Mean curvature flow

Yifan Zhang

March 14 2024

1 Introduction

Mean curvature flow describes the deformation of a curve with respect to its curvature. It is similar to the heat equation in terms of smoothening out peaks and valley. And it indeed finds application in understanding the deformation of cell membranes and minimal surfaces.
The PDE could be derived by taking the time derivative of a level set function

$$\begin{aligned} \phi(x(t), t) &= C \\ \Rightarrow \nabla\phi \cdot x'(t) + \partial_t \phi &= 0 \end{aligned}$$

and setting the velocity in the normal direction to be its curvature.

$$\begin{aligned} x'(t) \cdot n &= -\kappa \\ \Rightarrow x'(t) &= -\kappa \cdot n \\ &= -\left(\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}\right) \cdot \frac{\nabla\phi}{|\nabla\phi|} \\ \Rightarrow \partial_t \phi &= -\nabla\phi \cdot x'(t) \\ &= \left(\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}\right) \cdot |\nabla\phi| \end{aligned}$$

2 Scheme Derivation

To write down the implicit explicit scheme, we define

$$\nabla\phi = (\phi_x, \phi_y)$$

then we can write down terms in the right hand side step-by-step

$$\begin{aligned} |\nabla\phi| &= (\phi_x^2 + \phi_y^2)^{1/2} \\ \frac{\nabla\phi}{|\nabla\phi|} &= \left(\frac{\phi_x}{(\phi_x^2 + \phi_y^2)^{1/2}}, \frac{\phi_y}{(\phi_x^2 + \phi_y^2)^{1/2}}\right) \\ \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} &= \frac{\phi_{xx} \cdot (\phi_x^2 + \phi_y^2)^{1/2} - \phi_x \cdot \frac{\partial}{\partial x}(\phi_x^2 + \phi_y^2)^{1/2} + \phi_{yy} \cdot (\phi_x^2 + \phi_y^2)^{1/2} - \phi_y \cdot \frac{\partial}{\partial y}(\phi_x^2 + \phi_y^2)^{1/2}}{\phi_x^2 + \phi_y^2} \end{aligned}$$

note that

$$\frac{\partial}{\partial x}(\phi_x^2+\phi_y^2)^{1/2}=\frac{1}{2}\cdot(\phi_x^2+\phi_y^2)^{-1/2}\cdot(2\phi_x\phi_{xx}+2\phi_y\phi_{xy})$$
$$\frac{\partial}{\partial y}(\phi_x^2+\phi_y^2)^{1/2}=\frac{1}{2}\cdot(\phi_x^2+\phi_y^2)^{-1/2}\cdot(2\phi_x\phi_{xy}+2\phi_y\phi_{yy})$$

we obtain

$$\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}=\frac{(\phi_{xx}+\phi_{yy})\cdot(\phi_x^2+\phi_y^2)^{1/2}-(\phi_x^2+\phi_y^2)^{-1/2}\cdot(\phi_x^2\phi_{xx}+2\phi_x\phi_y\phi_{xy}+\phi_y^2\phi_{yy})}{\phi_x^2+\phi_y^2}$$

Thus

$$\begin{aligned}\partial_t\phi&=\left(\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}\right)\cdot|\nabla\phi|\\&=\phi_{xx}+\phi_{yy}-\frac{\phi_x^2\phi_{xx}+2\phi_x\phi_y\phi_{xy}+\phi_y^2\phi_{yy}}{\phi_x^2+\phi_y^2}\\\Rightarrow\partial_t\phi-\phi_{xx}-\phi_{yy}&=-\frac{\phi_x^2\phi_{xx}+2\phi_x\phi_y\phi_{xy}+\phi_y^2\phi_{yy}}{\phi_x^2+\phi_y^2}\end{aligned}\tag{1}$$

For the linear part, we need to define

$$\phi(x,y):=\sum a_{m,n}\cos(nx)T_m(y)-b_{m,n}\sin(nx)T_m(y)$$
$$\phi_y(x,y):=\sum a'_{m,n}\cos(nx)T_m(y)-b'_{m,n}\sin(nx)T_m(y)$$

we can write down the first order formula as

$$\partial_y\phi-\phi_y=0$$
$$\partial_t\phi+\phi_{xx}+\partial_y\phi_y=\frac{\phi_x^2\phi_{xx}+2\phi_x\phi_y\phi_{xy}+\phi_y^2\phi_{yy}}{\phi_x^2+\phi_y^2}$$

In terms of Fourier and Chebyshev basis:

$$\sum a_{m,n}\cos(nx)T'_m(y)-b_{m,n}\sin(nx)T'_m(y)-a'_{m,n}\cos(nx)T_m(y)+b'_{m,n}\sin(nx)T_m(y)=0$$
$$\partial_t\phi+\sum -n^2a_{m,n}\cos(nx)T_m(y)+n^2b_{m,n}\sin(nx)T_m(y)+a'_{m,n}\cos(nx)T'_m(y)-b'_{m,n}\sin(nx)T'_m(y)=R.H.S.$$

Putting into vector form with Fourier and Chebyshev basis and transform into U basis:

$$Da_{m,n}-Ca'_{m,n}+\tau_1(x)U_{N-1}(y)=0$$
$$Db_{m,n}-Cb'_{m,n}+\tilde{\tau}_1(x)U_{N-1}(y)=0$$
$$C\partial_t\phi-n^2Ca_{m,n}+Da'_{m,n}+\tau_2(x)U_{N-1}(y)=C@R.H.S.$$
$$C\partial_t\phi-n^2Cb_{m,n}+Db'_{m,n}+\tilde{\tau}_2(x)U_{N-1}(y)=C@R.H.S.$$

We have four $\tau_1, \tilde{\tau_1}, \tau_2, \tilde{\tau_2}$, so we require four boundary conditions $u(\pm 1) = 0, \partial_y u(\pm 1) = 0$.
In summary, we arrive with an implicit-explicit scheme

$$M\partial_t X + LX = F(X)$$

where X is the state vector containing coefficients $a_{m,n}, b_{m,n}, a'_{m,n}, b'_{m,n}, \tau_1, \tilde{\tau}_1, \tau_2, \tilde{\tau}_2$.
M is a $(4N+4) \times (4N+4)$ matrices like the following

$$M = \begin{bmatrix} Z & Z & Z & Z \\ Z & Z & Z & Z \\ C & Z & Z & Z \\ Z & C & Z & Z \end{bmatrix}$$

M also has four extra zero columns and four extra zero rows for τ's and boundary conditions (omitted from above).

$$L = \begin{bmatrix} D & Z & -C & Z \\ Z & D & Z & -C \\ -n^2 C & Z & D & Z \\ Z & -n^2 C & Z & D \end{bmatrix}$$

L has four extra rows for boundary conditions

```
BC_rows = np.zeros((4, 4*N))
BC_rows[0, :N] = (-1)**i
BC_rows[1, :N] = (+1)**i
BC_rows[2, N:2*N] = (-1)**i
BC_rows[3, N:2*N] = (+1)**i
```

four extra rows for τ's

```
cols = np.zeros((4*N,4))
cols[  N-1, 0] = 1
cols[2*N-1, 1] = 1
cols[3*N-1, 2] = 1
cols[4*N-1, 3] = 1
```

and an empty 4×4 corner.
The RHS $F(X)$ could be calculated step-by-step as follows:

- calculate u_x
  require the coefficients of u and multiply it by x derivative matrix
- calculate u_y
  require the coefficients of u, multiply it by y derivative matrix, and convert it back to the T coefficient space
  be careful about the axis direction of spsolve
- calculate u_{xx}
  require the coefficients of u_x and multiply it by x derivative matrix

- calculate u_{xy}
  require the coefficients of u_x, multiply it by y derivative matrix, and convert it back to the T coefficient space

- calculate u_{yy}
  require the coefficients of u_y, multiply it by y derivative matrix, and convert it back to T coefficient space

- calculate norm
  calculate $u_x^2 + u_y^2$ in the grid space

- calculate the RHS
  do the calculation as in equation 1 in the grid space
  convert the RHS from grid space to T space
  convert the RHS from T space to U space

3 Resolution Analysis

We simulate $(0, 2\pi) \times (0, 2\pi)$ grid space with resolution $N = 128, 256, 512$ from $t = 0$ to 0.6. The initial data defined as a Gaussian peak at (π, π):

$$\phi(x, y) = \exp\left(-(x - \pi)^2 - (y - \pi)^2\right)$$

The behavior of different resolutions look extremely similar in the timespan.

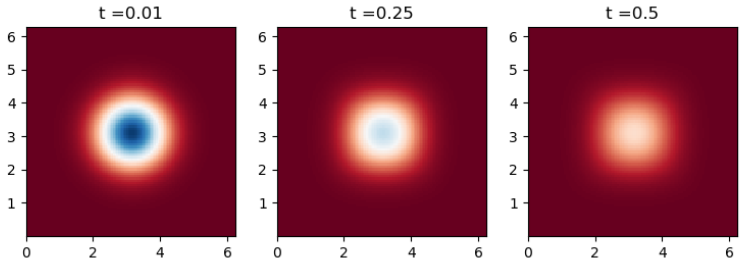

Figure 1: $N = 128$

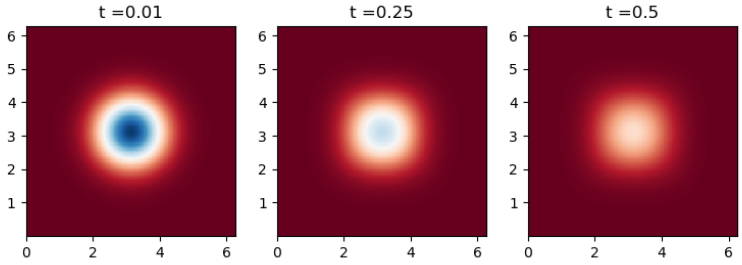

Figure 2: $N = 256$

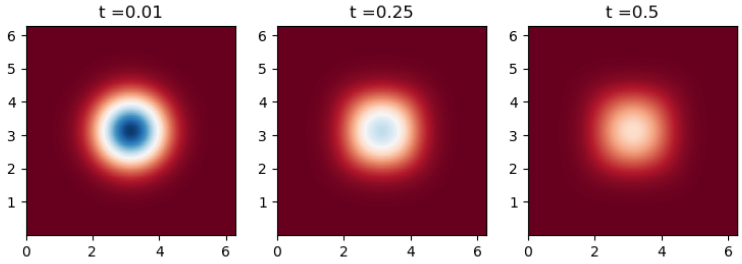

Figure 3: $N = 512$

4 Simulation Results

Now we are going to choose a specific level set on the grid and observe its evolution.
Recall that our initial data is a Gaussian peak centered at (π, π) We can pick the curve as the contour $\phi(x, y) = \exp(-1)$. That will give us

$$\begin{aligned} \phi(x, y) = \exp\left(-(x-\pi)^2 - (y-\pi)^2\right) &= \exp(-1) \\ \Rightarrow (x-\pi)^2 + (y-\pi)^2 &= 1 \end{aligned}$$

which is a circle centered at (π, π) with radius 1.
The following simulation shots exhibit the shrinkage of the contour.

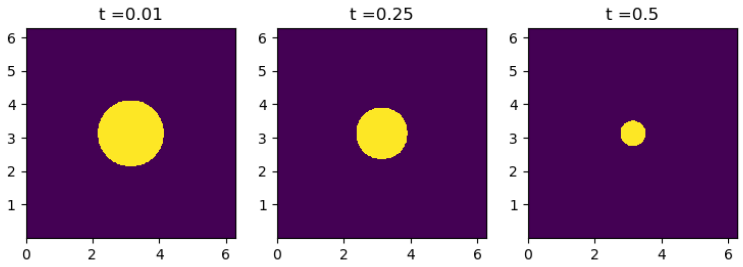

Figure 4: $N = 512$

Since the curvature of a circle is the inverse of its radius, and we know that the radius is shrinking with velocity equal to its curvature. We can write down the radius of circle as

$$\frac{dr}{dt} = -\frac{1}{r}$$

Solving with initial condition $r(t = 0) = 1$, we obtain

$$r(t) = \sqrt{1 - 2t}$$

This predicts that the radius shrinks to 0 at $t = 0.5$, while we have a not negligible area according to figure 4 above. In our simulation, the area disappears at around $t = 0.58$, and I believe that this is within an acceptable range. And the area does shrink faster at smaller radius (see the gif), so I would prefer to interpret the difference in disappearance timing as timestepping inaccuracy rather than an error in the algorithm.