Gen. Math. Notes, Vol. 3, No. 1, March 2011, pp.55-72

ISSN 2219-7184;

www.i-csrs.org

Available free online at http://www.geman.in

A Method for Orthogonal Grid Generation

Mehmet Ali Akinlar[1], Stephen Salako[2] and Guojun Liao[3]

[1]Faculty of Engineering and Natural Sciences
Sabanci University
34956, Istanbul, Turkey
E-mail: mehmetaliakinlar@gmail.com

[2]Department of Mathematics
The University of Texas at Arlington
Arlington, TX, 76019, USA
E-mail: st_salako@hotmail.com

[3]Department of Mathematics
The University of Texas at Arlington
Arlington, TX, 76019, USA
E-mail: liao@uta.edu

(Received: 30-11-10/ Accepted: 1-12-10)

Abstract

A grid is called orthogonal if all grid lines intersect at a right angle. An orthogonal grid offers significant advantages in the solution of systems of partial differential equations and in the simulations of computational fluid dynamics. In this paper we present a variational method which generates nearly orthogonal grids with suitable parameter values. We optimize a cost functional which consists of only area and orthogonality quantities. A significant contribution is that no volume functional needs to be added to the cost functional. We achieve to generate nearly orthogonal grids without changing the cell size distribution of the initial grids by well-achieved deformation based grid generation method.

Keywords: *Grid orthogonality, Grid deformation, Euler equations.*

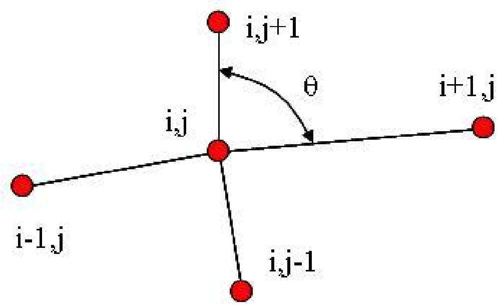

Figure 1: An illustration of intersection of grid lines

1 Introduction

Grids (or meshes) are significant tools and have broad application areas in many different areas of computational science. Typical uses of grids are simulation of physical domains in the computational fluid dynamics and solving partial differential equations (PDEs). A poorly constructed grid may cause erroneous results and effect the numerical computations negatively. A well-designed grid should be orthogonal. The orthogonality of a grid is characterized by the intersection angle θ between the grid lines (see Figures 1 and 3). A grid is called orthogonal if all grid lines intersect at a right angle.

In order to solve PDEs posed on spatial domains, a grid is imposed on the computational domain. There are several local features of the grids that are important for the accuracy and the efficiency of a computation. Some of these local properties are the expansion rate of adjacent grid cells, the ratio of the side lengths of the grid cells, and the orthogonality of the grid lines. In this paper our major concern will be the latter one.

This paper is organized as follows: In the second section we overview some related work for orthogonal grid generation. Section third deals with the so-called deformation based grid generation method which plays an important role for generating the initial non-orthogonal grid. In the section four we present our method which generates nearly orthogonal grids with suitable parameter values. We achieve this by minimizing a cost functional from which we obtain some Euler-Lagrange equations. In the final chapter we show how to solve these equations via successive-over-relaxation method and present some computational examples to illustrate the power of our method.

2 Orthogonal Grid Generation Methods

In this section we briefly review some of the other approaches for orthogonal grid generation. The algebraic, trajectory and field methods are the most common types of orthogonal grid generating methods in the literature.

The algebraic grid generation methods [1], [22] usually adopt transfinite interpolations combined with Bezier and B-spline curves to improve grid qualities. These techniques are usually good, however, they can not guarantee the orthogonality and the grid may be overlapped.

Trajectory methods generate an orthogonal grid from an existing non-orthogonal grid. In general, the trajectory methods use a marching process to recalculate the grid node distribution along the retained set of grid lines in such a way that the intersection between the new grid lines and the retained set of grid lines is orthogonal. These types of methods allow the specification of the grid node distribution on three of the four boundaries of the domain. The major shortcomings with the trajectory methods are the dependency of the orthogonal system on the non-orthogonal original grid and the requirement that in singly connected regions, the components of the boundary must be orthogonal; because, otherwise, trajectories may leave the physical domain. Further information about trajectory methods can be seen in [6], [22] and [23] and in the references therein.

The field methods are based on the solution of a system of PDEs. We can separate this approach to two main groups as hyperbolic systems and elliptic systems. Hyperbolic systems of PDEs which has some similarities with the orthogonal trajectories methods imply that one of the boundaries must be completely free. The solution is obtained by a marching procedure which starts from a known boundary and proceeds toward the free boundary. Elliptic systems of PDEs are based on conformal mappings. Drawbacks and advantages of field methods based on the solutions of elliptic PDEs are discussed in [22] and [23] in detail.

In our method we use a well-achieved method known as the deformation based grid generation method (or grid deformation method (GDM) in short) by means of which we first generate initial (non-orthogonal) grids and then we obtain nearly orthogonal grids with suitable parameters. The GDM is able to generate a grid with desired grid density distribution that is free from grid folding. This method gives direct control over the cell size of the adaptive grid and determines the node velocities directly. The adaptive grid system naturally distributes more grids to deprived areas. The positive monitor function disallows grid folding and provides a mean to control the ratio of the areas between the original and transformed domain. Next we present a formal description of the GDM.

3 The Grid Deformation Method

In this section we overview the grid deformation method (GDM) that is used for construction of differentiable and invertible transformations to solve mesh adaption problems. We formulate a moving-grid algorithm by means of the deformation method. The idea of this method is to move the nodes with correct velocities so that the nodal mapping has a desirable Jacobian determinant. This method was developed in ([10],[11], [16]) and was improved in [13] and [15]. It was used with a finite-volume solver in flow calculations in [18]. A 2D dimension version of the method was proposed in [14] and used with a discontinuous Galerkin finite-element method in solving a convection-diffusion problem in [20]. The method was applied to the registration of non-rigid 2D images and data set alignment at [9] and [12], respectively. Three versions of the grid deformation method are available [17] in the literature. In this paper we adopt the first version of the grid deformation method which is one of the steady versions of deformation method where the transformation Jacobian determinant is specified on the old grid before adaption. Next we describe the first version of the GDM.

Problem statement: Let $\Omega \subset \mathbf{R}^n$ be a bounded convex set with Lipschitz continuous boundary $\partial\Omega$, and a differentiable function $f : \Omega \to \mathbf{R}_+$, namely, monitor (or weight) function, is given such that

$$\int_\Omega (f-1) = 0, \quad \text{or equivalently} \quad \int_\Omega f = |\Omega|,$$

where $|\Omega|$ is the volume of the domain Ω. Find a mapping function

$$\phi_1 : \Omega \to \Omega, \quad \partial\Omega \to \partial\Omega,$$

such that

$$J(\phi_1) := \det \nabla\phi_1(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} = (x_1, x_2) \quad \text{for} \quad n = 2. \tag{3.1}$$

Construction of such a map: Using the following steps, we construct ϕ_1.

(1) Find a vector field $u(\mathbf{x})$ which satisfies

$$\begin{aligned} div\, u(\mathbf{x}) &= f(\mathbf{x}) - 1 \quad \text{in} \quad \Omega \\ n \cdot u(\mathbf{x}) &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned}$$

(2) Form a velocity vector field,

$$h(t,\mathbf{x}) = \frac{u(\mathbf{x})}{t + (1-t)f(\mathbf{x})}, \quad 0 < t \le 1$$

and then, find $\phi_t(\mathbf{x}) = \phi(t,\mathbf{x})$, $\phi_1(\mathbf{x}) = \phi(1,\mathbf{x})$ by solving the ordinary differential equation

$$\begin{aligned} \frac{d\phi(t,\mathbf{x})}{dt} &= h(t,\phi(t,\mathbf{x})), \quad 0 < t \le 1 \\ \phi(0,\mathbf{x}) &= \mathbf{x}, \end{aligned}$$

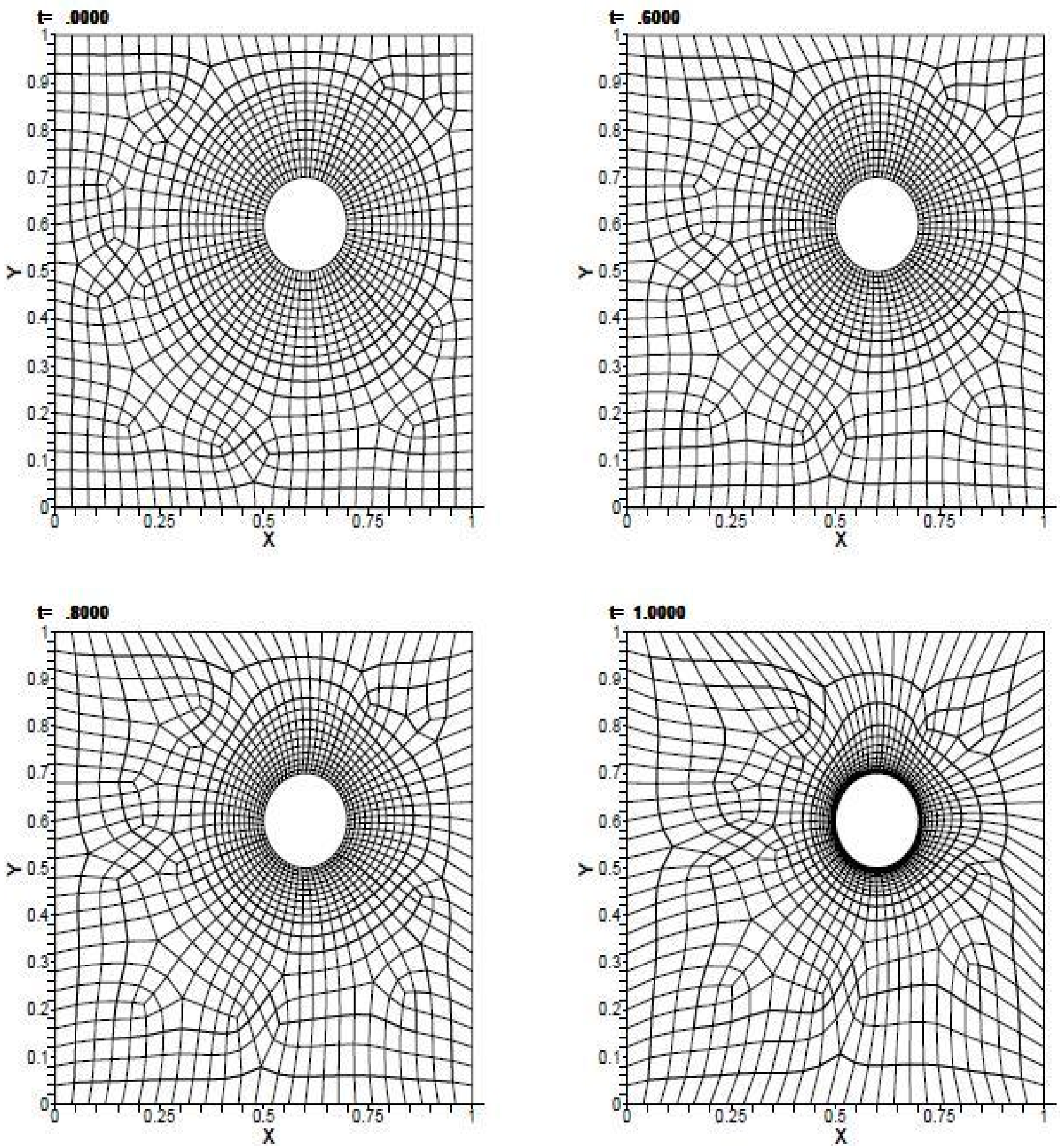

Figure 2: GDM applied to a circle centered at $(0.6, 0.6)$ with radius 0.1. (Image provided by D. L. Fleitas [8])

In [17], J. Liu proved that the map ϕ_1 constructed in this way satisfies the equality (3.1) by showing that the equality given as

$$\frac{\partial H(t,\mathbf{x})}{\partial t} = 0, \quad \text{where} \quad H(t,\mathbf{x}) = J(\phi_t(\mathbf{x}))[t + (1 - tf(\phi_t(\mathbf{x}))]$$

holds for each $t \in [0,1]$. Notice that the monitor function f which controls the movement of the grid nodes can be chosen any function with the properties described above. In realistic situations, however, f is subject to some additional constraints. The monitor function f 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 monitor function. Here we give some examples for possible relations between solutions u and the monitor function f:

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

where C is a normalizing factor that may depend on t. Another possibility is to consider relations of the form

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

or even simpler monitor function

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

where $u(t,\mathbf{x})$ is the solution of some PDEs. Interested reader can read [17] in conjunction with this paper to have more information about GDM and further applications of it. Figure 2 illustrates an application of the GDM to a circle centered at $(0.6, 0.6)$ with radius 0.1. Next we describe our method for orthogonal grid generation.

4 Method for Orthogonal Grid Generation

In this section we present a variational approach that generates nearly orthogonal grids with suitable parameter values. We optimize a continuous cost functional which consists of only area and orthogonality quantities. It does not require the use of any volume functional, which is one of the significant improvement of our method.

A continuous grid generation functional is a quantity of the form

$$J(\varphi) := \int_\Omega I(\varphi),$$

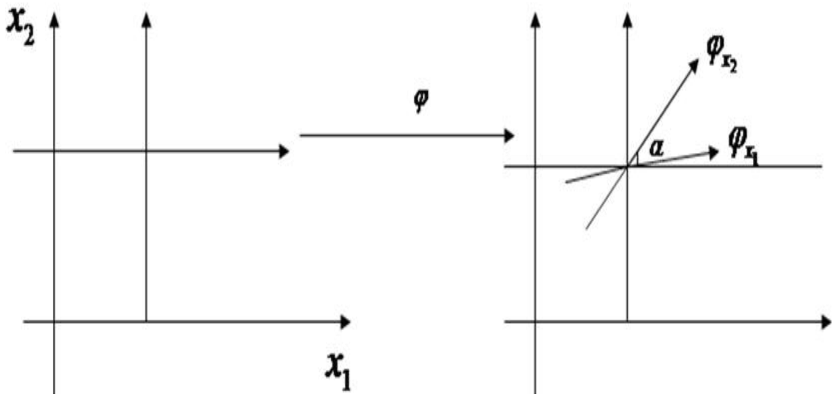

Figure 3: Rotations in grid nodes

where I is a function containing information about certain geometric quantities to be controlled in order to generate enough grids on Ω. The problem of continuous variational grid generation is to find φ^*, boundary conforming and minimizing the value of $J(\varphi)$. In other words, it is required to calculate φ^* such that

$$J(\varphi^*) = \min_\varphi J(\varphi).$$

From Variational Calculus it follows that φ^* must be a solution of a boundary value problem (BVP) [22], [23]. The resulting system from the solution of this BVP is a system of PDEs (mostly coupled and non-linear) with boundary conditions given by the value of the grid on the contour. Very often these conditions are called Euler-Lagrange equations. Perhaps the most widely used cost functionals are *area,* I_S, *volume,* I_V, and *orthogonality,* I_O. These functionals were typically combined ([1], [3]-[5], [21]) into one functional in different forms and in general the combined functional is given as

$$I = \lambda_S I_S + \lambda_V I_V + \lambda_O I_O,$$

where

$$\begin{aligned} I_S &= \frac{1}{2}\int_\Omega |\nabla\varphi|^2 \\ I_V &= \frac{1}{2}\int\int_\Omega \left(\frac{D(\varphi_1,\varphi_2)}{D(x_1,x_2)}\right) dx_1\, dx_2, \\ I_O &= \frac{1}{2}\int_\Omega \left(\varphi_{x_1}\cdot\varphi_{x_2}\right)^2, \end{aligned}$$

where λ_S, λ_V, λ_O are non-negative numbers, $\nabla\varphi$ is the gradient of $\varphi(\varphi_1,\varphi_2)$, $\frac{D(x,y)}{D(\xi,\phi)}$ is the Jacobian determinant. Detailed discussion on the applications of these variational functionals to the orthogonal grid generation methods can be seen in ([1], [3]-[5], [21]). Next we describe our method for the orthogonal grid generation.

Let Ω be a square domain in $\mathbf{R}^2$ and $\varphi = (\varphi_1, \varphi_2)$ be a grid function defined on Ω. We define the cost functional $J(\varphi)$ to be minimized as follows:

$$\begin{aligned} J(\varphi) &= \frac{1}{2}\int_\Omega (\varphi_{x_1}\cdot\varphi_{x_2})^2\, d\mathbf{x} + \frac{\lambda}{2}\int_\Omega |\nabla\varphi(\mathbf{x})|^2\, d\mathbf{x}, \\ &= \frac{1}{2}\int_\Omega (\varphi_{1x_1}\varphi_{1x_2} + \varphi_{2x_1}\varphi_{2x_2})^2\, d\mathbf{x} + \frac{\lambda}{2}\int_\Omega \left(\varphi^2_{1x_1} + \varphi^2_{1x_2} + \varphi^2_{2x_1} + \varphi^2_{2x_2}\right) d\mathbf{x} \end{aligned}$$

where $\lambda \in \mathbf{R}_+$, and φ_{x_i} denotes partial derivative of φ with respect to x_i for $i = 1, 2$, and

$$\varphi_{x_1}\cdot\varphi_{x_2}$$

denotes the inner product of φ_{x_1} and φ_{x_2}. Notice that

$$\varphi(\mathbf{x}) = \varphi_0(\mathbf{x}) + u(\mathbf{x}),$$

where $u(\mathbf{x})$ is the displacement which is added to the initial grid $\varphi_0(\mathbf{x})$ due to movements of grid nodes. Now our goal is to obtain $u(\mathbf{x})$ from the Euler-Lagrange equations.

Define a small variation of φ as:

$$\begin{aligned} \varphi_1 &\to \varphi_1 + \epsilon\,(\delta\varphi_1), \\ \varphi_2 &\to \varphi_2 + \epsilon\,(\delta\varphi_2). \end{aligned}$$

Taking the Fréchet derivative of the cost function $J(\varphi)$ with respect to φ_1 and φ_2, we obtain:

$$J_{\varphi_1} := \frac{\partial\, J(\varphi+\epsilon\delta\varphi)}{\partial\varphi_1} = \frac{d}{d\epsilon}\Big|_{\epsilon=0}\frac{1}{2}\int_\Omega [(\varphi_{1x_1}+\epsilon(\delta\varphi_1)_{x_1})(\varphi_{1x_2}+\epsilon(\delta\varphi_1)_{x_2})+\varphi_{2x_1}\varphi_2 x_2]^2+\frac{\lambda}{2}\int_\Omega ((\varphi_{1x_1}+\epsilon(\delta\varphi_1)_x$$

In the similar way, next we obtain J_{φ_2}.

$$J_{\varphi_2} := \frac{\partial\, J(\varphi+\epsilon\delta\varphi)}{\partial\varphi_2} = \frac{d}{d\epsilon}\Big|_{\epsilon=0}\frac{1}{2}\int_\Omega [(\varphi_{1x_1}\varphi_{1x_2}(\varphi_{2x_1}+\epsilon(\delta\varphi_2)_{x_1})+(\varphi_{2x_2}+\epsilon(\delta\varphi_2)_{x_2})]^2+\frac{\lambda}{2}\int_\Omega (\varphi^2_{1x_1}+\varphi^2_{1x_2})+($$

For the ease of notation, we define following variables:

$$A := \varphi^2_{1x_2}\varphi_{1x_1} + \varphi_{1x_2}\varphi_{2x_1}\varphi_{2x_2} \tag{4.2}$$

$$B := \varphi^2_{1x_1}\varphi_{1x_2} + \varphi_{1x_1}\varphi_{2x_1}\varphi_{2x_2} \tag{4.3}$$

$$C := \varphi^2_{2x_2}\varphi_{2x_1} + \varphi_{2x_2}\varphi_{1x_1}\varphi_{1x_2} \tag{4.4}$$

$$D := \varphi^2_{2x_1}\varphi_{2x_2} + \varphi_{2x_1}\varphi_{1x_1}\varphi_{1x_2} \tag{4.5}$$

$$\begin{aligned} \widetilde{A} &:= \varphi^2_{1x_2}\varphi_{1x_1} + \varphi_{1x_2}\varphi_{2x_1}\varphi_{2x_2} + \lambda\varphi_{1x_1} \\ \widetilde{B} &:= \varphi^2_{1x_1}\varphi_{1x_2} + \varphi_{1x_1}\varphi_{2x_1}\varphi_{2x_2} + \lambda\varphi_{1x_2} \\ \widetilde{C} &:= \varphi^2_{2x_2}\varphi_{2x_1} + \varphi_{2x_2}\varphi_{1x_1}\varphi_{1x_2} + \lambda\varphi_{2x_1} \\ \widetilde{D} &:= \varphi^2_{2x_1}\varphi_{2x_2} + \varphi_{2x_1}\varphi_{1x_1}\varphi_{1x_2} + \lambda\varphi_{2x_2} \end{aligned}$$

Now, letting $J_{\varphi_1} = 0$, we can write 4 as

$$\int_\Omega \left(\widetilde{A} \cdot \widetilde{B}\right) \cdot \left(\nabla \delta\varphi_1\right) = 0 \quad \text{for every } \delta\varphi_1,$$

and applying the divergence theorem and bearing in mind that $\delta\varphi_1 = 0$ on the boundary of Ω, $\partial\Omega$, we have

$$\int_\Omega \left(\nabla \cdot (\widetilde{A}, \widetilde{B})\right) \delta\varphi_1 = 0 \quad \text{for every } \delta\varphi_1,$$

which implies that

$$\nabla \cdot (\widetilde{A}, \widetilde{B}) \;=\; 0.$$

Applying the same argument to the equation (4) obtained from $J_{\varphi_2} = 0$, we have

$$\nabla \cdot (\widetilde{C}, \widetilde{D}) \;=\; 0.$$

Using the equations

$$\begin{aligned} \varphi(\mathbf{x}) &= \varphi_0(\mathbf{x}) + u(\mathbf{x}), \\ (\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x})) &= (\varphi_{01}(\mathbf{x}) + u_1(\mathbf{x}), \varphi_{02}(\mathbf{x}) + u_2(\mathbf{x})), \end{aligned}$$

(where $\varphi_0(\mathbf{x}) = (\varphi_{01}(\mathbf{x}), \varphi_{02}(\mathbf{x}))$ is the initial grid) and the equations (4.2) and (4.3), we have the first Euler-Lagrange equation, see, for instance, [7] and [19]:

$$\begin{aligned} \Delta\,\varphi_1(\mathbf{x}) &= \Delta\,\varphi_{01}(\mathbf{x}) \,+\, \Delta\,u_1(\mathbf{x}) \;=\; \frac{-div\,(A,B)}{\lambda}, \text{ from which we get:} \\ \Delta\,u_1(\mathbf{x}) &= \frac{-div\,(A,B)}{\lambda} - \Delta\,\varphi_{01}(\mathbf{x}), \\ &= \frac{-A_{x_1} - B_{x_2}}{\lambda} - \Delta\,\varphi_{01}(\mathbf{x}), \\ &= \frac{-(\varphi_{1x_2}^2\varphi_{1x_1} + \varphi_{1x_2}\varphi_{2x_1}\varphi_{2x_2})_{x_1}}{\lambda} - \frac{(\varphi_{1x_1}^2\varphi_{1x_2} + \varphi_{1x_1}\varphi_{2x_1}\varphi_{2x_2})_{x_2}}{\lambda} \\ &- \Delta\,\varphi_{01}(\mathbf{x}). \end{aligned}$$

By the equations (4.4) and (4.5) we have the second Euler-Lagrange equation:

$$\begin{aligned} \Delta\,\varphi_2(\mathbf{x}) &= \Delta\,\varphi_{02}(\mathbf{x}) \,+\, \Delta\,u_2(\mathbf{x}) \;=\; \frac{-div\,(C,D)}{\lambda}, \text{ from which we get:} \\ \Delta\,u_2(\mathbf{x}) &= \frac{-div\,(C,D)}{\lambda} - \Delta\,\varphi_{02}(\mathbf{x}), \\ &= \frac{-C_{x_1} - D_{x_2}}{\lambda} - \Delta\,\varphi_{02}(\mathbf{x}), \\ &= \frac{-(\varphi_{2x_2}^2\varphi_{2x_1} + \varphi_{2x_2}\varphi_{1x_1}\varphi_{1x_2})_{x_1}}{\lambda} - \frac{(\varphi_{2x_1}^2\varphi_{2x_2} + \varphi_{2x_1}\varphi_{1x_1}\varphi_{1x_2})_{x_2}}{\lambda} \\ &- \Delta\,\varphi_{02}(\mathbf{x}). \end{aligned}$$

We want to obtain $u(\mathbf{x}) = (u_1(\mathbf{x}), u_2(\mathbf{x}))$ by solving these Poisson equations. Although there are several sophisticated methods for solving these types of Poisson equations, we use finite-difference successive-over relaxation (SOR) method. Next we briefly overview the SOR method for the convenience of the reader and detailed treatment to this method can be seen, for example, in [8].

Successive Over Relaxation Method (SOR): SOR is a modified version of the Gauss-Seidel method for solving a linear system of equations, which results in faster convergence than Gauss-Seidel method.

We first illustrate the solution of 2D Poisson equations via finite-differences SOR method in detail. Let us consider a general Poisson equation given by

$$\Delta u = f,$$

where f and u stand for smooth functions defined on Ω. Using finite-differences method, we can express a 2D Poisson equation in a discrete form as

$$\tfrac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{\Delta x^2} + \tfrac{u_{i,j-1}-2u_{i,j}+u_{i,j+1}}{\Delta y^2} = f_{i,j}$$

Suppose that $\Delta x = \Delta y = h$, then, for all the interior points we have

$$u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 \times \, f_{i,j}\right).$$

The resulting system of linear algebraic equations is then solved by using SOR method in the following two steps:

$$\begin{aligned} \widetilde{u}_{i,j} &= \frac{1}{4}\left(u^{\text{new}}_{i-1,j} + u^{\text{old}}_{i+1,j} + u^{\text{new}}_{i,j-1} + u^{\text{old}}_{i,j+1} - h^2 \times \, f_{i,j}\right) \\ u^{\text{new}}_{i,j} &= (1-\rho)\, u^{\text{old}}_{i,j} + \rho\, \widetilde{u}_{i,j} \end{aligned}$$

It is possible to express this discretized equations as a system of $n \times n$ linear equations with unknown $\mathbf{x}$:

$$A\mathbf{x} = b, \tag{4.6}$$

Next decompose A into a diagonal component D, and strictly lower and upper triangular components L and U as $A = D+L+U$. Hence, the system of linear equations (4.6) can be expressed as

$$(D + \nu L)\mathbf{x} = \nu\, b - [\nu U + (\nu - 1)D]x \tag{4.7}$$

for some constant $\nu > 0$. SOR is an iterative method which solves the left hand side of (4.7) for $\mathbf{x}$, using forward substitution:

$$x_i^{(k+1)} = (1-\nu)x_i^{(k)} + \frac{\lambda}{a_{ii}}\left(b_i - \sum_{j>i} a_{ij}x_i^{(k)} - \sum_{j<i} a_{ij}x_j^{(k+1)}\right),$$

for $i = 1, \ldots, n$. The choice of relaxation factor, ν, depends upon the properties of the coefficient matrix. It can be proven that $0 < \nu < 2$ leads to convergence for symmetric, positive-definite matrices.

Table 1: **Case 1**

Parameters	**Value**	**Explanation**
λ	1	Control parameter in cost functional
μ	0.2	Quantity level for adding change u
ρ	0.001	Tolerance value for each iteration
Number of Iterations	200	Maximum number of iterations
ν	1.1	Relaxation factor for solving SOR
boundary type	1	1-fixed boundary; 2-mirror boundary
bdfactor	0	1-compute on boundary, 0-do not compute
tolr	0.001	Tolerance value in SOR method

5 Experimental Results

We use an initial grid with the size of 25×25 whose monitor function is defined as follows:

$$f(x,y) = \begin{cases} 0.1, & d(x,y) \leq 0; \\ 0.1 + 9 * d(x,y), & 0 \leq d(x,y) \leq 0.1; \\ 1, & 0.1 \leq d(x,y). \end{cases}$$

where

$$d(x,y) = \sqrt{(x-0.6)^2 + (y-0.5)^2} - (0.16)^2$$

Figure 3 represents the initial grid. Table 1 illustrates the use of some of the parameters used in SOR procedure. Explanations under the Figure from 4 to Figure 12 contain information for each case.

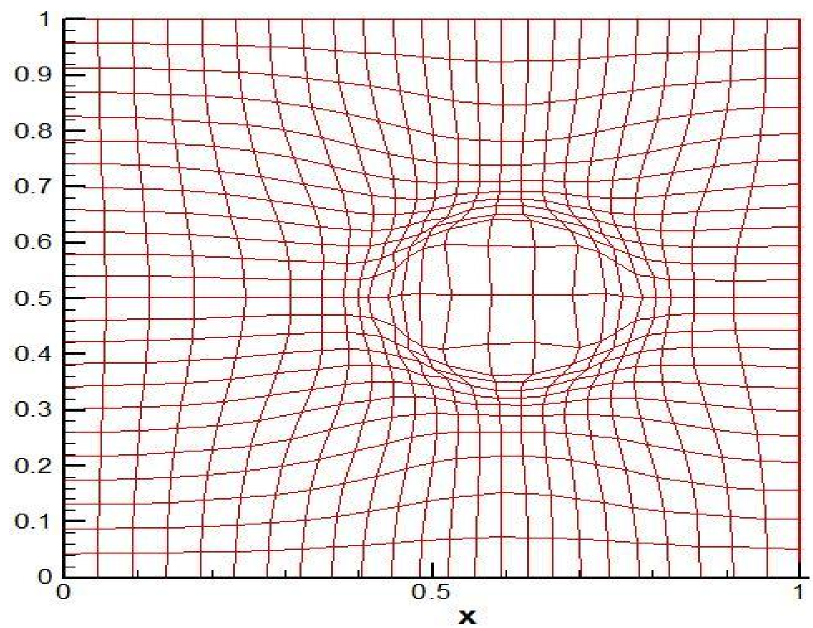

Figure 4: Initial Grid: φ_0

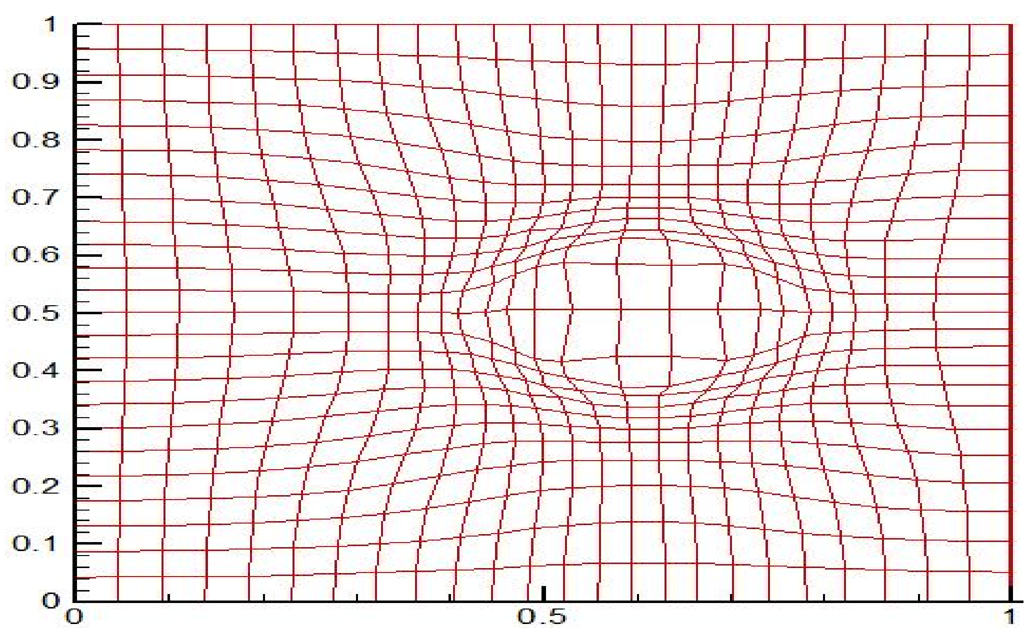

Figure 5: Case 1: $\lambda = 1,\ \mu = 0.2$

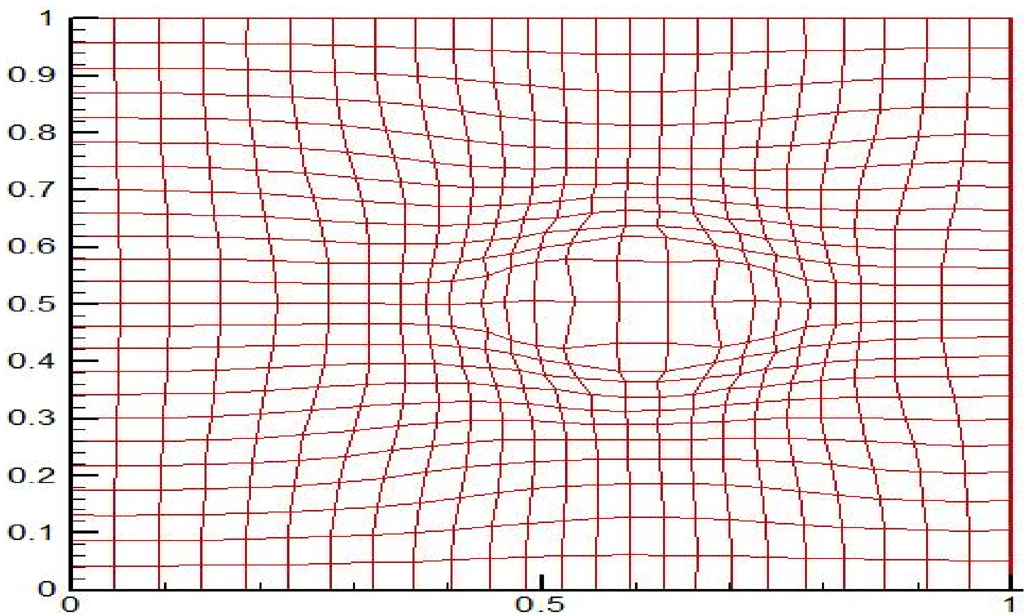

Figure 6: Case 2: $\lambda = 0.7,\ \mu = 0.4$

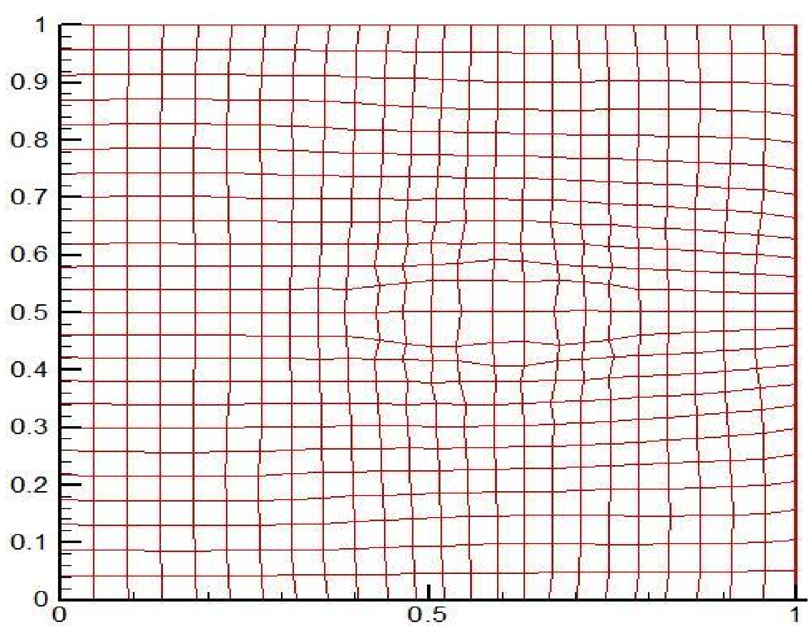

Figure 7: Case 3: $\lambda = 0.7,\ \mu = 0.9$

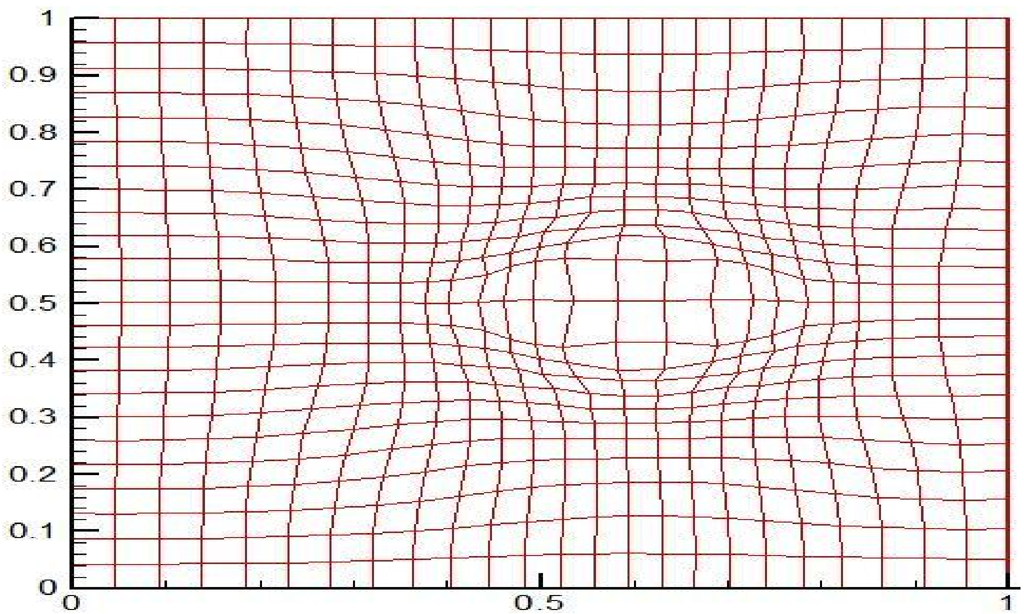

Figure 8: Case 4: $\lambda = 0.69,\ \mu = 0.4$

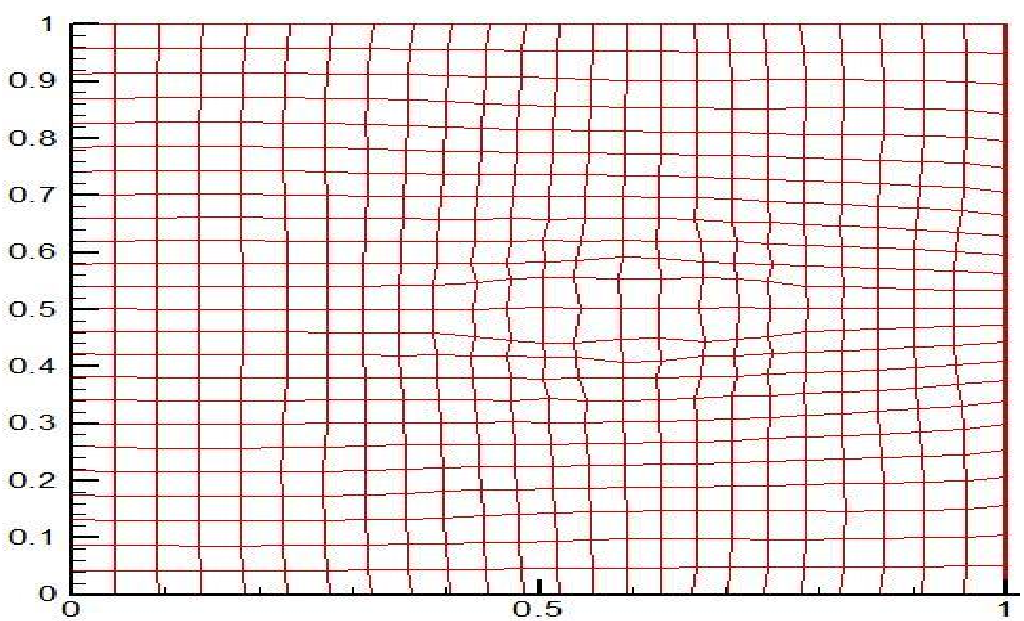

Figure 9: Case 5: $\lambda = 0.69,\ \mu = 0.9$

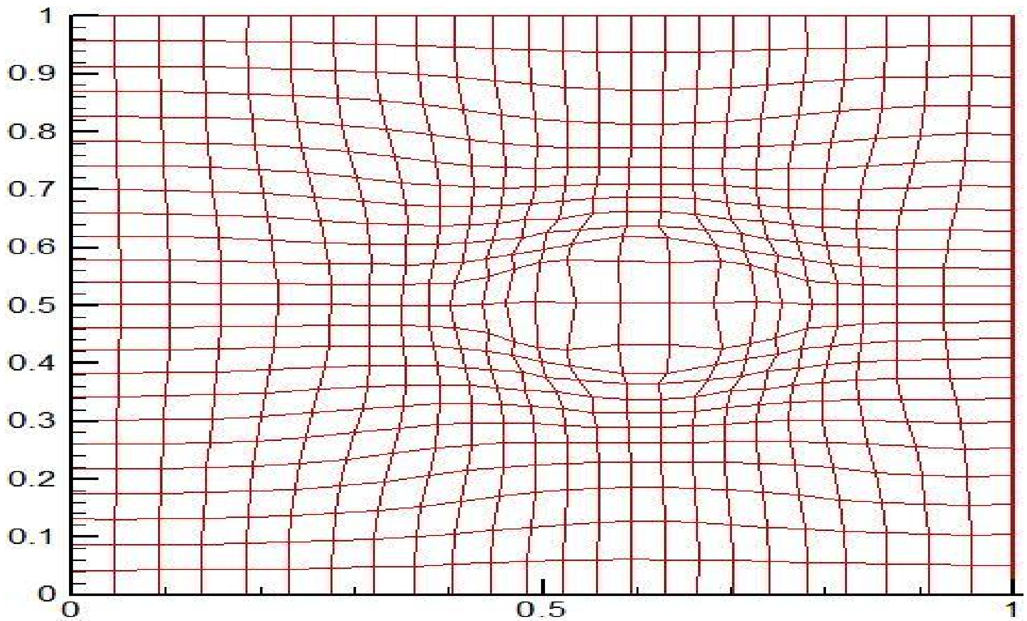

Figure 10: Case 6: $\lambda = 0.75,\ \mu = 0.4$

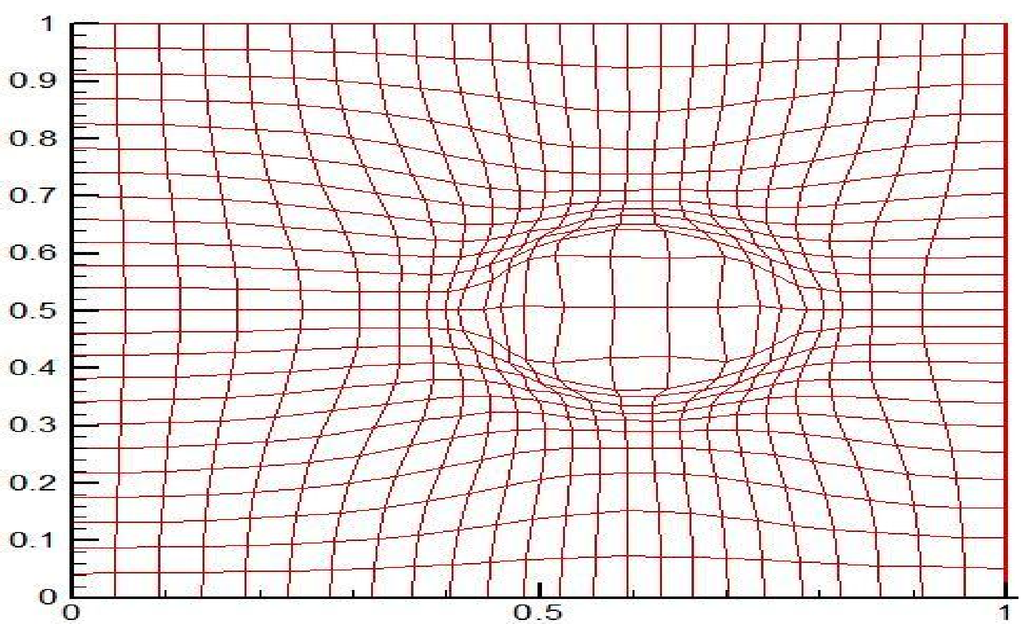

Figure 11: Case 7: $\lambda = 50,\ \mu = 0.2$

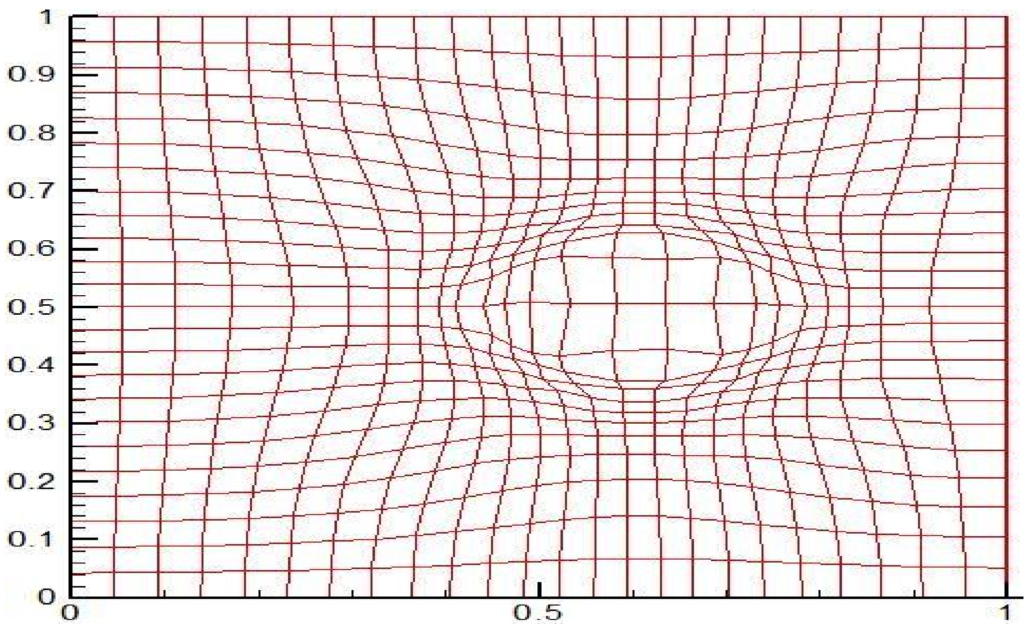

Figure 12: Case 8: $\lambda = 100,\ \mu = 0.2$

6 Discussion and Conclusion

In this paper we present a variational method which generates nearly orthogonal grids with suitable parameter values. We optimize a cost functional which consists of only area and orthogonality quantities. A significant contribution is that no volume functional needs to be added to the cost functional. We generate the initial non-orthogonal grid via deformation based grid generation method. We achieve to generate nearly orthogonal grids without changing the cell size distribution of the adaptive (initial) grids. Our computational results show that if the weighting parameters λ appearing in the smoothing regularizer in the cost functional is located between $0.6 - 1.0$ and the variation term μ of the initial grid is located between $0.1 - 1.0$, our method generates nearly orthogonal grids. We observe that the larger μ (close to 1.0) and the smaller λ (close to 0.6) increases the orthogonality of the grid. As we observe, larger λ such as $\lambda = 50, 100$ result in less orthogonal grids.

The algorithm was implemented in Fortran language on Intel Pentium D workstation running on the UNIX operating system. The computational examples given in the previous section were used to test the algorithm. Preliminary experiments show promising results and great potential for future extensions. Codes used in the implementation of our algorithm can be found in the webpage of the corresponding author [2].

References

[1] V. Akcelik, B. Jaramaz and O. Ghattas, Nearly orthogonal two-dimensional grid generation with aspect ratio control, *J. of Comp. Physics*, 171(2001), 805-821.

[2] M.A. Akinlar, http://myweb.sabanciuniv.edu/akinlar/

[3] J.U. Brackbill and J.S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J.Comp. Physics*, 46(1982), 342-368.

[4] J. Castillo, S. Steinberg and P.J. Roache, Parameter estimation in variational grid generation, *Appl. Math. and Comp.*, 28 (1988), 1-23.

[5] R. Courant, On a generalized form of Plateau's problem, *Trans. Amer. Math. Soc.*, 50(1941), 40-47.

[6] D.L. Fleitas, The least-squares finite-element for grid deformation and meshfree applications, *Ph.D thesis*, The University of Texas at Arlington, (2005).

[7] G. Guseinov and E. Ozyilmaz, Tangent lines of generalized regular curves parametrized by time scales, *Turkish J. of Math.*, 25(4) (2001), 553-562.

[8] A. Iserles, A first course in the numerical analysis of differential equations, *Cambridge Texts in Appl. Math.*, (2009).

[9] E. Lee and M. Gunzburger, An optimal control formulation of an image registration problem, *J. of Math. Imaging and Vision*, 36(1) (2008), 69-80.

[10] G. Liao, Variational approach to grid generation, *Num. methods for PDEs*, 8 (1992), 143-147.

[11] G. Liao and D. Anderson, A new approach to grid generation, *Applicable Anal.*, 44(3) (1992), 285-298.

[12] G. Liao, X. Cai, D. Fleitas, X. Luo, J. Wang and J. Xue, Optimal control approach to data set alignment, *Appl. Math. Letters*, 21(2008), 898-905.

[13] G. Liao, T. Pan and J. Su, A numerical grid generator based on Moser's deformation method, *Numerical Methods in PDE*, 10(1) (1994), 21-31.

[14] G. Liao and J. Su, A moving grid method for (1+1) dimension, *Appl. Math. Letters*, 8(4) (1995), 47-49.

[15] G. Liao and J. Su, A direct method in Dacorogna-Moser's approach of grid generation problems, *Appl. Anal.*, 49(1) (1993), 73-84.

[16] G. Liao and J. Su, Grid generation via deformation, *Appl. Math. Letters* 5(3) (1992), 27-29.

[17] J. Liu, New development of the deformation method, *Ph.D Thesis*, The University of Texas at Arlington, (2006).

[18] F. Liu, S. Ji and G. Liao, An adaptive grid method with cell-volume control and its applications to Euler flow calculations, *SIAM J. Sci. Comp.*, 20 (1998), 811825.

[19] A. Najati and F. Moradlou, Stability of an Euler-Lagrange type cubic functional equation, *Turkish J. Math.*, 33 (2009), 65-73.

[20] C. Shu and S. Osher, Efficient implementation of essentailly non-oscillatory shock capturing scheme, *J. Math. Physics*, 77(1988).

[21] S. Steinberg and P. Roache, Variational grid generation, *Num. Methods for PDEs.*, 2(1985), 71-96.

[22] J. Thompson, B. Soni and N. Weatherill, *Handbook of Grid Generation*, CRC Press, (1999).

[23] J. Tinoco-Ruiz, P. Barrera-Sanchez and A. Cortes-Medina, Some properties of area functionals in numerical grid generation, *X Meshing Round Table*, Newport Beach, California, USA, (2001).