Proceedings of the Czech–Japanese Seminar in Applied Mathematics 2006
Czech Technical University in Prague, September 14-17, 2006
pp. 156–165

APPLICATION OF NON-LINEAR DIFFUSION IN ALGORITHMS OF MATHEMATICAL VISUALIZATION

JAN MACH[1]

Abstract. The article briefly summarizes numerical solution of a parabolic equation used in the context of mathematical visualization. The presented nonlinear initial-boundary value problem serve in vector field visualization. Its numerical solution is based on spatial discretization by finite differences, and time discretization is given by the Runge-Kutta-Merson scheme. A simple parallelization using POSIX threads is shown to speed-up the numerical computation. The computational results demonstrate the process of vector field pattern sharpening in an initial noise and computation speed-up due to parallelization.

Key words. Degenerate diffusion, FDM, method of lines, mathematical visualization, parallel programming, POSIX threads.

AMS subject classifications. 35K55, 35K60, 35K65, 35K50, 35K45, 68N19

1. Introduction. Numerical simulations often result in large amounts of data which we need to display suitably to be able to understand them. The field studying these problems is called mathematical visualization. Numerical solutions of flow models simulating some natural phenomena such as liquid flow (see [5]), pollution transport in atmosphere (see [8]) give as results describing some vector field. That is why a vector field visualization is an important part of mathematical visualization. There are many methods which can be used for vector field visualization (see [6]) and new ones are still being developed. One of the modern methods is based on anisotropic diffusion. This paper deals with a numerical solution of degenerate parabolic equation, a key part of anisotropic diffusion model.

2. Anisotropic diffusion model. Let Ω be a bounded domain in $\mathbf{R}^2$. Let $v : \Omega \to \mathbf{R}^2$ be a given vector field on the domain Ω, which we assume to be continuous and non-vanishing. One of the modern methods used for vector field visualization is based on solving the following initial-boundary value problem for a non-linear anisotropic diffusion PDE (see [9])

$$\frac{\partial}{\partial t}\rho - div(\mathbf{A}(v, \|\nabla\rho_\varepsilon\|)\nabla\rho) = f(\rho) \qquad \text{in } \Omega \times (0,T), \tag{2.1a}$$

$$\rho(0,x) = \rho_0(x) \qquad x \in \Omega, \tag{2.1b}$$

$$\frac{\partial}{\partial \vec{n}}\rho = 0 \qquad x \in \partial\Omega, t \in (0,T), \tag{2.1c}$$

where $\boldsymbol{A}$ is nonlinear function of gradient $\nabla\rho_\varepsilon$ called diffusion matrix defined as

$$\boldsymbol{A}(v, \|\nabla\rho_\varepsilon\|) = \boldsymbol{B}(v)^T \begin{pmatrix} \alpha(\|v\|) & 0 \\ 0 & G(\|\nabla\rho_\varepsilon\|) \end{pmatrix} \boldsymbol{B}(v). \tag{2.2}$$

The function $\rho_\varepsilon = \mathcal{G}_\varepsilon * \rho$ where $\mathcal{G}_\varepsilon$ is a Gaussian kernel causes the problem (2.1) to have better mathematical properties (see [9]). $\mathbf{B}$ is a continuous orthogonal mapping

[1]Department of Mathematics, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University, Prague.

such that $\mathbf{B}(v)v = \|v\|e_1$ where $\{e_i\}_{i=1,2}$ are the axes directions in $\mathbf{R}^2$. Function $\alpha : \mathbf{R}_0^+ \to \mathbf{R}^+$ controls diffusion in the vector field direction.We choose it to be a monotone function satisfying the following properties

$$\lim_{s\to+\infty} \alpha(s) = \alpha_{max}, \quad \alpha_{max} \in \mathbf{R}^+, \tag{2.3a}$$

$$\alpha(0) > 0. \tag{2.3b}$$

The function $G : \mathbf{R}_0^+ \to \mathbf{R}^+$ is an edge enhancing coefficient which acts in the direction orthogonal to the vector field. We choose it to be a monotone function satisfying

$$G(0) = \beta, \quad \beta \in \mathbf{R}^+ \tag{2.4a}$$

$$\lim_{d\to+\infty} G(d) = 0, \tag{2.4b}$$

The evolution for a vanishing right hand side f would decrease image contrast due to diffusion. Therefore, we define $f : [0,1] \to \mathbf{R}$ such as

$$\begin{aligned} &f(0) = 0,\\ &f(1) = 0,\\ &\quad f < 0 \text{ on } (0.0, 0.5),\\ &\quad f > 0 \text{ on } (0.5, 1.0), \end{aligned} \tag{2.5}$$

to increase contrast of the evolving solution. Solving the above problem with some initial noise ρ_0 gives us a solution $\rho(t)$ for all times which can be written in a form where we can see the vector field pattern being successively sharpened.

3. Algorithm and implementation. We solved the problem (2.1) by the method of lines. This method consists of two steps. First, a spatial discretization of the problem is performed. It results in a system of ODE's which is then solved by a modified Runge-Kutta method.

Finite differences are used to perform a spatial discretization. We consider a regular rectangular grid $\omega_{\boldsymbol{h}}$ in 2D. Grid values and spatial differences of the function ρ are denoted as follows

$$\boldsymbol{h} = (h_1, h_2),\ h_1 = \frac{L_1}{N_1},\ h_2 = \frac{L_2}{N_2} \tag{3.1}$$

$$x_{ij} = [x_{ij}^1, x_{ij}^2],\ \rho_{ij} = \rho(x_{ij}) \tag{3.2}$$

$$\rho_{\overline{x}_1,ij} = \frac{\rho_{ij} - \rho_{i-1,j}}{h_1},\ \rho_{x_1,ij} = \frac{\rho_{i+1,j} - \rho_{ij}}{h_1}, \tag{3.3}$$

$$\rho_{\overline{x}_2,ij} = \frac{\rho_{ij} - \rho_{i,j-1}}{h_2},\ \rho_{x_2,ij} = \frac{\rho_{i,j+1} - \rho_{ij}}{h_2} \tag{3.4}$$

and

$$\overline{\nabla}_{\boldsymbol{h}}\rho = [\rho_{\overline{x}_1}, \rho_{\overline{x}_2}], \nabla_{\boldsymbol{h}}\rho = [\rho_{x_1}, \rho_{x_2}] \tag{3.5}$$

Using the above declared notation, we can write the difference scheme for the problem (2.1) in the following form

$$\frac{\mathrm{d}}{\mathrm{d}t}\rho = \nabla_{\boldsymbol{h}} \cdot (\mathbf{A}(v, \|\nabla_{\boldsymbol{h}}\rho_{\boldsymbol{h},\varepsilon}\|)\overline{\nabla}_{\boldsymbol{h}}\rho_{\boldsymbol{h}}) - f(\rho_{\boldsymbol{h}}) \qquad \text{in } \omega_{\boldsymbol{h}} \times (0,T), \tag{3.6a}$$

$$\rho_{\boldsymbol{h}}(0,x) = \rho_{0,\boldsymbol{h}}(x) \qquad x \in \overline{\omega}_{\boldsymbol{h}}, \tag{3.6b}$$

$$\frac{\partial}{\partial\vec{n}}\rho_{\boldsymbol{h}} = 0 \qquad x \in \gamma_{\boldsymbol{h}}, t \in (0,T), \tag{3.6c}$$

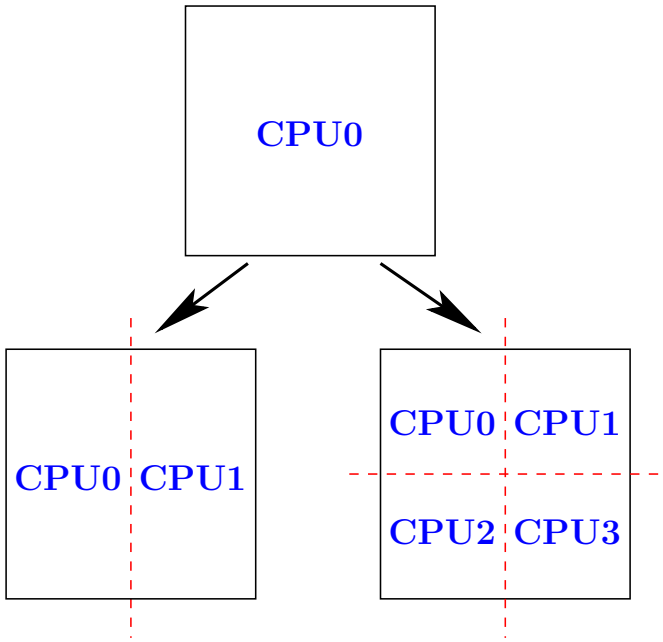


FIG. 3.1. *Splitting of the domain Ω for parallel computation.*

which can be rewritten as the following system of ODE's

$$\begin{aligned}\frac{\partial}{\partial t}\rho(t,x_{i,j}) = \rho_{i+1,j}\Big(\frac{A_{11}^{i+1,j}}{h_1^2} + \frac{A_{12}^{i+1,j}}{h_1h_2}\Big) - \rho_{i,j}\Big(\frac{A_{11}^{i+1,j}}{h_1^2} + \frac{A_{11}^{i,j}}{h_1^2} + \\ \frac{A_{12}^{i,j}}{h_1h_2} + \frac{A_{21}^{i,j}}{h_1h_2} + \frac{A_{22}^{i,j+1}}{h_2^2} + \frac{A_{22}^{i,j}}{h_2^2}\Big) + \rho_{i-1,j}\Big(\frac{A_{11}^{i,j}}{h_1^2} + \frac{A_{21}^{i,j}}{h_1h_2}\Big) - \\ - \rho_{i+1,j-1}\frac{A_{12}^{i,j}}{h_1h_2} + \rho_{i,j-1}\Big(\frac{A_{12}^{i,j}}{h_1h_2} + \frac{A_{22}^{i,j}}{h_2^2}\Big) + \\ + \rho_{i,j+1}\Big(\frac{A_{21}^{i,j+1}}{h_1h_2} + \frac{A_{22}^{i,j+1}}{h_2^2}\Big) - \rho_{i-1,j+1}\frac{A_{21}^{i,j+1}}{h_1h_2} + f(\rho_{ij}), \qquad (3.7a)\end{aligned}$$

$$\rho(0,x_{ij}) = \rho_0(x_{ij}), \qquad (3.7b)$$

$$\begin{aligned}\rho_{i,0}(t) = \rho_{i,1}(t),\ \rho_{N_1,j}(t) = \rho_{N_1-1,j}(t),\\ \rho_{i,N_2}(t) = \rho_{i,N_2-1}(t),\ \rho_{0,j}(t) = \rho_{1,j}(t), \qquad (3.7c)\\ \forall i \in 1,\dots,N_1-1, \forall j \in 1,\dots,N_2-1,\\ \forall t \in (0,T),\end{aligned}$$

where

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}. \qquad (3.8)$$

We used the Runge-Kutta-Merson method (see [11]) to solve the above system of ODE's. It is a modified Runge-Kutta method with adaptive time stepping. The time-step length adaptivity may shorten the time needed for computation. It can be written in the following algorithm

— *Algorithm R.-K.-M.* —

compute $k_{1,ij}(\mathrm{d}t)$
compute $k_{2,ij}(\mathrm{d}t)$
compute $k_{3,ij}(\mathrm{d}t)$
compute $k_{4,ij}(\mathrm{d}t)$
compute $k_{5,ij}(\mathrm{d}t)$
$q = \max\{|2k_{1,ij}(\mathrm{d}t) - 9k_{3,ij}(\mathrm{d}t) + 8k_{4,ij}(\mathrm{d}t) - k_{5,ij}(\mathrm{d}t)|/30\}$
if$(q < adaptivity)$
{
    $y_{ij}(t_0 + \mathrm{d}t) = y_{ij}(t_0) + (k_{1,ij}(\mathrm{d}t) + 4k_{4,ij}(\mathrm{d}t) + k_{5,ij}(\mathrm{d}t))/6$
    $t_0 = t_0 + \mathrm{d}t$
}
$\mathrm{d}t = \mathrm{d}t\omega(adaptivity/q)^{0.2}$

We usually choose $adaptivity \in [10^{-6}, 10^{-3}]$, $\omega \in [0.8, 0.9]$. Denoting f the right hand side in (3.7a) the coefficients $k_1, ..., k_5$ can be written in the following form

$$\begin{aligned}
k_1(\mathrm{d}t) &= \mathrm{d}t f(t_0, y(t_0)) \\
k_2(\mathrm{d}t) &= \mathrm{d}t f(t_0 + \mathrm{d}t/3, y(t_0) + k_1(\mathrm{d}t)/3) \\
k_3(\mathrm{d}t) &= \mathrm{d}t f(t_0 + \mathrm{d}t/3, y(t_0) + (k_1(\mathrm{d}t) + k_2(\mathrm{d}t))/6) \\
k_4(\mathrm{d}t) &= \mathrm{d}t f(t_0 + \mathrm{d}t/2, y(t_0) + 0.125k_1(\mathrm{d}t) + 0.375k_3(\mathrm{d}t)) \\
k_5(\mathrm{d}t) &= \mathrm{d}t f(t_0 + \mathrm{d}t, y(t_0) + 0.5k_1(\mathrm{d}t) - 1.5k_3(\mathrm{d}t) + 2.0k_4(\mathrm{d}t))
\end{aligned} \tag{3.9}$$

The numerical algorithm works on rectangular grids (computation of coefficients $k_1, ..., k_5$ and solution $y(t_0 + \mathrm{d}t)$) and can be easily parallelized. Splitting of the domain Ω is shown on Figure 3.1. Computations on all sub-grids can be performed in parallel. This was implemented using POSIX threads which are commonly available on UNIX systems and enable program parallelization on systems with shared memory. To measure a computation speed-up due to the performed parallelization of numerical algorithm, a series of computations on the following computers was done.

- 4x CPU HP PA-RISC, 1GHz, 12GB RAM
- 2x CPU Dual Core AMD Opteron 270, 2GHz, 4GB RAM

Tables 3.1, 3.2 show time needed to complete the computation for given grid dimension and number of threads. Numbers in brackets in the last two columns indicate, what portion of time needed for the sequential computation was needed for the corresponding parallel computation. We can see a significant speed-up of computations especially for large grids.

grid	1 thread	2 threads	4 threads
50× 50	0:11 (100.00%)	0:08 (72.73%)	0:07 (63.64%)
100× 100	0:46 (100.00%)	0:25 (54.34%)	0:16 (34.78%)
150× 150	1:46 (100.00%)	0:55 (51.89%)	0:32 (33.92%)
200× 200	3:07 (100.00%)	1:37 (51.87%)	0:53 (28.34%)
250× 250	4:54 (100.00%)	2:34 (52.38%)	1:20 (27.21%)
300× 300	7:40 (100.00%)	4:18 (56.08%)	2:02 (26.52%)
400× 400	12:41 (100.00%)	6:29 (51.12%)	3:18 (26.02%)
500× 500	21:06 (100.00%)	10:47 (51.11%)	5:07 (24.25%)
600× 600	31:46 (100.00%)	16:40 (52.47%)	7:26 (23.40%)
700× 700	45:16 (100.00%)	24:04 (53.17%)	11:37 (25.66%)
800× 800	1:00:44 (100.00%)	35:49 (58.97%)	14:49 (24.40%)
900× 900	1:17:16 (100.00%)	43:39 (56.49%)	19:39 (25.43%)
1000× 1000	1:37:50 (100.00%)	58:07 (59.40%)	24:12 (24.74%)
1100× 1100	1:57:52 (100.00%)	1:08:59 (58.52%)	31:55 (27.08%)
1300× 1300	2:50:03 (100.00%)	1:44:56 (61.71%)	48:19 (28.41%)
1600× 1600	4:14:46 (100.00%)	2:38:46 (62.32%)	1:22:50 (32.51%)

TABLE 3.1

Times spent for computation measured on the PARISC system for different grid dimensions and number of threads

grid	1 thread	2 threads	4 threads
50× 50	0:05.93 (100%)	0:05.88 (99.16%)	0:05.17 (87.18%)
100× 100	0:24.86 (100%)	0:15.47 (62.13%)	0:15.47 (62.13%)
150× 150	0:58.07 (100%)	0:34.29 (59.05%)	0:31.03 (53.43%)
200× 200	1:50.64 (100%)	1:07.57 (61.07%)	0:52.76 (47.69%)
250× 250	2:39.32 (100%)	1:32.88 (58.30%)	1:02.23 (39.06%)
300× 300	3:55.81 (100%)	2:13.84 (56.76%)	1:25.89 (36.42%)
400× 400	7:19.59 (100%)	4:05.14 (55.77%)	2:32.17 (34.62%)
500× 500	11:11.14 (100%)	6:16.87 (56.15%)	3:46.82 (33.80%)
600× 600	16:42.24 (100%)	9:15.33 (55.40%)	5:19.53 (31.88%)
700× 700	21:50.84 (100%)	12:27.02 (56.98%)	6:57.50 (31.85%)
800× 800	28:54.11 (100%)	16:18.26 (56.41%)	8:56.33 (30.93%)
900× 900	36:21.57 (100%)	20:45.55 (57.09%)	11:05.56 (30.51%)
1000× 1000	47:25.41 (100%)	26:26.64 (55.76%)	13:32.90 (28.57%)
1100× 1000	57:11.07 (100%)	31:42.46 (55.45%)	16:42.29 (29.21%)
1300× 1300	1:25:29.58 (100%)	49:13.30 (57.57%)	22:28.05 (26.28%)
1600× 1600	2:25:47.37 (100%)	1:24:32.78 (57.99%)	34:08.62 (23.42%)

TABLE 3.2

Times needed for computation measured on the Opteron system for different grid dimensions and number of threads

4. Model parameters. The functions f, G and α appearing in the anisotropic diffusion model description (see Section 2) were defined as follows

$$\begin{aligned} f(\rho) &= -\gamma\rho(\rho-0.5)(\rho-1.0), \quad \gamma = const > 0.0, \\ G(d) &= \frac{\beta}{1+d^2}, \quad \beta > 0.0, \\ \alpha(s) &= \alpha_{max}(1.0 - \frac{\delta}{1+s^2}), \quad \alpha_{max} > 0.0,\ 0.0 < \delta < 1.0. \end{aligned} \tag{4.1}$$

The parameters γ, β, α_{max}, δ influence the process of vector field pattern sharpening. For some parameter values combinations the results are not satisfactory (see Figures 4.1a, 4.1b, 4.1c).

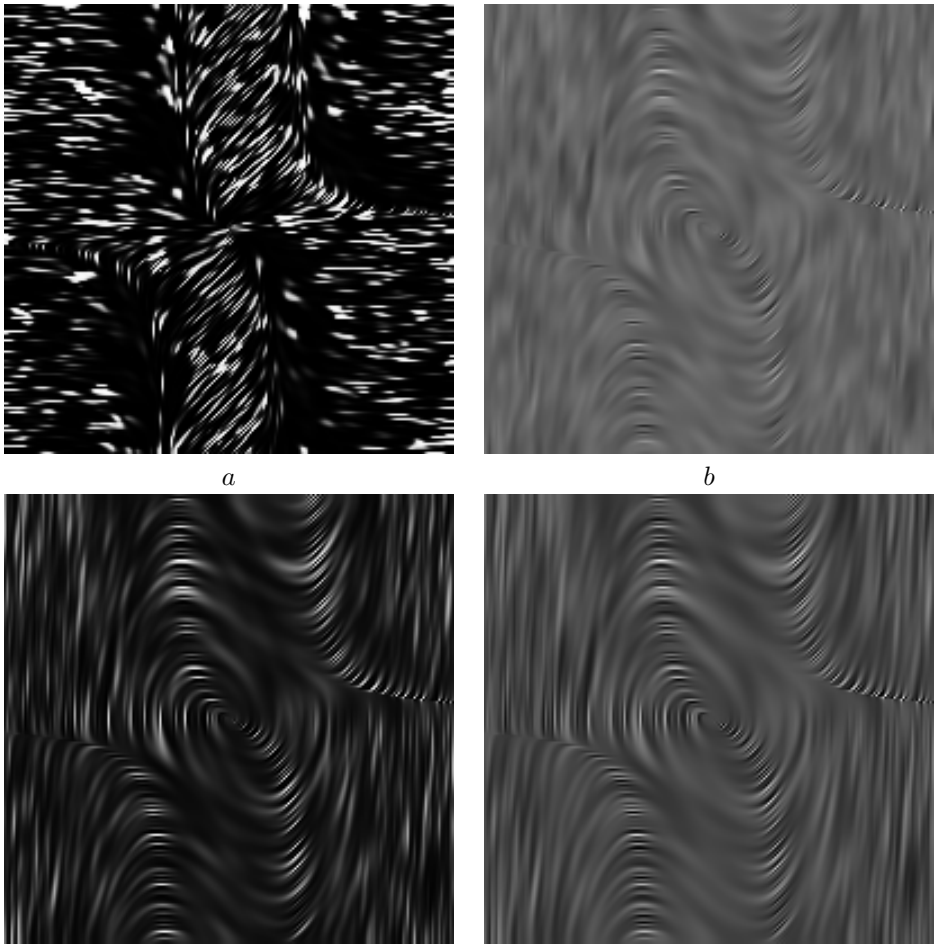


FIG. 4.1. *Results of vector field pattern sharpening for different parameters.*

The greater the parameter β is, the stronger is the diffusion in the direction orthogonal to vector field. Therefore, we must choose β relatively small in comparison with parameter α_{max} value, which influences the diffusion in vector field direction.

When this is not satisfied, we may get results like in Figures 4.1a, 4.1b. The first example shows the case, when diffusion in the orthogonal direction is much stronger than diffusion in vector field direction. In the second example, the diffusion in vector field direction is stronger, but not enough to produce sharp edges as we can see in Figure 4.1d. It is also necessary to choose parameter α_{max} value sufficiently higher than parameter γ value not to get results such as in Figure 4.1c.

To find out parameters which provide satisfactory results, such as in Figure 4.1d, a series of computations for different parameter values combinations was done. The vector field used for this test was $\vec{v} = [y, (1.0 - x^2)y - x]$, $x \in [-4, 4]$, $y \in [-4, 4]$. Having analyzed computed data, we can now recommend parameters in Table 4.1. Using these values the vector field pattern should be sharpened enough by the time $t = 2.0$. For some of our results see the next section.

parameter	recommended values
γ	0.001 - 3.001
β	0.0001
α_{max}	35 - 50
δ	0.5

TABLE 4.1
Recommended parameter values.

5. Results. Figures 5.3, 5.2 show the process of vector field pattern sharpening by solving the equation (2.1). In both cases we used $\beta = 0.0001$, $\gamma = 3.0$, $\alpha_{max} = 50.0$, $\delta = 0.5$ parameter values. We used vector fields $\vec{v} = [8.0, 0.0]$ (Figure 5.2) and Benard Convection (Figure 5.3), which can be downloaded from the web page [10]. These are visualized for comparison on Figure 5.1 by *matlab*'s built-in function *quiver*.

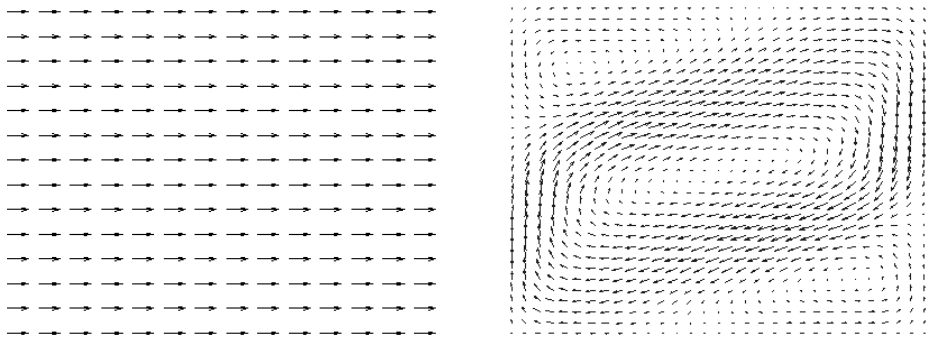
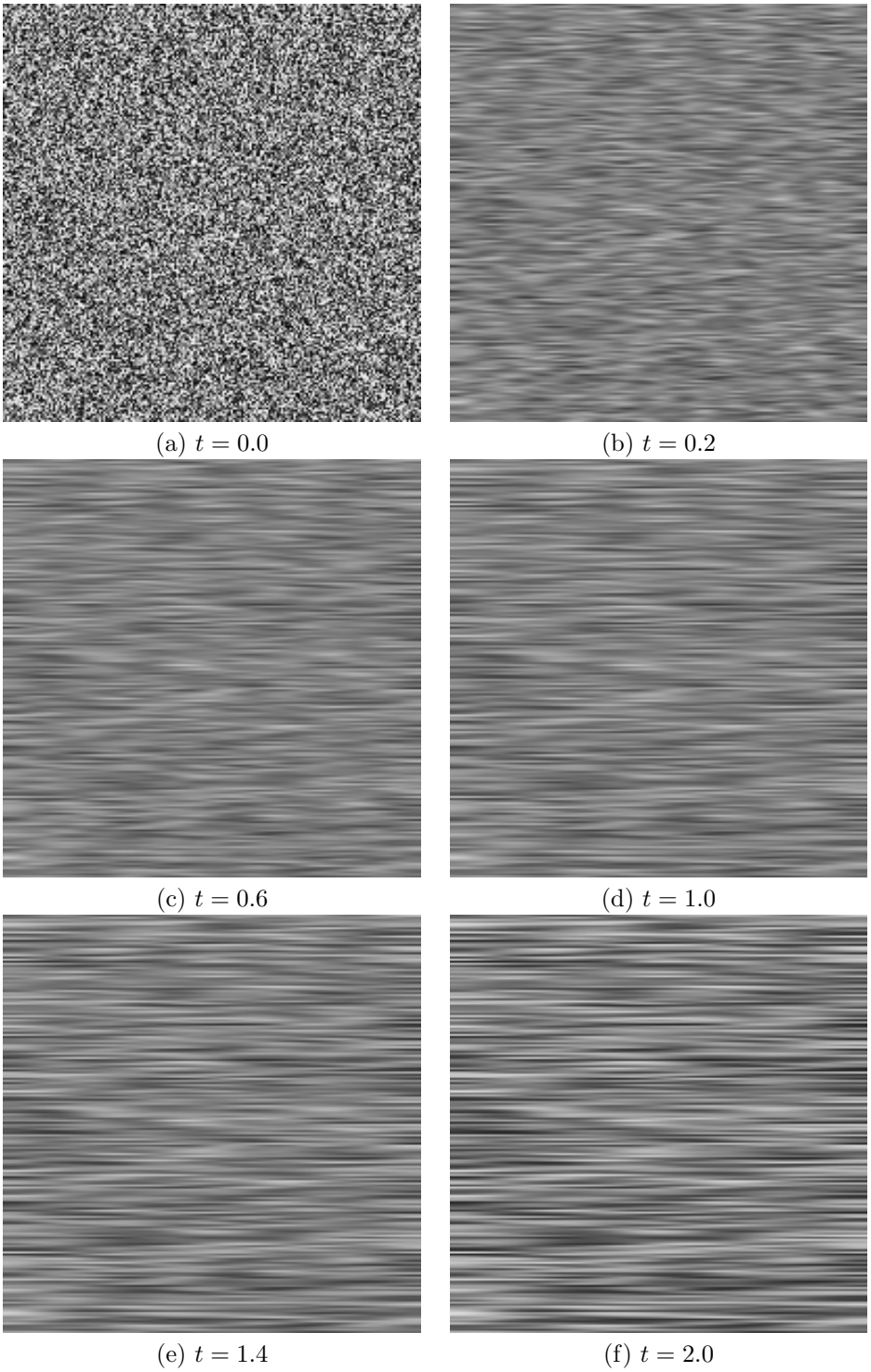


FIG. 5.1. *(a) Vector field represents a fluid motion (a) and rotation (b)*

Acknowledgement. Partial support of the project "Applied Mathematics in Physics and Technical Sciences", No. MSN6840770010 of the Ministry of Education, Youth and Sports of the Czech Republic is acknowledged.

REFERENCES

[1] M. BENEŠ, *Mathematical analysis of phase-field equations with numerically efficient coupling terms*, Interfaces and Free Boundaries **3** (2001), 201–221.

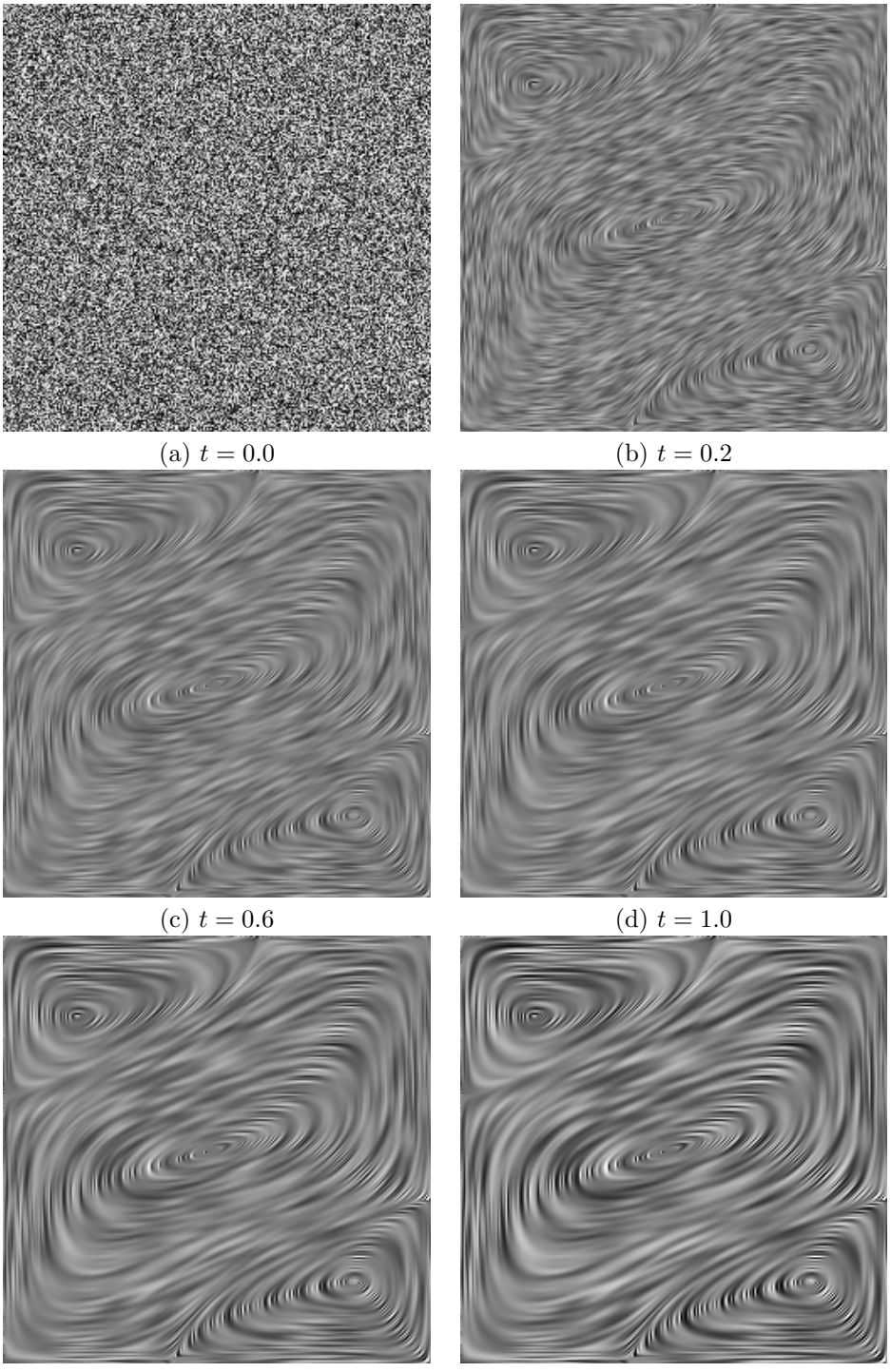


(a) $t = 0.0$ (b) $t = 0.2$

(c) $t = 0.6$ (d) $t = 1.0$

(e) $t = 1.4$ (f) $t = 2.0$

FIG. 5.2. *The process of vector field pattern sharpening by solving the problem (2.1):* $\beta = 0.0001$, $\gamma = 3.0$, $\alpha_{max} = 50.0$, $\delta = 0.5$



(a) $t = 0.0$ (b) $t = 0.2$

(c) $t = 0.6$ (d) $t = 1.0$

(e) $t = 1.4$ (f) $t = 2.0$

FIG. 5.3. *The process of vector field pattern sharpening by solving the problem (2.1):* $\beta = 0.0001$, $\gamma = 3.0$, $\alpha_{max} = 50.0$, $\delta = 0.5$

[2] M. Beneš, *Diffuse-Interface Treatment of the Anisotropic Mean-Curvature Flow*, Applications of Mathematics **48** (2003), 437–453.
[3] P. Perona and J. Malik, *Scale space and edge detection using anisotropic diffusion*, IEEE Trans. Pattern Anal. Mach. Intell. **12** (1990), 629–639.
[4] J. Kačur and K. Mikula, *Solution of nonlinear diffusion appearing in image smoothing and edge detection*, Appl. Numer. Math **17** (1995), 47–59.
[5] J. Mikyška, *Numerical Model for Simulation of Behaviour of Non-Aqueous Phase Liquids in Heterogeneous Porous Media Containing Sharp Texture Transitions*, PhD thesis, Czech technical University in Prague, FNSPE (2005).
[6] M. Petřek, *Algorithm of mathematical visualization and their use in mathematical modeling.* Diploma thesis, Dept. of Mathematics, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, 2004, (in Czech).
[7] H.-C. Hege and K. Polthier (eds), *Mathematical Visualization: Algorithms, Application and Numerics*, Springer-Verlag Berlin/Heidelberg, 1998.
[8] P. Bauer, *Mathematical Modelling and Numerical Simulation of Pollution Transport in the Atmospheric Boundary Layer,* Proceedings on the Conference Topical Problems of Fluid Mechanics, Praha (2005), 7–10.
[9] T. Preusser, M. Rumpf, Anisotropic Nonlinear Diffusion in Flow Visualization, *In IEEE Visuazliation'99*, 1999, 323–332.
[10] M. Rumpf, Research group web page, *http://numod.ins.uni-bonn.de/exports/tdFlowVis/index.html*.
[11] M. Holodniok, A. Klíč, M. Kubíček and M. Marek, *Methods of Analysis of Nonlinear Dynamical Models*, Academia Praha (1986), (in Czech).