UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Level Set Based Deformation Methods for Adaptive Grids

Guojun Liao
Feng Liu
Gary C. de la Pena
Danping Peng
Stanley Osher

January 1999

CAM Report 99-1

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

Level Set Based Deformation Methods for Adaptive Grids

Guojun Liao*
Department of Mathematics, University of Texas,
Arlington, Texas 76019
d127liao@utarlg.uta.edu

Feng Liu
Department of Mechanical and Aerospace Engineering, UC Irvine,
Irvine, California 92027
fliu@uci.eng.edu

Gary C. de la Pena†
Department of Mathematics, University of Texas,
Arlington, Texas 76019
gxd2639@utarlg.uta.edu

Danping Peng‡
Department of Mathematics, University of California at Los Angeles,
Los Angeles, California 90095-1555
dpeng@math.ucla.edu

Stanley Osher§
Department of Mathematics, University of California at Los Angeles,
Los Angeles, California 90095- 1555
sjo@math.ucla.edu

February 8, 1999

Subject Clasifications: 65M(N)05, 65M(N)50
Keywords: Adaptive grids, Deformation methods, Level Set Functions.

Abstract

Methods for generating adaptive grids are developed in this paper. Included is the static deformation method which generates a final grid by

*Supported by NSF under Grant DMS-9732742
†Supported by NSF under Grant DMS-9732742
‡Supported by NSF and DARPA under grants DMS-9706827 and DMS-9615854
§Supported by NSF and DARPA under grants DMS-9706827 and DMS-9615854

deforming an initial grid using artificial time. Supersonic flows through two wedges are calculated by using the deformation method. A time accurate deformation method is also developed as a significant advancement of this method. The deformation process takes place in real time as the physical variables evolve according to the host partial differential equation. Level set functions can be used to form the monitor function. These deformation methods are based on a deformation scheme that originated in differential geometry, we refer to this as the traditional deformation method.

A new deformation method for generating adaptive moving grids is formulated based on physical quantities. Level set functions are used to construct the adaptive grids. They are solutions of the standard level set evolution equation with the Cartesian coordinates as initial values. The intersection points of the level sets of the evolving functions form a new grid at each time. The velocity vector in the evolution equation is chosen according to a monitor function. A uniform grid is then deformed to a moving grid with desired cell volume distribution at each time.

The above methods achieve precise control over the Jacobian determinant of the grid mapping. The new method has all the advantages of the level set approach.

1 Introduction

A key problem in numerical simulation of time dependent partial differential equations is grid generation and grid adaptation. The problem this paper deals with is how to greatly enhance accuracy and efficiency by using adaptive moving grids.

The tasks of simulating transient problems on three dimensional domains become enormously difficult when tens of millions of nodes are needed. When a fixed grid is used the grid points are distributed in the physical domain prior to the calculation of the solution. A drawback occurs when the solution to the PDE exhibits large variations due to, for example, shock waves, moving fronts, and boundary layers. Because of its static feature the grid is unable to efficiently and accurately resolve such variations. This is especially so in transient problems where the solution may not only show large variations in some regions, but may also move rapidly with time. Thus the computed solution may fail to resolve the fine structures of the solution. For instance, to correctly simulate the dendritic growth of crystal line modeled by the Stefan problems, one must use fine grids near the interface between the solid phase and the liquid phase. Figure 8 of [24] is used here to show the importance of grid sizes. As can be seen in Figure 1, Using a state-of-the-art level set method, the 100×100, 200×200, and 300×300 fixed uniform grids on the unit square give rise to unsatisfactory results of the interface. It takes the 400×400 fixed grid to produce a sharp result. A 3D simulation would need 64 million nodes. This would be too costly, even for this method.

We can improve the accuracy and efficiency by using adaptive grids. The idea is to generate the grids according to the salient features of the solution,

so that the nodes will be concentrated in regions where the solution changes rapidly in order to improve accuracy, and fewer grid points are used in regions where negligible changes in the solution occur.

There are two types of grids: structured and unstructured. One could also use a hybrid grid. for instance, a structured grid could be used near the boundary for resolving the boundary layers and unstructured grid elsewhere for handling complex topology.

There are two strategies for grid adaptation:

1. Local refinement: Nodes are inserted where and when they are needed. It is flexible and easy to conform to the boundary. But the solver and data structure have to be modified after insertion and/or deletion of nodes. The computational overhead is usually considerably higher and the accuracy is unsatisfactory in some applications.

2. Moving grid: The total number of nodes and the connectivity between them are fixed. The nodes are redistributed where and when they are needed.

For general grid generation methods we refer to the books by Thompson et.al. [2], Zegeling [4], Knupp and Steinberg [5], Carey [6].

At present, adaptive grid methods are mainly used with finite volume and finite element schemes. This is because these schemes allow some flexibility on grids, and in particular, can be used on non-othogonal grids. Deviation from orthogonality affects the performance of these schemes as well. It is well known that the increase in skewness of a grid may decrease accuracy. There are applications in sciences and engineering where finite difference methods are commonly used. The orthogonality property of the grid is often required for these methods. Recently, S. Steinberg et al. developed a method to calculate derivatives on arbitrary grids with a criteria for which points are neighbors [32, 33]. The traditional deformation method described in section 1 of this paper accurately determines the node velocity and thus the time-dependent differential equations can be transformed by the nodal mapping into Cartesian coordinates on a logical domain. The transformed equation will then be simulated on a fixed orthogonal grid. This approach can be used in combination with various numerical schemes and with structured or unstructured grids. It is a moving grid method which generates grids that move in real time, adapting to the features of an unsteady solution. It is particularly suitable for use with structured grids, but it could be used with unstructured grids as well.

Suppose that we want to simulate a scalar or vector field $u(\mathbf{x}, t)$ satisfying

$$u_t(\mathbf{x}, t) = L(u) \tag{1}$$

here L is a differential operator defined on a physical domain $\Omega = D_2$ in R^n, n =1,2,3. A common idea is to construct a transformation $\phi : D_1 \times [0, T] \rightarrow D_2$ which moves a fixed number of grid points on Ω to adapt to the numerical solution as it is being computed on the logical domain D_1. To be qualified as a transformation, ϕ must be one to one and onto. Variational methods (cf. [1], [2])

and elliptic PDE methods (cf. [2]) define this transformation as the solution of a system of PDEs which is created to control various aspects of the grid such as orthogonality ("skewness"), smoothness, and cell size. The resulting system of PDEs for grid generation is often nonlinear and requires intensive computation. Significant contributions were made to dynamically adapt the grid by controlling the cell size through the Jacobian determinant of the transformation in [1, 2, 3, 7]. The moving finite element method was developed in [8] and is useful for certain unsteady problems. Recently, moving mesh methods based on Moving Mesh Partial Differential Equations [11] were developed with remarkable capability to track rapid spatial and temporal transitions for some model problems. Hybrid techniques that use both grid motion and local refinement showed their effectiveness for 2D problems ([9]).

Whereas current moving grid methods have associated stregths and weaknesses, they do not provide mathematical assurance that the "grid transformation ϕ" is indeed a transformation. The methods have difficulties in controlling the grid movements and in preventing singularities.

In this paper we will review the traditional form of the deformation method which is based on a result from differential geometry. Then we will formulate a new deformation method which is based on the level set approach and a standard formula from fluid dynamics. Both methods move the nodes with a proper velocity field so that the nodal mapping have the desired Jacobian determinant and thus precisely control the cell size distribution according to a positive monitor function. In both cases the velocity vector field is constructed by solving a Poisson equation according to the monitor function. The main difference between the two methods is that the former uses a system of ODEs to move the nodes directly by the vector field while the latter uses a system of PDEs to get the level set functions whose level sets generate the moving grid. The ODEs are, of course, the characteristic equations for the level set based PDEs.

2 The Traditional Deformation Method

The traditional deformation method is based on the deformation scheme of Jurgen Moser and B. Dacorogna [12], [13] in the study of volume elements and has recently been applied to numerical methodology [14]. The method provides direct control over the cell size of the adaptive grid and the node velocities are directly determined. The location of nodes can also be easily computed. The method inherently defines a transformation which is necessarily injective, thus ensuring that the transformed gridlines do not cross even in higher dimensions. The static version of the deformation method was used with a finite volume solver in flow calculation problems [18] (see Example 1). A one spatial dimension version of the deformation method was used with a discontinuous Galerkin finite element method in numerically solving a convection-diffusion problem [16].

Recall that the Jacobian determinant of a mapping $\phi(\xi, t)$ from D_1 to D_2 in R^n, n =1,2,3, is $J(\phi) = \det \nabla \phi = |dA'|/\ |dA|$,where dA' is the image of a

volume (area, in 2D) element dA. The deformation method constructs φ such that $J(\phi) = f(\phi,t)$, and thus, it assures precise control over the cell size relative to the fixed initial grid in any dimensions. Suppose that the solution to (1)has been computed at time step $t = t_{k-1}$, and a preliminary computation has been done at time level $t = t_k$. Assume that we are provided with some positive error estimator $\delta(\xi, t)$ at the time steps t_{k-1} and t_k.Define a monitor function,

$$f(\xi, t) = C_1/\delta(\xi, t), \tag{2}$$

where C_1 is a positive scaling parameter so that at each time step we have

$$\int_\Omega \left(\frac{1}{f(\xi, t_k)} - 1\right) dA = 0. \tag{3}$$

Note that f is small in regions where large error occurs and becomes larger in regions where the error is small. We then seek a transformation $\phi : D_1 \to D_2 = \Omega$ such that

$$\begin{aligned} \det \nabla\phi(\xi, t) &= f(\phi(\xi, t), t) \qquad t_{k-1} \le t \le t_k, \\ \phi(\xi, t_{k-1}) &= \phi_{k-1}(\xi) \qquad \xi \text{ on } D_1 \end{aligned} \tag{4}$$

where ξ is a grid node of an initial grid, $\phi_{k-1}(\xi)$ represents the coordinates of the node at $t = t_{k-1}$((3) is necessary for (4) to be true). We specify that $\phi(\xi) \in \partial\Omega$ for all $\xi \in \partial D_1$. Note that (4) ensures the size of the transformed cells will be proportional to f, i.e. the grid will be appropriately "condensed" in regions of high error and "stretched" in regions of small error. It is well known that if the Jacobian determinant of a transformation ϕ is positive in D_1, then ϕ is one-to-one in all of D_1. This ensures that the grid will not fold onto itself.

Solving (4) would appear to be a difficult problem in itself, raising questions as to the practicality of the approach. However recent advances [15, 16, 17, 21, 22] have resulted in a method which produces the solution easily. The underlying numerical method is finite element in [16], finite volume in [18], and finite difference in [22], respectively. The computation of ϕ essentially involves two steps. The first step is to find a vector field $\mathbf{v}(\xi, t)$ satisfying

$$\begin{aligned} \operatorname{div} \mathbf{v}(\xi, t) &= -\frac{\partial}{\partial t}\left(\frac{1}{f(\xi, t)}\right), \qquad \xi \in D_1,\ t_{k-1} \le t \le t_k \\ < \mathbf{v}, \mathbf{n} > &= 0 \quad \xi \in \partial D_1 \quad \mathbf{n} = \text{outward normal to } \partial D_1. \end{aligned} \tag{5}$$

The vector field $\mathbf{v}$ can be found by solving for w in the scalar Poisson equation (for a fixed t)

$$\triangle w(\xi, t) = -\frac{\partial}{\partial t}\left(\frac{1}{f(\xi, t)}\right), \quad \xi \in D_1 \ \frac{\partial w}{\partial \mathbf{n}} = 0,\ \xi \in \partial D_1 \tag{6}$$

then setting $\mathbf{v} = \nabla w$. Note: The constraint for the Neumann boundary condition is

$$\int \frac{\partial}{\partial t}\left(\frac{1}{f(\xi, t)}\right) = 0,$$

which is satisfied since $\int \frac{1}{f(\xi,t)}$ =area(D_1) =constant by (3).

The second step is to solve for $\phi(\xi,t)$ from the ODE system (referred to as the deformation ODEs)

$$\frac{d}{dt}\phi(\xi,t) = \eta(\phi(\xi,t),t) \qquad t_{k-1} \le t \le t_k,\ \xi \in D_1 \tag{7}$$

$$\phi(\xi,t_{k-1}) = \phi_{k-1}(\xi),$$

where the node velocity $\eta(\xi,t) = f(\xi,t)\mathbf{v}(\xi,t)$. A mathematical foundation of the method is provided by the following **Theorem** ([16], [21], [22]]): det $\nabla\phi(\xi,t) = f(\phi(\xi,t),t)$ for each ξ in $\overline{D_1}$ each $t > 0$. The theorem is proved by showing $\frac{d}{dt}(J(\phi)/f(\phi,t)) = 0$ and therefore $J/f = 1$ if $(J/f)\Big|_0 = 1$.

According to this theorem, $J(\phi) = f > 0$. Consequently, the mapping φ is injective (non-folding). Also by choosing f to be continuous (or smooth), we can make the Jacobian determinant change continuously (or smoothly), which is crucial in obtaining high accuracy in the computation.

A shortcoming of the deformation method is the lack of direct control over the orthogonality of the grid lines of the physical domains. The fact that the vector field $\mathbf{v}$ used in the deformation equation is irrotational, i.e. $\mathbf{v} = \nabla w$, and thus curl $\mathbf{v} = 0$, helps to prevent excessive skewness in the grid. This is why the method works well with finite volume algorithms in flow calculations.

For finite difference algorithms, we will transform equation (1) by $\mathbf{x} = \phi(\xi,t)$ and solve the transformed equation on a fixed orthogonal grid on the ξ-domain (the logical domain). This approach will enable us to have the benefits of the adaptive grids as well as the advantages of using a fixed orthogonal grid. This idea will be demonstrated by some model problems in [22].

We begin with an initial grid on the physical domain D_2, which corresponds to an orthogonal grid on a square (or a cubic) domain D_1. Then we deform the initial grid on the physical domain in real time according to a monitor function $f(\xi,t)$. Various equidistribution principles can be used to construct the monitor function. A posteriori error estimates (if available), residuals, and truncation errors, etc. are redistributed evenly over the whole domain. In most cases, we want to put refined grids in the regions where u changes rapidly. For instance, if the flow patterns exhibit shock waves, we can take (for Euler flows)

$$f = C_1/(1 + C_2|\nabla p|^2), \tag{8}$$

where p is the pressure, C_2 is a constant for adaptation intensity , C_1 is a normalization parameter. For viscous flow, we can use the Mach number in place of the pressure. In general, in addition to the gradient of the unknown variable u, terms involving the value of u and the second derivatives of u (or the curvature of its level sets) can also be included. For instance,

$$f = C_1/(1 + \alpha\,|u|^2 + \beta|\nabla u|^2 + \gamma\,|\nabla^2 u|^2). \tag{9}$$

For interface resolution, we can, for instance, construct f by using the signed distance function d as follows: Let f be piecewise linear such that

$$f = \begin{cases} 1 & \text{if } |d| > 0.1 \\ 0.2 & \text{if } d = 0. \end{cases} \tag{10}$$

Normalize f so that $\int_D (\frac{1}{f} - 1) = 0$, which is required if (4) is to be satisfied. The constants 0.1 and 0.2 in (10) can be changed according to the desired intensity of adaptation at the interface. The signed distance function can easily be computed as was done in [30].

2.1 Numerical Implementation

The moving grid method has been implemented for some model problems on 2D and 3D domains. The Neumann boundary condition for the Poisson's equation is implemented by introducing fictitious points outside the domain. A successive overrelaxation method (SOR) is then used to solve the resulting system of linear equations. The initial values were the results at the previous time step. The vector field $\mathbf{v}$ is then obtained by using second-order central-difference approximations to solve for $\mathbf{v} = \nabla w$. The coordinates of the new grid are then computed by solving (7) using a fouth-order Runge-Kutta method.

Example 1: Steady transonic and supersonic Euler flows around an airfoil [18].

Example 2: Calculation of $2D$ supersonic flow through wedges.

We implemented the deformation method for calculation of a supersonic flow field through two wedges as the first step toward simulation of flows through turbo machine blades. The geometry is shown in Figure $2a$, where a uniform grid is put in the region between two wedges. A finite volume flow solver as that in [19] is used on the uniform grid to simulate the flow field. In Figure $2b$ the computed pressure contour map is plotted over the uniform grid. It exhibits multiple directional shock pattern and the resolution is not satisfactory. This is because the grid is uniform and is constructed before the flow calculation starts.

Based on the computed data for the Mach number M, a monitor function f is constructed by the following formula:

$$f(x, y) = \frac{C_1}{1 + C_2 |\nabla M|^2},$$

where C_2 is an intensity constant and C_1 is a normalization factor. The monitor function indicates the magnitude of flow speed gradient: it is small when the speed varies rapidly, indicating the need of refined grid. A new grid is then calculated by using the deformation method which involves two steps: (1) Solving a Poisson equation, and, (2) Moving each node of the uniform grid by two deformation ordinary differential equations. The new grid is shown in Figure $2c$. The grid is clustered around the shock lines. Indeed the deformation method guarantees that the Jacobian determinant J of the transformation which takes the

initial uniform grid to the new grid be equal to the weight function f :

$$J(x, y) = f(x_{new}, y_{new})$$

for each node (x, y). Thus the cell size of the new grid is proportional to the weight function f. Finally, the flow is recalculated on the new grid and the resulting contour map is shown in Figure $2d$. This shows a sharp resolution of the shock pattern.

The above examples show that the deformation method is promising for solving realistic problems.

Example 3: A uniform grid is deformed into a grid concentrated around a pair of circles and the grid moves properly as the circles merge into each other (see Figure 3). Let d be defined by

$$d = ((x-a)^2 + (y-b)^2 - r^2)\,((x-c)^2 + (y-d)^2 - \rho^2), \tag{11}$$

here (a, b) is the (moving) center of the first circle, (c, d) is the (moving) center of the second circle, r and ρ are varying radii of the circles. Note: The two moving circles are the zero level set of d. But d is not the signed distance from the two circles. We use an initialization procedure developed in [23] to modify d so that it becomes the signed distance near the two circles. Then, the monitor function is defined by

$$f = \begin{cases} 0.2 - 8d & \text{if } -0.1 < d < 0; \\ 0.2 + 8d & \text{if } 0 < d < 0.1; \\ 1 & \text{if } |d| > 0.1. \end{cases} \tag{12}$$

In the example the circles initially intersect each other and then gradually merge to one expanding circle.

Example 4: An initial uniform grid is deformed to a grid clustered around the interface of ice and water during the solidification process modeled by the Stefan equations (See Figure 4). The monitor function f is defined as in Example 3 with d that is proportional to the level set function ϕ calculated by a level set method (cf. [24]).

3 A level Set Deformation Method

In this section, a new deformation method is formulated. The usual evolution equation for level set functions (as in [25]) with the Cartesian coordinates as initial values is solved. The velocity vector in the evolution equation can be chosen correctly according to a monitor function. The intersection points of the level sets of the evolving solutions will form a new grid at each time. Numerical examples will be provided in which a uniform grid is deformed to moving grids with prescribed cell size distribution at each time.

We first set up the principle of redistribution in the one-dimensional case before describe the method in multiple dimensions.

Suppose that we want to construct a grid of $N+1$ nodes on $[0,1]$ at each time t according to a desired distribution given by a positive monitor function $f(x,t)$:

$$x_0(t) = 0 < x_1(t) < x_2(t) < ... < x_i(t) < x_{i+1}(t) < ... < x_N(t) = 1,$$

with the length $x_{i+1} - x_i = f(x_i',t)/N$, where x_i' is the midpoint of the subinterval $[x_i, x_{i+1}]$. We seek a transformation ϕ from $[0,1]$ to $[0,1]$, which sends x_i to $k_i = \phi(x_i)$, where points $k_i = i/N$, $i = 0, 1, 2, ..., N$, form a uniform grid on $[0,1]$. The condition $x_{i+1} - x_i = f(x_i',t)/N$ is equivalent to

$$(1/N)/(x_{i+1} - x_i) = 1/f(x_i', t)$$

whose left hand side tends to the Jacobian determinant $\partial\phi/\partial x$ as $N \to \infty$. At the limit, we get the condition for ϕ that

$$\partial\phi/\partial x = 1/f(x,t) = g(x,t) \text{ (Denoting} g = 1/f). \tag{13}$$

In 1D, this equation can be solved by direct integration:

$$\phi(x) = \int_0^x (1/f(x,t))dx,$$

where f is normalized to satisfy the condition $\phi(1) = \int_0^1 (1/f(x,t))dx = 1$ for each t. The preimages of the evenly placed points $k_i = i/N$ under the transformation $x \to \phi(x,t)$ are the level sets of ϕ and they form the new nodes.

This idea of using level set of a mapping to define grid nodes can be extended to multi-dimensions. To begin, let us recall the basic concept of modeling a moving front by level sets. Suppose that there is a moving front in a fluid flow with a velocity field $\mathbf{v} = (x_t, y_t, z_t)$, where $\mathbf{x} = (x,y,z)$ is the position of a (fluid) particle at time t. We introduce a smooth function $\phi(x,y,z,t)$ with the property that the front is given by the zero level set of ϕ, i.e. $\phi(x,y,z,t) = 0$ for every t at the front. Differentiate the identity with respect to t, we get $\phi_t + \phi_x x_t + \phi_y y_t + \phi_z z_t = 0$, which can be written as

$$\phi_t(x,y,t) + < \nabla\phi, \mathbf{v} >= 0. \tag{14}$$

This is the evolution equation for the level set function ϕ. In the calculations of fluid flows, $\mathbf{v}$ is the actual velocity of the fluid. Other important geometric parameters such as the curvature of the front, the normal vector to the front, etc., can easily be determined by ϕ. See [31] for a thorough overview.

Our purpose is to generate an adaptive grid according to a positive monitor function f as in (13). In two dimensions, we construct two functions ϕ and ψ by (14) with a properly chosen $\mathbf{v}$. Then the intersections of their level set curves will be the new nodes. Thus, let $\phi(x,y,t)$ and $\psi(x,y,t)$ be solutions to the evolution PDE

$$\begin{cases} \phi_t(x,y,t) + < \nabla\phi, \mathbf{v} >= 0 \\ \psi_t(x,y,t) + < \nabla\psi, \mathbf{v} >= 0. \end{cases} \tag{15}$$

The initial conditions are $\phi(x,y,0) = x$, $\psi(x,y,0) = y$, respectively. The boundary conditions are

$$\phi(0,y,t) = 0, \phi(1,y,t) = 1,\ \phi(x,0,t) = \phi(x,1,0) = x;$$

$$\psi(x,0,t) = 0, \psi(x,1,t) = 1,\ \psi(0,y,t) = \psi(1,y,t) = y.$$

The vector field $\mathbf{v}$ is chosen so that the Jacobian determinant $J(\phi,\psi)$ is equal to the reciprocal of a positive monitor function f, namely

$$J(\phi,\psi) = D(\phi,\psi)/D(x,y) = 1/f(x,y,t). \tag{16}$$

Note that this condition is the natural extension of the 1D condition (13).

In three dimensions, we solve for three functions ϕ_1, ϕ_2, ϕ_3 from the equations

$$(\phi_i)_t + < \nabla\phi_i, \mathbf{v} >= 0,\ i = 1,2,3, \tag{17}$$

with the same type of initial and boundary conditions as in 2D.

Let $\phi = (\phi_1, \phi_2, \phi_3)$. A suitable vector field $\mathbf{v}$ can be determined so that the Jacobian determinant of the mapping ϕ is equal to the reciprocal of a monitor function f, namely

$$J(\phi) = D(\phi_1, \phi_2, \phi_3)/D(x,y,z) = 1/f(x,y,z,t). \tag{18}$$

The intersections of their level sets form the nodes of the moving grid.

The key to the success of the proposed method is to determine the velocity vector field $\mathbf{v}$ so that $J = g$ at every t. Thus, the grid cell size can be precisely controlled, resulting in a moving grid that is adaptive according to the monitor function f. We propose that the velocity vector $\mathbf{v}$ be determined by the condition

$$g_t + div\,(g\mathbf{v}) = 0. \tag{19}$$

This choice is based on the transport formula in fluid dynamics, which can be found in any standard textbook on fluid dynamics. See for instance, [27], or [28]).

Let Ω_t be the image of an initial region Ω_0 under the flow of the velocity field $\mathbf{v} = (x_t, y_t, z_t)$, where $\mathbf{x} = (x,y,z)$ is the position of a (fluid) particle at time t.

Theorem: (Transport Formula) For any function $h(\mathbf{x},t)$, we have

$$\frac{d}{dt}\int_{\Omega_t} h\,dV = \int_{\Omega_t} (\frac{\partial h}{\partial t} + div(h\mathbf{v}))\,dV. \tag{20}$$

Let $J = D(\phi_1,\phi_2,\phi_3)/D(x,y,z)$ be the Jacobian determinant of the transformation $(x,y,z) \to (\phi_1,\phi_2,\phi_3)$. Taking $h = g(\mathbf{x},t) = 1/f$ in (18), we get, by a change of variables, that

$$\frac{d}{dt}\int_{\Omega_t} g\,dV = \frac{d}{dt}\int_{\Omega_0} g\,J^{-1}\,dV = \int_{\Omega_t} (\frac{\partial g}{\partial t} + div(g\mathbf{v}))\,dV = 0, \tag{21}$$

where (19) is used to get the last equation. In the change of variables, $D(x,y,z)/D(\phi_1,\phi_2,\phi_3)=J^{-1}$ is used. (21) implies that gJ^{-1}=constant since $\Omega_0$is arbitrary. Choose $(gJ^{-1})|_{t=0}=1$, then we get $gJ^{-1}=1$ for any $t>0$ as desired.

To solve for $\mathbf{v}$ from (19), we first observe that in one dimension condition (19) becomes

$$\partial(gv)/\partial x = -g_t,$$

and we can solve for v by a direct integration and get

$$v(x,t) = -\int_0^x g_t/g.$$

This suggests a simple method for determining the velocity vector field $\mathbf{v}$ in multi-dimensions. Let $f(x,y,z,t)>0$ be the desired grid cell size distribution, which is constructed according to the physical variables being simulated and is normalized as in (3). Let $g=1/f$. We first solve for a real valued function (potential) w from the Poisson equation on the x,y,z, domain:

$$\triangle w = -g_t. \tag{22}$$

with the Neumann boundary condition. Then, we set

$$\mathbf{v} = \nabla w/g. \tag{23}$$

It follows that $div\,(g\mathbf{v}) = div(\nabla w) = \triangle w = -g_t$ as desired. The method is based on solving a scalar Poisson's equation, and thus it works on general three dimensional domains.

3.1 Numerical Implementation

In the examples below, the Poisson equation is solved by an SOR scheme as before. The evolution equations are solved with a second order ENO scheme (as in [29]). A bilinear interpolation is then performed to get the nodes.

Example 5: (Figure 5) Moving grids on [0,1]

Let $f=1/g$ where $g=1+10t(x^2-x+1/6)$. By a direct integration, we can verify that

$$\int_0^1 g\,dx = [x+10t(x^3/3-x^2/2+x/6]_{x=0}^{x=1} = 1, \text{ for every } t.$$

Thus the normalization condition (3), is analytically satisfied. We want to generate a moving grid which is a uniform grid on $[0,1]$ at $t=0$. In 1D the velocity field is a real valued function. Solving for v from condition (19)

$$div\,(gv) = (gv)_x = -g_t,$$

we get, by a direct integration,

$$v = 10(-x^3/3+x^2-x/6)/g.$$

Next, solve for $\phi : [0,1]\times[0,T] \rightarrow [0,1]$ from the evolution equation (14.1):

$$\phi_t(x,t) + < \phi_x, v >= 0,$$

with the initial and boundary conditions $\phi(x,0) = x$, $\phi(0,t) = 0$, $\phi(1,t) = 1$.

Let $1/N$ be the spacing of a uniform grid on $[0,1]$. The preimages of the nodes of the uniform grid on $[0,1]$ form the moving grid at selected time t. In figure 5, f and ϕ are plotted along with the nodes of the moving grid with $N = 60$. Note that the grid spacing near $x = 0$ and $x = 1$ is getting smaller and it is getting larger near $x = 0.5$. In fact, this is the intended distribution since $d\phi/dx = g = 1/f$, which means $\triangle x_i = f/N$.

Example 6: (Figure 6)

Example 3 of Section 2 is reproduced by the level set method. The vector field $\mathbf{v}$ is constructed by the method described in formulas (22) and (23). Then equations (14.2) are solved by a second order ENO scheme as in example 5.

Example 7:(Figure 7)

Example 4 of Section 2 is reproduced by the level set method.

4 Conclusion

The traditional deformation methods are reviewed and applied to flow field calculations. A new deformation method for moving grid is formulated. The standard evolution equation for level set functions is used to construct the grid mapping. It is shown that a suitable velocity vector field can be constructed from a positive monitor function by solving a scalar Poisson equation. The resulting moving grid has the desired cell size distribution at each time. Numerical examples are given to demonstrate the method.

5 Acknowledgment

The first author is grateful to the department of mathematics of UCLA for the hospitality he received during his visit in the fall of 1997.

References

[1] J. U. Brackbill and J. S. Saltzman, "Adaptive Zoning for Singular Problems in Two Dimensions", *J. Comp. Phys.*, 46, (1982).

[2] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin, Numerical Grid Generation, (1985).

[3] J. Castillo, S. Steinberg, and P. J. Roache,"Mathematical Aspects of Variational Grid Generation", *J. Comput. Appl. Math.*, 20, (1987).

[4] Paul A. Zegeling, Moving Grid Methods, (The Utrecht University Press, 1992).

[5] P. Knupp and S. Steinberg, The Fundamentals of Grid Generation, (CRC Press, 1994).

[6] G. Carey, Computational Grid Generation,(Taylor & Francis 1997).

[7] D. A. Anderson, "Grid Cell Volume Control with an Adaptive Grid Generator", *Applied Math. and Computation*, 35, (1990).

[8] K. Miller, "Recent Results on Finite Element Methods with Moving Nodes", in Accuracy Estimates and Adaptive Methods in Finite Element Computations, Babuska, Zienkiewicz, Gago, and Oliveira,eds.,(John Wiley & Sons , 1986).

[9] D. Arney and J. Flaherty,*ACM Transactions on Math software*, 16, 1, (1990), pp. 48-71.

[10] A. Hawken et.al., "Review of Some Adaptive Node Movement Techniques in Finite Element and Finite difference Solutions of Partial Difference Equations", *J. Comput. Phys.*, 95, 2, (1991).

[11] W. Huang, Y. Ren, and R. Russell, "Moving Mesh Methods Based on Moving Mesh Partial Differential Equations",*J. Comput. Phys.*, 113, 2, (1994).

[12] J. Moser "Volume Elements of a Riemann Manifold",*Trans AMS*, 120, (1965).

[13] B. Dacorogna and J. Moser, "On a PDE Involving the Jacobian Determinant", *Ann Inst. H Poincare*, 7, (1990).

[14] G. Liao and D. Anderson, "A New Approach to Grid Generation", *Applicable Analysis*, 44, (1992).

[15] G. Liao and J. Su, "A Moving Grid Method for (1+1) Dimension", *Appl Math Lett.*, 8, (1995).

[16] B. Semper and G. Liao,"A Moving Grid Finite Element Method using Grid Deformation", *Numerical Methods of PDEs*, 11, (1995).

[17] P. Bochev, G. Liao, and G. dela Pena, "Analysis and Computation of Adaptive Moving Grids by Deformation", *Numerical Methods of PDEs*, 12, (1996).

[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. Scientific Computing*, 20, 3, (1998), pp. 811-825. (published on www.siam.org)

[19] F. Liu and A. Jameson, "Multi-grid Navier-Stokes Calculation for Three-Dimensional Cascades", *AIAA J.*, 31, 10, (1993), pp. 1785-1791.

[20] F. Liu and Zheng, "A Strongly Coupled Time-Marching Method for Solving the Navier-Stokes and Turbulance Model Equations with Multigrid",*J. Comput. Phys.*, 128, (1996) pp. 289-300

[21] G. Liao, Proceedings of the 15th IAMCS World Congress, Vol. 2, pp. 155-160, Berlin, August 1997.

[22] G. Liao and G. dela Pena, "A Moving Grid Finite Difference Algorithm for PDEs". preprint.

[23] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang, "A PDE Based Fast Level Set Method", UCLA CAM Report 98-25, (1998).

[24] S. Chen, B. Merriman, S. Osher, P. Smereka, "A Simple Level Set Methods for Solving Stefan Problems", *J. Comput. Phys.*, 134, (1997), pp. 236-252.

[25] S. Osher and J. Sethian,"Fronts propogating with Curvature Dependent Speed: Algorithms based on Hamilton-Jacobi Formulations", *J. Comput. Phys.*, 79, 1, (1988), pp.12-49.

[26] Chorin and Marsden, A Mathematical Introduction to Fluid Dynamics, third edition, (Springer-Verlag, 1993).

[27] Meyer, Richard, Introduction to Mathematical Fluid Dynamics, (John Wiley and Son, Inc. 1971)

[28] S. Osher and C. W. Shu, "High order essentially non-oscillatory schemes for Hamilton -Jacobi equations," *SIAM J. Numer. Analys.*, Vol 28, (1991) pp. 907-921

[29] M. Sussman, P. Smereka and S. Osher, "A level set method for computing solutions to incompressible two-phase flow," *J. Comput. Phys.*, 114, (1994), pp. 146-159.

[30] B. Merriman, R. Caflisch and S. Osher, "Level set methods with an application to modelling thegrowth of thin films," Proc of 1997 Congress on Free boundary Problems, Heraklion, Crete, Greece, June 1997, to appear.

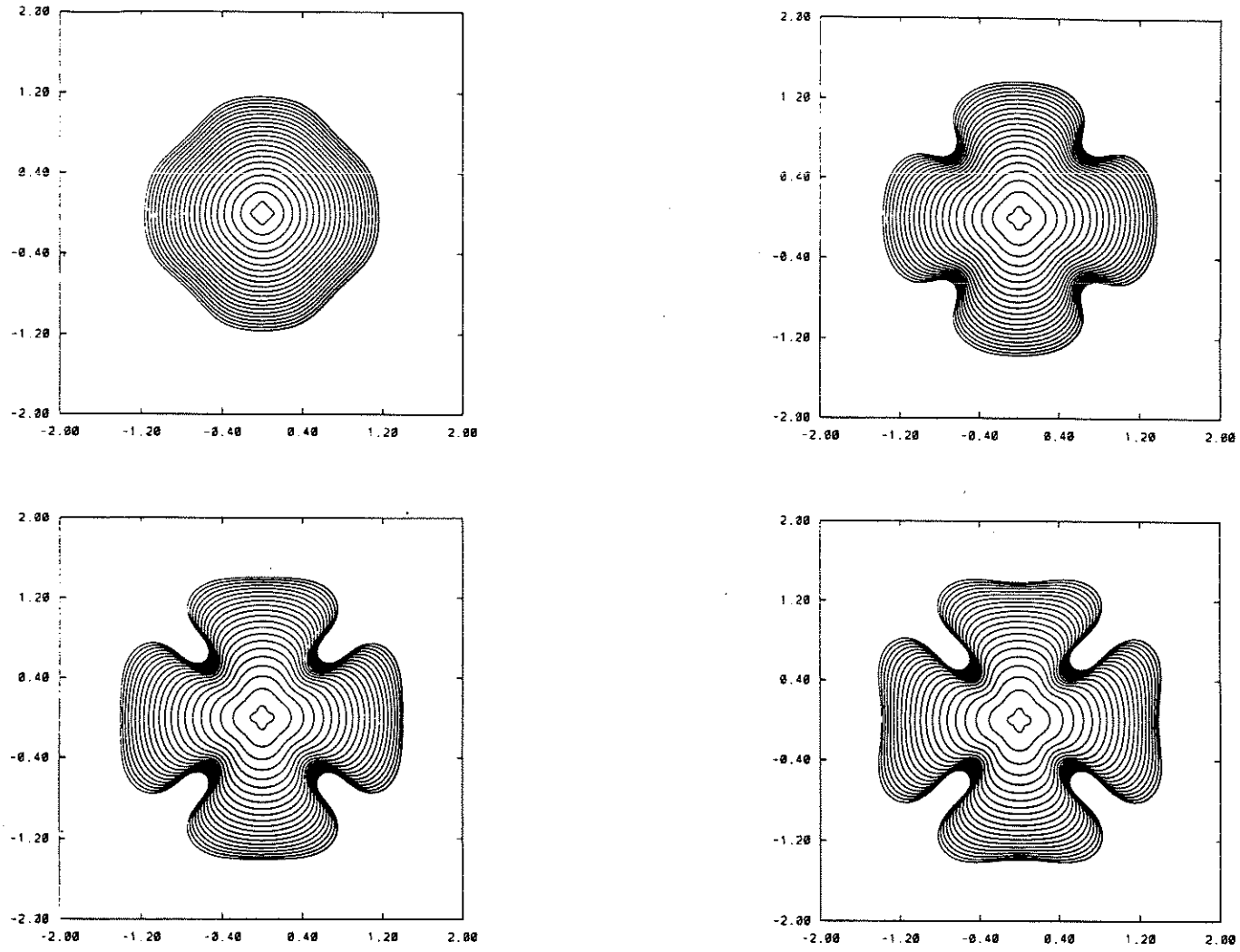

FIG. 8. Convergence study: growth histories for 4 grid resolutions. The grid sizes used are: 100 × 100 (top left), 200 × 200 (top right), 300 × 300 (bottom left), and 400 × 400 (bottom right).

S. Chen, B. Merriman, S. Osher, P. Smereka, "A Simple Level Set Method for Solving Stefan Problems", *J. of Comp. Physics*, **135**, (1997), pp.8-29.

Figure 1. Interface Calculation by a Level Set method on Different Uniform Grids [22].

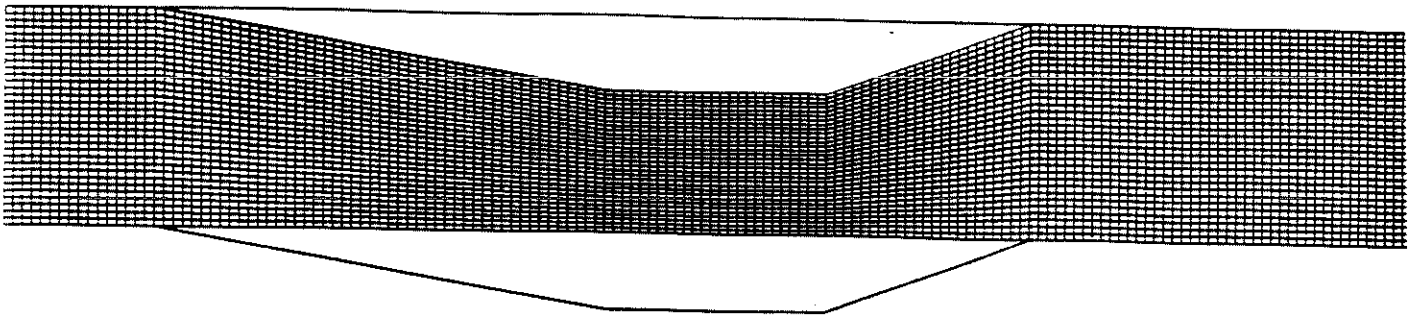

Figure 2a The Original 161 × 33 Grid on the Region between Two Wedges

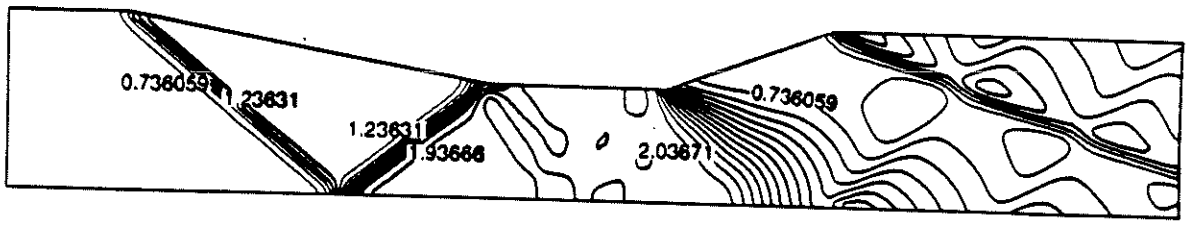

Figure 2b Pressure Contours Over the Original Grid

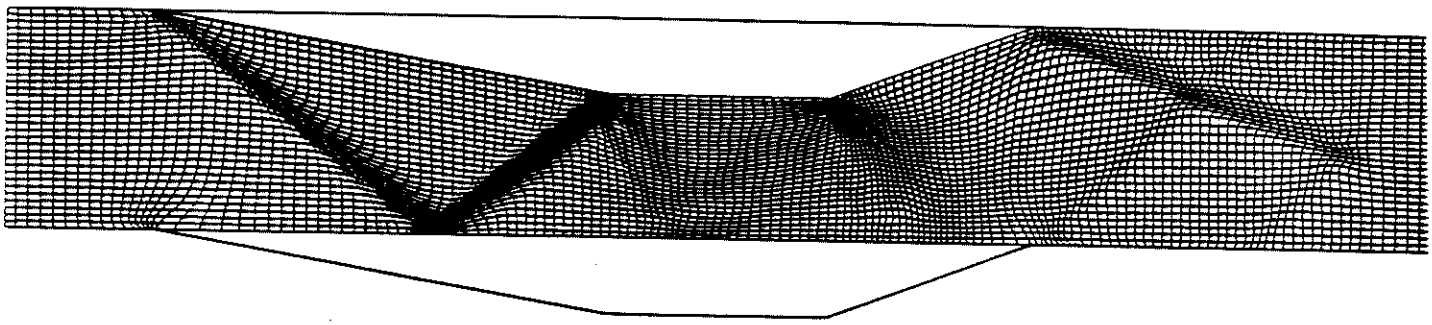

Figure 2c Adapted 161 × 33 Grid by the Deformation Method

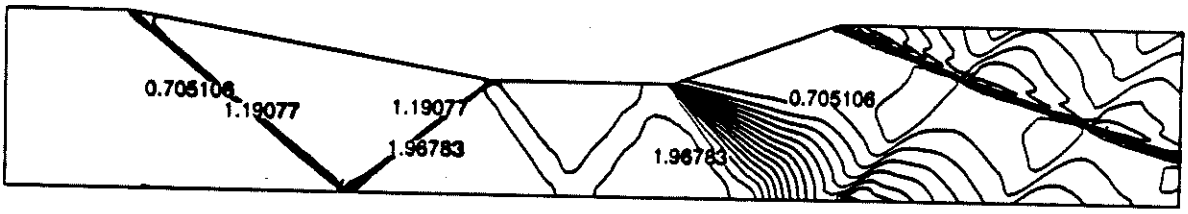

Figure 2d Presure Contours Over the Adapted Grid

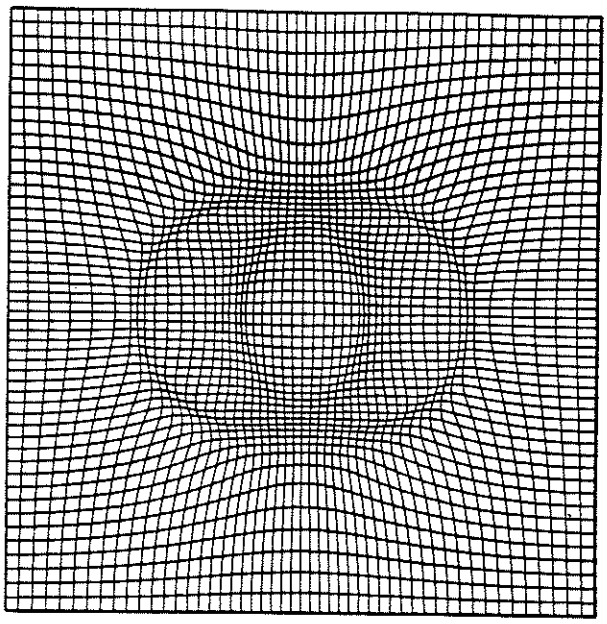

Figure 3a: Grid plot for example 3 for t = 1.375

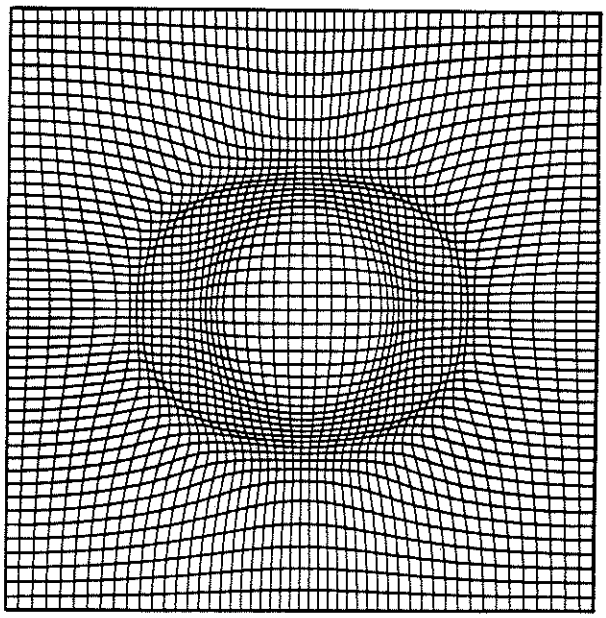

Figure 3b: Grid plot for example 3 for t = 2.75

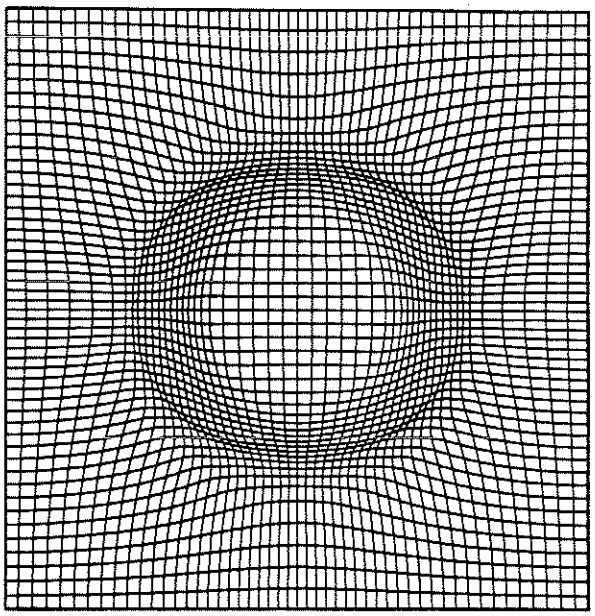

Figure 3c: Grid plot for example 3 for t = 4.125

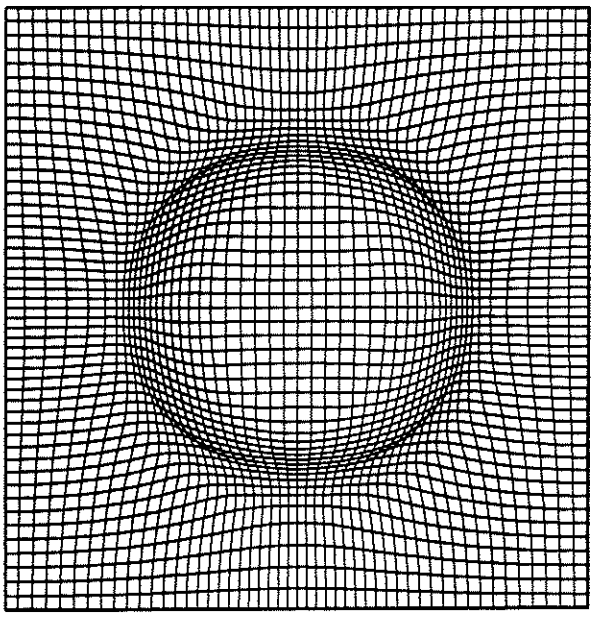

Figure 3d: Grid plot for example 3 for t = 5.5

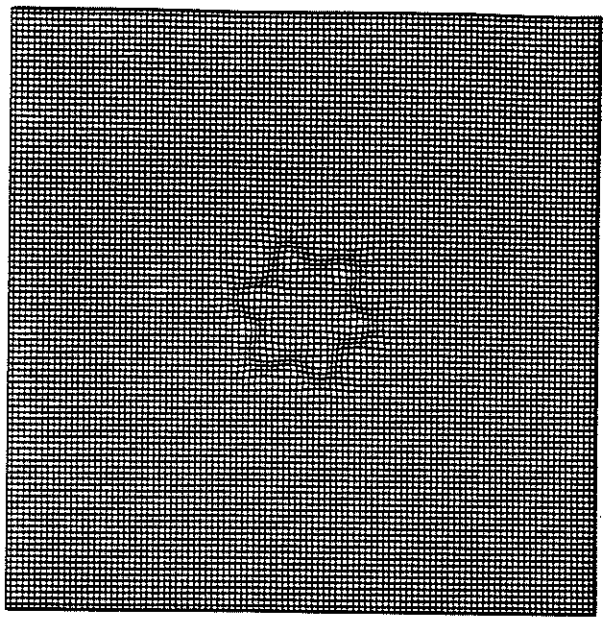

Figure 4a: Grid plot for example 4 for t = 0.0

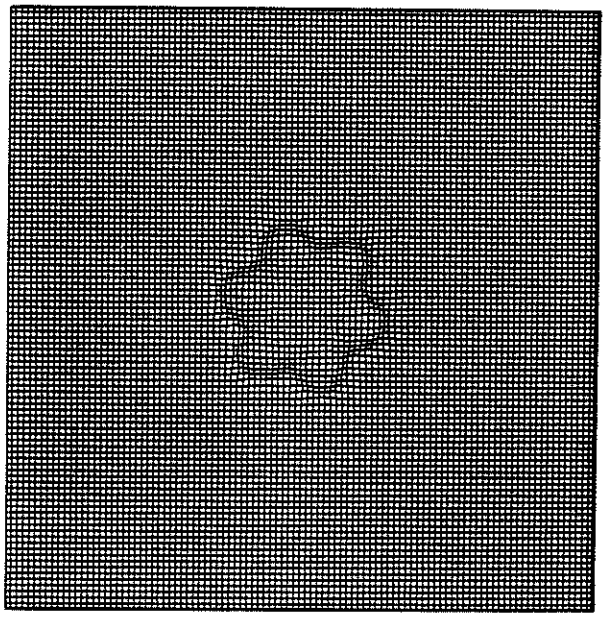

Figure 4b: Grid plot for example 4 for t = 0.005

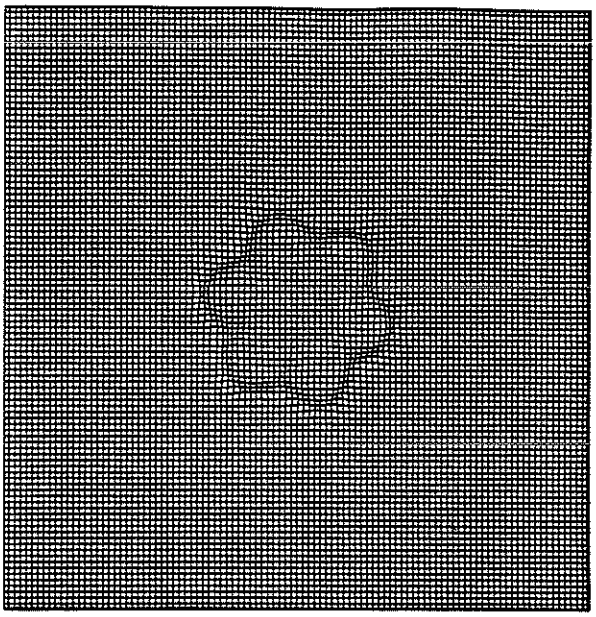

Figure 4c: Grid plot for example 4 for t = 0.01

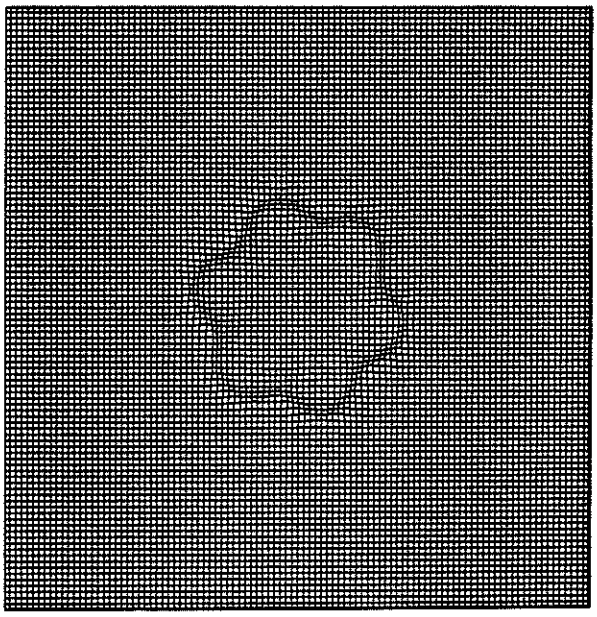

Figure 4d: Grid plot for example 4 for t = 0.015

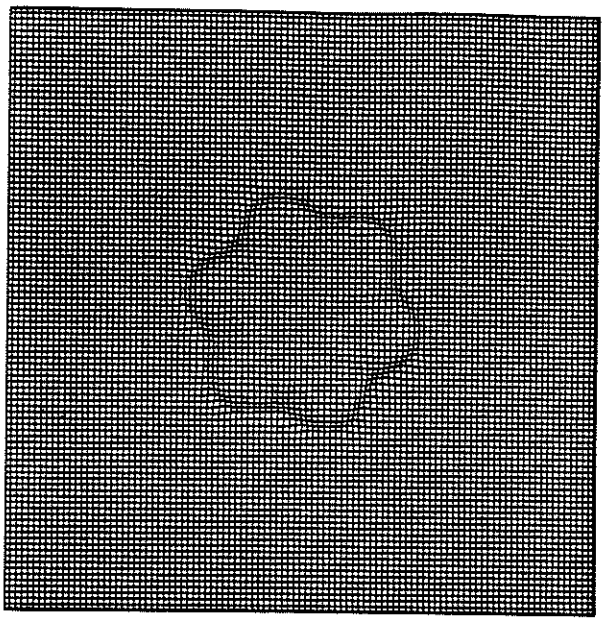

Figure 4e: Grid plot for example 4 for t = 0.02

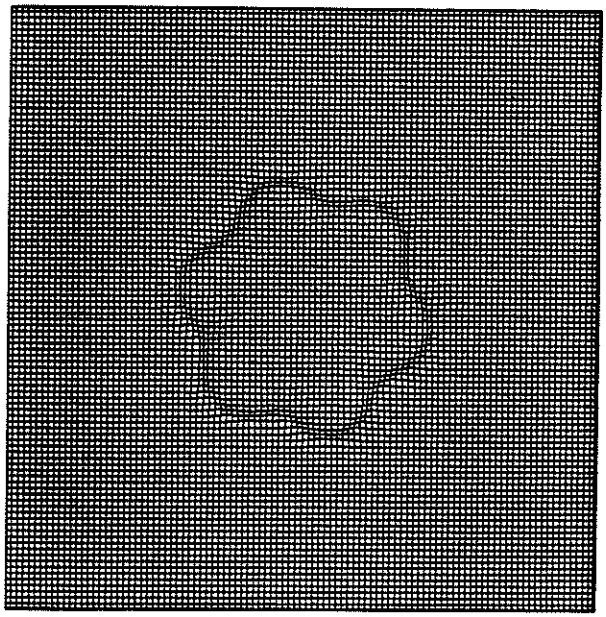

Figure 4f: Grid plot for example 4 for t = 0.025

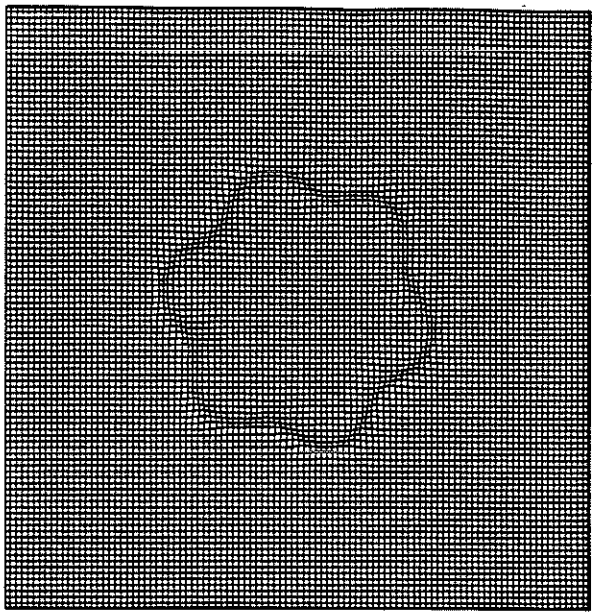

Figure 4g: Grid plot for example 4 for t = 0.03

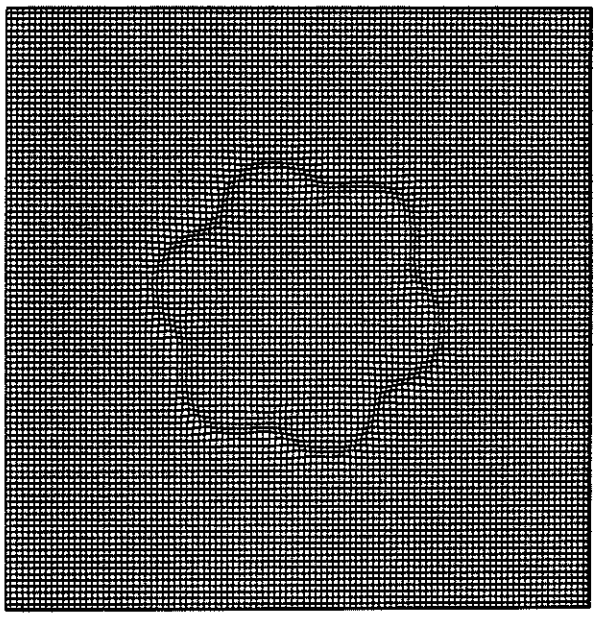

Figure 4h: Grid plot for example 4 for t = 0.035

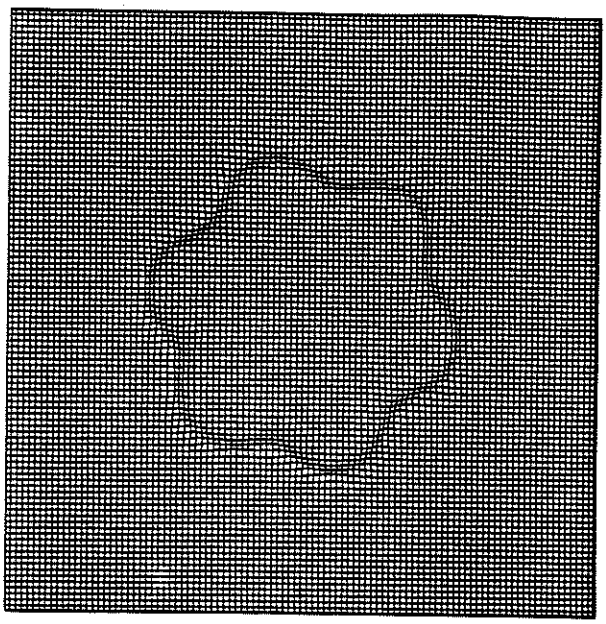

Figure 4i: Grid plot for example 4 for t = 0.04

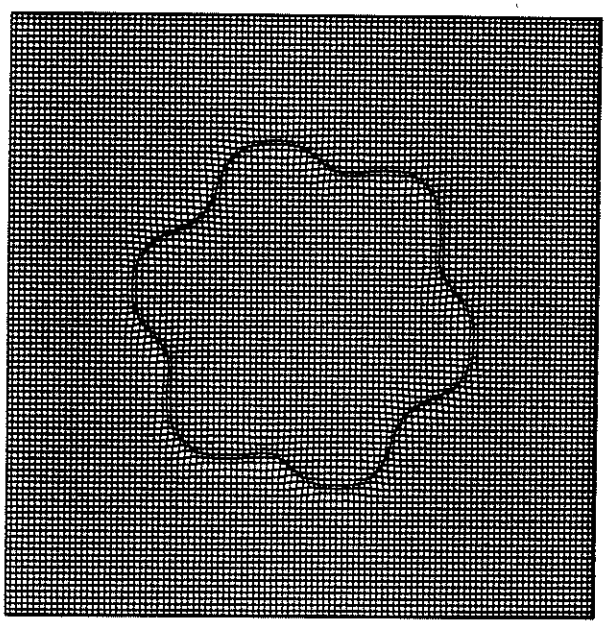

Figure 4j: Grid plot for example 4 for t = 0.05

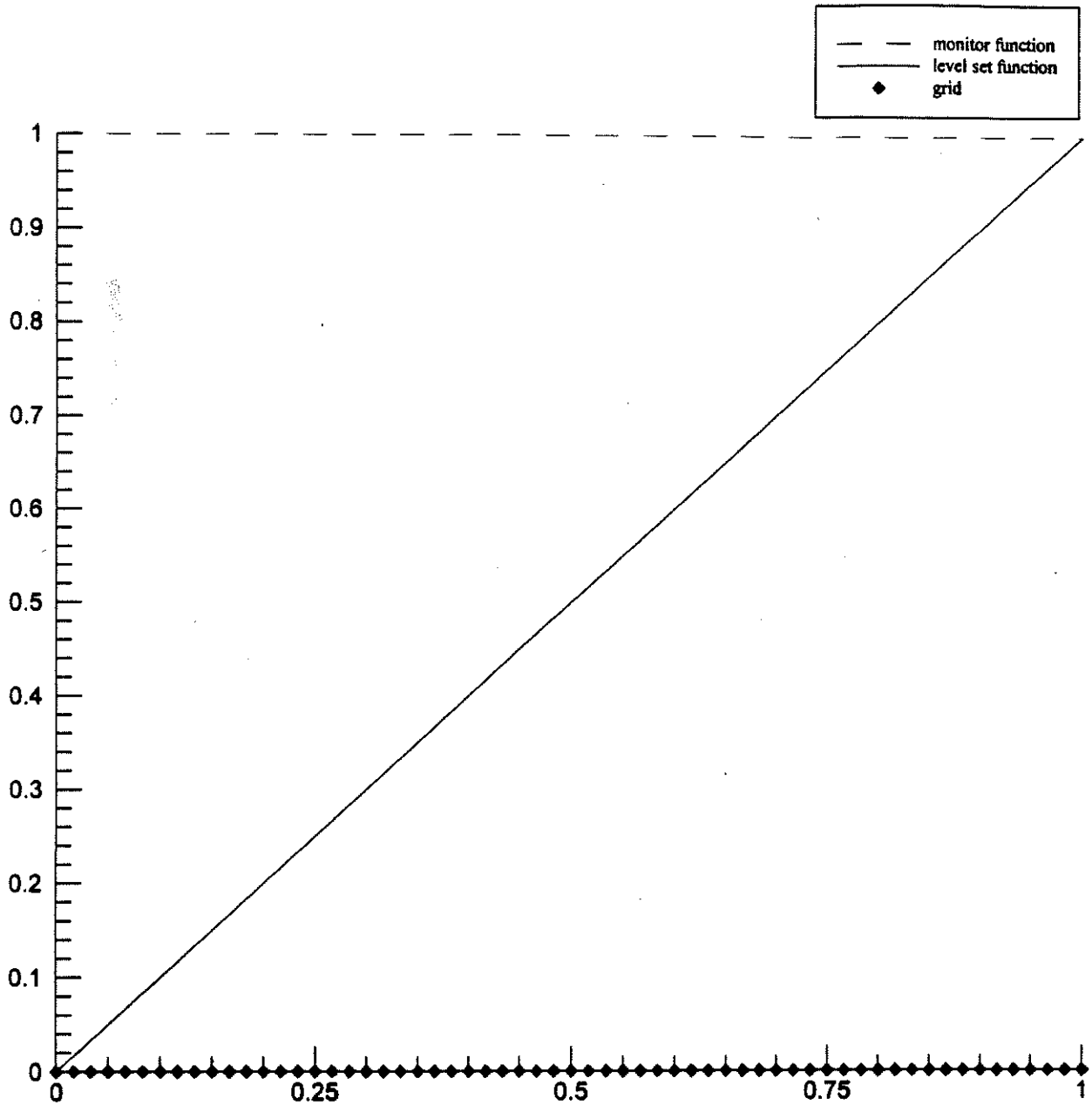

Figure 5a: Monitor function, level set function and grid plots for example 5 for t = 0.0

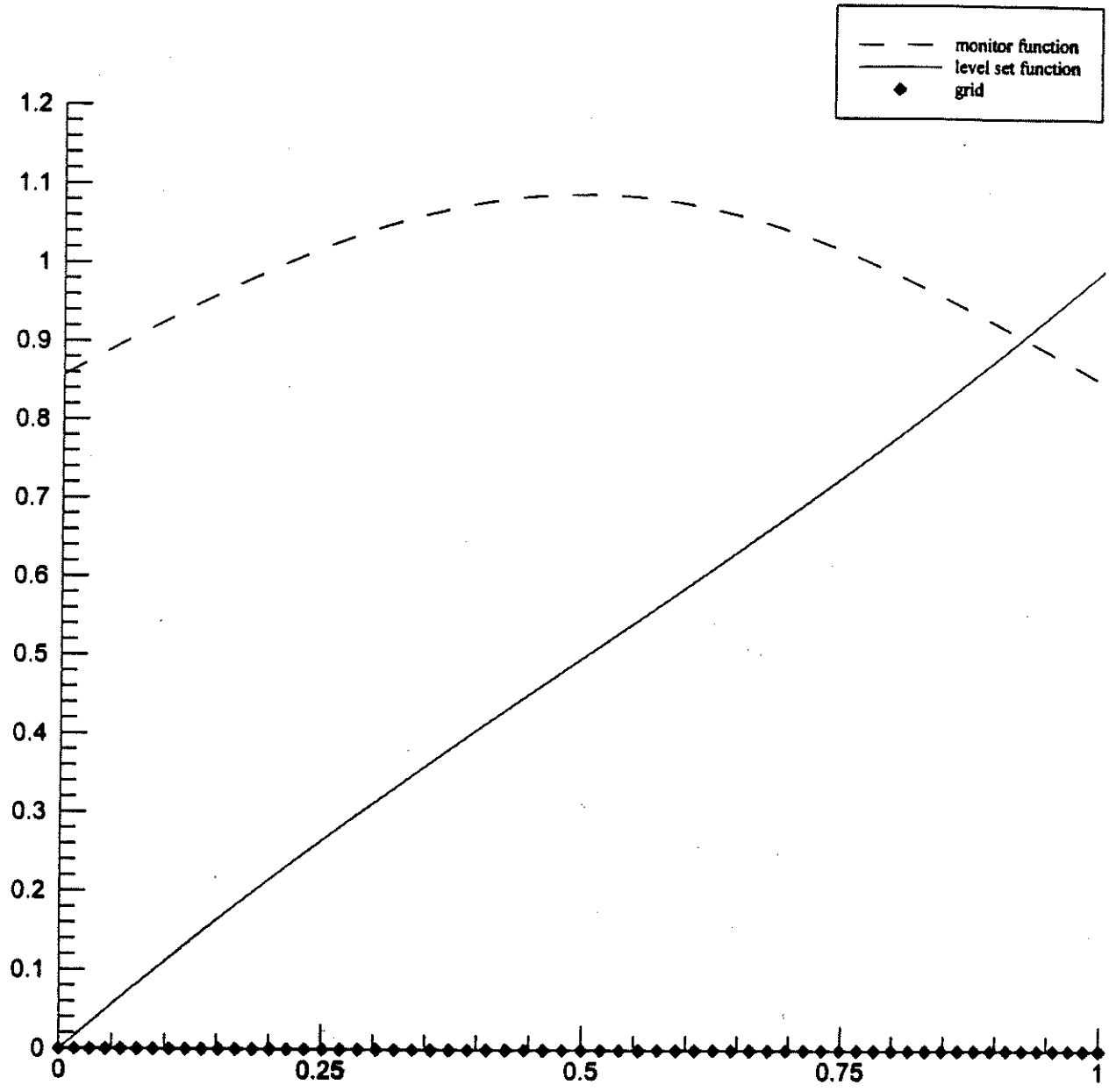

Figure 5b: Monitor function, level set function and grid plots for example 5 for t = 0.1

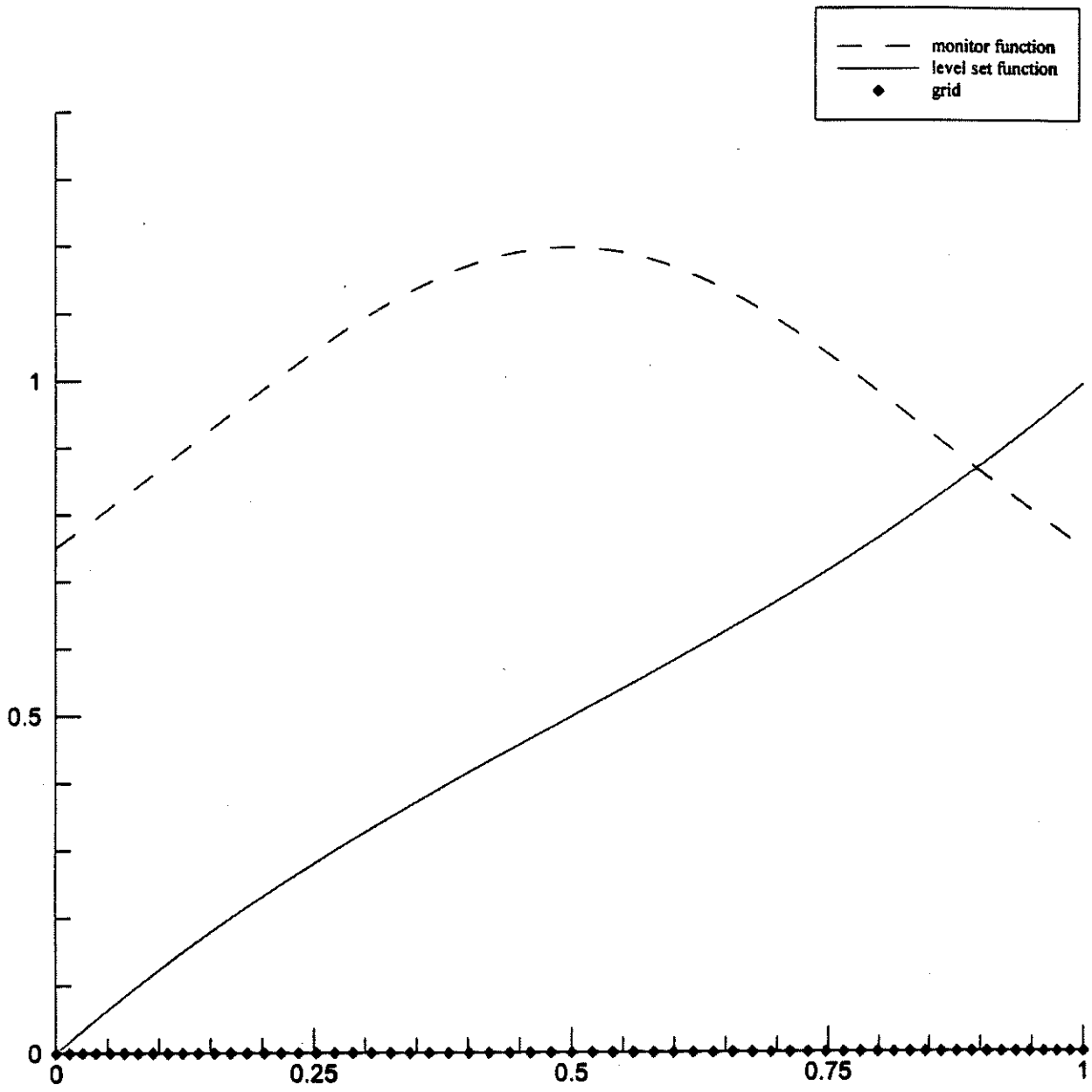

Figure 5c: Monitor function, level set function and grid plots for example 5 for t = 0.2

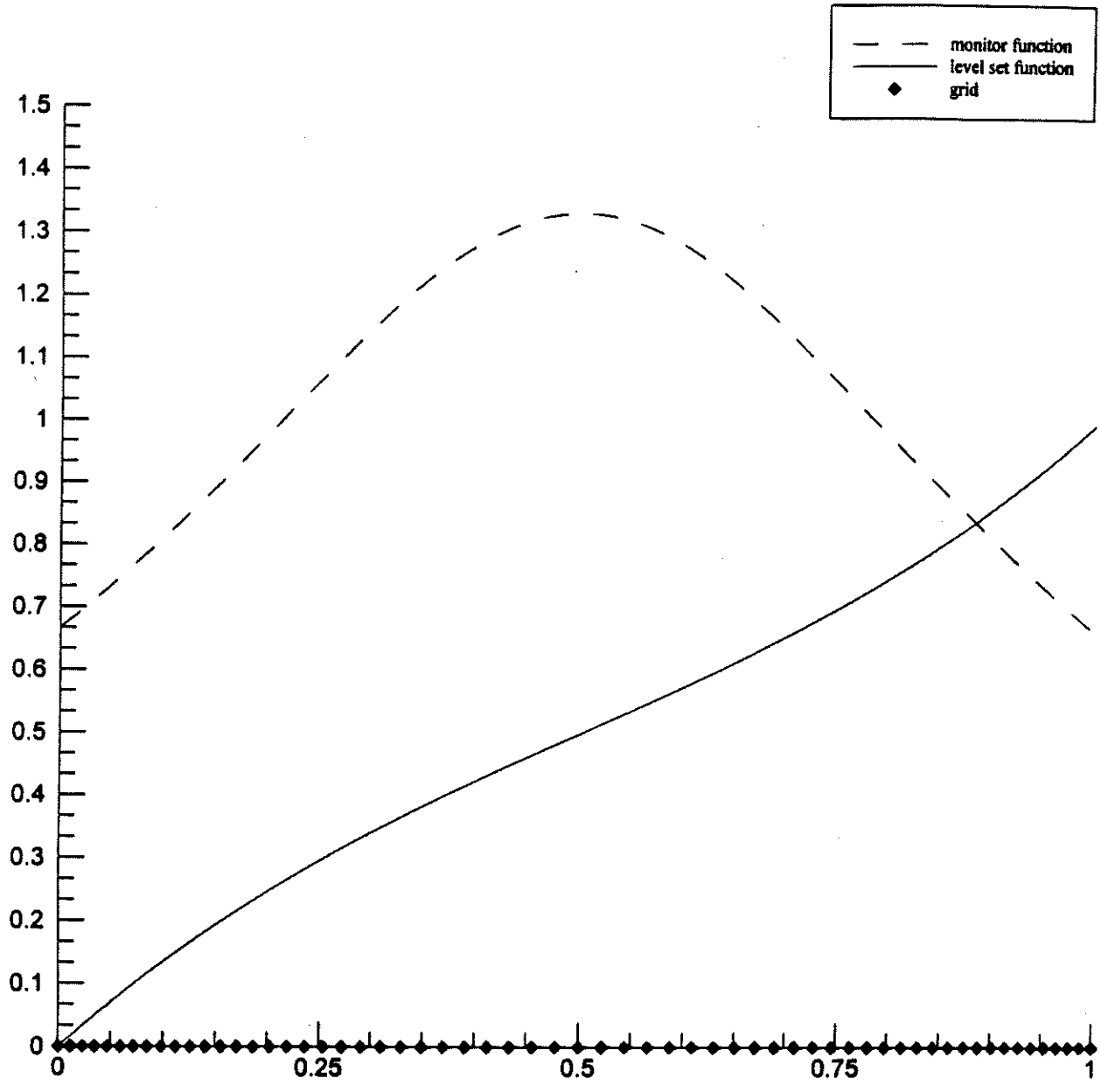

Figure 5d: Monitor function, level set function and grid plots for example 5 for t = 0.3

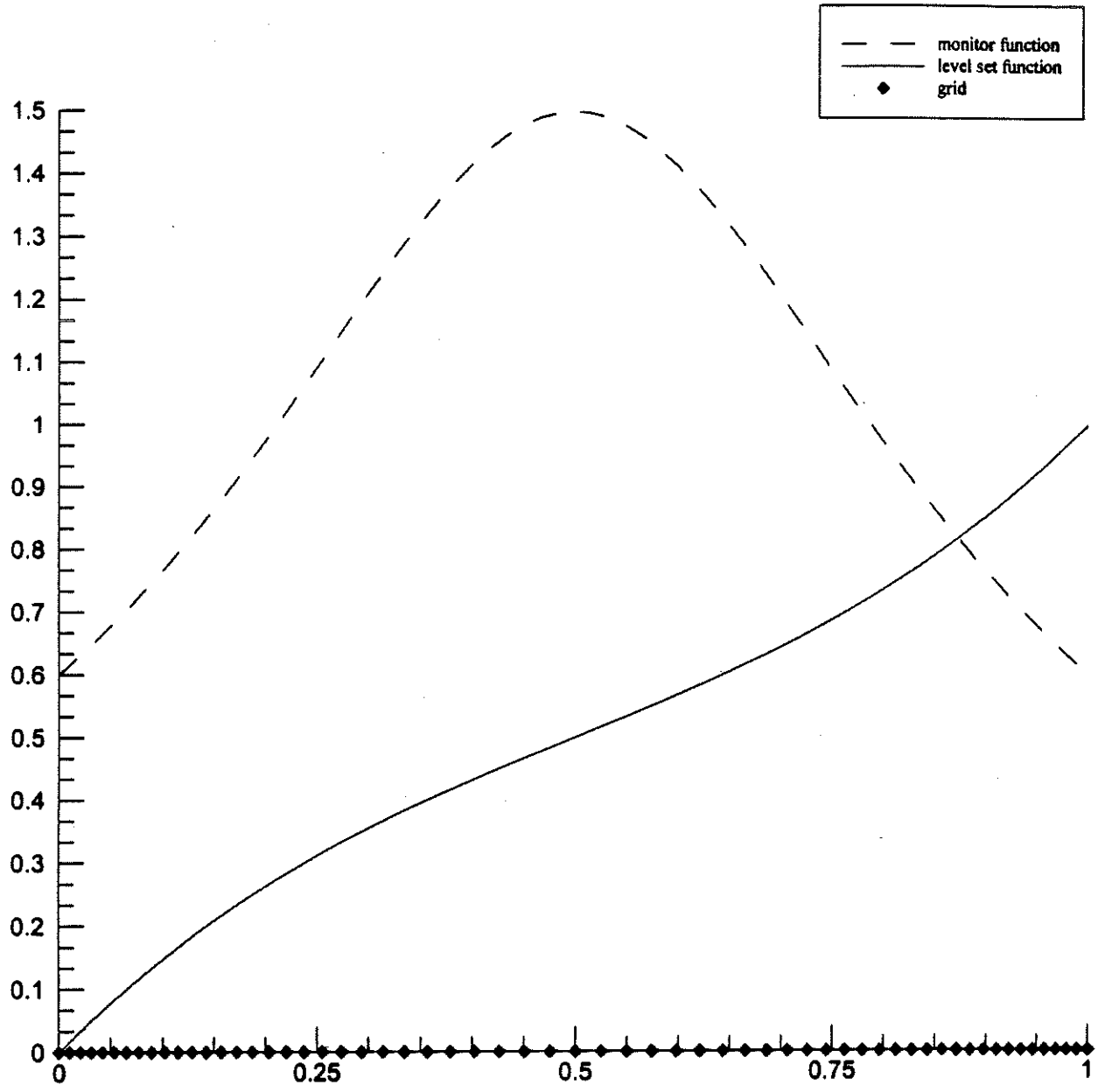

Figure 5e: Monitor function, level set function and grid plots for example 5 for t = 0.4

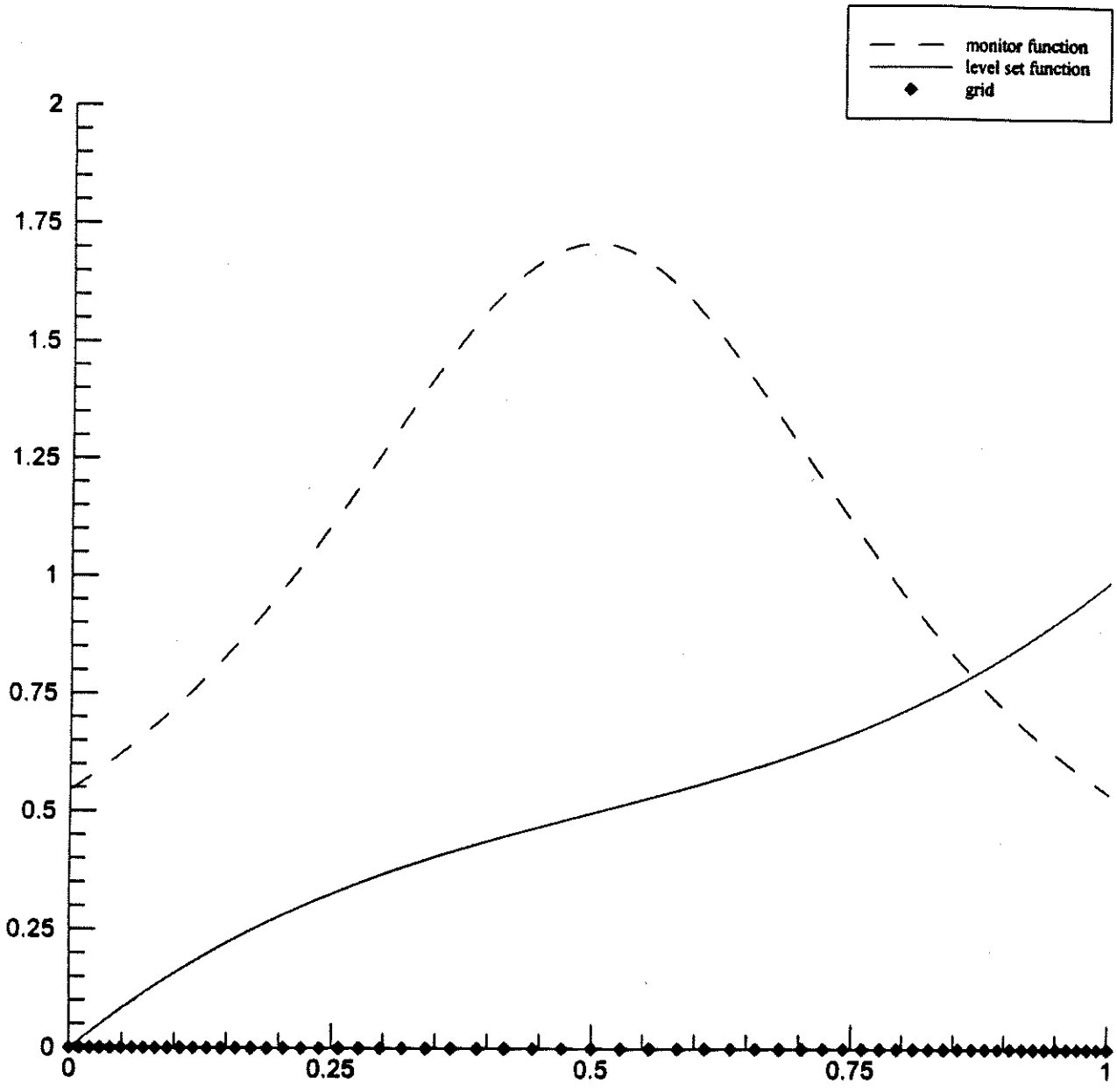

Figure 5f: Monitor function, level set function and grid plots for example 5 for t = 0.5

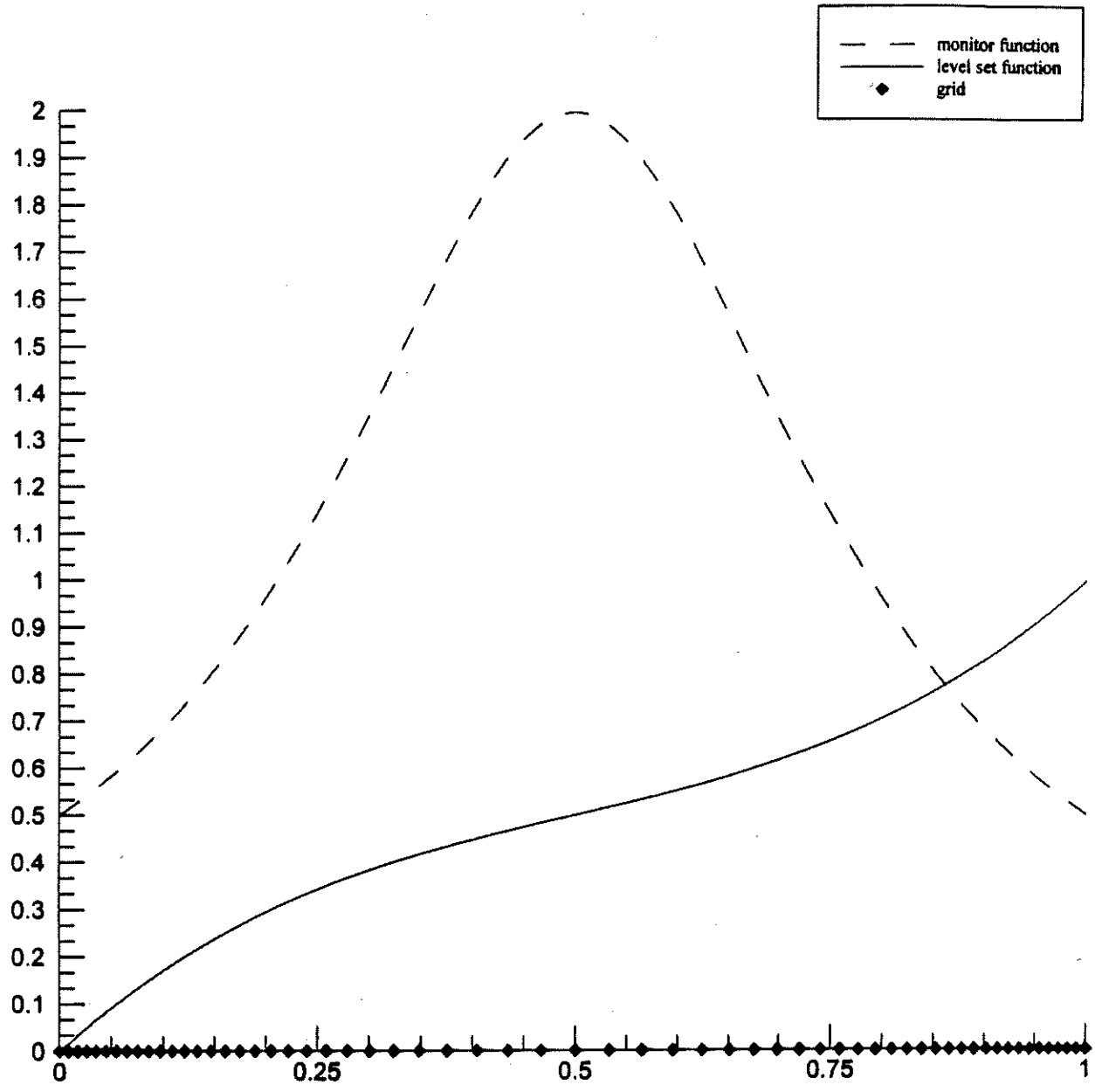

Figure 5g: Monitor function, level set function and grid plots for example 5 for t = 0.6

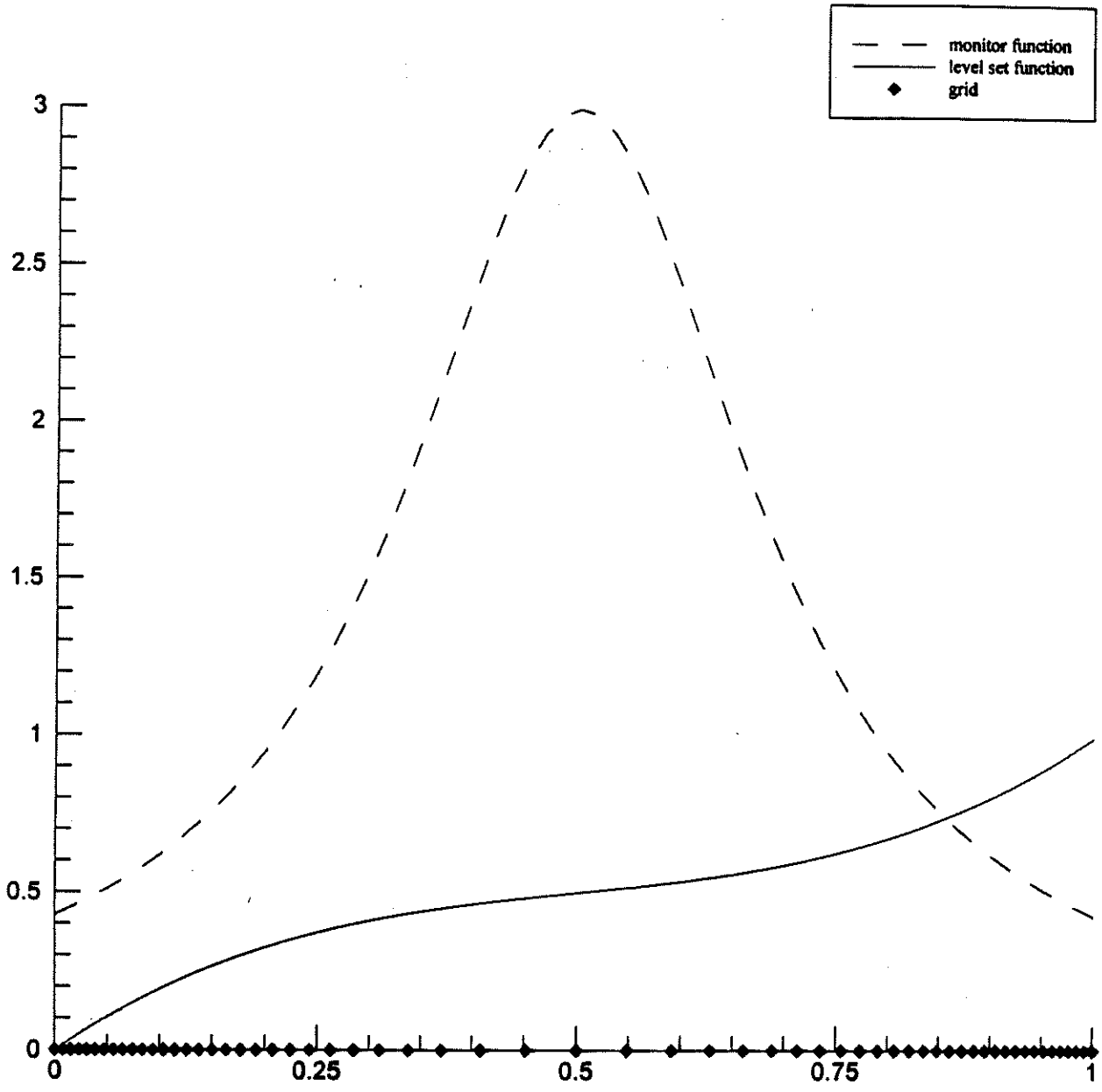

Figure 5h: Monitor function, level set function and grid plots for example 5 for t = 0.8

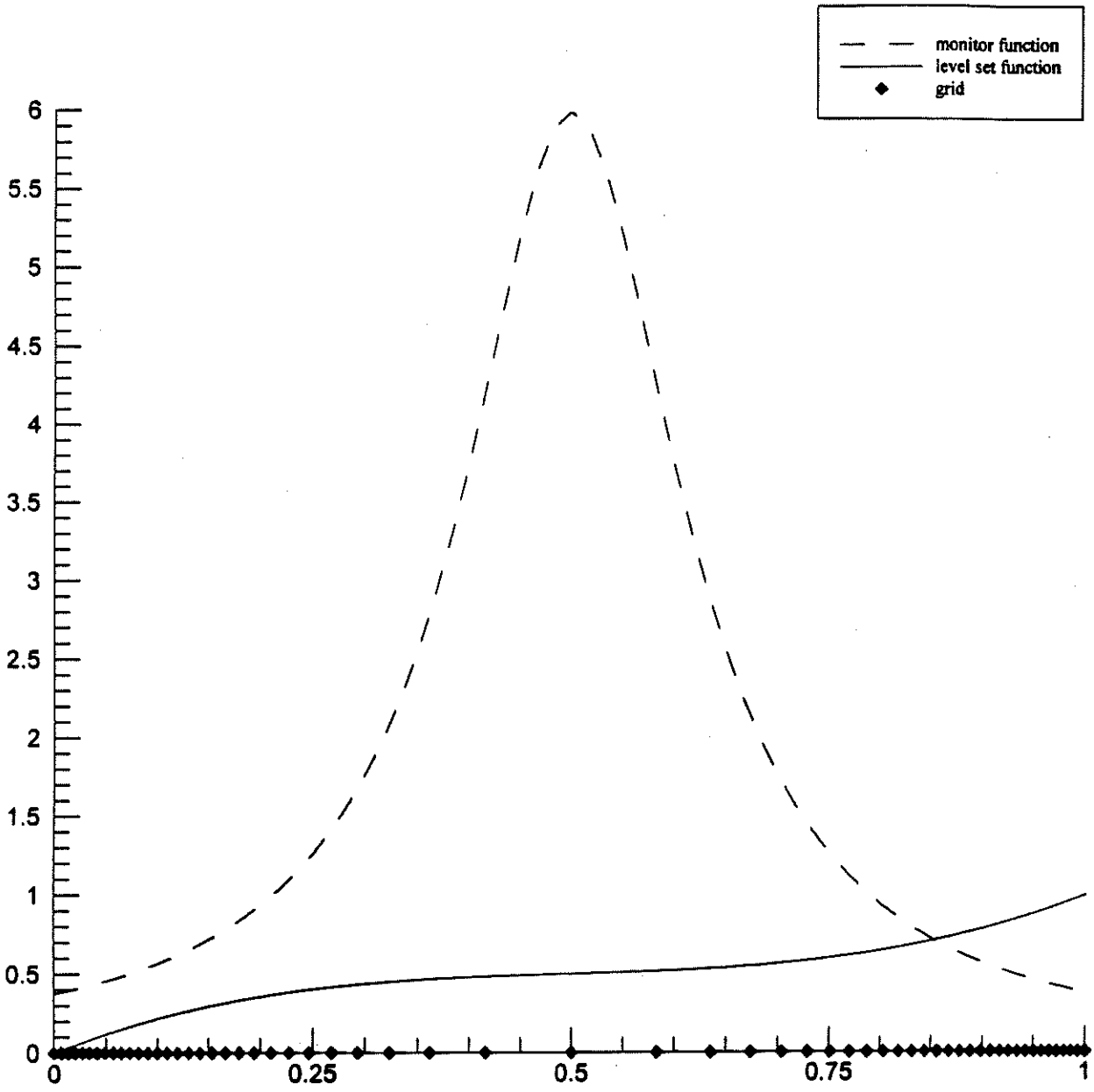

Figure 5i: Monitor function, level set function and grid plots for example 5 for t = 1.0

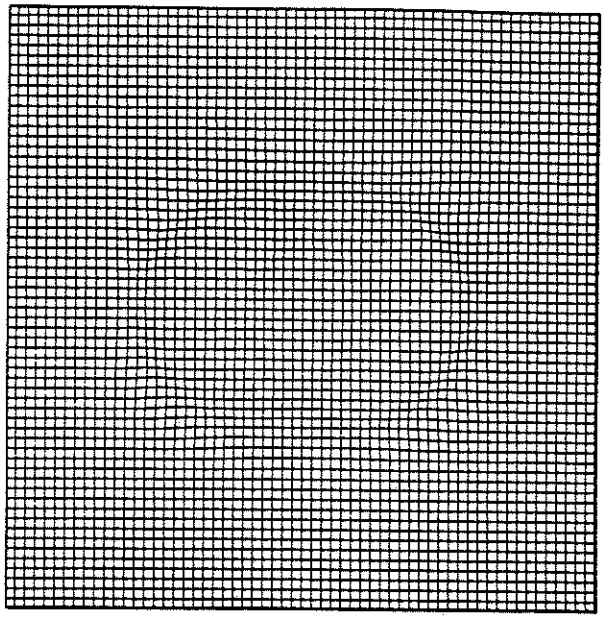

Figure 6a: Grid plot for example 6 for t = 0.25

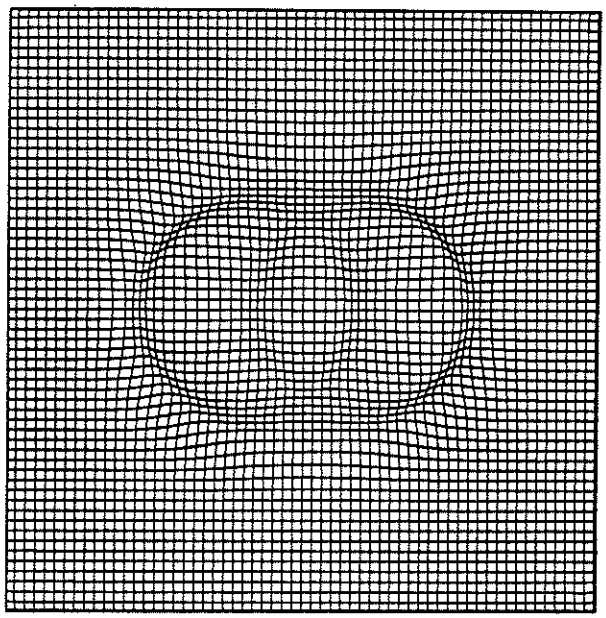

Figure 6b: Grid plot for example 6 for t = 0.5

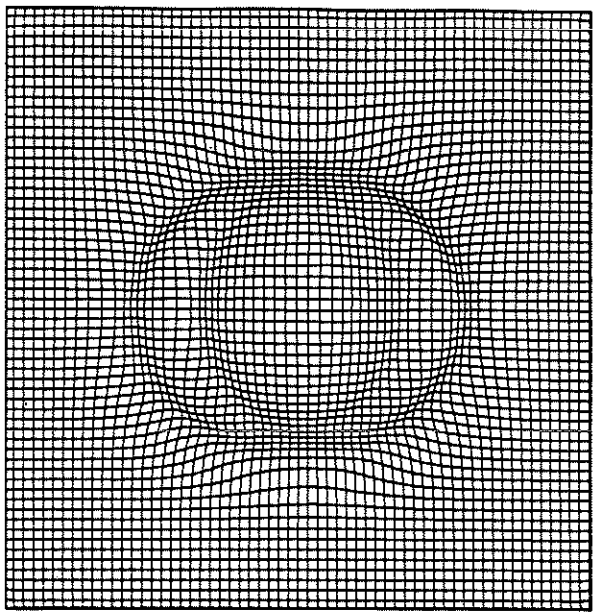

Figure 6c: Grid plot for example 6 for t = 0.75

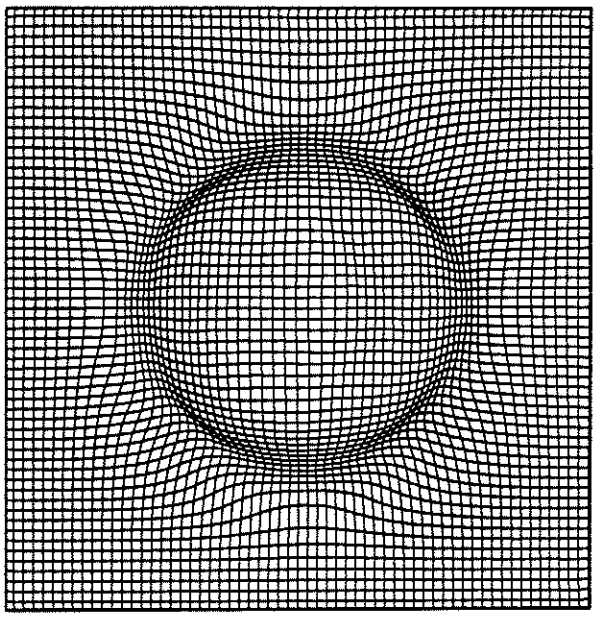

Figure 6d: Grid plot for example 6 for t = 1.0

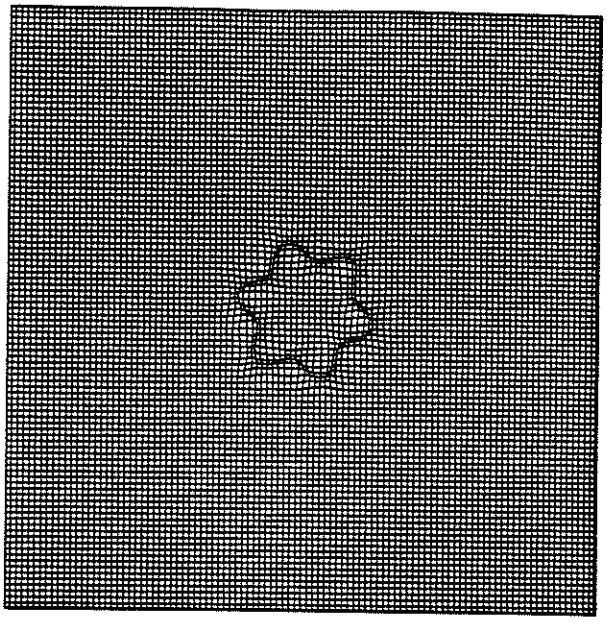

Figure 7a: Grid plot for example 7 for t = 0.0

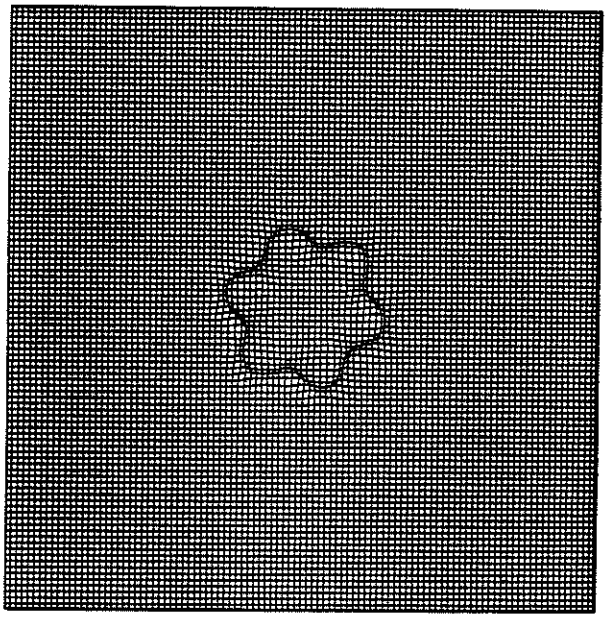

Figure 7b: Grid plot for example 7 for t = 0.005

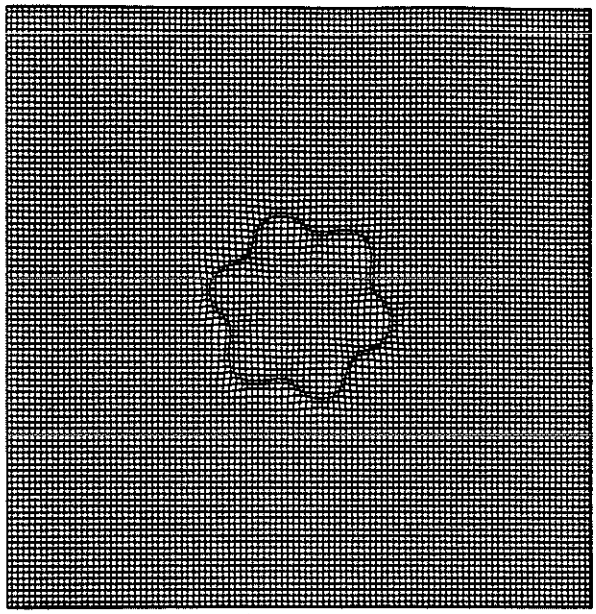

Figure 7c: Grid plot for example 7 for t = 0.01

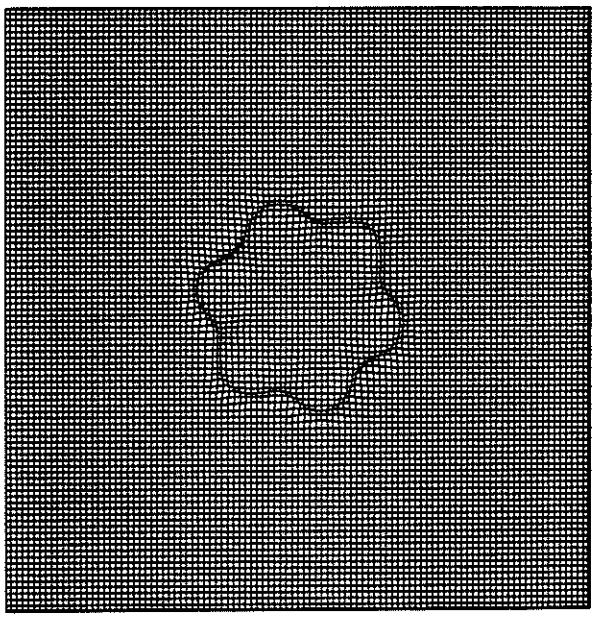

Figure 7d: Grid plot for example 7 for t = 0.015

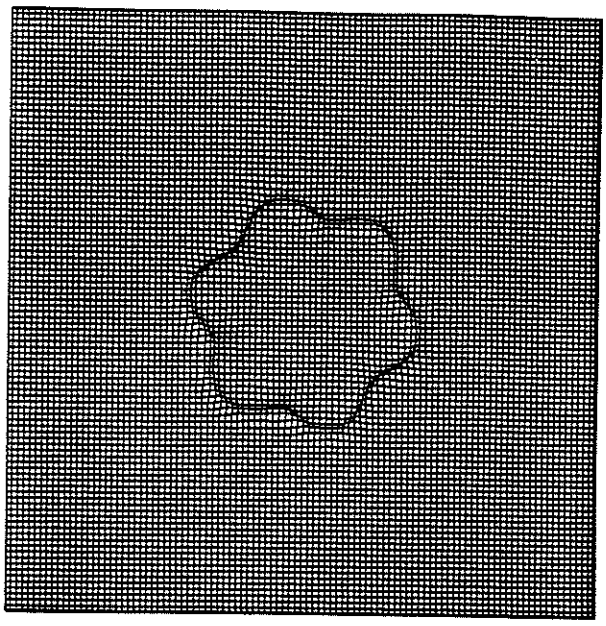

Figure 7e: Grid plot for example 7 for t = 0.02

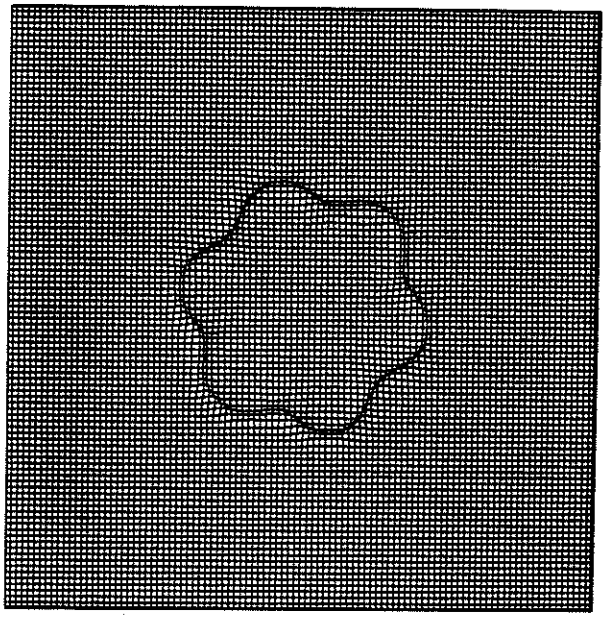

Figure 7f: Grid plot for example 7 for t = 0.025

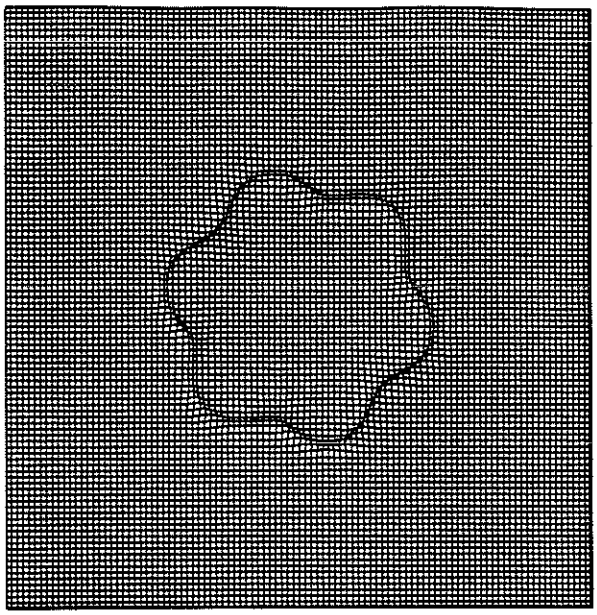

Figure 7g: Grid plot for example 7 for t = 0.03

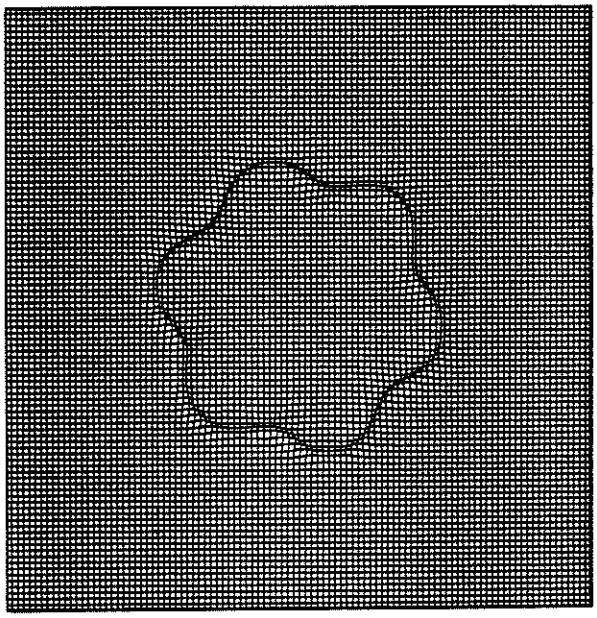

Figure 7h: Grid plot for example 7 for t = 0.035

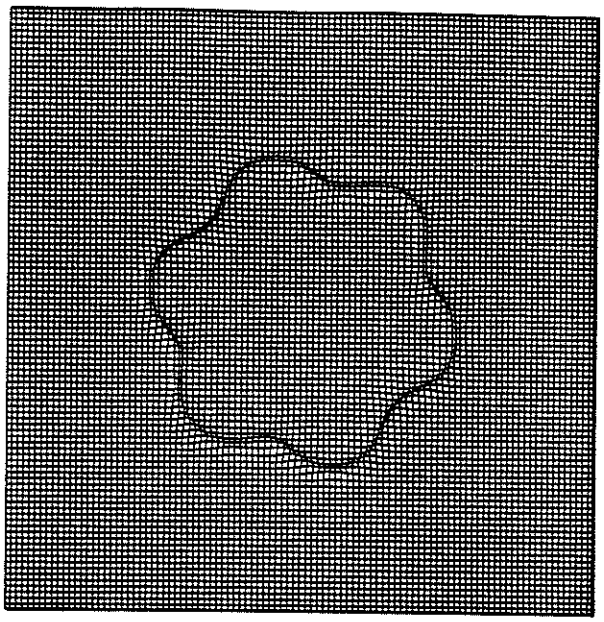

Figure 7i: Grid plot for example 7 for t = 0.04

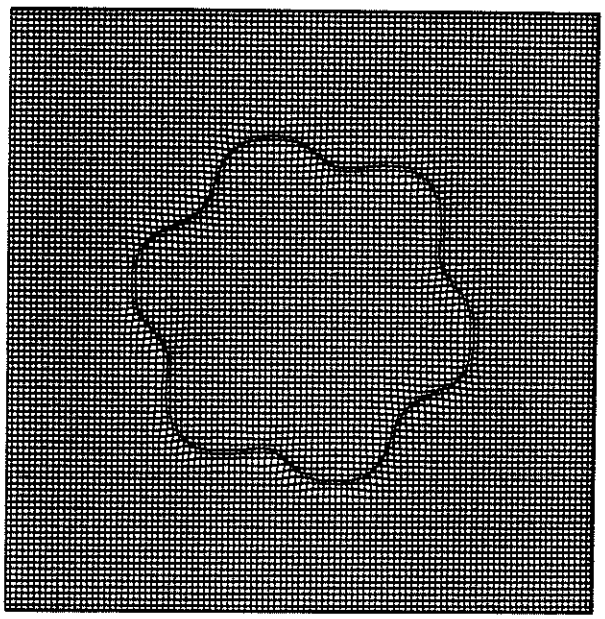

Figure 7j: Grid plot for example 7 for t = 0.05