COMP0115: Coursework 1

Mingzhou Hu

February 2019

1: 1D linear diffusion

The 1D Linear diffusion equation is: $\frac{\partial f}{\partial t} = \frac{\partial^2 f}{\partial x^2}$.
Firstly, I chose a number of point N at 256 to solve the problem and assumed that initial f was the 70^{th} row of the image. Then I implemented the 256×256 matrix A representing the Laplacian:

$$\text{A}=\begin{bmatrix} -1 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & -1 \end{bmatrix},$$

the sum of each row is 0. Next, I took one line from the test image to initialise the first step f^0 and set Δt and iterations(k) to 0.3 and 10 respectively. Lastly, I tested both explicit and implicit scheme, and also the convolution with a Gaussian filter which was generated by the function gaussmf in MATLAB.
The test results are below, we can know from the results that the results for explicit scheme, implicit scheme and Gaussian filter are very similar. So I think the results are reasonable.

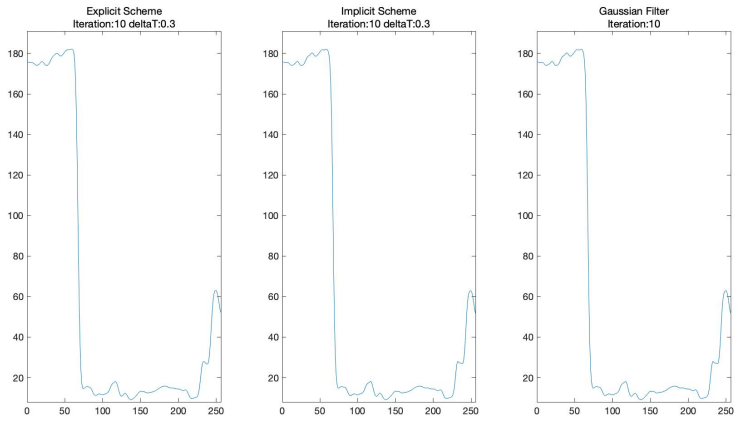

Figure 1: Iterations=10 Δt=0.3

Along with test after test, I found the largest Δt is 0.5 so that the explicit scheme is stable. The test results are below:

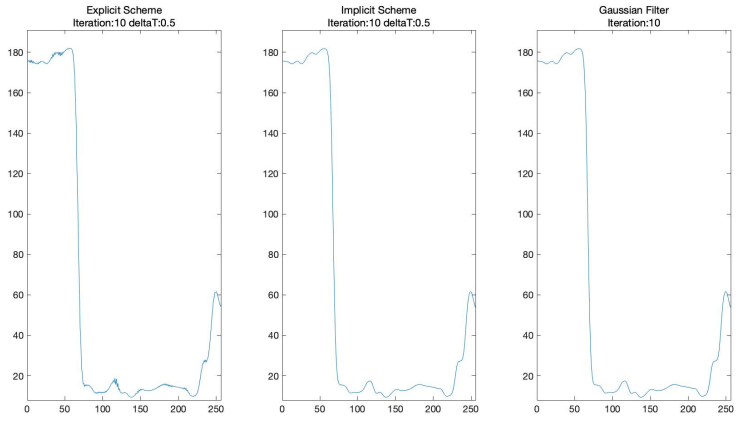

Figure 2: Iterations=10 Δt=0.5

In this case, I encountered a problem about implementing matrix A representing the Laplacian. The 1D Laplace equation is: $\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{\Delta x^2} = 0$, so the medium rows for A should consist of 1, -2 ,1 and 0. However, if I change the 1, -2, 1 into $\frac{1}{2}$, -1, $\frac{1}{2}$ respectively, the largest Δt will double to 1.0. Therefore, I can confirm that the largest Δt is related to the construction of matrix A.

2: 2D linear diffusion

The 2D Linear diffusion equation is: $\frac{\partial f}{\partial t} = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$.
In order to produce a diffusion scheme for a $N \times N$ image, I need to reshape the original image into a $N^2 \times 1$ vector and implement the $N^2 \times N^2$ matrix A. The A matrix is too large to be held in a dense "full" form, so I must use the sparse for A. Because Implicit scheme is computationally challenging, I chose semi-implicit scheme, involving splitting A into A^X and A^Y. In order to construct A^X and A^Y, dd matrix is need to be defined first, which is a $N^2 \times 3$ matrix with each row being [1 -2,1]. Then, I used function spdiags to construct A^X and A^Y in MATLAB($A^X = spdiags(dd, -1:1, N*N, N*N)$ and $A^Y = spdiags(dd, [-N, 0, N], N*N, N*N)$). Lastly, I set Δt and iterations(k) to 0.1 and 100 respectively to implement the tests about calculating a series of images and tracking the position of local maximum(using function find in MATLAB) over time in the succeeding images.
The tests results are below(the position of local maximum is marked by $*$): the iteration of the first figure is 1 and now the local maximum is 0.9813 in position [233,108] of the image, the iteration of the second figure is 19 and now the local maximum is 0.8892 in position [95, 141] of the image, the iteration of the third figure is 75 and now the local maximum is 0.7309 in position [27, 123] of the image, and the iteration of the last figure is 100 and now the local maximum is 0.7295 in position [25, 122] of the image. I conclude three points about the results of the tests. Firstly, the images appear as successive blurring over time. Secondly, the position of the local maximum always keep stable for a short time, and then the local maximum will be annihalated and the position of that will change. Thirdly, the value for local maximum becomes smaller and smaller over time. From the conclusion above, I can make sure that the results are reasonable in the tests.

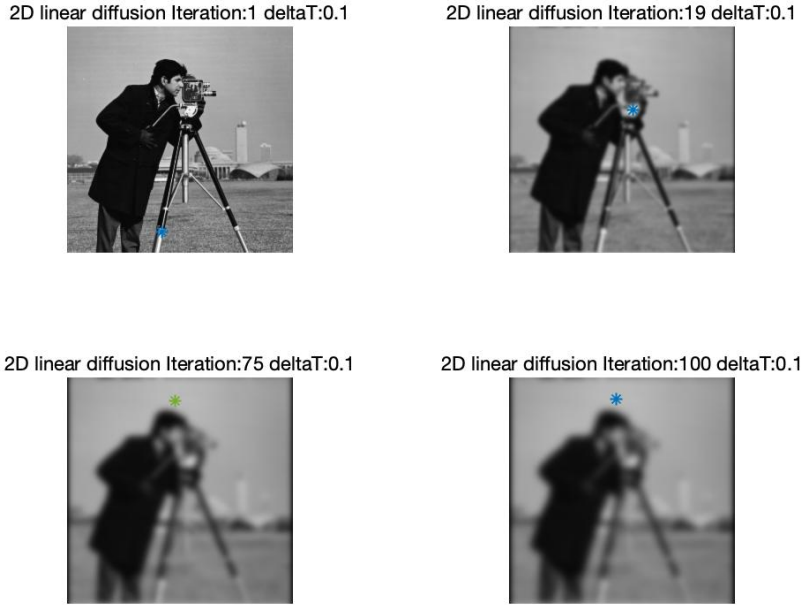

Figure 3: A series of images and local maximum tracking

In this case, I encountered a problem about function plot in MATLAB. I used $plot(y, N - x, '*')$ to mark the position of local maximum at first, but I found that the position of $*$ appeared in image is different from that I obtained by function find. So I searched the usage of function plot and then changed the code into $plot(y, x, '*')$. Eventually, I got the right results.

3: 2D Anisotropic diffusion

The 2D anisotropic diffusion equation is: $\frac{\partial f}{\partial t} = \frac{\partial}{\partial x}\gamma\frac{\partial f}{\partial x} + \frac{\partial}{\partial y}\gamma\frac{\partial f}{\partial y}$.
In this case, I implemented the scheme using Perona-Malik function. Firstly, I need to compute γ by the equation: $\gamma = \frac{1}{1+(\frac{|\bigtriangledown f|}{T})^2}$, $\bigtriangledown f$ can be computed by the gradient of image which is $\sqrt{g_x^2 + g_y^2}$ and the gradient can by computed by the function gradient in MATLAB, and T is threshold(0.29671 in this case) which can be computed by the equation: T=$\frac{max\bigtriangledown f}{2}$. Then, I used gamma computed above to create gam=spdiags(reshape(gamma,[],1),0:0,N*N,N*N). The next step was to create explicit 1^{st}-order derivative operator D_X and D_Y so that PM matrix can be created. In order to create D_X and D_Y, I need to construct dx and dy matrices at first which both are $N^2 \times 2$ matrices with each row being [1 -1], and then I can can computed D_X and D_Y by spdiags function in MATLAB($D_X = spdiags(dx, 0:1, N*N, N*N)$ and $D_Y = spdiags(dy, [0, N], N*N, N*N)$). Now I can create PM by the equation: $PM = -(Dx' * gam * Dx + Dy' * gam * Dy)$, which should has the same structure as the Laplacian I used in Q2. Lastly, I repeated the tests I performed in Q2.
The tests results are below(the position of local maximum is marked by $*$): the iteration of the first figure is 1 and now the local maximum is 0.9823 in position [233,108] of the image, the iteration of the second figure is 12 and now the local maximum is 0.9198 in position [94, 142] of the image, the iteration of the third figure is 75 and now the local maximum is 0.7313 in position [96, 140] of the image, and the iteration of the last figure is 100 and now the local maximum is 0.7457 in position [96, 140] of the image.

Figure 4: A series of images and local maximum tracking

The result of changes for the image are different from Q2, although the image appears as successive blurring over time just like that in Q2, the edge of the man is clearer than that in Q2. That is because we add γ to anisotropic diffusion equation, whoes value is depend on gradient. The position of local maximum changes only one time in this case, but it changes two times in Q2. The tendancy of the values for local maximum in this case is same to that in Q2, but the values are different.