Prosjekt 5

mariel.ah

December 2019

1 A

Diffusion equation
Some differential equations are impossible to solve and requires other methods to approach the solutions. Some of these are Eulers methods. Many points are generated in the interval where the solution lies.

There are many different numerical methods to solve PDEs, but one of the most used methods are the finite difference methods. It is relatively easy to understand and simpler to implement.

Explicit scheme: the approximate solution u_j^{n+1} at any interior point is obtained by a simple marching in time.

Implicit scheme: solving a linear system at each time step, in order to obtain all the approximate solution u_j^n at all interior points.

a) Setting up algorithms

2 Forward Euler, Explicit scheme

1- dimensional parabolic equation

$$u_t \approx \frac{u(x,t+\Delta t) - u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t) - u(x_i,t_j)}{\Delta t}$$

$$u_{xx} \approx \frac{u(x_i+\Delta x,t_j) - 2u(x_i,t_j) + u(x_i-\Delta x,t_j)}{\Delta x^2} \tag{1}$$

Backward Euler:

$$u_t \approx \frac{u(x,t) - u(x,t-\Delta t)}{\Delta t} = \frac{u(x_i,t_j) - u(x_i,t_j-\Delta t)}{\Delta t}$$

$$u_{xx} \approx \frac{u(x_i+\Delta x,t_j) - 2u(x_i,t_j) + u(x_i-\Delta x,t_j)}{\Delta x^2} \tag{2}$$

Crank-Nicolson scheme:

$$u_t \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t)-u(x_i,t_j)}{\Delta t}$$

$$u_{xx} \approx \frac{1}{2}(\frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2}$$
$$+\frac{u(x_i+\Delta x,t_j+\Delta t)-2u(x_i,t_j+\Delta t)+u(x_i-\Delta x,t_j+\Delta t)}{\Delta x^2}) \tag{3}$$

For the different we have the boundary conditions:

$$u(0,t) = a(t) = 0, fort \geq 0$$
$$u(L,t) = b(t) = 1, fort \geq 0$$

Generell løsning: u(x,t)=ax+F(x)G(t).
And we have the initial condition:

$$u(x,0) = g(x) = 0, for 0 < x < L \tag{4}$$

We also know that $u_{xx} = u_t$. With these conditions we can find the equations needed to implement for the respective methods.

3 1. Forward Euler

We assume that the density u obeys Gauss-Greens theorem.

$$u_{xx} = u_t$$
$$\frac{u(x_i,t_j+\Delta t)-u(x_i,t_j)}{\Delta t} = \frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2}$$

Which can be simplified as followed:

$$\frac{u_{i,j+1}-u_{i,j}}{\Delta t} = \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}$$

$$u_{i,j+1} = \frac{\Delta t}{\Delta x^2}(u_{i+1,j}-2u_{i,j}+u_{i-1,j})+u_{i,j}$$

Defining

$$\alpha = \frac{\Delta t}{\Delta x^2} \tag{5}$$

$$u_{i,j+1} = \alpha u_{i-1,j} + (1-2\alpha)u_{i,j} + \alpha u_{i+1,j} \tag{6}$$

Since the initial conditions are given it is possible to find $u_{i,1}$.

$$u_{i,1} = \alpha g(x_{i-1}) + (1-2\alpha)g(x_i) + \alpha g(x_{i+1}) = 0$$

We can imagine a steady state when t $\rightarrow \infty$.

$$u(x,t) = v(x,t) + ax$$
$$v(0,t) = c(t) = 0$$
$$v(L,t) = d(t) = 0$$

$$v(0,t) = v(L,t) = 0$$

The partial differential equation can be written as a vector:

$$V_j = \begin{vmatrix} u_{1,j} \\ u_{2,j} \\ ... \\ u_{L,j} \end{vmatrix}$$

To find V_{j+1} it is now possible to use matrix multiplication!

$$V_{j+1} = AV_j = ... = A^{j+1}V_0$$

V_0 is the vector with u(x,0) values with 0 ¡ x ¡ L.

Because of the initial condition given in equation 4 the vector V_0 is a zero vector. A can be written as a general tridiagonal matrix (because of the boundary conditions:

$$A = \begin{vmatrix} b & c & 0 & 0 & ... \\ a & b & c & 0 & ... \\ ... & ... & ... & ... & ... \\ 0 & ... & a & b & c \\ 0 & ... & ... & a & b \end{vmatrix}$$

From the equation 6 it is possible to see that a is represented by $\alpha u_{i-1,j}$, b is represented by $(1-2\alpha)g(x_i)$ and c is represented by $\alpha g(x_{i+1})$.

Therefore we get the tridiagonal matrix:

$$A = \begin{vmatrix} (1-2\alpha) & \alpha & 0 & 0 & ... \\ \alpha & (1-2\alpha) & \alpha & 0 & ... \\ ... & ... & ... & ... & ... \\ 0 & ... & \alpha & (1-2\alpha) & \alpha \\ 0 & ... & ... & \alpha & (1-2\alpha) \end{vmatrix} \tag{7}$$

4 Backward Euler, Implicit scheme

The same procedure is done for the Backward Euler (2), assuming the density u obeys Gauss-Greens theorem.

$$u_{xx} = u_t$$

$$\frac{u_{i,j} - u_{i,j-1}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

Again using equation 5:

$$u_{i,j-1} = -\alpha u_{i-1,j} + (2\alpha - 1)u_{i,j} - \alpha u_{i+1,j} \tag{8}$$

This gives us the matrix

$$A = \begin{vmatrix} (2\alpha-1) & -\alpha & 0 & 0 & ... \\ -\alpha & (2\alpha-1) & -\alpha & 0 & ... \\ ... & ... & ... & ... & ... \\ 0 & ... & -\alpha & (2\alpha-1) & -\alpha \\ 0 & ... & ... & -\alpha & (2\alpha-1) \end{vmatrix}$$

For the implicit scheme we have:

$$V_j = A^{-1}V_{j-1} = ... = A^{-j}V_0$$

5 Crank-Nicolson scheme

This method combines both explicit and implicit schemes and bisect the equation.

$$\frac{1}{2\Delta x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i-1,j-1} + 2u_{i,j-1} + u_{i+1,j-1}) = \frac{1}{\Delta t}(u_{i,j} - u_{i,j-1})$$

Using equation 5:

$$-\alpha u_{i-1,j} + (2\alpha + 2)u_{i,j} - \alpha(u_{i+1,j} = \alpha u_{i-1,j-1} + (2 - 2\alpha) + \alpha u_{i+1,j-1}$$

A matrix B can be used for both sides of the equation (switching between negative and positive), and we can rewrite the equation:

$$B = \begin{vmatrix} 2 & -1 & 0 & 0 & ... \\ -1 & 2 & -1 & 0 & ... \\ ... & ... & ... & ... & ... \\ 0 & ... & -1 & 2 & -1 \\ 0 & ... & ... & -1 & 2 \end{vmatrix}$$

$$(2I + \alpha B)V_j = (2I - \alpha B)V_{j-1}$$

This resolves in a definition of V_j:

$$V_j = (2I + \alpha B)^{-1}(2I - \alpha B)V_{j-1}$$

6 B - truncation error

For the forward Euler method, the LTE is O(h2). Hence, the method is referred to as a first order technique. In general, a method with O(hk+1) LTE is said to be of kth order.

The truncation error is defined in 9

$$y_{i+1} = y(t_i) + h(y_i' + ... + y^p(t_i)\frac{h^{(p-1)}}{p!}) + O(h^p) \tag{9}$$

For the Forward Euler 1 and Backward Euler it is easy to see that for u_t the equation yields for the first derivative and the truncation error is therefore $O(h^{p=1}) = O(h)$. The equation for u_x contains in both cases of first order and the truncation error is therefor $O(h^{p=2} = O(h^2))$. Since the Crank-Nicolson method is a combination of these two we have to approach the truncation error in a different way.

The Forward Euler approximation is only stable if α is smaller or equal to 1/2 while the Backward Euler and the Crank-Nicolson approximation is stable for all values of Δt and Δx. The stability is limited by the spectral radius condition:

$$\rho(A) < 1$$
$$\rho(A)max|\lambda| : det(A - \lambda\hat{I}) = 0$$

This is achieved if the matrix A is positive definite, if all the eigenvalues to A is positive. This is the case for the Backward Euler and the Crank-Nicolson approximation, but not for the Forward Euler method.

The matrix A 7 is possible to rewrite as: A=I-αB.

$$B = \begin{vmatrix} 2 & -1 & 0 & 0 & ... \\ -1 & 2 & -1 & 0 & ... \\ ... & ... & ... & ... & ... \\ 0 & ... & -1 & 2 & -1 \\ 0 & ... & ... & -1 & 2 \end{vmatrix} \tag{10}$$

The matrix has the eigenvalues: $b + 2\sqrt{ac}cos(\frac{i\pi}{n+1})$.

The eigenvalues for B is (2+2$\sqrt{1}$cos($\frac{i\pi}{n+1}$))??

Explicit scheme: Truncation error

From Computational partial differential equations:

If TE(x,t) -¿ 0 as Δt, Δx -¿ 0 for any (t,x), then the scheme is consistent with the differential equation. If for any point (t,x), $x_j- > x, t_n- > t$ implies that $u_j^n- > u(x,t)$. the numerical solution approximates an exact solution. It is said that the scheme is convergent. A PDE is "well-posed" if the solution exists and depends continously on the initial and boundary conditions.

Stability: fourier analysis (von Neumann)

Lax-Richtmyer lemma: for a consistent difference approximation to a well-posed linear time-dependent problem, the stability of the scheme is necessary and sufficient for convergence. s. 16 comp partial diff using matlab

TE: se s 24 comp partial diff using matlab

7 Closed form solution 1D

$$\frac{\partial^2 u(x,y)}{\partial^2 x} = \frac{\partial u(x,y,t)}{\partial t}, t > 0, x \in [0,1] \tag{11}$$

The solution can be written as:

$$u(x,t) = F(x)G(t) \tag{12}$$

We can now rewrite the equation 11 which is constant:

$$\frac{F''}{F} = \frac{G'}{G} = -\lambda^2$$

Now we have two different equations

$$F'' + F\lambda^2 = 0$$
$$G' + G\lambda^2 = 0$$

For each of these two differential equations there is the general solutions:

$$F(x) = Asin(\lambda x) + Bcos(\lambda x)$$
$$G(t) = Ce^{-\lambda^2 t}$$

From the boundary conditions we see that B is zero and that $\lambda = n\pi/L$we can now rewrite equation 12:

$$u(x,t) = Asin(n\pi x/L)e^{-n^2\pi^2 t/L^2}$$

Which is a possible solution.
To find A_n it is necessary to solve the equation:

$$A_n = \frac{2}{L}\int_0^L g(x)sin(n\pi x/L)dx$$

8 5.e moving to two dimensions

$$\frac{\partial^2 u(x,y,t)}{\partial^2 x} + \frac{\partial^2 u(x,y,t)}{\partial^2 y} = \frac{\partial u(x,y,t)}{\partial t}, t > 0, x, y \in [0,1] \tag{13}$$

Laplace: ${}^2u(x) = u_{xx} + u_{yy} = 0$
Poissons:

$$u_{xx} + u_{yy} = -\rho(x,y)$$

Which is a 2+1 dimensional differential equation. We now have

$$u_{xx} \approx \frac{u(x+\Delta x, y, t_j) - 2u(x,y,t) + u(x-\Delta x, y, t)}{\Delta x^2}$$

This can be written easier in an discretized verison:

$$u_{xx} \approx \frac{u^l_{i+1,j} - 2u^l_{i,j} + u^l_{i-1,j}}{\Delta x^2}$$

We assume an equal number of mesh points for x and y and we have: $x_i = x_0 + ih$ and $y_j = y_0 + *h$.

$$u_{yy} \approx \frac{u^l_{i,j+1} - 2u^l_{i,j} + u^l_{i,j-1}}{\Delta y^2}$$

$$u_t \approx \frac{u^{l+1}_{i,j} + u^l_{i,j}}{\Delta t}$$

If we now solve equation 13 for $u^{l+1}_{i,j}$:

$$u^{l+1}_{i,j} \approx u^l_{i,j} + \alpha[u^l_{i+1,j} + u^l_{i-1,j} + u^l_{i,j+1} + u^l_{i,j-1} - 4u^l_{i,j}] \tag{14}$$

We choose to set $h = \Delta x = \Delta y = \frac{L}{n+1}$

We need to solve

$$u_{i,j} = \frac{1}{4}[u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j}] + h^2\rho_{i,j}$$

We can define $\rho^{\sim} = h^2\rho$

$$4u_{i,j} - [u_{i,j+1} + u_{i,j-1} + u_{i+1,j} + u_{i-1,j}] = -\rho^{\sim}$$

This yields the following equations:

$$\begin{aligned}
4u_{1,1} - u_{1,2} - u_{1,0} - u_{2,1} - u_{0,1} &= -\rho^{\sim}_{1,1} \\
4u_{1,2} - u_{1,3} - u_{1,1} - u_{2,2} - u_{0,2} &= -\rho^{\sim}_{1,2} \\
4u_{2,1} - u_{2,2} - u_{2,0} - u_{3,1} - u_{1,1} &= -\rho^{\sim}_{2,1} \\
4u_{2,2} - u_{2,3} - u_{2,1} - u_{3,2} - u_{1,2} &= -\rho^{\sim}_{2,2}
\end{aligned} \tag{15}$$

The boundary conditions are given:

$$u_{i,0} = u_{i,L} = g_{i,0}, 0 \leq i \leq n+1$$

and

$$u_{0,j} = g_{0,j}$$

,

$$u_{L,j} = g_{L,j}$$

The equations in 15 can therefore be written as a matrix equation Ax = b, taking into account for the known boundary conditions in the right side of the equation.

$$\begin{vmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{vmatrix} \begin{vmatrix} u_{1,1} \\ u_{1,2} \\ u_{2,1} \\ u_{2,2} \end{vmatrix} = \begin{vmatrix} u_{0,1} + u_{1,0} - \rho^{\sim}_{1,1} \\ u_{1,3} + u_{0,2} - \rho^{\sim}_{1,2} \\ u_{2,0} + u_{3,1} - \rho^{\sim}_{2,1} \\ u_{2,3} + u_{3,2} - \rho^{\sim}_{2,2} \end{vmatrix} \tag{16}$$

We have to approach the solution differently than for the 1-dimensional case, as this is not a tridiagonal issue. Hence the LU decomposition will not be suitable. It is better with methods like the Jacobis og Gauss-Seidels method. I will for this explicit scheme take use of the Jacobis method.

The 4x4 matrix in eqation 16 can be in three different matrices: $A = D + U + L$. D is a diagonal matrix, L is a lower triangular matrix and U is an upper triangular matrix.

$$D = \begin{vmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 4 \end{vmatrix}, L = \begin{vmatrix} 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 \end{vmatrix}, U = \begin{vmatrix} 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

With the Jacobis method we could solve the unknown values in 16 and we could possibly now all the values in these two dimensions. Never the less we should not forget the third dimension. We can now go back to the equation 14 and rewrite it with our new definitions.

$$u^1_{1,1} = \frac{1}{4}(b_1 - u^0_{1,2} - u^0_{2,1})$$

$$u^1_{1,2} = \frac{1}{4}(b_2 - u^0_{1,1} - u^0_{2,2})$$

$$u^1_{2,1} = \frac{1}{4}(b_3 - u^0_{1,1} - u^0_{2,2})$$

$$u^1_{2,2} = \frac{1}{4}(b_4 - u^0_{1,2} - u^0_{2,1})$$

The unknown values (x) can also be solved with the decomposition of the matrix A:

$$x^{r+1} = D^{-1}(b - L + U)x^r)$$