Kshitij Patil

Diffusion generated motion by mean curvature

SFU APMA 35 by
Dr. Steven Ruuth
Spring 2022

1 Introduction

Motion by mean curvature is the art of moving a curve normal to itself with a velocity equal to some function of its curvature. There were quite a few methods developed that did this, but all were lacking in some way or the other. The table below summarises them.

Method	Idea	Good	Not good
Front tracking	Curve = discrete points	Accuracy, computational accuracy, arbitrary motion laws	Keep track of fronts that change topology, higher dimensions
Reaction diffusion	Curve = sharp front separating equilibrium states of a strong reaction, add small diffusion	Works well over shortcomings of front tracking	Computationally inefficient due to need of artificially fine grid; cannot handle self intersections, junctures
Level set	Curve = 0 level set of some function	Works well over shortcomings of front tracking and some of reaction diffusion	Cannot handle self intersections, junctures

The lack of a simple method that could move any arbitrary curve influenced by its curvature is what motivated the authors [1] to find such a method.

2 Intuition

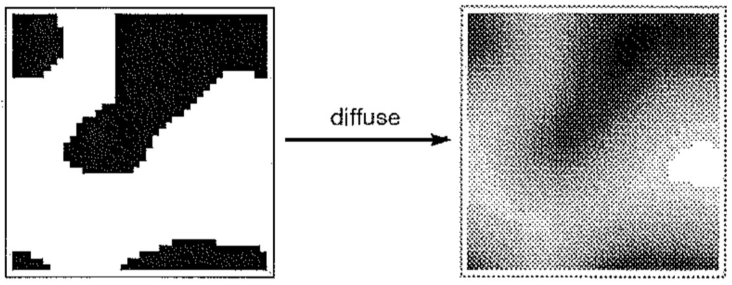

Figure 1: Diffusion of a set [1]

Suppose we diffuse the given points as shown in Fig. 1. If we are able to define the boundary of the diffused set, then we can diffuse the boundary again. Repeating this over and over causes the boundary of the curve to move. This is the leading idea behind diffusion generated by mean curvature.

3 Analysis

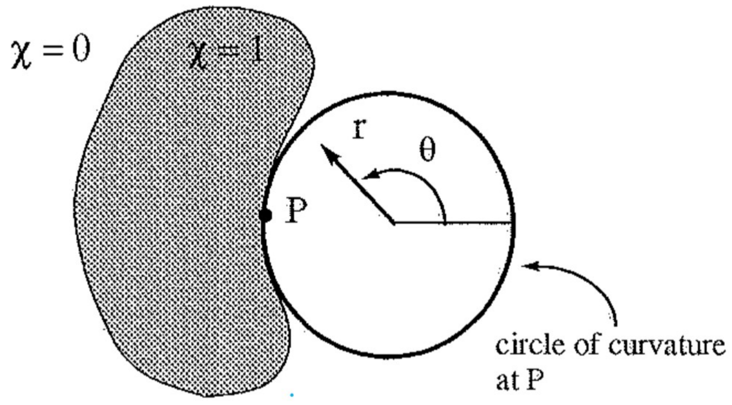

Figure 2: Diffusion of a set [1]

Consider diffusing the characteristic function χ of the set shown in Fig. 2, i.e.

$$\frac{\partial \chi}{\partial t} = D\nabla^2 \chi$$

For any point P on the boundary of the curve, let's construct a local polar coordinate system with the origin at the center of curvature of P. Since local analysis holds for a very small area around P, the change with respect to θ is negligible and so near P

$$\frac{\partial \chi}{\partial t} = \frac{D}{r}\frac{\partial \chi}{\partial r} + D\frac{\partial^2 \chi}{\partial^2 r}$$

This is an advection diffusion equation with advective velocity $\frac{D}{r}$. At P, $r = \rho = \frac{1}{\kappa}$ which means that the speed there is $D\kappa$.
So the boundary moves radially with speed $D\kappa$ while the χ is diffused radially. The diffusive part does not affect the the motion of the boundary. So the boundary moves with the advective velocity. Hence diffusion generates motion of the boundary by mean curvature.
Having a variable diffusivity coefficient $D = D(x, \chi, \nabla\chi, \Delta\chi)$ has equation $\frac{\partial \chi}{\partial t} = \nabla D \nabla \chi$ and allows for a wide class of motions.

4 Numerical method

Evolving a given curve σ is based on the following base algorithm.

```
Given:  a set with boundary $\sigma$, characteristic function $\chi$;
For:  $\chi$
       Construct $\chi(\tau)$ = diffusion of $\chi$ for time $\tau$
       Define $\chi$ as the characteristic function of the set $\{\chi(\tau) \geq 1/2\}$
       Define $\sigma$ as the boundary of this set.
Repeat;
```

the only parameter here is τ. we want it to be optimal such that the curve moves by at least one grid and also so that the local analysis does not breakdown. The speed of the curve at any given point is $D\kappa$. So the distance moved in time τ is $D\kappa\tau$. This distance should be more than grid spacing δx and smaller than the radius of curvature $\frac{1}{\kappa}$ for local analysis to hold, i.e.
$\delta x << D\kappa\tau << \frac{1}{\kappa}$ or $\frac{\delta x}{D\kappa} << \tau << \frac{1}{D\kappa^2}$ or $\frac{1}{\delta x\kappa} << \frac{\tau D}{\delta x^2} << \left(\frac{1}{\delta x\kappa}\right)^2$. If $\frac{1}{\delta x\kappa} = \frac{\rho}{\delta x}$ is large, i.e. if the grid resolves the radius of curvature, there are τ in this range.

5 Extension to arbitrary curves

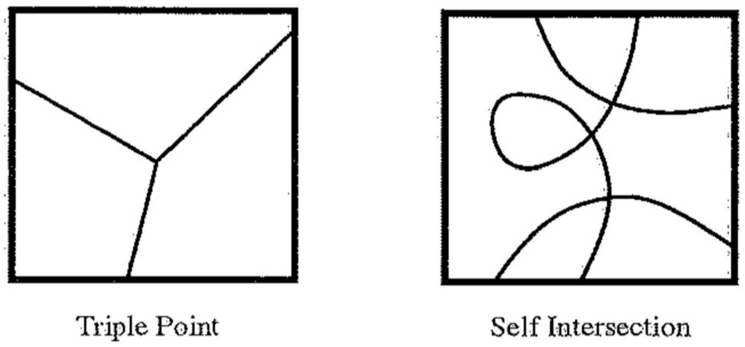

Figure 3: General curves of interest [1]

For practical use, the base algorithm needs to be modified so as to accommodate any type of curve, as shown in Fig. 3. The generalization must reduce to the base algorithm away from junctions and intersections and that it should allow different stable triple junctions. The base algorithm is a 2 region case and we can choose χ to be the characteristic function of either of them. This means that we can have 2 different χs, χ_1 and χ_2 essentially explaining the same curve. So we can move σ by diffusing either and then defining the new set as $\{\chi_1 \geq 1/2\}$ or $\{\chi_2 \geq 1/2\}$. These can be symmetrically characterized as $\{\chi_1 \geq \chi_2\}$ and $\{\chi_2 \geq \chi_1\}$. So we use both sets, diffuse them independently and then redefine them as sets where they are biggest. So the base algorithm can be extended to move a curve that divides the plane into N regions:

```
Given:  N sets that partition the plane with union of the boundaries $\sigma$,
        and characteristic functions $\{\chi_i\}_{i=1}^N$;
For:  $\{\chi_i\}_{i=1}^N$
       Construct $\chi_i(\tau) =$ diffusion of $\chi$ for time $\tau$, $i = 1, \cdots, N$
       Define $\chi_i$ as the characteristic function of the set
       $\{\chi_i(\tau) \geq \chi_j(\tau), j = 1, \cdots, N\}$
       Define $\sigma$ as the union of the boundaries of the sets represented by $\{\chi_i\}_{i=1}^N$.
Repeat;
```

Here we maximally select which set a given point lies in. Other ways of deciding where a point lies will result in different stable junction points.

6 Connection to Huygen's principle

- Huygen's principle provides a method to move a curve with a constant normal velocity.
- $\sigma = \partial\chi$, convolve with circular kernel K. Advance set as $\{\chi * K > 1/2\}$ to get diffusion generated motion by mean curvature.
- For any α, the set $\{\chi * K > \alpha\}$ gives a different type of motion. α is the fraction of circle area that lies on one side of the curve. For further reading, see [2].

7 Related work

In my opinion, this method is important not only because it overcame drawbacks of methods that existed at the time but also because it gave way to further improvements and modifications.

- In [3] a new, spectral discretization of these diffusion-generated methods is introduced which obtains greatly improved efficiency over the usual finite difference approach.
- In [4] the authors generalize diffusion-generated motion to a procedure that can be applied to objects of any dimension k inside of $\mathbb{R}^d, k < d$.
- In [5] the authors have shown that diffusion generated motion does converge to motion by mean curvature.

8 Implementation details

The grid is square of size M and uniform. Forward differences in time and centered in space. All schemes are explicit in time. It could be interesting to try out time implicit/alternating direction methods which are known to be stable.

- Constant diffusivity coefficient: We want to evolve: $\frac{\partial\chi}{\partial t} = D\nabla^2\chi$.
  $\frac{\chi^{n+1}_{i,j}-\chi^n_{i,j}}{dt} = D\frac{\chi^n_{i+1,j}+\chi^n_{i-1,j}-4\chi^n_{i,j}+\chi^n_{i,j+1}+\chi^n_{i,j-1}}{dx^2}$ or $\chi^{n+1}_{i,j} = \chi^n_{i,j}+dt*D\frac{\chi^n_{i+1,j}+\chi^n_{i-1,j}-4\chi^n_{i,j}+\chi^n_{i,j+1}+\chi^n_{i,j-1}}{dx^2}$

  The dicretized Laplacian forms the 5-point stencil $L_1 := \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

  I have also tried out 2 different 9-point stencils $L_2 := \begin{pmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & -3 & 0.5 \\ 0.25 & 0.5 & 0.25 \end{pmatrix}$ and $L_3 := \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

- Space dependent diffusivity coefficient: We want to evolve: $\frac{\partial\chi}{\partial t} = \nabla D(x,y)\nabla\chi$.
  $\frac{\chi^{n+1}_{i,j}-\chi^n_{i,j}}{dt} = (D_x, D_y)^T\left(\frac{\chi^n_{i+1,j}-\chi^n_{i-1,j}}{2dx}, \frac{\chi^n_{i,j+1}-\chi^n_{i-1,j}}{2dy}\right) = D_x\frac{\chi^n_{i+1,j}-\chi^n_{i-1,j}}{2dx} + D_y\frac{\chi^n_{i,j+1}-\chi^n_{i-1,j}}{2dx}$

$\chi_{i,j}^{n+1} = \chi_{i,j}^{n} + dt * D_x \frac{\chi_{i+1,j}^{n} - \chi_{i-1,j}^{n}}{2dx} + dt * D_y \frac{\chi_{i,j+1}^{n} - \chi_{i-1,j}^{n}}{2dx}$

Coefficients I've tried out:

1. $D_1(x, y) = a + cos(r), r = \sqrt{x^2 + y^2}, D_{1k} = \frac{-sin(r)k}{r}, k = x, y, a \in \mathbb{R}$
2. $D_2(x, y) = sin(x), D_{2x} = cos(x), D_{2y} = 0$
3. $D_3(x, y) = cos(y), D_{3x} = 0, D_{3y} = -sin(y)$
4. $D_4(x, y) = (x + y)(\log(x + y) - 1), D_{4k} = \log(x + y), k = x, y$
5. $D_5(x, y) = \log(x) + log(y), D_{5x} = 1/x, D_{5y} = 1/y$

It's interesting to note that we don't use D anywhere, just it's change in the spatial directions.

The schemes above are for non-boundary points, i.e. $2 \leq i, j \leq M - 1$. There is a reflective boundary condition, $\chi_{1,j} = \chi_{3,j}$; $\chi_{i,1} = \chi_{i,3}$; $\chi_{M,j} = \chi_{M-2,j}$ and $\chi_{i,M} = \chi_{i,M-2}$.
At every step, each region undergoes diffusion for a number of steps (25, can be changed in the code) before comparing intersections.
Based on the curve, I have also set a threshold which treats absolute values below it as 0. This is because at some grid points values like $10^{-k}, k > 7$ are identified as non-zero and this causes problems while comparing as the boundary has not really diffused to these grid points. For example, suppose that the actual last grid point has value 0.0025 and the point right next to it has value in the order of 10^{-k}. Clearly the boundary has negligible reach there. In this case the threshold would be 10^{-3}.
Reducing dt below a certain point relative to dx and D (constant case) makes the curve stand still as expected. Comparisons with level set method are only possible for the circle and square images. It was rather cumbersome to find a suitable function for the square. I prefer diffusion generated motion.
The code is at https://github.com/k37pi/Diffusion-generated-motion

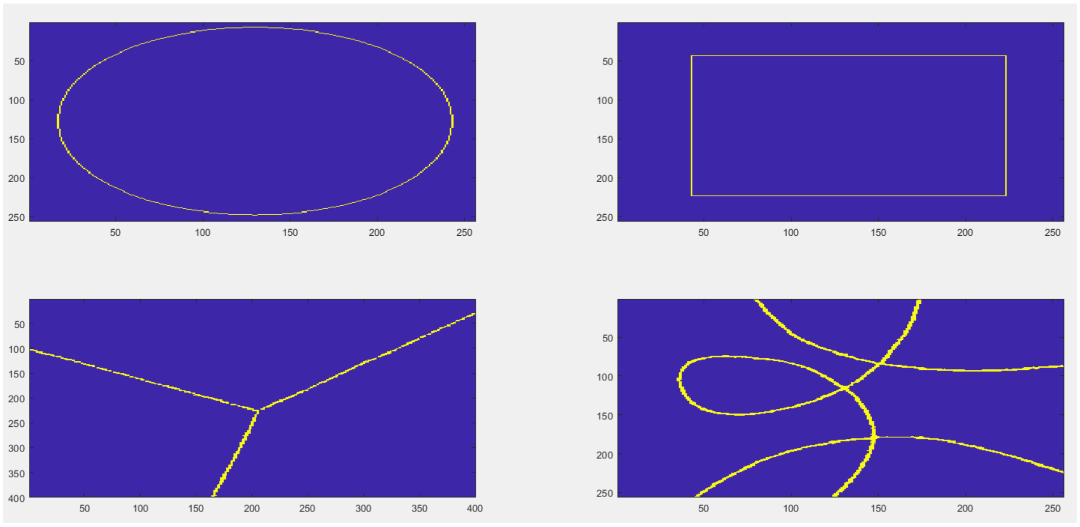

Figure 4: Initial configurations

9 Examples

9.1 Circle, Square (2 regions)

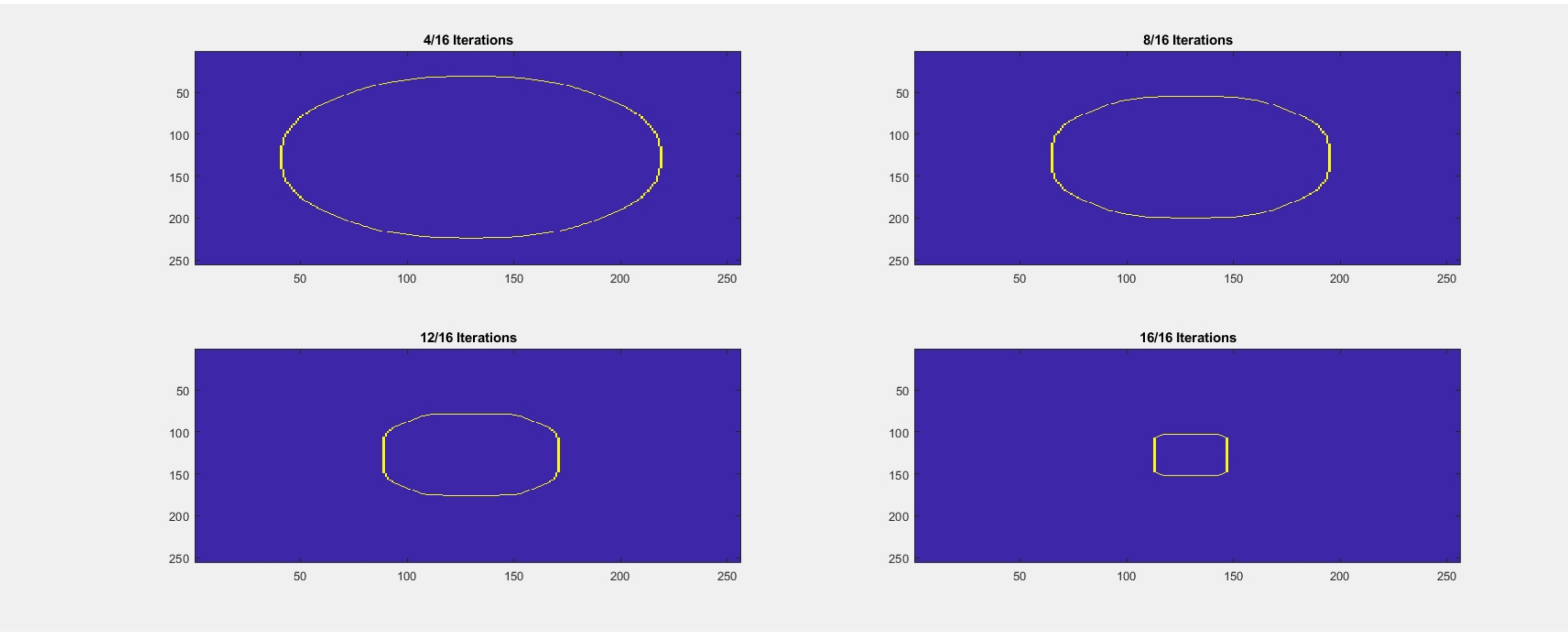

Figure 5: $dx = 0.9$, $dt = 0.5$, $D = 0.1$, stencil L_1

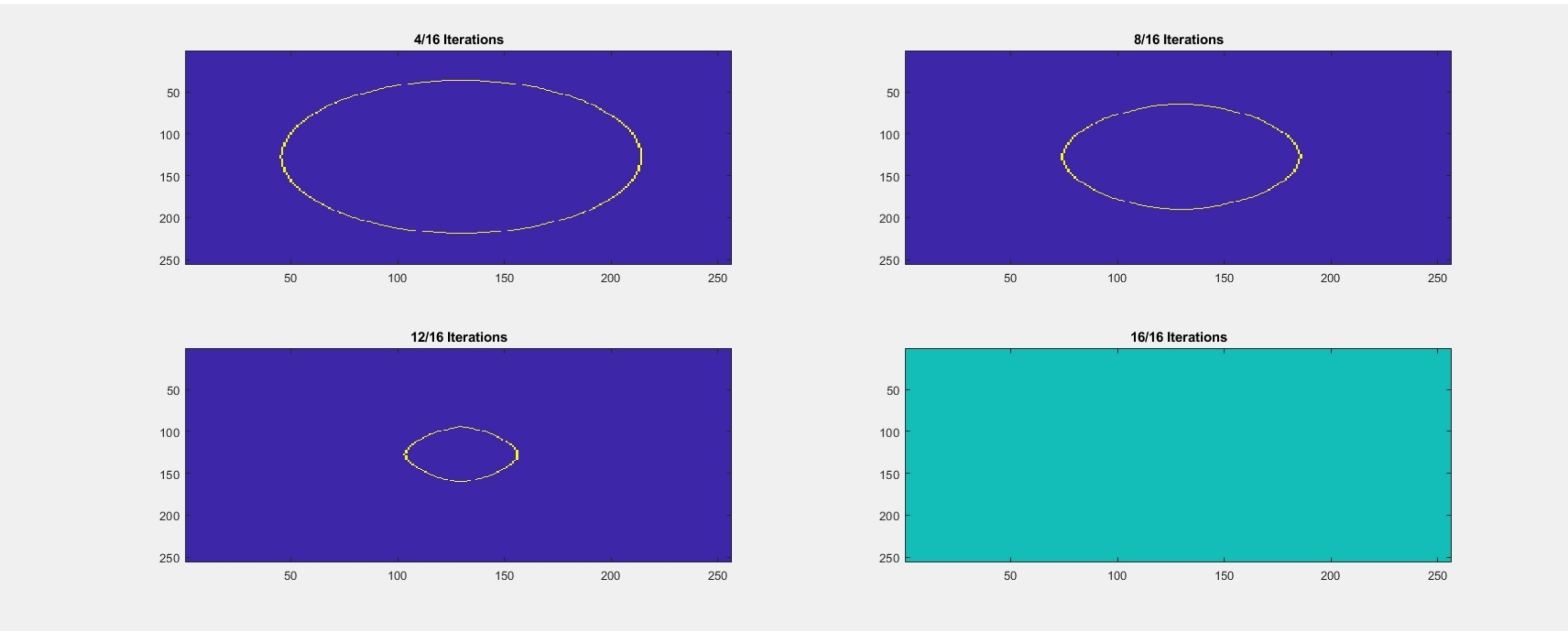

Figure 6: $dx = 0.9$, $dt = 0.5$, $D = 0.1$, stencil L_3

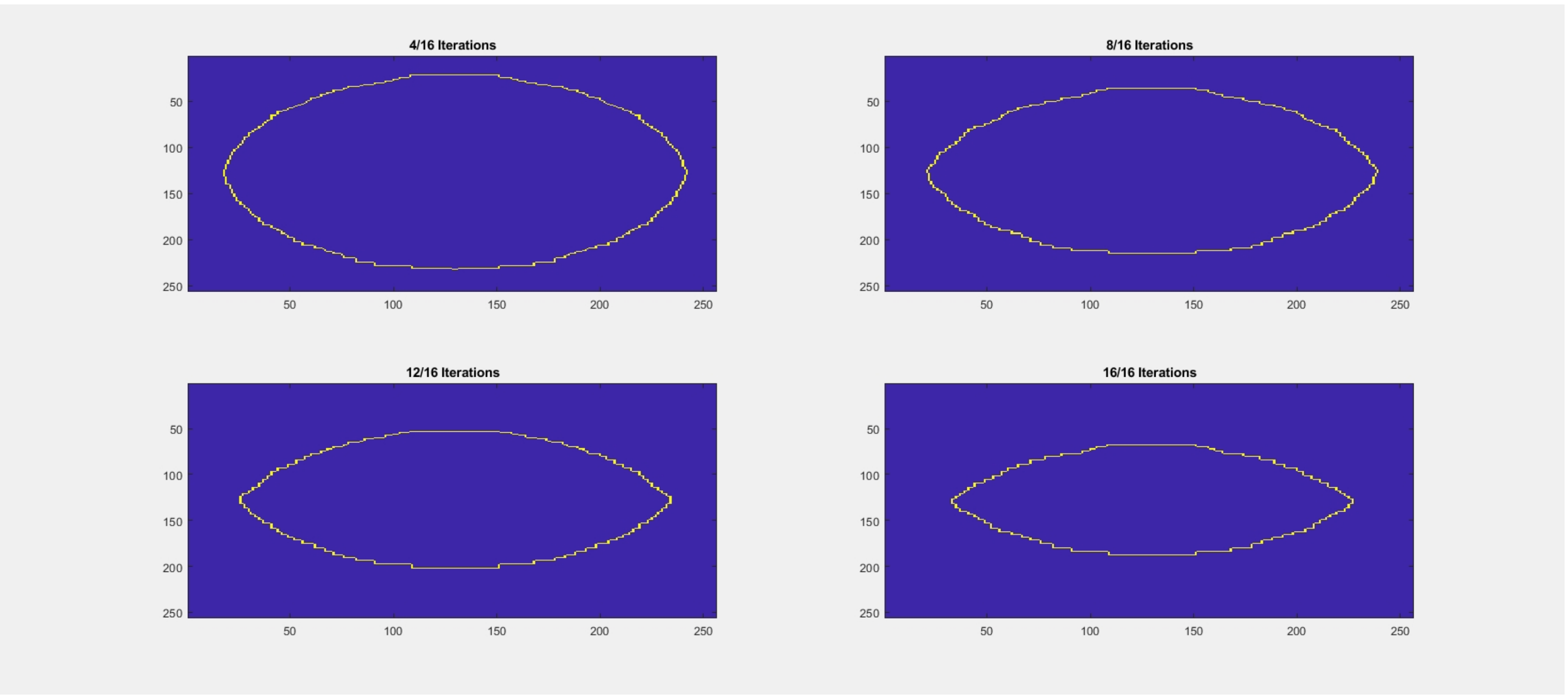

Figure 7: $dx = 0.7,\ dt = 0.05,\ D = D_2$

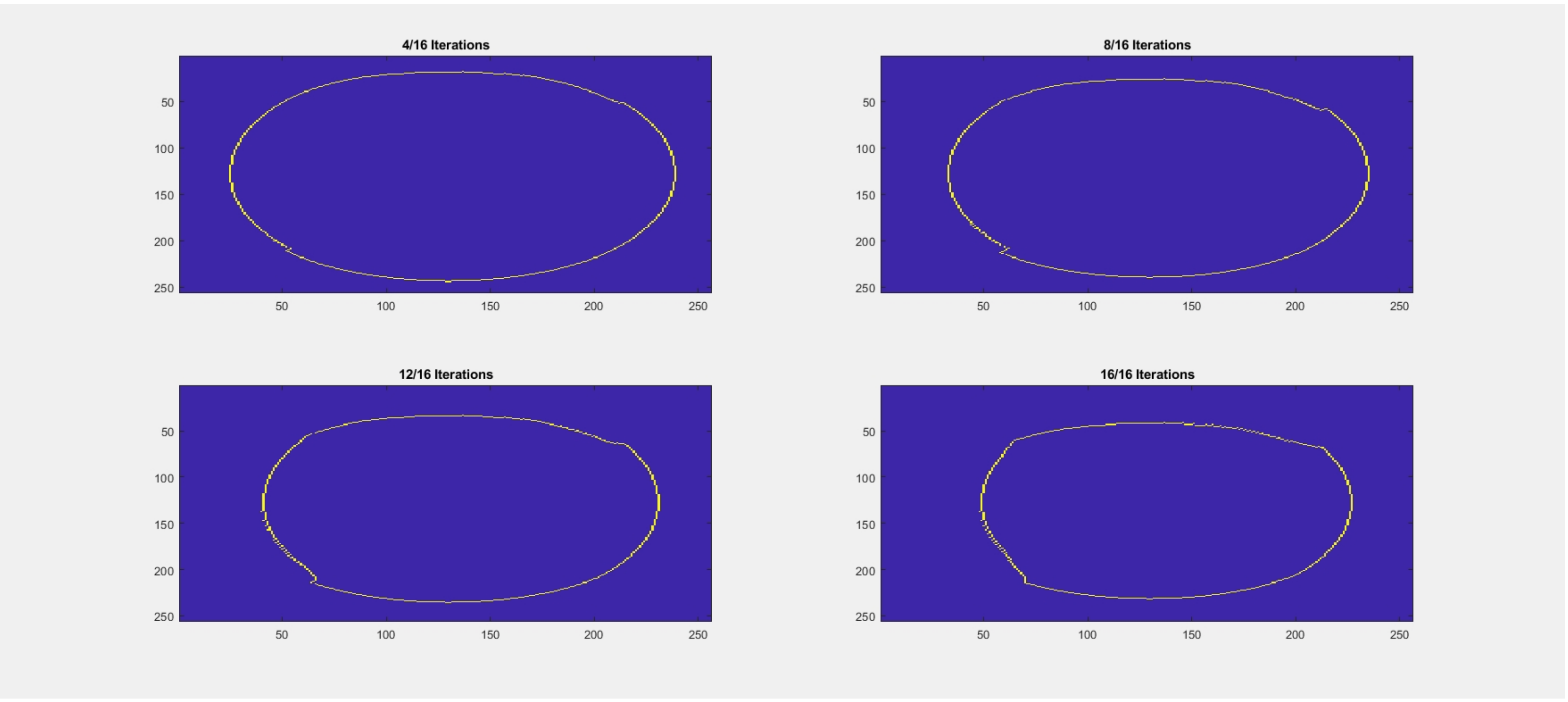

Figure 8: $dx = 0.9,\ dt = 0.05,\ D = D_5$

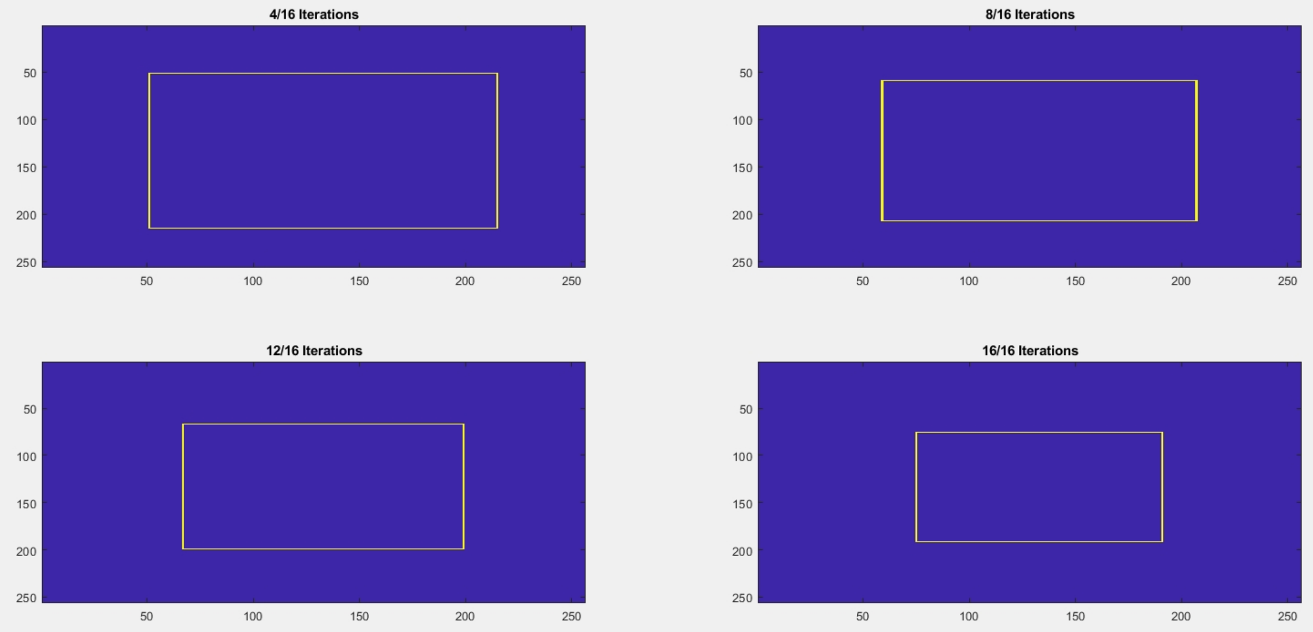

Figure 9: $dx = 0.5$, $dt = 0.001$, $D = 0.5$, stencil L_1

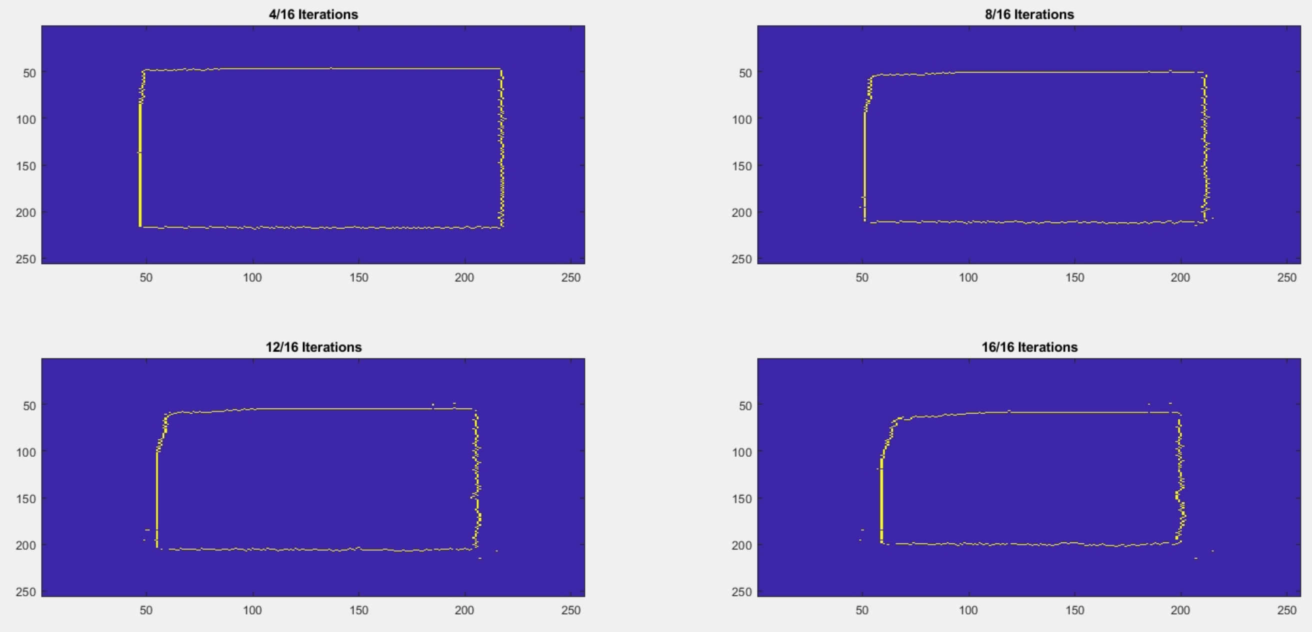

Figure 10: $dx = 0.5$, $dt = 0.0001$, $D = D_1$

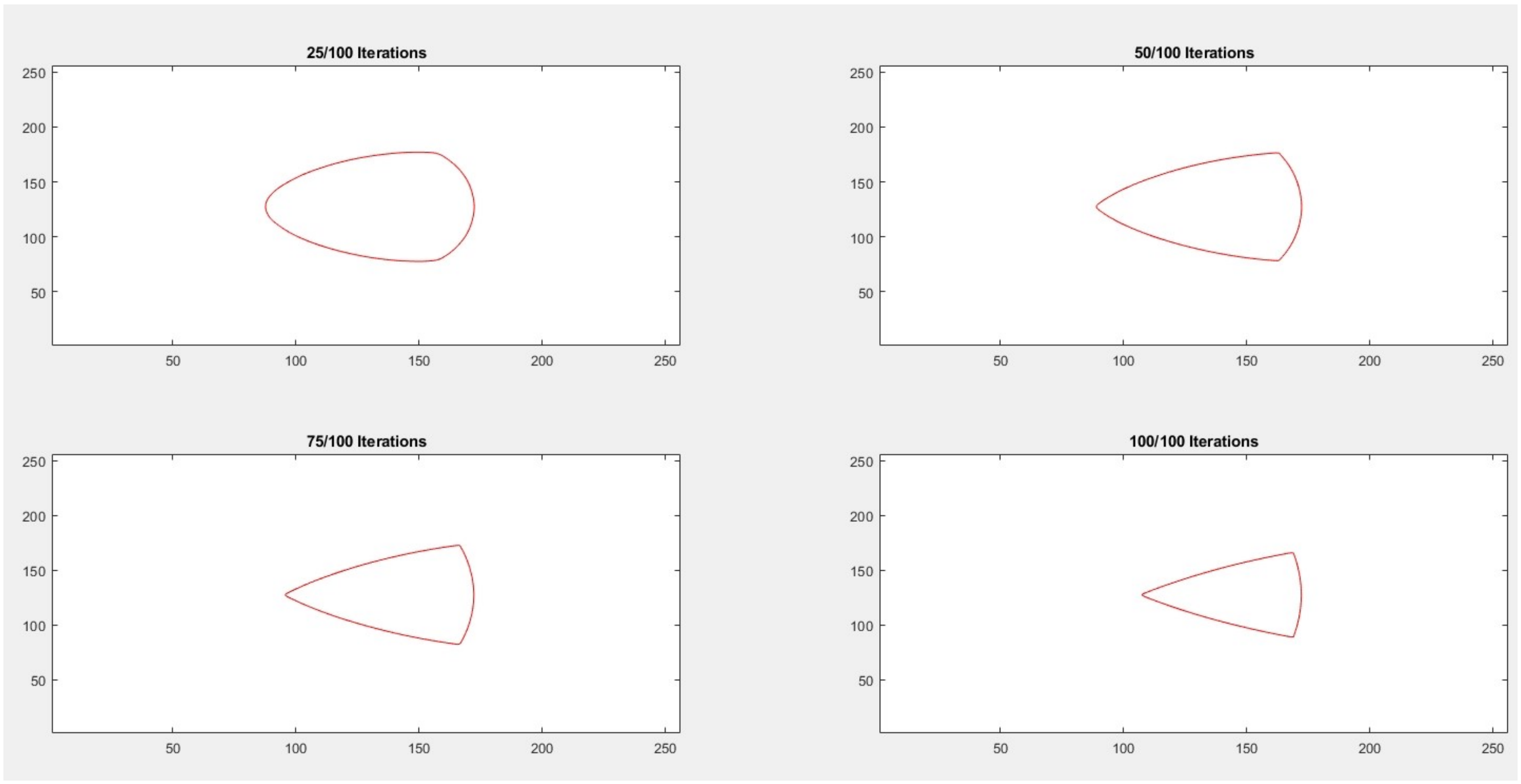

Figure 11: Circle level set

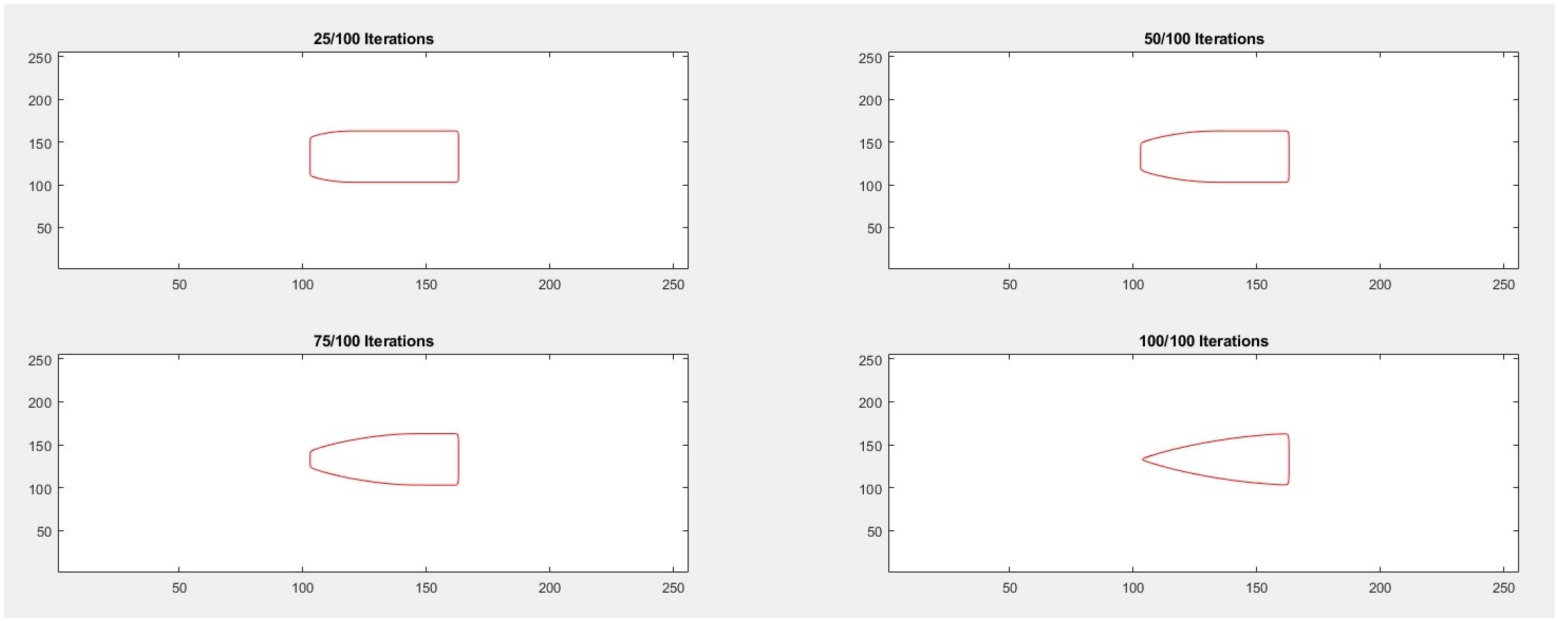

Figure 12: Square level set

9.2 Triple point (3 regions)

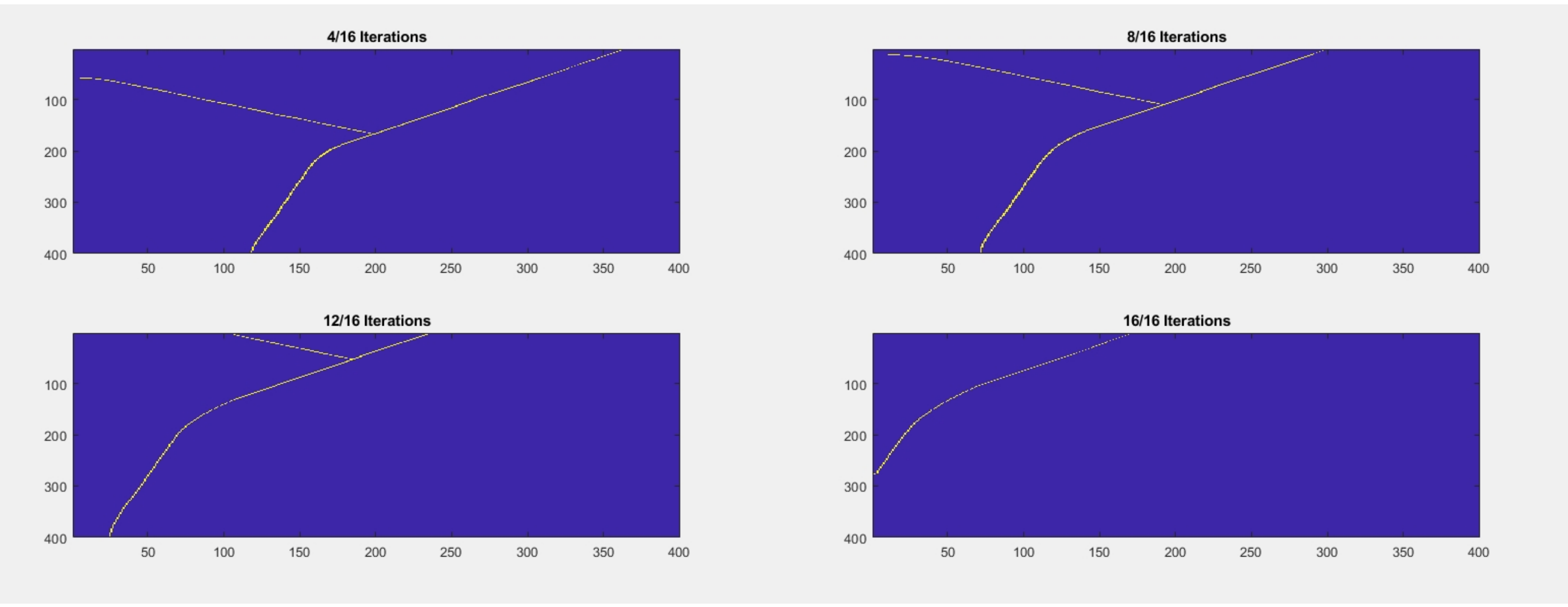

Figure 13: $dx = 0.5$, $dt = 0.5$, $D = 0.1$, stencil L_1

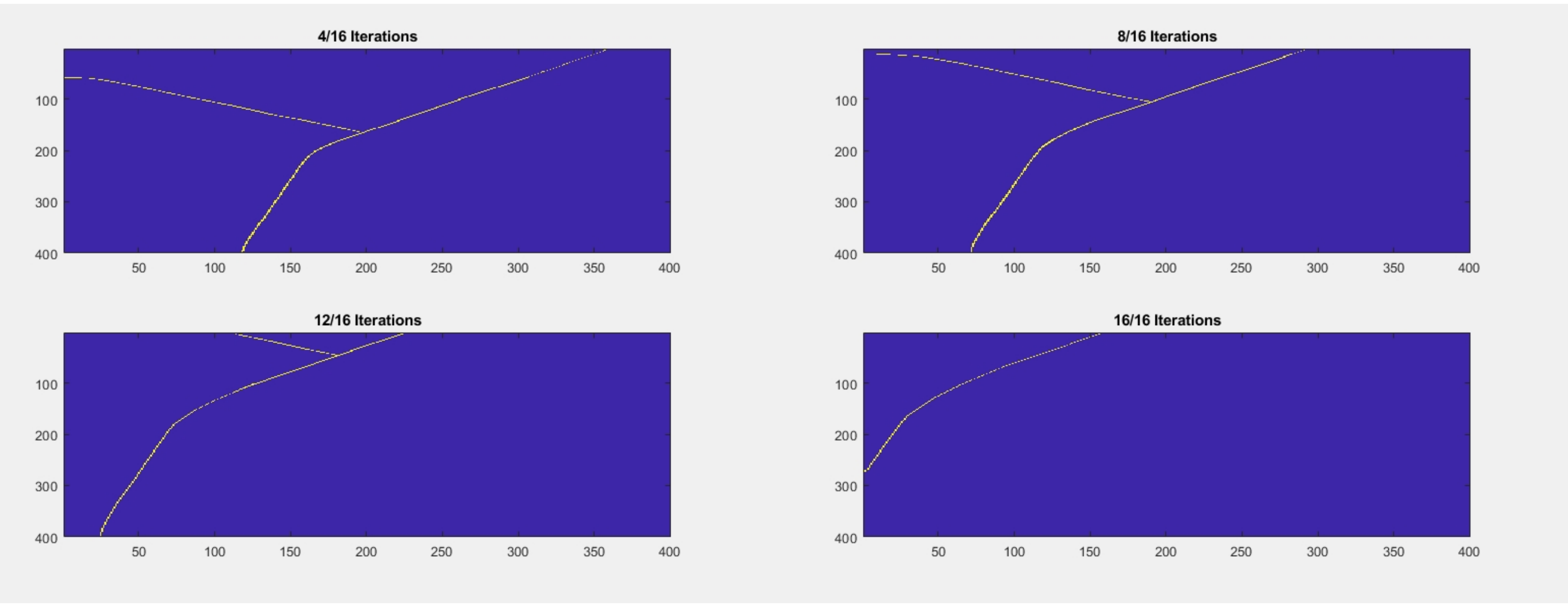

Figure 14: $dx = 0.5$, $dt = 0.5$, $D = 0.1$, stencil L_2

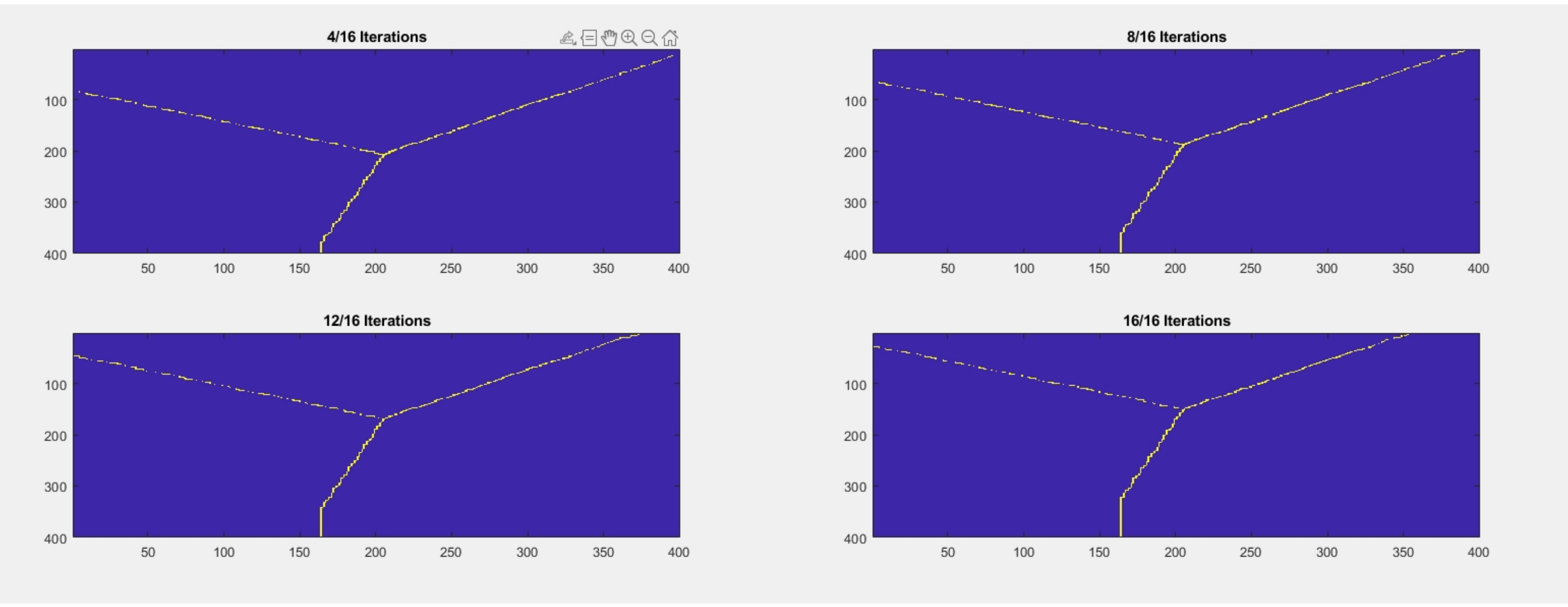

Figure 15: $dx = 0.5$, $dt = 0.01$, $D = D_2$

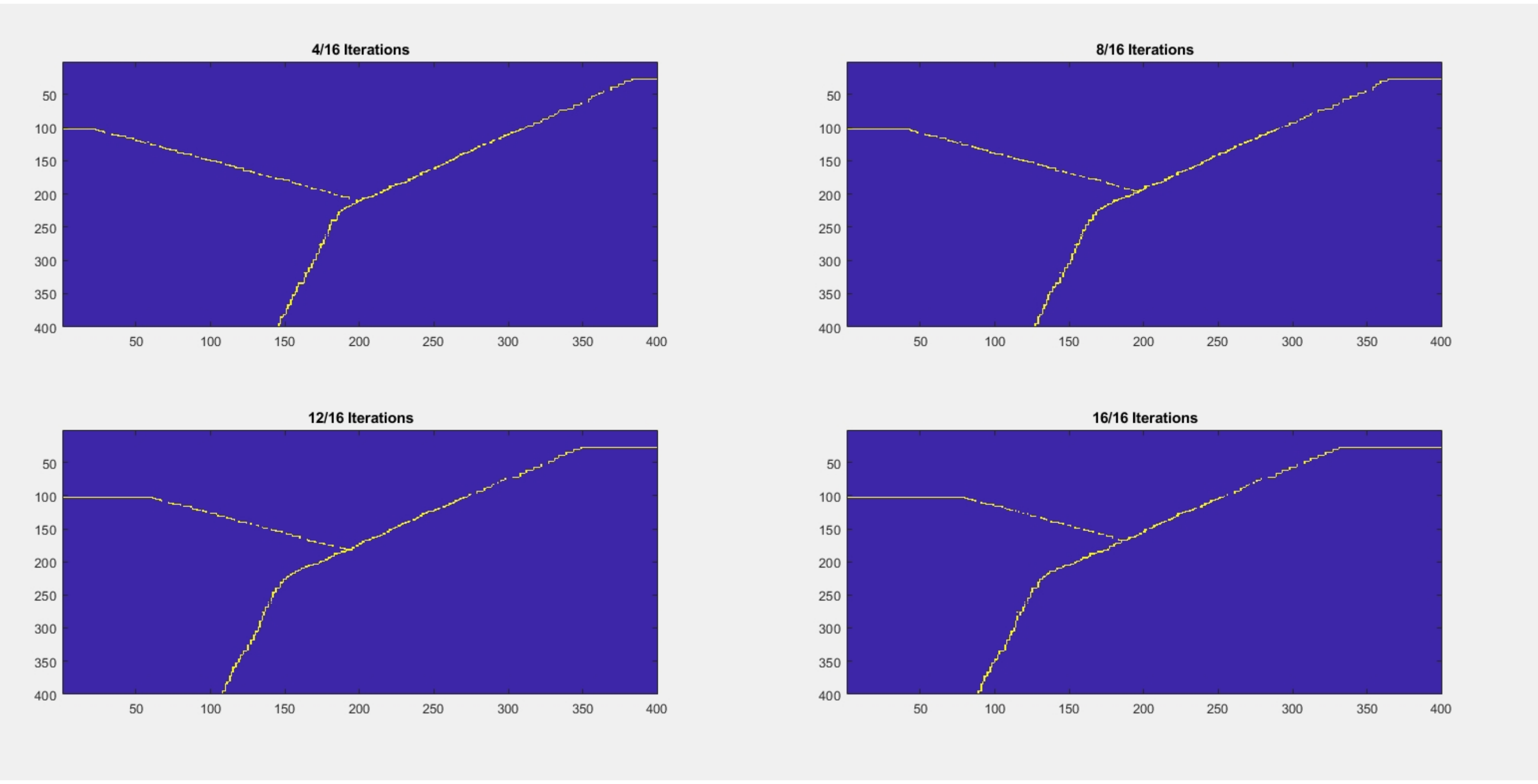

Figure 16: $dx = 0.5$, $dt = 0.01$, $D = D_3$

9.3 Self intersection (7 regions)

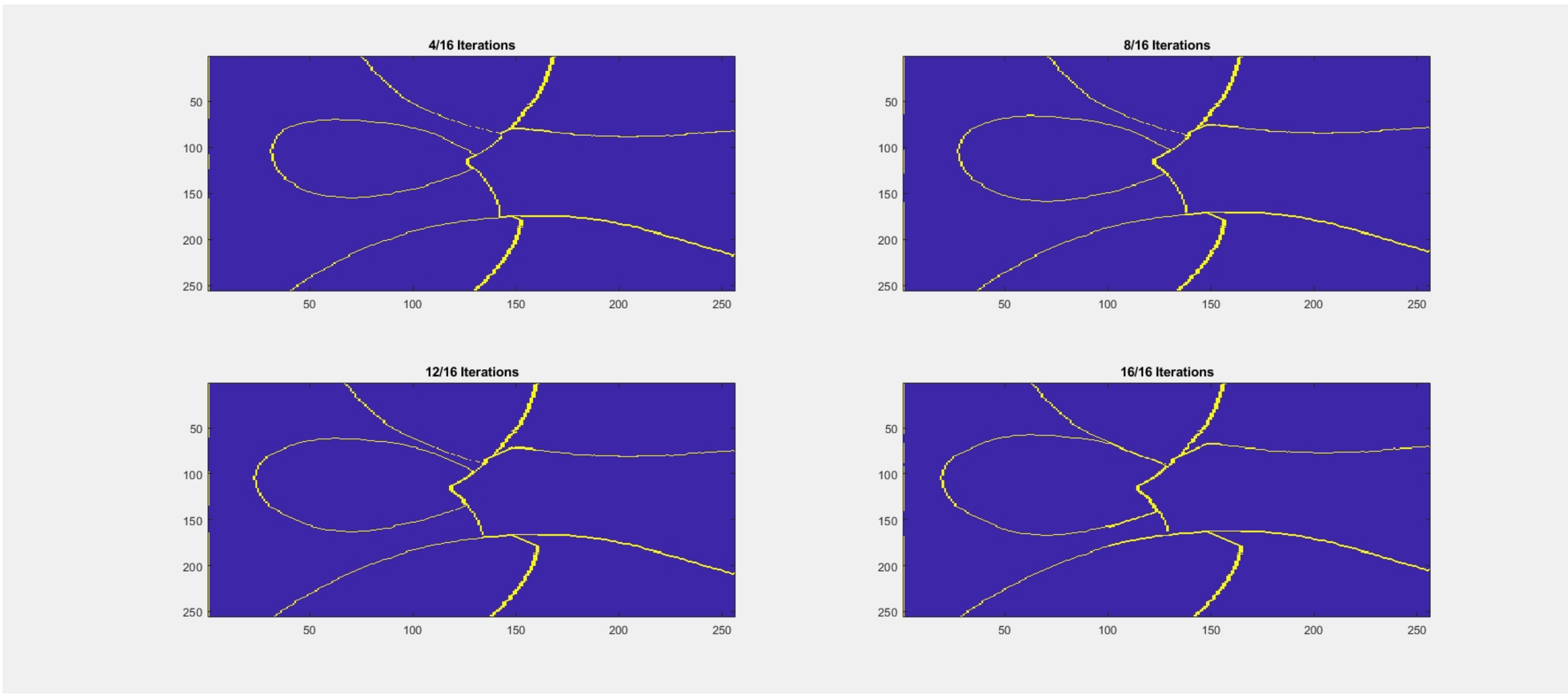

Figure 17: $dx = 0.5$, $dt = 0.001$, $D = 0.1$, stencil L_1

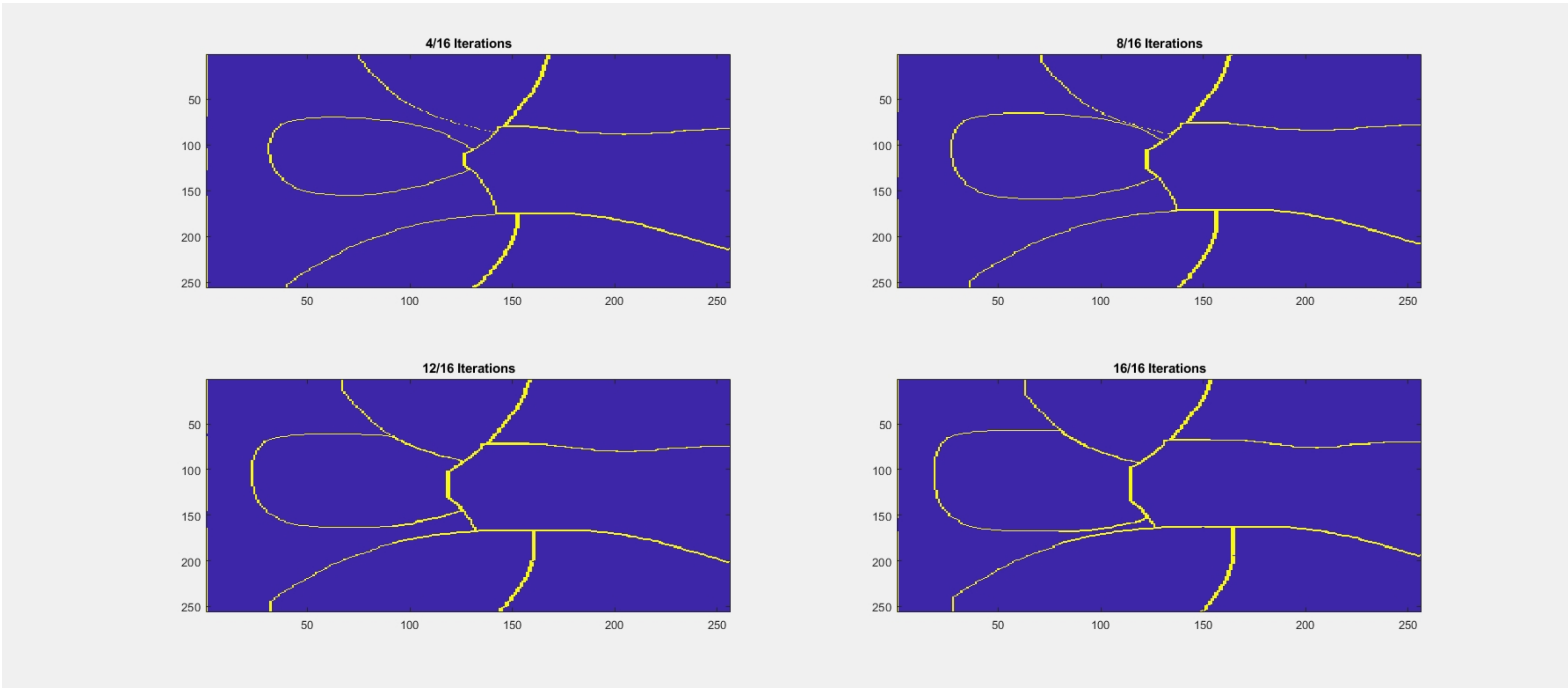

Figure 18: $dx = 0.5$, $dt = 0.001$, $D = 0.1$, stencil L_3

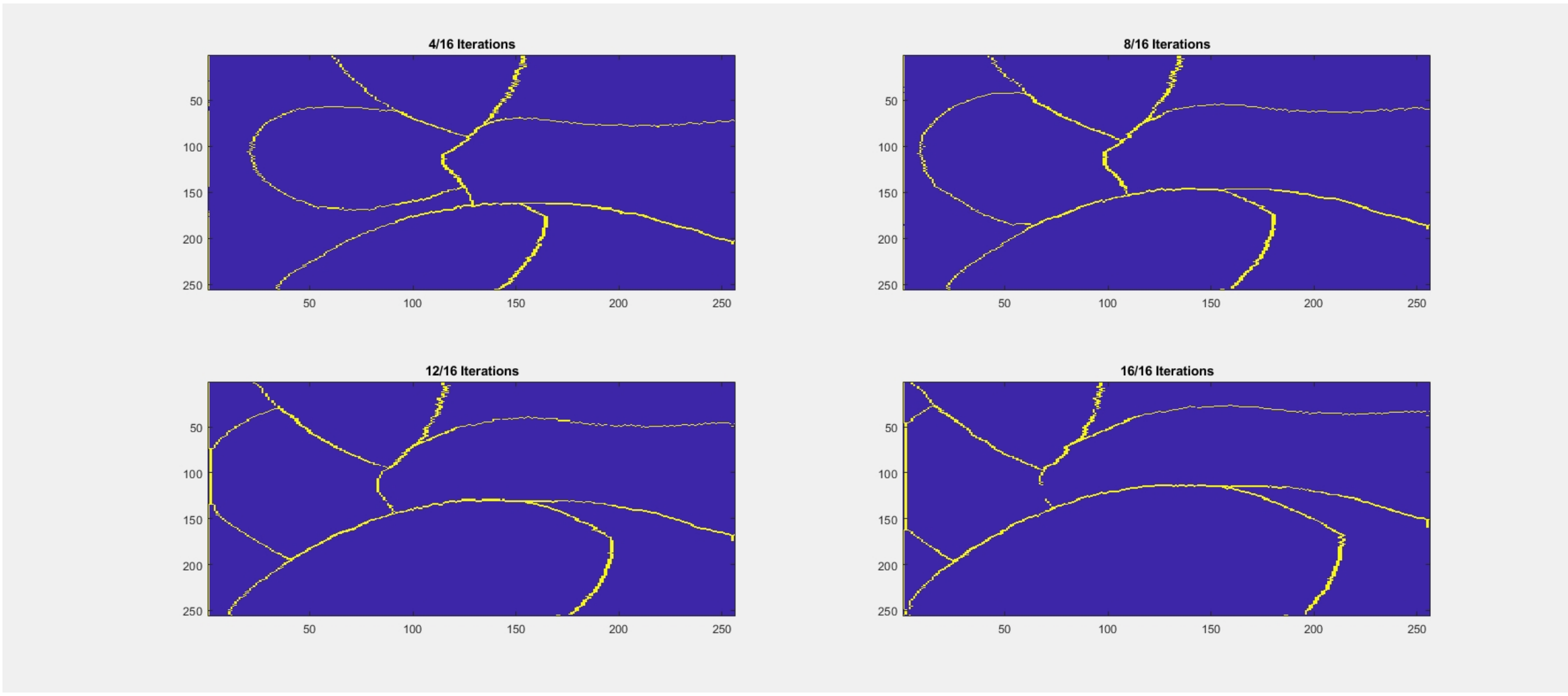

Figure 19: $dx = 0.5$, $dt = 0.01$, $D = D_1$

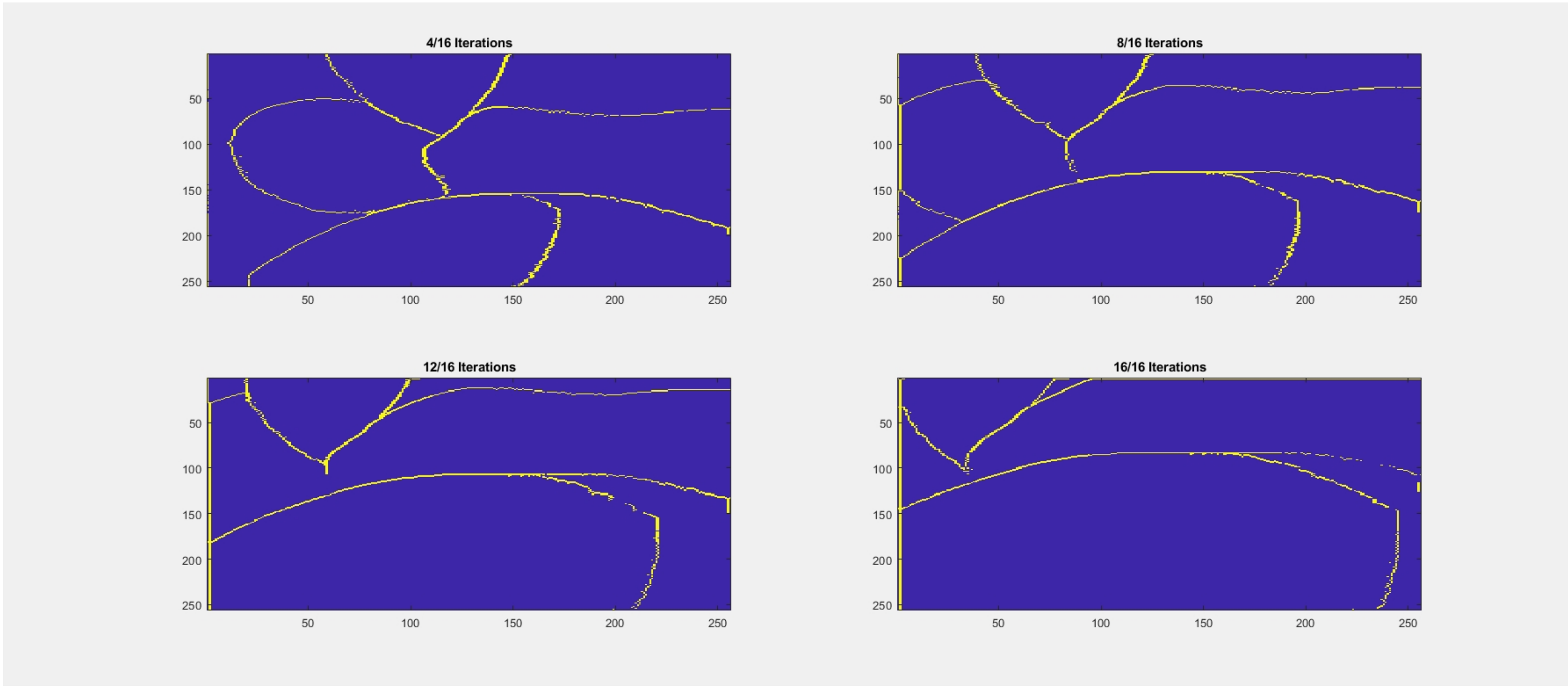

Figure 20: $dx = 0.5$, $dt = 0.01$, $D = D_4$

References

[1] B. Merriman, J. Bence, and S. Osher. Diffusion generated motion by mean curvature. In J.E. Taylor, editor, Computational Crystal Growers Workshop, pages 73–83. American Mathematical Society, Providence, Rhode Island, 1992. Also available as UCLA CAM Report 92-18, April 1992. `https://secure.math.ucla.edu/cam/camreport/cam92-18.pdf`

[2] Merriman, Barry and Steven J. Ruuth. "Convolution-Generated Motion and Generalized Huygens' Principles for Interface Motion." SIAM J. Appl. Math. 60 (2000): 868-890.

[3] Steven J. Ruuth, Efficient Algorithms for Diffusion-Generated Motion by Mean Curvature, Journal of Computational Physics, Volume 144, Issue 2, 1998, Pages 603-625, ISSN 0021-9991, https://doi.org/10.1006/jcph.1998.6025.

[4] Ruuth, S., Merriman, B., Xin, J. et al. Diffusion-Generated Motion by Mean Curvature for Filaments. J. Nonlinear Sci. 11, 473–493 (2001). https://doi.org/10.1007/s00332-001-0404-x

[5] Drew Swartz Nung Kwan Yip (2017) Convergence of diffusion generated motion to motion by mean curvature, Communications in Partial Differential Equations, 42:10, 1598-1643, DOI: 10.1080/03605302.2017.1383418