Crank-Nicolson

O método de Crank-Nicolson para a equação parabólica $u_t = u_{xx}$ resulta na seguinte equação discretizada:

$$\left(-\frac{\lambda}{2}\right) w_{i-1}^{j+1} + (1+\lambda) w_i^{j+1} + \left(-\frac{\lambda}{2}\right) w_{i+1}^{j+1} = \left(\frac{\lambda}{2}\right) w_{i-1}^{j} + (1-\lambda) w_i^{j} + \left(\frac{\lambda}{2}\right) w_{i+1}^{j}$$

O algoritmo 12.3 foi feito com condições de contorno homogêneas nas duas extremidades o que não é o nosso caso – no nosso caso a equação tem um termo forçante e uma condição é de Dirichlet e a outra de Neumann. Logo ele terá de ser modificado.

O algoritmo 12.3 foi apresentando com o domínio em x inicialmente começando em 1 e terminando em m – 1 (creio que aqui existe um erro, pois domínio termina em m+1, e último ponto interno é i = m, verifique). Considere o extremo a esquerda. A 1ª equação onde aparece a condição de Dirichlet é a de i = 2

$$\left(-\frac{\lambda}{2}\right) w_1^{j+1} + (1+\lambda) w_2^{j+1} + \left(-\frac{\lambda}{2}\right) w_3^{j+1} = \left(\frac{\lambda}{2}\right) w_1^{j} + (1-\lambda) w_2^{j} + \left(\frac{\lambda}{2}\right) w_3^{j}$$

Os valores no contornos são dados pela solução $w_1^j = u(0, jk)$, logo o primeiro termo tem que ser levado para o lado direito

Analisando a última equação, no caso i=m

$$\left(-\frac{\lambda}{2}\right) w_{m-1}^{j+1} + (1+\lambda) w_m^{j+1} + \left(-\frac{\lambda}{2}\right) w_{m+1}^{j+1} = \left(\frac{\lambda}{2}\right) w_{m-1}^{j} + (1-\lambda) w_m^{j} + \left(\frac{\lambda}{2}\right) w_{m+1}^{j}$$

Nesse caso a aplicação da condição de Neumann vai substituir o termo w_{m+1}^{j+1} , e essa substituição vai modificar os coeficientes de w_m^{j+1} e w_{m-1}^{j+1} e fará aparecer mais um termo a ser transportado para o lado direito.

E não se esqueçam que em todas as equações deve aparecer do lado direito o termo forçante

Minha recomendação não é seguir o algoritmo 12.3 e sim montar o sistema de forma correta no Python, usando a indexação que começa com 0.

Uma vez montado o sistema usar algum o algoritmo de solução de sistemas tridiagonais.

ALGORITHM 6.7

Crout Factorization for Tridiagonal Linear Systems

To solve the $n \times n$ linear system

$$\begin{aligned}
E_1 &: \quad a_{11}x_1 + a_{12}x_2 &&= a_{1,n+1},\\
E_2 &: \quad a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &&= a_{2,n+1},\\
&\vdots \qquad\qquad \vdots &&\vdots\\
E_{n-1} &: \quad a_{n-1,n-2}x_{n-2} + a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n &&= a_{n-1,n+1},\\
E_n &: \quad a_{n,n-1}x_{n-1} + a_{nn}x_n &&= a_{n,n+1},
\end{aligned}$$

which is assumed to have a unique solution:

INPUT the dimension n; the entries of A.

OUTPUT the solution $x_1, \ldots, x_n$.

(*Steps 1–3 set up and solve* $L\mathbf{z} = \mathbf{b}$.)

Step 1 Set $l_{11} = a_{11}$;
$u_{12} = a_{12}/l_{11}$;
$z_1 = a_{1,n+1}/l_{11}$.

Step 2 For $i = 2, \ldots, n-1$ set $l_{i,i-1} = a_{i,i-1}$; (*ith row of* L.)
$l_{ii} = a_{ii} - l_{i,i-1}u_{i-1,i}$;
$u_{i,i+1} = a_{i,i+1}/l_{ii}$; ($(i+1)$*th column of* U.)
$z_i = (a_{i,n+1} - l_{i,i-1}z_{i-1})/l_{ii}$.

Step 3 Set $l_{n,n-1} = a_{n,n-1}$; (*nth row of* L.)
$l_{nn} = a_{nn} - l_{n,n-1}u_{n-1,n}$.
$z_n = (a_{n,n+1} - l_{n,n-1}z_{n-1})/l_{nn}$.

(*Steps 4 and 5 solve* $U\mathbf{x} = \mathbf{z}$.)

Step 4 Set $x_n = z_n$.

Step 5 For $i = n-1, \ldots, 1$ set $x_i = z_i - u_{i,i+1}x_{i+1}$.

Step 6 OUTPUT $(x_1, \ldots, x_n)$;
STOP.