Diferencias Finitas para la solución de ecuaciones diferenciales parciales

Ecuación del calor Bidimensional

UHU – 4º Ingeniero Industrial – Curso 2008/09

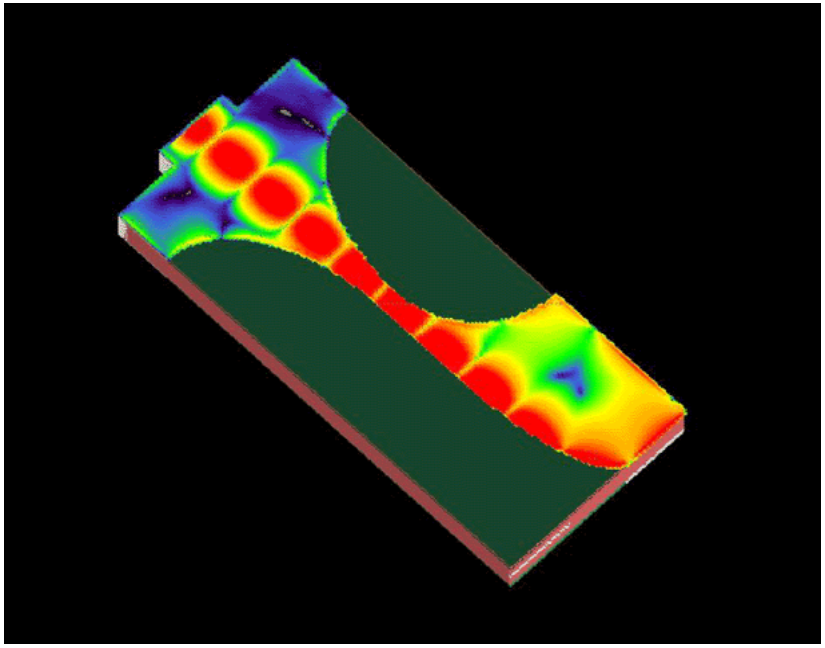
Profesor: Antonio Algaba Durán

Alumno: Luis Heredia Castillo

Asignatura: Ecuaciones Diferenciales y Métodos Numéricos

Para flujo bidimensional en régimen transitorio:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\dot{q}}{k} = \frac{\rho\, C_p}{k} \frac{\partial T}{\partial \tau}$$

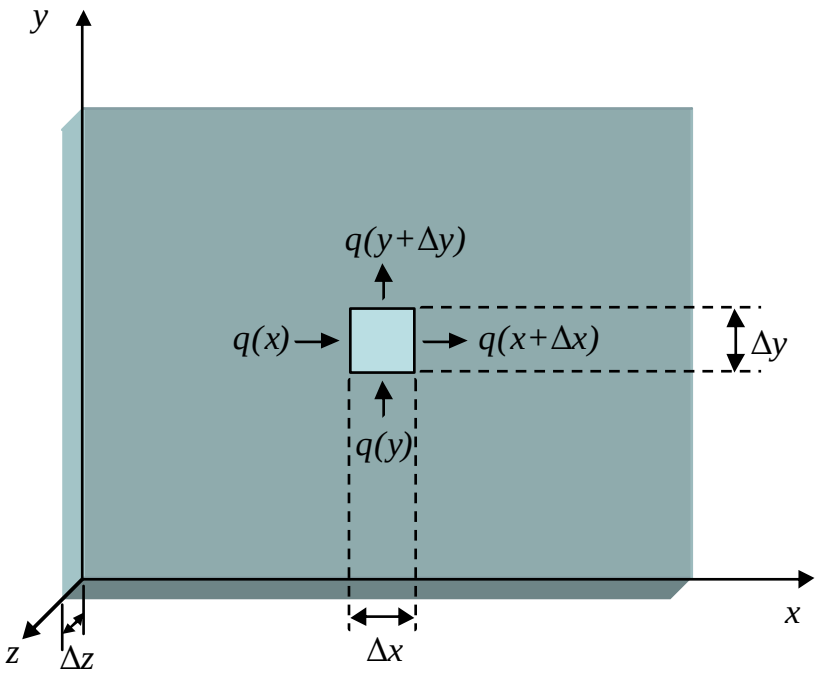


$$\frac{\partial u}{\partial x} = \lim_{\Delta x \to 0} \frac{u(x+\Delta x, y) - u(x, y)}{\Delta x}$$

$$\frac{\partial u}{\partial y} = \lim_{\Delta y \to 0} \frac{u(x, y+\Delta y) - u(x, y)}{\Delta y}$$

Diferencias finitas: ecuaciones parabólicas

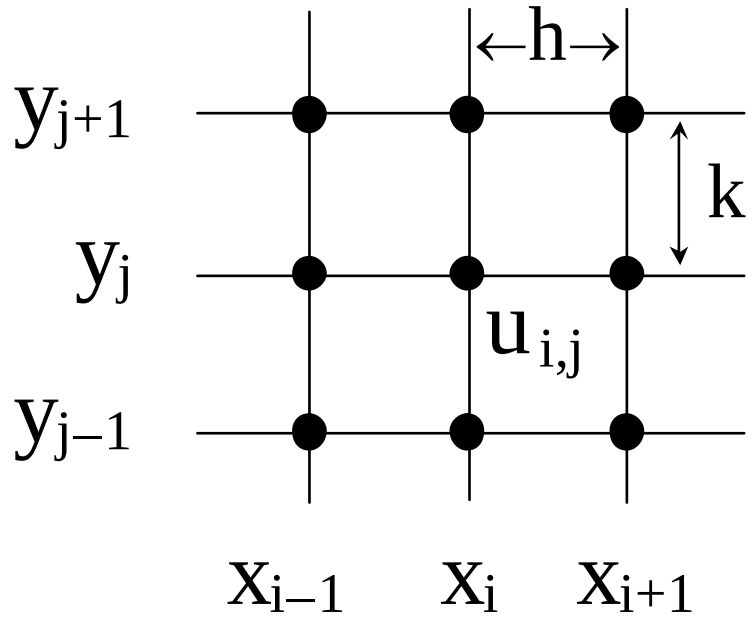
$$\frac{\partial T}{\partial t}(x,y,t) = c^2 * \left(\frac{\partial^2 u}{\partial x^2}(x,t) + \frac{\partial^2 u}{\partial x^2}(y,t) \right) + F(x,y,t)$$



q es el flujo de calor (cal/(cm^2 s)

Diferencias finitas

- Discretización: EDP ⇒ EDF
- Métodos explícitos
  - Sencillos
  - Inestables
- Métodos implícitos
  - Más complejos
  - Estables

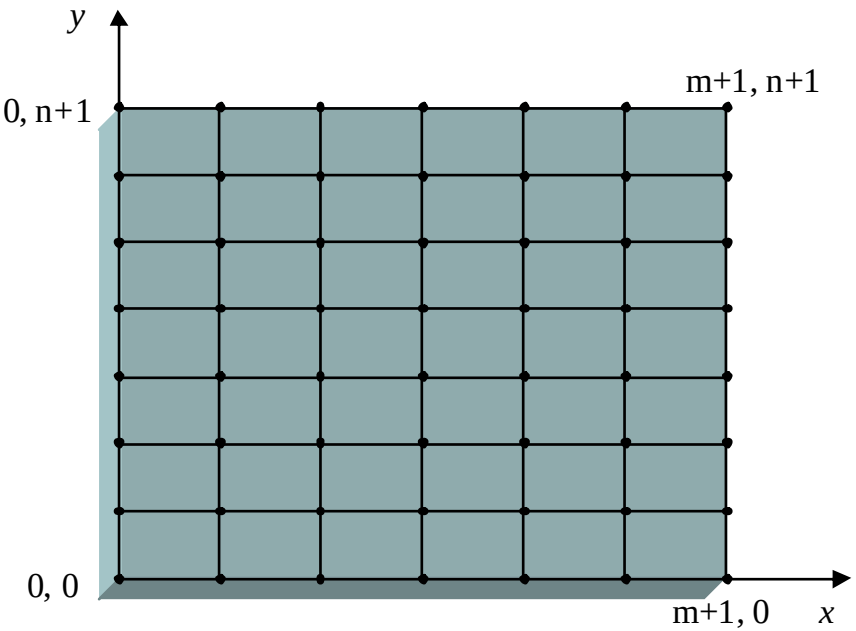


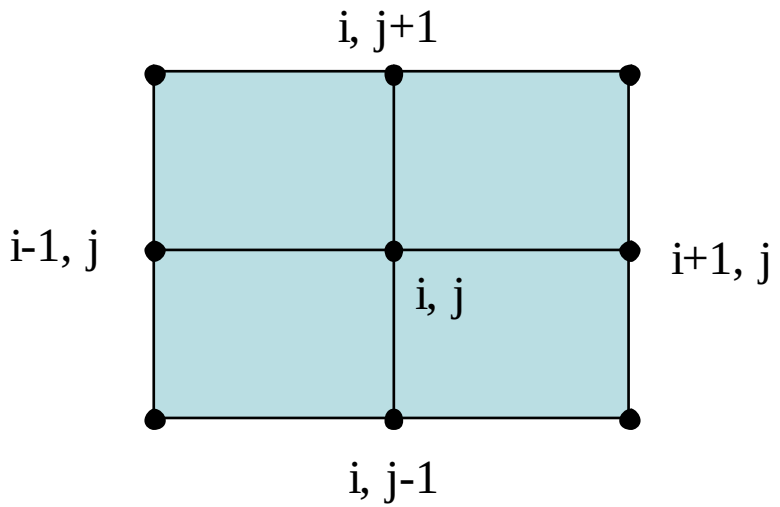
Técnica de solución: Métodos explícitos

$$\frac{\partial T}{\partial t} = \frac{T_i^{l+1} - T_i^{l}}{\Delta t} + O[\Delta t]$$

$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i+1,j}^{l} - 2T_{i,j}^{l} + T_{i-1,j}^{l}}{\Delta x^2} + \mathrm{O}[\Delta x^2]$$

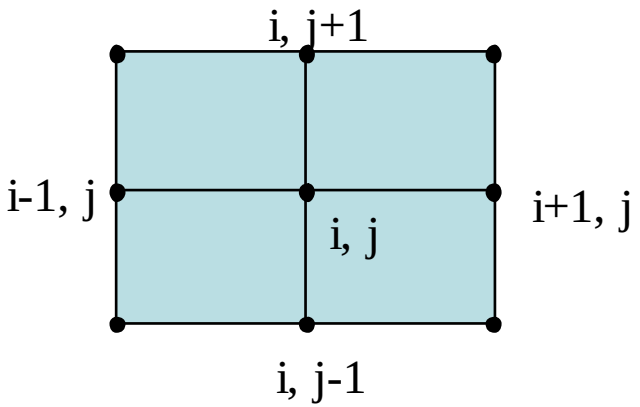
$$\frac{\partial^2 T}{\partial y^2} = \frac{T_{i,j+1}^{l} - 2T_{i,j}^{l} + T_{i,j-1}^{l}}{\Delta y^2} + \mathrm{O}[\Delta y^2]$$





Métodos explícitos - Condiciones Dirichlet

$$h = \frac{L}{m} \qquad k = \frac{T}{N}$$



$$\frac{W_{i,j}^{l+1} - W_{i,j}^{l}}{k} = c^2 \left(\frac{W_{i+1,j}^{l} - 2W_{i,j}^{l} + W_{i-1,j}^{l}}{h^2} + \frac{W_{i,j+1}^{l} - 2W_{i,j}^{l} + W_{i,j-1}^{l}}{h^2} \right) + F_{i,j}^{l}$$

$$\boxed{W_{i,j}^{l+1} = \lambda\left(W_{i+1,j}^{l} + W_{i-1,j}^{l} + W_{i,j+1}^{l} + W_{i,j-1}^{l}\right) + \left(1 - 4\lambda\right) * W_{i,j}^{l} + k * F_{i,j}^{l}}$$

$$\lambda = \frac{c^2 * k}{h^2}$$

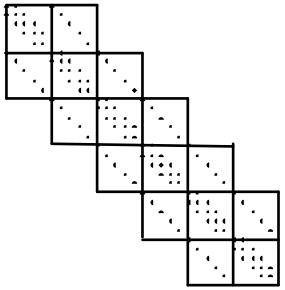
$$\boxed{\begin{aligned} W_0 &= \left(f(x_i, y_j)\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1} \\ W_{j+1} &= A * W_j + b_j \end{aligned}}$$

$$W_j = \left(W_{i,j}^{l}\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

$$Estabilidad \rightarrow \lambda \leq 1/4$$

Métodos explícitos – Matriz Tridiagonal

$$W_{i,j}^{l+1} = \lambda\left(W_{i+1,j}^{l} + W_{i-1,j}^{l} + W_{i,j+1}^{l} + W_{i,j-1}^{l}\right) + \left(1-4\lambda\right) * W_{i,j}^{l} + k * F_{i,j}^{l}$$



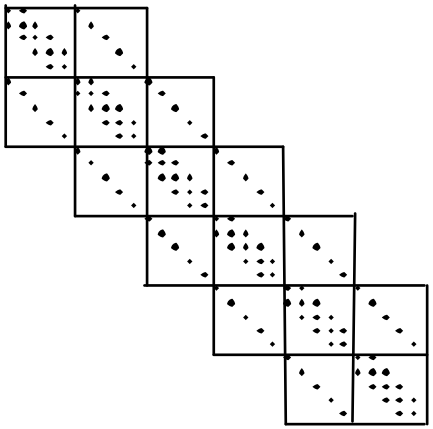
$$\begin{bmatrix} A & C & 0 \\ B & A & C \\ 0 & B & A \end{bmatrix}$$

$$W_0 = \left(f(x_i, y_j)\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

$$W_{j+1} = A * W_j + b_j$$

$$\left[\begin{array}{ccc|ccc|ccc}
1-4\lambda & \lambda & 0 & \lambda & 0 & 0 & 0 & 0 & 0 \\
\lambda & 1-4\lambda & \lambda & 0 & \lambda & 0 & 0 & 0 & 0 \\
0 & \lambda & 1-4\lambda & 0 & 0 & \lambda & 0 & 0 & 0 \\
\hline
\lambda & 0 & 0 & \ldots\ldots & \ldots\ldots & \ldots\ldots & \lambda & 0 & 0 \\
0 & \lambda & 0 & \ldots\ldots & \ldots\ldots & \ldots\ldots & 0 & \lambda & 0 \\
0 & 0 & \lambda & \ldots\ldots & \ldots\ldots & \ldots\ldots & 0 & 0 & \lambda \\
\hline
0 & 0 & 0 & \lambda & 0 & 0 & 1-4\lambda & \lambda & 0 \\
0 & 0 & 0 & 0 & \lambda & 0 & \lambda & 1-4\lambda & \lambda \\
0 & 0 & 0 & 0 & 0 & \lambda & 0 & \lambda & 1-4\lambda
\end{array}\right]$$

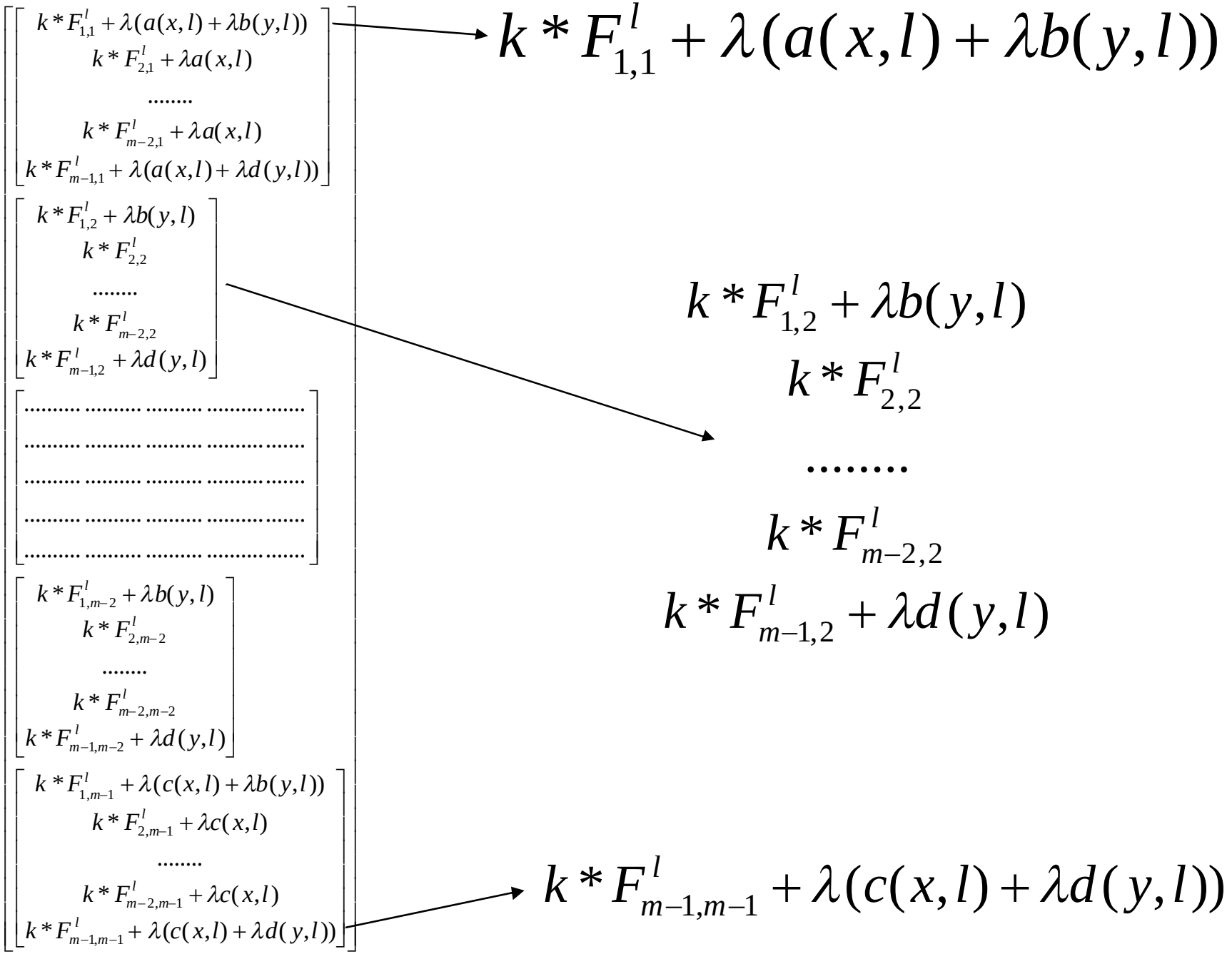
Programación Matriz Tridiagonal



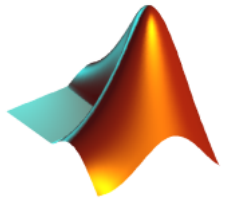
```
function AA=tridiagonal2D(a,b,c,m)
A=tridiagonal(a,b,c,m);
B=tridiagonal(b,0,0,m);
C=tridiagonal(c,0,0,m);
AA=zeros(m*m,m*m);
for i=2:m-1
    AA(((i-1)*m)+1:i*m,((i-2)*m)+1:(i-1)*m)=B;
    AA(((i-1)*m)+1:i*m,((i-1)*m)+1:(i+0)*m)=A;
    AA(((i-1)*m)+1:i*m,((i-0)*m)+1:(i+1)*m)=C;
end
AA(1:m,1:m)=A;
AA(1:m,m+1:2*m)=C;
AA(((m-1)*m)+1:m*m,((m-2)*m)+1:(m-1)*m)=B;
AA(((m-1)*m)+1:m*m,((m-1)*m)+1:m*m)=A;
```

```
A=tridiagonal2D(1-4*la,la,la,m-1);
```

Métodos explícitos – Vector Bj



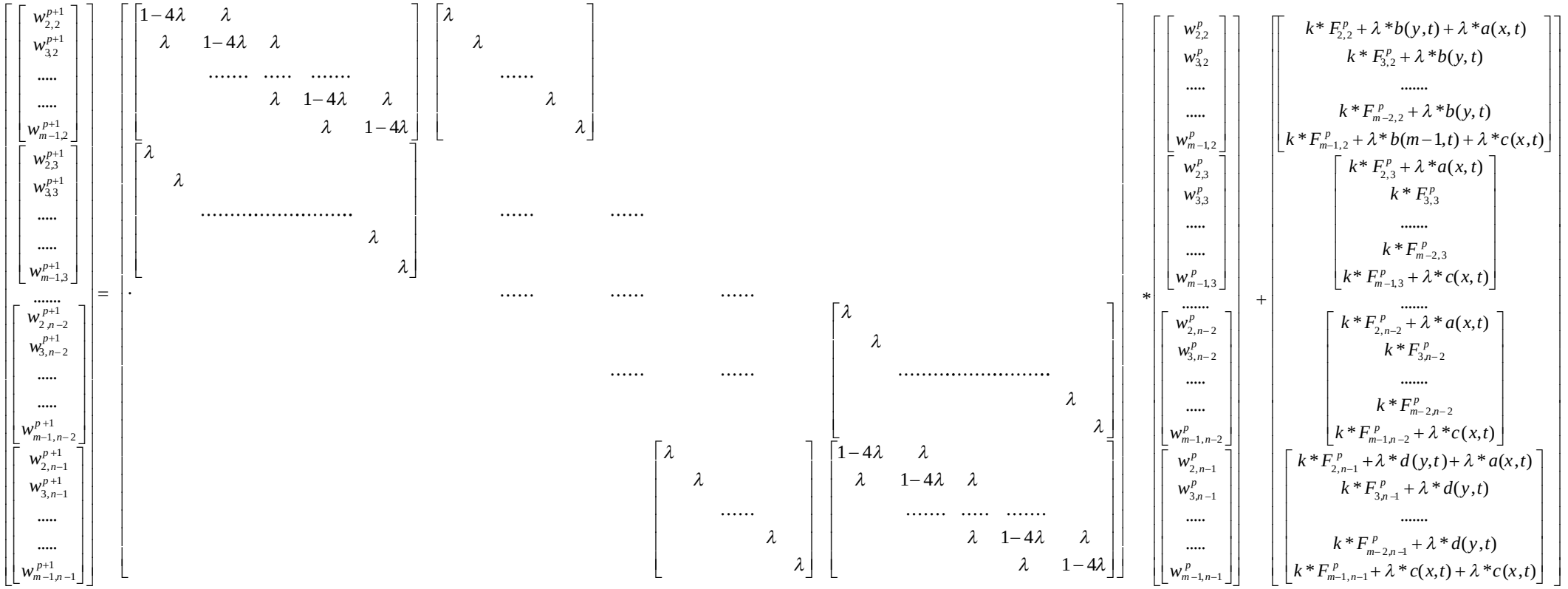
Programación Vector Cj



```
for j=0:N-1
    %construcción de Bj
    b=zeros(m-1,m-1);%lo creamos inicialmente como matriz
    %Fuerza externa
    for i=1:m-1
        for ii=1:m-1
            b(i,ii)=k*GG(i*h,ii*h,j*k);
        end
    end
    %En los bordes de la placa condiciones de Dirichlet
    for i=1:m-1
        b(i,1)=b(i,1)+la*aa(i*h,j*k);
        b(i,m-1)=b(i,m-1)+la*cc(i*h,j*k);
        b(1,i)=b(1,i)+la*bb(i*h,j*k);
        b(m-1,i)=b(m-1,i)+la*dd(i*h,j*k);
    end
end
```

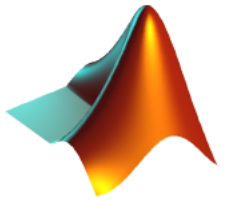
Métodos explícitos – Sistema ec.

W1= A *W0 + bj

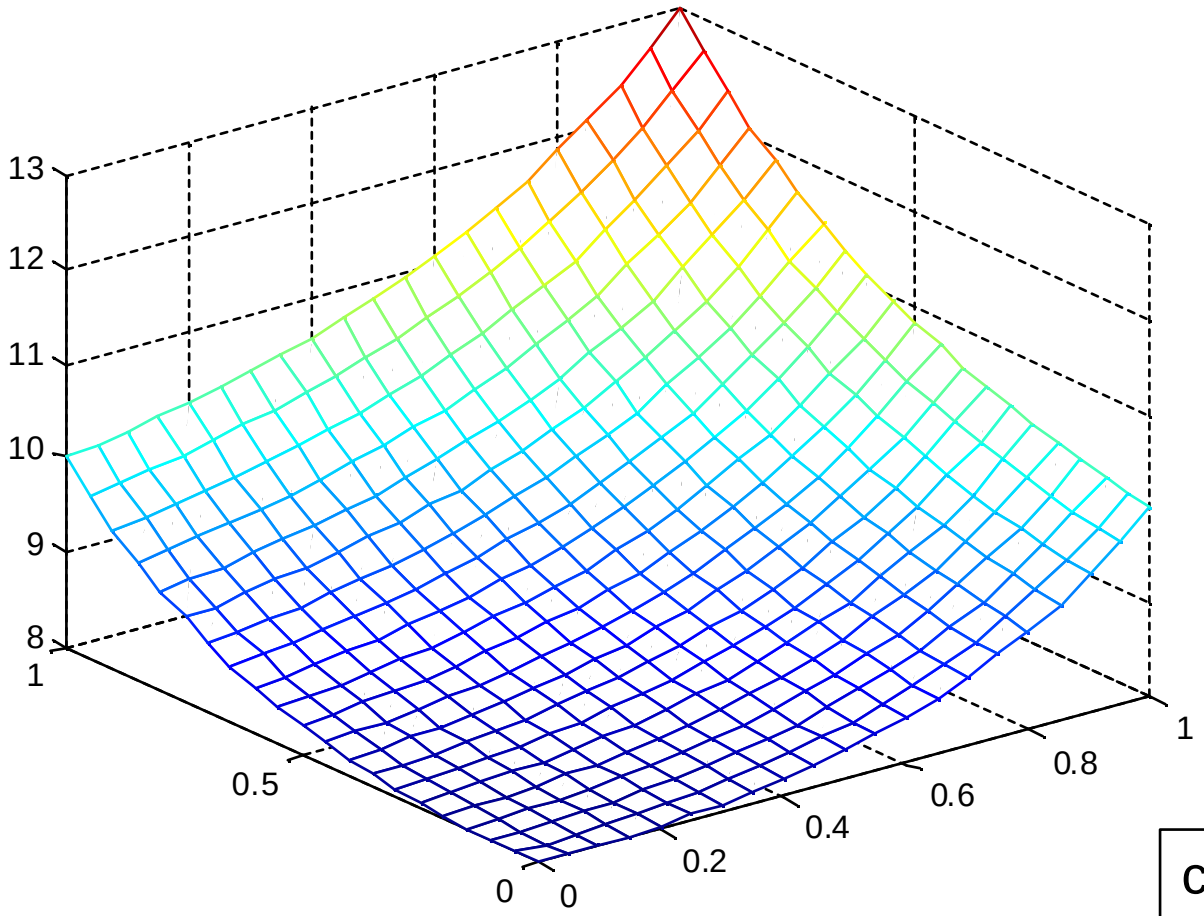


(m-1)*(m-1) ecuaciones

```
B=zeros((m-1)*(m-1),1);%lo convertimos en un vector de pie
  for i=1:m-1
      B(1+((i-1)*(m-1)):i*(m-1),1)=b(:,i);
  end
```

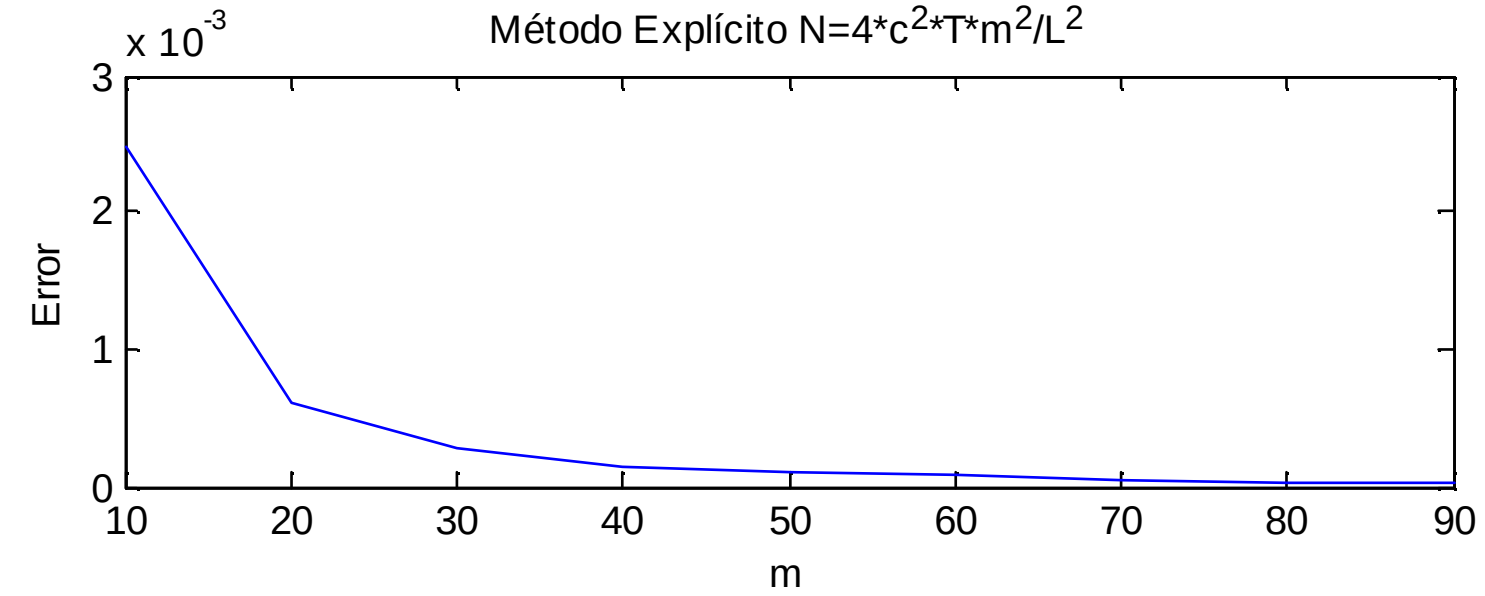


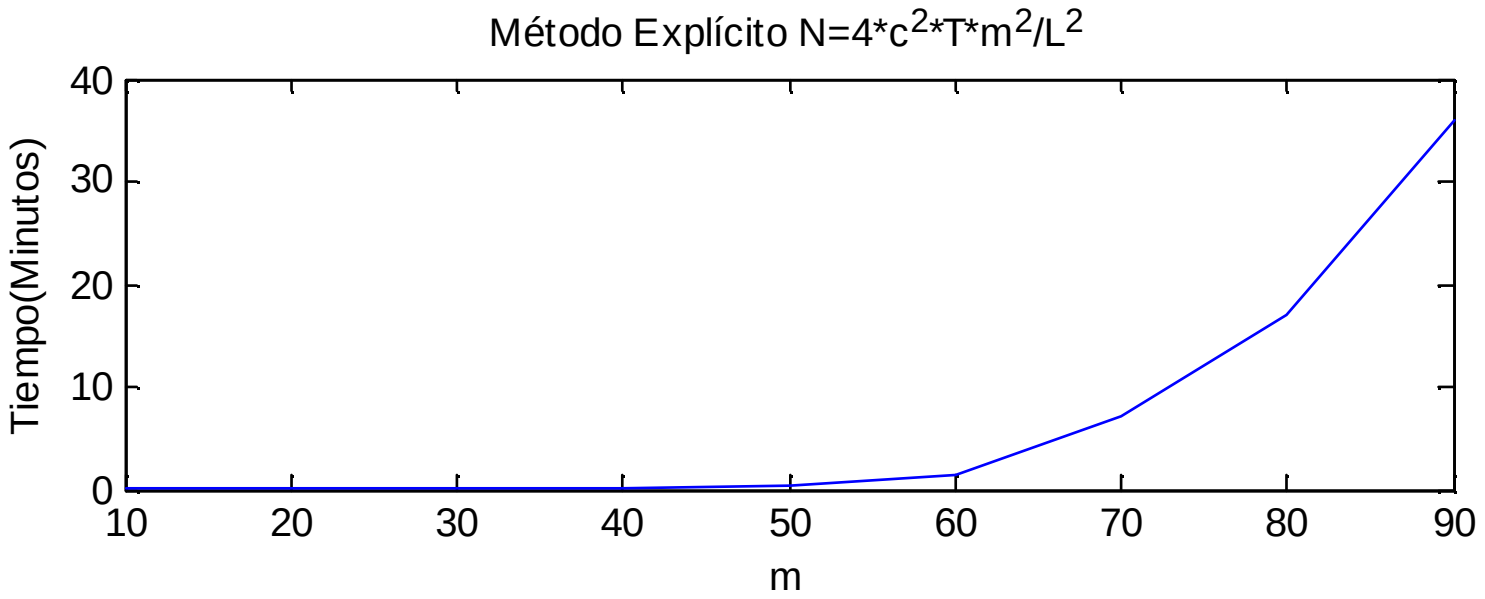
Métodos explícitos – Soluciones



c=1;L=1;T=1;
m=90;
h = 0.011111111111111
k = 3.086419753086420e-005
la = 0.250000000000000
error = 0.000030673019068

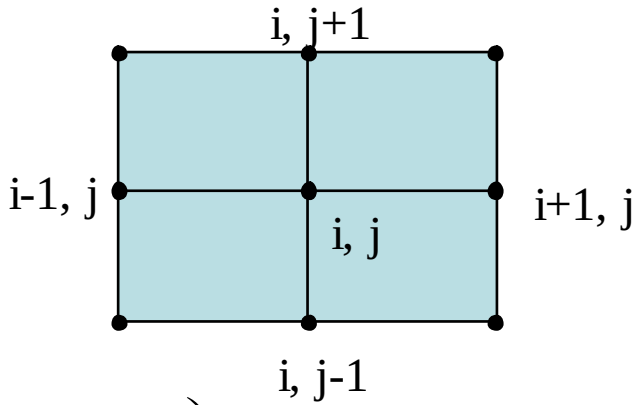
Métodos explícitos – Soluciones





Métodos Implícito - Condiciones Dirichlet

$$h = \frac{L}{m} \qquad k = \frac{T}{N} \qquad \lambda = \frac{c^2 * k}{h^2}$$



$$\frac{W_{i,j}^{l+1} - W_{i,j}^{l}}{k} = c^2 \left(\frac{W_{i+1,j}^{l+1} - 2W_{i,j}^{l+1} + W_{i-1,j}^{l+1}}{h^2} + \frac{W_{i,j+1}^{l+1} - 2W_{i,j}^{l+1} + W_{i,j-1}^{l+1}}{h^2} \right) + F_{i,j}^{l+1}$$

$$\boxed{(1+4\lambda) * W_{i,j}^{l+1} - \lambda \left(W_{i+1,j}^{l+1} + W_{i-1,j}^{l+1} + W_{i,j+1}^{l+1} + W_{i,j-1}^{l+1} \right) = W_{i,j}^{l} + k * F_{i,j}^{l+1}}$$

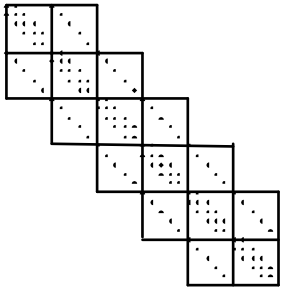
$$\boxed{\begin{aligned} W_0 &= \left(f(x_i, y_j) \right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1} \\ A * W_{j+1} &= W_j + c_j \end{aligned}}$$

$$W_j = \left(W_{i,j}^{l} \right)_{l=1,\ldots,N-1}^{i=1,\ldots,m-1}{}^{j=1,\ldots,m-1}$$

Incondicionalmente estable

Métodos implícito – Matriz Tridiagonal

$$(1+4\lambda)*W_{i,j}^{l+1}-\lambda\left(W_{i+1,j}^{l+1}+W_{i-1,j}^{l+1}+W_{i,j+1}^{l+1}+W_{i,j-1}^{l+1}\right)=W_{i,j}^{l}+k*F_{i,j}^{l+1}$$



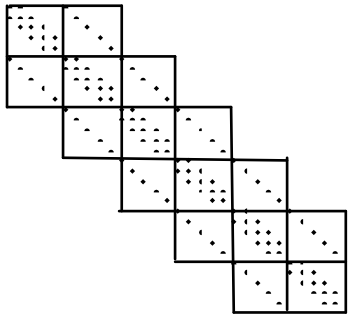
$$\left[\begin{array}{c|c|c} A & C & 0 \\ \hline B & A & C \\ \hline 0 & B & A \end{array}\right]$$

$$W_0=\left(f(x_i,y_j)\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

$$A*W_{j+1}=W_j+c_j$$

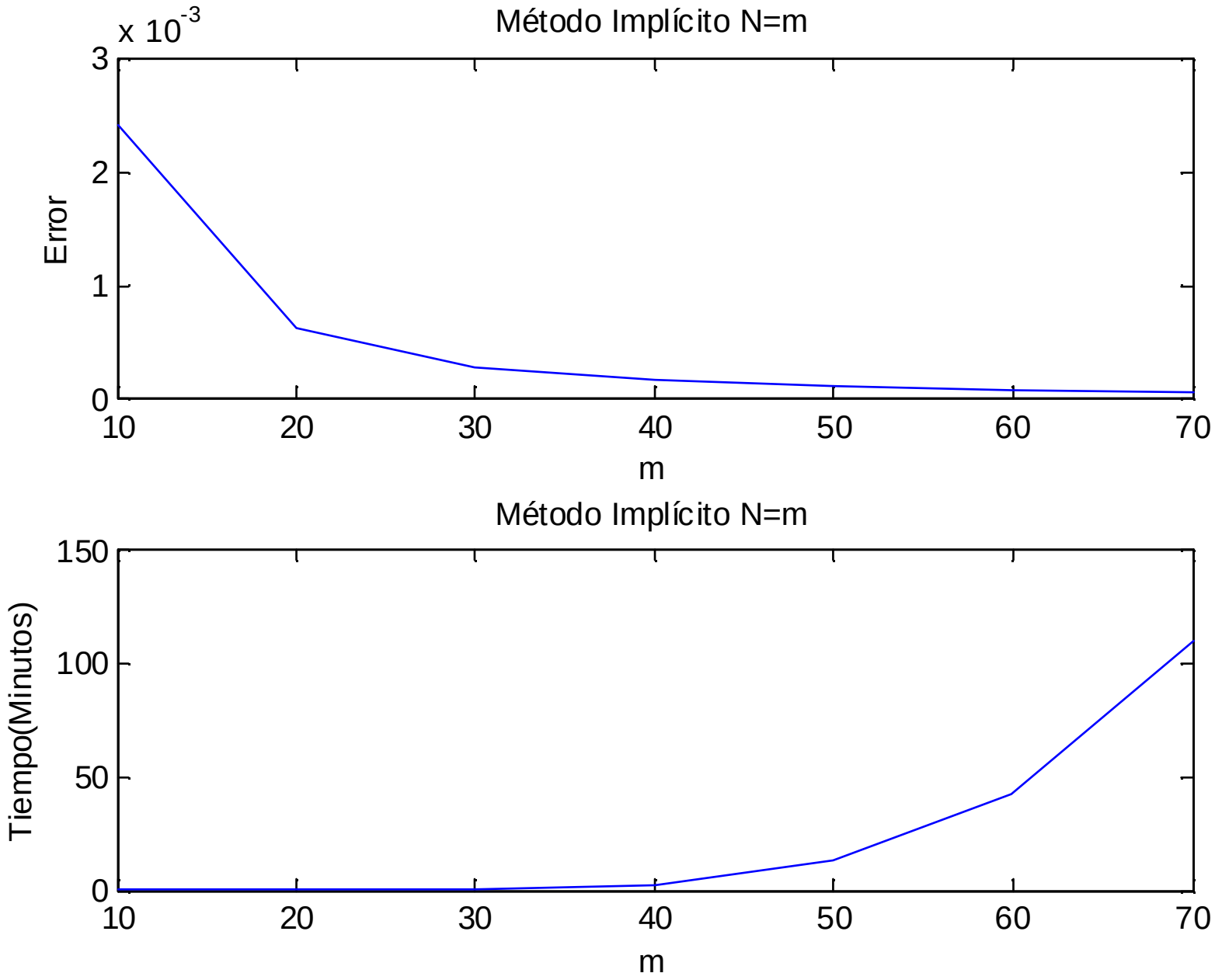
$$\left[\begin{array}{ccc|ccc|ccc} 1+4\lambda & -\lambda & 0 & -\lambda & 0 & 0 & 0 & 0 & 0 \\ -\lambda & 1+4\lambda & -\lambda & 0 & -\lambda & 0 & 0 & 0 & 0 \\ 0 & -\lambda & 1+4\lambda & 0 & 0 & -\lambda & 0 & 0 & 0 \\ \hline -\lambda & 0 & 0 & \ldots\ldots & \ldots\ldots & \ldots\ldots & -\lambda & 0 & 0 \\ 0 & -\lambda & 0 & \ldots\ldots & \ldots\ldots & \ldots\ldots & 0 & -\lambda & 0 \\ 0 & 0 & -\lambda & \ldots\ldots & \ldots\ldots & \ldots\ldots & 0 & 0 & -\lambda \\ \hline 0 & 0 & 0 & -\lambda & 0 & 0 & 1+4\lambda & -\lambda & 0 \\ 0 & 0 & 0 & 0 & -\lambda & 0 & -\lambda & 1+4\lambda & -\lambda \\ 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & -\lambda & 1+4\lambda \end{array}\right]$$

Programación Matriz Tridiagonal



```
function WN=calorimplicito2D(m,N,L,c,T)
%m=10;N=10;L=1;c=1;T=1;
k=T/N;h=L/m;la=c^2*k/h^2
%definir A
A=tridiagonal2D(1+4*la,-la,-la,m-1);
%calculo de W0
…….
%Calculo W1,W2,....,WN
……..
%En los bordes de la placa se aplican condiciones de Dirichlet
……...
%lo convertimos en un vector de pie
……..
%Cálculo de W1,..,WN
    W=A\(W+C);
end
```

Métodos implícitos – Soluciones



Métodos Implicito - CranK - Nicolson

$$\frac{W_{i,j}^{l+1}-W_{i,j}^{l}}{k}=\frac{c^2}{2}\left(\frac{W_{i+1,j}^{l+1}-2W_{i,j}^{l+1}+W_{i-1,j}^{l+1}}{h^2}+\frac{W_{i,j+1}^{l+1}-2W_{i,j}^{l+1}+W_{i,j-1}^{l+1}}{h^2}\right)+$$

$$+\frac{c^2}{2}\left(\frac{W_{i+1,j}^{l}-2W_{i,j}^{l}+W_{i-1,j}^{l}}{h^2}+\frac{W_{i,j+1}^{l}-2W_{i,j}^{l}+W_{i,j-1}^{l}}{h^2}\right)+F_{i,j}^{l+1/2}$$

$$(1+2\lambda)*W_{i,j}^{l+1}-\frac{\lambda}{2}\left(W_{i+1,j}^{l+1}+W_{i-1,j}^{l+1}+W_{i,j+1}^{l+1}+W_{i,j-1}^{l+1}\right)=$$

$$(1-2\lambda)*W_{i,j}^{l}+\frac{\lambda}{2}\left(W_{i+1,j}^{l}+W_{i-1,j}^{l}+W_{i,j+1}^{l}+W_{i,j-1}^{l}\right)+k*F_{i,j}^{l+1/2}$$

$$\lambda=\frac{c^2*k}{h^2}$$

$$W_0=\left(f(x_i,y_j)\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

$$A*W_{j+1}=B*W_j+c_j$$

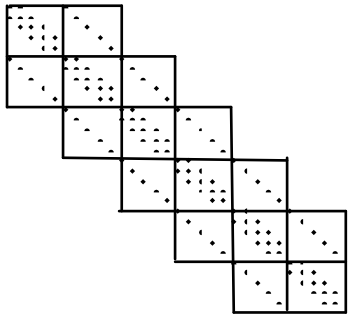
$$W_j=\left(W_{i,j}^{l}\right)_{l=1,\ldots,N-1}^{i=1,\ldots,m-1\ j=1,\ldots,m-1}$$

Incondicionalmente estable

CranK - Nicolson– Vector Cj

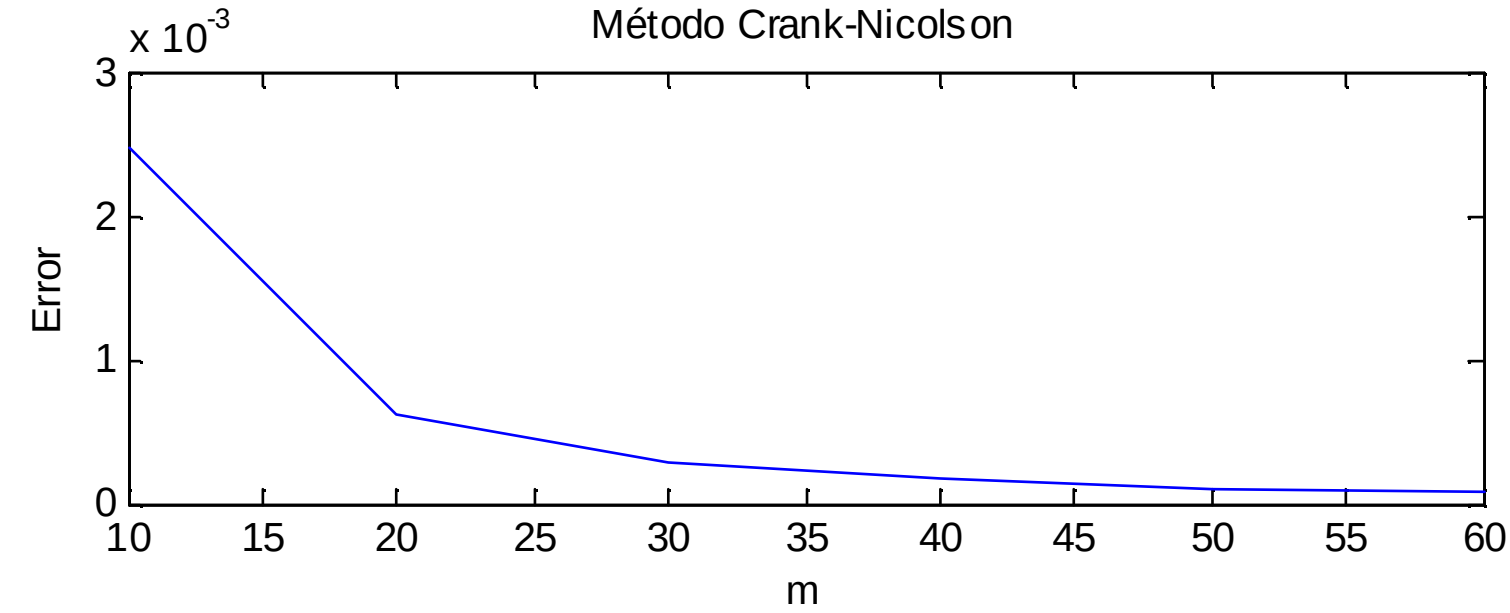
$$
\begin{bmatrix}
\begin{bmatrix}
k*F_{1,1}^{l}+\frac{\lambda}{2}(a(x,l)+a(x,l+1))+\frac{\lambda}{2}(b(x,l)+b(x,l+1)) \\
k*F_{2,1}^{l}+\frac{\lambda}{2}(a(x,l)+a(x,l+1)) \\
\cdots\cdots\cdots \\
k*F_{m-2,1}^{l}+\frac{\lambda}{2}(a(x,l)+a(x,l+1)) \\
k*F_{m-1,1}^{l}+\frac{\lambda}{2}(a(x,l)+a(x,l+1))+\frac{\lambda}{2}(d(x,l)+d(x,l+1))
\end{bmatrix} \\
\begin{bmatrix}
k*F_{1,2}^{l}+\frac{\lambda}{2}(b(x,l)+b(x,l+1)) \\
k*F_{2,2}^{l} \\
\cdots\cdots\cdots \\
k*F_{m-2,2}^{l} \\
k*F_{m-1,2}^{l}+\frac{\lambda}{2}(d(x,l)+d(x,l+1))
\end{bmatrix} \\
\begin{bmatrix}
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots
\end{bmatrix} \\
\begin{bmatrix}
k*F_{1,m-2}^{l}+\frac{\lambda}{2}(b(x,l)+b(x,l+1)) \\
k*F_{2,m-2}^{l} \\
\cdots\cdots\cdots \\
k*F_{m-2,m-2}^{l} \\
k*F_{m-1,m-2}^{l}+\frac{\lambda}{2}(d(x,l)+d(x,l+1))
\end{bmatrix} \\
\begin{bmatrix}
k*F_{1,m-1}^{l}+\frac{\lambda}{2}(c(x,l)+c(x,l+1))+\frac{\lambda}{2}(b(x,l)+b(x,l+1)) \\
k*F_{2,m-1}^{l}+\frac{\lambda}{2}(c(x,l)+c(x,l+1)) \\
\cdots\cdots\cdots \\
k*F_{m-2,m-1}^{l}+\frac{\lambda}{2}(c(x,l)+c(x,l+1)) \\
k*F_{m-1,m-1}^{l}+\frac{\lambda}{2}(c(x,l)+c(x,l+1))+\frac{\lambda}{2}(d(x,l)+d(x,l+1))
\end{bmatrix}
\end{bmatrix}
$$

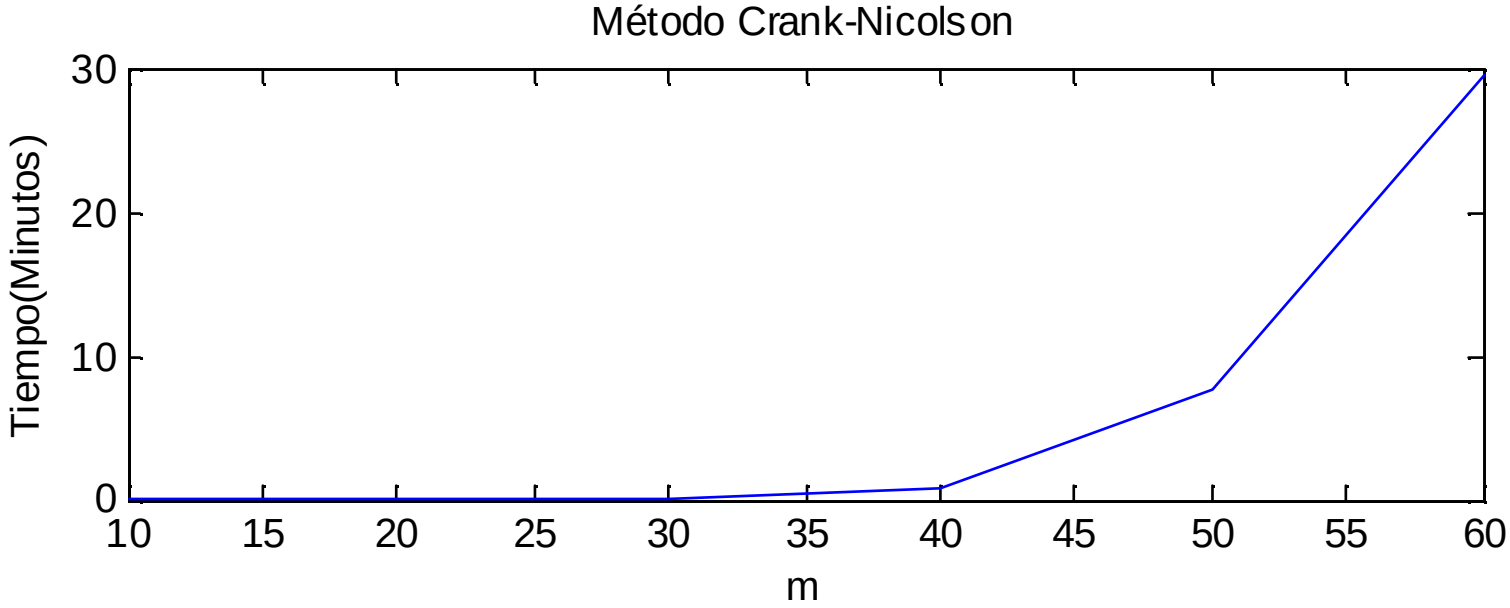
Programación Matriz Tridiagonal



```
function WN=calorimplicito2D(m,N,L,c,T)
%m=10;N=10;L=1;c=1;T=1;
k=T/N;h=L/m;la=c^2*k/h^2
%definir A
A=tridiagonal2D(1+2*la,-la/2,-la/2,m-1);
B=tridiagonal2D(1-2*la,la/2,la/2,m-1);
%calculo de W0 como un vector columna
…….
%Calculo W1,W2,....,WN
……..
%En los bordes de la placa se aplican condiciones de Dirichlet
……...
%lo convertimos en un vector de pie
……..
%Cálculo de W1,..,WN
    W=A\(B*W+C);
end
```

CranK - Nicolson– Soluciones

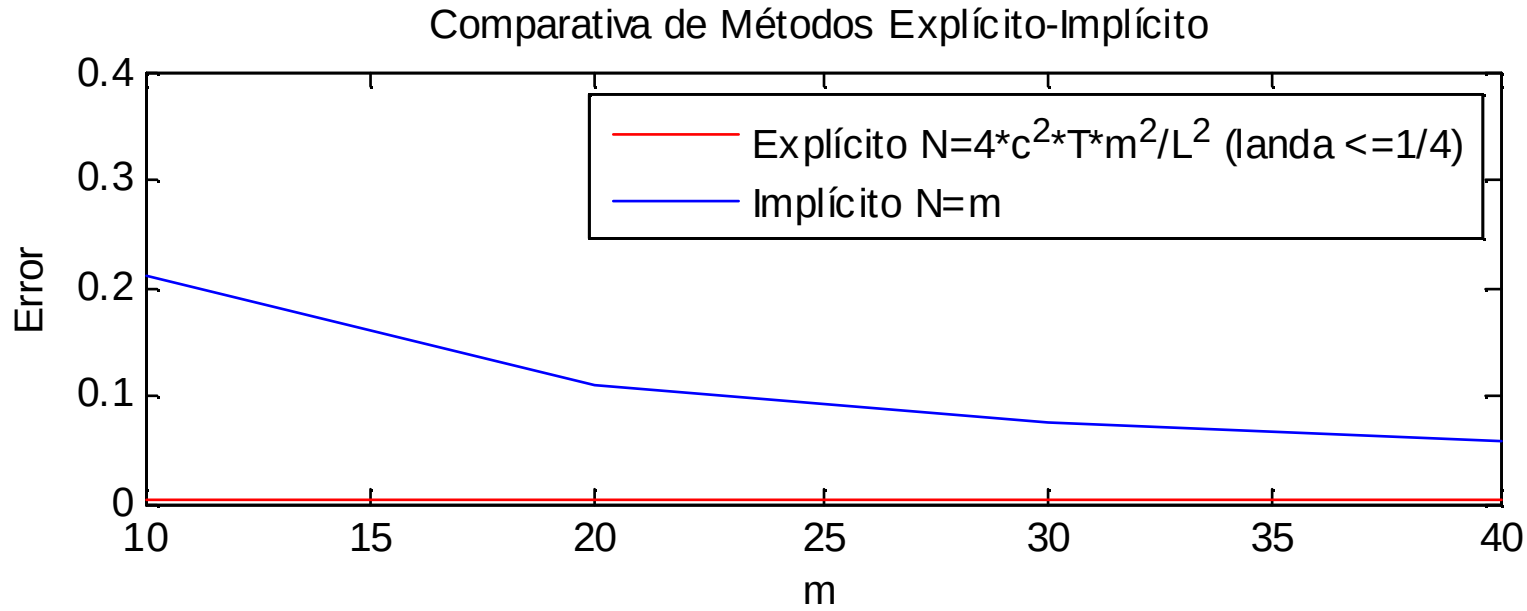


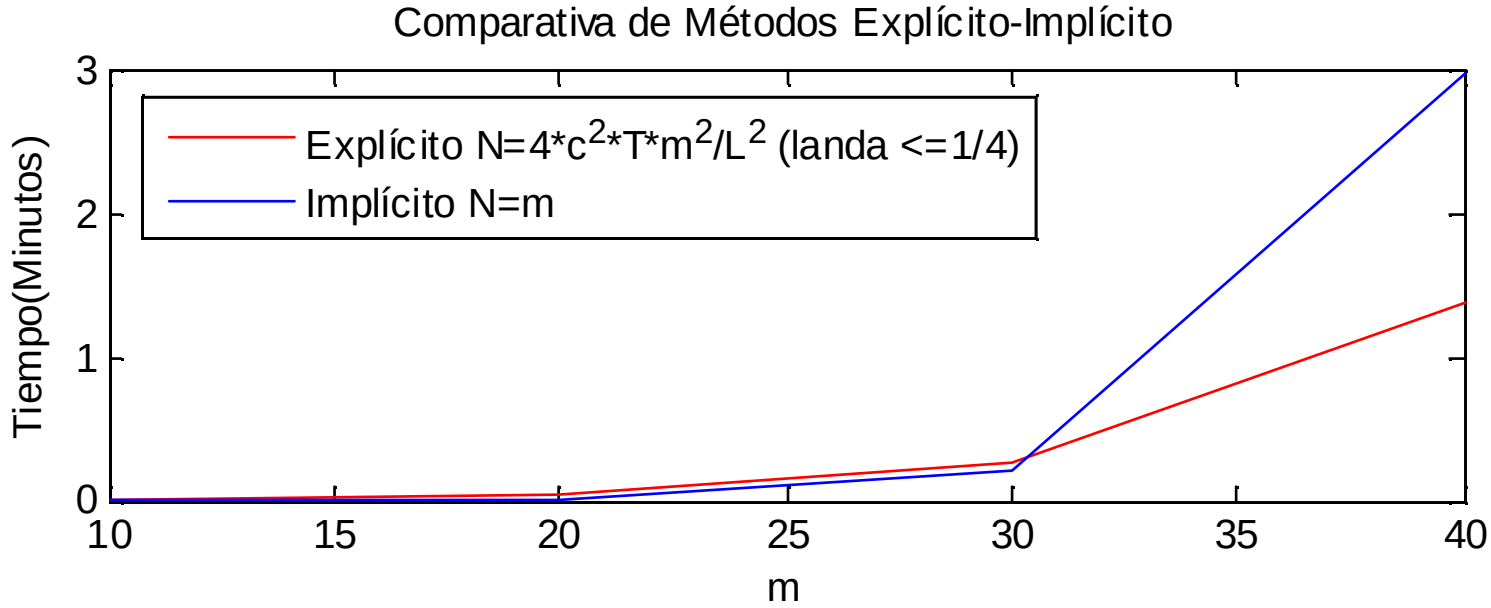


COMPARACIÓN DE LOS MÉTODOS

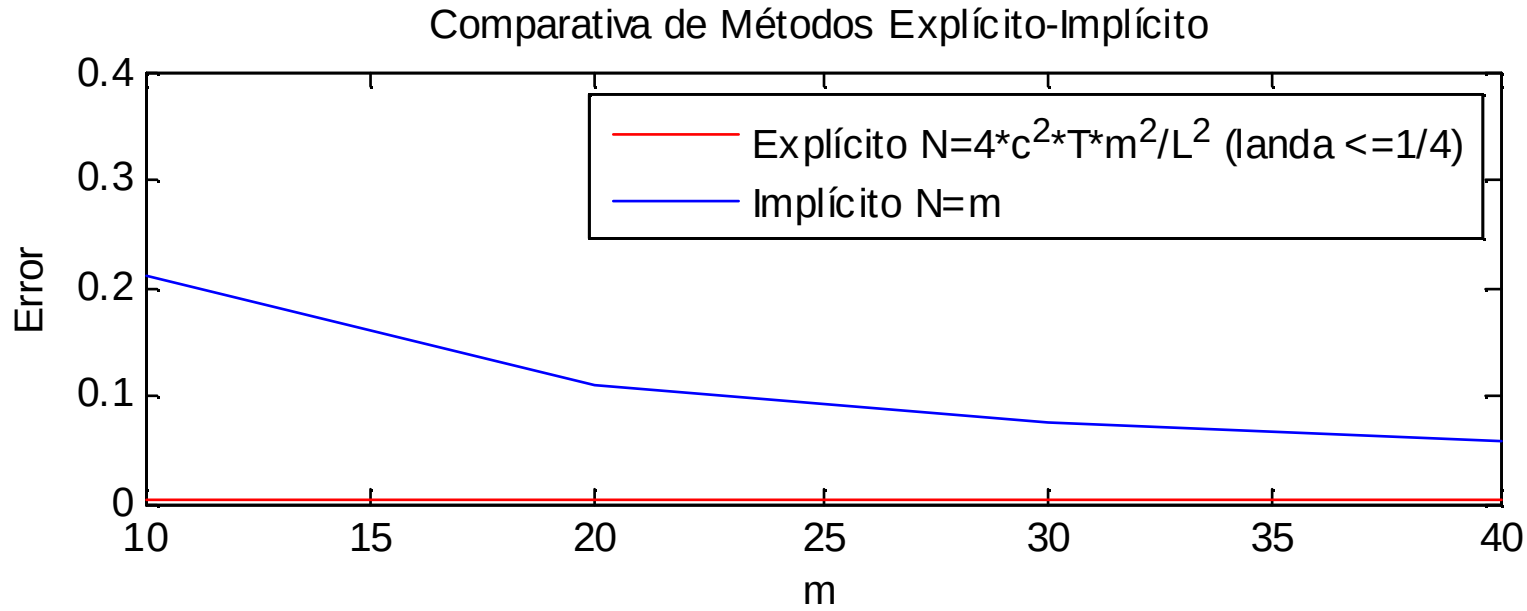
Explícito	**Implícito**	**Crank-Nicolson**
Solución directa	Sistema de ecuaciones	Sistema de ecuaciones
Condicionalmente estable	Incondicionalmente estable	Incondicionalmente estable
Segundo orden en espacio $O(\Delta x^2)$ y primer orden en tiempo $O(\Delta t)$	Segundo orden en espacio $O(\Delta x^2)$ y primer orden en tiempo $O(\Delta t)$	Segundo orden en espacio y en tiempo $O(\Delta x^2+\Delta t^2)$

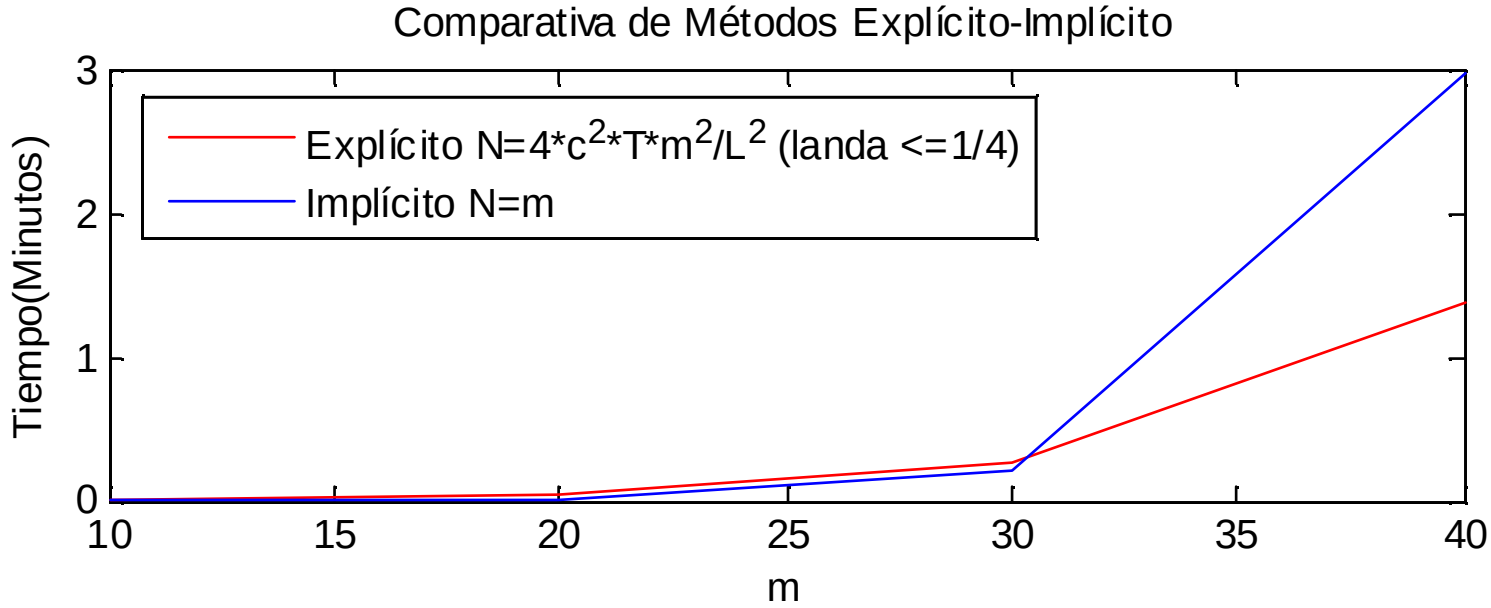
COMPARACIÓN DE LOS MÉTODOS



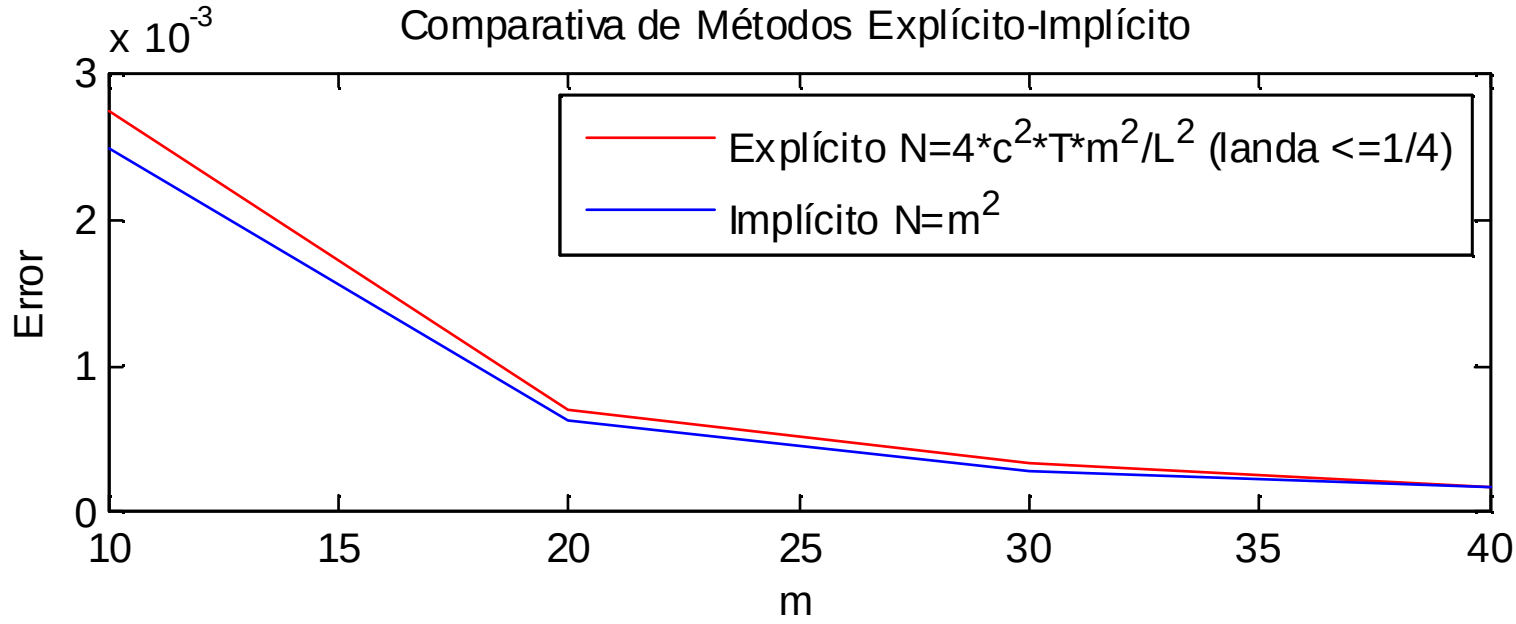


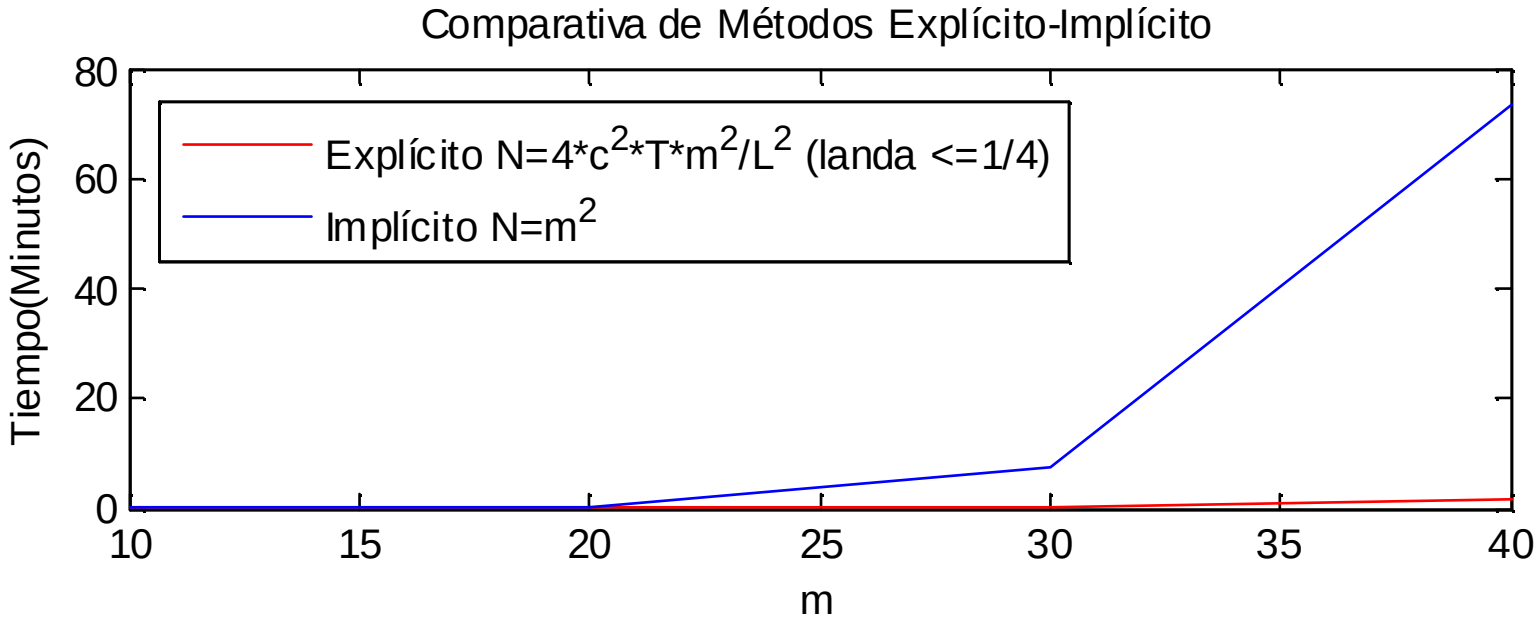
COMPARACIÓN DE LOS MÉTODOS



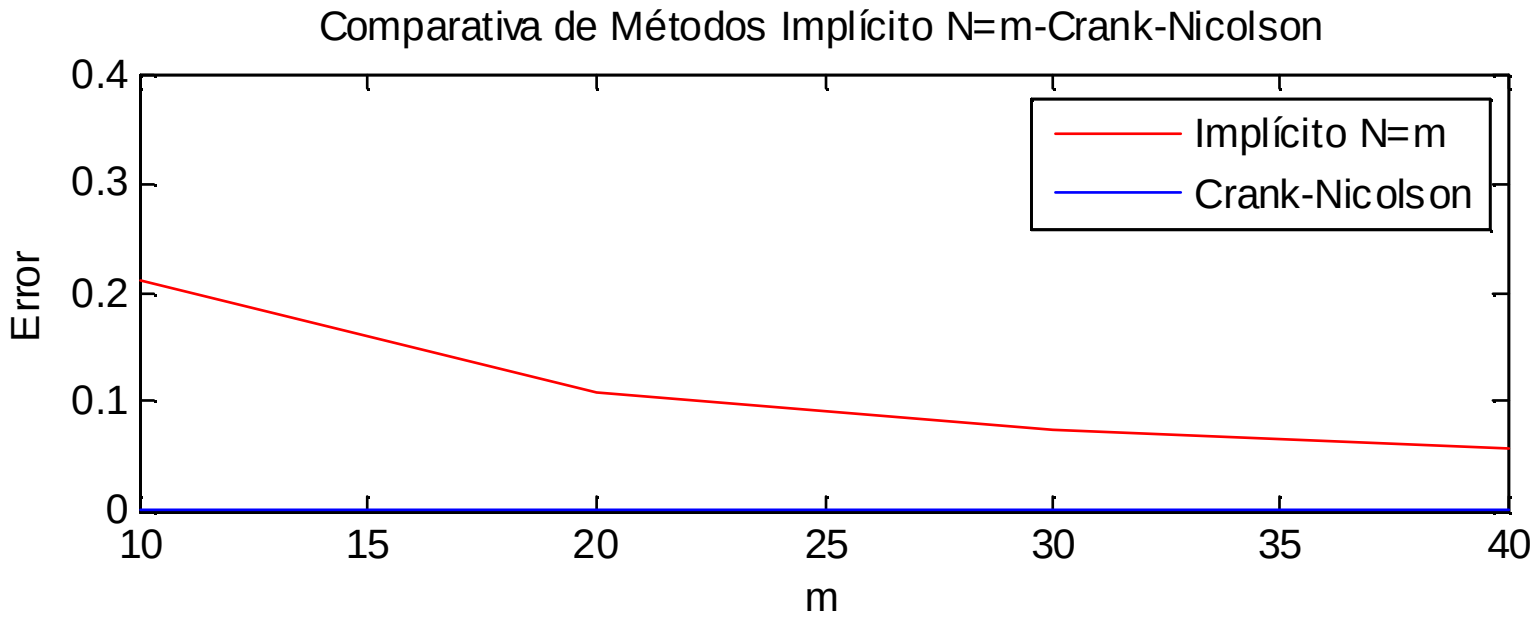


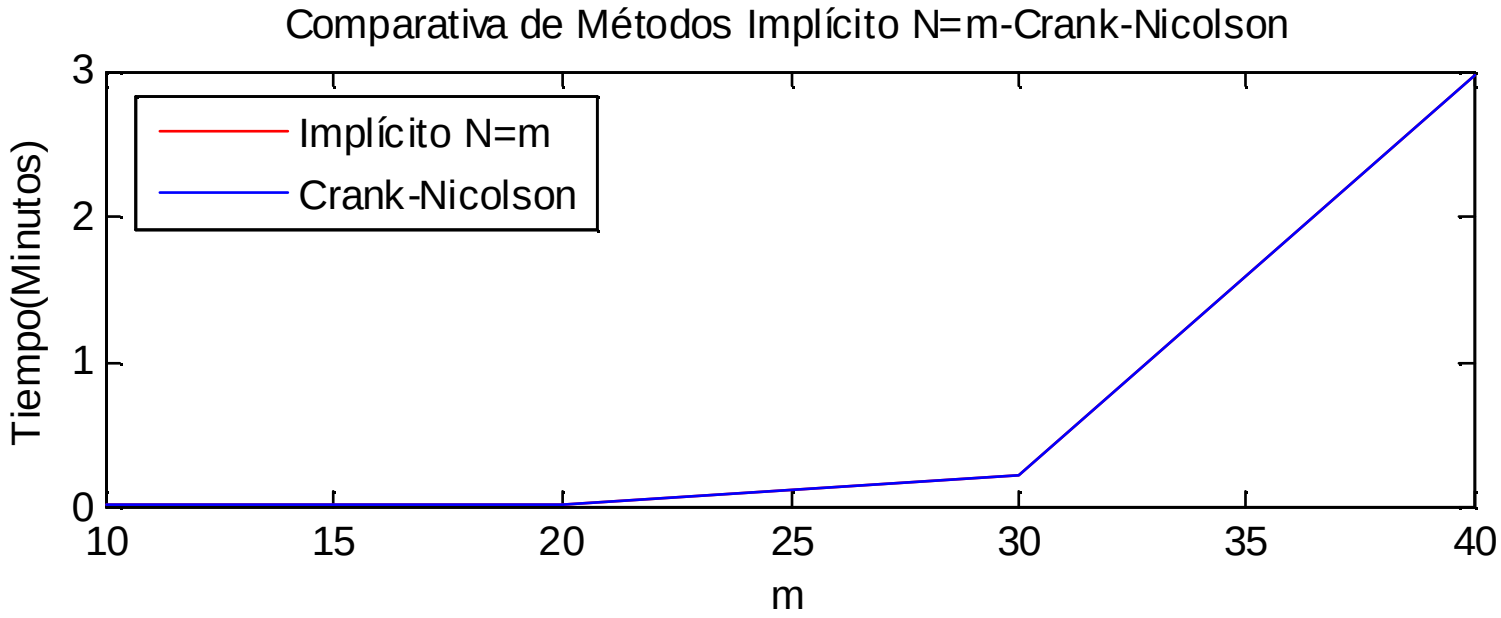
COMPARACIÓN DE LOS MÉTODOS





COMPARACIÓN DE LOS MÉTODOS





COMPARACIÓN DE LOS MÉTODOS

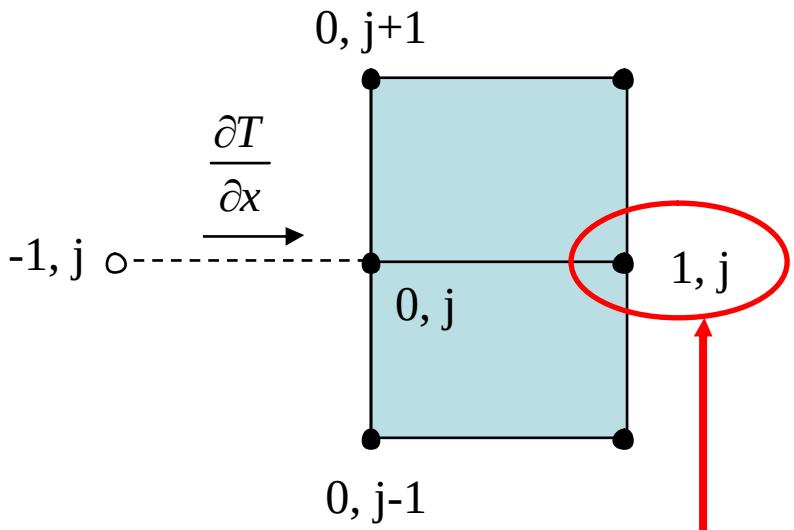
%Implícito N=m

%ERR =	**m**	**error**	**valor obtenido**	**t(seg)**
ERRIM =[	10	0.210527916576604	9.085527916576604	0.078125000000000
	20	0.109648267558974	8.984648267558974	0.953125000000000
	30	0.074041942769215	8.949041942769215	12.21875000000000
	40	**0.055880048754092**	8.930880048754093	178.218750000000];

%Crank-Nicolson

%ERR =	**m**	**error**	**valor obtenido**	**t(seg)**
ERRCR=[	10	0.002662809420785	8.872337190579215	0.109375000000000
	20	0.000671809989671	8.874328190010330	1.031250000000000
	30	0.000299169436829	8.874700830563171	12.34375000000000
	40	**0.000168396595161**	8.874831603404839	178.218750000000];

Condiciones en la frontera de Neumann

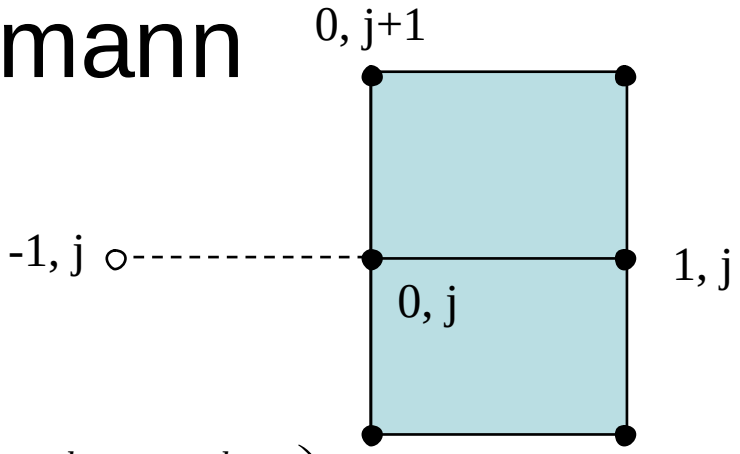


$$\frac{\partial T}{\partial x} \cong \frac{T_{1,j} - T_{-1,j}}{2\Delta x} \rightarrow T_{-1,j} = T_{1,j} - 2\Delta x \frac{\partial T}{\partial x}$$

Estas condiciones generan ecuaciones adicionales para caracterizar a los nodos frontera a los cuales se especifican las derivadas

Métodos explícitos – Condiciones de contorno tipo Neumann

$$h = \frac{L}{m} \qquad k = \frac{T}{N}$$



$$\frac{W_{i,j}^{l+1} - W_{i,j}^{l}}{k} = c^2 \left(\frac{W_{i+1,j}^{l} - 2W_{i,j}^{l} + W_{i-1,j}^{l}}{h^2} + \frac{W_{i,j+1}^{l} - 2W_{i,j}^{l} + W_{i,j-1}^{l}}{h^2} \right) + F_{i,j}^{l}$$

$$W_{i,j}^{l+1} = \lambda \left(W_{i+1,j}^{l} + W_{i-1,j}^{l} + W_{i,j+1}^{l} + W_{i,j-1}^{l} \right) + \left(1 - 4\lambda\right) * W_{i,j}^{l} + k * F_{i,j}^{l}$$

$$\lambda = \frac{c^2 * k}{h^2}$$

$$W_0 = \left(f(x_i, y_j) \right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

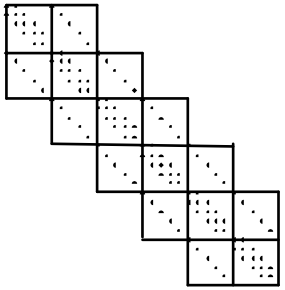
$$W_{j+1} = A * W_j + b_j$$

$$W_j = \left(W_{i,j}^{l} \right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

$$Estabilidad \rightarrow \lambda \leq 1/4$$

Expícito tipo Neumann

$$W_{i,j}^{l+1} = \lambda\left(W_{i+1,j}^{l} + W_{i-1,j}^{l} + W_{i,j+1}^{l} + W_{i,j-1}^{l}\right) + \left(1-4\lambda\right) * W_{i,j}^{l} + k * F_{i,j}^{l}$$



$$\begin{bmatrix} A & C & 0 \\ B & A & C \\ 0 & B & A \end{bmatrix}$$

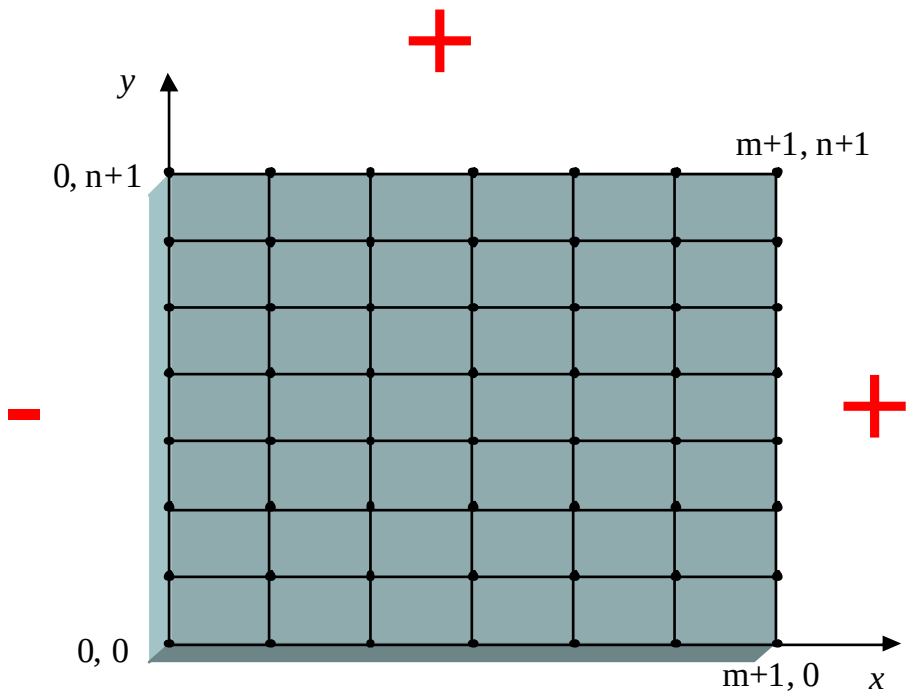
$$W_0 = \left(f(x_i, y_j)\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

$$W_{j+1} = A * W_j + b_j$$

$$\left[\begin{array}{ccc|ccc|ccc}
1-4\lambda & 2\lambda & 0 & 2\lambda & 0 & 0 & 0 & 0 & 0 \\
\lambda & 1-4\lambda & \lambda & 0 & 2\lambda & 0 & 0 & 0 & 0 \\
0 & 2\lambda & 1-4\lambda & 0 & 0 & 2\lambda & 0 & 0 & 0 \\
\hline
\lambda & 0 & 0 & \ldots\ldots & \ldots\ldots & \ldots\ldots & \lambda & 0 & 0 \\
0 & \lambda & 0 & \ldots\ldots & \ldots\ldots & \ldots\ldots & 0 & \lambda & 0 \\
0 & 0 & \lambda & \ldots\ldots & \ldots\ldots & \ldots\ldots & 0 & 0 & \lambda \\
\hline
0 & 0 & 0 & 2\lambda & 0 & 0 & 1-4\lambda & 2\lambda & 0 \\
0 & 0 & 0 & 0 & 2\lambda & 0 & \lambda & 1-4\lambda & \lambda \\
0 & 0 & 0 & 0 & 0 & 2\lambda & 0 & 2\lambda & 1-4\lambda
\end{array}\right]$$

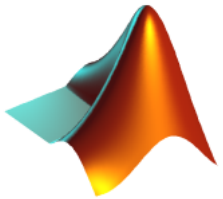
Neumann Explícito– Vector Bj

$$
\begin{bmatrix}
\begin{bmatrix} k*F^{l}_{1,1} - 2\lambda(a(x,l) - 2\lambda b(y,l)) \\ k*F^{l}_{2,1} - 2\lambda a(x,l) \\ \cdots\cdots \\ k*F^{l}_{m-2,1} - 2\lambda a(x,l) \\ k*F^{l}_{m-1,1} - 2\lambda(a(x,l) + 2\lambda d(y,l)) \end{bmatrix} \\
\begin{bmatrix} k*F^{l}_{1,2} - 2\lambda b(y,l) \\ k*F^{l}_{2,2} \\ \cdots\cdots \\ k*F^{l}_{m-2,2} \\ k*F^{l}_{m-1,2} + 2\lambda d(y,l) \end{bmatrix} \\
\begin{bmatrix} \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \end{bmatrix} \\
\begin{bmatrix} k*F^{l}_{1,m-2} - 2\lambda b(y,l) \\ k*F^{l}_{2,m-2} \\ \cdots\cdots \\ k*F^{l}_{m-2,m-2} \\ k*F^{l}_{m-1,m-2} + 2\lambda d(y,l) \end{bmatrix} \\
\begin{bmatrix} k*F^{l}_{1,m-1} + 2\lambda(c(x,l) - 2\lambda b(y,l)) \\ k*F^{l}_{2,m-1} + 2\lambda c(x,l) \\ \cdots\cdots \\ k*F^{l}_{m-2,m-1} + 2\lambda c(x,l) \\ k*F^{l}_{m-1,m-1} + 2\lambda(c(x,l) + 2\lambda d(y,l)) \end{bmatrix}
\end{bmatrix}
$$

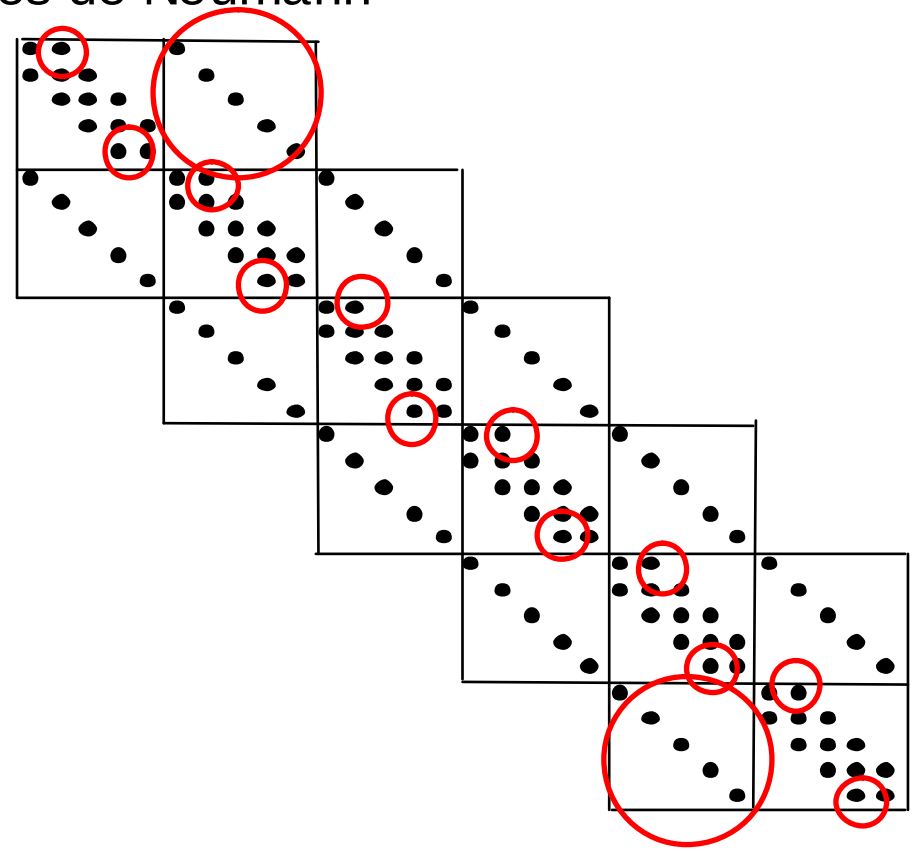


Signos en los extremos

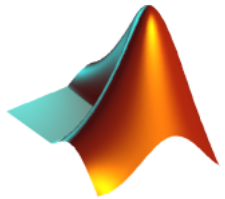
Programación Matriz Tridiagonal



```
function AA=tridiagonalNeumann2D(a,b,c,m)
A=tridiagonal(a,b,c,m);
%modificaciones de la matriz A para condiciones de Neumann
A(1,2)=2*c;
A(m,m-1)=2*c;
B=tridiagonal(b,0,0,m);
C=tridiagonal(c,0,0,m);
AA=zeros(m*m,m*m);
for i=2:m-1
    AA(((i-1)*m)+1:i*m,((i-2)*m)+1:(i-1)*m)=B;
    AA(((i-1)*m)+1:i*m,((i-1)*m)+1:(i+0)*m)=A;
    AA(((i-1)*m)+1:i*m,((i-0)*m)+1:(i+1)*m)=C;
end
%rellenar la fila 1 y la fila m
AA(1:m,1:m)=A;
AA(((m-1)*m)+1:m*m,((m-1)*m)+1:m*m)=A;
%modificaciones de la matriz A para condiciones de Neumann
C=tridiagonal(2*c,0,0,m);
AA(1:m,m+1:2*m)=C;
AA(((m-1)*m)+1:m*m,((m-2)*m)+1:(m-1)*m)=C;
```

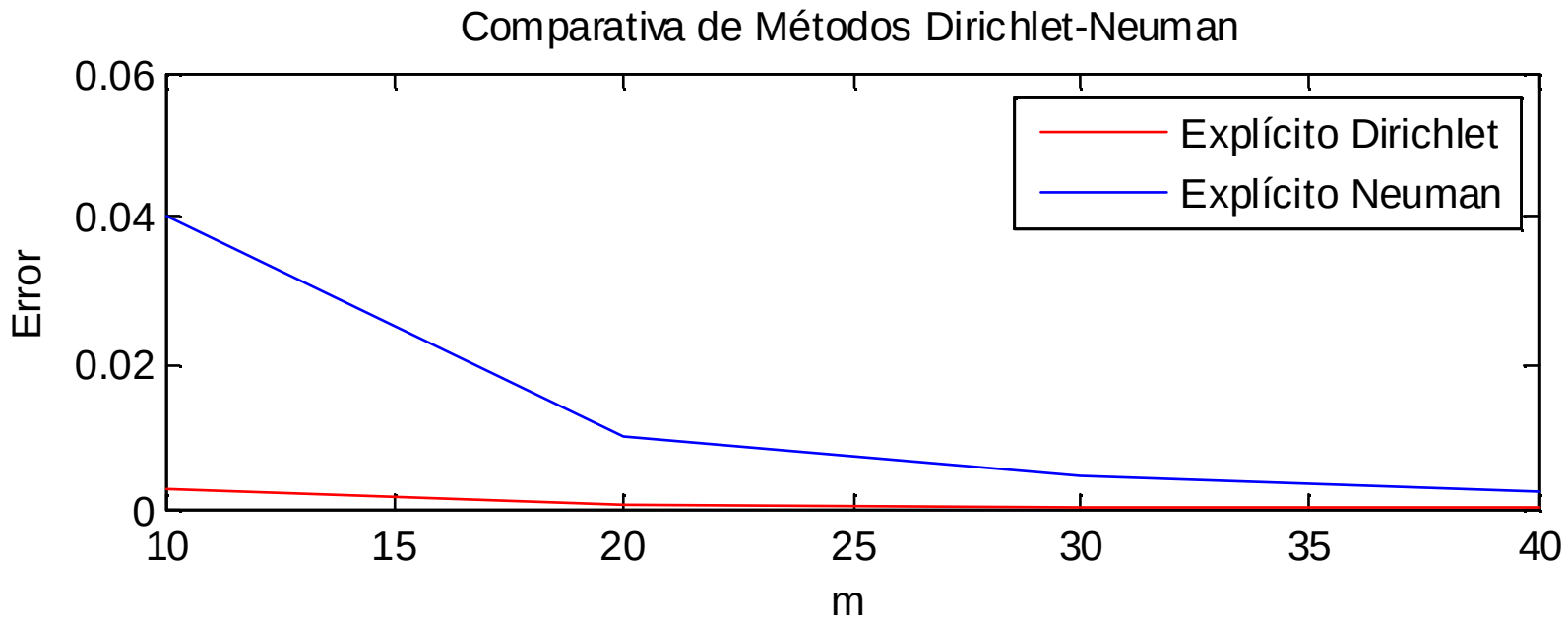


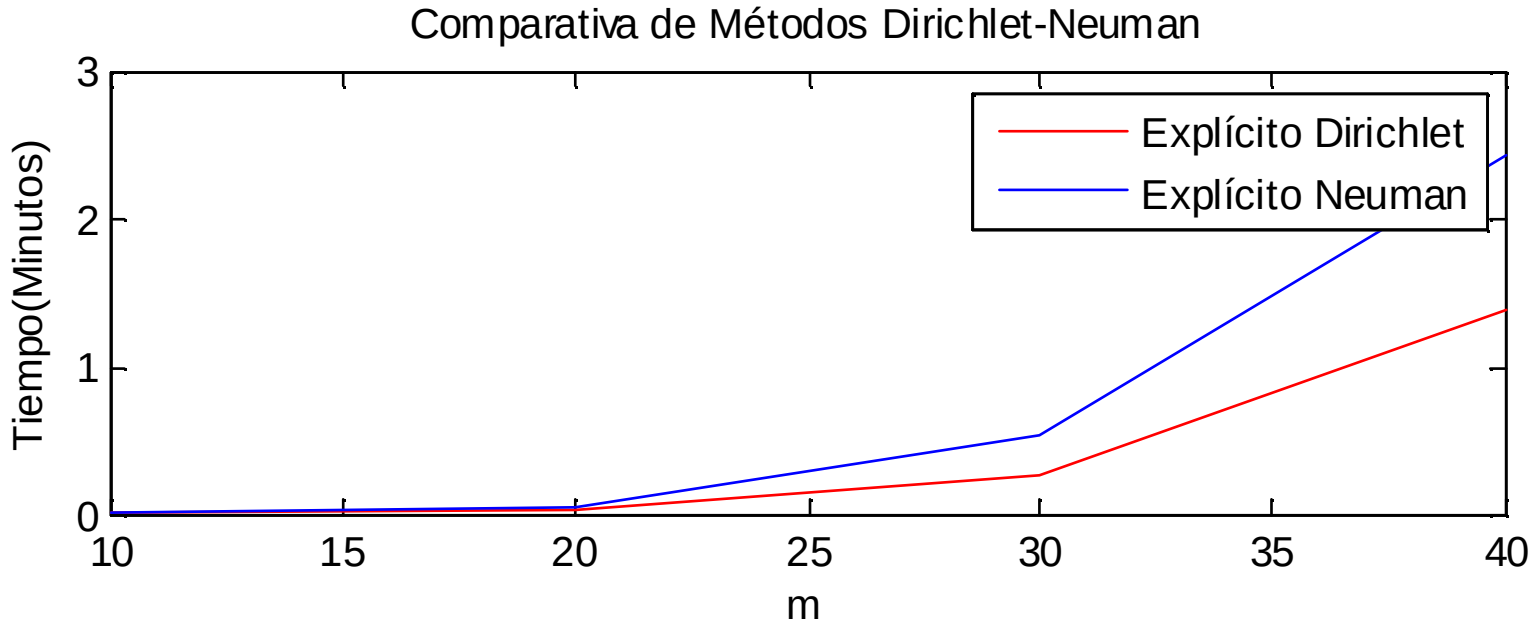
Neumann Explícito - Vector Bj



```
for i=0:m
    for ii=0:m
        b(i+1,ii+1)=k*GG(i*h,ii*h,j*k);
    end
end
%En los bordes de la placa se aplican condiciones de Neumann
for i=0:m
    b(i+1,1)=b(i+1,1)-2*la*h*aa(i*h,j*k);
    b(1,i+1)=b(1,i+1) -2*la*h*bb(i*h,j*k);
    b(i+1,m+1)=b(i+1,m+1)+2*la*h*cc(i*h,j*k);
    b(m+1,i+1)=b(m+1,i+1)+2*la*h*dd(i*h,j*k);
end
```

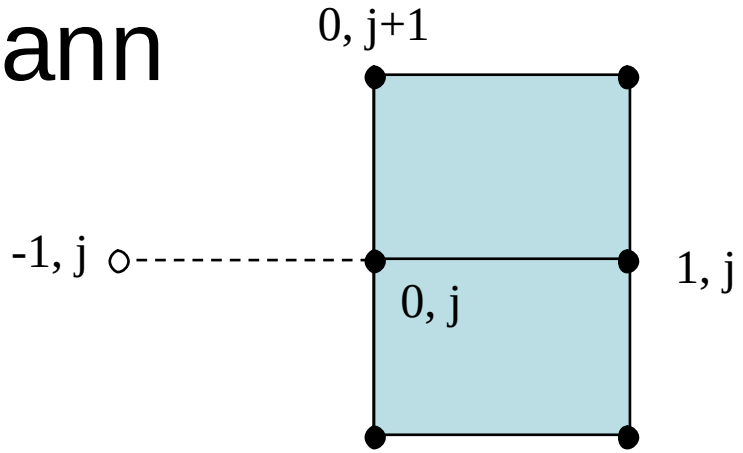
Resultados Neumann-Explícito





Métodos Implícito - Condiciones de contorno tipo Neumann

$$h = \frac{L}{m} \qquad k = \frac{T}{N} \qquad \lambda = \frac{c^2 * k}{h^2}$$



$$\frac{W_{i,j}^{l+1} - W_{i,j}^{l}}{k} = c^2 \left(\frac{W_{i+1,j}^{l+1} - 2W_{i,j}^{l+1} + W_{i-1,j}^{l+1}}{h^2} + \frac{W_{i,j+1}^{l+1} - 2W_{i,j}^{l+1} + W_{i,j-1}^{l+1}}{h^2} \right) + F_{i,j}^{l+1}$$

$$(1+4\lambda) * W_{i,j}^{l+1} - \lambda \left(W_{i+1,j}^{l+1} + W_{i-1,j}^{l+1} + W_{i,j+1}^{l+1} + W_{i,j-1}^{l+1} \right) = W_{i,j}^{l} + k * F_{i,j}^{l+1}$$

$$W_0 = \left(f(x_i, y_j) \right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

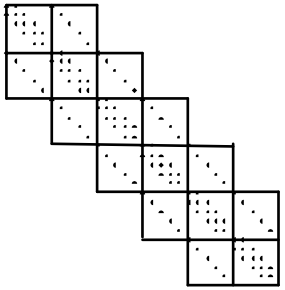
$$A * W_{j+1} = W_j + c_j$$

$$W_j = \left(W_{i,j}^{l} \right)_{l=1,\ldots,N-1}^{i=1,\ldots,m-1,\; j=1,\ldots,m-1}$$

Incondicionalmente estable

Métodos implícito – Matriz Tridiagonal

$$(1+4\lambda)*W_{i,j}^{l+1}-\lambda\left(W_{i+1,j}^{l+1}+W_{i-1,j}^{l+1}+W_{i,j+1}^{l+1}+W_{i,j-1}^{l+1}\right)=W_{i,j}^{l}+k*F_{i,j}^{l+1}$$



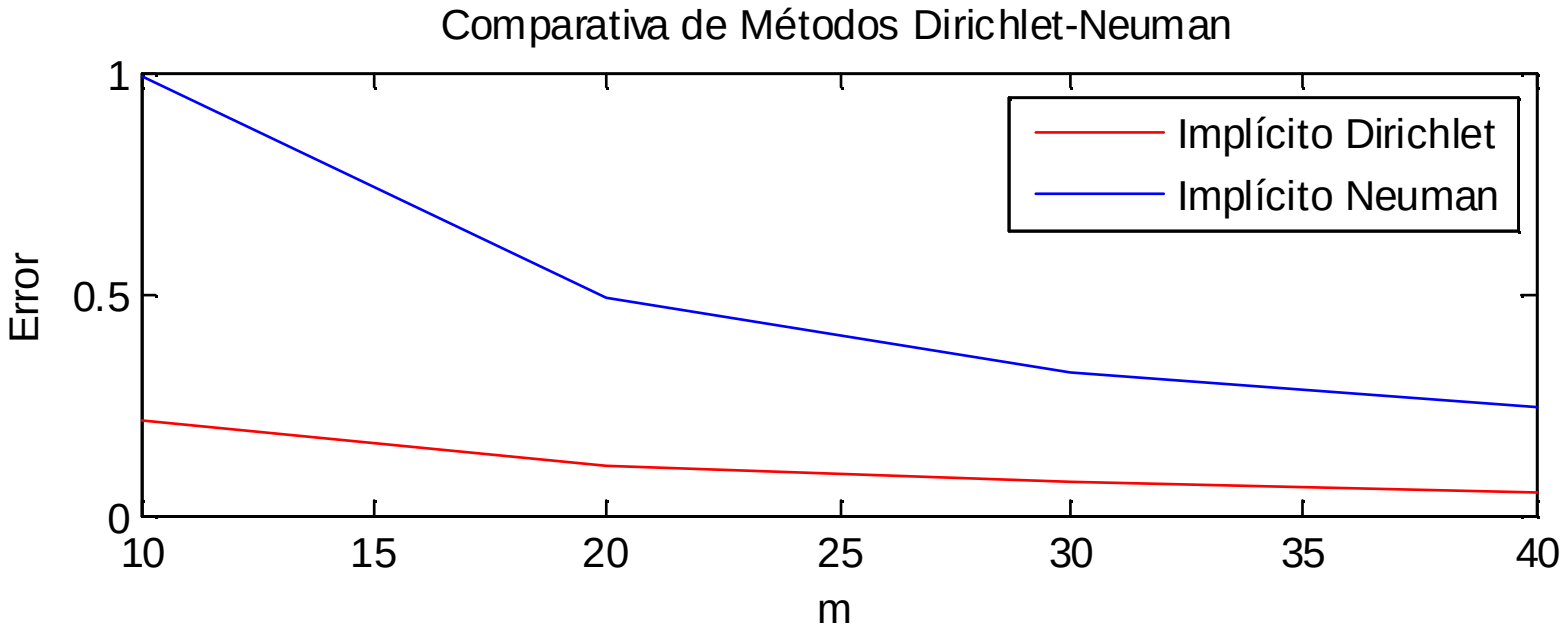
$$\begin{bmatrix} A & C & 0 \\ B & A & C \\ 0 & B & A \end{bmatrix}$$

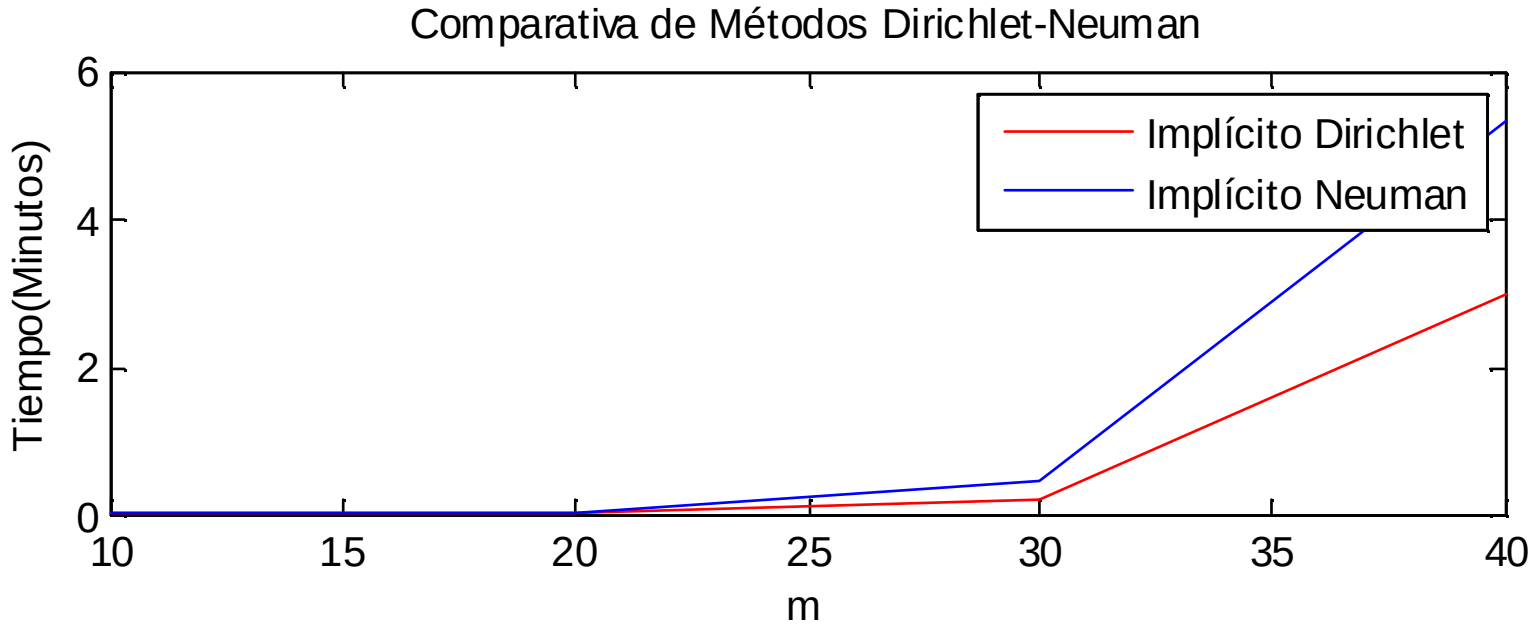
$$W_0=\left(f(x_i,y_j)\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

$$A*W_{j+1}=W_j+c_j$$

$$\left[\begin{array}{ccc|ccc|ccc}
1+4\lambda & -2\lambda & 0 & -2\lambda & 0 & 0 & 0 & 0 & 0 \\
-\lambda & 1+4\lambda & -\lambda & 0 & -2\lambda & 0 & 0 & 0 & 0 \\
0 & -2\lambda & 1+4\lambda & 0 & 0 & -2\lambda & 0 & 0 & 0 \\
\hline
-\lambda & 0 & 0 & \ldots\ldots & \ldots\ldots & \ldots\ldots & -\lambda & 0 & 0 \\
0 & -\lambda & 0 & \ldots\ldots & \ldots\ldots & \ldots\ldots & 0 & -\lambda & 0 \\
0 & 0 & -\lambda & \ldots\ldots & \ldots\ldots & \ldots\ldots & 0 & 0 & -\lambda \\
\hline
0 & 0 & 0 & -2\lambda & 0 & 0 & 1+4\lambda & -2\lambda & 0 \\
0 & 0 & 0 & 0 & -2\lambda & 0 & -\lambda & 1+4\lambda & -\lambda \\
0 & 0 & 0 & 0 & 0 & -2\lambda & 0 & -2\lambda & 1+4\lambda
\end{array}\right]$$

Resultados Neumann-Implícito





CranK - Nicolson - Implicito Neumann

$$\frac{W_{i,j}^{l+1} - W_{i,j}^{l}}{k} = \frac{c^2}{2}\left(\frac{W_{i+1,j}^{l+1} - 2W_{i,j}^{l+1} + W_{i-1,j}^{l+1}}{h^2} + \frac{W_{i,j+1}^{l+1} - 2W_{i,j}^{l+1} + W_{i,j-1}^{l+1}}{h^2}\right) +$$

$$+ \frac{c^2}{2}\left(\frac{W_{i+1,j}^{l} - 2W_{i,j}^{l} + W_{i-1,j}^{l}}{h^2} + \frac{W_{i,j+1}^{l} - 2W_{i,j}^{l} + W_{i,j-1}^{l}}{h^2}\right) + F_{i,j}^{l+1/2}$$

$$(1+2\lambda) * W_{i,j}^{l+1} - \frac{\lambda}{2}\left(W_{i+1,j}^{l+1} + W_{i-1,j}^{l+1} + W_{i,j+1}^{l+1} + W_{i,j-1}^{l+1}\right) =$$

$$(1-2\lambda) * W_{i,j}^{l} + \frac{\lambda}{2}\left(W_{i+1,j}^{l} + W_{i-1,j}^{l} + W_{i,j+1}^{l} + W_{i,j-1}^{l}\right) + k * F_{i,j}^{l+1/2}$$

$$\lambda = \frac{c^2 * k}{h^2}$$

$$W_0 = \left(f(x_i, y_j)\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}$$

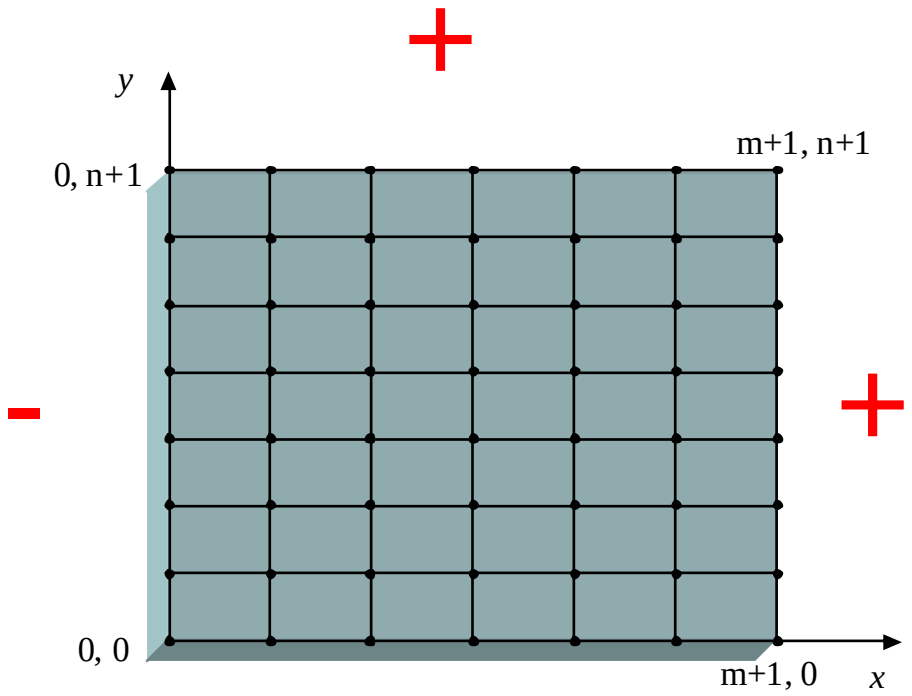
$$A * W_{j+1} = B * W_j + c_j$$

$$W_j = \left(W_{i,j}^{l}\right)_{i=1,\ldots,m-1}^{j=1,\ldots,m-1}{}_{l=1,\ldots,N-1}$$

Incondicionalmente estable

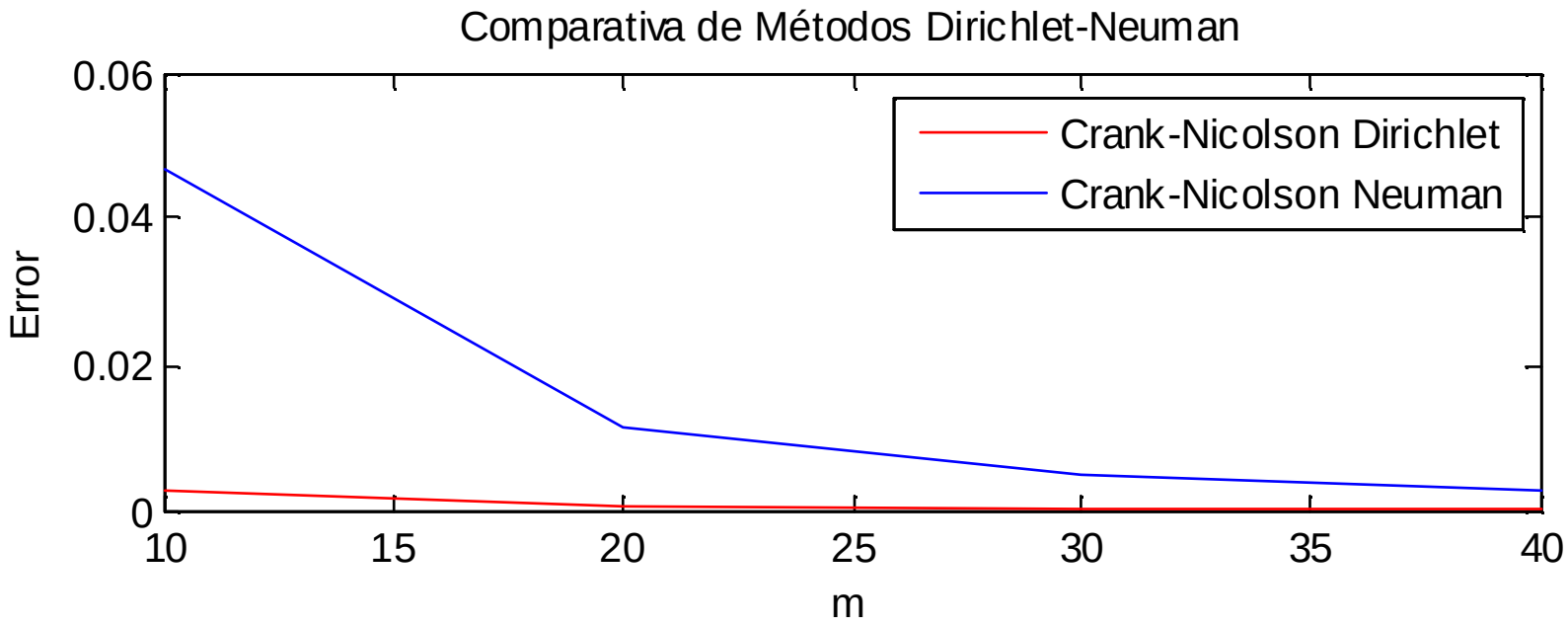
CranK - Nicolson– Vector Cj

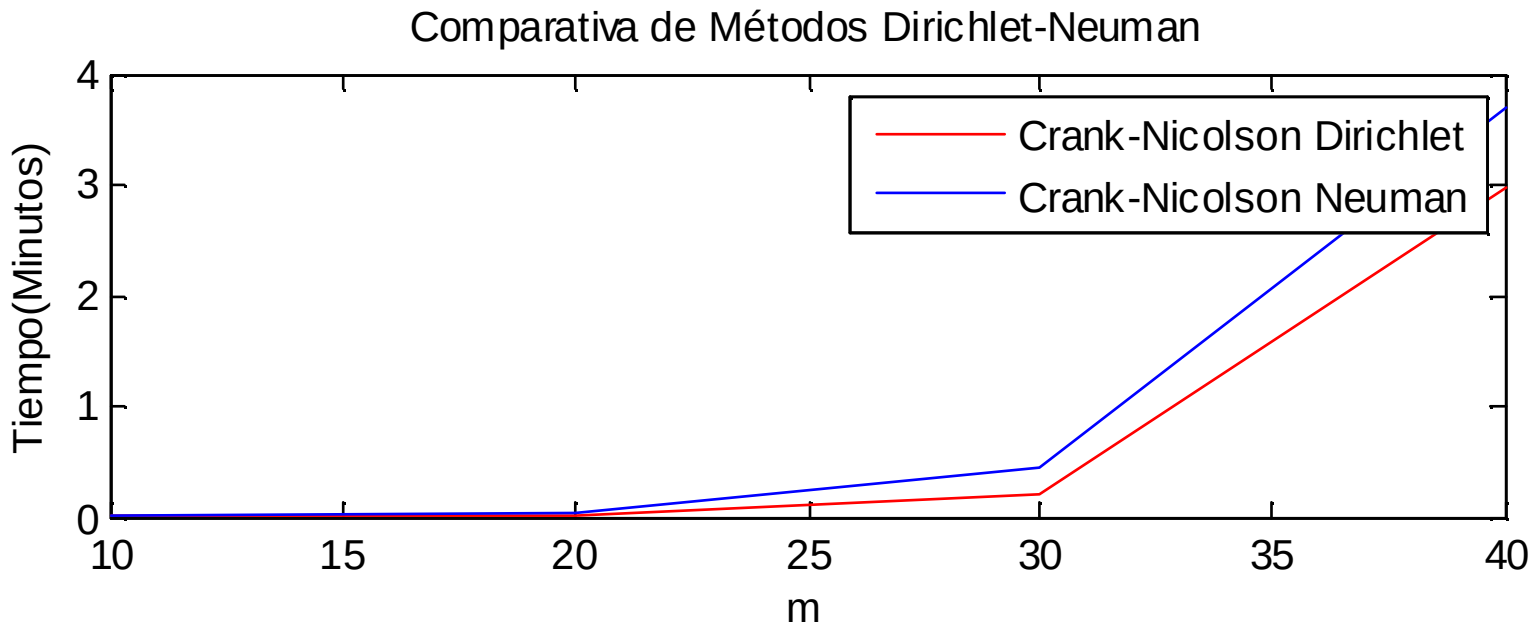
$$
\begin{bmatrix}
k*F^{l}_{1,1} - \lambda(a(x,l)+a(x,l+1)) - \lambda(b(x,l)+b(x,l+1)) \\
k*F^{l}_{2,1} - \lambda(a(x,l)+a(x,l+1)) \\
\cdots\cdots \\
k*F^{l}_{m-2,1} - \lambda(a(x,l)+a(x,l+1)) \\
k*F^{l}_{m-1,1} - \lambda(a(x,l)+a(x,l+1)) + \lambda(d(x,l)+d(x,l+1))
\end{bmatrix}
\begin{bmatrix}
k*F^{l}_{1,2} - \lambda(b(x,l)+b(x,l+1)) \\
k*F^{l}_{2,2} \\
\cdots\cdots \\
k*F^{l}_{m-2,2} \\
k*F^{l}_{m-1,2} + \lambda(d(x,l)+d(x,l+1))
\end{bmatrix}
\begin{bmatrix}
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\
\cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots
\end{bmatrix}
\begin{bmatrix}
k*F^{l}_{1,m-2} - \lambda(b(x,l)+b(x,l+1)) \\
k*F^{l}_{2,m-2} \\
\cdots\cdots \\
k*F^{l}_{m-2,m-2} \\
k*F^{l}_{m-1,m-2} + \lambda(d(x,l)+d(x,l+1))
\end{bmatrix}
\begin{bmatrix}
k*F^{l}_{1,m-1} + \lambda(c(x,l)+c(x,l+1)) - \lambda(b(x,l)+b(x,l+1)) \\
k*F^{l}_{2,m-1} + \lambda(c(x,l)+c(x,l+1)) \\
\cdots\cdots \\
k*F^{l}_{m-2,m-1} + \lambda(c(x,l)+c(x,l+1)) \\
k*F^{l}_{m-1,m-1} + \lambda(c(x,l)+c(x,l+1)) + \lambda(d(x,l)+d(x,l+1))
\end{bmatrix}
$$



Signos en los extremos

Resultados CranK - Nicolson - Neumann





Ecuación del calor Bidimensional

Gracias por vuestra atención

UHU – 4º Ingeniero Industrial - Curso 2008/09

Profesor: Antonio Algaba Durán

Alumno: Luis Heredia Castillo

Asignatura: Ecuaciones Diferenciales y Métodos Numéricos

Anexo 1: algoritmos para la discretización de la ecuación bidimensional del calor.

Condiciones de frontera tipo Dirichlet.

Algoritmos comunes a los tres métodos estudiados:

```
function Dirichlet
%Cálculo de las condiciones de contorno tipo Dirichlet
%clc
%clear all
c=1;L=1;
syms x y t
%Función solución exacta.
u=x^4+y^4+t^4+x^2+y^2+3*t^5+4;

fprintf('u=');
pretty(u);
ux=diff(u,'x');
uy=diff(u,'y');
ut=diff(u,'t');
uxx=diff(ux,'x');
uyy=diff(uy,'y');
%
%Calculo la función F(x,y,t) desde la ECD la llamo GG
GG=ut-c*(uxx+uyy);
GG=vectorize(GG);
GG=inline(GG,'x','y','t')
%
%Calculo f(x,y) y la llamo ff
syms x y t
t=0;
ff=eval(vectorize(u));
ff=inline(ff)
%
%Calculo a(x,t) la llamo aa(x,t)=u(x,0,t)
syms x y t
y=0;
aa=eval(vectorize(u));
aa=inline(aa)
%
%Calculo b(y,t) la llamo bb(y,t)=u(0,y,t)
syms x y t
x=0;
bb=eval(vectorize(u));
bb=inline(bb)
%
%Calculo c(x,t) la llamo cc(x,t)=u(x,L,t)
syms x y t
y=L;
cc=eval(vectorize(u));
cc=inline(cc)
%
%Calculo d(y,t) la llamo dd(y,t)=u(L,y,t)
syms x y t
x=L;
dd=eval(vectorize(u));
```

```
dd=inline(dd)
```

```
function y=aa(x,t)
y=x.^4+t.^4+x.^2+3.*t.^5+4;
```

```
function y=bb(y,t)
y=y.^4+t.^4+y.^2+3.*t.^5+4;
```

```
function y=cc(x,t)
y=x.^4+6+t.^4+x.^2+3.*t.^5+x.*t;
```

```
function y=dd(y,t)
y=6+y.^4+t.^4+y.^2+3.*t.^5+y.*t;
```

```
function y=ff(x,y)
y=x.^4+y.^4+x.^2+y.^2+4;
```

```
function y=GG(x,y,t)
y=4.*t.^3+15.*t.^4+x.*y-12.*x.^2-4-12.*y.^2;
```

```
function u=exacta(x,y,t)
u=x.^4+y.^4+t.^4+x.^2+y.^2+3.*t.^5+x.*y.*t+4;
```

```
function A=tridiagonal(a,b,c,m)
A=zeros(m,m);%primer elemento es la fila, y el segundo la columna.
for i=2:m-1
    A(i,i-1)=b;
    A(i,i)=a;
    A(i,i+1)=c;
end
A(1,1)=a;
A(1,2)=c;
A(m,m-1)=b;
A(m,m)=a;
```

```
function AA=tridiagonal2D(a,b,c,m)
A=tridiagonal(a,b,c,m);
B=tridiagonal(b,0,0,m);
C=tridiagonal(c,0,0,m);
AA=zeros(m*m,m*m);
for i=2:m-1
    AA(((i-1)*m)+1:i*m,((i-2)*m)+1:(i-1)*m)=B;
    AA(((i-1)*m)+1:i*m,((i-1)*m)+1:(i+0)*m)=A;
    AA(((i-1)*m)+1:i*m,((i-0)*m)+1:(i+1)*m)=C;
end
AA(1:m,1:m)=A;
AA(1:m,m+1:2*m)=C;
AA(((m-1)*m)+1:m*m,((m-2)*m)+1:(m-1)*m)=B;
AA(((m-1)*m)+1:m*m,((m-1)*m)+1:m*m)=A;
```

Condiciones de frontera tipo Dirichlet . Método Explícito:

```
function [WN Err]=calorexplicito2D(m,N,L,c,T)
%definir k,h,la
```

```
%m=10;N=10;L=1;c=1;T=1;
k=T/N;
h=L/m;
la=c^2*k/h^2;
%definir A
A=tridiagonal2D(1-4*la,la,la,m-1);
%max(eig(inv(A)))
%calculo de W0
W0=zeros(m-1,m-1);%lo creamos inicialmente como matriz
for i=1:m-1
    for ii=1:m-1
        W0(i,ii)=ff(i*h,ii*h);
    end
end
W=zeros((m-1)*(m-1),1);%lo convertimos en un vector de pie
for i=1:m-1
    W(1+((i-1)*(m-1)):i*(m-1),1)=W0(:,i);
end

%Calculo W1,W2,....,WN
for j=0:N-1
    %construcción de Bj
    b=zeros(m-1,m-1);%lo creamos inicialmente como matriz
    %Fuerza externa
    for i=1:m-1
        for ii=1:m-1
            b(i,ii)=k*GG(i*h,ii*h,j*k);
        end
    end
    %En los bordes de la placa condiciones de Dirichlet
    for i=1:m-1
        b(i,1)=b(i,1)+la*aa(i*h,j*k);
        b(i,m-1)=b(i,m-1)+la*cc(i*h,j*k);
        b(1,i)=b(1,i)+la*bb(i*h,j*k);
        b(m-1,i)=b(m-1,i)+la*dd(i*h,j*k);
    end
    B=zeros((m-1)*(m-1),1);%lo convertimos en un vector de pie
    for i=1:m-1
        B(1+((i-1)*(m-1)):i*(m-1),1)=b(:,i);
    end
    %Cálculo de W1,..,WN
    W=(A*W)+B; %con Jacobi: W=Jacobi(A,W+c,...)
end
WN=zeros(m+1,m+1);%Para dar el resultado como una matriz
for i=2:m
    WN(i,1)=aa((i-1)*h,T);
    WN(i,m+1)=cc((i-1)*h,T);
    WN(1,i)=bb((i-1)*h,T);
    WN(m+1,i)=dd((i-1)*h,T);
    WN(2:m,i)=[W(1+((i-2)*(m-1)):(i-1)*(m-1),1)];
end
%los valores de las esquinas
WN(1,1)=aa(0,T);
WN(1,m+1)=aa(L,T);
WN(m+1,1)=cc(0,T);
WN(m+1,m+1)=cc(L,T);
Wexac=exacta(L/2,L/2,T);
Err=norm(Wexac-WN(1+(m/2),1+(m/2)));
```

```
%Dibujar el error en función de m en el centro de la placa con
condiciones
%de frontera tipo Dirichelt
clc
clear all
format long
c=1;
L=1;
T=1;
ERR=zeros(4,4);
for i=10:10:40
    m=i;
    N=4*c^2*T*m^2/L^2;%para que cumpla la condición de estabilidad
la<=1/4
    t = cputime;
    [WN Err]=calorexplicito2D(m,N,L,c,T);
    e=cputime-t;
    ERR(i/10,:)=[m Err WN(1+(m/2),1+(m/2)) e]
end

%Elog=log(ERR(:,2));
%Min=min(Elog);
%Elogdesp=Elog-Min*ones(size(Elog),1);
subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
title('Método Explícito N=4*c^2*T*m^2/L^2');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Explícito N=4*c^2*T*m^2/L^2');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

```
%Resultado método Explícito con frontera tipo Dirichelt
%N=4*c^2*T*m^2/L^2
ERR =[10 0.002480317444713   4.652610317444713   0.140625000000
      20 0.000620860723848   4.650750860723848   0.187500000000
      30 0.000276007140810   4.650406007140810   1.734375000000
      40 0.000155267813037   4.650285267813037   8.265625000000
      50 0.000099375510253   4.650229375510253   29.00000000000
      60 0.000069012325659   4.650199012325659   82.07812500000
      70 0.000050703622783   4.650180703622783   428.9218750000
      80 0.000038820304554   4.650168820304554   1008.312500000
      90 0.000030673019068   4.650160673019068   2154.906250000];

subplot(2,1,1);
plot(ERR(:,1),ERR(:,2));
title('Método Explícito N=4*c^2*T*m^2/L^2');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Explícito N=4*c^2*T*m^2/L^2');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

Condiciones de frontera tipo Dirichlet . Método Implícito:

```
function [WN Err]=calorimplicito2D(m,N,L,c,T)
%definir k,h,la
%m=10;N=10;L=1;c=1;T=1;
k=T/N;
h=L/m;
la=c^2*k/h^2;
%definir A
A=tridiagonal2D(1+4*la,-la,-la,m-1);
%Radio Espectral de la matriz
%max(eig(inv(A)))
%calculo de W0
W0=zeros(m-1,m-1);%lo creamos inicialmente como matriz
for i=1:m-1
    for ii=1:m-1
        W0(i,ii)=ff(0+i*h,0+ii*h);
    end
end
W=zeros((m-1)*(m-1),1);%lo convertimos en un vector de pie
for i=1:m-1
    W(1+((i-1)*(m-1)):i*(m-1),1)=W0(:,i);
end

%Calculo W1,W2,....,WN
for j=0:N-1
    %construcción de Cj
    c=zeros(m-1,m-1);%lo creamos inicialmente como matriz
    %Fuerza externa
    for i=1:m-1
        for ii=1:m-1
            c(i,ii)=k*GG(i*h,ii*h,(j+1)*k);
        end
    end
    %En los bordes de la placa se aplican condiciones de Dirichlet
    for i=1:m-1
        c(i,1)=c(i,1)+la*aa(i*h,(j+1)*k);
        c(i,m-1)=c(i,m-1)+la*cc(i*h,(j+1)*k);
        c(1,i)=c(1,i)+la*bb(i*h,(j+1)*k);
        c(m-1,i)=c(m-1,i)+la*dd(i*h,(j+1)*k);
    end
    C=zeros((m-1)*(m-1),1);%lo convertimos en un vector de pie
    for i=1:m-1
        C(1+((i-1)*(m-1)):i*(m-1),1)=c(:,i);
    end
    %Cálculo de W1,..,WN
    W=A\(W+C); %con Jacobi: W=Jacobi(A,W+c,...)
end
WN=zeros(m+1,m+1);%Para dar el resultado como una matriz
for i=2:m
    WN(i,1)=aa((i-1)*h,T);
    WN(i,m+1)=cc((i-1)*h,T);
    WN(1,i)=bb((i-1)*h,T);
    WN(m+1,i)=dd((i-1)*h,T);
    WN(2:m,i)=[W(1+((i-2)*(m-1)):(i-1)*(m-1),1)];
end
%los valores de las esquinas
WN(1,1)=aa(0,T);
WN(1,m+1)=aa(L,T);
WN(m+1,1)=cc(0,T);
```

```
WN(m+1,m+1)=cc(L,T);

Wexac=exacta(L/2,L/2,T);
Err=abs(Wexac-WN(1+(m/2),1+(m/2)));
```

```
%Dibujar el error en función de m en el centro de la placa con
condiciones
%de frontera tipo Dirichelt – Algoritmo implícito
clc
clear all
format long
c=1;
L=1;
T=1;
ERR=zeros(4,4);
for i=10:10:40
    m=i;
    N=4*c^2*T*m^2/L^2;
    %N=m^2;
    t = cputime;
    [WN Err]=calorimplicito2D(m,N,L,c,T);
    e=cputime-t;
    ERR(i/10,:)=[m Err WN(1+(m/2),1+(m/2)) e]
end
subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
%title('Error');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
%title('Tiempo (Minutos)');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

```
%Resultado método implícito con frontera tipo Dirichelt
%N=m
ERR =[10   0.002402983205486   4.652532983205486   0.078125000000000
      20   0.000621970768977   4.650751970768977   1.000000000000000
      30   0.000281086222335   4.650411086222335   12.20312500000000
      40   0.000160136992987   4.650290136992987   89.25000000000000
      50   0.000103641458773   4.650233641458772   767.8906250000000
      60   0.000072726606555   4.650202726606555   2537.921875000000
      70   0.000053965667774   4.650183965667774   6537.953125000000];

subplot(2,1,1);
plot(ERR(:,1),ERR(:,2));
title('Método Implícito N=m');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Implícito N=m');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

Condiciones de frontera tipo Dirichlet . Método Crank Nicolson:

```
function [WN Err]=calorCrank2D(m,N,L,c,T)
%definir k,h,la
%m=10;N=10;L=1;c=1;T=1;
k=T/N;
h=L/m;
la=c^2*k/h^2;
%definir A
A=tridiagonal2D(1+2*la,-la/2,-la/2,m-1);
%max(eig(inv(A)))
B=tridiagonal2D(1-2*la,la/2,la/2,m-1);
%calculo de W0
W0=zeros(m-1,m-1);%lo creamos inicialmente como matriz
for i=1:m-1
    for ii=1:m-1
        W0(i,ii)=ff(0+i*h,0+ii*h);
    end
end
W=zeros((m-1)*(m-1),1);%lo convertimos en un vector de pie
for i=1:m-1
    W(1+((i-1)*(m-1)):i*(m-1),1)=W0(:,i);
end

%Calculo W1,W2,....,WN
for j=0:N-1
    %construcción de Cj
    c=zeros(m-1,m-1);%lo creamos inicialmente como matriz
    %Fuerza externa
    for i=1:m-1
        for ii=1:m-1
            c(i,ii)=k*GG(i*h,ii*h,(j+1/2)*k);
        end
    end
    %En los bordes de la placa condiciones de Dirichlet
    for i=1:m-1
        c(i,1)=c(i,1)+la/2*(aa(i*h,(j+1)*k)+aa(i*h,(j)*k));
        c(i,m-1)=c(i,m-1)+la/2*(cc(i*h,(j+1)*k)+cc(i*h,(j)*k));
        c(1,i)=c(1,i)+la/2*(bb(i*h,(j+1)*k)+bb(i*h,(j)*k));
        c(m-1,i)=c(m-1,i)+la/2*(dd(i*h,(j+1)*k)+dd(i*h,(j)*k));
    end
    C=zeros((m-1)*(m-1),1);%lo convertimos en un vector de pie
    for i=1:m-1
        C(1+((i-1)*(m-1)):i*(m-1),1)=c(:,i);
    end
    %Cálculo de W1,..,WN
    W=A\(B*W+C); %con Jacobi: W=Jacobi(A,W+c,...)
end
WN=zeros(m+1,m+1);%Para dar el resultado como una matriz
for i=2:m
    WN(i,1)=aa((i-1)*h,T);
    WN(i,m+1)=cc((i-1)*h,T);
    WN(1,i)=bb((i-1)*h,T);
    WN(m+1,i)=dd((i-1)*h,T);
    WN(2:m,i)=[W(1+((i-2)*(m-1)):(i-1)*(m-1),1)];
end
%los valores de las esquinas
WN(1,1)=aa(0,T);
WN(1,m+1)=aa(L,T);
WN(m+1,1)=cc(0,T);
```

```
WN(m+1,m+1)=cc(L,T);
Wexac=exacta(L/2,L/2,T);
Err=abs(Wexac-WN(1+(m/2),1+(m/2)));
```

```
%Dibujar el error en función de m en el centro de la placa con
condiciones
%de frontera tipo Dirichelt – Algoritmo Crank Nicolson
clc
clear all
format long
c=1;
L=1;
T=1;
ERR=zeros(4,4);
for i=10:10:40
    m=i;
    N=m;
    t = cputime;
    [WN Err]=calorCrank2D(m,N,L,c,T);
    e=cputime-t;
    ERR(i/10,:)=[m Err WN(1+(m/2),1+(m/2)) e]
end

%Extrapolación de Richarson
%Ri=Richarson(ERR(:,3),exacta(L/2,L/2,T))

%Dibujar el error en función de m
%Elog=log(ERR(:,2));
%Min=min(Elog);
%Elogdesp=Elog-Min*ones(size(Elog),1);
subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
%title('Error');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
%title('Tiempo (Minutos)');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

```
%Resultado método Crank Nicolson con frontera tipo Dirichelt implícito
%N=m
%ERR = m          error              valor exacto            t(seg)
ERR =[10   0.002466302821425   4.652596302821425   0.093750000000000
      20   0.000620992257281   4.650750992257281   0.484375000000000
      30   0.000276364329434   4.650406364329434   6.718750000000000
      40   0.000155527520534   4.650285527520534  47.796875000000000
      50   0.000099559135272   4.650229559135272  464.09375000000000
      60   0.000069146410941   4.650199146410941  1761.7968750000000]
% Para m=80 "Out of memory"
subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
title('Método Crank-Nicolson');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
```

```
title('Método Crank-Nicolson');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

Anexo 2: algoritmos para la discretización de la ecuación bidimensional del calor.

Condiciones de frontera tipo Neuman.

Algoritmos comunes a los tres métodos estudiados:

```
function Neumann
%Cálculo de las condiciones de contorno tipo Dirichlet
%clc
%clear all
c=1;L=1;
syms x y t
%Función solución exacta.
u=t.*x.^4+t.*y.^4+t.^4+x.^2+y.^2+3.*t.^5+x.*y.*t+4;

fprintf('u=');
pretty(u);
ux=diff(u,'x');
uy=diff(u,'y');
ut=diff(u,'t');
uxx=diff(ux,'x');
uyy=diff(uy,'y');
%
%Calculo la función F(x,y,t) desde la ECD la llamo GG
GG=ut-c*(uxx+uyy);
GG=vectorize(GG);
GG=inline(GG,'x','y','t')
%
%Calculo f(x,y) y la llamo ff
syms x y t
t=0;
ff=eval(vectorize(u));
ff=inline(ff)
%
%Calculo a(x,t) la llamo aa(x,t)=u(x,0,t)
syms x y t
y=0;
aa=eval(vectorize(uy));
aa=inline(aa)
%
%Calculo b(y,t) la llamo bb(y,t)=u(0,y,t)
syms x y t
x=0;
bb=eval(vectorize(ux));
bb=inline(bb)
%
%Calculo c(x,t) la llamo cc(x,t)=u(x,L,t)
syms x y t
y=L;
cc=eval(vectorize(uy));
cc=inline(cc)
%
%Calculo d(y,t) la llamo dd(y,t)=u(L,y,t)
syms x y t
x=L;
dd=eval(vectorize(ux));
```

```
dd=inline(dd)
```

```
function y=aa(x,t)
y=x.*t;
```

```
function y=bb(y,t)
y=y.*t;
```

```
function y=cc(x,t)
y=4.*t+2+x.*t;
```

```
function y=dd(y,t)
y=4.*t+2+y.*t;
```

```
function y=ff(x,y)
y=x.^2+y.^2+4;
```

```
function y=GG(x,y,t)
y=x.^4+y.^4+4.*t.^3+15.*t.^4+x.*y-12.*t.*x.^2-4-12.*t.*y.^2;
```

```
function u=exacta(x,y,t)
u=t.*x.^4+t.*y.^4+t.^4+x.^2+y.^2+3.*t.^5+x.*y.*t+4;
```

```
function A=tridiagonal(a,b,c,m)
A=zeros(m,m);%primer elemento es la fila, y el segundo la columna.
for i=2:m-1
    A(i,i-1)=b;
    A(i,i)=a;
    A(i,i+1)=c;
end
A(1,1)=a;
A(1,2)=c;
A(m,m-1)=b;
A(m,m)=a;
```

```
function AA=tridiagonalNeumann2D(a,b,c,m)
A=tridiagonal(a,b,c,m);
%modificaciones de la matriz A para condiciones de Neumann
A(1,2)=2*c;
A(m,m-1)=2*c;
B=tridiagonal(b,0,0,m);
C=tridiagonal(c,0,0,m);
AA=zeros(m*m,m*m);
for i=2:m-1
    AA(((i-1)*m)+1:i*m,((i-2)*m)+1:(i-1)*m)=B;
    AA(((i-1)*m)+1:i*m,((i-1)*m)+1:(i+0)*m)=A;
    AA(((i-1)*m)+1:i*m,((i-0)*m)+1:(i+1)*m)=C;
end
%rellenar la fila 1 y la fila m
AA(1:m,1:m)=A;
AA(((m-1)*m)+1:m*m,((m-1)*m)+1:m*m)=A;
%modificaciones de la matriz A para condiciones de Neumann
C=tridiagonal(2*c,0,0,m);
AA(1:m,m+1:2*m)=C;
AA(((m-1)*m)+1:m*m,((m-2)*m)+1:(m-1)*m)=C;
```

Condiciones de frontera tipo Neumann . Método Explícito:

```
function [WN Err]=calorexplicito2D(m,N,L,c,T)
%definir k,h,la
%m=10;N=10;L=1;c=1;T=1;
k=T/N;
h=L/m;
la=c^2*k/h^2
%definir A
A=tridiagonalNeumann2D(1-4*la,la,la,m+1);
%Radio Espectral de la matriz
%max(eig(inv(A)))
%calculo de W0
W0=zeros(m+1,m+1);%lo creamos inicialmente como matriz
for i=0:m
    for ii=0:m
        W0(i+1,ii+1)=ff(0+i*h,0+ii*h);
    end
end
W=zeros((m+1)*(m+1),1);%lo convertimos en un vector de pie
for i=0:m
    W(1+(i*(m+1)):(i+1)*(m+1),1)=W0(:,(i+1));
end

%Calculo W1,W2,....,WN
for j=0:N-1
    %construcción de Cj
    b=zeros(m+1,m+1);%lo creamos inicialmente como matriz
    %Fuerza externa
    for i=0:m
        for ii=0:m
            b(i+1,ii+1)=k*GG(i*h,ii*h,j*k);
        end
    end
    %En los bordes de la placa se aplican condiciones de Neumann
    for i=0:m
        b(i+1,1)=b(i+1,1)-2*la*h*aa(i*h,j*k);
        b(1,i+1)=b(1,i+1)-2*la*h*bb(i*h,j*k);
        b(i+1,m+1)=b(i+1,m+1)+2*la*h*cc(i*h,j*k);
        b(m+1,i+1)=b(m+1,i+1)+2*la*h*dd(i*h,j*k);
    end
    C=zeros((m+1)*(m+1),1);%lo convertimos en un vector de pie
    for i=0:m
        B(1+(i*(m+1)):(i+1)*(m+1),1)=b(:,(i+1));
    end
    %Cálculo de W1,..,WN
    W=A*W+B; %con Jacobi: W=Jacobi(A,W+c,...)
end
WN=zeros(m+1,m+1);%Para dar el resultado como una matriz
for i=0:m
    WN(1:m+1,(i+1))=[W(1+(i*(m+1)):(i+1)*(m+1),1)];
end
Wexac=exacta(L/2,L/2,T);
Err=abs(Wexac-WN(1+(m/2),1+(m/2)))
```

```
%Dibujar el error en función de m en el centro de la placa con
condiciones
%de frontera tipo Neuman considerando sólo el punto central de la
placa T=1
clc
```

```
clear all
format long
c=1;
L=1;
T=1;
ERR=zeros(4,4);
for i=10:10:40
    m=i;
    N=4*c^2*T*m^2/L^2-1;%para que cumpla la condición de estabilidad
la<=1/4
    t = cputime;
    [WN Err]=calorexplicito2D(m,N,L,c,T);
    e=cputime-t;
    ERR(i/10,:)=[m Err WN(1+(m/2),1+(m/2)) e]
end

%Ri=Richarson(ERR(:,3),exacta(L/2,L/2,T))

%Dibujar el error en función de m

%Elog=log(ERR(:,2));
%Min=min(Elog);
%Elogdesp=Elog-Min*ones(size(Elog),1);
subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
title('Método Explícito N=4*c^2*T*m^2/L^2');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Explícito N=4*c^2*T*m^2/L^2');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

```
%Resultado método Explícito con frontera tipo Neumann
%N=4*c^2*T*m^2/L^2
ERR =[10   0.002480317444713   4.652610317444713   0.140625000000
      20   0.000620860723848   4.650750860723848   0.187500000000
      30   0.000276007140810   4.650406007140810   1.734375000000
      40   0.000155267813037   4.650285267813037   8.265625000000
      50   0.000099375510253   4.650229375510253   29.00000000000
      60   0.000069012325659   4.650199012325659   82.07812500000
      70   0.000050703622783   4.650180703622783   428.9218750000
      80   0.000038820304554   4.650168820304554   1008.312500000
      90   0.000030673019068   4.650160673019068   2154.906250000];

subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
title('Método Explícito N=4*c^2*T*m^2/L^2');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Explícito N=4*c^2*T*m^2/L^2');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

Condiciones de frontera tipo Neumann . Método Implícito:

```
function [WN Err]=calorimplicito2D(m,N,L,c,T)
%definir k,h,la
%m=10;N=10;L=1;c=1;T=1;
k=T/N;
h=L/m;
la=c^2*k/h^2
%definir A
A=tridiagonalNeumann2D(1+4*la,-la,-la,m+1);
%Radio Espectral de la matriz
%max(eig(inv(A)))
%calculo de W0
W0=zeros(m+1,m+1);%lo creamos inicialmente como matriz
for i=0:m
    for ii=0:m
        W0(i+1,ii+1)=ff(0+i*h,0+ii*h);
    end
end
W=zeros((m+1)*(m+1),1);%lo convertimos en un vector de pie
for i=0:m
    W(1+(i*(m+1)):(i+1)*(m+1),1)=W0(:,(i+1));
end

%Calculo W1,W2,....,WN
for j=0:N-1
    %construcción de Cj
    c=zeros(m+1,m+1);%lo creamos inicialmente como matriz
    %Fuerza externa
    for i=0:m
        for ii=0:m
            c(i+1,ii+1)=k*GG(i*h,ii*h,(j+1)*k);
        end
    end
    %En los bordes de la placa se aplican condiciones de Neumann
    for i=0:m
        c(i+1,1)=c(i+1,1)-2*la*h*aa(i*h,(j+1)*k);
        c(1,i+1)=c(1,i+1)-2*la*h*bb(i*h,(j+1)*k);
        c(i+1,m+1)=c(i+1,m+1)+2*la*h*cc(i*h,(j+1)*k);
        c(m+1,i+1)=c(m+1,i+1)+2*la*h*dd(i*h,(j+1)*k);
    end
    C=zeros((m+1)*(m+1),1);%lo convertimos en un vector de pie
    for i=0:m
        C(1+(i*(m+1)):(i+1)*(m+1),1)=c(:,(i+1));
    end
    %Cálculo de W1,..,WN
    W=A\(W+C); %con Jacobi: W=Jacobi(A,W+c,...)
end
WN=zeros(m+1,m+1);%Para dar el resultado como una matriz
for i=0:m
    WN(1:m+1,(i+1))=[W(1+(i*(m+1)):(i+1)*(m+1),1)];
end
Wexac=exacta(L/2,L/2,T);
Err=abs(Wexac-WN(1+(m/2),1+(m/2)))
```

```
%Dibujar el error en función de m en el centro de la placa con
condiciones
%de frontera tipo Neumann – Algoritmo implícito
clc
clear all
```

```
format long
c=1;
L=1;
T=1;
ERR=zeros(4,4);
for i=10:10:40
    m=i;
    N=m;
    t = cputime;
    [WN Err]=calorimplicito2D(m,N,L,c,T);
    e=cputime-t;
    ERR(i/10,:)=[m Err WN(1+(m/2),1+(m/2)) e]
end

%Ri=Richarson(ERR(:,3),exacta(L/2,L/2,T))

%Dibujar el error en función de m

%Elog=log(ERR(:,2));
%Min=min(Elog);
%Elogdesp=Elog-Min*ones(size(Elog),1);
subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
title('Método Implícito - Neuman  N=m');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Implícito - Neuman  N=m');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

```
% Resultado método Implicito tipo Neuman N=m  considerando sólo el
punto central de la placa T=1
ERR =[10   0.991247100016251   9.866247100016251   0.109375000000000
      20   0.485559720312555   9.360559720312555   2.093750000000000
      30   0.321396741944769   9.196396741944769  27.156250000000000
      40   0.240171050317260   9.115171050317260  318.6250000000000];

subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
title('Método Implícito N=m');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Implícito N=m');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

Condiciones de frontera tipo Neuman . Método Crank Nicolson:

```
function [WN Err]=Crack2D(m,N,L,c,T)
%definir k,h,la
%m=10;N=10;L=1;c=1;T=1;
```

```
k=T/N;
h=L/m;
la=c^2*k/h^2
%definir A
A=tridiagonalNeumann2D(1+2*la,-la/2,-la/2,m+1);
%Radio Espectral de la matriz
%max(eig(inv(A)))
B=tridiagonalNeumann2D(1-2*la,la/2,la/2,m+1);
%calculo de W0
W0=zeros(m+1,m+1);%lo creamos inicialmente como matriz
for i=0:m
    for ii=0:m
        W0(i+1,ii+1)=ff(0+i*h,0+ii*h);
    end
end
W=zeros((m+1)*(m+1),1);%lo convertimos en un vector de pie
for i=0:m
    W(1+(i*(m+1)):(i+1)*(m+1),1)=W0(:,(i+1));
end

%Calculo W1,W2,....,WN
for j=0:N-1
    %construcción de Cj
    c=zeros(m+1,m+1);%lo creamos inicialmente como matriz
    %Fuerza externa
    for i=0:m
        for ii=0:m
            c(i+1,ii+1)=k*GG(i*h,ii*h,(j+(1/2))*k);
        end
    end
    %En los bordes de la placa se aplican condiciones de Neumann
    for i=0:m
        c(i+1,1)=c(i+1,1)-la*h*(aa(i*h,(j+1)*k)+aa(i*h,j*k));
        c(1,i+1)=c(1,i+1)-la*h*(bb(i*h,(j+1)*k)+bb(i*h,j*k));
        c(i+1,m+1)=c(i+1,m+1)+la*h*(cc(i*h,(j+1)*k)+cc(i*h,j*k));
        c(m+1,i+1)=c(m+1,i+1)+la*h*(dd(i*h,(j+1)*k)+dd(i*h,j*k));
    end
    C=zeros((m+1)*(m+1),1);%lo convertimos en un vector de pie
    for i=0:m
        C(1+(i*(m+1)):(i+1)*(m+1),1)=c(:,(i+1));
    end
    %Cálculo de W1,..,WN
    W=A\(B*W+C); %con Jacobi: W=Jacobi(A,W+c,...)
end
WN=zeros(m+1,m+1);%Para dar el resultado como una matriz
for i=0:m
    WN(1:m+1,(i+1))=[W(1+(i*(m+1)):(i+1)*(m+1),1)];
end
Wexac=exacta(L/2,L/2,T);
Err=abs(Wexac-WN(1+(m/2),1+(m/2)))
```

```
%Dibujar el error en función de m en el centro de la placa con
condiciones
%de frontera tipo Neumann – Algoritmo Crank-Nicolson
clc
clear all
format long
c=1;
L=1;
T=1;
```

```
ERR=zeros(4,4);
for i=10:10:40
    m=i;
    N=m;
    t = cputime;
    [WN Err]=Crack2D(m,N,L,c,T);
    e=cputime-t;
    ERR(i/10,:)=[m Err WN(1+(m/2),1+(m/2)) e]
end

%Ri=Richarson(ERR(:,3),exacta(L/2,L/2,T))

%Dibujar el error en función de m

%Elog=log(ERR(:,2));
%Min=min(Elog);
%Elogdesp=Elog-Min*ones(size(Elog),1);
subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
title('Método Implícito - Neuman  N=m');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Implícito - Neuman  N=m');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

```
%Resultado método Crank Nicolson con frontera tipo Neuman implícito
%N=m considerando sólo el punto central de la placa T=1
%ERR = m         error           valor exacto           t(seg)
ERR =[10   0.046659318490631   8.828340681509369   0.093750000000000
      20   0.011684420487903   8.863315579512097   2.109375000000000
      30   0.005194693230068   8.869805306769932  26.375000000000000
      40   0.002922333472373   8.872077666527627  222.1718750000000];
subplot(2,1,1);
%plot(ERR(:,1),Elogdesp);
plot(ERR(:,1),ERR(:,2));
title('Método Crank-Nicolson con Frontera tipo Neumann');
xlabel('m');
ylabel('Error')
subplot(2,1,2);
plot(ERR(:,1),ERR(:,4)/60);
title('Método Crank-Nicolson con Frontera tipo Neumann');
xlabel('m');
ylabel('Tiempo(Minutos)')
```

Comparativa de métodos

```
%Comparativa de métodos T=1

%Explícito N=4*c^2*T*m^2/L^2 (landa <=1/4) Dirichlet
%ERR = m           error            valor exacto           t(seg)
ERREX =[10   0.002728599244071   8.872271400755929   0.1406250000000
        20   0.000684935104768   8.874315064895232   1.5312500000000
        30   0.000304649141921   8.874695350858080  16.6250000000000
        40   0.000171411303432   8.874828588696568  83.1562500000000];

%Implícito N=m Dirichlet
%ERR = m           error            valor exacto           t(seg)
ERRIM =[10   0.210527916576604   9.085527916576604   0.0781250000000
        20   0.109648267558974   8.984648267558974   0.9531250000000
        30   0.074041942769215   8.949041942769215   12.218750000000
        40   0.055880048754092   8.930880048754093   178.21870000000];

%Implícito N=m^2 Dirichlet
%ERR = m           error            valor exacto           t(seg)
ERRIM2 =[10   0.002466302821425   8.900299036673710   0.140625000000
         20   0.000620992257281   8.881401240611737   9.468750000000
         30   0.000276364329434   8.877851370741265   418.3437500000
         40   0.000155527520534   8.876605154490269   4387.625000000];

%Crank-Nicolson Dirichlet
%ERR = m           error            valor exacto           t(seg)
ERRCR=[10   0.002662809420785   8.872337190579215   0.10937500000000
       20   0.000671809989671   8.874328190010330   1.03125000000000
       30   0.000299169436829   8.874700830563171  12.34375000000000
       40   0.000168396595161   8.874831603404839  178.2187500000000];


%Error Explicito-Neuman
%ERR = m           error            valor exacto           t(seg)
ERREXN =[10   0.040424610310490   8.834575389689510   0.265625000000
         20   0.010122894309092   8.864877105690908   3.437500000000
         30   0.004500441472013   8.870499558527987  31.437500000000
         40   0.002531769466778   8.872468230533222  145.20312500000];

%Error Explicito-Neuman N=m
%ERR = m           error            valor exacto           t(seg)
ERRIMN =[10   0.991247100016251   9.866247100016251   0.109375000000
         20   0.485559720312555   9.360559720312555   2.093750000000
         30   0.321396741944769   9.196396741944769  27.156250000000
         40   0.240171050317260   9.115171050317260  318.62500000000];

%Error Crank Nicolson-Neuman N=m
%ERR = m           error            valor exacto           t(seg)
ERRCRN =[10   0.046659318490631   8.828340681509369   0.093750000000
         20   0.011684420487903   8.863315579512097   2.109375000000
         30   0.005194693230068   8.869805306769932  26.375000000000
         40   0.002922333472373   8.872077666527627  222.17187500000];

figure(1);
subplot(2,1,1);
h=plot(ERREX(:,1),ERREX(:,2),ERRIM(:,1),ERRIM(:,2));
set(h,{'Color'},{'r';'b'});
```

```
title('Comparativa de Métodos Explícito-Implícito');
xlabel('m');
ylabel('Error');
legend(h,'Explícito N=4*c^2*T*m^2/L^2 (landa <=1/4)','Implícito N=m')
subplot(2,1,2);
hh=plot(ERREX(:,1),ERREX(:,4)/60,ERRIM(:,1),ERRIM(:,4)/60);
set(hh,{'Color'},{'r';'b'});
title('Comparativa de Métodos Explícito-Implícito');
xlabel('m');
ylabel('Tiempo(Minutos)');
legend(hh,'Explícito N=4*c^2*T*m^2/L^2 (landa <=1/4)','Implícito N=m')

figure(2);
subplot(2,1,1);
h=plot(ERREX(:,1),ERREX(:,2),ERRIM2(:,1),ERRIM2(:,2));
set(h,{'Color'},{'r';'b'});
title('Comparativa de Métodos Explícito-Implícito');
xlabel('m');
ylabel('Error');
legend(h,'Explícito N=4*c^2*T*m^2/L^2 (landa <=1/4)','Implícito
N=m^2')
subplot(2,1,2);
hh=plot(ERREX(:,1),ERREX(:,4)/60,ERRIM2(:,1),ERRIM2(:,4)/60);
set(hh,{'Color'},{'r';'b'});
title('Comparativa de Métodos Explícito-Implícito');
xlabel('m');
ylabel('Tiempo(Minutos)');
legend(hh,'Explícito N=4*c^2*T*m^2/L^2 (landa <=1/4)','Implícito
N=m^2')

figure(3);
subplot(2,1,1);
h=plot(ERRIM(:,1),ERRIM(:,2),ERRCR(:,1),ERRCR(:,2));
set(h,{'Color'},{'r';'b'});
title('Comparativa de Métodos Implícito N=m-Crank-Nicolson');
xlabel('m');
ylabel('Error');
legend(h,'Implícito N=m','Crank-Nicolson')
subplot(2,1,2);
hh=plot(ERRIM(:,1),ERRIM(:,4)/60,ERRCR(:,1),ERRCR(:,4)/60);
set(hh,{'Color'},{'r';'b'});
title('Comparativa de Métodos Implícito N=m-Crank-Nicolson');
xlabel('m');
ylabel('Tiempo(Minutos)');
legend(hh,'Implícito N=m','Crank-Nicolson')


figure(4);
subplot(2,1,1);
h=plot(ERREX(:,1),ERREX(:,2),ERREXN(:,1),ERREXN(:,2));
set(h,{'Color'},{'r';'b'});
title('Comparativa de Métodos Dirichlet-Neuman');
xlabel('m');
ylabel('Error');
legend(h,'Explícito Dirichlet','Explícito Neuman')
subplot(2,1,2);
hh=plot(ERREX(:,1),ERREX(:,4)/60,ERREXN(:,1),ERREXN(:,4)/60);
set(hh,{'Color'},{'r';'b'});
title('Comparativa de Métodos Dirichlet-Neuman');
xlabel('m');
ylabel('Tiempo(Minutos)');
```

```
legend(hh,'Explícito Dirichlet','Explícito Neuman')

figure(5);
subplot(2,1,1);
h=plot(ERRIM(:,1),ERRIM(:,2),ERRIMN(:,1),ERRIMN(:,2));
set(h,{'Color'},{'r';'b'});
title('Comparativa de Métodos Dirichlet-Neuman');
xlabel('m');
ylabel('Error');
legend(h,'Implícito Dirichlet','Implícito Neuman')
subplot(2,1,2);
hh=plot(ERRIM(:,1),ERRIM(:,4)/60,ERRIMN(:,1),ERRIMN(:,4)/60);
set(hh,{'Color'},{'r';'b'});
title('Comparativa de Métodos Dirichlet-Neuman');
xlabel('m');
ylabel('Tiempo(Minutos)');
legend(hh,'Implícito Dirichlet','Implícito Neuman')

figure(6);
subplot(2,1,1);
h=plot(ERRCR(:,1),ERRCR(:,2),ERRCRN(:,1),ERRCRN(:,2));
set(h,{'Color'},{'r';'b'});
title('Comparativa de Métodos Dirichlet-Neuman');
xlabel('m');
ylabel('Error');
legend(h,'Crank-Nicolson Dirichlet','Crank-Nicolson Neuman')
subplot(2,1,2);
hh=plot(ERRCR(:,1),ERRCR(:,4)/60,ERRCRN(:,1),ERRCRN(:,4)/60);
set(hh,{'Color'},{'r';'b'});
title('Comparativa de Métodos Dirichlet-Neuman');
xlabel('m');
ylabel('Tiempo(Minutos)');
legend(hh,'Crank-Nicolson Dirichlet','Crank-Nicolson Neuman')
```