}, t)$ from the computational domain Ω into the physical domain Ω. In order for the gridpoints to be equidistributed we now require that

$$\det \nabla \boldsymbol{\varphi}(\mathbf{x}, t) = f(\boldsymbol{\varphi}(\mathbf{x}, t), t), \quad \text{all } \mathbf{x} \in \Omega, \text{ and } t > t_0. \tag{2}$$

Indeed, recall that the volume element in the physical variable is given by $\det \nabla \boldsymbol{\varphi}(\mathbf{x}, t) d\mathbf{x}$, i.e., it is proportional to $\det \nabla \boldsymbol{\varphi}(\mathbf{x}, t)$. As a result, (2) guarantees that volume elements in the physical

grid will satisfy the equidistribution principle. In what follows we shall focus our attention on the construction of the mapping $\boldsymbol{\varphi}$ and its analytical properties. For this purpose f can be any function with the properties described above. In realistic situations, however, the weight function is subject to some additional constraints. For example, it should reflect the need for grid refinement of the underlying problem, i.e., one has to choose the mechanism under which the changes in the exact (or approximate) solution will be accounted for by the weight function. One possible relation between solutions $\mathbf{u}$ and weight functions can be described as

$$f(\mathbf{x},t) = \frac{C}{1+\alpha_1|\nabla\mathbf{u}|^2+\alpha_2|\mathbf{u}|^2}\,,$$

where C is a normalizing factor which may depend on t; see, e.g., [26]. Another possibility discussed in [14] is to consider relations of the form

$$f(\mathbf{x},t) = \frac{C}{\sqrt{1+\alpha|\nabla\mathbf{u}|^2}}\,,$$

or, the, even simpler, weight function, see [14],

$$f(\mathbf{x},t) = \frac{C}{1+|\mathbf{u}|}\,.$$

In addition, in realistic problems one has to consider discrete weight functions which are generated by the solvers error estimator. For this purpose one can use, for example, local posteriori estimates of errors based on local residuals; see, e.g., [3] and [4]. To treat discrete weight functions within the framework of our grid moving method, we assume that such functions are replaced by suitably defined interpolants, which we denote again by f.

Let us now describe the grid moving method. In the center of our approach is construction, for a given weight function f, of a mapping $\boldsymbol{\varphi}$ which satisfies the equidistribution principle (2). For this purpose we first define a vector field $\mathbf{v}$, using the given weight function f, as follows:

$$\begin{aligned}
\operatorname{div}\mathbf{v}(\mathbf{x},t) &= -\frac{\partial}{\partial t}\left(\frac{1}{f(\mathbf{x},t)}\right) \quad \text{in } \Omega && (3)\\
\mathbf{curl}\,\mathbf{v} &= 0 \quad \text{in } \Omega && (4)\\
\mathbf{v}\cdot\mathbf{n} &= 0 \quad \text{on } \Gamma\,. && (5)
\end{aligned}$$

Then, in order to construct the diffeomorphism $\boldsymbol{\varphi}$ we consider the following system of ordinary differential equations for a fixed node $\mathbf{x}$ in Ω:

$$\begin{aligned}
\frac{\partial}{\partial t}\boldsymbol{\varphi}(\mathbf{x},t) &= \mathbf{v}(\boldsymbol{\varphi},t)\cdot f(\boldsymbol{\varphi},t) \quad t > t_0 && (6)\\
\boldsymbol{\varphi}(\mathbf{x},t_0) &= \boldsymbol{\varphi}_0(\mathbf{x}) \quad t = t_0\,. && (7)
\end{aligned}$$

Note that conditions (3) and (6) imply (2). Equation (4) is imposed for two reasons. First, together with the boundary condition (5) it ensures the existence of a unique vector field $\mathbf{v}$. Second, (4) also means that $\mathbf{v}$ is irrotational, which helps to improve the shape of grid cells. Our goal now will be to show that the mapping $\boldsymbol{\varphi}(\mathbf{x},t)$ obeys the principle of equidistribution (2), i.e., that the new coordinates of the grid points computed by $\boldsymbol{\varphi}(\mathbf{x},t)$ are equidistributed according to the given weight $f(\mathbf{x},t)$. This result is established in the next theorem.

Theorem 1 *Let $\boldsymbol{\varphi}$ denote the solution of (6), (7) and let*

$$H = \frac{\det \nabla \boldsymbol{\varphi}(\mathbf{x}, t)}{f(\boldsymbol{\varphi}(\mathbf{x}, t), t)}.$$

Then

$$\frac{\partial H}{\partial t} = 0, \quad \textit{for all } t \geq t_0.$$

Proof:

We have that

$$\frac{\partial H}{\partial t} = \frac{1}{f(\boldsymbol{\varphi}, t)} \frac{\partial}{\partial t} \det \nabla \boldsymbol{\varphi}(\mathbf{x}, t) + \det \nabla \boldsymbol{\varphi}(\mathbf{x}, t) \frac{d}{dt} \left(\frac{1}{f(\boldsymbol{\varphi}, t)} \right).$$

Recall that if

$$\frac{d}{dt} \psi(t) = \mathrm{A}(t) \psi(t)$$

then

$$\frac{d}{dt} \det \psi(t) = \mathrm{Tr A}(t) \det \psi(t).$$

Applying the above with $\psi(t) = \nabla_x \boldsymbol{\varphi}$ and (6), we have

$$\frac{\partial}{\partial t} \nabla(\boldsymbol{\varphi}) = \nabla \frac{\partial}{\partial t}(\boldsymbol{\varphi}) = \nabla(\mathbf{v}(\boldsymbol{\varphi}, t) f(\boldsymbol{\varphi}, t)).$$

Let $\eta = \mathbf{v}(\boldsymbol{\varphi}, t) f(\boldsymbol{\varphi}, t)$. Then

$$\nabla(\mathbf{v}(\boldsymbol{\varphi}, t) f(\boldsymbol{\varphi}, t)) = \nabla_{\boldsymbol{\varphi}} \eta \nabla_x \boldsymbol{\varphi} = (\nabla_{\boldsymbol{\varphi}} \eta) \psi,$$

and

$$\frac{d}{dt} \det\, \psi(t) = \mathrm{Tr}(\nabla_{\boldsymbol{\varphi}} \eta)(\det \psi) = \mathrm{Tr}(\nabla_{\boldsymbol{\varphi}} \eta) \det \nabla_x \boldsymbol{\varphi}.$$

As a result,

$$\frac{d}{dt} J(t) = \operatorname{div}_{\boldsymbol{\varphi}} \eta J(\boldsymbol{\varphi}),$$

where $J(\boldsymbol{\varphi}) = \det\, \psi(t)$. Consider

$$\frac{\dot{H}}{J} = \frac{\operatorname{div} \eta}{f(\boldsymbol{\varphi}, t)} + \frac{d}{dt} \left(\frac{1}{f(\boldsymbol{\varphi}, t)} \right) \tag{8}$$

Since $\mathbf{v} = \eta / f$, it follows that

$$\operatorname{div}_{\boldsymbol{\varphi}} \mathbf{v}(\boldsymbol{\varphi}, t) = \frac{\operatorname{div}_{\boldsymbol{\varphi}} \eta}{f} + \langle \eta, \nabla_{\boldsymbol{\varphi}} \frac{1}{f} \rangle. \tag{9}$$

Now

$$\begin{aligned} \frac{d}{dt} \left(\frac{1}{f(\boldsymbol{\varphi}, t)} \right) &= \left\langle \nabla_{\boldsymbol{\varphi}} \frac{1}{f}, \dot{\boldsymbol{\varphi}} \right\rangle + \frac{\partial}{\partial t} \frac{1}{f(\boldsymbol{\varphi}, t)} \\ &= \left\langle \nabla_{\boldsymbol{\varphi}} \frac{1}{f}, \eta \right\rangle - \operatorname{div}_{\boldsymbol{\varphi}} \mathbf{v}(\boldsymbol{\varphi}, t). \end{aligned} \tag{10}$$

Then, by virtue of (9) and (10), the right hand side of (8) vanishes:

$$\operatorname{div}\boldsymbol{\varphi}\ \mathbf{v} - \left\langle \eta, \nabla\boldsymbol{\varphi}\frac{1}{f}\right\rangle + \left\langle \nabla\boldsymbol{\varphi}\frac{1}{f}, \eta\right\rangle - \operatorname{div}\boldsymbol{\varphi}\mathbf{v}(\boldsymbol{\varphi},t) = 0.$$

As a result, we have that $\frac{\partial H}{\partial t} \equiv 0$.

Corollary 1 *The mapping $\boldsymbol{\varphi}$ constructed above satisfies*

$$\det\ \nabla\boldsymbol{\varphi}(\mathbf{x},t) = f(\boldsymbol{\varphi}(\mathbf{x},t),t) \quad \textit{for all } t \geq t_0, \textit{ all } x \in \Omega.$$

Proof: At $t = 0$, $H(0) = 1$. Hence by Theorem 1

$$H(t) = \frac{\det\ \nabla\boldsymbol{\varphi}(\mathbf{x},t)}{f(\boldsymbol{\varphi}(\mathbf{x},t),t)} = 1$$

for all $t \geq t_0$, all $x \in \Omega$.

The above deformation method was outlined in [16] and [17]. It has its origin in Riemannian geometry through the work of J. Moser [22]. The method has been applied to the grid generation problem in steady form in [15] and [19]. The moving grid method in $1+1$ dimensions was proposed in [18] and was implemented in [16].

Let us now discuss the issue of grid tangling under the mapping $\boldsymbol{\varphi}$. It is well-known that a mapping $\boldsymbol{\sigma} : D_1 \mapsto D_2$ in $\mathbf{R}^n$ maps $\bar{D_1}$ one to one and onto $\bar{D_2}$, if

1. $\boldsymbol{\sigma}_{|_{\partial D_1}}$ maps ∂D_2 one to one and onto ∂D_1;
2. $J(\boldsymbol{\sigma}) > 0$ in the interior of D_1.

The main idea is that $J(\boldsymbol{\sigma}) > 0$ at $\mathbf{x} \in D_1$ implies the existence of an open set around $\mathbf{x}$ which is mapped one to one onto an open set around $\boldsymbol{\sigma}(\mathbf{x}) \in D_2$ by the inverse mapping theorem. By an argument based on degree theory, the global one to one property of $\boldsymbol{\sigma}$ can be obtained if $\boldsymbol{\sigma}$, restricted to ∂D_1, maps ∂D_1 onto ∂D_2 in one to one manner; see e.g., [10] and [23]. Recall that in the present context the Jacobian $J(\boldsymbol{\varphi})$ is equal to the weight function f, which is strictly positive Therefore, if $\boldsymbol{\varphi}(\mathbf{x},t)$ holds each boundary node fixed, then $\boldsymbol{\varphi}$ is one to one and no grid tangling can occur. When the grid points on the boundary are allowed to move along the boundary, then the grid points in the interior will not tangle, provided the boundary nodes do not tangle.

3 Numerical implementation

Numerical implementation of the moving grid algorithm is essentially equivalent to approximate computation of the action of the mapping $\boldsymbol{\varphi}$ on the grid point coordinates $\mathbf{x}$. Thus, successful

implementation of this algorithm hinges on efficient and fast solution of the div-curl system (3)-(5) and the ordinary differential equation (6), (7). Solution of the system (6), (7) is essentially independent from the dimension of the region Ω, and does not pose a serious problem. Here, for this purpose we have chosen an explicit 4-th order Runge-Kutta method. We recall that given an ODE of the form

$$y' = \eta(t, y); \quad y(t_0) = y_0 \,,$$

the 4-th order Runge-Kutta scheme is given by

$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \,, \\ k_1 &= \Delta t \eta(x_i, y_i) \\ k_2 &= \Delta t \eta\left(t_i + \frac{\Delta t}{2}, y_i + \frac{k_1}{2}\right) \\ k_3 &= \Delta t \eta\left(t_i + \frac{\Delta t}{2}, y_i + \frac{k_2}{2}\right) \\ k_4 &= \Delta t \eta(t_i + \Delta t, y_i + k_3) \,., \end{aligned} \tag{11}$$

where Δt denotes the time step. More sophisticated methods, such as predictor-corrector schemes can be used in the implementation as well; see, e.g., [19].

Within the context of the grid moving method an effective solver for (3)-(5) must satisfy several stringent conditions. First, the solver should be easy to implement for regions in $\mathbf{R}^2$ and $\mathbf{R}^3$ with complex geometries. This requirement stems from the fact that boundary conditions of the form (5) are not easy to satisfy exactly, unless the domain Ω is polyhedron or polygon. Second, this solver should not be computationally demanding. Last, but not least, the solver should not require significant additional storage. To meet all these criteria here we shall use a finite element solver based on application of least-squares variational principles. The use of such principles avoids the inf-sup condition and permits discretization of the equations (3), (4) using standard finite element spaces. Moreover, the resulting algebraic system is symmetric and positive definite, and can be solved using robust and efficient iterative methods. For example, solution of this system can be accomplished by means of preconditioned conjugate gradients method. As a result, solution method for (3)-(5) can be devised without assembly of the discretization matrix, even at the element level. Finally, the use of least-squares principles allows one to enforce boundary condition (5) in a weak, variational sense, which is beneficial when (3)-(5) must be solved on regions with complicated geometries.

3.1 Finite Element Method for div-curl System

In this section we introduce a least-squares variational principle for the div-curl system

$$\operatorname{div} \mathbf{u} = f \quad \text{in } \Omega \tag{12}$$

$$\mathbf{curl}\, \mathbf{u} = \mathbf{g} \quad \text{in } \Omega \tag{13}$$

along with the boundary conditions (5). We first show that the least-squares principle leads to a well-posed variational problem. Then we formulate a finite element method for approximate numerical solution of (12)-(13), and (5), and present discretization error estimates.

We consider the following least-squares quadratic functional

$$\mathcal{J}(\mathbf{u}, f, \mathbf{g}) = \|\mathrm{div}\,\mathbf{u} - f\|_0^2 + \|\mathbf{curl}\,\mathbf{u} - \mathbf{g}\|_0^2 \tag{14}$$

The least-squares variational principle for (14) is given by

$$\text{Seek } \mathbf{u} \in \mathbf{H}_n^1(\Omega) \text{ such that} \quad \mathcal{J}(\mathbf{u}, f, \mathbf{g}) \leq \mathcal{J}(\mathbf{v}, f, \mathbf{g}) \quad \text{for all } \mathbf{v} \in \mathbf{H}_n^1(\Omega). \tag{15}$$

It is not difficult to see that the Euler-Lagrange equation for (14) is equivalent to following variational problem

$$\text{Seek } \mathbf{u} \in \mathbf{H}_n^1(\Omega) \text{ such that} \quad \mathcal{B}(\mathbf{u}, \mathbf{v}) = \mathcal{F}(\mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{H}_n^1(\Omega). \tag{16}$$

where

$$\mathcal{B}(\mathbf{u}, \mathbf{v}) = (\mathrm{div}\,\mathbf{u}, \mathrm{div}\,\mathbf{v}) + (\mathbf{curl}\,\mathbf{u}, \mathbf{curl}\,\mathbf{v}) \tag{17}$$

and

$$\mathcal{F}(\mathbf{v}) = (\mathrm{div}\,\mathbf{v}, f) + (\mathbf{curl}\,\mathbf{v}, \mathbf{g})\ . \tag{18}$$

To prove existence and uniqueness of minimizers of (14) out of $\mathbf{H}_n^1(\Omega)$ we shall need the following result which can be found in [12].

Lemma 1 *Assume that Ω is simply connected domain in $\mathbf{R}^n$, $n = 2, 3$. Assume that the boundary Γ is Lipschitz-continuous or piecewise smooth with no reentrant corners. Then, for all functions in $\mathbf{H}_n^1(\Omega)$ there exists a positive constant C such that*

$$\|\mathbf{v}\|_1 \leq C\left(\|\mathrm{div}\,\mathbf{v}\|_0 + \|\mathbf{curl}\,\mathbf{v}\|_0\right) \tag{19}$$

Lemma 1 essentially means that $\sqrt{\mathcal{B}(\mathbf{u}, \mathbf{u})}$ is equivalent to the H^1 norm on the space $\mathbf{H}_n^1(\Omega)$. Then, using a standard application of Lax-Milgram Lemma one can prove the following theorem.

Theorem 1 *Assume that $f \in L^2(\Omega)$ and $\mathbf{g} \in L^2(\Omega)$ for $N = 2$ or $\mathbf{g} \in L^2(\Omega)^3$ for $N = 3$. Then the variational problem (16) has unique solution $\mathbf{u} \in \mathbf{H}_n^1(\Omega)$ and*

$$\|\mathbf{u}\|_1 \leq C\left(\|f\|_0 + \|\mathbf{g}\|_0\right)\ . \tag{20}$$

In what follows we define the least-squares finite element method for (12)-(13) and (5), and present the error estimates. For more detailed studies of methods of least-squares type we refer the reader to [2], [5] and references therein. Our main goal is to prove that the finite element approximations converge to all sufficiently smooth solutions of (12)-(13) and (5) at optimal rate. For simplicity error analysis is presented under the assumption that the finite element spaces satisfy exactly the boundary condition (5). To devise a method in which discrete spaces are not required to satisfy (5) the functional (14) can be augmented with appropriately weighted boundary norm of $\mathbf{u} \cdot \mathbf{n}$, i.e., one has to consider functionals of the form

$$\mathcal{J}(\mathbf{u}, f, \mathbf{g}) = \|\mathrm{div}\,\mathbf{u} - f\|_0^2 + \|\mathbf{curl}\,\mathbf{u} - \mathbf{g}\|_0^2 + h^{-1}\|\mathbf{u} \cdot \mathbf{n}\|_{0,\Gamma}^2\,. \tag{21}$$

For examples of such methods, and their analysis we refer the reader to [2] and [25], among others.

The finite element method for (12), (13) and (5) is defined by conforming discretization of variational problem (16). This can be done in a completely standard manner; thus, here we only sketch the details. Let $\mathcal{T}$ denote uniformly regular triangulation of the domain Ω into finite elements. We consider finite element spaces $S_h \subset H^1(\Omega)$ with the following approximation property: there exists an integer $d \geq 0$ such that for all $u \in H^{d+1}(\Omega)^n$ one can find $u^h \in S_h$ with

$$\|u - u^h\|_r \leq Ch^{d+1-r}\|u\|_{d+1} \tag{22}$$

$r = 0, 1$. For details concerning construction of such spaces the reader may consult [6] and [7]. Let us define the following finite element space

$$\mathbf{S}_h = \{\mathbf{u}^h \in S_h \times S_h \,|\, \mathbf{u}^h \cdot \mathbf{n} = 0 \quad \text{on } \Gamma\}\,. \tag{23}$$

Then the discrete counterpart of (16) is given by

$$\text{Seek } \mathbf{u}^h \in \mathbf{S}_h \text{ such that} \quad \mathcal{B}(\mathbf{u}^h, \mathbf{v}^h) = \mathcal{F}(\mathbf{v}^h) \quad \text{for all } \mathbf{v}^h \in \mathbf{S}_h. \tag{24}$$

It is not difficult to see that (24) is necessary condition for the discrete least-squares principle

$$\text{Seek } \mathbf{u}^h \in \mathbf{S}_h \text{ such that} \quad \mathcal{J}(\mathbf{u}^h, f, \mathbf{g}) \leq \mathcal{J}(\mathbf{v}^h, f, \mathbf{g}) \quad \text{for all } \mathbf{v}^h \in \mathbf{S}_h. \tag{25}$$

As a result, one can show that (24) is a linear system of algebraic equations with *symmetric and positive definite* matrix. The error estimates for the least-squares method are summarized in the following theorem.

Theorem 2 *The discrete problem (24) has unique solution $\mathbf{u}^h \in \mathbf{S}_h$. This solution is the minimizer of (14) out of $\mathbf{S}_h$. Furthermore, assume that f and $\mathbf{g}$ are such that $\mathbf{u} \in \mathbf{H}^1_n(\Omega) \cap [H^{m+1}(\Omega)]^N$ and let $\tilde{d} = \min\{d, m\}$ where d is the integer from (22). Then, there exists a positive constant C, independent of h, such that*

$$\|\mathbf{u} - \mathbf{u}^h\|_0 \leq Ch^{\tilde{d}}\|\mathbf{u}\|_{\tilde{d}+1} \tag{26}$$

The proof of this theorem follows by standard elliptic finite element theory, and is omitted.

3.2 Implementation

Let us assume that $\{\mathbf{x}_i\}_{i=1}^N$ is an initial uniform grid defined in Ω, and let Δt denote given time step. We seek to approximate the mapping $\boldsymbol{\varphi}(\mathbf{x}, t_0 + \Delta t)$ which transforms the initial uniform grid into the grid $\{\boldsymbol{\varphi}(\mathbf{x}_i, t_0 + \Delta t)\}_{i=1}^N$. For this purpose we first compute finite element approximation for the vector field $\mathbf{v}$. Then, for each grid point $\mathbf{x}_i$ we solve the system (6-(7) using the Runge-Kutta scheme (11). As a result, we obtain an approximate grid of the form $\{\boldsymbol{\varphi}^h(\mathbf{x}_i, t_0 + \Delta t)\}_{i=1}^N$, where $\boldsymbol{\varphi}^h$ denotes the approximation of the mapping $\boldsymbol{\varphi}$ resulting from the above process. Then, we repeat the same process, taking the grid computed at $t = t_0 + \Delta t$ as initial grid for the next time step.

4 Numerical results

In this section we report results of several computational experiments with the grid moving method in one and two-dimensions. Our main objectives are to illustrate the analytic properties of the moving grid algorithm established in Section 2. In particular, we wish to demonstrate the adaptivity of the resulting grids to the underlying solutions, and second, we wish to illustrate the lack of mesh tangling in two-dimensional grids. For this purpose we begin with uniform grids of various sizes in one and two-dimensions. Then we choose an analytic weight function and apply the moving grid algorithm to the initial uniform grid. In two-dimensions experiments were carried on for rectangular, and L-shaped regions. Although this setting is simplified, it allows us to demonstrate the relevant analytic properties of our grid moving algorithm. In the experiments we have used several well-known weight functions which serve as popular test cases for adaptive methods; see e.g., [1], [11], and [14].

The least-squares method (24) has been implemented using uniformly regular triangulations $\mathcal{T}$ of Ω into triangles $\mathcal{K}$. The finite element space S_h has been chosen as follows

$$S_h = \{u^h \in C^0(\Omega) \,|\, u^h_{|\mathcal{K}} \in P_2\}\,,$$

where P_2 denotes the space of all second degree polynomials in $\mathbf{R}^n$. For such spaces approximation property (22) holds with $d = 2$, see e.g. [6]. To solve the system (6), (7) we use fourth order Runge-Kutta scheme (11). All codes were written in FORTRAN and were ran on a VAX Station 4000/90 and a Power Macintosh 8100/100 in double precision. In what follows we present the weight functions used in computations and briefly discuss the results.

Example 1

We consider a weight function of the form

$$f(x,t) = \frac{C}{\sqrt{1 + \left(\frac{\partial u}{\partial x}\right)^2}}\,,$$

where the function $u(x,t)$ is given by

$$u(x,t) = e^{-\pi^2 t}\sin(\pi x), \qquad 0 \le x \le 1\,.$$

This function has been used to study stability of grids for time-dependent PDE's in [14]. Since $u_x(x,t) = \pi e^{-\pi^2 t}\cos(\pi x) \to 0$, as $t \to +\infty$, we have that $f(x,t) \to 1$. We recall that since the weight function $f(x,t)$ is strictly positive, theoretically no grid crossing should occur. Our computational results are summarized on Fig. 1. This figure illustrates the evolution paths of the initial uniform grid under the mapping $\boldsymbol{\varphi}$ for $t \in [0,1]$. In particular, we start with 20 uniformly spaced grid points and compute each new grid point coordinate using $\Delta t = 0.1$. Computations with final times $t > 1$ were also carried on with very similar results. In all cases paths of the grid points did not cross each other.

Example 2

In this example we consider weight function similar to one in [14]. We set

$$f(x,t) = \frac{C}{u(x,t)},$$

where for $0 \le x \le 1$

$$u(x,t) = \begin{cases} \dfrac{ta(t)}{\cosh^2(a(t)(x-t-0.3)) + 1.0 - 8t} & 0 \le t \le 0.1 \\ \dfrac{0.1a(t)}{\cosh^2(a(t)(x-t-0.3)) + 0.2} & t > 1 \end{cases},$$

$$a(t) = 1 + 3[1 + \tanh(20(t-0.2))].$$

The function $u(x,t)$ is initially smooth but develops a steep gradient at approximately $t = 0.2$. Then, as t approaches infinity, $u(x,t)$ again becomes smooth. For this example we have also started with a grid consisting of 20 uniformly spaced points in the interval $[0,1]$. Evolution paths of these points computed for $t \in [0,1]$ using time step $\Delta t = 0.1$, are presented on Fig.2. This figure illustrates very well how the grid moving algorithm captures accurately the salient features of $u(x,t)$. Most importantly, we see that after $u(x,t)$ smoothes, the grid returns to a uniform configuration.

Example 3

The weight function for this example is motivated by an oil reservoir simulation problem; see [11]. We let

$$u(x,y,t) = \sqrt{\frac{td}{\pi}} e^{-tdr} + 1.0 - 0.5t \quad 0 \le x, y \le 1,$$

where $a = 0.5, b = 0.5, d = 40.0$ and $r = (x-a)^2 + (y-b)^2$. The weight function is then defined as follows

$$f(x,y,t) = \frac{C}{u(x,y,t)}.$$

Note that $f(x,y,0) = 1$ which is consistent with an initial uniform grid.

The function u represents a spike function, where location of the spike is determined by the constants a and b. In this example, the spike is located in the center of the unit square in $\mathbf{R}^2$. We have applied the grid moving algorithm for various uniform grids defined on the unit square $[0,1] \times [0,1]$ in $\mathbf{R}^2$, and on an L-shaped region. Computations were carried out with $\Delta t = 0.1$, and $t \in [0,1]$. For both regions we have obtained grids similar to the ones presented on Figures 3-4, respectively. In particular, Fig.3 illustrates evolution of 25x25 uniform grid as the spike develops (see the contour plots). We stress upon the fact that in addition to the good adaptivity of resulting grids, the moving grid algorithm does not introduce mesh tangling.

Example 4

The weight function $f(x,y,t)$ is as in the previous example, however, now we consider the following function $u(x,y,t)$:

$$u(x,y,t) = \sqrt{\frac{td}{\pi}} e^{-tdr_1} + ce^{-2tdr_2} + 1.0 - 0.5t \quad 0 \le x, y \le 1,$$

where

$$\begin{aligned} r_1 &= (x-a_1)^2 + (y-b_1)^2, \\ r_2 &= (x-a_2)^2 + (y-b_2)^2, \end{aligned}$$

$a_1 = b_1 = 0.25, a_2 = b_2 = 0.75, d = 40$ and $c = 1$. The function $u(x, y, t)$ represents a double-spike function. Locations of the spikes are determined by the constants a_1, b_1, a_2, b_2. Again, at $t = 0$ we have that $f(x, y, 0) = 1$, that is, the initial grid can be any uniform grid. Initially, both spikes start to grow. One of the spikes, however, stops growing at $t = 0.5$, after which it gradually disappears. Contour plots of the weight function and the corresponding grids, computed with $\Delta t = 0.1$, are presented on Figure 5. This figure illustrates capability of our grid moving algorithm to redistribute the grid points in a way which reflects the behavior of the underlying solution. We also note that no grid entanglements were encountered .

Example 5

Our final example illustrates an adaptive grid clustered around a sinusoidal curve. The weight function f (up to a normalization constant) is given by:

$$f(x,y,t) = \begin{cases} 1 & 0 \le y < r - 0.2 \\ 0.5 - 2.5(y-r)t + (1-t) & r - 0.2 \le y \le r \\ 0.5 + 2.5(y-r)t + (1-t) & r < y \le r + 0.2 \\ 1 & r + 0.2 < y \le 1 \end{cases},$$

where

$$r = \frac{1}{2} + \frac{1}{4}\sin(2\pi x).$$

This example is similar to the one considered in [1]. On Figure 6 we present surface plots of the weight function f along with the corresponding grids for $t = 0$, $t = 0.5$ and $t = 1.0$. At the initial time $t = 0$ we have specified a 25 by 25 uniform grid. We see that as the sinusoidal shape of the weight function becomes more pronounced, it is immediately reflected in a similar sinusoidal clustering of the grid points. Computations for this example were carried on with $\Delta t = 0.1$ and $t \in [0, 1]$. We note that as in the previous examples, no grid crossing occurs.

5 Conclusion

We have formulated and analyzed a moving grid approach to adaptive grid generation in one, two, and three-dimensions. Most importantly, we have established rigorously that our method does not lead to tangled grids in two and three-dimensions. Computational experiments in one and two-dimensions indicate that the moving grid method is robust and efficient, and that it can accurately capture the regions where the need for higher resolution exists. Furthermore, efficiency of the method in the context of a PDE solver for transient problems is enhanced by its ability to redistribute the grid points according to the changes occurring in the underlying solution. Here we have only considered the moving grid algorithm. An adaptive solver which combines the moving grid method with a streamline upwind Petrov-Galerkin scheme in one-dimension has been proposed in [16]. This method has been successfully used for numerical solution of convection-diffusion problem

with a shock. Presently, coupling of the moving grid method with PDE solvers in two-dimensions, as well as a design of an efficient solver for the div-curl system in three-dimensions are underway. This work will be reported in a forthcoming paper.

Acknowledgments. Authors are grateful to the anonymous referees for the thoughtful and thorough remarks which helped to improve significantly the style and contents of this paper.

References

[1] D. A. Anderson, *Equidistribution Schemes, Poisson Generators, and Adaptive Grids* Applied Mathematics and Computation, 24 (1987) pp. 211-227.

[2] A. Aziz, R. Kellogg, and A. Stephens; Least-squares methods for elliptic systems, Math. of Comp., **44**,169, 1985, 53-70.

[3] I. Babuska, O.C. Zienkiewicz, J. Cago and E.R. Oliveira, eds., *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, J. Wiley, New York 1986.

[4] R.E. Bank, A.H. Sherman, *An adaptive, Multi-Level Method for Elliptic Boundary Value Problems*, Computing 26/2, 1980, 92-105.

[5] P. Bochev, M.D. Gunzburger, *Analysis of least squares methods for the Stokes equations*, Math. Comp., 63/108, pp. 479-506, 1994.

[6] S.C. Brenner and L.R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer, 1994.

[7] P. Ciarlet, *Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1978.

[8] J.M. Coyle, J.E. Flaherty and R. Ludwig, *On the Stability of Mesh Equidistribution Strategies for Time- dependent Partial Differential Equations*,J. Comput. Phys., 62 (1986), pp. 26-39.

[9] A. Dorfi and L. Drury, *Simple adaptive grids for 1-D initial value problems,* J. Comput. Phys., 69 (1987) pp. 175-195.

[10] J. Dugundji, *Topology*, Allyn and Bacon, Boston, (1966).

[11] R.E. Ewing, R.D. Lazarov, T.F. Russell, P.S. Vassilevski, *Local Refinement Via Domain Decomposition Techniques for Mixed Finite Element Methods with Rectangular Raviart-Thomas Elements* in: Proceedings 1st Int. Conf. on Domain Decomposition methods, 1988, 98-113.

[12] V. Girault, P.A. Raviart, *Finite Element Methods for Navier-Stokes Equations*, Springer, 1986.

[13] D. F. Hawken, J. J. Gottlieb and J. S. Hansen *Review of some adaptive node-movement techniques in finite element and finite difference solutions of PDEs*, J. Comput. Phys., 95 (1991) pp. 254-302.

[14] W. Huang, Y. Ren and R. Rusell, *Moving Mesh Partial Differential Equations (MMPDEs) based on the Equidistribution Principle* SIAM J. Numer. Anal., 31 (1994) pp. 709-730.

[15] G. Liao and D. Anderson, *A New Approach to Grid Generation*, Appl. Anal. 44, 285 (1992)

[16] B. Semper and G. Liao, *A Moving Grid Finite Element Method using Grid Deformation,* Numer. Methods for Partial Differential Equations 11, pp. 603-615, (1995)

[17] G. Liao, *"Adaptive Moving Grid Methods by Real Time Deformation"*, presentation to the Special Session on Grid Generation, SIAM Summer Meetings, San Diego, 1994

[18] G. Liao and J. Su, *A Moving Grid Method for* $(1+1)$ *dimension*, Appl. Math. Letters, Vol. 8, 4, pp.47-49, 1995

[19] G. Liao, T. Pan and J. Su, *Numerical Grid Generator Based on Moser's Deformation Method*, Numer. Methods for Partial Differential Equations, 10 (1994), pp. 21-31.

[20] K. Miller, *Moving Finite Elements II* SIAM J. Numer. Anal., 18 (1981) pp. 1033-1057.

[21] K. Miller and R.N. Miller, *Moving Finite Elements I* SIAM J. Numer. Anal., 18 (1981), pp. 1019-1032.

[22] J. Moser, *N, The Volume Elements on a Manifold*, Trans. Am. Math. Soc. 120, 286 (1965)

[23] S. S. Sritharan, Mathematical aspects of harmonic grid generation, in *Mathematical Aspects of Grid Generation*, SIAM Frontiers in Applied Mathematics, edited by J. Castillo, SIAM Philadelphia, 1991

[24] J.F. Thompson, Z.U.A. Warsi and W.C. Mastin, *Numerical Grid Generation:Foundations and Applications*, North-Holland, New York, 1985.

[25] W. Wendland, *Elliptic Systems in the Plane*, Pitman, London, 1979.

[26] P.A. Zegeling, *Moving-Grid Methods for Time-Dependent Partial Differential Equations*, CWI Tract, Netherlands, 1993.

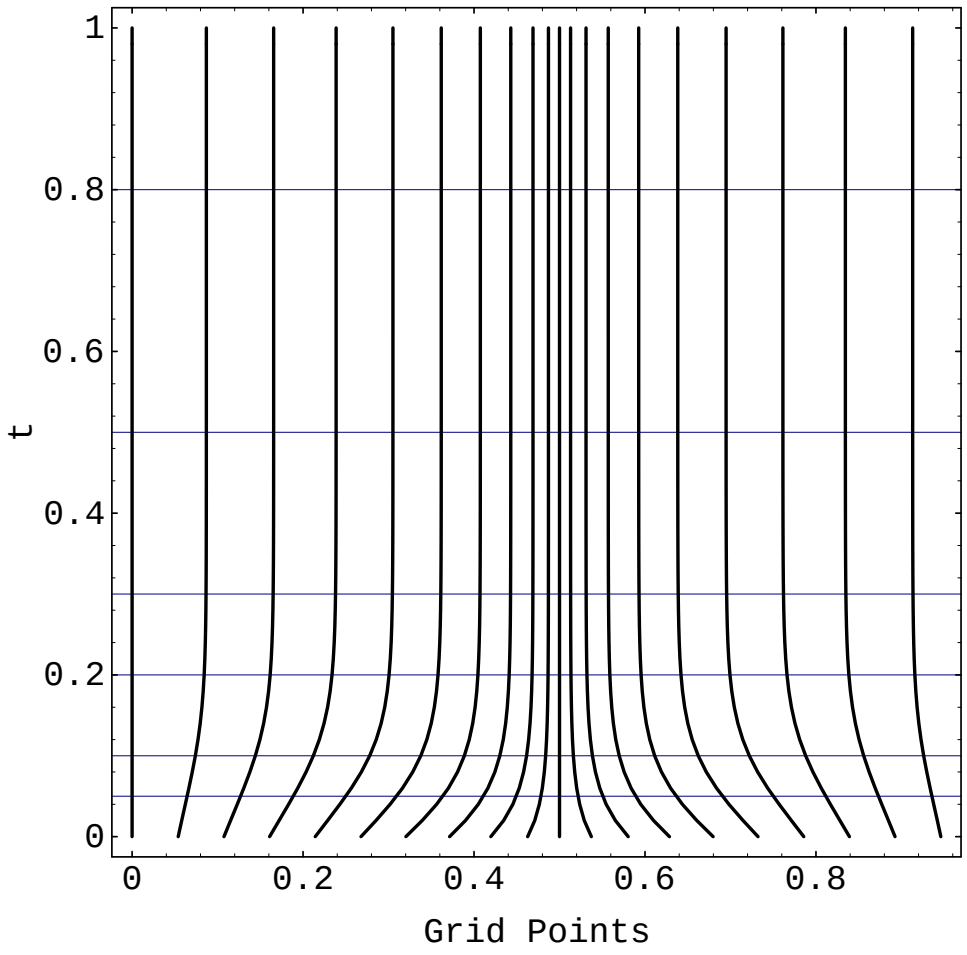

Figure 1: One dimensional grids for Example 1

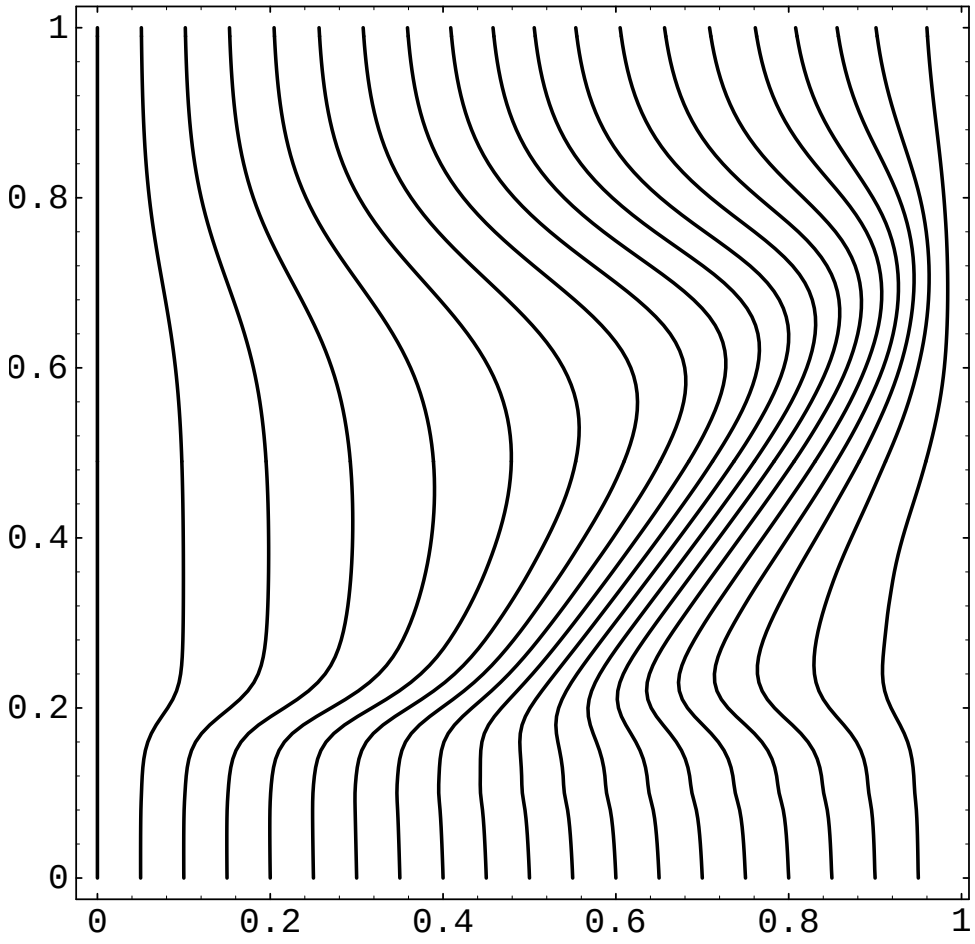

Figure 2: One dimensional grids for Example 2

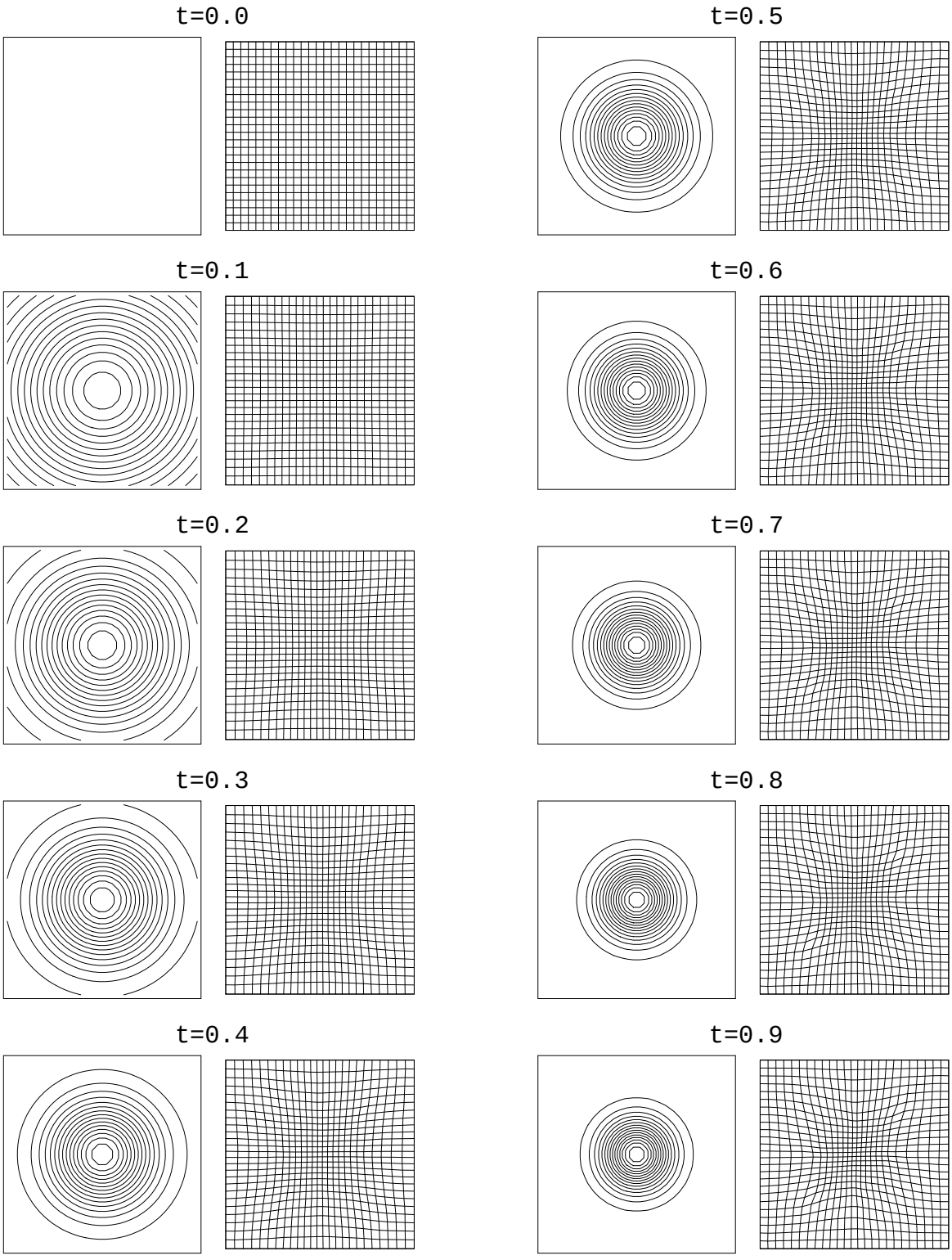

Figure 3: Weight function contours and corresponding 25x25 grids for Example 3

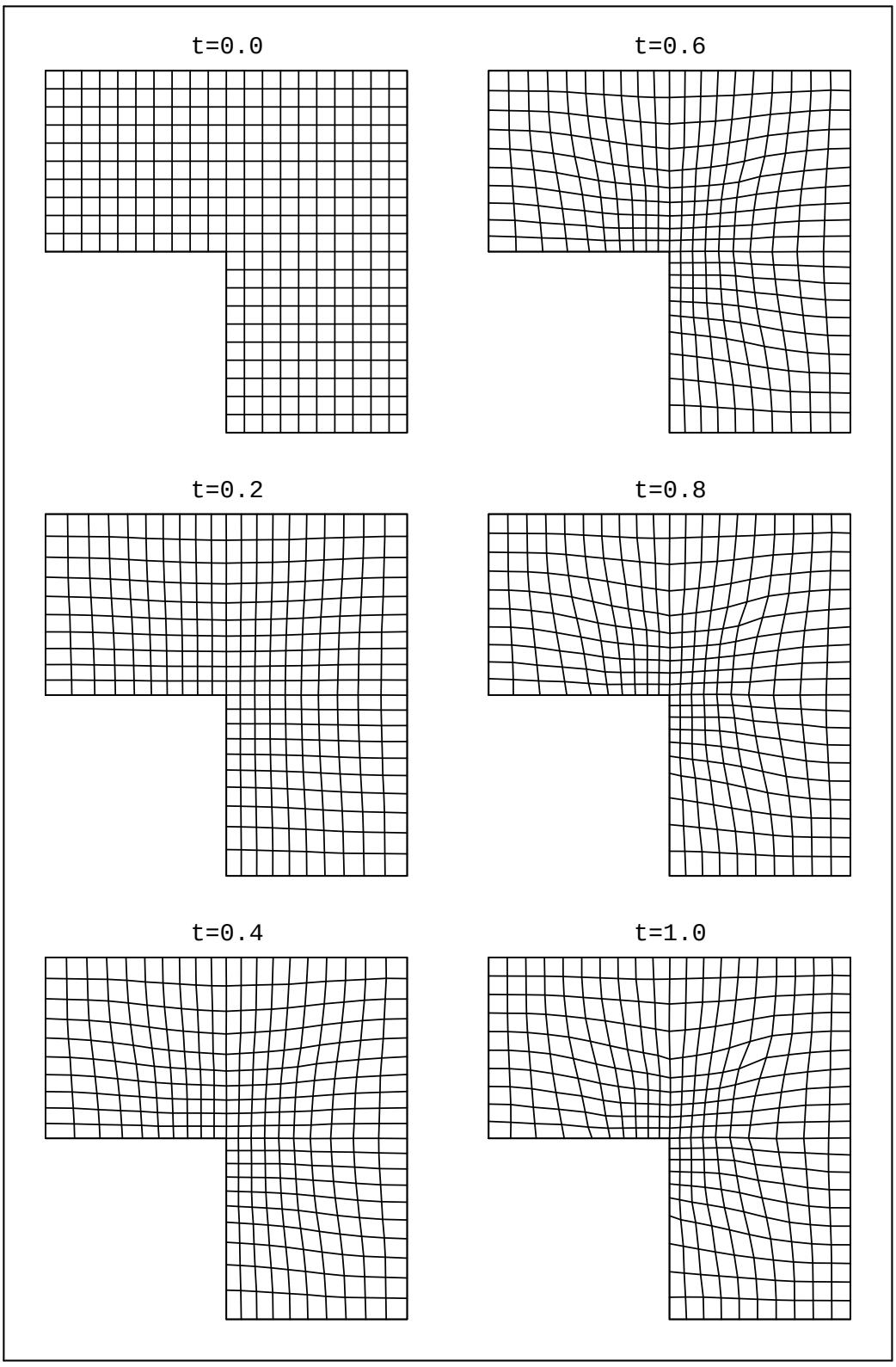

Figure 4: 20x20 grids for Example 3 on L-shaped domain

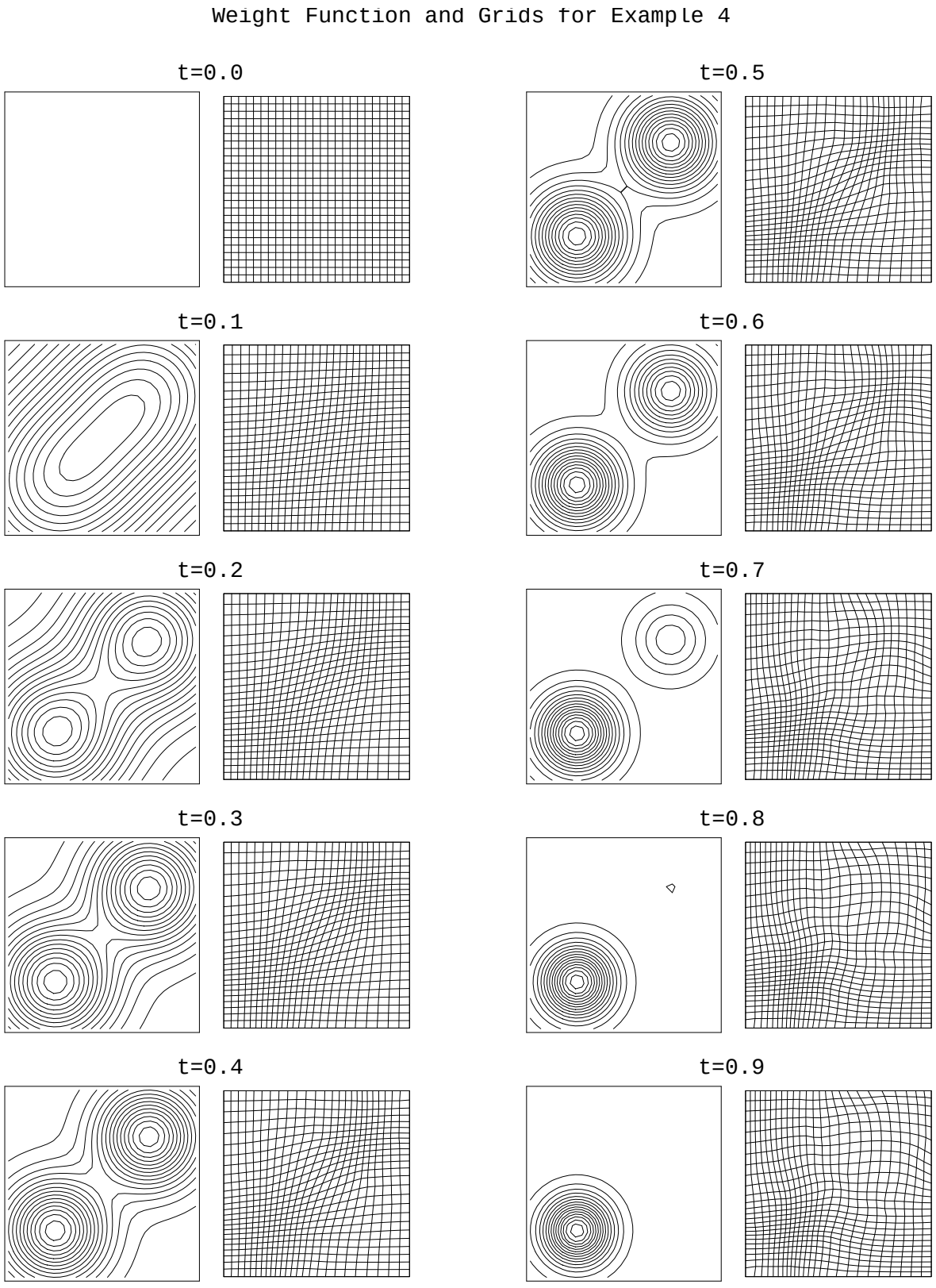

Figure 5: Weight function contours and corresponding 25x25 grids for Example 4

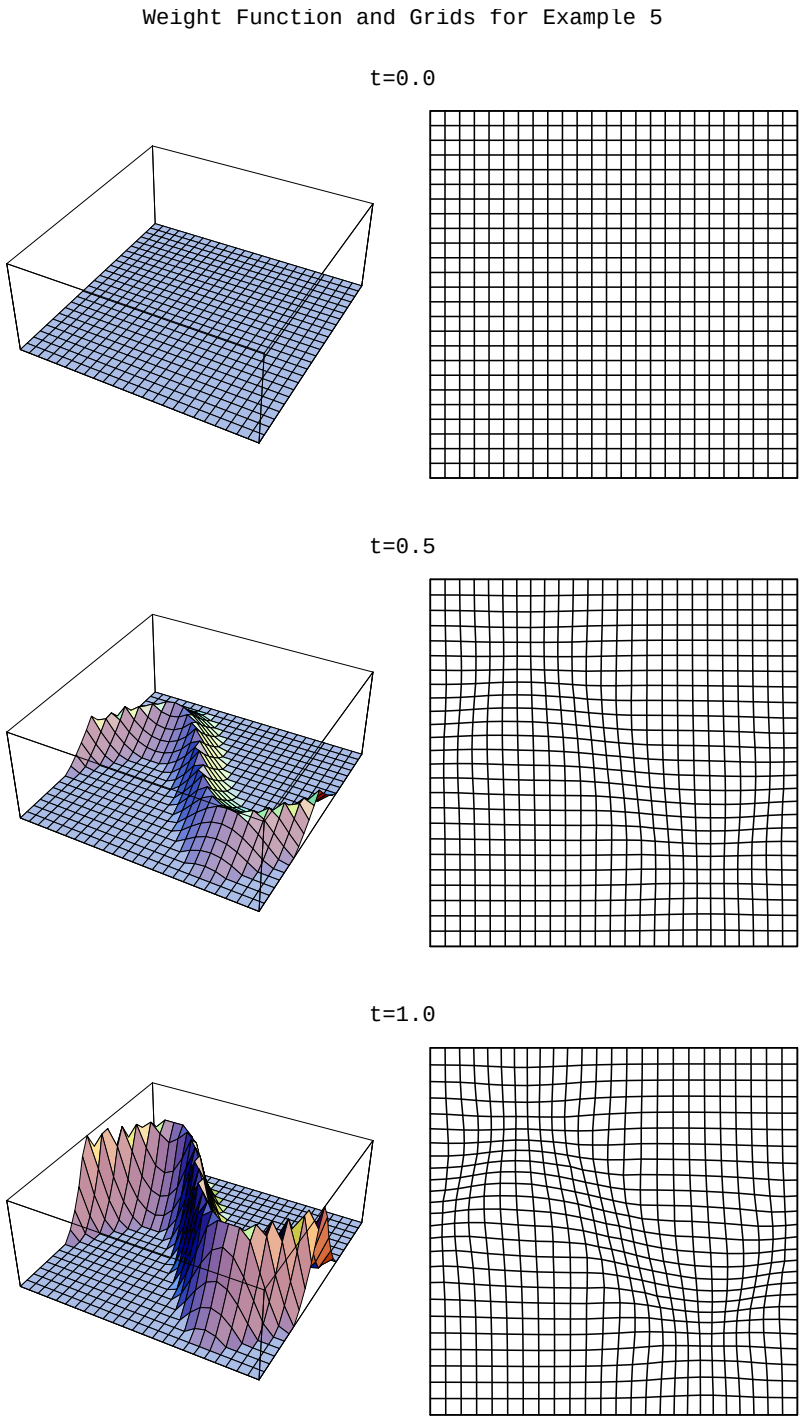

Figure 6: Weight function surface plots and corresponding 25x25 grids for Example 5