Análisis Numérico II:

Memoria teórica y aplicaciones personales

Autor: Alfonso Sancho Miró

Curso 2004-2005

Índice

Ecuación elíptica. Ecuación de Laplace. 5

Ecuación parabólica. Ecuación del calor o de difusión. 30

- Método progresivo. 31
- Método regresivo. 33
- Método de Crank-Nicolson. 35
- Variación de la ecuación principal: no se conoce la temperatura en el contorno. 64
- Variación de la ecuación principal: ecuación del calor para trabajar con fluidos. 83

Ecuación hiperbólica. Ecuación de ondas. 98

- Variación de la ecuación principal: ecuación de ondas con fricción. 108

Ecuación del calor: calentamiento de una plancha. 123

Aplicaciones personales. 143

- Calentamiento de una plancha en forma de "L" 143
- Calentamiento de una plancha no cuadrada 150
- Calentamiento de un cubo 158
- Problema de control 169

Apéndice: formas de discretización 177

1. Ecuación elíptica: ecuación de Laplace

1.1 Planteamiento del problema. Ecuación.

Esta ecuación también se conoce como ecuación de Laplace, de Poisson o del potencial. El enunciado de nuestro problema es:

Ecuación E1.1

Encontrar una función $u : R_0 \subset R^2 \rightarrow R$

$$R_0 = \{(x,y) / a \leq x \leq b, c \leq y \leq d\}$$

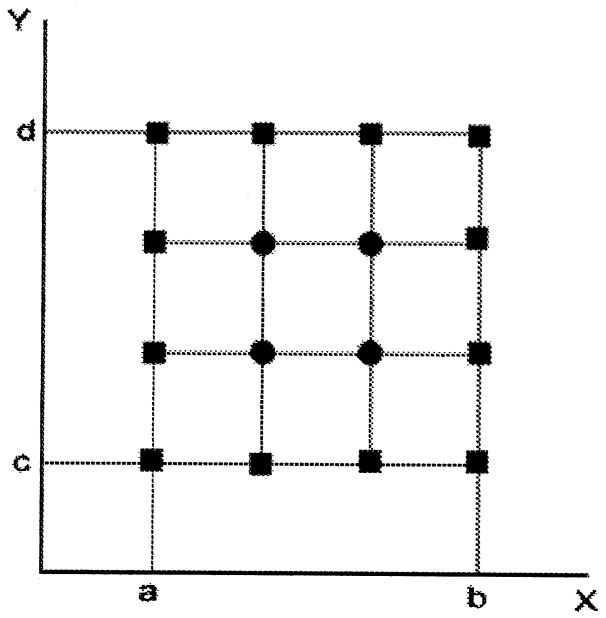
tal que

$$\begin{cases} \Delta u(x,y) = \nabla^2 u(x,y) = f(x,y) \\ u(x,y) = g(x,y) \end{cases}$$

donde

$$\Delta u(x,y) = \frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y)$$

El problema puede representarse gráficamente de la siguiente manera:



Los puntos redondos del gráfico representan los puntos que no conocemos, es decir, los que queremos hallar. Los puntos cuadrados son los puntos conocidos. Podríamos estar hablando, por ejemplo, de una plancha de la cual conocemos la temperatura en su contorno y queremos hallar la temperatura en los puntos de su interior.

1.2 Discretización del problema.

En este problema, conocemos los valores en el contorno, es decir, lo que conocemos es:

$$\left.\begin{aligned} u(x_0,y_j) &= g(x_0,y_j) \\ u(x_n,y_j) &= g(x_n,y_j) \end{aligned}\right\}_{j=0,\ldots,m}$$
$$\left.\begin{aligned} u(x_i,y_0) &= g(x_i,y_0) \\ u(x_i,y_m) &= g(x_i,y_m) \end{aligned}\right\}_{i=0,\ldots,n}$$

De la ecuación principal (ecuación E1.1) deducimos que

$$\frac{\partial^2 u}{\partial x^2}(x,y)+\frac{\partial^2 u}{\partial y^2}(x,y)=f(x,y)$$

Usando la discretización para la segunda derivada (ver apéndice), la ecuación anterior quedaría así;

$$\frac{u(x+h,y)-2u(x,y)+u(x-h,y)}{h^2}+O(h^2)+\frac{u(x,y+k)-2u(x,y)+u(x,y-k)}{k^2}+O(k^2)$$

Con esta ecuación, llamando a $x=x_i$, $y=y_j$, $f(x_i,y_j)=f_{i,j}$, ($u(x_i,y_j) \approx U_{i,j}$ y despreciando $O(h^2)$ y $O(k^2)$ tenemos lo siguiente:

$$\frac{U_{i+1,j}-2U_{i,j}+U_{i-1,j}}{h^2}+\frac{U_{i,j+1}-2U_{i,j}+U_{i,ji-1}}{k^2}=f_{i,j}$$

Y, agrupando los términos, nos queda finalmente:

Ecuación E1.2

$$\left(\frac{-2}{h^2}-\frac{2}{k^2}\right)\cdot U_{i,j}+\frac{1}{h^2}U_{i+1,j}+\frac{1}{h^2}U_{i-1,j}+\frac{1}{k^2}U_{i,j+1}+\frac{1}{k^2}U_{i,j-1}=f_{i,j}$$

donde i=1,...,n y j=1,...,n.

Puede observarse que lo que tenemos es un sistema lineal de (n-1)(m-1) ecuaciones y otras tantas incógnitas. Podemos representarlo en forma matricial Ax=b, con lo que nuestro objetivo será calcular las matrices A y b, para lo que usaremos un algoritmo,

Para poder calcular la matriz A, primero tenemos que establecer un orden en las ecuaciones. Hay diferentes tipos de ordenaciones, pero nosotros veremos primero la ordenación izquierda-derecha y arriba-abajo; y luego pasaremos a ver la ordenación arriba-abajo e izquierda-derecha. Esta segunda ordenación es la óptima para este problema.

1.3 Algoritmo A1.1: Cálculo de matrices A y b para la ecuación de Laplace, con ordenación izquierda-derecha y arriba-abajo.

La ordenación que vamos a usar puede expresarse así:

A cada $U_{i,j}$ le corresponde el puesto $X_{(j-1)(n-1)+i}$

Datos que recibe el algoritmo:

- f: función f(x,y)
- g: function g(x,y)
- a: valor de x_0
- b: valor de x_n
- c: valor de y_0
- d: valor de y_n
- n: número de divisiones que se harán en el eje x, entre a y b
- m: número de divisiones que se harán en el eje y, entre c y d

Datos que devuelve el algoritmo:

- A: matriz cuadrada de orden (n-1)(m-1), que representa a la matriz A de coeficientes del sistema
- b: vector de (n-1)(m-1) posiciones, que representa al vector b de términos independientes del sistema

```
function [A,b] = calculoAyb(f,g,a,b,c,d,n,m)
A=zeros(((m-1)*(n-1)),((m-1)*(n-1)));
Eq=0;
h=(b-a)/n;
k=(d-c)/m;
for i=1:n-1
    for j=1:m-1
        Eq=Eq+1;
        x=(a+i*h);
        y=(c+j*k);
        b(Eq)=feval(f,x,y);
        if i==1
            if j<m-1
                b(Eq)=b(Eq)-((1/(h^2))*feval(g,a,y));
            end
        else
            A(Eq,(j-1)*(n-1)+i-1)=1/h^2;
        end
        if j==1
            if i<n-1
```

```
                b(Eq)=b(Eq)-((1/(k^2))*feval(g,x,c));
            end
        else
            A(Eq,(j-1)*(n-1)+i)=1/k^2;
        end
        A(Eq,(j-1)*(n-1)+i)=-(2/(h^2))-(2/(k^2));
    end
end
```

Con el algoritmo anterior ya queda definido el sistema de ecuaciones que se deduce de la ecuación E1.2. Para resolver dicho sistema, pondríamos x=A\b en la línea de comandos de Matlab. Una vez resuelto queda reconstruir la matriz U, para lo que usaremos el siguiente sencillo algoritmo, sabiendo que a un X_h le corresponde el U_{ij} tal que h/(n-1)=j-1+i (j-1:cociente, i:resto).

Datos que recibe el algoritmo:

- X: vector solución del sistema Ax=b
- n: número de divisiones que se hicieron en el eje X

Datos que devuelve el algoritmo:

- U: matriz donde el valor de U(i,j) representa el valor aproximado de $u(x_i,y_j)$

```
function U = calculoU(X,n,m)
lim=length(X);
for h=0:lim-1
    j=fix(h/(n-1))+1;
    i=(h+1)-((j-1)*(n-1));
    U(i,j)=X(h+1);
end
```

1.4 Algoritmo A1.2: Cálculo de matrices A y b para la ecuación de Laplace, con ordenación arriba-abajo e izquierda-derecha.

La ordenación que vamos a usar puede expresarse así:

A cada $U_{i,j}$ le corresponde el puesto $X_{(m-1-j)(n-1)+i}$

Datos que recibe el algoritmo:

- f: función f(x,y)
- g: función g(x,y)
- a: valor de x_0

- b: valor de x_n
- c: valor de y_0
- d: valor de y_n
- n: número de divisiones que se harán en el eje x, entre a y b
- m: número de divisiones que se harán en el eje y, entre c y d

Datos que devuelve el algoritmo:

- A: matriz cuadrada de orden (n-1)(m-1), que representa a la matriz A de coeficientes del sistema
- b: vector de (n-1)(m-1) posiciones, que representa al vector b de términos independientes del sistema

```
function [A1,b1] = calculoAyb(f,g,a,b,c,d,n,m)
A1=zeros((m-1)*(n-1))
Eq=0;
h=(b-a)/n;
k=(d-c)/m;
for i=1:n-1
    for j=1:m-1
        Eq=Eq+1;
        x=(a+i*h);
        y=(c+j*k);
        b1(Eq)=feval(f,x,y);
        if i==1
            b1(Eq)=b1(Eq)-((1/(h^2))*feval(g,a,y));
        else
            A1(Eq,(j-1)*(n-1)+i-1)=1/h^2;
        end

        if j==1
            b1(Eq)=b1(Eq)-((1/(k^2))*feval(g,x,c));
        else
            A1(Eq,(j-2)*(n-1)+i)=1/k^2;
        end

        A1(Eq,(j-1)*(n-1)+i)=-(2/(h^2))-(2/(k^2));
    end
end
```

Una vez resuelto queda reconstruir la matriz U, para lo que usaremos el siguiente sencillo algoritmo, sabiendo que a un X_k le corresponde el $U_{m-1-c,r+1}$ tal que k/(n-1)=c+r (c:cociente, r:resto).

Datos que recibe el algoritmo:

- X: vector solución del sistema Ax=b
- n: número de divisiones que se hicieron en el eje X

Datos que devuelve el algoritmo:

- U: matriz donde el valor de U(i,j) representa el valor aproximado de $u(x_i,y_j)$

```
function U = calculoU(X,m,n)
lim=(n-1)*(m-1);
for k=0:lim-1
    c=fix(k/(n-1));
    r=k-(c*(n-1));
    U(m-1-c,r+1)=X(k+1);
end
U=U';
```

1.5 Ejemplos

• *Ejemplo 1.1*

Tenemos el siguiente planteamiento:

$$\frac{\partial^2 u}{\partial x^2}(x,y)+\frac{\partial^2 u}{\partial y^2}(x,y)=0$$

$$0 < x < 05$$
$$0 < y < 0,5$$

Y las siguientes condiciones de contorno:

$$u(0,y)=0$$
$$u(0'5,y)=200y$$
$$u(x,0)=0$$
$$u(x,0'5)=200x$$

Siendo la siguiente la solución de este ejemplo:

$$u(x,y)=400xy$$

Ahora llamaremos al algoritmo A1.1 para intentar aproximar la solución de este ejemplo. Los datos que pasaremos son:

a=0, b=0'5, c=0, d=0'5, n=4, m=5

Los archivos f.m y g.m son, respectivamente:

f.m

```
function z=f(x,y)
z=0;
```

g.m

```
function z=g(x,y)
if x==0
    if y<0.5
        z=0;
    end
else if x==0.5
        if y>0
            z=200*y;
        end
    end
end
if y==0
    if x<0.5
        z=0;
    end
else if y==0.5
        if x>0
            z=200*x;
        end
    end
end
```

Primero calculamos el valor de las matrices A y b.

>> [A,b] = calculoAyb('f','g',0,0.5,0,0.5,4,5)

```
A =

  Columns 1 through 7

 -328.0000         0         0         0         0         0         0
  100.0000         0         0 -328.0000         0         0         0
         0         0         0  100.0000         0         0 -328.0000
         0         0         0         0         0         0  100.0000
   64.0000 -328.0000         0         0         0         0         0
         0  100.0000         0   64.0000 -328.0000         0         0
         0         0         0         0  100.0000         0   64.0000
```

```
         0         0         0         0         0         0         0
         0   64.0000 -328.0000         0         0         0         0
         0         0  100.0000         0   64.0000 -328.0000         0
         0         0         0         0         0  100.0000         0
         0         0         0         0         0         0         0


  Columns 8 through 12

         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0 -328.0000         0         0
         0         0         0         0         0
         0         0         0         0         0
 -328.0000         0         0         0         0
  100.0000         0   64.0000 -328.0000         0
         0         0         0         0         0
         0         0         0         0         0
   64.0000 -328.0000         0         0         0
         0  100.0000         0   64.0000 -328.0000


b =

  Columns 1 through 11

     0     0     0     0     0     0     0     0     0     0     0

  Column 12

     0
```

Luego hallamos X:

>> X=A\b'

```
X =
     0
     0
     0
     0
     0
     0
     0
     0
     0
```

```
     0
     0
     0
```

Y, por último, reconstruimos los valores de U.

>> U=calculoU(X,4)

```
U =

     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Como conocemos la solución correcta de este problema, podemos calcularla para comprobar si esta aproximación es buena o no.

Sabemos que la solución es

$$u(x,y) = 400xy$$

Por lo que podemos construir un algoritmo que, dados a, b, c, d, n y m, nos calcule el valor de u(x,y) para los puntos representados en la malla. Dicho algoritmo es el siguiente:

```
function Z=u(a,b,c,d,n,m)
n1=(b-a)/n
n2=(d-c)/m
for j=m-1:-1:1
    y=c+j*n2;
    for i=1:n-1
        x=a+i*n1;
        Z(i,j)=400*x*y;
    end
end
```

Llamando al algoritmo, obtenemos que:

>> U=u(0,0.5,0,0.5,4,5)

```
U =
    5.0000   10.0000   15.0000
   10.0000   20.0000   30.0000
   15.0000   30.0000   45.0000
   20.0000   40.0000   60.0000
```

Solución que no tiene nada que ver con la que hemos intentado aproximar. Lo que está sucediendo es que la ordenación elegida es muy mala y el algoritmo no converge. Por eso vamos a elegir a continuación la ordenación arriba-abajo, izquierda-derecha, que es la óptima.

- ***Ejemplo 1.2***

Intentaremos resolver el ejemplo 1.1 con el algoritmo A1.2, por lo que los datos del problema son los mismos que para dicho ejemplo.
Primero calculamos el valor de las matrices A y b.

>> [A,b] = calculoAyb('f','g',0,0.5,0,0.5,4,5)

```
A =

  Columns 1 through 7

    5.1250   -1.0000         0   -1.5625         0         0         0
   -1.0000    5.1250   -1.0000         0   -1.5625         0         0
         0   -1.0000    5.1250         0         0   -1.5625         0
   -1.5625         0         0    5.1250   -1.0000         0   -1.5625
         0   -1.5625         0   -1.0000    5.1250   -1.0000         0
         0         0   -1.5625         0   -1.0000    5.1250         0
         0         0         0   -1.5625         0         0    5.1250
         0         0         0         0   -1.5625         0   -1.0000
         0         0         0         0         0   -1.5625         0
         0         0         0         0         0         0   -1.5625
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 12

         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
   -1.5625         0         0         0         0
         0   -1.5625         0         0         0
   -1.0000         0   -1.5625         0         0
    5.1250   -1.0000         0   -1.5625         0
```

```
   -1.0000    5.1250         0         0   -1.5625
         0         0    5.1250   -1.0000         0
   -1.5625         0   -1.0000    5.1250   -1.0000
         0   -1.5625         0   -1.0000    5.1250


b =

   39.0625
   78.1250
  197.1875
         0
         0
   60.0000
         0
         0
   40.0000
         0
         0
   20.0000
```

Luego hallamos X:

>> X=A\b'

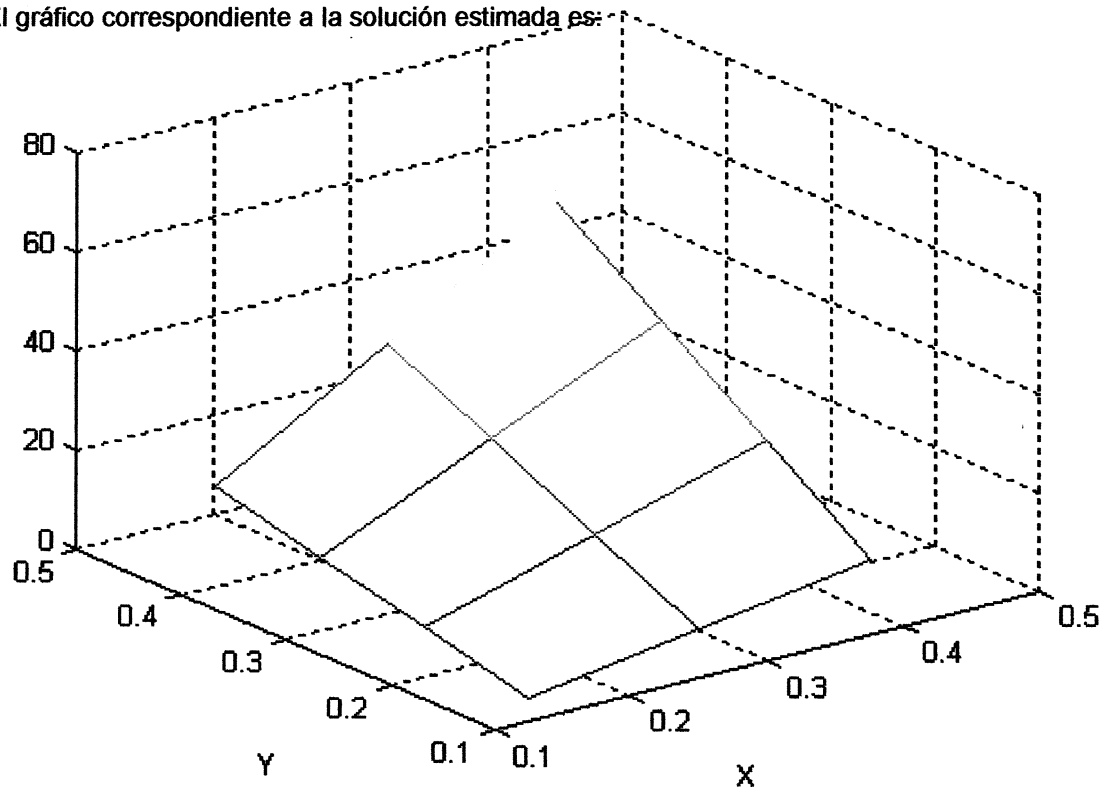
```
X =
   20.0000
   40.0000
   60.0000
   15.0000
   30.0000
   45.0000
   10.0000
   20.0000
   30.0000
    5.0000
   10.0000
   15.0000
```

Y, por último, reconstruimos los valores de U.

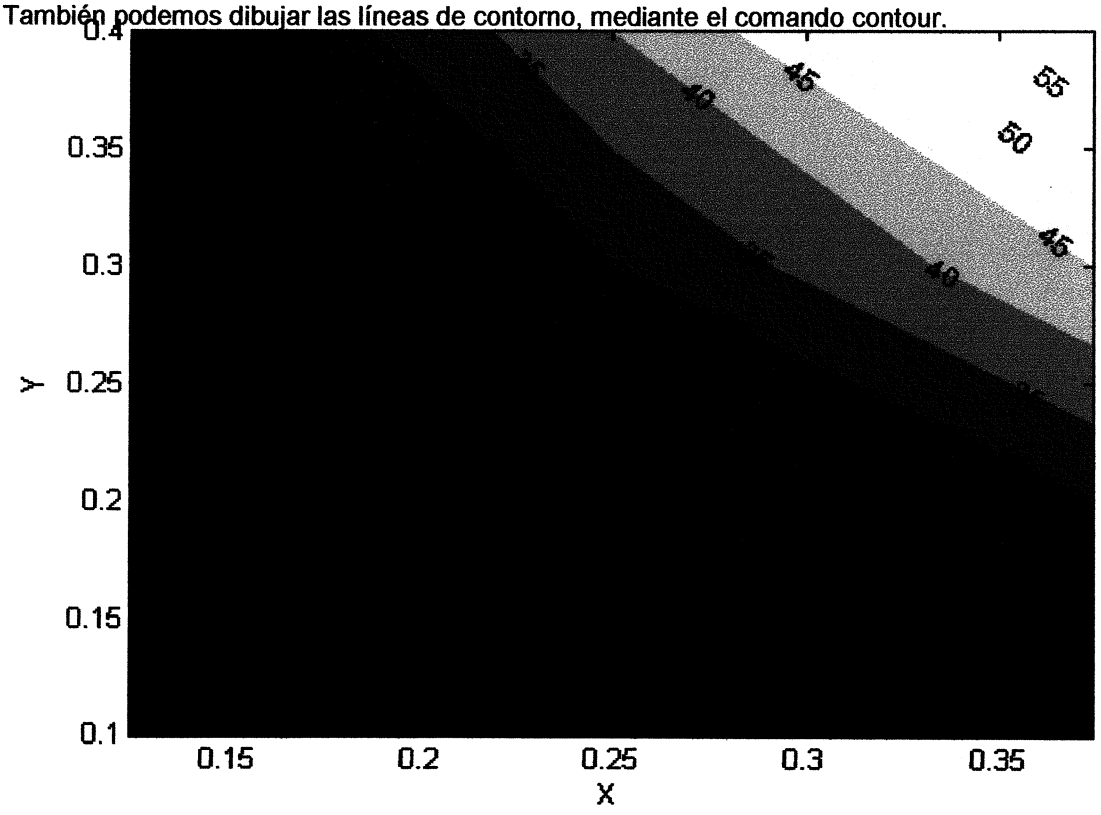
>> U=calculoU(X,4,5)

```
U =
    5.0000   10.0000   15.0000   20.0000
   10.0000   20.0000   30.0000   40.0000
   15.0000   30.0000   45.0000   60.0000
```

El gráfico correspondiente a la solución estimada es:



También podemos dibujar las líneas de contorno, mediante el comando contour.



Como conocemos la solución correcta de este problema, podemos calcularla para comprobar si esta aproximación es buena o no.

Sabemos que la solución es

$$u(x,y) = 400xy$$

Por lo que podemos construir un algoritmo que, dados a, b, c, d, n y m, nos calcule el valor de u(x,y) para los puntos representados en la malla. Dicho algoritmo es el siguiente:

```
function Z=u(a,b,c,d,n,m)
n1=(b-a)/n
n2=(d-c)/m
for i=1:n-1
    x=a+i*n1;
    for j=m-1:-1:1
        y=c+j*n2;
        Z(i,j)=400*x*y;
    end
end
```

>> U2=u(0,0.5,0,0.5,4,5)

```
U2 =
    5.0000   10.0000   15.0000   20.0000
   10.0000   20.0000   30.0000   40.0000
   15.0000   30.0000   45.0000   60.0000
```

Hemos obtenido el valor real de u(x,y), que como puede comprobarse es exactamente igual que el valor estimado con nuestro algoritmo. El algoritmo A1.2 tiene la ordenación óptima. Los siguientes ejemplos se basan todos en dicho algoritmo.

- ***Ejemplo 1.3***

Tenemos el siguiente planteamiento:

$$\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = 0$$

$$0 < x < 1$$
$$0 < y < 1$$

Y las siguientes condiciones de contorno:

$$u(0,y) = y, \quad 0 < y < 1$$
$$u(1,y) = y^2, \quad 0 < y < 1$$
$$u(x,0) = x, \quad 0 < x < 1$$
$$u(x,1) = x^2, \quad 0 < x < 1$$

No conocemos la solución, pero intentaremos aproximarla mediante el algoritmo A1.2. Primero definimos las funciones f y g:

f.m

```
function z=f(x,y)
z=0;
```

g.m

```
function z=g(x,y)
if x==0
    if y<1
        z=y;
    end
else if x==1
        if y>0
            z=y^2;
        end
    end
end
if y==0
    if x<1
        z=x;
    end
else if y==1
        if x>=0
            x^2;
        end
    end
end
```

Para llamar al algoritmo A1.2 usaremos los siguientes parámetros:

a=0, b=1, c=0, d=1, n=11, m=11

:

>> [A,b] = calculoAyb('f','g',0,1,0,1,11,11)

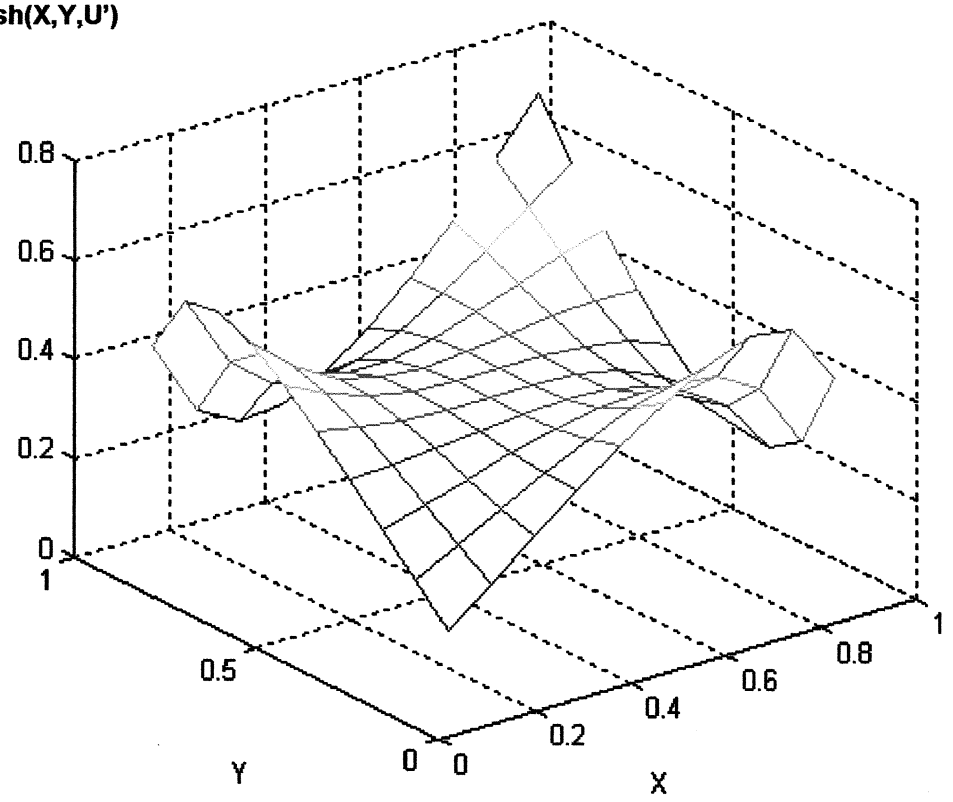
Debido a que sale una matriz excesivamente grande, no la pongo aquí puesto que sería difícil de visualizar. Directamente calculamos el vector X.

>> X=A\b

```
X =

    0.4395
    0.2800
    0.2355
    0.2441
    0.2836
    0.3443
    0.4214
    0.5115
    0.6120
    0.7192
    0.5608
    0.4120
    0.3433
    0.3252
    0.3394
    0.3748
    0.4246
    0.4838
    0.5479
    0.6120
    0.5736
    0.4637
    0.4005
    0.3742
    0.3738
    0.3909
    0.4186
    0.4512
    0.4838
    0.5115
    0.5426
    0.4687
    0.4210
    0.3970
    0.3909
    0.3964
    0.4075
    0.4186
    0.4246
    0.4214
    0.4916
    0.4476
```

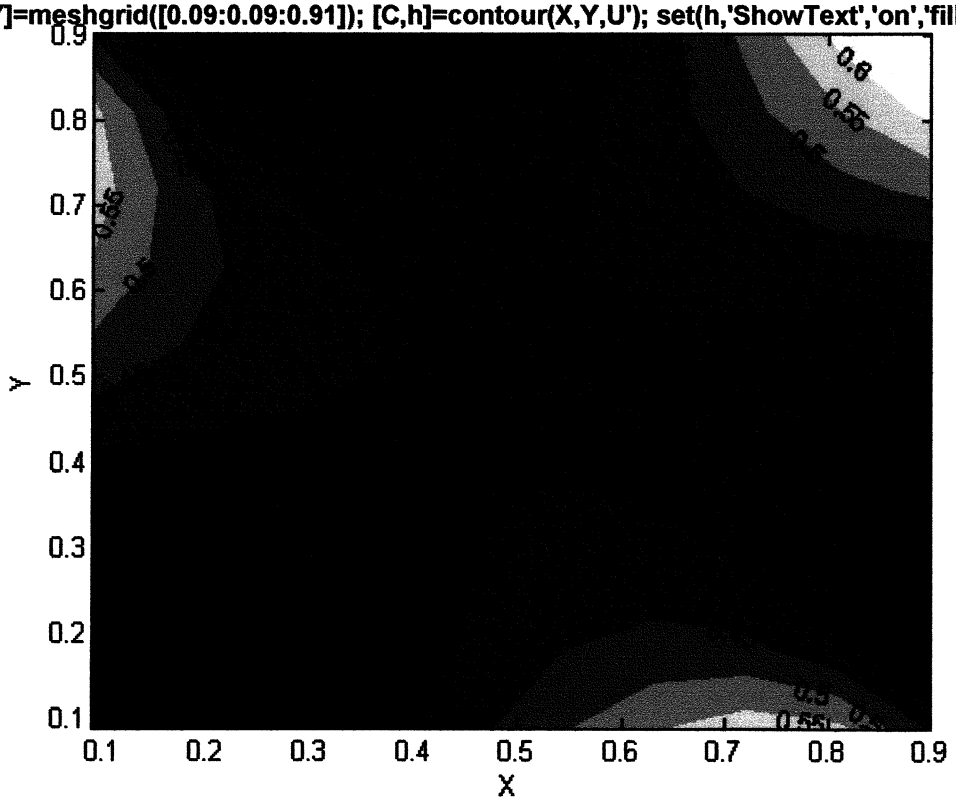
```
0.4178
0.4019
0.3964
0.3964
0.3964
0.3909
0.3748
0.3443
0.4309
0.4121
0.4007
0.3964
0.3964
0.3964
0.3909
0.3738
0.3394
0.2836
0.3653
0.3693
0.3766
0.3865
0.3964
0.4019
0.3970
0.3742
0.3252
0.2441
0.2975
0.3232
0.3499
0.3766
0.4007
0.4178
0.4210
0.4005
0.3433
0.2355
0.2288
0.2760
0.3232
0.3693
0.4121
0.4476
0.4687
0.4637
```

>> mesh(X,Y,U')



O también podemos dibujar las líneas de contorno:

>> [X,Y]=meshgrid([0.09:0.09:0.91]); [C,h]=contour(X,Y,U'); set(h,'ShowText','on','fill','on');



- ***Ejemplo 1.4***

Tenemos el siguiente planteamiento:

$$\frac{\partial^2 u}{\partial x^2}(x,y)+\frac{\partial^2 u}{\partial y^2}(x,y)=0$$

$$0<x<\pi$$
$$0<y<\pi/2$$

Y las siguientes condiciones de contorno:

$$u(0,y)=0, \quad 0<y<\pi/2$$
$$u(\pi,y)=e^{-y^2}sen(2\pi y), \quad 0<y<\pi/2$$
$$u(x,0)=0, \quad 0<x<\pi$$
$$u(x,\pi/2)=e^{-x^2}sen(\pi x), \quad 0<x<\pi$$

No conocemos la solución, pero intentaremos aproximarla mediante el algoritmo A1.2. Primero definimos las funciones f y g:

f.m

```
function z=f(x,y)
z=0;
```

g.m

```
function z=g(x,y)
if x==0
    if y<=pi/2
        if y>=0
            z=0;
        end
    end
else if x==pi
        if y>=0
            if y<=pi/2
                z=exp(-
y^2)*sin(2*pi*y);
            end
        end
    end
end
if y==0
    if x<=pi
        if x>=0
            z=0;
        end
```

```
    end
else if y==pi/2
        if x>=0
            if x<=pi
                z=exp(-
x^2)*sin(pi*x);
            end
        end
    end
end
```

Para llamar al algoritmo A1.2 usaremos los siguientes parámetros:

a=0, b=pi, c=0, d=pi/2, n=8, m=4

>> [A,b] = calculoAyb('f','g',0,pi,0,pi/2,8,4)

Debido a que sale una matriz excesivamente grande, no la pongo aquí puesto que sería difícil de visualizar. Directamente calculamos el vector X.

>> X=A\b

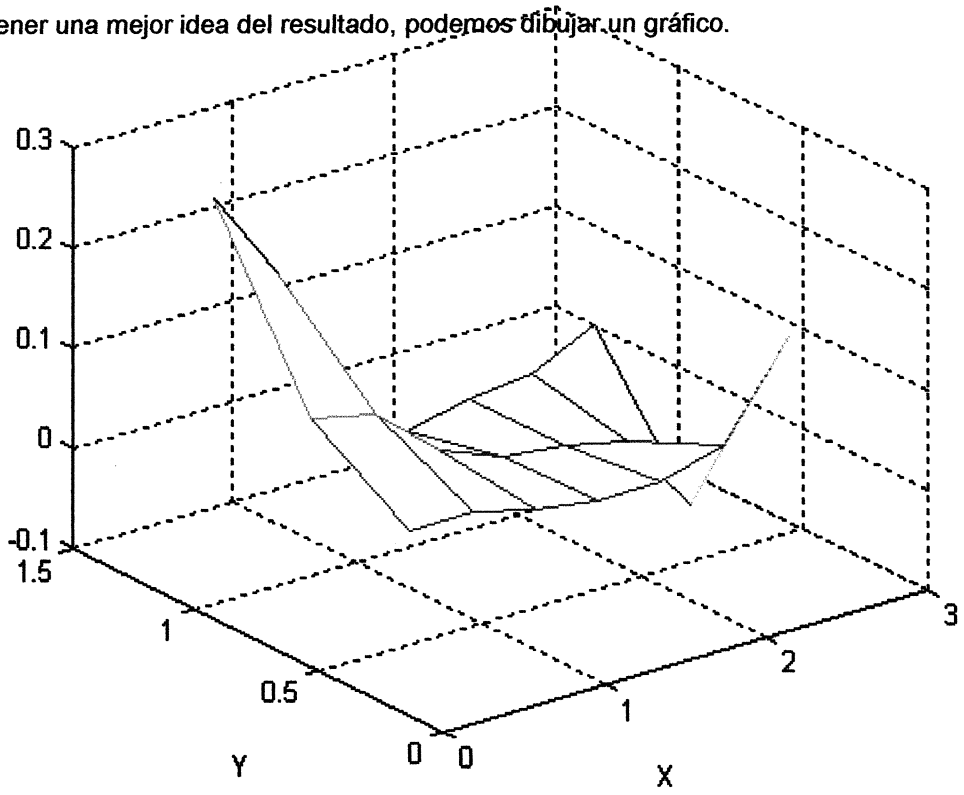
```
X =
    0.2706
    0.1765
    0.0143
   -0.0172
   -0.0047
    0.0032
    0.0323
    0.0969
    0.0843
    0.0303
    0.0045
   -0.0026
   -0.0180
   -0.0991
    0.0327
    0.0337
    0.0179
    0.0076
    0.0078
    0.0263
    0.1156
```

Y, por último, reconstruimos los valores de U.

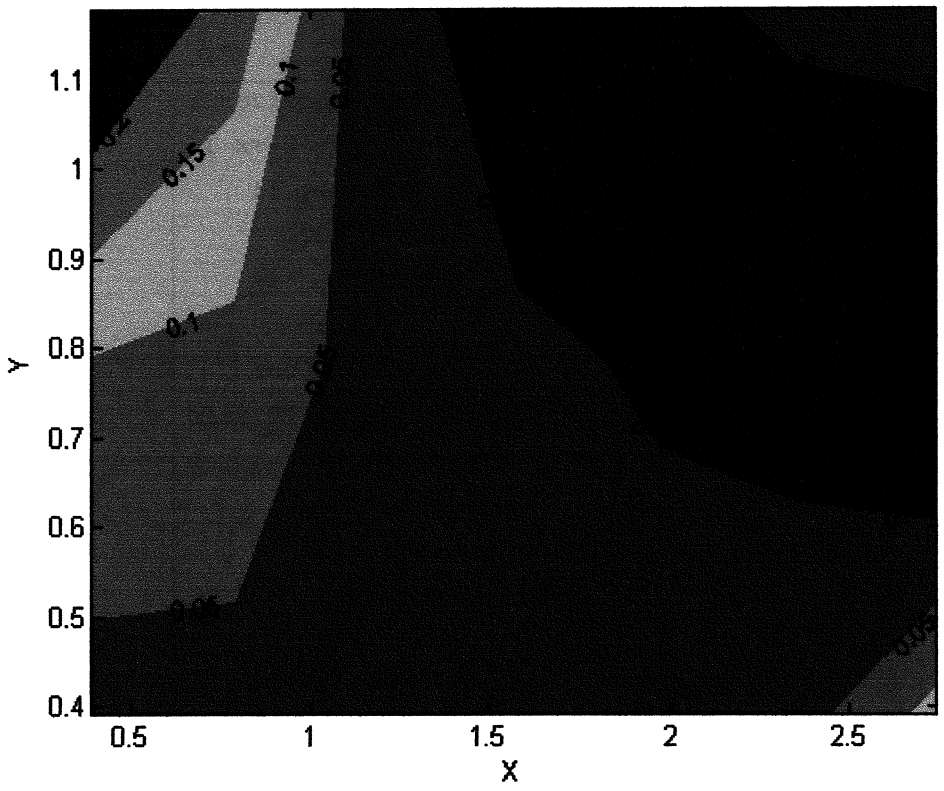
>> U=calculoU(X,8,4)

```
U =
    0.0327    0.0969    0.2706
    0.0337    0.0843    0.1765
    0.0179    0.0303    0.0143
    0.0076    0.0045   -0.0172
    0.0078   -0.0026   -0.0047
    0.0263   -0.0180    0.0032
    0.1156   -0.0991    0.0323
```

Para tener una mejor idea del resultado, podemos dibujar un gráfico.



O también podemos dibujar las líneas de contorno:



- ***Ejemplo 1.5***

Tenemos el siguiente planteamiento:

$$\frac{\partial^2 u}{\partial x^2}(x,y)+\frac{\partial^2 u}{\partial y^2}(x,y)=-y\,sen(x)-x\cos(y)$$

$$0<x<\pi$$
$$0<y<\pi/2$$

Y las siguientes condiciones de contorno:

$$u(0,y)=0,\quad 0<y<\pi/2$$
$$u(\pi,y)=\pi\cos(y),\quad 0<y<\pi/2$$
$$u(x,0)=x,\quad 0<x<\pi$$
$$u(x,\pi/2)=\pi/2\,sen(x),\quad 0<x<\pi$$

No conocemos la solución, pero intentaremos aproximarla mediante el algoritmo A1.2. Primero definimos las funciones f y g:

f.m

```
function z=f(x,y)
z=-y*sin(x)-x*cos(y);
```

2. Ecuación parabólica: ecuación del calor

2.1 Planteamiento del problema. Ecuación.

Esta ecuación también se conoce como ecuación de difusión, y en ella interviene la variable tiempo. La ecuación es:

Ecuación E2.1

$$\frac{\partial u}{\partial t}(x,t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x,t) + f(x,t)\,,\ t > 0\,,\ x_0 < x < L$$

Lo que podemos expresar también de la siguiente manera:

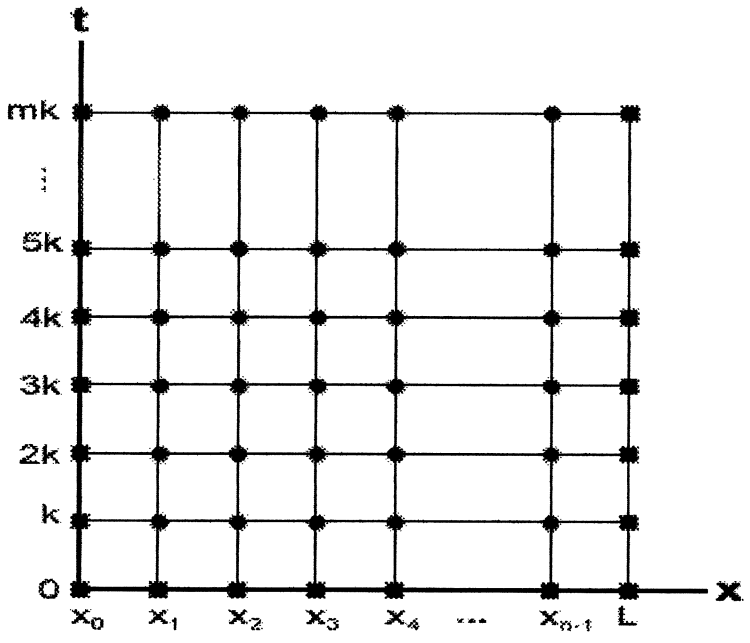
$$U_t = \alpha^2 U_{xx} + f$$

Y lo que conocemos es el contorno y la situación inicial, lo que expresamos así:

$$u(x_0,t) = h_1(t)$$
$$u(x_n,t) = h_2(t)$$

$$u(x,0) = g(x)$$

Nuestro problema consiste en calcular la temperatura en una barra (una línea) en un instante del tiempo determinado, siendo conocida la temperatura inicial y la de los extremos. Lo podemos expresar gráficamente:



Siendo los puntos redondos los que queremos calcular y, los cuadrados, los conocidos. Se observa que lo que vamos a hacer es una malla de puntos. El eje x lo dividiremos en n partes, mientras que el y lo dividiremos en m. Nuestro objetivo es calcular la temperatura de la barra en el instante k·m.

2.2 Método progresivo.

La primera forma de la que vamos a tratar de resolver este problema es mediante un método progresivo. Para ello el primer paso será discretizar el problema.

2.2.1 Discretización del problema.

Para este método usaremos la primera forma de discretización de la primera derivada y la discretización de la segunda derivada (ver apéndice).

Por tanto, tomando $x=x_i$, $t=t_j$, la ecuación E2.1 quedaría así:

$$\frac{u(x_i,t_j+k)-u(x_i,t_j)}{k}+O(k)=\alpha^2\frac{u(x_i+h,t_j)-2u(x_i,t_j)+u(x_i-h,t_j)}{h^2}+O(h^2)+f(x_i,t_j)$$

Si eliminamos los residuos y llamamos $u(x_i,t_j)\approx W_{i,j}$ y $\lambda=\alpha^2k/h^2$ la ecuación anterior se queda en:

$$W_{i,j+1}=(1-2\lambda)W_{i,j}+\lambda W_{i+1,j}+\lambda W_{i-1,j}+kf_{i,j}$$

Según la ecuación anterior, conociendo los puntos para un valor de t, podemos hallar los puntos para el valor de t inmediatamente superior. Por tanto, basta con hacer un algoritmo que lo calcule: aquí no hay sistema de ecuaciones, es un método explícito.

2.2.2 Algoritmo A2.1: Resolución de la ecuación del calor mediante método progresivo.

El método progresivo puede expresarse matricialmente de la siguiente manera:

$$\underbrace{\begin{pmatrix} W_{1,j+1}\\ W_{2,j+1}\\ W_{3,j+1}\\ W_{4,j+1}\\ \vdots\\ W_{n-1,j+1}\end{pmatrix}}_{W_{j+1}}=\underbrace{\begin{pmatrix} 1-2\lambda & \lambda & 0 & 0 & 0 & 0 & \cdots & 0 & 0\\ \lambda & 1-2\lambda & \lambda & 0 & 0 & 0 & \cdots & 0 & 0\\ 0 & \lambda & 1-2\lambda & \lambda & 0 & 0 & \cdots & 0 & 0\\ 0 & 0 & 0 & \lambda & 1-2\lambda & \lambda & \cdots & 0 & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \lambda & 1-2\lambda\end{pmatrix}}_{A}\underbrace{\begin{pmatrix} W_{1,j}\\ W_{2,j}\\ W_{3,j}\\ W_{4,j}\\ \vdots\\ W_{n-1,j}\end{pmatrix}}_{W_j}+\underbrace{\begin{pmatrix} kf_{1,j}-\lambda h_1(t_j)\\ kf_{2,j}\\ kf_{3,j}\\ kf_{4,j}\\ \vdots\\ kf_{n-1,j}-\lambda h_2(t_j)\end{pmatrix}}_{b}$$

Resumiendo, el método progresivo es:

$$\begin{cases} W_{j+1}=AW_j+b\\ W_0=\{g(x_i)\}_{i=1\ldots n-1}\end{cases}$$

Este método tiene una estabilidad y una convergencia bastante malas. Para que este método sea estable, λ tiene que ser menor que1/2. Es decir, k tiene que ser mucho más pequeño que el cuadrado de h. El error de convergencia es $O(k+h^2)$.

A continuación tenemos el algoritmo en Matlab del método progresivo.

```
        xi=x0+i*h;
        if i==1
            b(i)=k*feval(f,xi,tj)-lambda*feval(h1,tj);
        else if i==n-1
                b(i)=k*feval(f,xi,tj)-lambda*feval(h2,tj);
            else
                b(i)=k*feval(f,xi,tj);
            end
        end
    end
    for i=1:n-1
        WJ(i)=A(i,:)*W(:,j)+b(i);
    end
    W(:,j+1)=WJ;
end
```

2.3 Método regresivo.

Ya que el método anterior no es incondicionalmente estable, nuestro objetivo ahora es conseguir un método que sí lo sea

2.3.1 Discretización del problema.

Para este método usaremos la tercera forma de discretización de la primera derivada y la discretización de la segunda derivada (ver apéndice).

Por tanto, tomando $x=x_i$, $t=t_j$, la ecuación E2.1 quedaría así:

$$\frac{u(x_i,t_j-k)-u(x_i,t_j)}{-k}+O(k)=\alpha^2\frac{u(x_i+h,t_j)-2u(x_i,t_j)+u(x_i-h,t_j)}{h^2}+O(h^2)+f(x_i,t_j)$$

Si eliminamos los residuos y llamamos $u(x_i,t_j)\approx W_{i,j}$ y $\lambda=\alpha^2k/h^2$ la ecuación anterior se queda en:

$$W_{i,j-1}-W_{i,j}=-\lambda W_{i+1,j}+2\lambda W_{i,j}-\lambda W_{i-1,j}-kf_{i,j}$$

Y, agrupando:

Ecuación E2.2

$$(1+2\lambda)W_{i,j}-\lambda W_{i+1,j}-\lambda W_{i-1,j}=kf_{i,j}+W_{i,j-1}$$

Esta ecuación nos da el método regresivo.

2.3.2 Algoritmo A2.2: Resolución de la ecuación del calor mediante método regresivo.

Si representamos matricialmente la ecuación E2.2, obtenemos lo siguiente:

$$\underbrace{\begin{pmatrix} 1+2\lambda & -\lambda & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & -\lambda & 1+2\lambda & -\lambda & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & -\lambda & 1+2\lambda \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} W_{1,j} \\ W_{2,j} \\ W_{3,j} \\ W_{4,j} \\ \vdots \\ W_{n-1,j} \end{pmatrix}}_{W_j} = \underbrace{\begin{pmatrix} kf_{1,j} + W_{1,j-1} + \lambda h_1(t_j) \\ kf_{2,j} + W_{2,j-1} \\ kf_{3,j} + W_{3,j-1} \\ \vdots \\ kf_{n-2,j} + W_{n-2,j-1} \\ kf_{n-1,j} + W_{n-1,j-1} + \lambda h_2(t_j) \end{pmatrix}}_{W_{j-1}+b}$$

Por tanto el método queda expresado así:

$$\begin{cases} AW_j = W_{j-1} + b \\ W_0 = \{g(x_i)\}_{i=1\ldots n-1} \end{cases}$$

En este caso ya no tenemos un método explícito, sino implícito, donde tendremos que resolver un sistema de ecuaciones lineales. En este método, la estabilidad es buena para cualquier valor de λ, es decir, es incondicionalmente estable. Sin embargo la convergencia sigue siendo igual de mala que la del método progresivo ($O(k+h^2)$).

A continuación tenemos el algoritmo en Matlab del método regresivo.

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- f: función f(x,t)
- g: función g(x)
- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y
- alpha: valor de α

Datos que devuelve el algoritmo:

- W: matriz de n-1 filas y m+1 columnas. El valor (i,j) de dicha matriz representa el valor de x_i en el instante (j*k)-1.

```
function W = metodoregesivo(x0,n,L,k,f,g,h1,h2,m,alpha)
h=(L-x0)/n;
lambda=alpha^2*k/h^2;
W=zeros(n-1,m+1);
W0=zeros(n-1,1);
for i=1:n-1
    x=x0+i*h;
```

```
    W0(i)=feval(g,x);
end
W(:,1)=W0;
A=zeros(n-1);
for i=1:n-1
    A(i,i)=1+2*lambda;
    if i>1
        A(i,i-1)=-1*lambda;
    end
    if i<n-1
        A(i,i+1)=-1*lambda;
    end
end
for j=1:m
    WJ=zeros(n-1,1);
    tj=j*k;
    b=zeros(n-1,1);
    for i=1:n-1
        xi=x0+i*h;
        if i==1
            b(i)=k*feval(f,xi,tj)+W(i,j)+lambda*feval(h1,tj);
        else if i==n-1
                b(i)=k*feval(f,xi,tj)+W(i,j)+lambda*feval(h2,tj);
            else
                b(i)=k*feval(f,xi,tj)+W(i,j);
            end
        end
    end
    WJ=A\b; %b ya tiene incluido Wj-1 (metodo: AWj=Wj-1 + b)
    %en este caso seria AWj=b2, donde b2=Wj-1 + b
    W(:,j+1)=WJ;
end
```

2.4 Método de Crank-Nicolson.

Con el método regresivo ya habíamos solucionado el problema de la estabilidad en el método progresivo, sin embargo nos queda por encontrar un método que, además de eso, nos proporcione una buena convergencia. Dicho método es el método de Crank-Nicolson.

2.4.1 Discretización del problema.

El método de Crank-Nicolson lo que hace es un promedio entre la diferencia progresiva en el paso j en t,

$$W_{i,j+1} = (1-2\lambda)W_{i,j} + \lambda W_{i+1,j} + \lambda W_{i-1,j} + kf_{i,j}$$

y la diferencia regresiva en el paso (j+1) en t,

$$(1+2\lambda)W_{i,j+1} - \lambda W_{i+1,j+1} - \lambda W_{i-1,j+1} = kf_{i,j+1} + W_{i,j}$$

Por tanto, dicho método de diferencia promediada, llamado método de Crank-Nicolson, quedaría así:

Ecuación E2.3

$$W_{i,j+1} - W_{i,j} - \frac{\lambda}{2}\left[W_{i+1,j} - 2W_{i,j} + W_{i-1,j} + W_{i+1,j+1} - 2W_{i,j+1} + W_{i-1,j+1}\right] = k\frac{f_{i,j} + f_{i,j+1}}{2}$$

2.4.2 Algoritmo A2.3: Resolución de la ecuación del calor mediante método Crank-Nicolson.

Si representamos matricialmente la ecuación E2.3, obtenemos lo siguiente:

$$\underbrace{\begin{pmatrix} 1+\lambda & -\lambda/2 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ -\lambda/2 & 1+\lambda & -\lambda/2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\lambda/2 & 1+\lambda & -\lambda/2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & -\lambda/2 & 1+\lambda & -\lambda/2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & -\lambda/2 & 1+\lambda \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} W_{1,j+1} \\ W_{2,j+1} \\ W_{3,j+1} \\ W_{4,j+1} \\ \vdots \\ W_{n-1,j+1} \end{pmatrix}}_{W_{j+1}} =$$

$$= \underbrace{\begin{pmatrix} 1-\lambda & \lambda/2 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \lambda/2 & 1-\lambda & \lambda/2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \lambda/2 & 1-\lambda & \lambda/2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \lambda/2 & 1-\lambda & \lambda/2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \lambda/2 & 1-\lambda \end{pmatrix}}_{B} \underbrace{\begin{pmatrix} W_{1,j} \\ W_{2,j} \\ W_{3,j} \\ W_{4,j} \\ \vdots \\ W_{n-1,j} \end{pmatrix}}_{W_j} +$$

$$+ \underbrace{\begin{pmatrix} k\frac{f_{1,j} + f_{1,j+1}}{2} + \frac{\lambda}{2}h_1(t_j) + \frac{\lambda}{2}h_1(t_{j+1}) \\ k\frac{f_{2,j} + f_{2,j+1}}{2} \\ k\frac{f_{3,j} + f_{3,j+1}}{2} \\ \vdots \\ k\frac{f_{n-2,j} + f_{n-2,j+1}}{2} \\ k\frac{f_{n-1,j} + f_{n-1,j+1}}{2} + \frac{\lambda}{2}h_2(t_j) + \frac{\lambda}{2}h_2(t_{j+1}) \end{pmatrix}}_{b}$$

Por tanto el método queda expresado así:

$$\begin{cases} AW_{j+1} = BW_j + b \\ W_0 = \{g(x_i)\}_{i=1\ldots n-1} \end{cases}$$

Este método es incondicionalmente estable y su convergencia es buena, de $O(k^2+h^2)$.

A continuación tenemos el algoritmo en Matlab del método Crank-Nicolson.

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- f: función f(x,t)
- g: función g(x)
- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y
- alpha: valor de α

Datos que devuelve el algoritmo:

- W: matriz de n-1 filas y m+1 columnas. El valor (i,j) de dicha matriz representa el valor de x_i en el instante (j*k)-1.

```
function W = metodocranknic(x0,n,L,k,f,g,h1,h2,m,alpha)
h=(L-x0)/n;
lambda=alpha^2*k/h^2;
W=zeros(n-1,m+1);
W0=zeros(n-1,1);
for i=1:n-1
    x=x0+i*h;
    W0(i)=feval(g,x);
end
W(:,1)=W0;
A=zeros(n-1);
for i=1:n-1
    A(i,i)=1+lambda;
    if i>1
        A(i,i-1)=-1*lambda/2;
    end
    if i<n-1
        A(i,i+1)=-1*lambda/2;
    end
end
```

```
B=zeros(n-1);
for i=1:n-1
    B(i,i)=1-lambda;
    if i>1
        B(i,i-1)=lambda/2;
    end
    if i<n-1
        B(i,i+1)=lambda/2;
    end
end
for j=1:m
    WJ=zeros(n-1,1);
    tj=j*k;
    tj1=(j+1)*k;
    b=zeros(n-1,1);
    for i=1:n-1
        xi=x0+i*h;
        if i==1
b(i)=(lambda/2)*feval(h1,tj)+k*((feval(f,xi,tj)+feval(f,xi,tj1))/2)+(l
ambda/2)*feval(h1,tj1);
        else if i==n-1
b(i)=(lambda/2)*feval(h2,tj)+k*((feval(f,xi,tj)+feval(f,xi,tj1))/2)+(l
ambda/2)*feval(h2,tj1);
            else
                b(i)=k*(feval(f,xi,tj)+feval(f,xi,tj1))/2;
            end
        end
    end
    C=zeros(n-1,1);
    for i=1:n-1
        C(i)=B(i,:)*W(:,j)+b(i);
    end
    WJ=A\C;
    W(:,j+1)=WJ;
end
```

2.5 Ejemplos

• *Ejemplo 2.1*

Tenemos el siguiente planteamiento:

$$u_t = u_{xx} + e^x(-\cos(t) - sen(t))$$

Y las siguientes condiciones de contorno:

$$u(0,t) = \cos(t)$$
$$u(1,t) = e \cdot \cos(t)$$

Con la condición inicial:

$$u(x,0) = e^x$$

La solución al problema es:

$$u(x,t) = e^x \cdot \cos(t)$$

A continuación vamos a intentar aproximar la solución mediante los tres métodos que hemos visto.

MEDIANTE EL MÉTODO PROGRESIVO

Llamamos al algoritmo A2.1 con los siguientes parámetros:

$$x0=0;n=10;L=1;k=0.001;m=50;\alpha=1$$

Las funciones f, g, h1 y h2 son:

f.m

```
function z=f(x,t)
z=exp(x)*(-cos(t)-sin(t));
```

g.m

```
function z=g(x)
z=exp(x);
```

h1.m

```
function z=h1(t)
z=cos(t);
```

h2.m

```
function z=h2(t)
z=exp(1)*cos(t);
```

>> W=metodoprogresivo(0,10,1,0.001,'f','g','h1','h2',50,1)

```
    2.2228
    2.4565
```

Ahora vamos a comparer el resultado obtenido con la solución real:

>> N=abs(U-W(:,51))

```
N =
    1.5281
    1.1239
    0.8451
    0.7439
    0.8667
    1.2542
    1.9326
    2.8975
    4.0972
```

>> norm(N,inf)

```
ans =
    4.0972
```

Puede observarse, por los resultados obtenidos, que la aproximación es muy mala. A continuación utilizaremos los métodos regresivo y Crank-Nicolson para obtener una aproximación mejor.

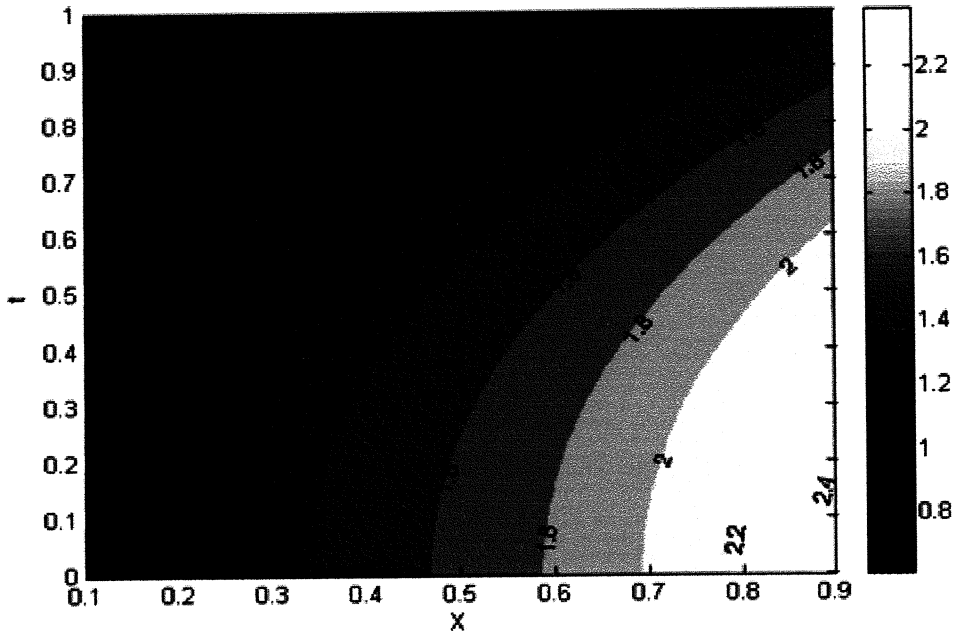
MEDIANTE EL MÉTODO REGRESIVO

Llamamos al algoritmo A2.2 con los mismos datos que en el caso anterior.

>> W=metodoregresivo(0,10,1,0.001,'f','g','h1','h2',50,1)

Al igual que antes, nos quedaremos con la última columna, correspondiente al instante t=0,05 segundos:

```
W(:,51) =
      1.21989145523137
      1.34819159121994
      1.48998347706017
      1.64668638103507
      1.81986843980261
      2.01126228609811
      2.22278233804944
      2.45654395443074
```



MEDIANTE EL MÉTODO DE CRANK-NICOLSON

Llamamos al algoritmo A2.3 con los mismos datos que en el caso anterior.

>> W=metodocranknic(0,10,1,0.001,'f','g','h1','h2',50,1)

Al igual que antes, nos quedaremos con la última columna, correspondiente al instante t=0,05 segundos:

```
W(:,51) =
         1.21985295097921
         1.34815371863190
         1.48994370101291
         1.64664238949506
         1.81981756333287
         2.01120081414481
         2.22270476144078
         2.45644220325031
```

Si la comparamos con la solución real:

>> X=[0.1:0.1:0.9];

>> U=u(X,0.05)'

```
U =
   1.10378974220900
   1.21987632275960
   1.34817183556297
   1.48996030523286
   1.64666079843047
```

```
   1.81984162636058
   2.01123604095721
   2.22275958185151
   2.45652924773627
```

Ahora vamos a comparer el resultado obtenido con la solución real:
>> N=abs(U-W(:,51))

```
N =
  1.0e-004 *
   0.33567843457760
   0.23371780392356
   0.18116931068235
   0.16604219954974
   0.18408935411829
   0.24063027717069
   0.35226812398648
   0.54820410731349
   0.87044485964416
```

>> norm(N,inf)

```
ans =
  8.704448596441594e-005
```

Es decir, la diferencia mayor entre la solución aproximada y la real es de aproximadamente 0,00008. Recordemos que en el caso del método regresivo, la diferencia máxima era de unos 0,00002, por tanto la solución aproximada mediante el método regresivo era mejor que la aproximada por Crank-Nicolson.

Podemos comparar las soluciones de ambos métodos para el instante t=1.

>> W=metodoregresivo(0,10,1,0.01,'f','g','h1','h2',100,1);
>> X=[0.1:0.1:0.9];
>> U=u(X,1)';
>> N=abs(U-W(:,101));
>> norm(N,inf)

```
ans =
  5.473208331819857e-004
```

>> W=metodocranknic(0,10,1,0.01,'f','g','h1','h2',100,1);
>> N=abs(U-W(:,101));
>> norm(N,inf)

```
ans =
  0.02071918600716
```

De nuevo, la solución aproximada mediante el método regresivo es mejor que la aproximada mediante el método de Crank-Nicolson.

• *Ejemplo 2.2*

Tenemos el siguiente planteamiento:

$$u_t = 0.86u_{xx}$$
$$0 < x < 2$$
$$0 < t < 1$$

Y las siguientes condiciones de contorno:

$$u(0,t) = 0$$
$$u(2,t) = 0$$

Con la condición inicial:

$$u(x,0) = \begin{cases} 4x/3 & \text{si } 0 \le x \le 0.75; \\ 1 & \text{si } 0.75 \le x \le 1.25; \\ 4(2-x)/3 & \text{si } 1.25 \le x \le 2; \end{cases}$$

A continuación vamos a intentar aproximar la solución mediante el método regresivo y el método de Crank-Nicolson.

MEDIANTE EL MÉTODO REGRESIVO

Llamamos al algoritmo A2.2 con los siguientes parámetros:

x0=0;n=20;L=2;k=0.001;m=500;α=0.86

Las funciones f, g, h1 y h2 son:

f.m

```
function z=f(x,t)
z=0;
```

g.m

```
function z=g(x)
if x>=0
   if x<0.75
      z=4*x/3;
   end
   if x>=0.75
```

```
    if x<1.25
       z=1;
    end
    if x>=1.25
       if x<2
          z=4*(2-x)/3;
       end
    end
  end
end
```

h1.m

```
function z=h1(t)
z=0;
```

h2.m

```
function z=h2(t)
z=0;
```

>> W=metodogresivo(0,20,2,0.001,'f','g','h1','h2',500,0.86)

Del resultado nos quedamos con el valor de la última columna, que equivale al valor de la temperatura en el instante t=m*k=500*0.001=0,5 segundos.

```
W(:,501) =
         0.06281812736870
         0.12409080869845
         0.18231035882396
         0.23604374843011
         0.28396777462535
         0.32490166624318
         0.35783631586391
         0.38195938442962
         0.39667560426150
         0.40162171503981
         0.39667560426149
         0.38195938442962
         0.35783631586391
         0.32490166624318
         0.28396777462535
         0.23604374843011
         0.18231035882396
```

```
        0.12409080869845
        0.06281812736870
```

Cada valor corresponde a un x_i.

Ahora vamos a hallar la temperatura para t=1 segundo.

>> W=metodoregresivo(0,20,2,0.001,'f','g','h1','h2',1000,0.86)

```
W(:,1001) =
        0.02529513529798
        0.04996742086771
        0.07340934350972
        0.09504368546738
        0.11433773740491
        0.13081641548715
        0.14407395957333
        0.15378392446483
        0.15970721817053
        0.16169798923887
        0.15970721817053
        0.15378392446483
        0.14407395957333
        0.13081641548715
        0.11433773740490
        0.09504368546738
        0.07340934350972
        0.04996742086771
        0.02529513529798
```

MEDIANTE EL MÉTODO CRANK-NICOLSON

Podemos calcular la solución en t=1 por el método Crank-Nicolson para así compararla con la solución dada por el método regresivo.

>> W=metodocranknic(0,20,2,0.001,'f','g','h1','h2',1000,0.86)

```
W(:,1001) =
        0.02525326206755
        0.04988470541002
        0.07328782245629
        0.09488635095076
        0.11414846340839
        0.13059986247944
        0.14383545977091
```

```
        0.15352935054323
        0.15944283865546
        0.16143031413886
        0.15944283865546
        0.15352935054323
        0.14383545977091
        0.13059986247944
        0.11414846340840
        0.09488635095077
        0.07328782245629
        0.04988470541002
        0.02525326206755
```

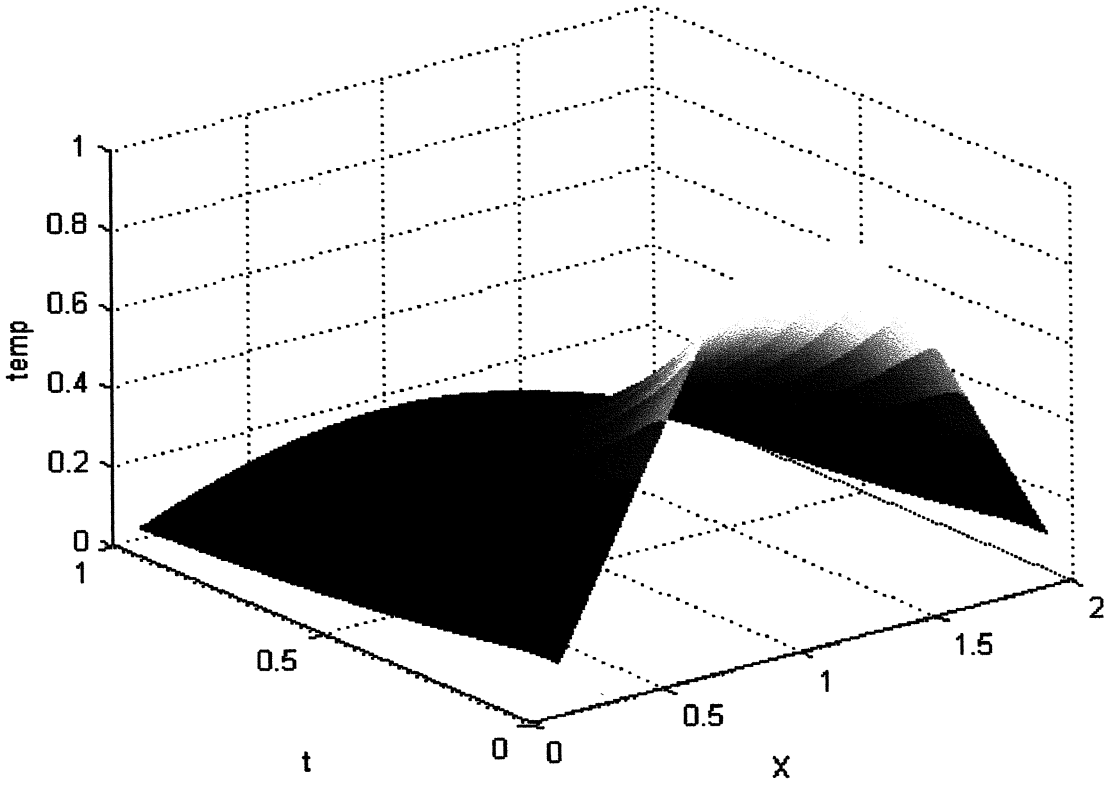
La diferencia entre la solución del regresivo y la del Crank-Nicolson es la siguiente:

```
  1.0e-003 *
   0.04187323043006
   0.08271545769854
   0.12152105343072
   0.15733451661425
   0.18927399651042
   0.21655300770826
   0.23849980241866
   0.25457392159586
   0.26437951506741
   0.26767510001804
   0.26437951506664
   0.25457392159414
   0.23849980241630
   0.21655300770523
   0.18927399650663
   0.15733451661049
   0.12152105342757
   0.08271545769660
   0.04187323042911
```

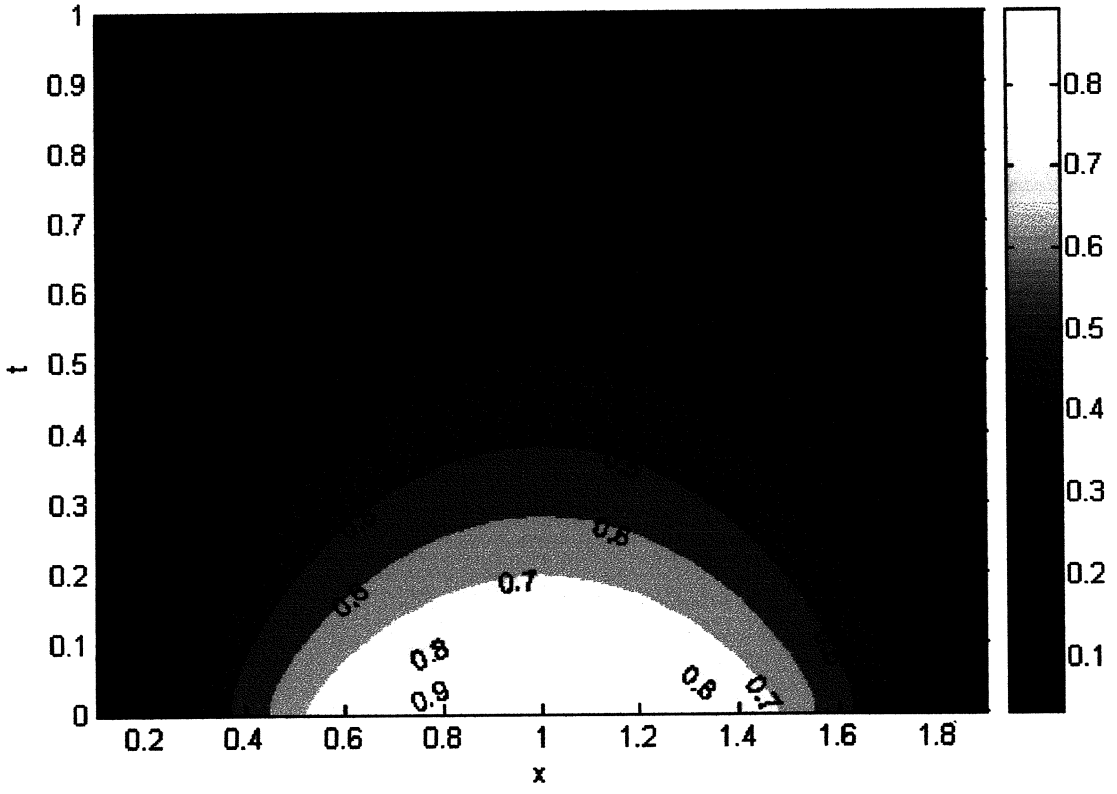
Siendo la norma infinita:

```
ans =
    2.676751000180444e-004
```

Lo que quiere decir que ambos métodos nos han proporcionado soluciones muy similares. A continuación tenemos un gráfico de la solución hallada por Crank-Nicolson:



Y también tenemos las curvas de nivel:



• *Ejemplo 2.3*

Tenemos el siguiente planteamiento:

$$u_t = \frac{1}{100} u_{xx}$$
$$0 < x < 1$$
$$0 < t < 1$$

Y las siguientes condiciones de contorno:

$$u(0,t) = 0$$
$$u(1,t) = 0$$

Con la condición inicial:

$$u(x,0) = x^2(1-x)$$

A continuación vamos a intentar aproximar la solución mediante el método regresivo y el método de Crank-Nicolson.

MEDIANTE EL MÉTODO REGRESIVO

Llamamos al algoritmo A2.2 con los siguientes parámetros:

x0=0;n=10;L=1;k=0.001;m=100;α=0.01

Las funciones f, g, h1 y h2 son:

f.m

```
function z=f(x,t)
z=0;
```

g.m

```
function z=g(x)
z=x^2*(1-x);
```

h1.m

```
function z=h1(t)
z=0;
```

h2.m

```
function z=h2(t)
z=0;
```

>> W=metodoregresivo(0,10,1,0.001,'f','g','h1','h2',100,0.01)

Del resultado nos quedamos con el valor de la última columna, que equivale al valor de la temperatura en el instante t=m*k=100*0.001=0,1 segundos.

```
W(:,101) =
   0.00901398990686
   0.03200799999657
   0.06300200000000
   0.09599600000000
   0.12499000000000
   0.14398400000000
   0.14697800000000
   0.12797200000686
   0.08096602018627
```

Vamos a hallar a continuación el valor de la temperatura en el instante 0,8 segundos:

>> W=metodoregresivo(0,10,1,0.001,'f','g','h1','h2',800,0.01)

```
W(:,801) =
   0.00911136260903
   0.03206399830060
   0.06301599999659
   0.09596799999999
   0.12491999999999
   0.14387200000000
   0.14682400000680
   0.12777600339878
   0.08072927478194
```

MEDIANTE EL MÉTODO DE CRANK-NICOLSON

Vamos a calcular también el valor de la temperatura en t=0,8 segundos:

>> W=metodocranknic(0,10,1,0.001,'f','g','h1','h2',800,0.01)

```
W(:,801) =
   0.00911136339635
   0.03206399830691
   0.06301599999662
   0.09596800000000
   0.12492000000001
   0.14387200000002
   0.14682400000677
   0.12777600338619
   0.08072927320731
```

La diferencia entre la solución del regresivo y la del Crank-Nicolson es la siguiente:

```
1.0e-008 *
   0.07873203252545
   0.00063069480194
   0.00000342642581
   0.00000142663659
   0.00000183048021
   0.00000213162821
   0.00000277555756
   0.00125850441179
   0.15746261794458
```

Siendo la norma infinita:

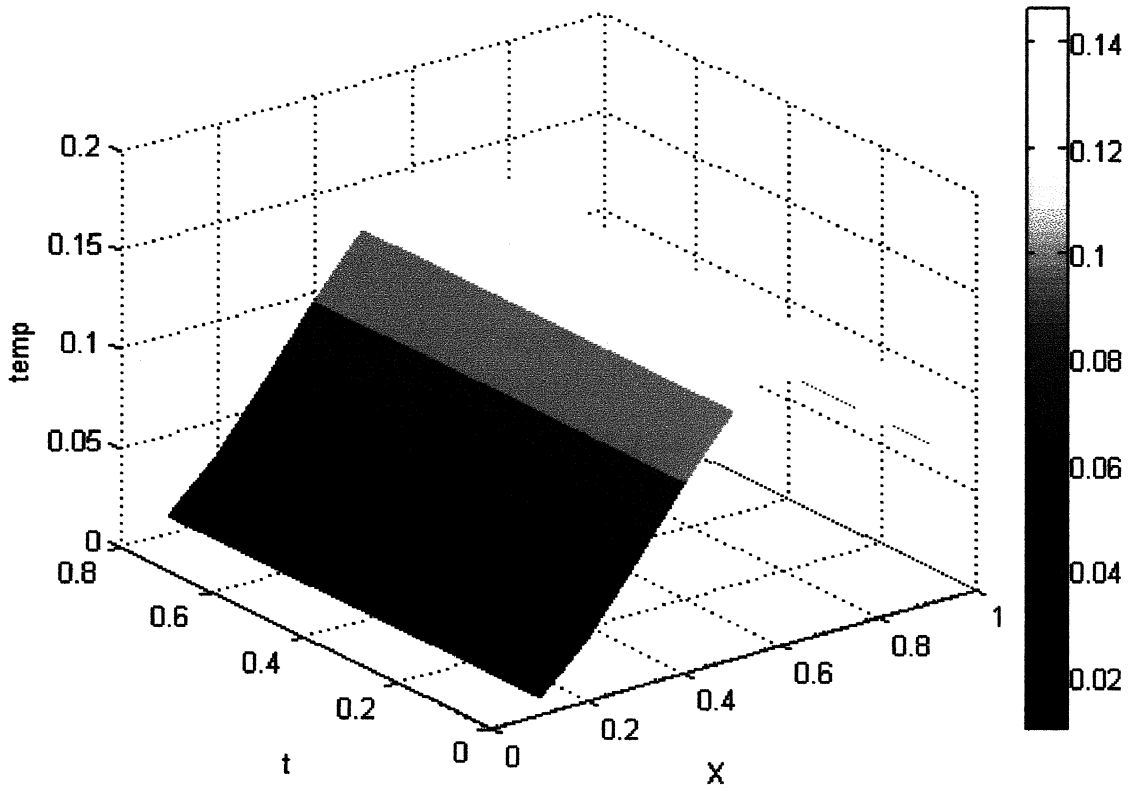
```
ans =
    1.574626179445815e-009
```

Lo que quiere decir que ambos métodos nos han proporcionado soluciones muy similares. En general podemos calcular la diferencia de todas las aproximaciones de las temperaturas de los x_i para cada tiempo $t=t_j$ hasta t=0,8. Sería así:

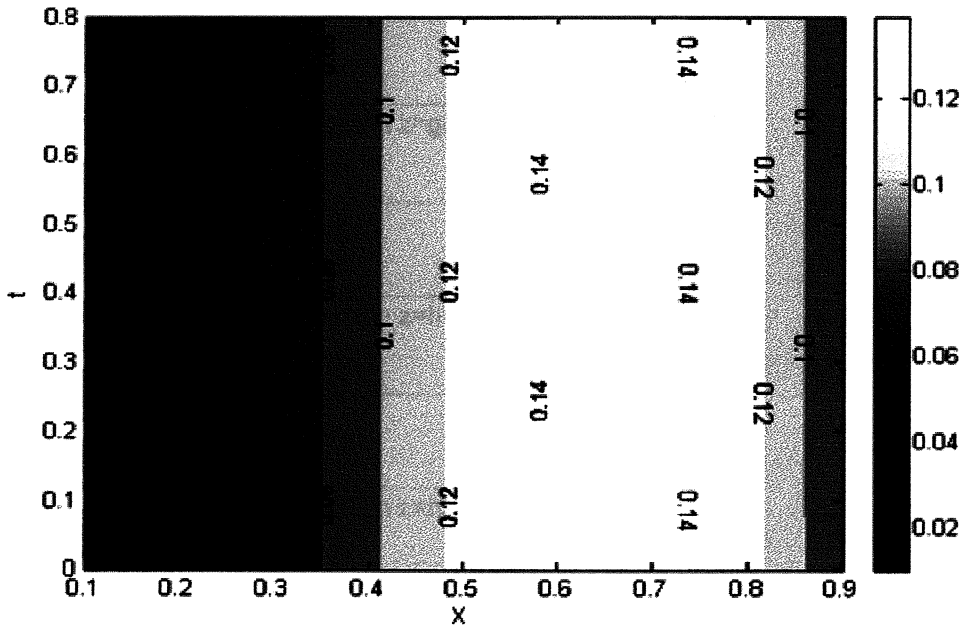
```
>> W1=metodoregresivo(0,10,1,0.001,'f','g','h1','h2',800,0.01);
>> W2=metodocranknic(0,10,1,0.001,'f','g','h1','h2',800,0.01);
>> norm(abs(W1-W2),inf)
```

```
ans =
  6.340001002524964e-007
```

A continuación tenemos el gráfico correspondiente a la solución aproximada por Crank-Nicolson:



Y a continuación tenemos las curvas de nivel:



- ***Ejemplo 2.4***

Tenemos el siguiente planteamiento:

```
   0.04424332802440
  -0.02948417098452
```

>> u(X,0.1)'

```
ans =
   0.00200000000000
   0.00800000000000
   0.01800000000000
   0.03200000000000
   0.05000000000000
   0.07200000000000
   0.09800000000000
   0.12800000000000
   0.16200000000000
```

>> N=abs(U-W(:,101))

```
N =
   0.00320882386967
   0.00452455991080
   0.00515110317333
   0.00594108698906
   0.00822057240330
   0.01524576511502
   0.03488943205404
   0.08375667197560
   0.19148417098452
```

>> norm(N,inf)

```
ans =
   0.19148417098452
```

Con estos datos podemos afirmar que la aproximación para t=0.1 segundos es mejor que la aproximación paa t=1 segundo. Esto es debido a la poca estabilidad que tiene este método, así como su mala convergencia.

- ***Ejemplo 2.6***

Volvemos a aproximar la solución al ejemplo 2.4, para t=1 segundo, pero esta vez con Crank-Nicolson.

>> W=metodocranknic(0,10,1,0.001,'f','g','h1','h2',1000,0.5);

>> W(:,1001)

```
ans =
  -0.09980479683510
  -0.12617704894934
  -0.08421829575218
   0.02246704814979
   0.19173297029891
   0.42286699658802
   0.71658162081912
   1.07502286762196
   1.50179515160313
```

Y ahora la comparamos con U (solución real):

>> N=abs(U-W(:,101))

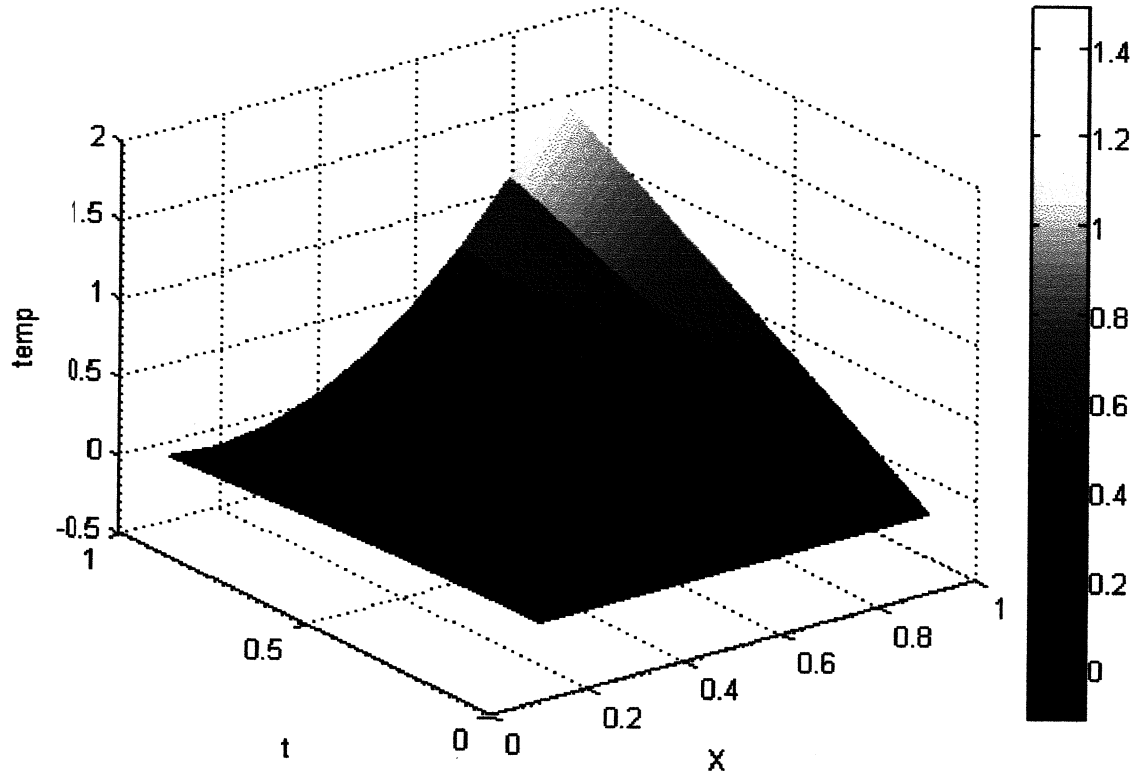
```
N =
   0.11980479683510
   0.20617704894934
   0.26421829575218
   0.29753295185021
   0.30826702970109
   0.29713300341198
   0.26341837918088
   0.20497713237804
   0.11820484839687
```

>> norm(N,inf)

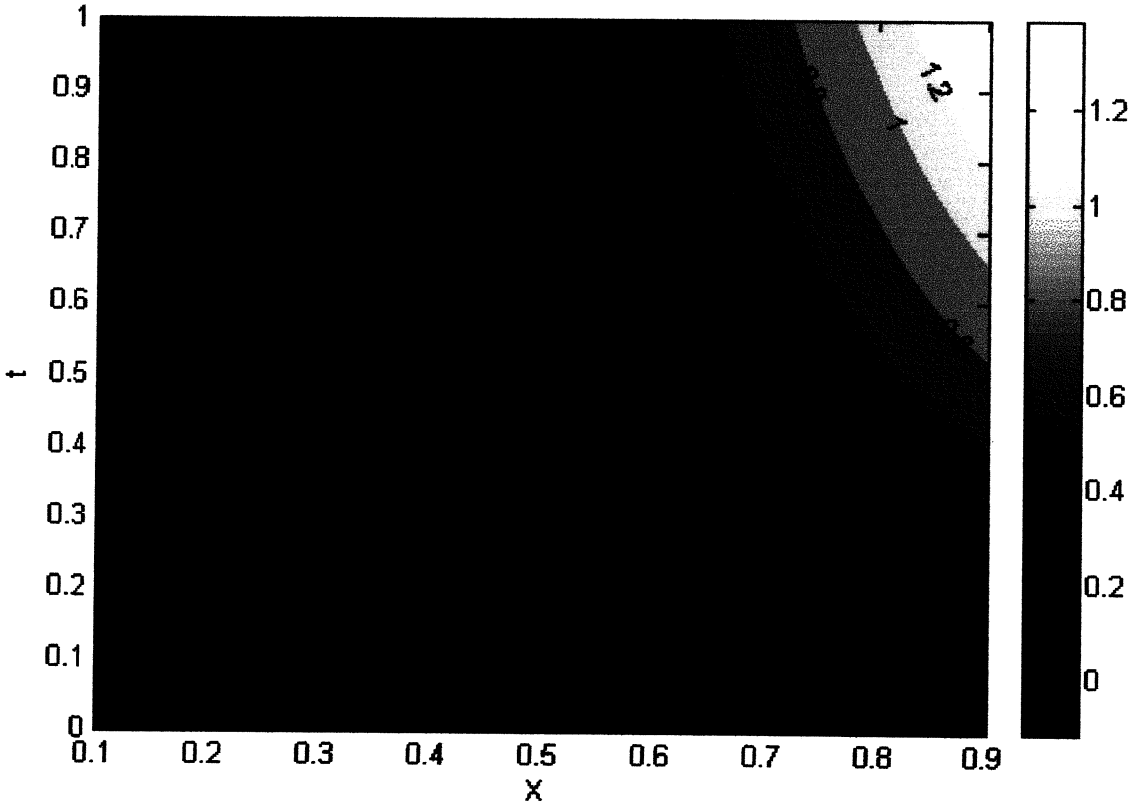
```
ans =
    0.30826702970109
```

Si recordamos, en el caso del método regresivo, este último dato era 0.30830171443292, cifra que es prácticamente la misma que la obtenida en el método Crank Nicolson. Los dos métodos han obtenido soluciones de igual precisión.

A continuación tenemos el gráfico que representa a la solución estimada con Crank-Nicolson:



Y, a continuación, las curvas de nivel:



• *Ejemplo 2.7*

Tenemos el siguiente planteamiento:

$$u_t = u_{xx}$$
$$0 < x < 1$$
$$0 < t$$

Y las siguientes condiciones de contorno:

$$u(0,t) = 0$$
$$u(1,t) = 0$$

Con la condición inicial:

$$u(x,0) = sen(\pi x)$$

Y sabemos que la solución en t=0,5 segundos es:

>> U=u(X,0,5)'

```
ans=
      0.00222241
      0.00422728
      0.00581836
      0.00683989
      0.00719188
      0.00683989
      0.00581836
      0.00422728
      0.00222241
```

Vamos a hacer una comparativa de los tres métodos para este ejemplo.

Llamamos al algoritmo correspondiente con los siguientes parámetros:

x0=0;n=10;L=1;k=0.01;m=50;α=1

Las funciones f, g, h1 y h2 son:

f.m

```
function z=f(x,t)
z=0;
```

g.m

```
function z=g(x)
z=sin(pi*x);
```

h1.m

```
function z=h1(t)
z=0;
```

h2.m

```
function z=h2(t)
z=0;
```

>> Wprog=metodoprogresivo(0,10,1,0.01,'f','g','h1','h2',50,1);

>> Wreg=metodoregresivo(0,10,1,0.01,'f','g','h1','h2',50,1);

>> Wcrank=metodocranknic(0,10,1,0.01,'f','g','h1','h2',50,1);

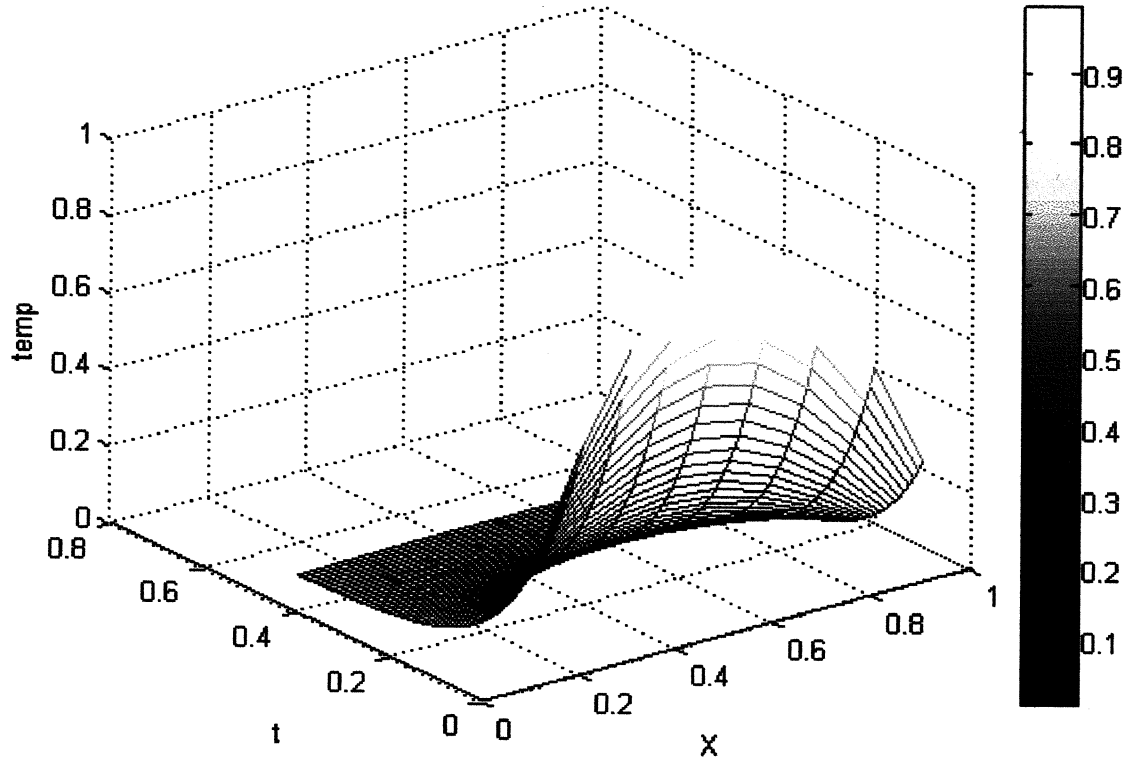
Tabla 1: Soluciones aproximadas y solución real. t=0,5 segundos.

X	Wprog(:,51)	Wreg(:,51)	Wcrank(:,51)	Solución real
0.1	0.34540324e+06	0.00289801	0.00230512	0.00222241
0.2	-0.6559307e+06	0.00551235	0.00438460	0.00422728
0.3	0.90052692e+06	0.00758710	0.00603489	0.00581836
0.4	-1.0552476e+06	0.00891917	0.00709444	0.00683989
0.5	1.10560546e+06	0.00937817	0.00745953	0.00719188
0.6	-1.0477369e+06	0.00891917	0.00709444	0.00683989
0.7	0.88837425e+06	0.00758710	0.00603489	0.00581836
0.8	-0.6437780e+06	0.00551235	0.00438460	0.00422728
0.9	0.33789247e+06	0.00289801	0.00230512	0.00222241

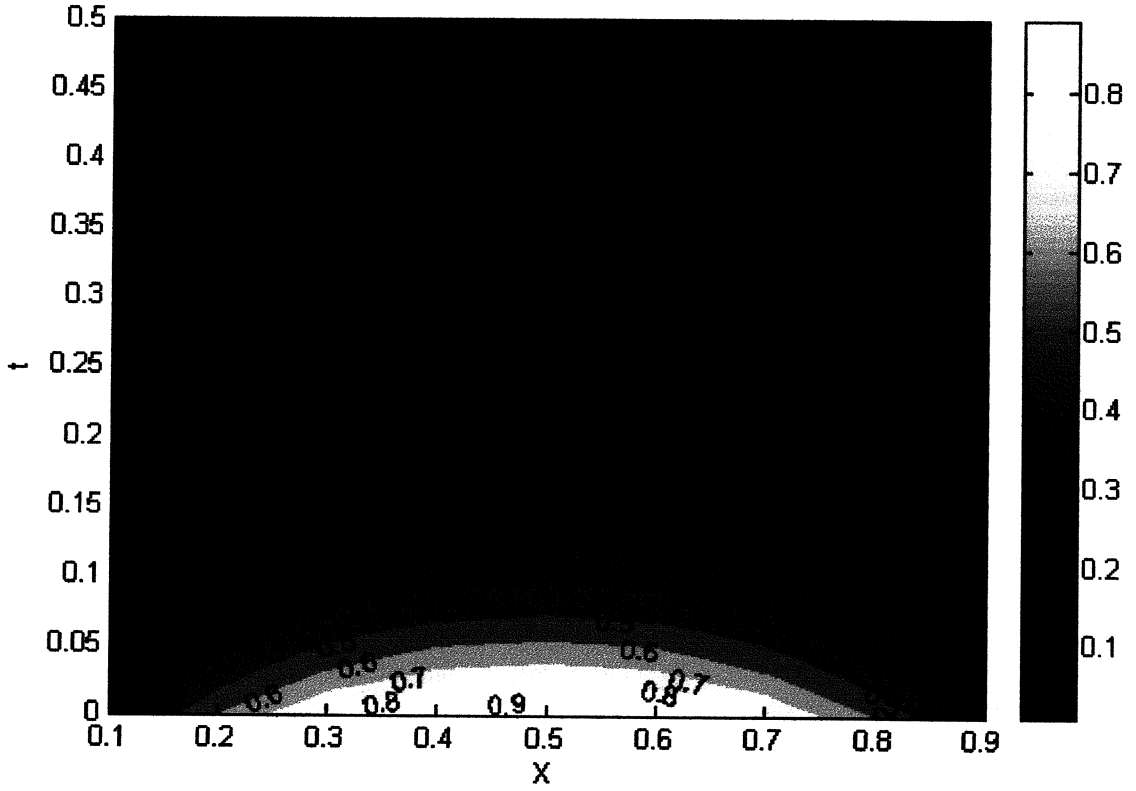
Tabla 2: Diferencias entre las soluciones aproximadas y solución real. t=0,5 segundos.

X	$\lvert Wprog(:,51)-U_{i,0'5}\rvert$	$\lvert Wreg(:,51)-U_{i,0'5}\rvert$	$\lvert Wcrank(:,51)-U_{i,0'5}\rvert$
0.1	0.34540323e+06	0.00067560	0.00008271
0.2	0.65593076e+06	0.00128507	0.00015732
0.3	0.90052692e+06	0.00176874	0.00021653
0.4	1.05524767e+06	0.00207928	0.00025455
0.5	1.10560545e+06	0.00218629	0.00026765
0.6	1.04773691e+06	0.00207928	0.00025455
0.7	0.88837425e+06	0.00176874	0.00021653
0.8	0.64377809e+06	0.00128507	0.00015732
0.9	0.33789247e+06	0.00067560	0.00008271

Se observa claramente que la solución aportada por el método progresivo es muy mala, mientras que la del regresivo y la del Crank-Nicolson son buenas, siendo la de Crank-Nicolson la mejor. A continuación tenemos el gráfico de la solución estimada por Crank-Nicolson:



Y a continuación tenemos las isotermas:



2.6 Variación de la ecuación principal: no se conoce la temperatura en el contorno.

Seguimos con el mismo planteamiento de al ecuación del calor, sólo que ahora no conocemos la temperatura de la barra en los extremos, sino conocemos el flujo de temperatura que pasa por ellos. El planteamiento sería el siguiente:

$$\frac{\partial u}{\partial t}(x,t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x,t) + f(x,t)\,,\ t > 0\,,\ x_0 < x < L$$

Lo que podemos expresar también de la siguiente manera:

$$U_t = \alpha^2 U_{xx} + f(x,t)$$

Siendo las siguientes, respectivamente, las condiciones de contorno e inicial:

$$\frac{\partial u}{\partial x}(x_0,t) = h_1(t)$$
$$\frac{\partial u}{\partial x}(x_n,t) = h_2(t)$$

$$u(x,0) = g(x)$$

Nuestro objetivo ahora es adaptar los métodos progresivo, regresivo y Crank-Nicolson a este nuevo planteamiento. En realidad la adaptación es sencilla, basta con discretizar las condiciones de contorno. Para discretizar la primera condición de contorno usaremos la primera forma de discretización de la primera derivada (ver apéndice). Tenemos entonces que, tomando $t=t_j$:

$$\frac{u(x_0+h,t_j)-u(x_0,t_j)}{h} = h_1(t_j)$$

Es decir,

$$u(x_0+h,t_j)-u(x_0,t_j) = h\cdot h_1(t_j)$$

Si llamamos $u(x_i,t_j)\approx W_{i,j}$ nos queda que:

$$W_{1,j} - W_{0,j} = h\cdot h_1(t_j)$$

Para discretizar la segunda condición de contorno, usaremos la tercera forma de discretización de la primera derivada (ver apéndice). Tenemos entonces que, tomando $t=t_j$:

$$\frac{u(x_n-h,t_j)-u(x_n,t_j)}{-h} = h_2(t_j)$$

Es decir,

$$u(x_n-h,t_j)-u(x_n,t_j) = -h\cdot h_2(t_j)$$

Si llamamos $u(x_i,t_j)\approx W_{i,j}$ nos queda que:

$$W_{n-1,j} - W_{n,j} = -h\cdot h_2(t_j)$$

2.6.1 Algoritmo A2.4: Ecuación del calor sin conocer la temperatura en el contorno, sino el flujo. Resolución mediante método progresivo.

Ahora, la expresión matricial del método progresivo quedaría así:

$$\underbrace{\begin{pmatrix} W_{0,j+1} \\ W_{1,j+1} \\ W_{2,j+1} \\ W_{3,j+1} \\ W_{4,j+1} \\ W_{5,j+1} \\ \vdots \\ W_{n,j+1} \end{pmatrix}}_{W_{j+1}} = \underbrace{\begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \lambda & 1-2\lambda & \lambda & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \lambda & 1-2\lambda & \lambda & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \lambda & 1-2\lambda & \lambda & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda & 1-2\lambda & \lambda & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \lambda & 1-2\lambda & \lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & -1 \end{pmatrix}}_{\tilde{A}} \underbrace{\begin{pmatrix} W_{0,j} \\ W_{1,j} \\ W_{2,j} \\ W_{3,j} \\ W_{4,j} \\ W_{5,j} \\ \vdots \\ W_{n,j} \end{pmatrix}}_{W_j} + \underbrace{\begin{pmatrix} h \cdot h_1(t_j) \\ kf_{1,j} \\ kf_{2,j} \\ kf_{3,j} \\ kf_{4,j} \\ \vdots \\ kf_{n-1,j} \\ -h \cdot h_2(t_j) \end{pmatrix}}_{\tilde{b}}$$

Siendo ahora el siguiente el método progresivo:

$$\begin{cases} W_{j+1} = \tilde{A}W_j + \tilde{b} \\ W_0 = \{g(x_i)\}_{i=1\dots n-1} \end{cases}$$

A continuación tenemos el algoritmo en Matlab que implementa este método.

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- f: función f(x,t)
- g: función g(x)
- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y
- alpha: valor de α

Datos que devuelve el algoritmo:

- W0: vector que contiene los valores de la temperatura inicial de la barra.
- W: matriz de n+1 filas y m columnas. El valor (i,j) de dicha matriz representa el valor de x_i en el instante (j*k).

```
function [W0,W] = metodoprogesivo(x0,n,L,k,f,g,h1,h2,m,alpha)
h=(L-x0)/n;
lambda=alpha^2*k/h^2;
W=zeros(n+1,m);
W0=zeros(n-1,1);
for i=1:n-1
    x=x0+i*h;
```

```
    W0(i)=feval(g,x);
end
A=zeros(n+1);
A(1,1)=-1;
A(1,2)=1;
A(n+1,n)=1;
A(n+1,n+1)=-1;
for i=2:n
    A(i,i)=1-2*lambda;
    A(i,i-1)=lambda;
    A(i,i+1)=lambda;
end
for j=1:m
    WJ=zeros(n+1,1);
    tj=j*k;
    b=zeros(n+1,1);
    b(1)=h*feval(h1,tj);
    b(n+1)=-h*feval(h2,tj);
    for i=2:n
        xi_1=x0+(i-1)*h;
        if j==1
            b(i)=k*feval(f,xi_1,tj)+W0(i-1);
        else
            b(i)=k*feval(f,xi_1,tj)+W(i,j-1);
        end
    end
    if j==1
        Wcero=zeros(n+1,1);
        for i=2:n
            Wcero(i)=W0(i-1);
        end
        for i=1:n+1
            WJ(i)=A(i,:)*Wcero+b(i);
        end
    else
        for i=1:n+1
            WJ(i)=A(i,:)*W(:,j-1)+b(i);
        end
    end
    W(:,j)=WJ;
end
```

```
    W0(i)=feval(g,x);
end
A=zeros(n+1);
A(1,1)=-1;
A(1,2)=1;
A(n+1,n)=1;
A(n+1,n+1)=-1;
for i=2:n
    A(i,i)=1-2*lambda;
    A(i,i-1)=lambda;
    A(i,i+1)=lambda;
end
for j=1:m
    WJ=zeros(n+1,1);
    tj=j*k;
    b=zeros(n+1,1);
    b(1)=h*feval(h1,tj);
    b(n+1)=-h*feval(h2,tj);
    for i=2:n
        xi_1=x0+(i-1)*h;
        if j==1
            b(i)=k*feval(f,xi_1,tj)+W0(i-1);
        else
            b(i)=k*feval(f,xi_1,tj)+W(i,j-1);
        end
    end
    if j==1
        Wcero=zeros(n+1,1);
        for i=2:n
            Wcero(i)=W0(i-1);
        end
        for i=1:n+1
            WJ(i)=A(i,:)*Wcero+b(i);
        end
    else
        for i=1:n+1
            WJ(i)=A(i,:)*W(:,j-1)+b(i);
        end
    end
    W(:,j)=WJ;
end
```

2.6.2 Algoritmo A2.5: Ecuación del calor sin conocer la temperatura en el contorno, sino el flujo. Resolución mediante método regresivo.

Ahora, la expresión matricial del método regresivo quedaría así:

$$\underbrace{\begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & -\lambda & 1+2\lambda & -\lambda & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda & 1+2\lambda & -\lambda & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & -\lambda & 1+2\lambda & -\lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & -1 \end{pmatrix}}_{\tilde{A}} \underbrace{\begin{pmatrix} W_{0,j} \\ W_{1,j} \\ W_{2,j} \\ W_{3,j} \\ W_{4,j} \\ W_{5,j} \\ \vdots \\ W_{n,j} \end{pmatrix}}_{W_j} = \underbrace{\begin{pmatrix} h \cdot h_1(t_j) \\ kf_{1,j} + W_{1,j-1} \\ kf_{2,j} + W_{2,j-1} \\ kf_{3,j} + W_{3,j-1} \\ \vdots \\ kf_{n-2,j} + W_{n-2,j-1} \\ kf_{n-1,j} + W_{n-1,j-1} \\ -h \cdot h_2(t_j) \end{pmatrix}}_{\widetilde{[W_{j-1}+b]}}$$

Por tanto el método queda expresado así:

$$\begin{cases} \tilde{A}W_j = \widetilde{[W_{j-1}+b]} \\ W_0 = \{g(x_i)\}_{i=1\dots n-1} \end{cases}$$

A continuación tenemos el algoritmo en Matlab que implementa este método.

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- f: función f(x,t)
- g: función g(x)
- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y
- alpha: valor de α

Datos que devuelve el algoritmo:

- W0: vector que contiene los valores de la temperatura inicial de la barra.
- W: matriz de n+1 filas y m columnas. El valor (i,j) de dicha matriz representa el valor de x_i en el instante (j*k).

```
function [W0,W] = metodoregresivo(x0,n,L,k,f,g,h1,h2,m,alpha)
h=(L-x0)/n;
lambda=alpha^2*k/h^2;
W=zeros(n+1,m);
W0=zeros(n-1,1);
```

$$\underbrace{\begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ -\lambda/2 & 1+\lambda & -\lambda/2 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & -\lambda/2 & 1+\lambda & -\lambda/2 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & -\lambda/2 & 1+\lambda & -\lambda/2 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda/2 & 1+\lambda & -\lambda/2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & -\lambda/2 & 1+\lambda & -\lambda/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & -1 \end{pmatrix}}_{\tilde{A}} \underbrace{\begin{pmatrix} W_{0,j+1} \\ W_{1,j+1} \\ W_{2,j+1} \\ W_{3,j+1} \\ W_{4,j+1} \\ W_{5,j+1} \\ \vdots \\ W_{n,j+1} \end{pmatrix}}_{W_{j+1}} =$$

$$= \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \lambda/2 & 1-\lambda & \lambda/2 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \lambda/2 & 1-\lambda & \lambda/2 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \lambda/2 & 1-\lambda & \lambda/2 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda/2 & 1-\lambda & \lambda/2 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \lambda/2 & 1-\lambda & \lambda/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{pmatrix}}_{\tilde{B}} \underbrace{\begin{pmatrix} W_{0,j} \\ W_{1,j} \\ W_{2,j} \\ W_{3,j} \\ W_{4,j} \\ W_{5,j} \\ \vdots \\ W_{n,j} \end{pmatrix}}_{W_j} +$$

$$+ \underbrace{\begin{pmatrix} h \cdot h_1(t_{j+1}) \\ k\dfrac{f_{1,j}+f_{1,j+1}}{2} \\ k\dfrac{f_{2,j}+f_{2,j+1}}{2} \\ k\dfrac{f_{3,j}+f_{3,j+1}}{2} \\ \vdots \\ k\dfrac{f_{n-2,j}+f_{n-2,j+1}}{2} \\ k\dfrac{f_{n-1,j}+f_{n-1,j+1}}{2} \\ -h \cdot h_2(t_{j+1}) \end{pmatrix}}_{\tilde{b}}$$

Por tanto el método queda expresado así:

$$\begin{cases} \tilde{A}W_{j+1} = \tilde{B}W_j + \tilde{b} \\ W_0 = \{g(x_i)\}_{i=1\dots n-1} \end{cases}$$

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- f: función f(x,t)
- g: función g(x)

- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y
- alpha: valor de α

Datos que devuelve el algoritmo:

- W0: vector de valores iniciales
- W: matriz de n-1 filas y m+1 columnas. El valor (i,j) de dicha matriz representa el valor de x_i en el instante (j*k).

```
function [W0,W] = metodocranknic(x0,n,L,k,f,g,h1,h2,m,alpha)
h=(L-x0)/n;
lambda=alpha^2*k/h^2;
W=zeros(n+1,m);
W0=zeros(n-1,1);
for i=1:n-1
    x=x0+i*h;
    W0(i)=feval(g,x);
end
A=zeros(n+1);
A(1,1)=-1;
A(1,2)=1;
A(n+1,n)=1;
A(n+1,n+1)=-1;
for i=2:n
    A(i,i)=1+lambda;
    A(i,i-1)=-1*lambda/2;
    A(i,i+1)=-1*lambda/2;
end
B=zeros(n+1);
for i=2:n
    B(i,i)=1-lambda;
    B(i,i-1)=lambda/2;
    B(i,i+1)=lambda/2;
end
for j=1:m
    WJ=zeros(n+1,1);
    tj=j*k;
    tj1=(j+1)*k;
    b=zeros(n+1,1);
    b(1)=h*feval(h1,tj1);
    b(n+1)=-h*feval(h2,tj1);
```

```
    for i=2:n
        xi=x0+(i-1)*h;
        b(i)=k*(feval(f,xi,tj)+feval(f,xi,tj1))/2;
    end
    C=zeros(n-1,1);
    if j==1
        Wcero=zeros(n+1,1);
        for i=2:n
            Wcero(i)=W0(i-1);
        end
        for i=1:n+1
            C(i)=B(i,:)*Wcero+b(i);
        end
    else
        for i=1:n+1
            C(i)=B(i,:)*W(:,j-1)+b(i);
        end
    end
    WJ=A\C;
    W(:,j)=WJ;
end
```

2.7 Ejemplos

- ***Ejemplo 2.8***

Tenemos el siguiente planteamiento:

$$u_t = u_{xx} + e^x(-\cos(t) - sen(t))$$

Y las siguientes condiciones de contorno:

$$u_x(0,t) = \cos(t)$$
$$u_x(1,t) = e \cdot \cos(t)$$

Con la condición inicial:

$$u(x,0) = e^x$$

La solución al problema es:

$$u(x,t) = e^x \cdot \cos(t)$$

A continuación vamos a intentar aproximar la solución mediante los tres métodos que hemos visto.

MEDIANTE EL MÉTODO PROGRESIVO

Llamamos al algoritmo A2.1 con los siguientes parámetros:

$$x0=0;n=10;L=1;k=0.001;m=500;\alpha=1$$

Las funciones f, g, h1 y h2 son:

f.m

```
function z=f(x,t)
z=exp(x)*(-cos(t)-sin(t));
```

g.m

```
function z=g(x)
z=exp(x);
```

h1.m

```
function z=h1(t)
z=cos(t);
```

h2.m

```
function z=h2(t)
z=exp(1)*cos(t);
```

>> [W0,W]=metodoprogresivo(0,10,1,0.001,'f','g','h1','h2',500,1)

Del resultado nos quedamos con el valor de la última columna, que equivale al valor de la temperatura en el instante t=m*k=500*0.001=0,5 segundos.

```
W(:,500) =
        1.0e+149 *
         1.21783598100892
         3.64360647150836
         5.77313089785103
         7.43305828850592
         8.48795976441173
         8.85139768783250
         8.49311108484838
         7.44168681027561
         5.78243245194339
         3.65055816125215
         1.22017858825862
```

Cada valor corresponde a un x_i.

Como tenemos la solución del problema, vamos a calcularla y a compararla con la solución estimada mediante el método progresivo. Recordemos que la solución era:

$$u(x,t) = e^x \cdot \cos(t)$$

Lo que en Matlab podemos escribir como:

```
function z=u(x,t)
z=exp(x)*cos(t);
```

Aunque nosotros la escribiremos de la siguiente forma:

u.m

```
function Z=u(X,t)
for i=1:length(X)
   Z(i)=exp(X(i))*cos(t);
end
```

Así está preparada para trabajar con un vector. Por tanto, pedimos a Matlab que calcule lo siguiente:

>> X=[0:0.1:1];

>> U=u(X,0.5)'

```
U =
   0.87758256189037
   0.96987872561156
   1.07188176160617
   1.18461255054283
   1.30919934004736
   1.44688903658417
   1.59905968491532
   1.76723426003562
   1.95309590961830
   2.15850479952265
   2.38551673095914
```

Ahora vamos a comparer el resultado obtenido con la solución real:

>> N=abs(U-W(:,500))

```
N =
  1.0e+149 *
   1.21783598100892
   3.64360647150836
   5.77313089785103
   7.43305828850592
   8.48795976441173
   8.85139768783250
   8.49311108484838
```

```
   7.44168681027561
   5.78243245194339
   3.65055816125215
   1.22017858825862
```

>> norm(N,inf)

```
ans =
    8.851397687832497e+149
```

Puede observarse, por los resultados obtenidos, que la aproximación es muy mala, el método no converge. A continuación utilizaremos los métodos regresivo y Crank-Nicolson para obtener una aproximación mejor.

MEDIANTE EL MÉTODO REGRESIVO

Llamamos al algoritmo A2.2 con los mismos datos que en el caso anterior.

>> [W0,W]=metodoregresivo(0,10,1,0.001,'f','g','h1','h2',500,1)

Al igual que antes, nos quedaremos con la última columna, correspondiente al instante t=0,5 segundos:

```
W(:,500) =
         0.97392205201809
         1.06168030820712
         1.16095097394632
         1.27277072056067
         1.39827512624403
         1.53870994011100
         1.69544330995761
         1.86997911778576
         2.06397158376680
         2.27924131475214
         2.51779298784805
```

Si la comparamos con la solución real:

>> X=[0:0.1:1];

>> U=u(X,0.5)'

```
U =
   0.87758256189037
   0.96987872561156
   1.07188176160617
   1.18461255054283
```

```
   1.30919934004736
   1.44688903658417
   1.59905968491532
   1.76723426003562
   1.95309590961830
   2.15850479952265
   2.38551673095914
```

Ahora vamos a comparar el resultado obtenido con la solución real:

>> N=abs(U-W(:,500))

```
N =
   0.09633949012771
   0.09180158259556
   0.08906921234015
   0.08815817001784
   0.08907578619667
   0.09182090352683
   0.09638362504229
   0.10274485775015
   0.11087567414850
   0.12073651522949
   0.13227625688892
```

>> norm(N,inf)

```
ans =
  0.13227625688892
```

Los datos obtenidos nos demuestran que la aproximación es mejor que la obtenida por el método progresivo y, en general, es aceptable.

MEDIANTE EL MÉTODO DE CRANK-NICOLSON

Llamamos al algoritmo A2.2 con los mismos datos que en el caso anterior.

>> [W0,W]=metodocranknic(0,10,1,0.001,'f','g','h1','h2',500,1)

Al igual que antes, nos quedaremos con la última columna, correspondiente al instante t=0,5 segundos:

```
W(:,500) =
           0.95310073488481
         1.04081100464885
         1.14002493619636
```

```
        1.25177879873225
        1.37720817910964
        1.51755909210932
        1.67420000336526
        1.84863493242973
        2.04251782331177
        2.25766838525695
        2.49608961772578
```

Si la comparamos con la solución real:

>> X=[0:0.1:1];

>> U=u(X,0.5)'

```
U =
    0.87758256189037
    0.96987872561156
    1.07188176160617
    1.18461255054283
    1.30919934004736
    1.44688903658417
    1.59905968491532
    1.76723426003562
    1.95309590961830
    2.15850479952265
    2.38551673095914
```

Ahora vamos a comparar el resultado obtenido con la solución real:

>> N=abs(U-W(:,500))

```
N =
    0.07551817299444
    0.07093227903729
    0.06814317459019
    0.06716624818942
    0.06800883906228
    0.07067005552515
    0.07514031844995
    0.08140067239411
    0.08942191369347
    0.09916358573430
    0.11057288676665
```

• *Ejemplo 2.9*

Tenemos el siguiente planteamiento:

$$u_t = \frac{1}{2}u_{xx} + 2x^2 - 2t$$
$$0 < x < 1$$
$$0 < t$$

Y las siguientes condiciones de contorno:

$$u_x(0,t) = 0$$
$$u_x(1,t) = 4t$$

Con la condición inicial:

$$u(x,0) = 0$$

La solución es:

$$u(x,t) = 2x^2 \cdot t$$

Vamos a aproximar la solución mediante el método regresivo.

Llamamos al algoritmo A2.2 con los siguientes parámetros:

x0=0;n=10;L=1;k=0.01;m=100;α=0.5

Las funciones f, g, h1 y h2 son:

f.m

```
function z=f(x,t)
z=2*x^2-2*t;
```

g.m

```
function z=g(x)
z=0;
```

h1.m

```
function z=h1(t)
z=0;
```

h2.m

```
function z=h2(t)
z=4t;
```

>> [W0,W]=metodoregresivo(0,10,1,0.01,'f','g','h1','h2',100,0.5)

El valor de los x para t=1 segundo es:

```
   0.40966911330679
   0.42966911330679
   0.44439504975870
   0.45446913995950
   0.46033582080778
   0.46226175233446
   0.46033582080778
   0.45446913995950
   0.44439504975870
   0.42966911330680
   0.40966911330680
```

>> norm(N,inf)

```
ans =
   0.46226175233446
```

La aproximación es aceptable pero no demasiado buena. Veámos qué nos dice el método Crank-Nicolson.

- ***Ejemplo 2.10***

Vamos a resolver el ejemplo anterior mediante Crank-Nicolson.

>> [W0,W]=metodocranknic(0,10,1,0.01,'f','g','h1','h2',100,0.5)

El valor de los x para t=1 segundo es:

```
W(:,100)=
  -0.41973935934514
  -0.41973935934514
  -0.37406374279332
  -0.28332955776493
  -0.14797152427216
   0.03175679315803
   0.25578140546461
   0.52420610202870
   0.83731065333898
   1.19554923018520
   1.59954923018520
```

Ahora comparamos la solución obtenida con la real:

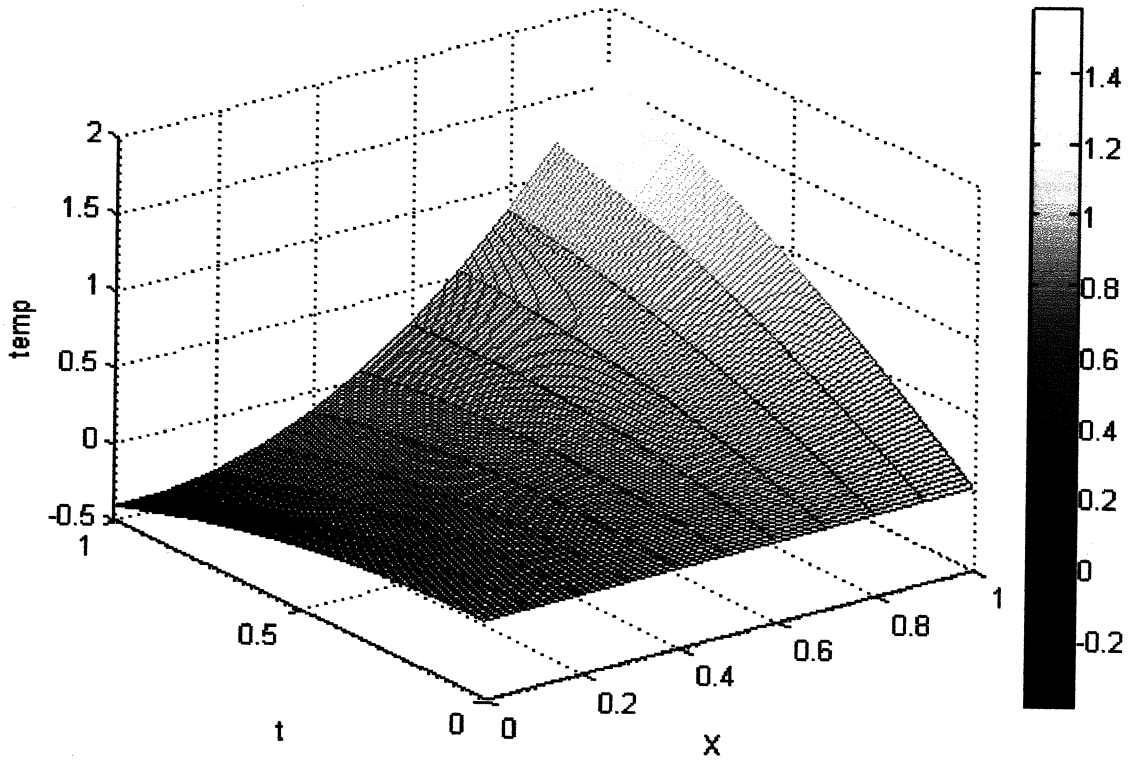
>> N=abs(U-W(:,100))

```
N =
   0.41973935934514
   0.43973935934514
   0.45406374279332
   0.46332955776493
   0.46797152427216
   0.46824320684197
   0.46421859453539
   0.45579389797130
   0.44268934666102
   0.42445076981480
   0.40045076981480
```

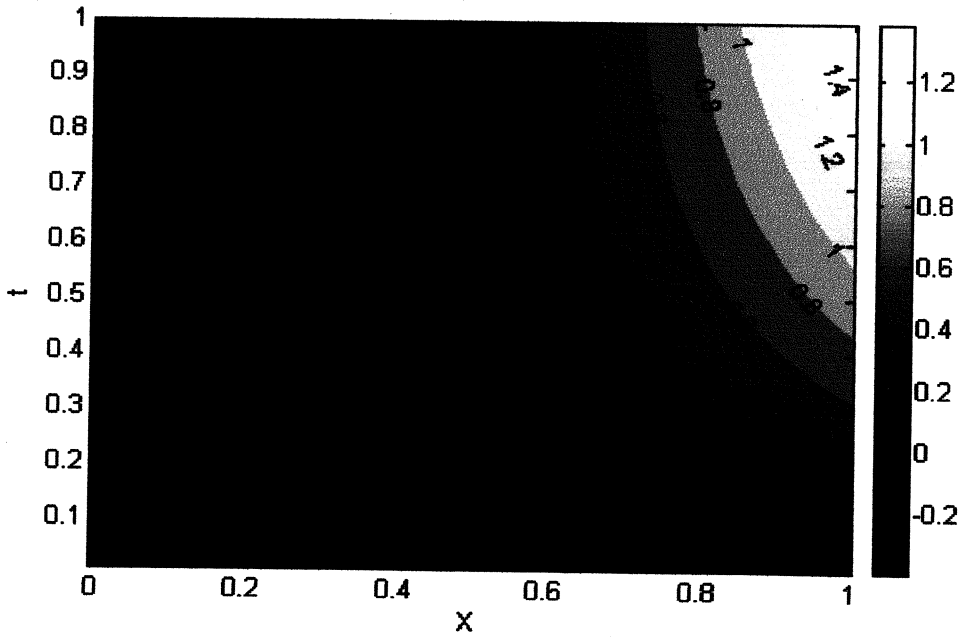
>> norm(N,inf)

```
ans =
   0.46824320684197
```

La aproximación es aceptable pero no demasiado buena. En este caso la solución aportada por el método regresivo era algo mejor. Veámos el gráfico correspondiente a la solución del méodo regresivo:



Y ahora veámos las curvas de nivel:



2.8 Variación de la ecuación principal: ecuación del calor para trabajar con fluidos

Ahora continuamos con el planteamiento anterior: tenemos la ecuación del calor y no conocemos la temperatura de la barra en sus extremos, sino el flujo de temperatura que pasa a través de ellos. En esta ocasión vamos a añadirle a la ecuación del calor un nuevo término que modela cuando se trabaje con fluidos. Dicho término es:

$$\varepsilon \frac{\partial u}{\partial x}(t,x)$$

2.8.1 Planteamiento del problema. Ecuación.

La ecuación que tenemos ahora es:

$$\frac{\partial u}{\partial t}(x,t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x,t) + f(x,t) + \varepsilon \frac{\partial u}{\partial x}(t,x),\ t > 0,\ x_0 < x < L$$

Lo que podemos expresar también de la siguiente manera:

$$U_t = \alpha^2 U_{xx} + f + \varepsilon U_x$$

Siendo las siguientes, respectivamente, las condiciones de contorno e inicial:

$$\frac{\partial u}{\partial x}(x_0,t) = h_1(t)$$
$$\frac{\partial u}{\partial x}(x_n,t) = h_2(t)$$

$$u(x,0) = g(x)$$

2.8.2 Discretización.

La ecuación del calor ya la tenemos discretizada, lo único que tenemos que discretizar aquí es el nuevo término. Para discretizarlo usaremos la primera forma de discretización de la primera derivada (ver apéndice).

El término discretizado quedaría así:

$$\varepsilon \frac{u(x_i + h, t_j) - u(x_i, t_j)}{h}$$

2.8.3 Método progresivo.

Si a la discretización del método progresivo,

$$\frac{u(x_i,t_j + k) - u(x_i,t_j)}{k} = \alpha^2 \frac{u(x_i + h,t_j) - 2u(x_i,t_j) + u(x_i - h,t_j)}{h^2} + f(x_i,t_j)$$

le añadimos el nuevo término,

$$\frac{u(x_i,t_j + k) - u(x_i,t_j)}{k} = \alpha^2 \frac{u(x_i + h,t_j) - 2u(x_i,t_j) + u(x_i - h,t_j)}{h^2} + f(x_i,t_j) + \varepsilon \frac{u(x_i + h,t_j) - u(x_i,t_j)}{h}$$

la ecuación del método progresivo quedaría ahora así:

$$W_{i,j+1} = (1-2\lambda-\beta)W_{i,j} + (\lambda+\beta)W_{i+1,j} + \lambda W_{i-1,j} + kf_{i,j}$$

donde $\beta = \frac{k\varepsilon}{h}$.

Algoritmo A2.7: Método progresivo.

Podemos expresar matricialmente este método de la siguiente manera:

$$\underbrace{\begin{pmatrix} W_{0,j+1} \\ W_{1,j+1} \\ W_{2,j+1} \\ W_{3,j+1} \\ W_{4,j+1} \\ W_{5,j+1} \\ \vdots \\ W_{n,j+1} \end{pmatrix}}_{W_{j+1}} = \underbrace{\begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \lambda & 1-2\lambda-\beta & \lambda+\beta & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \lambda & 1-2\lambda-\beta & \lambda+\beta & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \lambda & 1-2\lambda-\beta & \lambda+\beta & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda & 1-2\lambda-\beta & \lambda+\beta & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \lambda & 1-2\lambda-\beta & \lambda+\beta \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & -1 \end{pmatrix}}_{\tilde{A}} \cdot$$

$$\cdot \underbrace{\begin{pmatrix} W_{0,j} \\ W_{1,j} \\ W_{2,j} \\ W_{3,j} \\ W_{4,j} \\ W_{5,j} \\ \vdots \\ W_{n,j} \end{pmatrix}}_{W_j} + \underbrace{\begin{pmatrix} h \cdot h_1(t_j) \\ kf_{1,j} \\ kf_{2,j} \\ kf_{3,j} \\ kf_{4,j} \\ \vdots \\ kf_{n-1,j} \\ -h \cdot h_2(t_j) \end{pmatrix}}_{\tilde{b}}$$

Siendo ahora el siguiente el método progresivo:

$$\begin{cases} W_{j+1} = \tilde{A}W_j + \tilde{b} \\ W_0 = \{g(x_i)\}_{i=1\dots n-1} \end{cases}$$

A continuación tenemos el algoritmo en Matlab que implementa este método.

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- f: función f(x,t)
- g: función g(x)
- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y

- alpha: valor de α
- E: valor de ε

Datos que devuelve el algoritmo:

- W0: vector que contiene los valores de la temperatura inicial de la barra.
- W: matriz de n+1 filas y m columnas. El valor (i,j) de dicha matriz representa el valor de x_i en el instante j*k.

```
function [W0,W] = metodoprogesivo(x0,n,L,k,f,g,h1,h2,m,alpha,E)
h=(L-x0)/n;
lambda=alpha^2*k/h^2;
beta=k*E/h;

W=zeros(n+1,m);
W0=zeros(n-1,1);
for i=1:n-1
    x=x0+i*h;
    W0(i)=feval(g,x);
end
A=zeros(n+1);
A(1,1)=-1;
A(1,2)=1;
A(n+1,n)=1;
A(n+1,n+1)=-1;
for i=2:n
    A(i,i)=1-2*lambda-beta;
    A(i,i-1)=lambda;
    A(i,i+1)=lambda+beta;
end
for j=1:m
    WJ=zeros(n+1,1);
    tj=j*k;
    b=zeros(n+1,1);
    b(1)=h*feval(h1,tj);
    b(n+1)=-h*feval(h2,tj);
    for i=2:n
        xi_1=x0+(i-1)*h;
        if j==1
            b(i)=k*feval(f,xi_1,tj)+W0(i-1);
        else
            b(i)=k*feval(f,xi_1,tj)+W(i,j-1);
        end
```

```
    end
    if j==1
        Wcero=zeros(n+1,1);
        for i=2:n
            Wcero(i)=W0(i-1);
        end
        for i=1:n+1
            WJ(i)=A(i,:)*Wcero+b(i);
        end
    else
        for i=1:n+1
            WJ(i)=A(i,:)*W(:,j-1)+b(i);
        end
    end
    W(:,j)=WJ;
end
```

2.8.4 Método regresivo.

Si a la discretización del método regresivo,

$$\frac{u(x_i,t_j-k)-u(x_i,t_j)}{-k}=\alpha^2\frac{u(x_i+h,t_j)-2u(x_i,t_j)+u(x_i-h,t_j)}{h^2}+f(x_i,t_j)$$

le añadimos el nuevo término,

$$\frac{u(x_i,t_j-k)-u(x_i,t_j)}{-k}=\alpha^2\frac{u(x_i+h,t_j)-2u(x_i,t_j)+u(x_i-h,t_j)}{h^2}+f(x_i,t_j)+\varepsilon\frac{u(x_i+h,t_j)-u(x_i,t_j)}{h}$$

la ecuación del método regresivo quedaría ahora así:

$$(1+2\lambda+\beta)W_{i,j}-(\lambda+\beta)W_{i+1,j}-\lambda W_{i-1,j}=kf_{i,j}+W_{i,j-1}$$

donde $\beta=\dfrac{k\varepsilon}{h}$.

Algoritmo A2.8: Método regresivo.

Podemos expresar matricialmente este método de la siguiente manera:

$$\underbrace{\begin{pmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0\\
-\lambda & 1+2\lambda+\beta & -(\lambda+\beta) & 0 & 0 & 0 & \dots & 0 & 0 & 0\\
0 & -\lambda & 1+2\lambda+\beta & -(\lambda+\beta) & 0 & 0 & \dots & 0 & 0 & 0\\
0 & 0 & -\lambda & 1+2\lambda+\beta & -(\lambda+\beta) & 0 & \dots & 0 & 0 & 0\\
0 & 0 & 0 & -\lambda & 1+2\lambda+\beta & -(\lambda+\beta) & \dots & 0 & 0 & 0\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots\\
0 & 0 & 0 & 0 & 0 & 0 & \dots & -\lambda & 1+2\lambda+\beta & -(\lambda+\beta)\\
0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & -1
\end{pmatrix}}_{\tilde{A}}.$$

$$\underbrace{\begin{pmatrix} W_{0,j} \\ W_{1,j} \\ W_{2,j} \\ W_{3,j} \\ W_{4,j} \\ W_{5,j} \\ \vdots \\ W_{n,j} \end{pmatrix}}_{W_j} = \underbrace{\begin{pmatrix} h \cdot h_1(t_j) \\ kf_{1,j} + W_{1,j-1} \\ kf_{2,j} + W_{2,j-1} \\ kf_{3,j} + W_{3,j-1} \\ \vdots \\ kf_{n-2,j} + W_{n-2,j-1} \\ kf_{n-1,j} + W_{n-1,j-1} \\ -h \cdot h_2(t_j) \end{pmatrix}}_{\left[\widetilde{W_{j-1}+b}\right]}$$

Por tanto el método queda expresado así:

$$\begin{cases} \tilde{A}W_j = \left[\widetilde{W_{j-1}+b}\right] \\ W_0 = \{g(x_i)\}_{i=1\ldots n-1} \end{cases}$$

A continuación tenemos el algoritmo en Matlab que implementa este método.

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- f: función f(x,t)
- g: función g(x)
- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y
- alpha: valor de α
- E: valor de ε

Datos que devuelve el algoritmo:

- W0: vector que contiene los valores de la temperatura inicial de la barra.
- W: matriz de n+1 filas y m columnas. El valor (i,j) de dicha matriz representa el valor de x_i en el instante j*k.

```
function [W0,W] = metodoregresivo(x0,n,L,k,f,g,h1,h2,m,alpha,E)
h=(L-x0)/n;
lambda=alpha^2*k/h^2;
beta=k*E/h;

W=zeros(n+1,m);
```

```
W0=zeros(n-1,1);
for i=1:n-1
    x=x0+i*h;
    W0(i)=feval(g,x);
end
A=zeros(n+1);
A(1,1)=-1;
A(1,2)=1;
A(n+1,n)=1;
A(n+1,n+1)=-1;
for i=2:n
    A(i,i)=1+2*lambda+beta;
    A(i,i-1)=-1*lambda;
    A(i,i+1)=-1*(lambda+beta);
end
for j=1:m
    WJ=zeros(n+1,1);
    tj=j*k;
    b=zeros(n+1,1);
    b(1)=h*feval(h1,tj);
    b(n+1)=-h*feval(h2,tj);
    for i=2:n
        xi_1=x0+(i-1)*h;
        if j==1
            b(i)=k*feval(f,xi_1,tj)+W0(i-1);
        else
            b(i)=k*feval(f,xi_1,tj)+W(i,j-1);
        end
    end
    WJ=A\b;
    W(:,j)=WJ;
end
```

2.8.5 Método Crank-Nicolson.

El método de Crank-Nicolson es un método de diferencia promediada entre el método progresivo en el paso j y el método regresivo en el paso j+1.

La ecuación del método progresivo en el paso j es:

$$W_{i,j+1} = (1-2\lambda-\beta)W_{i,j} + (\lambda+\beta)W_{i+1,j} + \lambda W_{i-1,j} + kf_{i,j}$$

Y la del método regresivo en el paso j+1:

$$(1+2\lambda+\beta)W_{i,j+1}-(\lambda+\beta)W_{i+1,j+1}-\lambda W_{i-1,j+1}=kf_{i,j+1}+W_{i,j}$$

donde $\beta=\dfrac{k\varepsilon}{h}$.

Por tanto, el método de Crank-Nicolson quedaría se la siguiente manera:

$$W_{i,j+1}-W_{i,j}-\frac{\lambda}{2}\left[W_{i+1,j}-2W_{i,j}+W_{i-1,j}+W_{i+1,j+1}-2W_{i,j+1}+W_{i-1,j+1}\right]+\frac{\beta}{2}\left[W_{i,j}+W_{i,j+1}-W_{i+1,j+1}-W_{i+1,j}\right]=$$
$$=k\frac{f_{i,j}+f_{i,j+1}}{2}$$

Agrupando, tenemos que:

$$(1+\lambda+\frac{\beta}{2})W_{i,j+1}-(\frac{\lambda}{2}+\frac{\beta}{2})W_{i+1,j+1}-\frac{\lambda}{2}W_{i-1,j+1}=(1-\lambda-\frac{\beta}{2})W_{i,j}+(\frac{\lambda}{2}+\frac{\beta}{2})W_{i+1,j}+\frac{\lambda}{2}W_{i-1,j}+k\frac{f_{i,j}+f_{i,j+1}}{2}$$

Algoritmo A2.9: Método Crank-Nicolson.

Podemos expresar matricialmente este método de la siguiente manera:

$$\underbrace{\begin{pmatrix}
-1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
-\lambda/2 & 1+\lambda+\frac{\beta}{2} & -(\frac{\lambda}{2}+\frac{\beta}{2}) & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & -\lambda/2 & 1+\lambda+\frac{\beta}{2} & -(\frac{\lambda}{2}+\frac{\beta}{2}) & 0 & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & -\lambda/2 & 1+\lambda+\frac{\beta}{2} & -(\frac{\lambda}{2}+\frac{\beta}{2}) & 0 & \dots & 0 & 0 & 0 \\
0 & 0 & 0 & -\lambda/2 & 1+\lambda+\frac{\beta}{2} & -(\frac{\lambda}{2}+\frac{\beta}{2}) & \dots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & \dots & -\lambda/2 & 1+\lambda+\frac{\beta}{2} & -(\frac{\lambda}{2}+\frac{\beta}{2}) \\
0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 & -1
\end{pmatrix}}_{A}.$$

$$\underbrace{\begin{pmatrix} W_{0,j+1} \\ W_{1,j+1} \\ W_{2,j+1} \\ W_{3,j+1} \\ W_{4,j+1} \\ W_{5,j+1} \\ \vdots \\ W_{n,j+1} \end{pmatrix}}_{W_{j+1}} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{\lambda}{2} & 1-\lambda-\frac{\beta}{2} & \frac{\lambda}{2}+\frac{\beta}{2} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{\lambda}{2} & 1-\lambda-\frac{\beta}{2} & \frac{\lambda}{2}+\frac{\beta}{2} & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & \frac{\lambda}{2} & 1-\lambda-\frac{\beta}{2} & \frac{\lambda}{2}+\frac{\beta}{2} & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\lambda}{2} & 1-\lambda-\frac{\beta}{2} & \frac{\lambda}{2}+\frac{\beta}{2} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & \frac{\lambda}{2} & 1-\lambda-\frac{\beta}{2} & \frac{\lambda}{2}+\frac{\beta}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{pmatrix}}_{\tilde{B}} \cdot \underbrace{\begin{pmatrix} W_{0,j} \\ W_{1,j} \\ W_{2,j} \\ W_{3,j} \\ W_{4,j} \\ W_{5,j} \\ \vdots \\ W_{n,j} \end{pmatrix}}_{W_j} + \underbrace{\begin{pmatrix} h \cdot h_1(t_{j+1}) \\ k\frac{f_{1,j}+f_{1,j+1}}{2} \\ k\frac{f_{2,j}+f_{2,j+1}}{2} \\ k\frac{f_{3,j}+f_{3,j+1}}{2} \\ \vdots \\ k\frac{f_{n-2,j}+f_{n-2,j+1}}{2} \\ k\frac{f_{n-1,j}+f_{n-1,j+1}}{2} \\ -h \cdot h_2(t_{j+1}) \end{pmatrix}}_{\tilde{b}}$$

Por tanto el método queda expresado así:

$$\begin{cases} \tilde{A}W_{j+1} = \tilde{B}W_j + \tilde{b} \\ W_0 = \{g(x_i)\}_{i=1\dots n-1} \end{cases}$$

A continuación tenemos el algoritmo en Matlab que implementa este método.

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- f: función f(x,t)
- g: función g(x)
- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y
- alpha: valor de α

- E: valor de ε

Datos que devuelve el algoritmo:

- W0: vector de valores iniciales
- W: matriz de n+1 filas y m columnas. El valor (i,j) de dicha matriz representa el valor de x_i en el instante j*k.

```
metodocranknic(x0,n,L,k,f,g,h1,h2,m,alpha,E)
h=(L-x0)/n;
lambda=alpha^2*k/h^2;
beta=k*E/h;

W=zeros(n+1,m);
W0=zeros(n-1,1);
for i=1:n-1
    x=x0+i*h;
    W0(i)=feval(g,x);
end
A=zeros(n+1);
A(1,1)=-1;
A(1,2)=1;
A(n+1,n)=1;
A(n+1,n+1)=-1;
for i=2:n
    A(i,i)=1+lambda+beta/2;
    A(i,i-1)=-1*lambda/2;
    A(i,i+1)=-1*(lambda/2+beta/2);
end
B=zeros(n+1);
for i=2:n
    B(i,i)=1-lambda-beta/2;
    B(i,i-1)=lambda/2;
    B(i,i+1)=lambda/2+beta/2;
end
for j=1:m
    WJ=zeros(n+1,1);
    tj=j*k;
    tj1=(j+1)*k;
    b=zeros(n+1,1);
    b(1)=h*feval(h1,tj1);
    b(n+1)=-h*feval(h2,tj1);
    for i=2:n
        xi=x0+(i-1)*h;
```

```
        b(i)=k*(feval(f,xi,tj)+feval(f,xi,tj1))/2;
    end
    C=zeros(n-1,1);
    if j==1
        Wcero=zeros(n+1,1);
        for i=2:n
            Wcero(i)=W0(i-1);
        end
        for i=1:n+1
            C(i)=B(i,:)*Wcero+b(i);
        end
    else
        for i=1:n+1
            C(i)=B(i,:)*W(:,j-1)+b(i);
        end
    end
    WJ=A\C;
    W(:,j)=WJ;
end
```

2.9 Ejemplos

• *Ejemplo 2.11*

Tenemos el siguiente planteamiento:

$$u_t = u_{xx} + u_x + e^x(\cos(t) - 2sen(t))$$
$$0 < x < 1$$
$$0 < t$$

Y las siguientes condiciones de contorno:

$$u_x(0,t) = sen(t)$$
$$u_x(1,t) = e \cdot sen(t)$$

Con la condición inicial:

$$u(x,0) = 0$$

La solución es:

$$u(x,t) = e^x sen(t)$$

Vamos a hacer una comparativa de los tres métodos.

Las funciones f, g, h1 y h2 son:

f.m

```
function z=f(x,t)
z=exp(x)*(cos(t)-2*sin(t));
```

g.m

```
function z=g(x)
z=0;
```

h1.m

```
function z=h1(t)
z=sin(t);
```

h2.m

```
function z=h2(t)
z=exp(1)*sin(t);
```

Primera comparativa: t=0.5 segundos

```
>> [W0,Wprog]=metodoprogresivo(0,10,1,0.01,'f','g','h1','h2',50,1,1);
>> [W0,Wreg]=metodoregresivo(0,10,1,0.01,'f','g','h1','h2',50,1,1);
>> [W0,Wcrank]=metodocranknic(0,10,1,0.01,'f','g','h1','h2',50,1,1);
```

Tabla 1: Soluciones aproximadas y solución real. t=0,5 segundos.

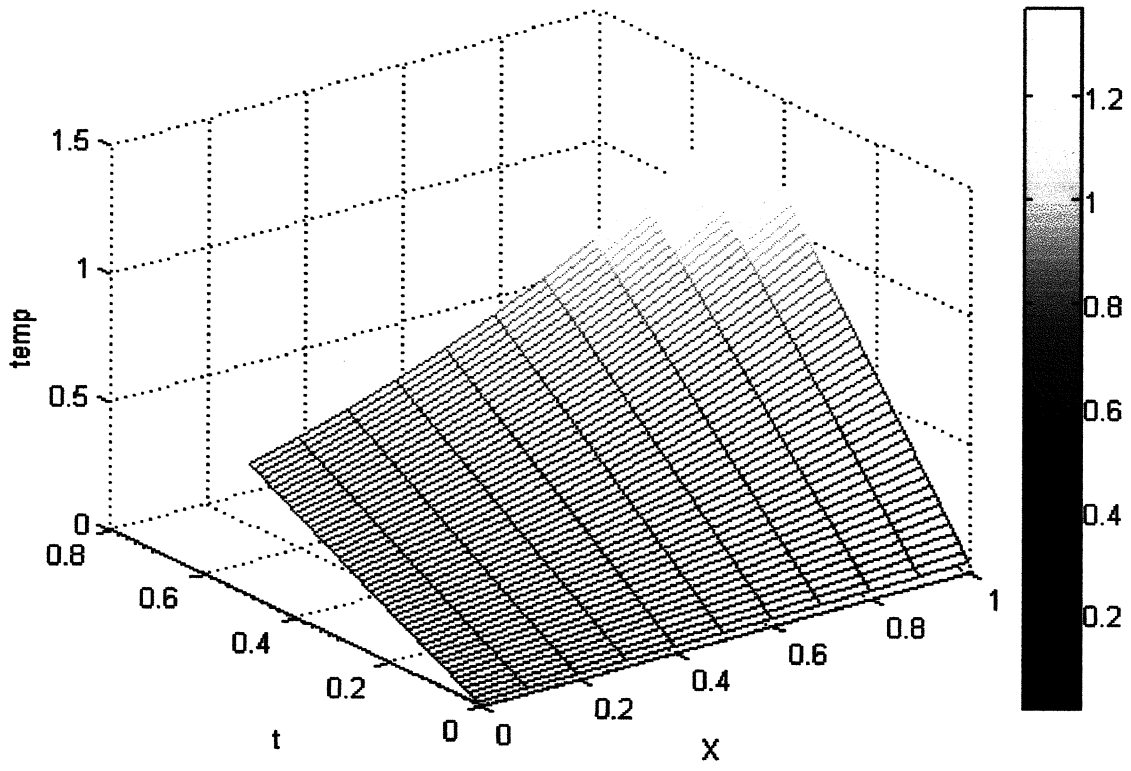
X	Wprog(:,50)	Wreg(:,50)	Wcrank(:,50)	Solución real
0.0	2.79×10^{12}	0.51879866	0.51098340	0.47942553
0.1	-3.11×10^{12}	0.56674122	0.55980113	0.52984716
0.2	3.85×10^{12}	0.62136345	0.61543567	0.58557167
0.3	-3.70×10^{12}	0.68305854	0.67828435	0.64715678
0.4	4.41×10^{12}	0.75231410	0.74884043	0.71521885
0.5	-3.94×10^{12}	0.82971236	0.82769323	0.79043908
0.6	4.41×10^{12}	0.91593143	0.91552932	0.87357028
0.7	-3.79×10^{12}	1.01174799	1.01313522	0.96544447
0.8	3.87×10^{12}	1.11804120	1.12140125	1.06698115
0.9	-3.24×10^{12}	1.23579792	1.24132684	1.17919654
1.0	2.91×10^{12}	1.36611929	1.37402718	1.30321372

Tabla 2: Diferencias entre las soluciones aproximadas y solución real. t=0,5 segundos.

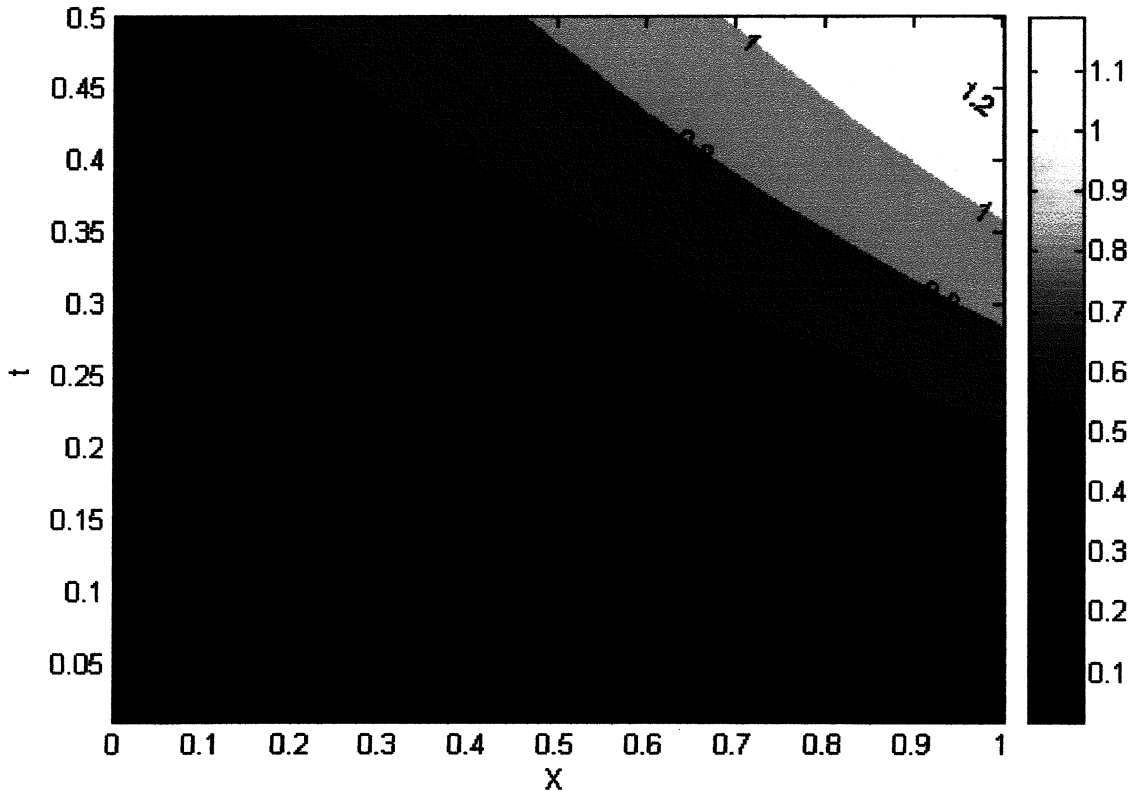
X	$\lvert Wprog(:,50)-U_{i,0'5}\rvert$	$\lvert Wreg(:,50)-U_{i,0'5}\rvert$	$\lvert Wcrank(:,50)-U_{i,0'5}\rvert$
0.0	2.79×10^{12}	0.03937312	0.03155786
0.1	3.11×10^{12}	0.03689405	0.02995396
0.2	3.85×10^{12}	0.03579178	0.02986400

0.3	3.70×10^{12}	0.03590175	0.03112756
0.4	4.41×10^{12}	0.03709524	0.03362157
0.5	3.94×10^{12}	0.03927328	0.03725414
0.6	4.41×10^{12}	0.04236114	0.04195903
0.7	3.79×10^{12}	0.04630352	0.04769074
0.8	3.87×10^{12}	0.05106004	0.05442009
0.9	3.24×10^{12}	0.05660137	0.06213030
1.0	2.91×10^{12}	0.06290556	0.07081345

Se observa claramente que la solución aportada por el método progresivo es muy mala, Las soluciones de los otros dos métodos son muy parecidas entre sí: en algunos casos la más aproximada es la de Crank-Nicolson y en otros, la del regresivo. Veámos el gráfico correspondiente a la solución de Crank-Nicolson:



Y ahora veamos las curvas de nivel:



Segunda comparativa: t=1 segundo

>> [W0,Wprog]=metodoprogresivo(0,10,1,0.01,'f','g','h1','h2',100,1,1);

>> [W0,Weg]=metodoregresivo(0,10,1,0.01,'f','g','h1','h2',100,1,1);

>> [W0,Wcrank]=metodocranknic(0,10,1,0.01,'f','g','h1','h2',100,1,1);

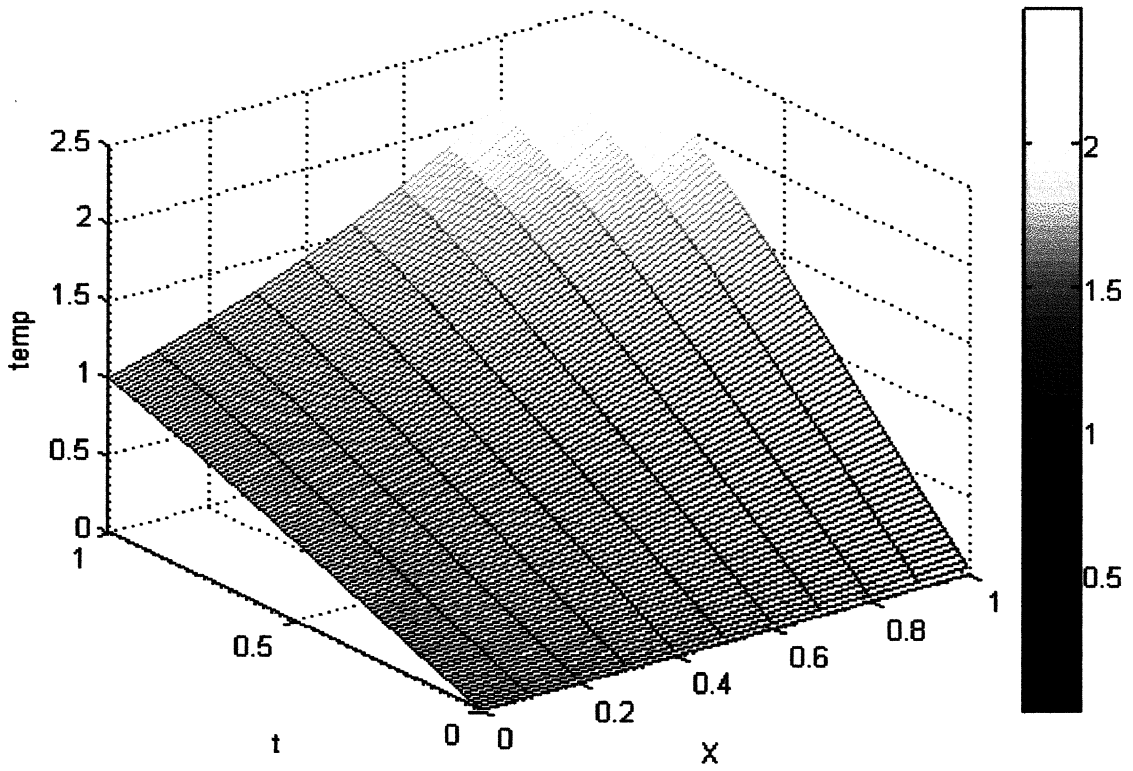
Tabla 1: Soluciones aproximadas y solución real. t=1 segundo.

X	Wprog(:,100)	Wreg(:,100)	Wcrank(:,100)	Solución real
0.0	2.0×10^{29}	0.99489033	0.98729767	0.84147098
0.1	-2.3×10^{29}	1.07903743	1.07198086	0.92996926
0.2	2.5×10^{29}	1.17507541	1.16865882	1.02777498
0.3	-2.7×10^{29}	1.28369742	1.27802260	1.13586702
0.4	2.8×10^{29}	1.40573645	1.40090378	1.25532719
0.5	-2.7×10^{29}	1.54217062	1.53827985	1.38735111
0.6	2.6×10^{29}	1.69413002	1.69128099	1.53326010
0.7	-2.5×10^{29}	1.86290520	1.86119860	1.69451447
0.8	2.3×10^{29}	2.04995744	2.04949554	1.87272811
0.9	-2.0×10^{29}	2.25693081	2.25781829	2.06968465
1.0	1.7×10^{29}	2.48566633	2.48801105	2.28735528

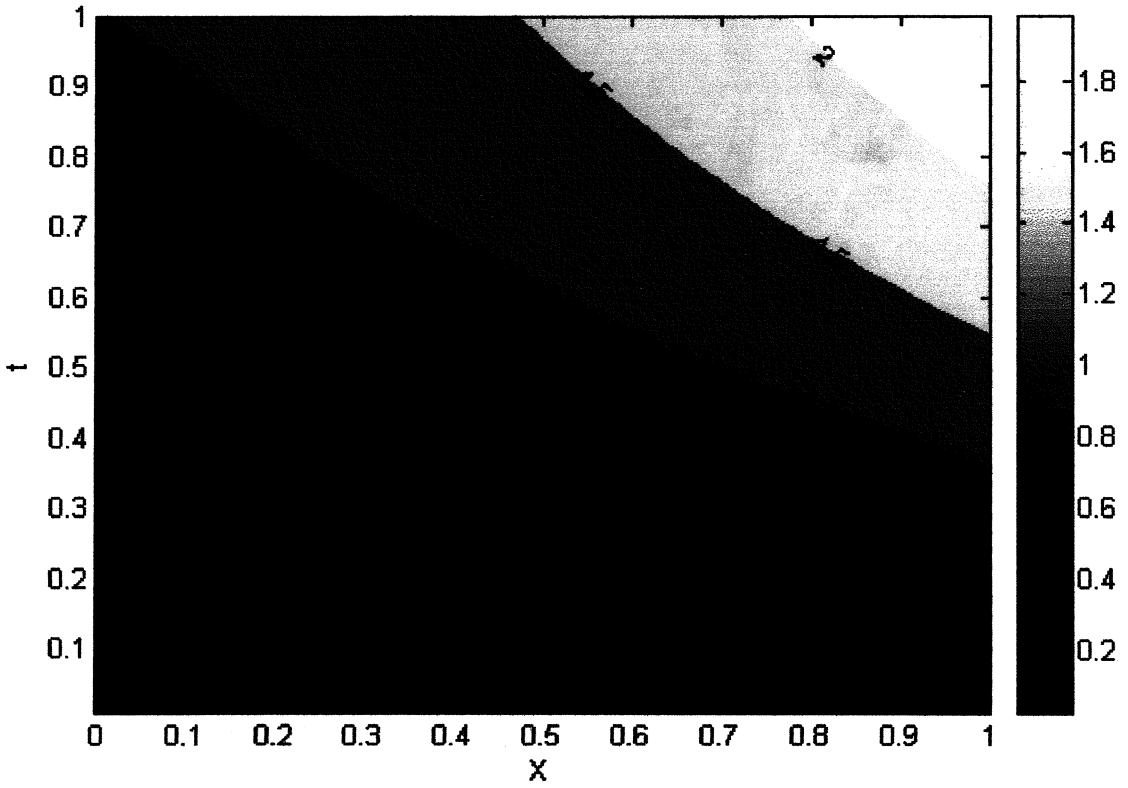
Tabla 2: Diferencias entre las soluciones aproximadas y solución real. t=1 segundo.

X	$\|Wprog(:,100)-U_{I,1}\|$	$\|Wreg(:,100)-U_{I,1}\|$	$\|Wcrank(:,100)-U_{I,1}\|$
0.0	2.0×10^{29}	0.15341934	0.14582669
0.1	-2.3×10^{29}	0.14906817	0.14201159
0.2	2.5×10^{29}	0.14730043	0.14088384
0.3	-2.7×10^{29}	0.14783040	0.14215558
0.4	2.8×10^{29}	0.15040925	0.14557659
0.5	-2.7×10^{29}	0.15481951	0.15092874
0.6	2.6×10^{29}	0.16086992	0.15802089
0.7	-2.5×10^{29}	0.16839073	0.16668413
0.8	2.3×10^{29}	0.17722932	0.17676743
0.9	-2.0×10^{29}	0.18724615	0.18813364
1.0	1.7×10^{29}	0.19831105	0.20065577

Se observa claramente que la solución aportada por el método progresivo es muy mala, Las soluciones de los otros dos métodos son muy parecidas entre sí: en algunos casos la más aproximada es la de Crank-Nicolson y en otros, la del regresivo. Veámos el gráfico correspondiente a la solución de Crank-Nicolson:



Y ahora veamos las curvas de nivel:


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
t
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
X
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2

3. Ecuación hiperbólica: ecuación de ondas

3.1 Planteamiento del problema. Ecuación.

La ecuación es:

Ecuación E3.1

$$\frac{\partial^2 u}{\partial t^2}(x,t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x,t) + H(x,t)$$

Lo que podemos expresar también de la siguiente manera:

$$U_{tt} = \alpha^2 U_{xx} + H$$

Y lo que conocemos es el contorno y la situación inicial, lo que expresamos así:

$$u(x_0,t) = h_1(t)$$
$$u(x_n,t) = h_2(t)$$

$$u(x,0) = f(x)$$
$$\frac{\partial u}{\partial t}(x,0) = g(x)$$

3.2 Discretización del problema.

Discretizamos el problema mediante la discretización de la segunda derivada (ver apéndice). La ecuación E3.1 quedaría así:

$$\frac{u(x,t+k) - 2u(x,t) + u(x,t-k)}{k^2} + O(k^2) = \alpha^2 \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2} + O(h^2) + H(x,t)$$

Si despreciamos los errores y tomamos $x=x_i$, $t=t_j$, $u(x_i, t_j) \approx W_{i,j}$, y $\lambda=\alpha k/h$, tenemos que:

$$W_{i,j+1} - 2W_{i,j} + W_{i,j-1} = \lambda^2 [W_{i+1,j} - 2W_{i,j} + W_{i-1,j}] + k^2 H_{i,j}$$

Y, despejando $W_{i,j+1}$,

Ecuación E3.2

$$W_{i,j+1} = \lambda^2 W_{i-1,j} + 2(1-\lambda^2) W_{i,j} + \lambda^2 W_{i+1,j} - W_{i,j-1} + k^2 H_{i,j}$$

3.3 Algoritmo A3.1: Ecuación de ondas

La ecuación E3.2 puede expresarse matricialmente de la siguiente manera:

$$\underbrace{\begin{pmatrix} W_{1,j+1} \\ \vdots \\ \vdots \\ W_{n-1,j+1} \end{pmatrix}}_{W_{j+1}} = \underbrace{\begin{pmatrix} 2(1-\lambda^2) & \lambda^2 & 0 & \cdots & 0 & 0 \\ \lambda^2 & 2(1-\lambda^2) & \lambda^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \lambda^2 & 2(1-\lambda^2) \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} W_{1,j} \\ \vdots \\ \vdots \\ W_{n-1,j} \end{pmatrix}}_{W_j} - \underbrace{\begin{pmatrix} W_{1,j-1} \\ \vdots \\ \vdots \\ W_{n-1,j-1} \end{pmatrix}}_{W_{j-1}} + \underbrace{\begin{pmatrix} k^2 H_{1,j} + \lambda^2 h_1(t_j) \\ k^2 H_{2,j} \\ \vdots \\ k^2 H_{n-1,j} + \lambda^2 h_2(t_j) \end{pmatrix}}_{b}$$

Por tanto, el método queda expresado así:

$$\begin{cases} W_{j+1} = AW_j - W_{j-1} + b \\ W_0 = \{f(x_i)\}_{i=1\ldots n-1} \\ W_1 = ¿? \end{cases}$$

Como se observa, nos falta por conocer el valor de W_1. A continuación se muestran dos formas de calcularlo; nosotros nos quedaremos con la segunda, que es mejor.

3.3.1. Primera propuesta para hallar W_1

Lo que queremos calcular es $u(x_i,t_1)$, siendo $t_1=k$. Por tanto:

$$u(x_i,t_1) = u(x_i,k) \overset{TAYLOR}{=} u(x_i,0) + k\frac{\partial u}{\partial t}(x_i,0) + \frac{k^2}{2!}\frac{\partial^2 u}{\partial t^2}(x_i,0) + O(k^3)$$

Si despreciamos los dos últimos sumandos:

$$W_1 = \{f(x_i) + k \cdot g(x_i)\}_{i=1}^{n-1}$$

3.3.2. Segunda propuesta para hallar W_1

El término $\frac{k^2}{2!}\frac{\partial^2 u}{\partial t^2}(x_i,0)$ lo hemos despreciado en la propuesta anterior. Sin embargo, podemos observar que la parte $\frac{\partial^2 u}{\partial t^2}(x_i,0)$ podemos sacarla de la ecuación E3.1:

$$\frac{\partial^2 u}{\partial t^2}(x_i,0) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x_i,0) + H(x_i,0)$$

Llamando $f''(x_i) = \frac{\partial^2 u}{\partial x^2}(x_i,0) + H(x_i,0)$, tenemos que:

$$W_1 = \left\{ f(x_i) + k \cdot g(x_i) + \frac{k^2\alpha^2}{2} f''(x_i) \right\}_{i=1}^{n-1}$$

Por tanto, el método queda expresado así:

$$\begin{cases} W_{j+1} = AW_j - W_{j-1} + b \\ W_0 = \{f(x_i)\}_{i=1\ldots n-1} \\ W_1 = \left\{ f(x_i) + k \cdot g(x_i) + \frac{k^2\alpha^2}{2} f''(x_i) \right\}_{i=1}^{n-1} \end{cases}$$

Este método es condicionalmente estable. Para que sea estable, λ tiene que ser menor o igual que 1. La convergencia es buena (de orden $O(h^2+k^2)$).

Algoritmo

Datos que recibe el algoritmo:

- x0: valor inicial de x
- n: número de divisiones del eje X
- L: x_n
- k: valor del paso en la variable tiempo
- H: función H(x,t)
- f: función f(x)
- fpp: función f''(x)
- g: función g(x)
- h1: función $h_1(t)$
- h2: función $h_2(t)$
- m: número de divisiones que se harán en el eje Y
- alpha: valor de α

Datos que devuelve el algoritmo:

- W0: vector que contiene los valores iniciales
- W1: vector que contiene los valores para t=k
- W: matriz de n-1 filas y m-1 columnas. El valor (i,j) representa al valor de x_i en el instante (j+1)*k

```
function [W0,W1,W] = econdas(x0,n,L,k,H,f,fpp,g,h1,h2,m,alpha)
h=(L-x0)/n;
lambda=alpha*k/h;
W=zeros(n-1,m-1);
W0=zeros(n-1,1);
W1=zeros(n-1,1);

for i=1:n-1
    x=x0+i*h;
    W0(i)=feval(f,x);
end

for i=1:n-1
    x=x0+i*h;
    W1(i)=feval(f,x)+k*feval(g,x)+(k^2/2)*alpha^2*feval(fpp,x);
end

A=zeros(n-1);
for i=1:n-1
    A(i,i)=2*(1-lambda^2);
```

```
    if i>1
        A(i,i-1)=lambda^2;
    end
    if i<n-1
        A(i,i+1)=lambda^2;
    end
end
for j=1:m-1
    WJ=zeros(n-1,1);
    tj=j*k;
    b=zeros(n-1,1);
    for i=1:n-1
        xi=x0+i*h;
        b(i)=k^2*(feval(H,0,0,0,xi,tj));
        if i==1
            b(i)=b(i)+(lambda^2)*feval(h1,tj);
        end
        if i==n-1
            b(i)=b(i)+(lambda^2)*feval(h2,tj);
        end
    end
    if j==1
        for i=1:n-1
            WJ(i)=(A(i,:)*W1-W0(i)+b(i));
        end
    else if j==2
            for i=1:n-1
                WJ(i)=(A(i,:)*W(:,j-1)-W1(i)+b(i));
            end
        else
            for i=1:n-1
                WJ(i)=(A(i,:)*W(:,j-1)-W(i,j-2)+b(i));
            end
        end
    end
    W(:,j)=WJ;
End
```

3.4 Ejemplos

• *Ejemplo 3.1*

Tenemos el siguiente planteamiento:

$$U_{tt} = U_{xx} - 6xt$$

Con las siguientes condiciones de contorno:

$$u(0,t) = 0$$
$$u(1,t) = sen(1) \cdot sen(t) + t$$

Y las siguientes condiciones iniciales:

$$u(x,0) = 0$$
$$u_t(x,0) = \text{s}en(x) + x^3$$

Siendo la solución:

$$u(x,t) = sen(x) \cdot sen(t) + x^3 \cdot t$$

Llamaremos al algoritmo A3.1 para intentar aproximar la solución.

>> [W0,W1,W] = econdas(0,10,1,0.01,'H','f','fpp','g','h1','h2',100,1);

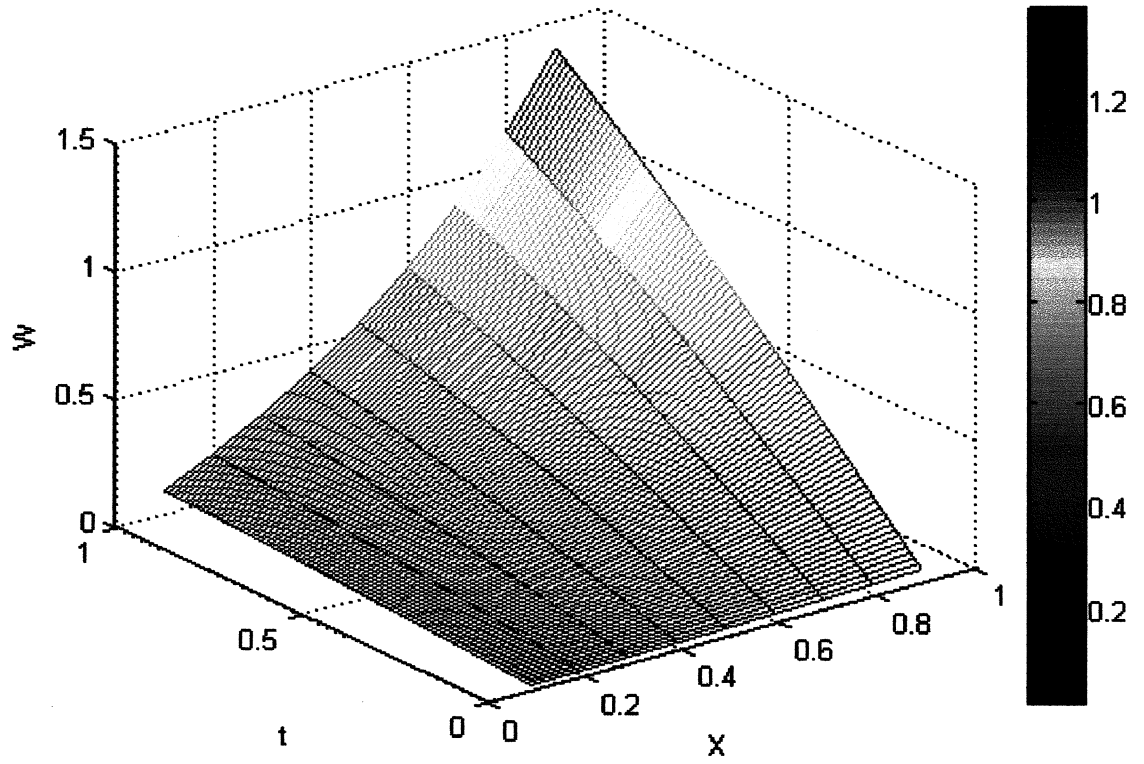
Con la instrucción anterior hemos calculado los valores de la ecuación de ondas desde t=0 a t=1. Veamos primero la comparativa entre la solución aproximada para t=0.5 y la solución real.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.04836268954660	0.04836515941960	0.0000024698729
0.09924715092056	0.09925206522135	0.0000049143007
0.15517993424704	0.15518723657060	0.0000073023235
0.21869709850368	0.21870666511574	0.0000095666120
0.29234884706593	0.29236033811291	0.0000114910469
0.37870402192622	0.37871657573662	0.0000125538103
0.48035441168228	0.48036660669192	0.0000121950096
0.59991883025051	0.59992922391735	0.0000103936668
0.74004692555132	0.74005356975455	0.0000066442032

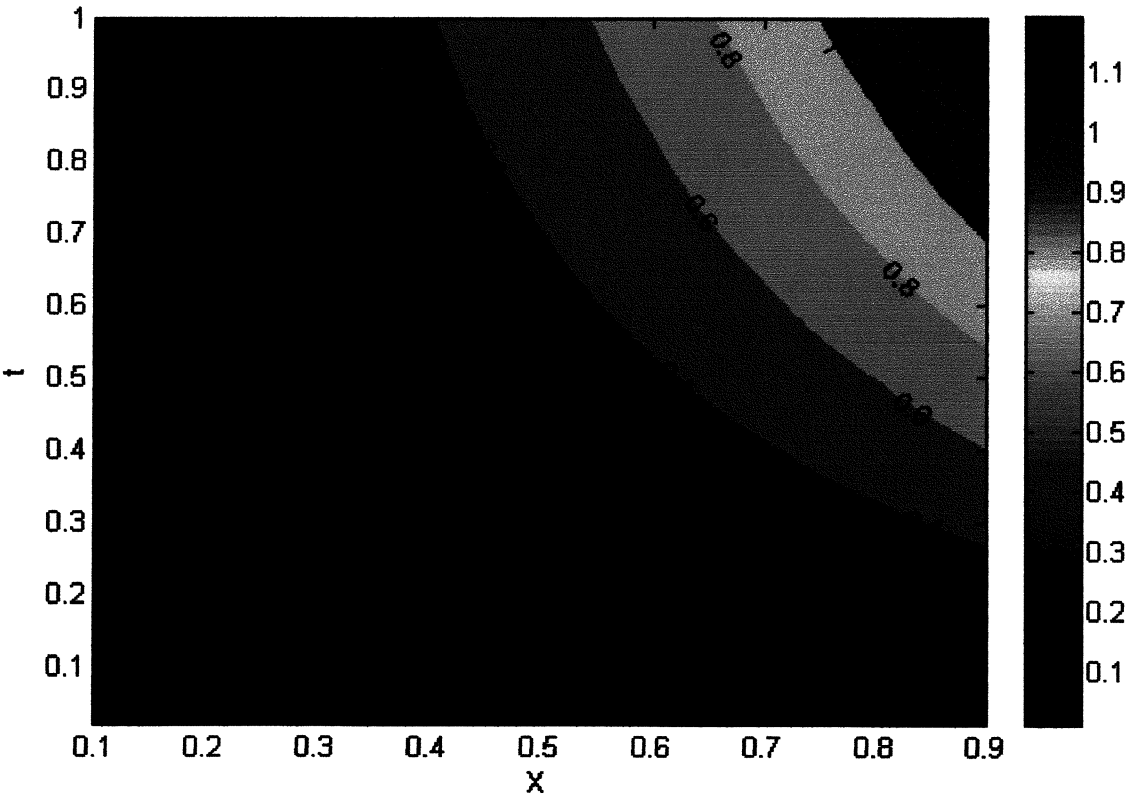
Ahora, la comparativa entre la solución aproximada para t=1 y la solución real.

Solución real t=1	Solución aproximada	Diferencia absoluta
0.08500692342254	0.08501933626214	0.0000124128395
0.17517447743525	0.17519823388959	0.0000237564543
0.27567167932995	0.27570504290323	0.0000333635732
0.39168423600472	0.39172509907731	0.0000408630725
0.52842268011133	0.52846808553522	0.0000454054238
0.69113025815209	0.69117620258967	0.0000459444375
0.88509049171057	0.88513267917136	0.0000421874607
1.11563433626716	1.11566790833342	0.0000335720662
1.38814686607076	1.38816644875670	0.0000195826859

Como vemos, la aproximación realizada es excelente. Veamos un gráfico que refleja la solución:



Y las curvas de nivel:



- ***Ejemplo 3.2***

Tenemos el siguiente planteamiento:

$$U_{tt} = U_{xx} - 20x^3 + 20t^3$$

Con las siguientes condiciones de contorno:

$$u(0,t) = t^5$$
$$u(1,t) = 1 + t^5$$

Y las siguientes condiciones iniciales:

$$u(x,0) = x^5$$
$$u_t(x,0) = 0$$

Siendo la solución:

$$u(x,t) = x^5 + t^5$$

Llamaremos al algoritmo A3.1 para intentar aproximar la solución. Esta vez usaremos un k más pequeño que en el ejemplo anterior, para ver si la aproximación es mejor o peor.

>> [W0,W1,W] = econdas(0,10,1,0.001,'H','f','fpp','g','h1','h2',1000,1);

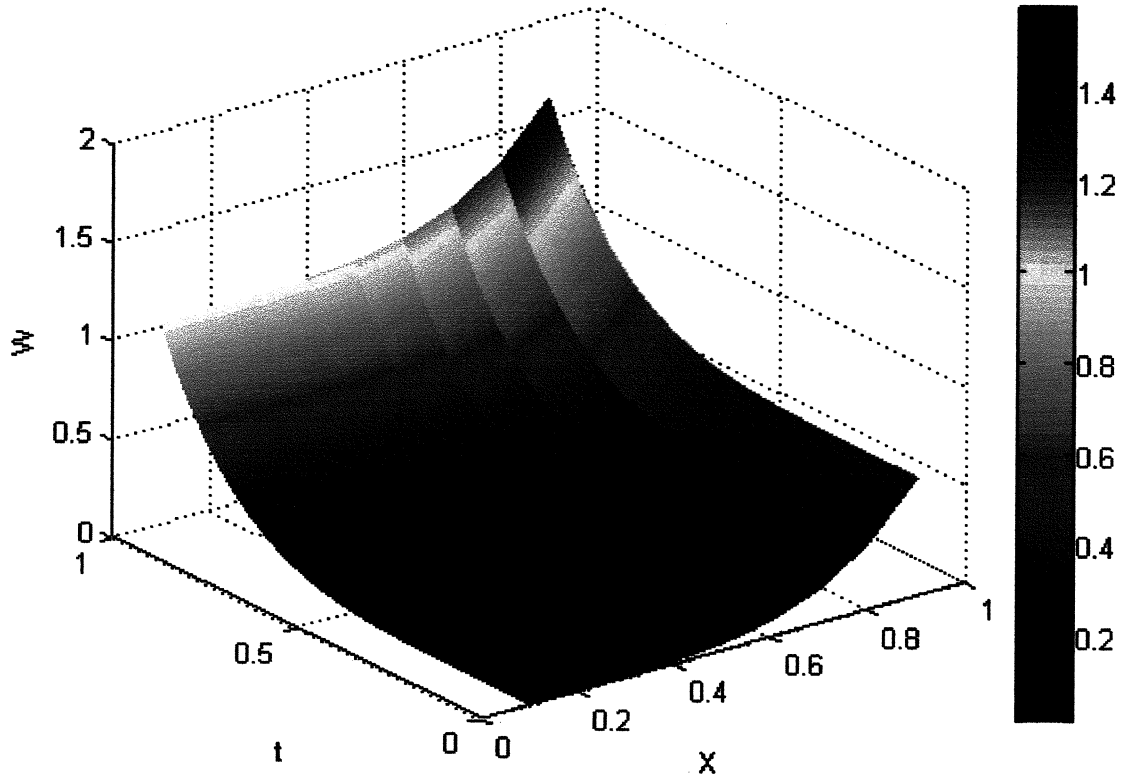
Con la instrucción anterior hemos calculado los valores de la ecuación de ondas desde t=0 a t=1. Veamos primero la comparativa entre la solución aproximada para t=0.5 y la solución real.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.03126000000000	0.03252456043514	0.00126456043514
0.03157000000000	0.03410647260712	0.00253647260712
0.03368000000000	0.03750112772988	0.00382112772988
0.04149000000000	0.04659725763930	0.00510725763930
0.06250000000000	0.06880052143837	0.00630052143837
0.10901000000000	0.11608356967671	0.00707356967671
0.19932000000000	0.20619628093674	0.00687628093674
0.35893000000000	0.36444947841479	0.00551947841479
0.62174000000000	0.62500368791917	0.00326368791917

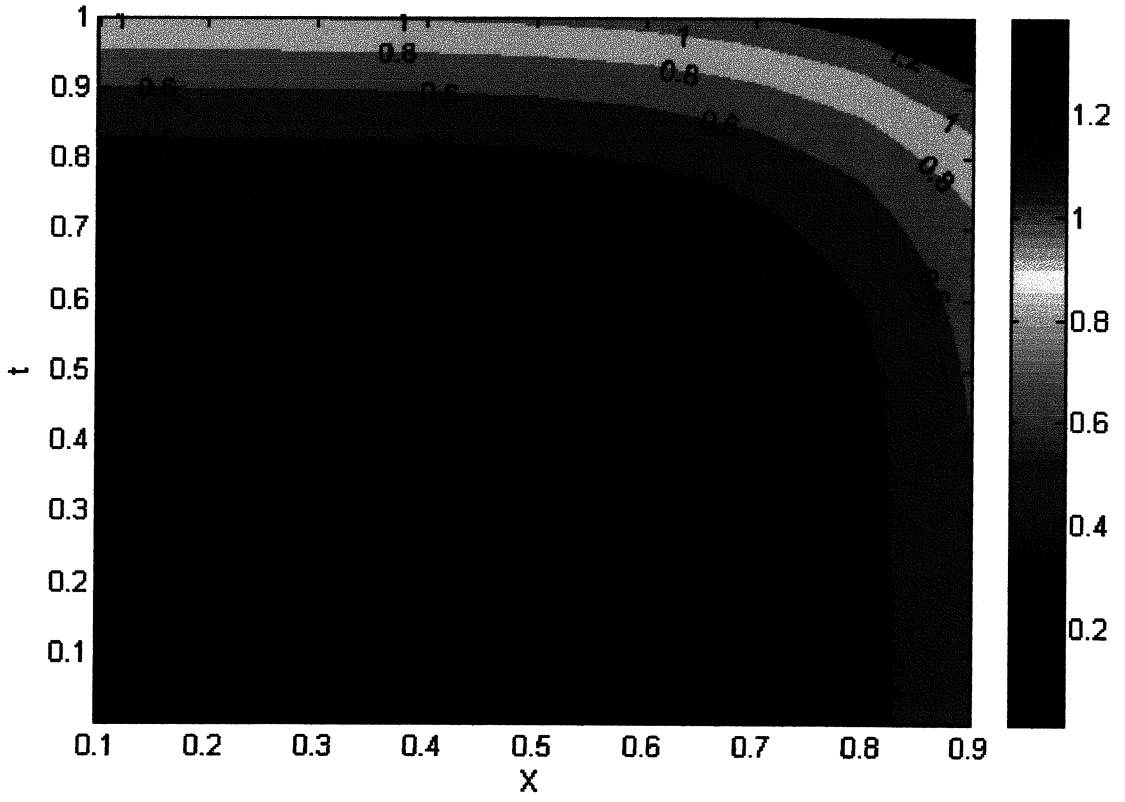
Ahora, la comparativa entre la solución aproximada para t=1 y la solución real.

Solución real t=1	Solución aproximada	Diferencia absoluta
1.00001000000000	1.00439683397623	0.00438683397623
1.00032000000000	1.00837851977669	0.00805851977669
1.00243000000000	1.01298867766086	0.01055867766086
1.01024000000000	1.02219525535076	0.01195525535076
1.03125000000000	1.04374006042485	0.01249006042485
1.07776000000000	1.08977713016017	0.01201713016017
1.16807000000000	1.17855196262200	0.01048196262200
1.32768000000000	1.33568693461965	0.00800693461965
1.59049000000000	1.59498637769632	0.00449637769632

Como vemos, la aproximación realizada es muy buena, aunque no tan buena como la realizada en el ejemplo anterior. Veamos un gráfico que refleja la solución:



Y las curvas de nivel:



• *Ejemplo 3.3*

Tenemos el siguiente planteamiento:

$$U_{tt} = U_{xx}$$

Con las siguientes condiciones de contorno:

$$u(0,t) = t^2$$
$$u(1,t) = 1 + t^2$$

Y las siguientes condiciones iniciales:

$$u(x,0) = x^2$$
$$u_t(x,0) = 0$$

Siendo la solución:

$$u(x,t) = x^2 + t^2$$

Llamaremos al algoritmo A3.1 para intentar aproximar la solución. Vamos a comparar, para el mismo instante de tiempo (t=1), cómo de buena es la aproximación dependiendo del k usado.

>> [W0,W1,W] = econdas(0,10,1,0.01,'H','f','fpp','g','h1','h2',100,1);

Comparativa para k=0.01, en t=1 segundo.

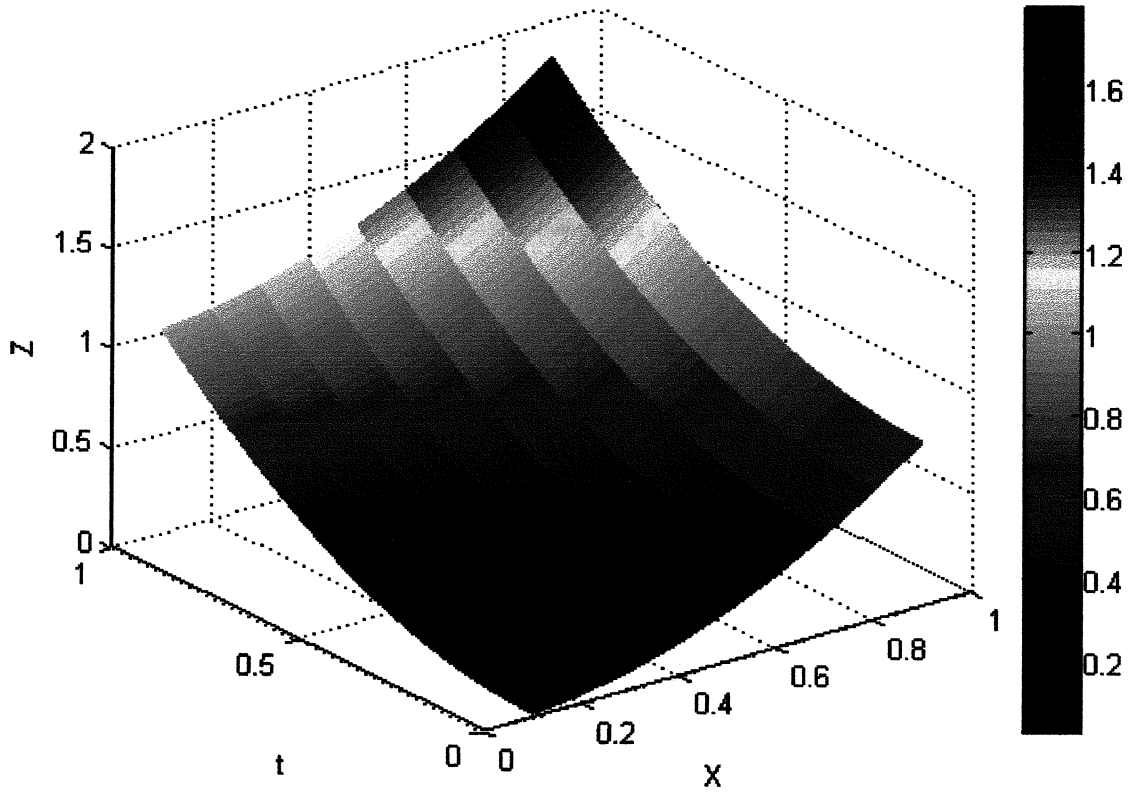
Solución real t=1	Solución aproximada	Diferencia absoluta
1.01000000000000	1.01000833791410	0.0000083379141
1.04000000000000	1.03998522204625	0.0000147779537
1.09000000000000	1.09001267576160	0.0000126757616
1.16000000000000	1.16001016986817	0.0000101698681
1.25000000000000	1.24994513726862	0.0000548627313
1.36000000000000	1.36006690146305	0.0000669014630
1.49000000000000	1.49004701546150	0.0000470154615
1.64000000000000	1.63980801095137	0.0001919890486
1.81000000000000	1.80975182533868	0.0002481746613

>> [W0,W1,W] = econdas(0,10,1,0.001,'H','f','fpp','g','h1','h2',1000,1);

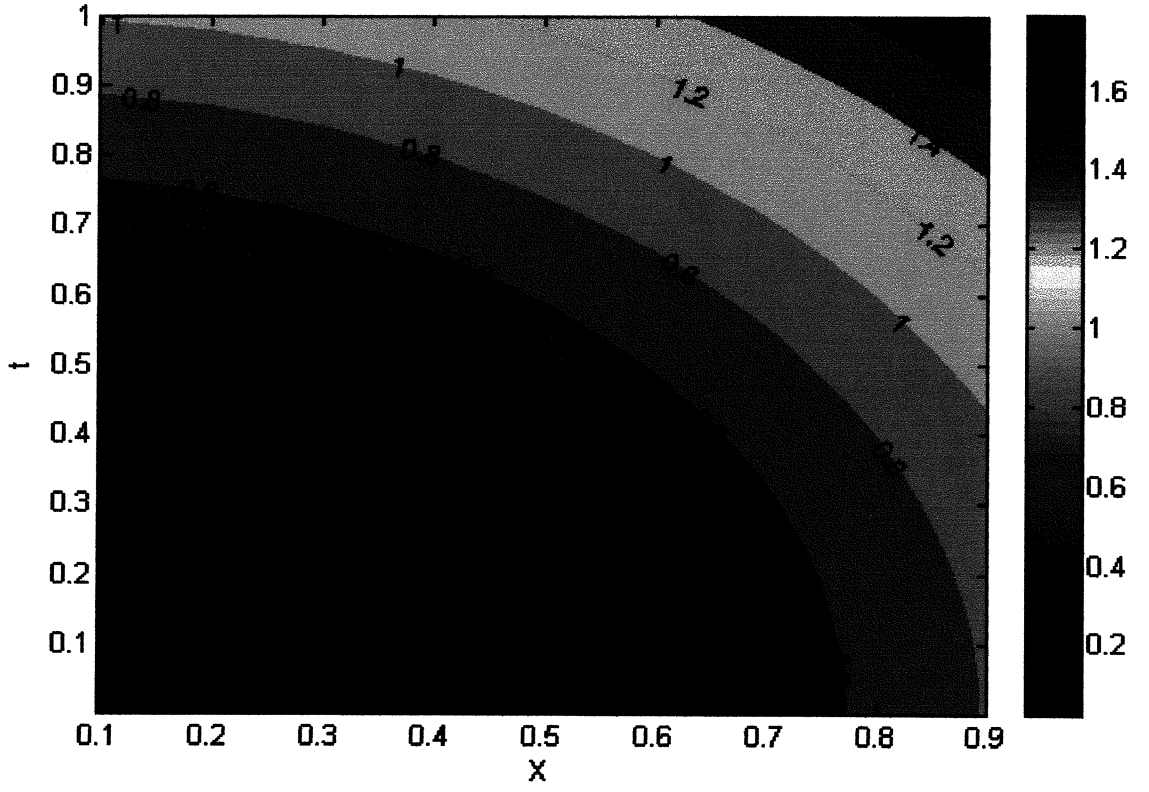
Comparativa para k=0.001, en t=1 segundo.

Solución real t=1	Solución aproximada	Diferencia absoluta
1.01000000000000	1.01000084531979	0.00000084531979
1.04000000000000	1.03999848440339	0.00000151559660
1.09000000000000	1.09000134859649	0.00000134859648
1.16000000000000	1.16000091012012	0.00000091012012
1.25000000000000	1.24999455370144	0.00000544629855
1.36000000000000	1.36000680906214	0.00000680906214
1.49000000000000	1.49000455818683	0.00000455818682
1.64000000000000	1.63998063300677	0.00001936699322
1.81000000000000	1.80997515199612	0.00002484800387

Se observa claramente que la aproximación con k=0.001 es mejor que la aproximación con k=0.01. Veamos un gráfico de la última aproximación:



Y las curvas de nivel:



3.5 Variación de la ecuación principal: ecuación de ondas con fricción

A la ecuación E3.1 le vamos a añadir un nuevo término que modela cuando hay fricción. Dicho término es el siguiente:

$$A\frac{\partial u}{\partial x}(x,t)+B\cdot u(x,t)+C\frac{\partial u}{\partial t}(x,t)$$

Con lo que la ecuación E3.1 quedaría:

Ecuación E3.3

$$\frac{\partial^2 u}{\partial t^2}(x,t)=\alpha^2\frac{\partial^2 u}{\partial x^2}(x,t)+A\frac{\partial u}{\partial x}(x,t)+B\cdot u(x,t)+C\frac{\partial u}{\partial t}(x,t)+H(x,t)$$

Las condiciones de contorno e iniciales son las mismas.

3.5.1 Discretización

Para la discretización, usaremos la discretización de la segunda derivada y la segunda forma de discretización de la primera derivada (ver apéndice). De esta forma, la ecuación E3.3 quedaría:

$$\frac{u(x,t+k)-2u(x,t)+u(x,t-k)}{k^2}+O(k^2)=\alpha^2\frac{u(x+h,t)-2u(x,t)+u(x-h,t)}{h^2}+O(h^2)+$$
$$+A\frac{u(x+h,t)-u(x-h,t)}{2h}+O(h^2)+Bu(x,t)+C\frac{u(x,t+k)-u(x,t-k)}{2k}+O(k^2)+H(x,t)$$

Si despreciamos los errores y tomamos $x=x_i$, $t=t_j$ y $u(x_i, t_j) \approx W_{i,j}$ tenemos que:

$$\frac{W_{i,j+1}-2W_{i,j}+W_{i,j-1}}{k^2}=\alpha^2\frac{W_{i+1,j}-2W_{i,j}+W_{i-1,j}}{h^2}+A\frac{W_{i+1,j}-W_{i-1,j}}{2h}+BW_{i,j}+C\frac{W_{i,j+1}-W_{i,j-1}}{2k}+H_{i,j}$$

Tomando $\lambda=\alpha k/h$ nos queda:

$$W_{i,j+1}-2W_{i,j}+W_{i,j-1}=\lambda^2(W_{i+1,j}-2W_{i,j}+W_{i-1,j})+\frac{k^2A}{2h}(W_{i+1,j}-W_{i-1,j})+k^2BW_{i,j}+$$
$$+\frac{kC}{2}(W_{i,j+1}-W_{i,j-1})+k^2H_{i,j}$$

Y, despejando y agrupando:

Ecuación E3.4

$$(1-\frac{kC}{2})W_{i,j+1}=(2-2\lambda^2+k^2B)W_{i,j}-(1+\frac{kC}{2})W_{i,j-1}+(\lambda^2+\frac{k^2A}{2h})W_{i+1,j}+(\lambda^2-\frac{k^2A}{2h})W_{i-1,j}+k^2H_{i,j}$$

3.5.2 Algoritmo A3.2: Ecuación de ondas con fricción

La ecuación E3.4 puede expresarse matricialmente de a siguiente forma:

$$(1+\frac{kC}{2})\underbrace{\begin{pmatrix}W_{1,j+1}\\ \vdots\\ \vdots\\ W_{n-1,j+1}\end{pmatrix}}_{w_{j+1}}=\underbrace{\begin{pmatrix}2(1-\lambda^2)+k^2B & \lambda^2+\frac{k^2A}{2h} & 0 & \cdots & 0 & 0\\ \lambda^2-\frac{k^2A}{2h} & 2(1-\lambda^2)+k^2B & \lambda^2+\frac{k^2A}{2h} & \cdots & 0 & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & \lambda^2-\frac{k^2A}{2h} & 2(1-\lambda^2)+k^2B\end{pmatrix}}_{\hat{A}}.$$

```
for i=1:n-1
    x=x0+i*h;
    W0(i)=feval(f,x);
end
W1=zeros(n-1,1);
for i=1:n-1
    x=x0+i*h;
    W1(i)=feval(f,x)+k*feval(g,x)+(k^2/2)*alpha^2*feval(fpp,x);
end
A=zeros(n-1);
for i=1:n-1
    A(i,i)=2*(1-lambda^2)+k^2*B1;
    if i>1
        A(i,i-1)=lambda^2-((k^2*A1)/(2*h));
    end
    if i<n-1
        A(i,i+1)=lambda^2+((k^2*A1)/(2*h));
    end
end
for j=1:m-1
    WJ=zeros(n-1,1);
    tj=j*k;
    b=zeros(n-1,1);
    for i=1:n-1
        xi=x0+i*h;
        b(i)=k^2*(feval(H,A1,B1,C1,xi,tj));
        if i==1
            b(i)=b(i)+(lambda^2-((k^2*A1)/(2*h)))*feval(h1,tj);
        end
        if i==n-1
            b(i)=b(i)+(lambda^2+((k^2*A1)/(2*h)))*feval(h2,tj);
        end
    end
    if j==1
        for i=1:n-1
            WJ(i)=(A(i,:)*W1-(1+(k*C1/2))*W0(i)+b(i))*(1/(1-
(k*C1/2)));
        end
    else if j==2
            for i=1:n-1
```

```
                WJ(i)=(A(i,:)*W(:,1)-(1+(k*C1/2))*W1(i)+b(i))*(1/(1-
(k*C1/2)));
            end
        else
            for i=1:n-1
                WJ(i)=(A(i,:)*W(:,j-1)-(1+(k*C1/2))*W(i,j-
2)+b(i))*(1/(1-(k*C1/2)));
            end
        end
    end
    W(:,j)=WJ;
end
```

3.6 Ejemplos

- ***Ejemplo 3.4***

Tenemos el siguiente planteamiento:

$$U_{tt} = U_{xx}\text{-(6xt+Asen(t)cos(x)+3Ax}^2\text{t+Bsin(x)sin(t)+Bx}^3\text{t+Csin(x)cos(t)+Cx}^3\text{);}$$

En este primer ejemplo vamos a tomar A=1 y B=C=0.

Con las siguientes condiciones de contorno:

$$u(0,t) = 0$$
$$u(1,t) = sen(1)\cdot sen(t) + t$$

Y las siguientes condiciones iniciales:

$$u(x,0) = 0$$
$$u_t(x,0) = sen(x) + x^3$$

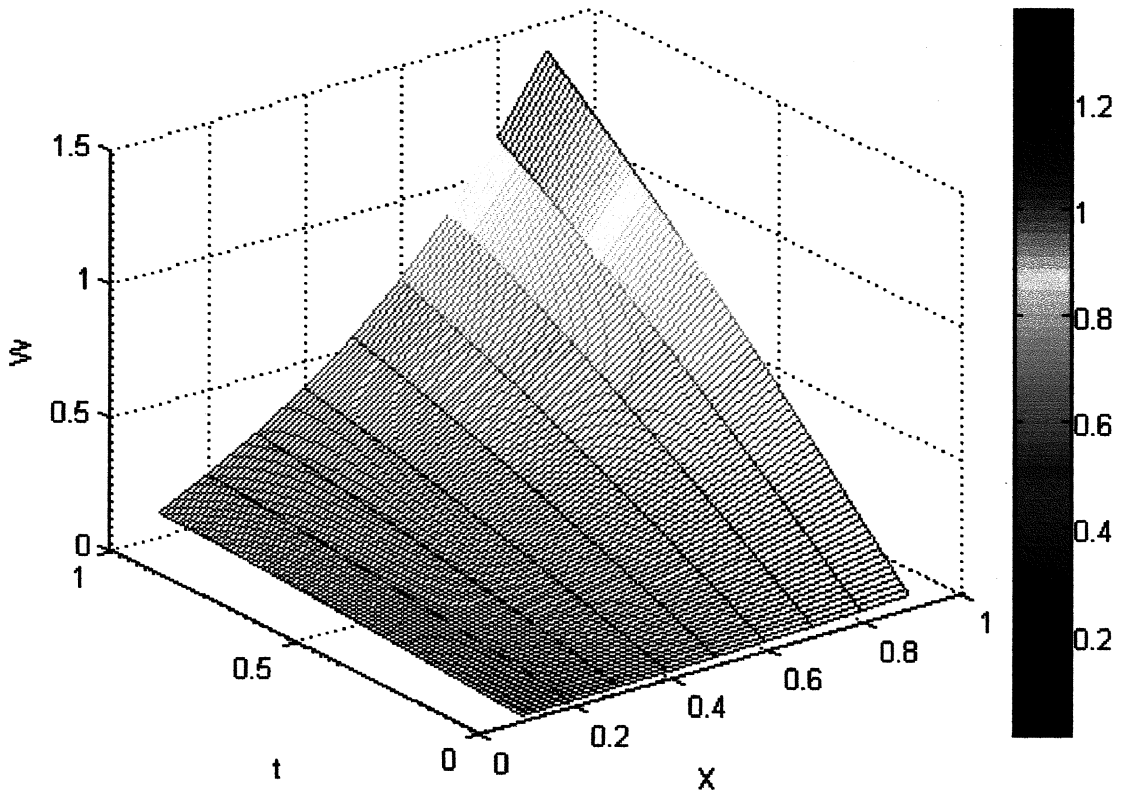
Siendo la solución:

$$u(x,t) = sen(x)\cdot sen(t) + x^3 \cdot t$$

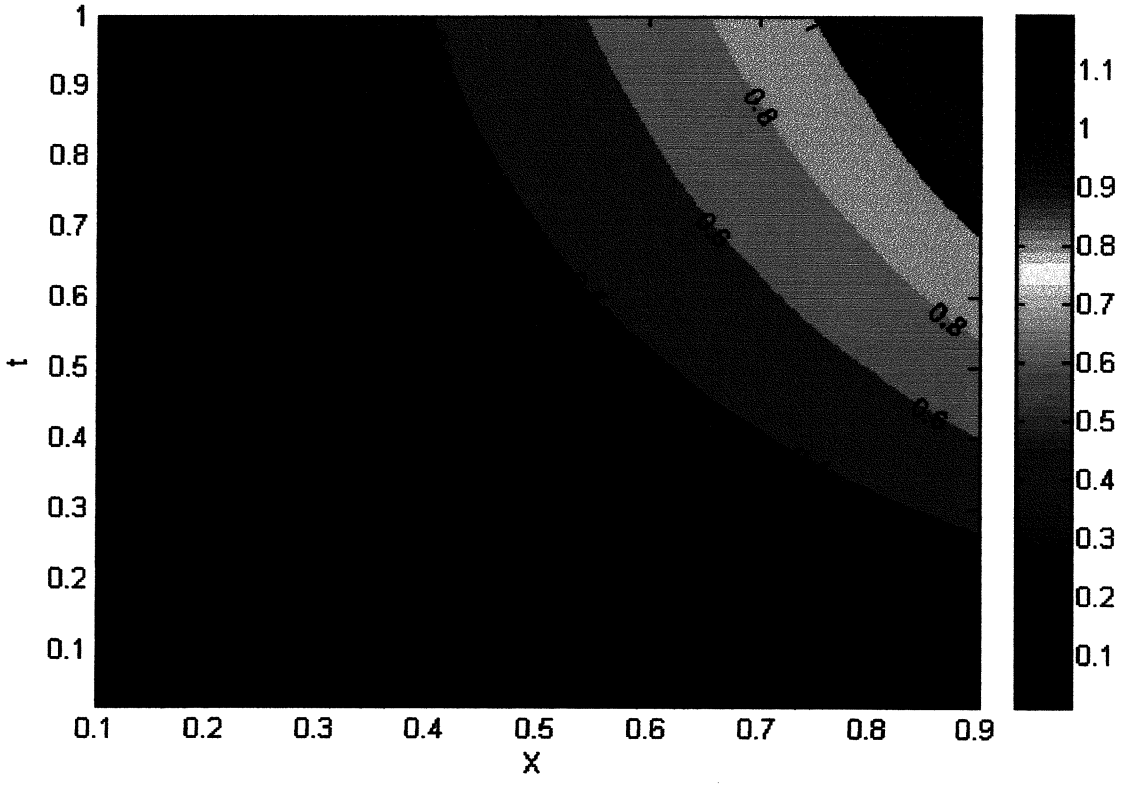
>> [W0,W1,W] = econdasfriccion(0,10,1,0.01,'H','f','fpp','g','h1','h2',50,1,1,0,0);

Comparativa para k=0.01, en t=0.5 segundos.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.04836268954660	0.04845508764917	0.00009239810256
0.09924715092056	0.09939330982185	0.00014615890128
0.15517993424704	0.15535322338995	0.00017328914290
0.21869709850368	0.21888195349500	0.00018485499132
0.29234884706593	0.29253841130444	0.00018956423850
0.37870402192622	0.37889340971176	0.00018938778553
0.48035441168228	0.48053350543109	0.00017909374880
0.59991883025051	0.60006793394356	0.00014910369304
0.74004692555132	0.74013814105972	0.00009121550839



Y las curvas de nivel:



• *Ejemplo 3.5*

Tenemos el siguiente planteamiento:

$$U_{tt} = U_{xx}\text{-(6xt+Asen(t)cos(x)+3Ax}^2\text{t+Bsin(x)sin(t)+Bx}^3\text{t+Csin(x)cos(t)+Cx}^3\text{);}$$

En este segundo ejemplo vamos a tomar A=C=0 y B=1.

Con las siguientes condiciones de contorno:

$$u(0,t) = 0$$
$$u(1,t) = sen(1)\cdot sen(t) + t$$

Y las siguientes condiciones iniciales:

$$u(x,0) = 0$$
$$u_t(x,0) = sen(x) + x^3$$

Siendo la solución:

$$u(x,t) = sen(x)\cdot sen(t) + x^3 \cdot t$$

>> [W0,W1,W] = econdasfriccion(0,10,1,0.01,'H','f','fpp','g','h1','h2',50,1,0,1,0);

Comparativa para k=0.01, en t=0.5 segundos.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.04836268954660	0.04836521478617	0.00000252523956
0.09924715092056	0.09925217539566	0.00000502447510
0.15517993424704	0.15518740038959	0.00000746614255
0.21869709850368	0.21870688042481	0.00000978192113
0.29234884706593	0.29236059965227	0.00001175258633
0.37870402192622	0.37871686823226	0.00001284630603
0.48035441168228	0.48036689502203	0.00001248333974
0.59991883025051	0.59992945823899	0.00001062798847
0.74004692555132	0.74005370474507	0.00000677919374

>> [W0,W1,W] = econdasfriccion(0,10,1,0.001,'H','f','fpp','g','h1','h2',500,1, 0,1,0);

Comparativa para k=0.001, en t=0.5 segundos.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.04836268954660	0.04836440881463	0.00000171926802
0.09924715092056	0.09925057223355	0.00000342131298
0.15517993424704	0.15518502289094	0.00000508864390
0.21869709850368	0.21870379874477	0.00000670024108
0.29234884706593	0.29235705779823	0.00000821073230
0.37870402192622	0.37871348819916	0.00000946627293
0.48035441168228	0.48036443648428	0.00001002480199
0.59991883025051	0.59992793300298	0.00000910275246
0.74004692555132	0.74005287751340	0.00000595196207

Se observa que la aproximación con k=0.001 es ligeramente mejor que la aproximación con k=0.01.

Ahora vamos a ver qué ocurre en t=1.

>> [W0,W1,W] = econdasfriccion(0,10,1,0.01,'H','f','fpp','g','h1','h2',100,1,0,1,0);

Comparativa para k=0.01, en t=1 segundo.

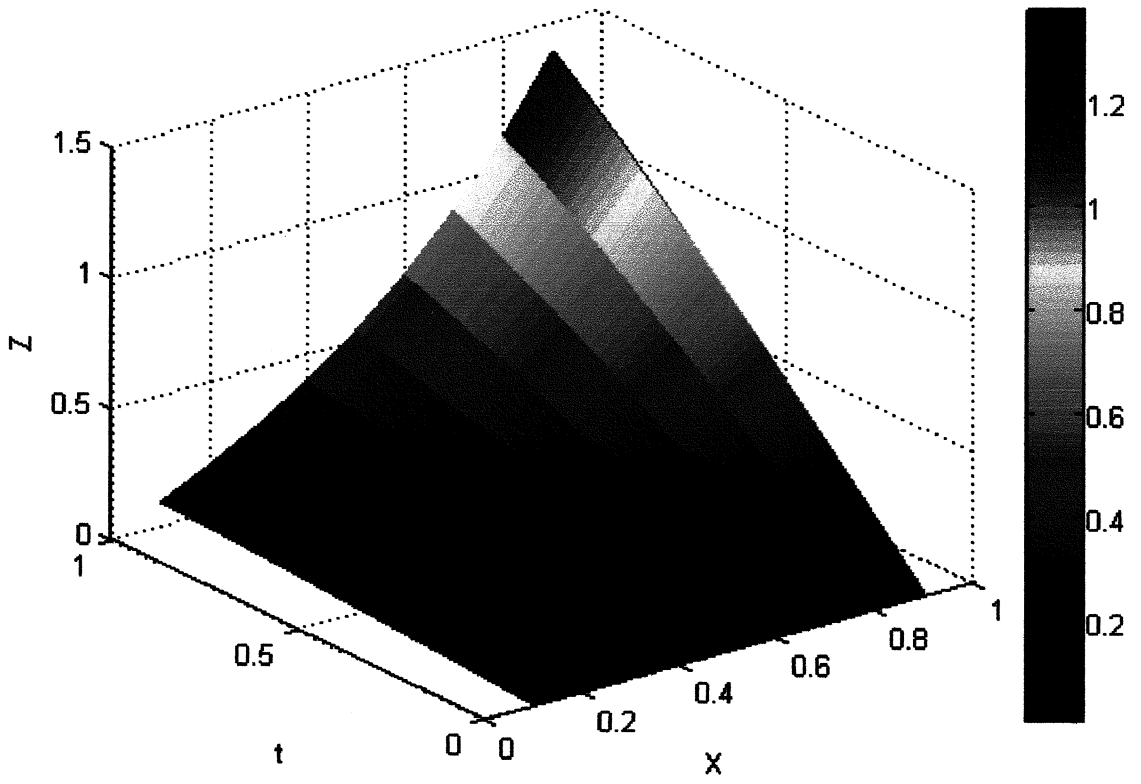
Solución real t=1	Solución aproximada	Diferencia absoluta
0.08500692342254	0.08502020127843	0.00001327785588
0.17517447743525	0.17519989591111	0.00002541847586
0.27567167932995	0.27570736523387	0.00003568590392
0.39168423600472	0.39172789095656	0.00004365495184
0.52842268011133	0.52847111479709	0.00004843468575
0.69113025815209	0.69117918629612	0.00004892814403
0.88509049171057	0.88513530427695	0.00004481256638
1.11563433626716	1.11566987769150	0.00003554142433
1.38814686607076	1.38816750965764	0.00002064358687

>> [W0,W1,W] = econdasfriccion(0,10,1,0.001,'H','f','fpp','g','h1','h2',1000,1,0,1,0);

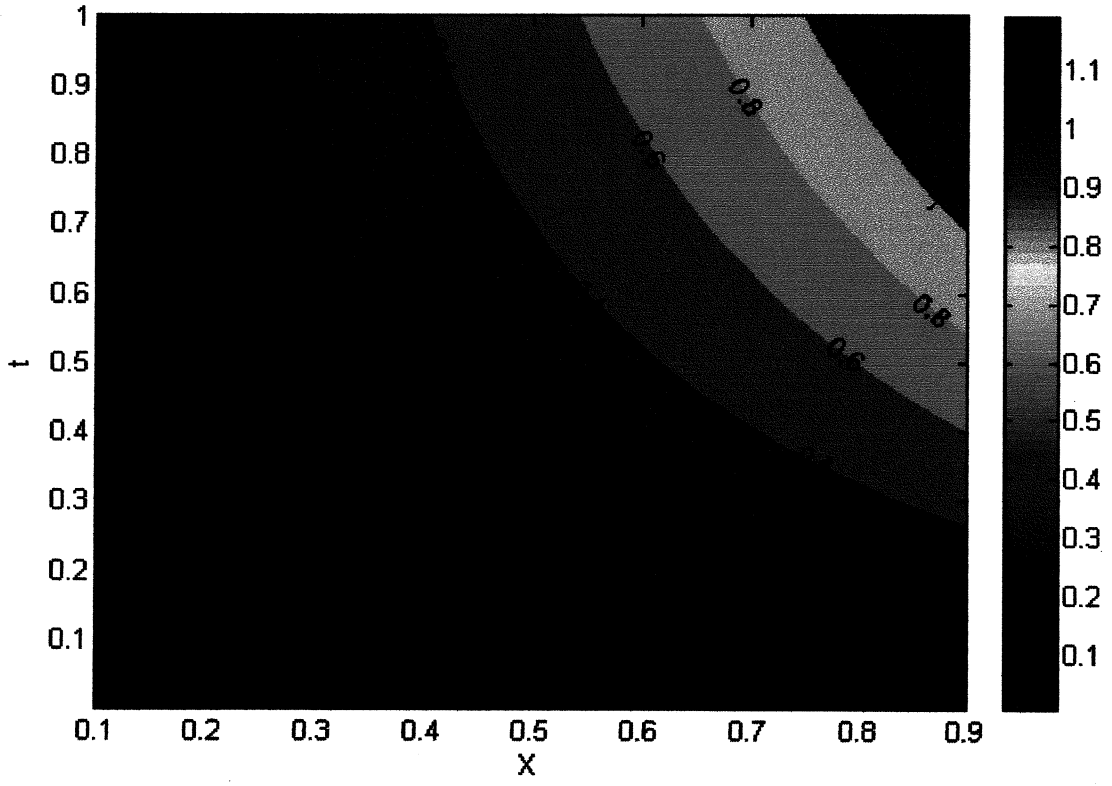
Comparativa para k=0.001, en t=1 segundo.

Solución real t=1	Solución aproximada	Diferencia absoluta
0.08500692342254	0.08501975963338	0.00001283621083
0.17517447743525	0.17519947534332	0.00002499790807
0.27567167932995	0.27570728036553	0.00003560103558
0.39168423600472	0.39172787977753	0.00004364377281
0.52842268011133	0.52847098814116	0.00004830802982
0.69113025815209	0.69117920540825	0.00004894725616
0.88509049171057	0.88513537714207	0.00004488543150
1.11563433626716	1.11566993366077	0.00003559739360
1.38814686607076	1.38816758412903	0.00002071805826

En este caso ambasa aproximaciones son igual de buenas. Veamos un gráfico de la segunda aproximación:



Y las curvas de nivel:



- ***Ejemplo 3.6***

Tenemos el siguiente planteamiento:

$$U_{tt} = U_{xx} - (6xt + A\,sen(t)\cos(x) + 3Ax^2t + B\sin(x)\sin(t) + Bx^3t + C\sin(x)\cos(t) + Cx^3);$$

En este tercer ejemplo vamos a tomar A=B=0 y C=1.

Con las siguientes condiciones de contorno:

$$u(0,t) = 0$$
$$u(1,t) = sen(1)\cdot sen(t) + t$$

Y las siguientes condiciones iniciales:

$$u(x,0) = 0$$
$$u_t(x,0) = \text{s}en(x) + x^3$$

Siendo la solución:

$$u(x,t) = sen(x)\cdot sen(t) + x^3 \cdot t$$

>> [W0,W1,W] = econdasfriccion(0,10,1,0.01,'H','f','fpp','g','h1','h2',50,1,0,0,1);

Comparativa para k=0.01, en t=0.5 segundos.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.04836268954660	0.04836539124942	0.00000270170281

0.09924715092056	0.09925252637591	0.00000537545534
0.15517993424704	0.15518792069152	0.00000798644448
0.21869709850368	0.21870755366131	0.00001045515763
0.29234884706593	0.29236136940367	0.00001252233774
0.37870402192622	0.37871759730203	0.00001357537580
0.48035441168228	0.48036743296397	0.00001302128169
0.59991883025051	0.59992981571136	0.00001098546085
0.74004692555132	0.74005389167782	0.00000696612649

>> [W0,W1,W] = econdasfriccion(0,10,1,0.001,'H','f','fpp','g','h1','h2',500,1,0,0,1);

Comparativa para k=0.001, en t=0.5 segundos.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.04836268954660	0.04836462131144	0.00000193176483
0.09924715092056	0.09925099509316	0.00000384417260
0.15517993424704	0.15518565176279	0.00000571751575
0.21869709850368	0.21870462622079	0.00000752771710
0.29234884706593	0.29235806808104	0.00000922101510
0.37870402192622	0.37871463548675	0.00001061356052
0.48035441168228	0.48036559790204	0.00001118621975
0.59991883025051	0.59992889033848	0.00001006008796
0.74004692555132	0.74005341607496	0.00000649052364

Se observa que la aproximación con k=0.001 es ligeramente mejor que la aproximación con k=0.01.

Ahora vamos a ver qué ocurre en t=1.

>> [W0,W1,W] = econdasfriccion(0,10,1,0.01,'H','f','fpp','g','h1','h2',100,1,0,0,1);

Comparativa para k=0.01, en t=1 segundo.

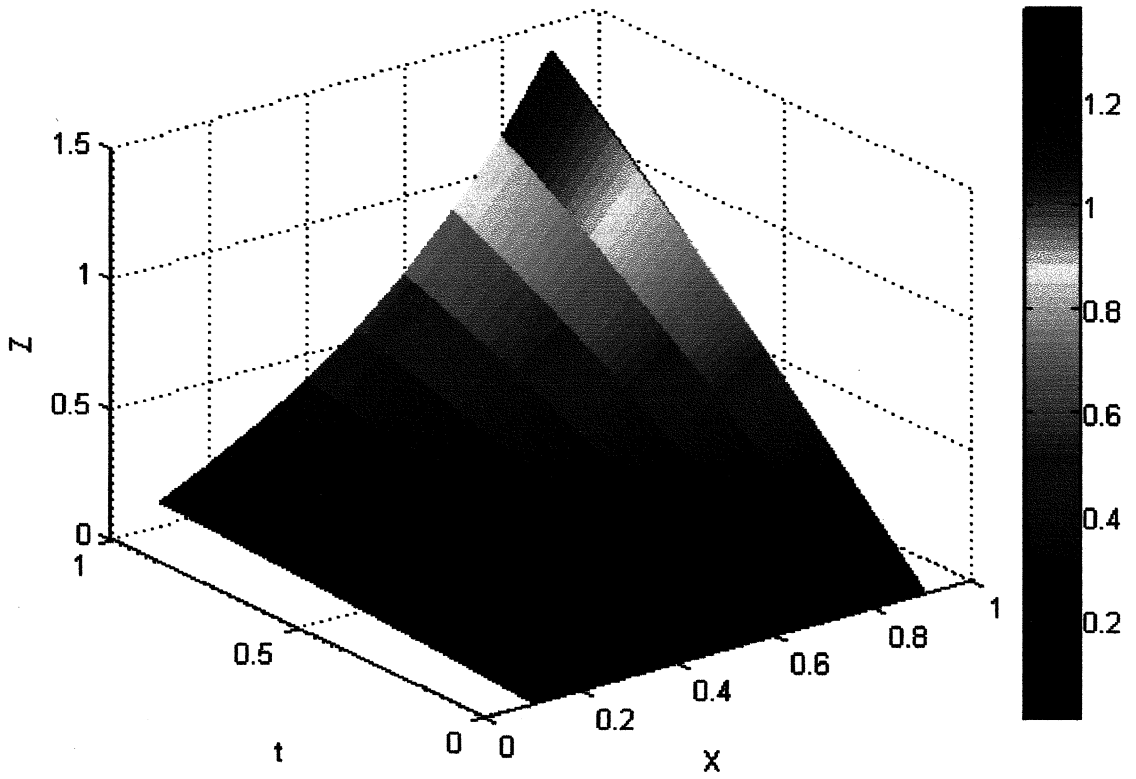
Solución real t=1	Solución aproximada	Diferencia absoluta
0.08500692342254	0.08502244289117	0.00001551946863
0.17517447743525	0.17520399284606	0.00002951541081
0.27567167932995	0.27571282861162	0.00004114928166
0.39168423600472	0.39173430903321	0.00005007302849
0.52842268011133	0.52847786315698	0.00005518304564
0.69113025815209	0.69118542839042	0.00005517023833
0.88509049171057	0.88514050986262	0.00005001815205
1.11563433626716	1.11567359263494	0.00003925636777
1.38814686607076	1.38816940864500	0.00002254257423

>> [W0,W1,W] = econdasfriccion(0,10,1,0.001,'H','f','fpp','g','h1','h2',1000,1,0,0,1);

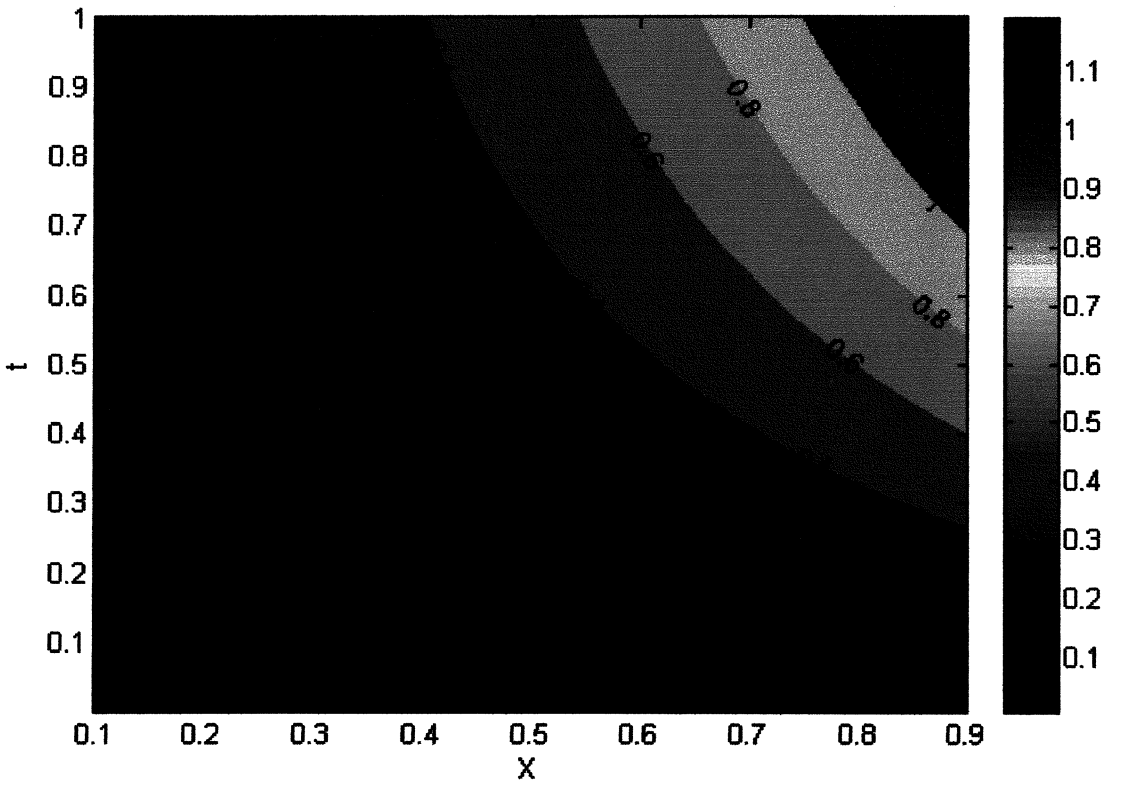
Comparativa para k=0.001, en t=1 segundo.

Solución real t=1	Solución aproximada	Diferencia absoluta
0.08500692342254	0.08502279225548	0.00001586883294
0.17517447743525	0.17520525308490	0.00003077564965
0.27567167932995	0.27571517685248	0.00004349752253
0.39168423600472	0.39173698753626	0.00005275153154
0.52842268011133	0.52848034212281	0.00005766201147
0.69113025815209	0.69118792595910	0.00005766780701
0.88509049171057	0.88514266124345	0.00005216953288
1.11563433626716	1.11567514532681	0.00004080905964
1.38814686607076	1.38817029841775	0.00002343234698

En este caso la primera aproximación es ligeramente mejor que la segunda. Veamos un gráfico de la primera aproximación:



Y las curvas de nivel:



• *Ejemplo 3.7*

Tenemos el siguiente planteamiento:

$$U_{tt} = U_{xx}\text{-(6xt+Asen(t)cos(x)+3Ax}^2\text{t+Bsin(x)sin(t)+Bx}^3\text{t+Csin(x)cos(t)+Cx}^3\text{);}$$

En este tercer ejemplo vamos a tomar A=B=C=1.

Con las siguientes condiciones de contorno:

$$u(0,t) = 0$$
$$u(1,t) = sen(1)\cdot sen(t) + t$$

Y las siguientes condiciones iniciales:

$$u(x,0) = 0$$
$$u_t(x,0) = sen(x) + x^3$$

Siendo la solución:

$$u(x,t) = sen(x)\cdot sen(t) + x^3 \cdot t$$

>> [W0,W1,W] = econdasfriccion(0,10,1,0.01,'H','f','fpp','g','h1','h2',50,1,1,1,1);

Comparativa para k=0.01, en t=0.5 segundos.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.04836268954660	0.04846684391797	0.00010415437136
0.09924715092056	0.09941381498759	0.00016666406703
0.15517993424704	0.15537885625010	0.00019892200306
0.21869709850368	0.21890981435970	0.00021271585602
0.29234884706593	0.29256699626575	0.00021814919982
0.37870402192622	0.37892159518023	0.00021757325400
0.48035441168228	0.48055918999950	0.00020477831721
0.59991883025051	0.60008772031570	0.00016889006518
0.74004692555132	0.74014878250434	0.00010185695301

>> [W0,W1,W] = econdasfriccion(0,10,1,0.001,'H','f','fpp','g','h1','h2',500,1,1,1,1);

Comparativa para k=0.001, en t=0.5 segundos.

Solución real t=0.5	Solución aproximada	Diferencia absoluta
0.04836268954660	0.04846586236427	0.00010317281766
0.09924715092056	0.09941194158041	0.00016479065985
0.15517993424704	0.15537617589239	0.00019624164534
0.21869709850368	0.21890647228701	0.00020937378332
0.29234884706593	0.29256335298832	0.00021450592238
0.37870402192622	0.37891847616459	0.00021445423836
0.48035441168228	0.48055742795266	0.00020301627037
0.59991883025051	0.60008695094093	0.00016812069042
0.74004692555132	0.74014840712824	0.00010148157691

Se observa que la aproximación con k=0.001 es ligeramente mejor que la aproximación con k=0.01.

Ahora vamos a ver qué ocurre en t=1.

>> [W0,W1,W] = econdasfriccion(0,10,1,0.01,'H','f','fpp','g','h1','h2',100,1,1,1,1);

Comparativa para k=0.01, en t=1 segundo.

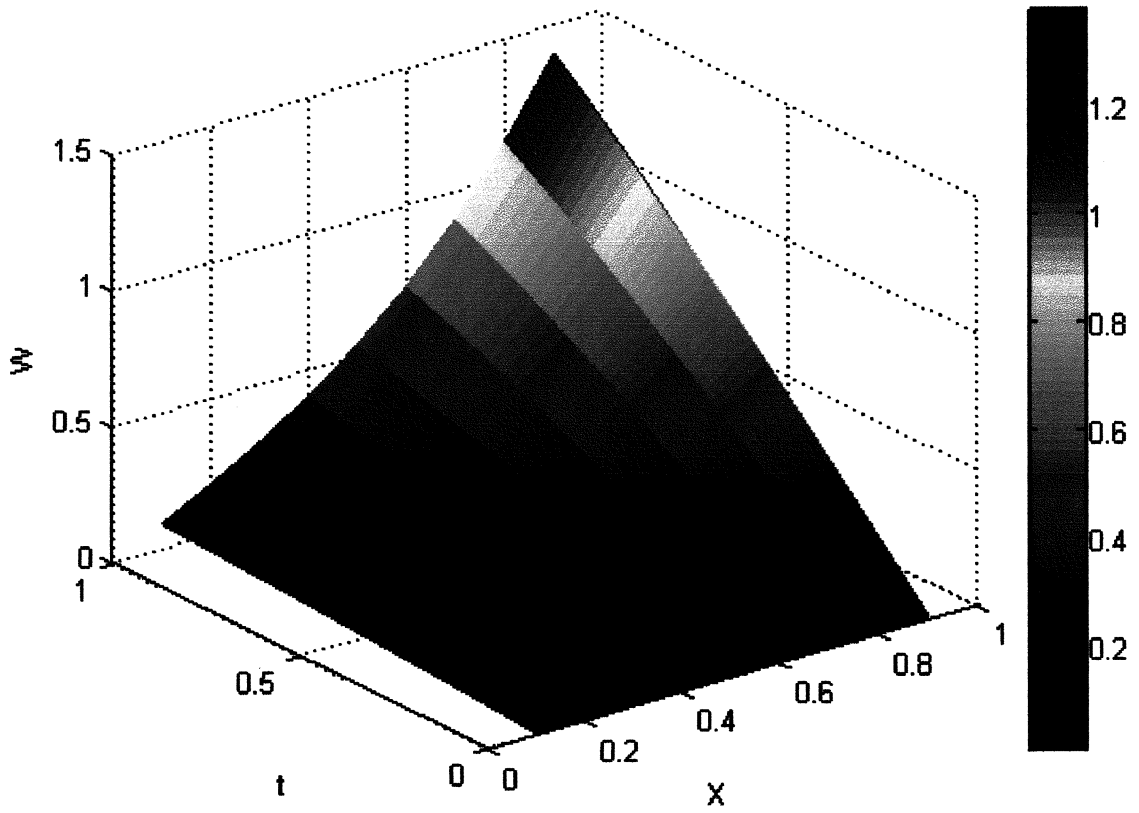
Solución real t=1	Solución aproximada	Diferencia absoluta
0.08500692342254	0.08559443201256	0.00058750859002
0.17517447743525	0.17620162388555	0.00102714645030
0.27567167932995	0.27699245420970	0.00132077487975
0.39168423600472	0.39315440379770	0.00147016779298
0.52842268011133	0.52990694486070	0.00148426474937
0.69113025815209	0.69250722602089	0.00137696786880
0.88509049171057	0.88624999938001	0.00115950766944
1.11563433626716	1.11648061083346	0.00084627456630
1.38814686607076	1.38860154399826	0.00045467792749

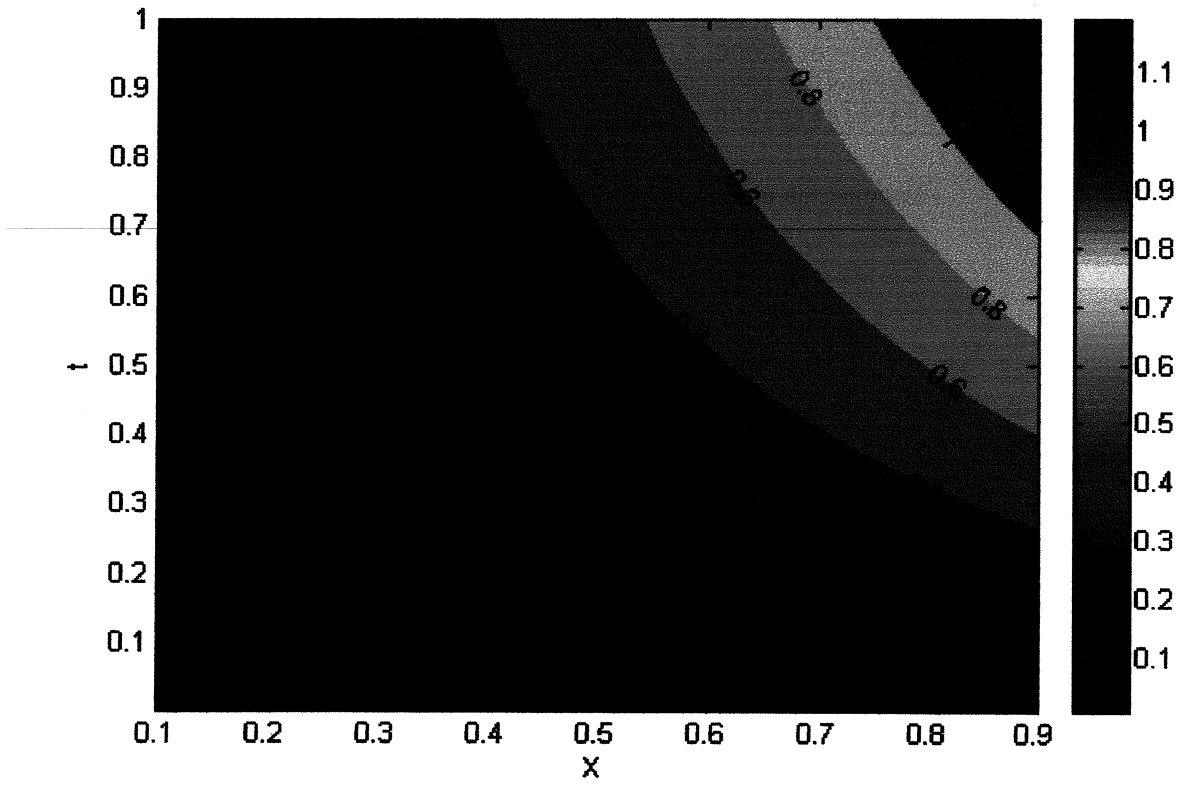
>> [W0,W1,W] = econdasfriccion(0,10,1,0.001,'H','f','fpp','g','h1','h2',1000,1,1,1,1);

Comparativa para k=0.001, en t=1 segundo.

Solución real t=1	Solución aproximada	Diferencia absoluta
0.08500692342254	0.08559442011481	0.00058749669227
0.17517447743525	0.17620253265321	0.00102805521797
0.27567167932995	0.27699463223498	0.00132295290503
0.39168423600472	0.39315685628741	0.00147262028269
0.52842268011133	0.52990902934599	0.00148634923465
0.69113025815209	0.69250930794827	0.00137904979618
0.88509049171057	0.88625173873814	0.00116124702757
1.11563433626716	1.11648180096900	0.00084746470183
1.38814686607076	1.38860222263344	0.00045535656268

En este caso las aproximaciones son igual de buenas. Veamos un gráfico de la segunda aproximación:





- ***Ejemplo 3.8***

En los ejemplos anteriores hemos resuelto el mismo problema pero con distintos coeficientes A, B y C. Vamos a comparar la diferencia que hay en el resultado con cada coeficiente:
Lo que vamos a comparar es la precisión de la aproximación en cada caso.

Para t=0.5, las diferencias entre la solución real y la aproximada eran:

A=B=C=0	A=1,B=C=0	A=C=0,B=1
0.0000024698729	0.00009147400277	0.00000171926802
0.0000049143007	0.00014439208685	0.00000342131298
0.0000073023235	0.00017075446458	0.00000508864390
0.0000095666120	0.00018167410513	0.00000670024108
0.0000114910469	0.00018603468176	0.00000821073230
0.0000125538103	0.00018620045910	0.00000946627293
0.0000121950096	0.00017700056192	0.00001002480199
0.0000103936668	0.00014794570809	0.00000910275246
0.0000066442032	0.00009060918657	0.00000595196207

A=B=0,C=1	A=B=C=1
0.00000193176483	0.00010317281766
0.00000384417260	0.00016479065985
0.00000571751575	0.00019624164534

4. Ecuación del calor: calentamiento de una plancha

4.1 Planteamiento del problema. Ecuación.

Tenemos una plancha cuadrada y queremos calcular como va cambiando la temperatura en el interior a lo largo del tiempo. Sólo conocemos la temperatura del contorno de la plancha.

El planteamiento es el siguiente:

$$\frac{\partial u}{\partial t} = \alpha^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + H(t,x,y)$$
$$0 \le x \le L$$
$$0 \le y \le L$$
$$t > 0$$

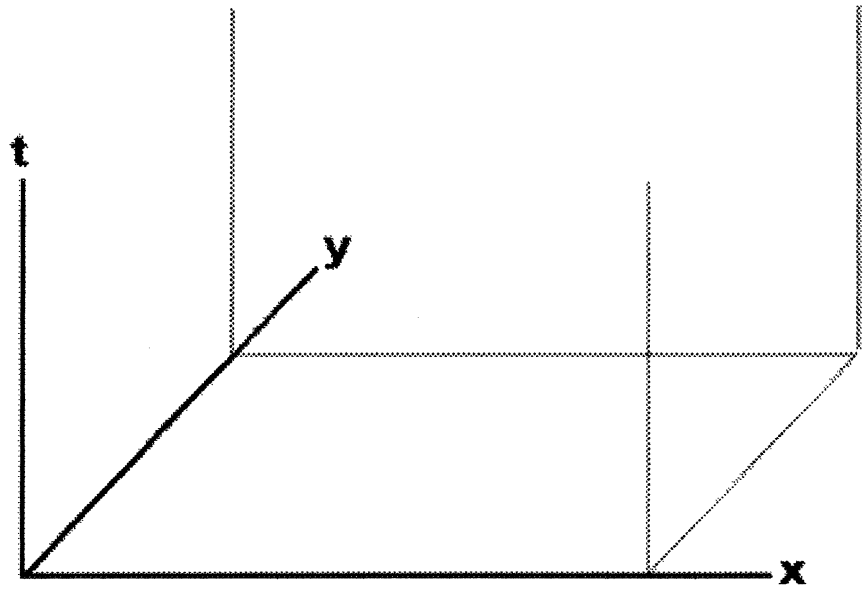
Con las siguientes condiciones de contorno:

$$u(t,0,y) = h_1(t,y)$$
$$u(t,L,y) = h_2(t,y)$$
$$u(t,x,0) = h_3(t,x)$$
$$u(t,x,L) = h_4(t,x)$$

Y la condición inicial:

$$u(0,x,y) = f(x,y)$$

Gráficamente tendríamos la siguiente situación:



4.2 Discretización

El primer paso es discretizar el problema. Para ello nos basamos en la primera discretización de la primera derivada y en la discretización de la segunda derivada (ver apéndice). Por tanto, el problema discretizado quedaría:

$$\frac{u(t+k,x,y)-u(t,x,y)}{k}+O(k)=$$
$$=\alpha^2\left[\frac{u(t,x+h,y)-2u(t,x,y)+u(t,x-h,y)}{h^2}+\frac{u(t,x,y+h)-2u(t,x,y)+u(t,x,y-h)}{h^2}+O(h^2)\right]$$

Si despreciamos O(k) y $O(h^2)$, y además tomamos $x=x_i$, $y=y_j$, $t=t_l$, $\lambda=k\alpha^2/h^2$ y $W_{ij}^l \approx u(t_l,x_i,y_j)$, tenemos un método progresivo para calcular la temperatura en un punto t una vez conocida la temperatura en t-1:

$$\begin{cases} W_{i,j}^{l+1}=\lambda W_{i+1,j}^{l}+(1-4\lambda)W_{i,j}^{l}+\lambda W_{i-1,j}^{l}+\lambda W_{i,j+1}^{l}+\lambda W_{i,j-1}^{l}+kH_{i,j}^{l} \\ W_{i,j}^{0}=\{f(x_i,y_j)\}_{\substack{i=1,\dots,n-1\\ j=1,\dots,n-1}} \end{cases}$$

donde n es el numero de divisiones que se harán.

Este método es estable si $\lambda \leq 1/4$.

A continuación implementaremos este método progresivo en MATLAB.

4.3 Algoritmo A4.1: Ecuación del calor en una plancha cuadrada / rectangular.

Método progresivo. Es estable si lambda es menor o igual que ¼.

Datos que recibe el algoritmo:

- x0: valor inicial de x
- y0: valor inicial de y
- nx: número de divisiones que se tomarán en x
- ny: número de divisiones que se tomarán en y
- Lx: valor final de x
- Ly: valor final de y
- k: valor del paso en t
- f: condición inicial
- H: función H(t,x,y)
- h1: condición de contorno cuando x=0
- h2: condición de contorno cuando x=Lx
- h3: condición de contorno cuando y=0

- h4: condición de contorno cuando y=Ly
- p: valor final de t
- alpha: valor de α

Datos que devuelve el algoritmo:

- W: matriz de p filas, donde cada fila es una matriz de orden (nx-1)*(ny-1). El valor W(i,j,p) representará a $W_{i,j,p-1}$

```
function W =
metodoprogresivo(x0,y0,nx,ny,Lx,Ly,k,f,H,h1,h2,h3,h4,p,alpha)
hx=(Lx-x0)/nx;
hy=(Ly-y0)/ny;
lambda=alpha^2*k/(hx*hy);
W=zeros(nx-1,ny-1,p);
W0=zeros(nx-1,ny-1);
for j=1:ny-1
    y=y0+j*hy;
    for i=1:nx-1
        x=x0+i*hx;
        W0(i,j)=feval(f,x,y);
    end
end
W(:,:,1)=W0;

for l=1:p
    WL=zeros(nx-1,ny-1);
    t=l*k;
    for j=1:ny-1
        y=y0+j*hy;
        for i=1:nx-1
            x=x0+i*hx;
            if j==1
                if i==1
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h1,
y,t)+lambda*feval(h3,x,t);
                else if i==nx-1
                    WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h2,y,t)+lambda*
feval(h3,x,t);
```

```
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h3,x,t);
                    end
                end
            else if j==ny-1
                    if i==1
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t)+lambda*feval(h4,x,t);
                    else if i==nx-1
                        WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h2,y,t)+lambda*feval(h4,x,t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h4,x,t);
                        end
                    end
                else
                    if i==1
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                    else if i==nx-1
                        WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h2,y,t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                        end
                    end
                end
            end
        end
    end
    W(:,:,l+1)=WL;
```

```
end
```

4.4 Ejemplos

• *Ejemplo 4.1*

El planteamiento es el siguiente:

$$U_t=U_{xx}+U_{yy}+(\cos(t)-2),$$

$$\text{con } x,y \in [0,1] \text{ y } t>0$$

Con las siguientes condiciones de contorno:

$$u(0,y,t) = \sin(t)$$

$$u(1,y,t) = \sin(t) + 1$$

$$u(x,0,t) = \sin(t) + x^2$$

$$u(x,1,t) = \sin(t) + x^2$$

Y la condición inicial:

$$u(x,y,0) = x^2$$

La solución de este ejemplo es:

$$u(x,y,t)=\sin(t) + x^2$$

Ahora llamaremos al algoritmo anterior (A4.1) para intentar aproximar la solución de ese ejemplo. Los datos que pasaremos son:

x0=0, y0=0, nx=10, ny=10, Lx=1, Ly=1, k=0.001, p=200, alpha=1

Los archivos f.m, H.m, h1.m, h2.m, h3.m, y h4.m son, respectivamente:

f.m

```
function z=f(x,y)
z=x^2;
```

H.m

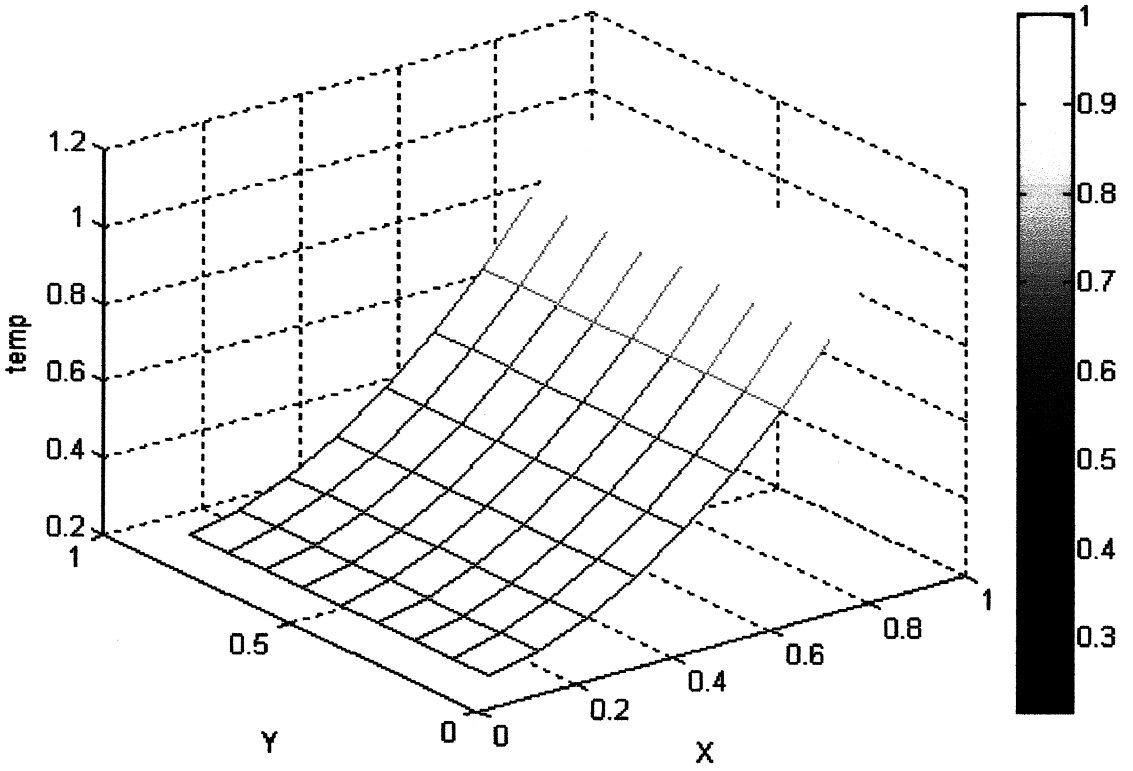
```
function z=H(t,x,y)
z=cos(t)-2;
```

h1.m

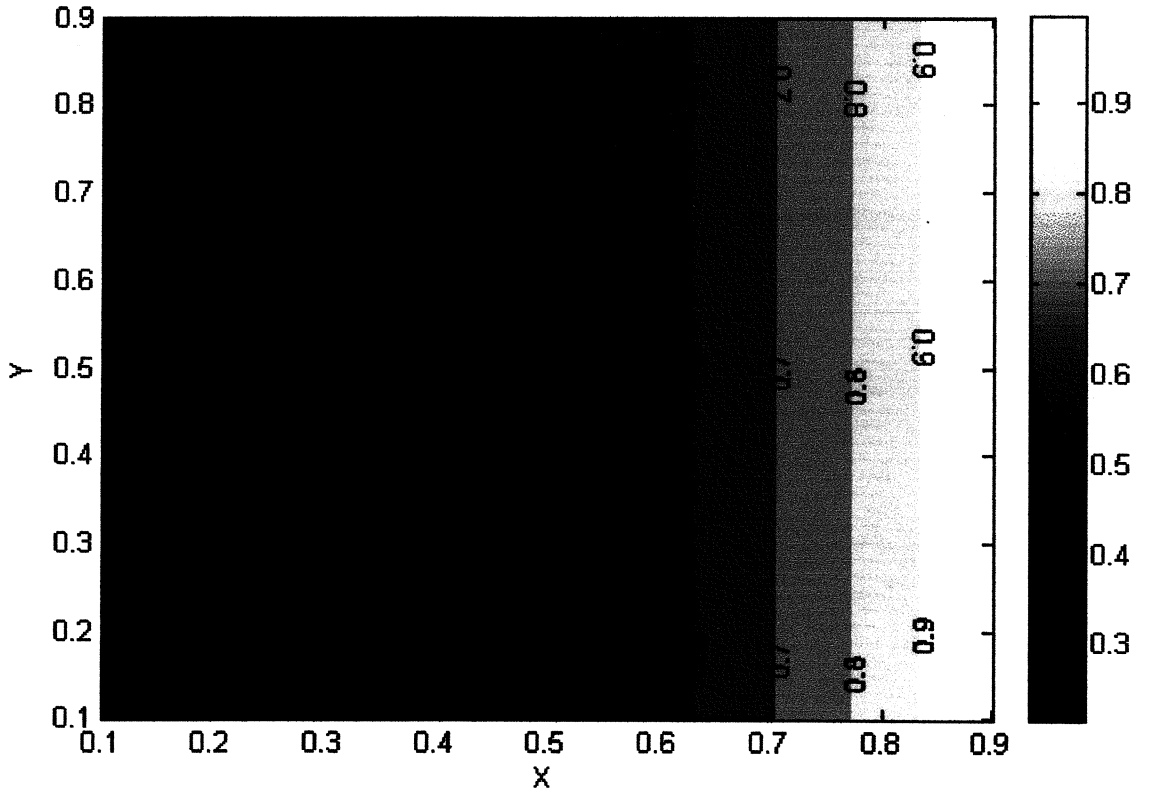
```
function z=h1(y,t)
z=sin(t);
```

h2.m

```
function z=h2(y,t)
z=sin(t)+1;
```



Isotermas correspondientes a la estimación $W_{i,j,200}$ (t=0.2 segundos):



Ahora vamos a calcular el error cometido en la aproximación.

Siendo u.m

```
function Z=u(X,Y,t)
```

```
nx=length(X);
ny=length(Y);
Z=zeros(nx,ny);
for j=1:ny
    for i=1:nx

Z(i,j)=sin(t)+X(i)^2;
    end
end
```

y

V=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]

haremos la llamada a u.m para calcular el valor de la solución en t=200*0.001=0.2, que es el último valor que hemos calculado con el método progresivo.

>> Z=u(V,V,0.2)

```
Z =
  Columns 1 through 7
    0.2087    0.2087    0.2087    0.2087    0.2087    0.2087    0.2087
    0.2387    0.2387    0.2387    0.2387    0.2387    0.2387    0.2387
    0.2887    0.2887    0.2887    0.2887    0.2887    0.2887    0.2887
    0.3587    0.3587    0.3587    0.3587    0.3587    0.3587    0.3587
    0.4487    0.4487    0.4487    0.4487    0.4487    0.4487    0.4487
    0.5587    0.5587    0.5587    0.5587    0.5587    0.5587    0.5587
    0.6887    0.6887    0.6887    0.6887    0.6887    0.6887    0.6887
    0.8387    0.8387    0.8387    0.8387    0.8387    0.8387    0.8387
    1.0087    1.0087    1.0087    1.0087    1.0087    1.0087    1.0087

  Columns 8 through 9
    0.2087    0.2087
    0.2387    0.2387
    0.2887    0.2887
    0.3587    0.3587
    0.4487    0.4487
    0.5587    0.5587
    0.6887    0.6887
    0.8387    0.8387
    1.0087    1.0087
```

Ahora calcularemos el valor absoluto de la diferencia entre el valor real y el estimado, para t=0.2

>> N=abs(Z-W(:,:,201))

```
N =
  1.0e-003 *
  Columns 1 through 7
    0.9781    0.9761    0.9743    0.9732    0.9728    0.9732    0.9743
    0.9761    0.9721    0.9687    0.9665    0.9657    0.9665    0.9687
    0.9743    0.9687    0.9641    0.9610    0.9599    0.9610    0.9641
    0.9732    0.9665    0.9610    0.9573    0.9561    0.9573    0.9610
    0.9728    0.9657    0.9599    0.9561    0.9547    0.9561    0.9599
    0.9732    0.9665    0.9610    0.9573    0.9561    0.9573    0.9610
    0.9743    0.9687    0.9641    0.9610    0.9599    0.9610    0.9641
    0.9761    0.9721    0.9687    0.9665    0.9657    0.9665    0.9687
    0.9781    0.9761    0.9743    0.9732    0.9728    0.9732    0.9743

  Columns 8 through 9
    0.9761    0.9781
    0.9721    0.9761
    0.9687    0.9743
    0.9665    0.9732
    0.9657    0.9728
    0.9665    0.9732
    0.9687    0.9743
    0.9721    0.9761
    0.9761    0.9781
```

>> norm(N,inf)

```
ans =
    0.0088
```

Por los datos obtenidos se observa que el error cometido es pequeño, por tanto la aproximación es bastante buena.

- ***Ejemplo 4.2***

El planteamiento es el siguiente:

$$U_t=U_{xx}+U_{yy}+\cos(2\pi x)\sin(2\pi y)[\cos(t)+8\pi^2\sin(t)],$$

$$\text{con } x,y \in [0,1] \text{ y } t>0$$

Con las siguientes condiciones de contorno:

$$u(0,y,t) = \sin(t)\sin(2\pi y)$$

$$u(1,y,t) = \sin(t)\sin(2\pi y)$$

$$u(x,0,t) = 0$$

$$u(x,1,t) = 0$$

Y la condición inicial:

$$u(x,y,0) = 0$$

La solución de este ejemplo es:

$$u(x,y,t)=\sin(t)\cos(2\pi x)\sin(2\pi y)$$

Ahora llamaremos al algoritmo A4.1 para intentar aproximar la solución de ese ejemplo. Los datos que pasaremos son:

x0=0, y0=0, nx=10, ny=10, Lx=1, Ly=1, k=0.001, p=200, alpha=1

Los archivos f.m, H.m, h1.m, h2.m, h3.m, y h4.m son, respectivamente:

f.m

```
function z=f(x,y)
z=0;
```

H.m

```
function z=H(t,x,y)
z= cos(2*pi*x)*sin(2*pi*y)*(cos(t)+8*(pi^2)*sin(t));
```

h1.m

```
function z=h1(y,t)
z=sin(t)*sin(2*pi*y);
```

h2.m

```
function z=h2(y,t)
z=sin(t)*sin(2*pi*y);
```

h3.m

```
function z=h3(x,t)
z=0;
```

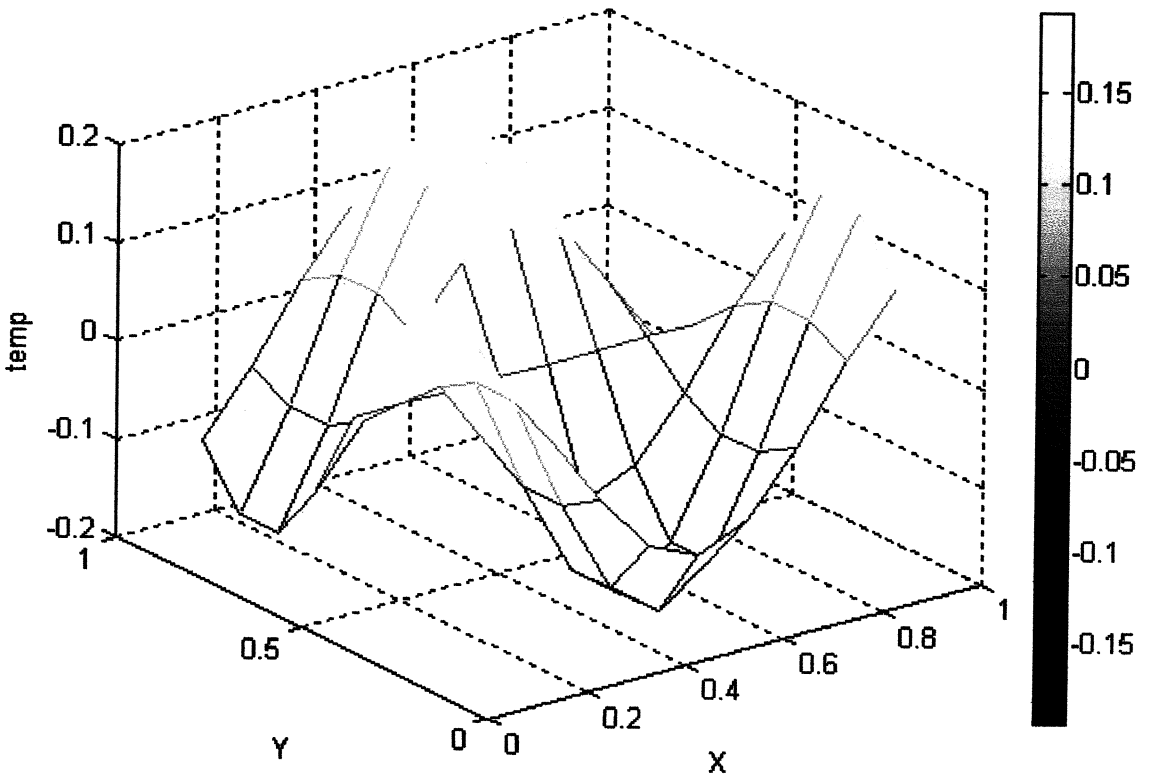
h4.m

```
function z=h4(x,t)
z=0;
```

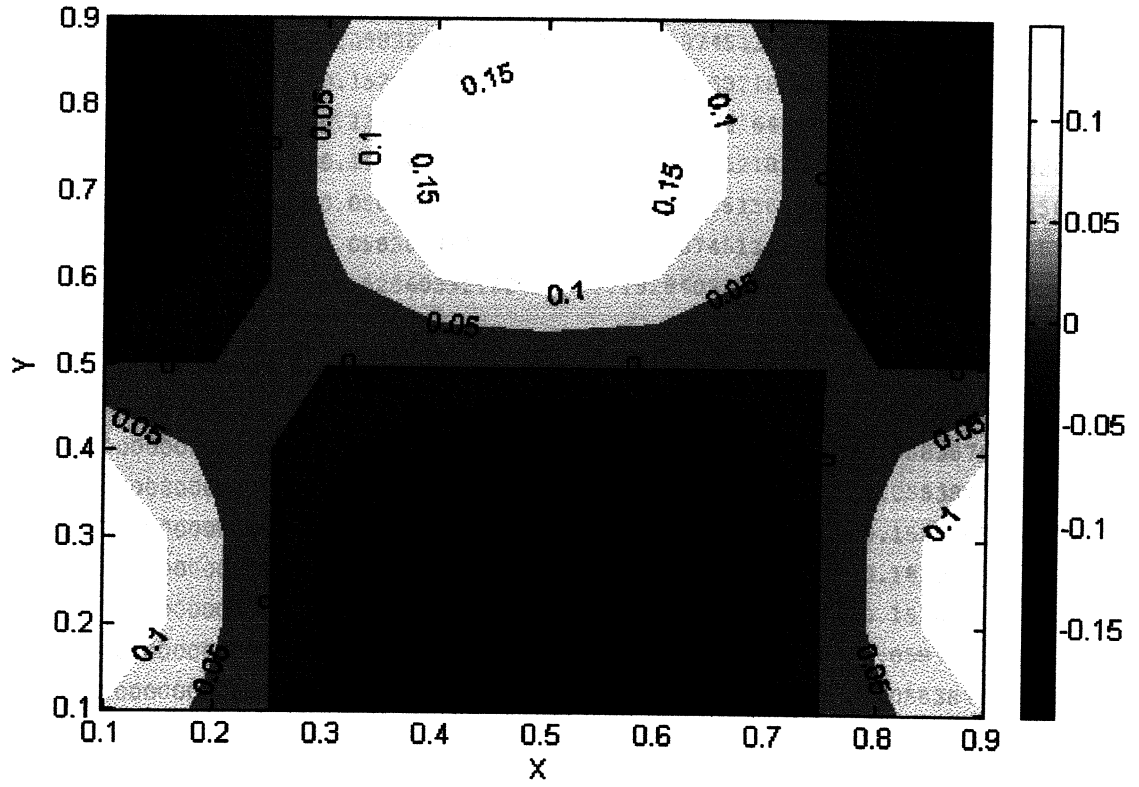
>> W=metodoprogresivo(0,0,10,10,1,1,0.001,'f','H','h1','h2','h3','h4',200,1)

```
...
W(:,:,201) =
  Columns 1 through 4
   0.09598954747058   0.15531435037212   0.15531435037212   0.09598954747058
   0.03638232652231   0.05886784090289   0.05886784090289   0.03638232652231
  -0.03795973041151  -0.06142013400961  -0.06142013400961  -0.03795973041151
  -0.09826238317298  -0.15899187578945  -0.15899187578945  -0.09826238317298
  -0.12131557634946  -0.19629272589821  -0.19629272589821  -0.12131557634946
  -0.09826238317298  -0.15899187578945  -0.15899187578945  -0.09826238317298
  -0.03795973041151  -0.06142013400961  -0.06142013400961  -0.03795973041151
   0.03638232652231   0.05886784090289   0.05886784090289   0.03638232652231
   0.09598954747058   0.15531435037212   0.15531435037212   0.09598954747058

  Columns 5 through 8
   0.00000000000000  -0.09598954747058  -0.15531435037212  -0.15531435037212
   0.00000000000000  -0.03638232652231  -0.05886784090289  -0.05886784090289
   0.00000000000000   0.03795973041151   0.06142013400961   0.06142013400961
  -0.00000000000000   0.09826238317298   0.15899187578945   0.15899187578945
  -0.00000000000000   0.12131557634946   0.19629272589821   0.19629272589821
  -0.00000000000000   0.09826238317298   0.15899187578945   0.15899187578945
  -0.00000000000000   0.03795973041151   0.06142013400961   0.06142013400961
   0.00000000000000  -0.03638232652231  -0.05886784090289  -0.05886784090289
   0.00000000000000  -0.09598954747058  -0.15531435037212  -0.15531435037212

  Column 9
  -0.09598954747058
  -0.03638232652231
   0.03795973041151
   0.09826238317298
   0.12131557634946
   0.09826238317298
   0.03795973041151
  -0.03638232652231
  -0.09598954747058
```

Gráfico correspondiente a la estimación $W_{i,j,200}$ (t=0.2 segundos):



Isotermas correspondientes a la estimación $W_{i,j,200}$ (t=0.2 segundos):



Ahora vamos a calcular el error cometido en la aproximación.

Siendo u.m

```
function Z=u(X,Y,t)
```

>> norm(N,inf)

```
ans =
    0.04755055153153
```

Por los datos obtenidos se observa que el error cometido es pequeño, por tanto la aproximación es bastante buena.

- ***Ejemplo 4.3***

El planteamiento es el siguiente:

$$u_t = u_{xx} + u_{yy} + 2\pi^2 sen(\pi x)sen(\pi t)\cos(\pi y) + \pi\cos(\pi t)sen(\pi x)\cos(\pi y),$$

con x,y ∈ [0,1] y t>0

Con las siguientes condiciones de contorno:

$$u(0,y,t) = 0$$

$$u(1,y,t) = \sin(\pi)\sin(\pi t)\cos(\pi y)$$

$$u(x,0,t) = \sin(\pi t)\sin(\pi x)$$

$$u(x,1,t) = -\sin(\pi t)\sin(\pi x)$$

Y la condición inicial:

$$u(x,y,0) = 0$$

La solución de este ejemplo es:

$$u(x,y,t)=\sin(\pi t)\sin(\pi x)\cos(\pi y)$$

Ahora llamaremos al algoritmo A4.1 para intentar aproximar la solución de ese ejemplo. Los datos que pasaremos son:

x0=0, y0=0, nx=10, ny=10, Lx=1, Ly=1, k=0.001, p=200, alpha=1

Los archivos f.m, H.m, h1.m, h2.m, h3.m, y h4.m son, respectivamente:

f.m

```
function z=f(x,y)
z=0;
```

H.m

```
function z=H(t,x,y)
z=2*pi^2*sin(pi*x)*sin(pi*t)*cos(pi*y)+pi*cos(pi*t)
*sin(pi*x)*cos(pi*y);
```

h1.m

```
function z=h1(y,t)
```

```
z=0;
```

h2.m

```
function z=h2(y,t)
z=sin(pi)*sin(pi*t)*cos(pi*y);
```

h3.m

```
function z=h3(x,t)
z=sin(pi*t)*sin(pi*x);
```

h4.m

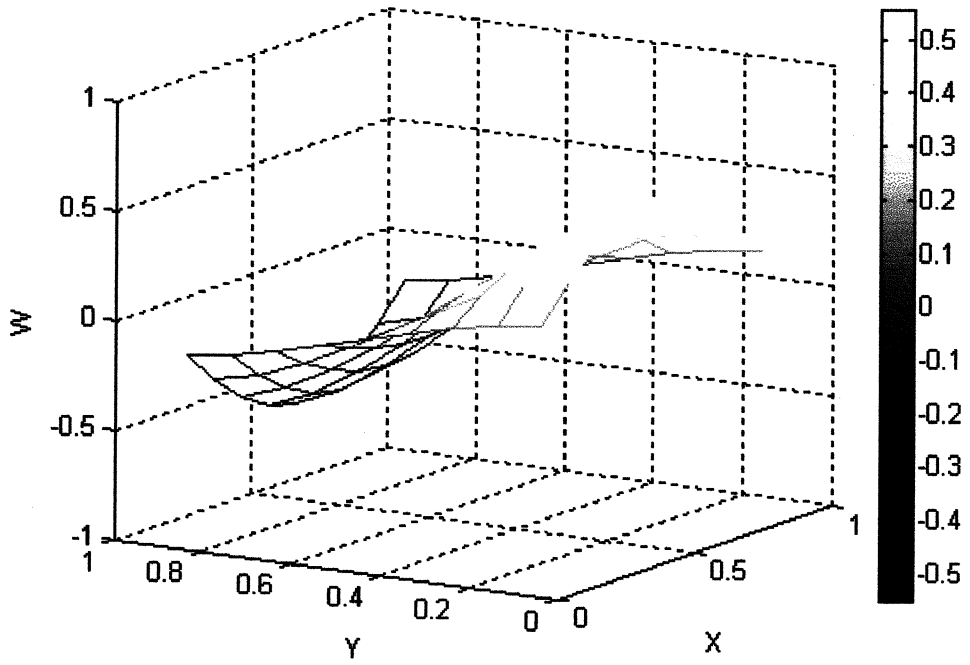
```
function z=h4(x,t)
z=sin(pi*t)*sin(pi*x);
```

>> W=metodoprogresivo(0,0,10,10,1,1,0.001,'f','H','h1','h2','h3','h4',200,1)

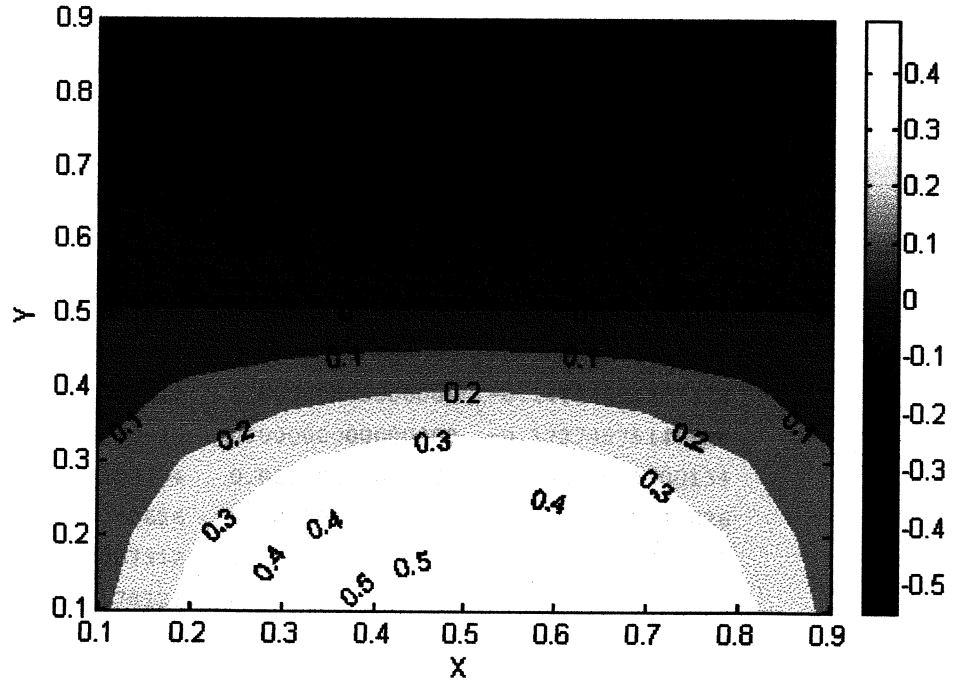
```
...
W(:,:,201) =
    Columns 1 through 3
   0.17470603958132   0.14968783166036   0.10988674780756
   0.33231170836509   0.28472521753977   0.20901982584846
   0.45739012898661   0.39189501968802   0.28769713506814
   0.53769775429660   0.46070671142937   0.33821719261336
   0.56537285095175   0.48442326299560   0.35563300634448
   0.53770531397982   0.46072109080079   0.33823698411923
   0.45740236081101   0.39191828599971   0.28772915839735
   0.33232394018950   0.28474848385146   0.20905184917767
   0.17471359926455   0.14970221103178   0.10990653931345

  Columns 4 through 6
   0.05925183395377   0.00278099799887  -0.05396206128736
   0.11270698978507   0.00529324679627  -0.10263863582341
   0.15513494674492   0.00729297742754  -0.14126287899515
   0.18238256120485   0.00858444275845  -0.16605398035828
   0.19178060495243   0.00903907822162  -0.17458725604542
   0.18240582751448   0.00860890640375  -0.16603071404865
   0.15517259242470   0.00733256043714  -0.14122523331537
   0.11274463546485   0.00533282980587  -0.10260099014363
   0.05927510026341   0.00280546164419  -0.05393879497773

  Columns 7 through 9
  -0.10538699812527  -0.14641857189251  -0.17298728789790
  -0.20045517186492  -0.27850263169158  -0.32904030117782
  -0.27589684866953  -0.38332160909987  -0.45288282001553
```

-0.32432727123018	-0.45061509203745	-0.53239227567187
-0.34100746926519	-0.47379718747446	-0.55978639209409
-0.32430747972430	-0.45060071266603	-0.53238471598865
-0.27586482534033	-0.38329834278818	-0.45287058819113
-0.20042314853571	-0.27847936537988	-0.32902806935341
-0.10536720661939	-0.14640419252108	-0.17297972821467

Gráfico correspondiente a la estimación $W_{i,j,200}$ (t=0.2 segundos):



Isotermas correspondientes a la estimación $W_{i,j,200}$ (t=0.2 segundos):



```
   0.00364367184715   0.00141209935873   0.00061633959739
   0.00265166160985   0.00102913180759   0.00044612427895
   0.00139553916182   0.00054212055203   0.00023397680841
```

>> norm(N,inf)

```
ans =
    0.05860929402501
```

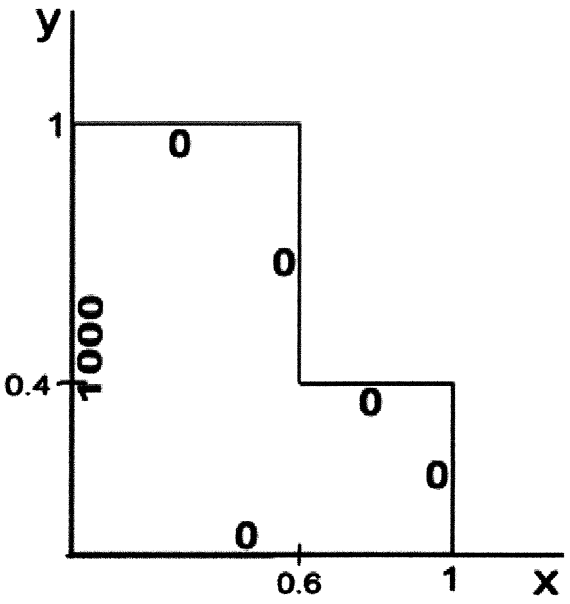
Por los datos obtenidos se observa que el error cometido es pequeño, por tanto la aproximación es bastante buena.

5. Aplicaciones personales

En este apartado se desarrollan una serie de problemas no incluidos en la teoría básica impartida en clase. Personalmente he decidido centrar mi trabajo en la ecuación del calor. Este apartado se compone de una serie de problemas planteados y resueltos.

PROBLEMA 1. CALENTAMIENTO DE UNA PLANCHA EN FORMA DE "L"

Tenemos una plancha con forma de L:



Los valores en rojo representan las temperaturas en el contorno de la plancha. Por tanto, tenemos el siguiente planteamiento:

$$U_t=U_{xx}+U_{yy},$$
$$\text{con } x,y \in [0,1] \text{ y } t>0$$

Con las siguientes condiciones de contorno:

$$u(0,y,t) = 1000$$
$$u(1,y,t) = 0,\ 0 < y < 0.4$$
$$u(0.6,y,t) = 0,\ 0.4 < y < 1$$
$$u(x,0,t) = 0$$
$$u(x,0.4,t) = 0,\ 0.6 < x < 1$$
$$u(x,1,t) = 0,\ 0 < x < 0.6$$

Y la condición inicial:

$$u(x,y,0) = 0$$

Se pide: Calcular, para un determinado t, cómo se distribuyen las isotermas.

Resolución:

Lo primero que hay que hacer es adaptar el algoritmo que hemos para calcular el calentamiento de una plancha cuadrada (algoritmo A4.1). Lo que voy a hacer es un algoritmo exclusivo para este tipo de problema, al que apenas se le pasan parámetros.

ALGORITMO [A5.1]

Método progresivo. Es estable si λ es menor o igual que ¼.

Datos que recibe el algoritmo:

- H: función H(t,x,y)
- h1: condición de contorno
- h2: condición de contorno
- h3: condición de contorno
- h4: condición de contorno
- h5: condición de contorno
- h6: condición de contorno
- p: valor final de t

Datos que devuelve el algoritmo:

- W: matriz de p filas, donde cada fila es una matriz de orden (nx-1)*(ny-1). El valor W(i,j,p) representará a $W_{i,j;p-1}$

```
function W = metodoprogresivo(f,H,h1,h2,h3,h4,h5,h6,p)
x0=0;
y0=0;
hx=0.1;
hy=0.1;
k=0.001;
lambda=k/(hx*hy);
W=zeros(9,9,p);
W0=zeros(9,9);
W(:,:,1)=W0;
for l=1:p
    WL=zeros(9,9);
    t=l*k;
    for j=1:3
        y=y0+j*hy;
        for i=1:9
```

```
            x=x0+i*hx;
            if j==1
                if i==1
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h1,
y,t)+lambda*feval(h4,x,t);
                else if i==9
                    WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h2,y,t)+lambda*
feval(h4,x,t);
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h4,x,t);
                    end
                end
            else
                if j==3
                    if i>6
                        if i==9
                            WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h2,y,t)+lambda*feval(h5,x,t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h5,x,t);
                        end
                    else
                        if i==1
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);

                        end
                    end
                else
```

```
                    if i==1
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                    else if i==9
                        WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h2,y,t);
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                    end
                end
            end
        end
    end
    for j=4:9
        y=y0+j*hy;
        for i=1:5
            x=x0+i*hx;
            if j==9
                if i==1
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t)+lambda*feval(h6,x,t);
                else if i==5
                    WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h3,y,t)+lambda*feval(h6,x,t);
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h6,x,t);
                    end
                end
            else
                if i==1
```

```
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                else if i==5
                    WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h3,y,t);
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                    end
                end
            end
        end
    end
    W(:,:,l+1)=WL;
end
```

Ahora hacemos la llamada al algoritmo. Pondremos p=200 para que nos calcule la temperatura en t=0,2 segundos.

Los archivos f.m, H.m, h1.m, h2.m, h3.m, h4.m, h5.m, y h6.m son, respectivamente:

f.m

```
function z=f(x,y)
z=0;
```

H.m

```
function z=H(t,x,y)
z=0;
```

h1.m

```
function z=h1(y,t)
z=1000;
```

h2.m

```
function z=h2(y,t)
z=0;
```

h3.m

```
function z=h3(x,t)
```

```
z=0;
```

h4.m

```
function z=h4(x,t)
z=0;
```

h5.m

```
function z=h5(x,t)
z=0;
```

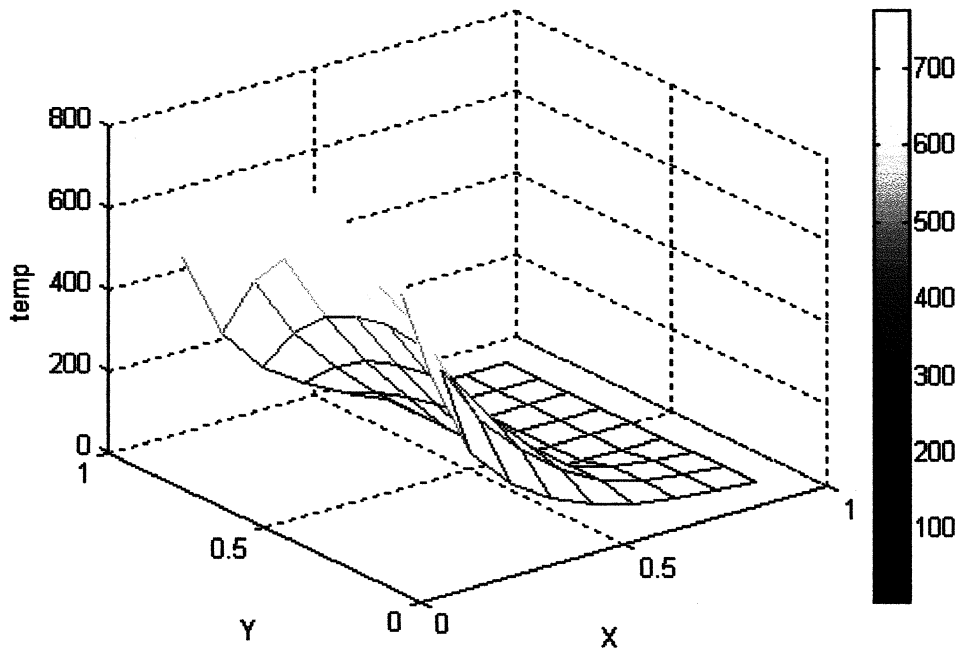
h6.m

```
function z=h6(x,t)
z=0;
```

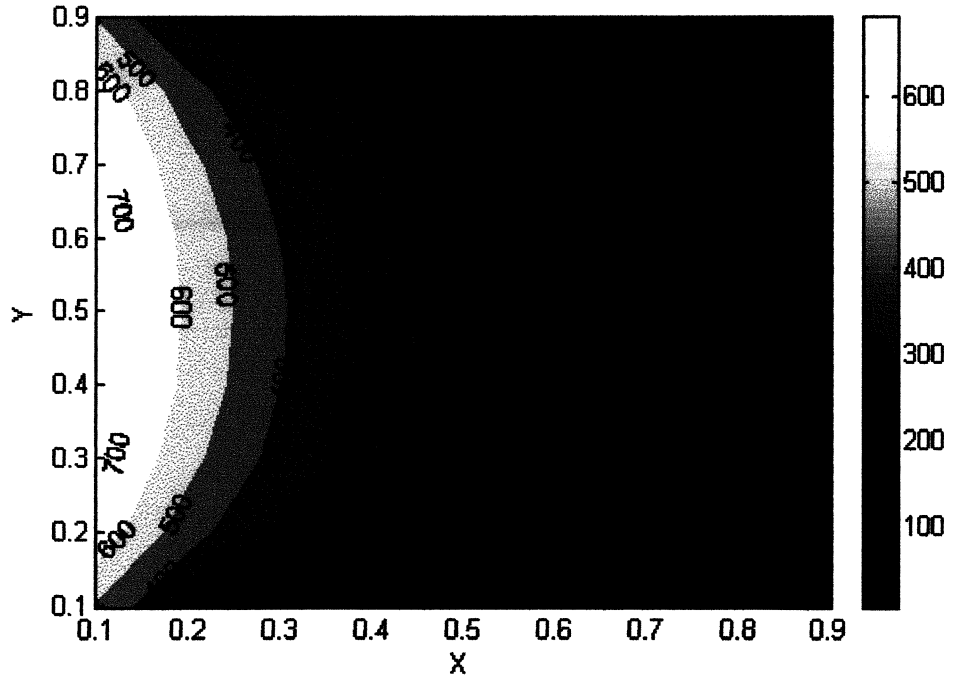
>> W=metodoprogresivo2('f','H','h1','h2','h3','h4','h5','h6',200)

```
...

W(:,:,201) =
  Columns 1 through 7
  484.4574  666.0508  741.8531  773.6371  782.1963  772.7539  740.3647
  271.8046  437.9409  527.7895  570.5733  582.4699  568.5253  524.2505
  164.8660  286.2046  360.9049  398.5265  408.7173  394.7498  353.9724
  101.5111  181.2106  231.2360  254.0637  259.2777  247.9268  218.1961
   60.0236  105.9919  128.8935  127.3528  126.5395  119.6083  104.0396
   32.6379   53.9201   51.0870         0         0         0         0
   16.6410   26.0161   21.5815         0         0         0         0
    7.9310   11.9542    9.2481         0         0         0         0
    3.1412    4.6394    3.4704         0         0         0         0

  Columns 8 through 9
  664.5134  483.4893
  434.2421  269.4658
  278.8122  160.1703
  166.9490   92.4477
   78.4139   42.7098
         0         0
         0         0
         0         0
         0         0
```

Gráfica de la estimación de la temperatura de la plancha en t=0.2 segundos:

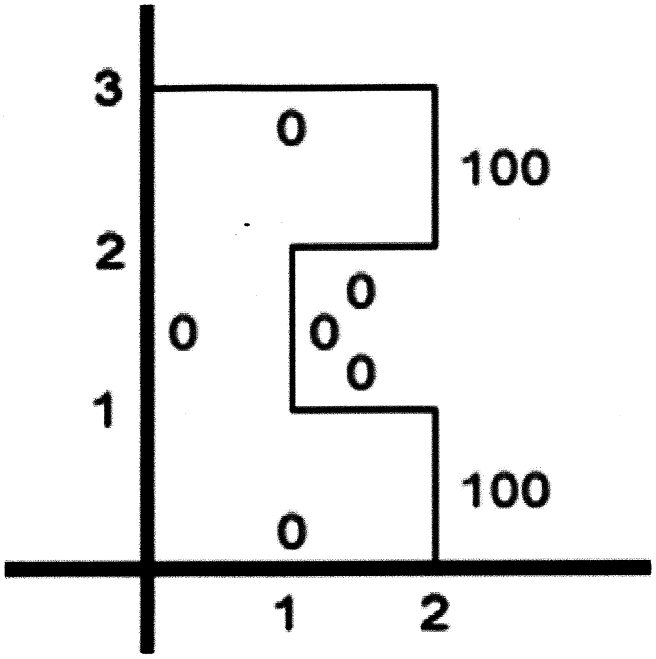


Isotermas de la plancha en t=0.2 segundos:



PROBLEMA 2. CALENTAMIENTO DE UNA PLANCHA NO CUADRADA

Tenemos una plancha con la siguiente forma:



Los valores en rojo representan las temperaturas en el contorno de la plancha. Por tanto, tenemos el siguiente planteamiento:

$$U_t=U_{xx}+U_{yy},$$

Con las siguientes condiciones de contorno:

$$u(0,y,t) = 0$$
$$u(1,y,t) = 0,\ 1 < y < 2$$
$$u(2,y,t) = 100,\ 0 < y < 1$$
$$u(2,y,t) = 100,\ 2 < y < 3$$
$$u(x,0,t) = 0$$
$$u(x,1,t) = 0,\ 1 < x < 2$$
$$u(x,2,t) = 0,\ 1 < x < 2$$
$$u(x,3,t) = 0$$

Y la condición inicial:

$$u(x,y,0) = 0$$

Se pide: Calcular, para un determinado t, cómo se distribuyen las isotermas.

Resolución:

Lo primero que hay que hacer es adaptar el algoritmo que hemos para calcular el calentamiento de una plancha cuadrada (algoritmo A4.1). Lo que voy a hacer es un algoritmo exclusivo para este tipo de problema, al que apenas se le pasan parámetros.

ALGORITMO [A5.2]

Método progresivo. Es estable si λ es menor o igual que ¼.

Datos que recibe el algoritmo:

- H: función H(t,x,y)
- h1: condición de contorno
- h2: condición de contorno
- h3: condición de contorno
- h4: condición de contorno
- h5: condición de contorno
- h6: condición de contorno
- h7: condición de contorno
- h8: condición de contorno
- p: valor final de t

Datos que devuelve el algoritmo:

- W: matriz de p filas, donde cada fila es una matriz de orden (nx-1)*(ny-1). El valor W(i,j,p) representará a $W_{i,j;p-1}$

```
function W = metodoprogresivo(f,H,h1,h2,h3,h4,h5,h6,h7,h8,p)
x0=0;
y0=0;
hx=0.2;
hy=0.2;
k=0.001;
lambda=k/(hx*hy);
W=zeros(9,14,p);
W0=zeros(9,14);
W(:,:,1)=W0;
for l=1:p
    WL=zeros(9,14);
    t=l*k;
    for j=1:4
        y=y0+j*hy;
        for i=1:9
```

```
            x=x0+i*hx;
            if j==1
                if i==1
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h1,
y,t)+lambda*feval(h5,x,t);
                else if i==9
                    WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h3,y,t)+lambda*
feval(h5,x,t);
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h5,x,t);
                    end
                end
            else
                if j==4
                    if i>4
                        if i==9
                            WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h3,y,t)+lambda*feval(h6,x,t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h6,x,t);
                        end
                    else
                        if i==1
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);


                        end
                    end
                else
```

```
                    if i==1
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                    else if i==9
                            WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h3,y,t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                        end
                    end
                end
            end
        end
    end
    for j=5:10
        y=y0+j*hy;
        for i=1:4
            x=x0+i*hx;
            if i==1
                WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
            else if i==4
                WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h2,y,t);
                else
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                end
            end
        end
    end
    for j=11:14
        y=y0+j*hy;
        for i=1:9
```

```
            x=x0+i*hx;
            if j==14
                if i==1
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t)+lambda*feval(h8,x,t);
                else if i==9
                    WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h4,y,t)+lambda*feval(h8,x,t);
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h8,x,t);
                    end
                end
            else
                if i==1
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                else if i==9
                    WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h4,y,t);
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                    end
                end
            end
        end
    end
    W(:,:,l+1)=WL;
end
```

Ahora hacemos la llamada al algoritmo. Pondremos p=200 para que nos calcule la temperatura en t=0,2 segundos.

Los archivos f.m, H.m, h1.m, h2.m, h3.m, h4.m, h5.m, h6.m, h7.m y h8.m son, respectivamente:

f.m

```
function z=f(x,y)
z=0;
```

H.m

```
function z=H(t,x,y)
z=0;
```

h1.m

```
function z=h1(y,t)
z=0;
```

h2.m

```
function z=h2(y,t)
z=0;
```

h3.m

```
function z=h3(x,t)
z=100;
```

h4.m

```
function z=h4(x,t)
z=100;
```

h5.m

```
function z=h5(x,t)
z=0;
```

h6.m

```
function z=h6(x,t)
z=0;
```

h7.m

```
function z=h7(x,t)
z=0;
```

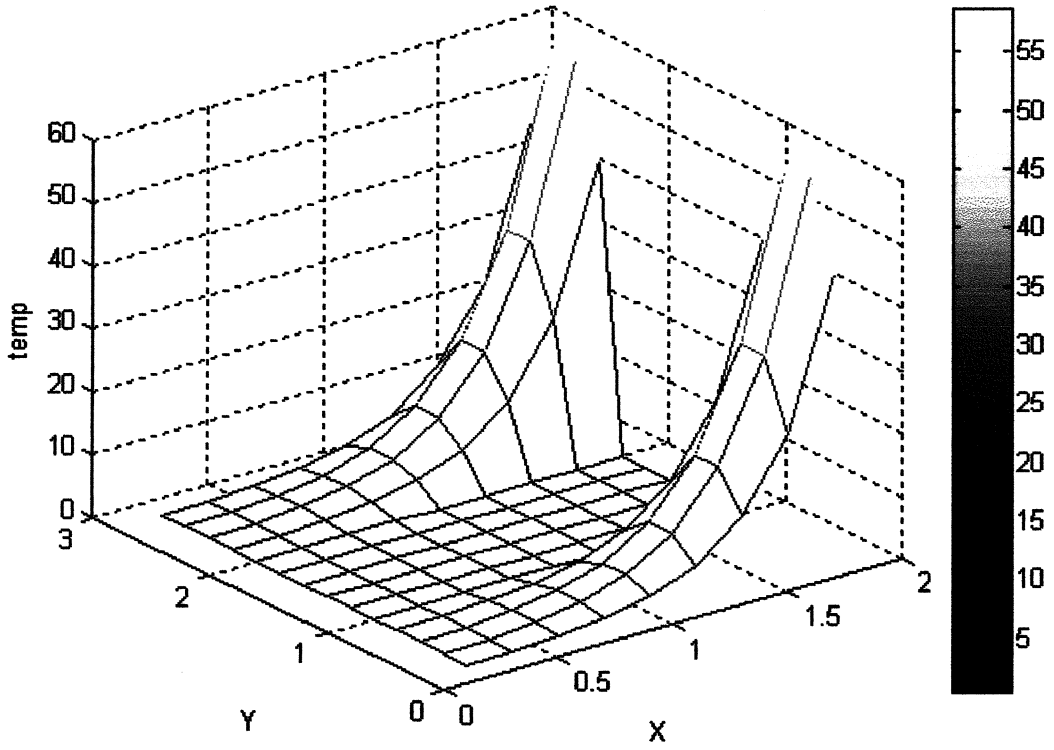
h8.m

```
function z=h8(x,t)
z=0;
```

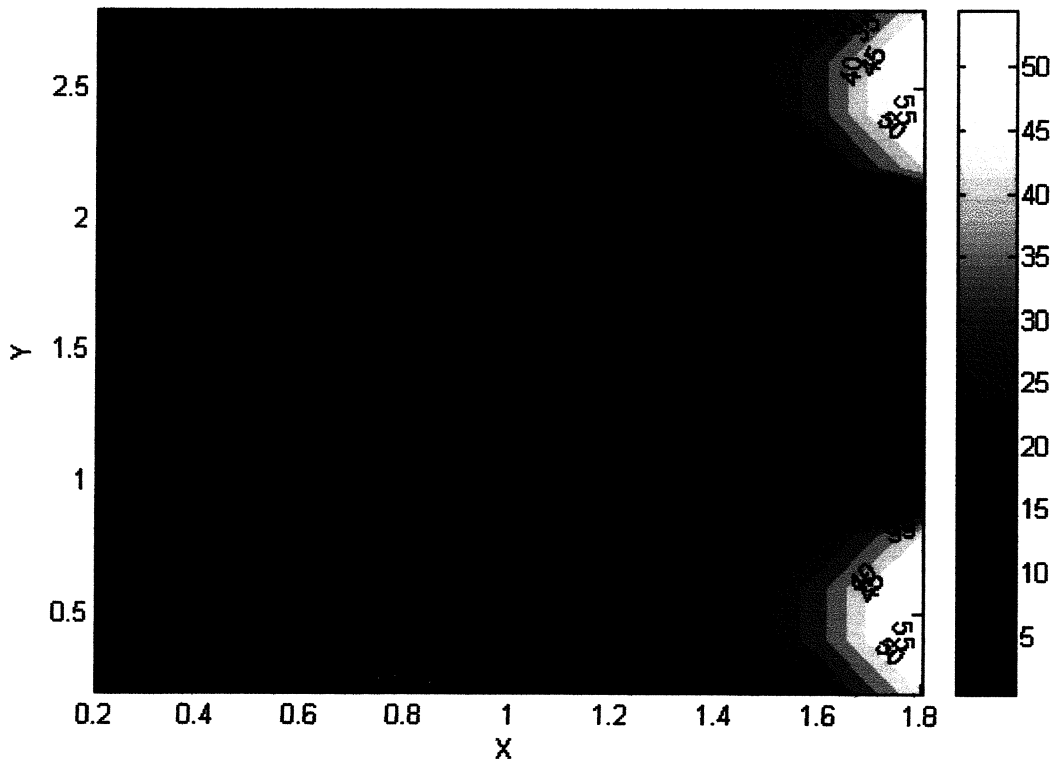
>> W=metodoprogresivo3('f','H','h1','h2','h3','h4','h5','h6','h7','h8',200)

```
...

W(:,:,201) =
    Columns 1 through 7

    0.0794    0.1313    0.1408    0.1126    0.0694    0.0370    0.0216
    0.2257    0.3710    0.3919    0.3019    0.1704    0.0835    0.0453
    0.5441    0.8875    0.9200    0.6697    0.3161    0.1333    0.0646
    1.2299    1.9906    2.0244    1.3700    0.4227    0.1385    0.0570
    2.6513    4.2566    4.2701    2.6876         0         0         0
    5.4965    8.7290    8.7333    5.5060         0         0         0
   11.0773   17.1936   17.1949   11.0797         0         0         0
   22.1214   32.5869   32.5872   22.1220         0         0         0
   45.2535   59.1443   59.1444   45.2536         0         0         0

  Columns 8 through 14

    0.0216    0.0370    0.0694    0.1126    0.1408    0.1313    0.0794
    0.0453    0.0835    0.1704    0.3019    0.3919    0.3710    0.2257
    0.0646    0.1333    0.3161    0.6697    0.9200    0.8875    0.5441
    0.0570    0.1385    0.4227    1.3700    2.0244    1.9906    1.2299
         0         0         0    2.6876    4.2701    4.2566    2.6513
         0         0         0    5.5060    8.7333    8.7290    5.4965
         0         0         0   11.0797   17.1949   17.1936   11.0773
         0         0         0   22.1220   32.5872   32.5869   22.1214
         0         0         0   45.2536   59.1444   59.1443   45.2535
```

Gráfica de la estimación de la temperatura de la plancha en t=0.2 segundos:

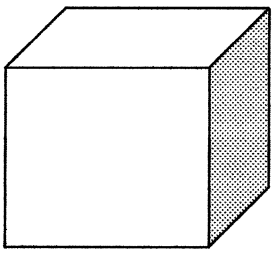


Isotermas de la plancha en t=0.2 segundos:



PROBLEMA 3. CALENTAMIENTO DE UN CUBO

En los casos anteriores hemos estudiado la ecuación del calor en una plancha. En esta ocasión vamos a intentar estudiar como evoluciona la temperatura de los puntos de un cubo.
Imaginemos que tenemos un cubo



donde $0 \le x \le 1$, $0 \le y \le 1$, $0 \le z \le 1$.
La ecuación del calor para tres dimensiones es la siguiente:

$$\frac{\partial u}{\partial t} = \alpha^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) + H(t,x,y,z)$$
$$t > 0$$

Con las siguientes condiciones de contorno:

$$u(t,0,y,z) = h_1(t,y,z) = 1000$$
$$u(t,1,y,z) = h_2(t,y,z) = 0$$
$$u(t,x,0,z) = h_3(t,x,z) = 0$$
$$u(t,x,1,z) = h_4(t,x,z) = 0$$
$$u(t,x,y,0) = h_5(t,x,y) = 0$$
$$u(t,x,z,1) = h_6(t,x,y) = 0$$

Y la condición inicial:

$$u(0,x,y,z) = f(x,y,z)$$

Lo primero que vamos a hacer es discretizar la ecuación. Si despreciamos los errores, tenemos:

$$\frac{u(t+k,x,y,z)-u(t,x,y,z)}{k} = \alpha^2 \left(\frac{u(t,x+h,y,z)-2u(t,x,y,z)+u(t,x-h,y,z)}{h^2} + \right.$$
$$\frac{u(t,x,y+h,z)-2u(t,x,y,z)+u(t,x,y-h,z)}{h^2} +$$
$$\left. \frac{u(t,x,y,z+h)-2u(t,x,y,z)+u(t,x,y,z-h)}{h^2} \right) + H(t,x,y,z)$$

Si tomamos $x=x_i$, $y=y_j$, $z=z_p$, $t=t_l$, $\lambda=k\alpha^2/h^2$ y $W_{ijp}^{\ l} \approx u(t_l,x_i,y_j,z_p)$, tenemos:

$$\begin{cases} W_{i,j,p}^{l+1} = (1-6\lambda)W_{i,j,p}^{l} + \lambda W_{i+1,j,p}^{l} + \lambda W_{i-1,j,p}^{l} + \lambda W_{i,j+1,p}^{l} + \lambda W_{i,j-1,p}^{l} + \lambda W_{i,j,p+1}^{l} + \lambda W_{i,j,p-1}^{l} \\ W_{i,j,p}^{0} = \left\{ f(x_i,y_j,z_p) \right\}_{\substack{i=1,\dots,n-1 \\ j=1,\dots,n-1 \\ p=1,\dots,n-1}} \end{cases}$$

donde n es el numero de divisiones que se harán.
Este método es estable si $\lambda \le 1/4$.

ALGORITMO [A5.3]

Método progresivo. Es estable si λ es menor o igual que ¼.

Datos que recibe el algoritmo:

- H: función H(t,x,y,z)
- h1: condición de contorno
- h2: condición de contorno
- h3: condición de contorno
- h4: condición de contorno
- h5: condición de contorno
- h6: condición de contorno
- q: valor final de t

Datos que devuelve el algoritmo:

- W: matriz de q filas, donde cada fila es una matriz de (n-1) filas. Cada una de (n-1) filas es a su vez una matriz cuadrada de orden (n-1). El valor W(i,j,p,q) representará a $W_{i,j,p;q-1}$

```
function W = metodoprogresivo3d(f,H,h1,h2,h3,h4,h5,h6,q)
x0=0;
y0=0;
z0=0;
h=0.1;
k=0.001;
alpha=1;
lambda=alpha^2*k/h^2;
n=10;

W=zeros(n-1,n-1,n-1,p);

W0=zeros(n-1,n-1,n-1);

W(:,:,:,1)=W0;

for l=1:q
    WL=zeros(n-1,n-1,n-1);
    t=l*k;
    for p=1:n-1
        z=z0+p*h;
        for j=1:n-1
            y=y0+j*h;
```

```
            for i=1:n-1
                x=x0+i*h;
                if p==1
                    if j==1
                        if i==1
                            WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H
,t,x,y,z)+lambda*feval(h1,y,z,t)+lambda*feval(h3,x,z,t)+lambda*feval(h
5,x,y,t);
                        else if i==n-1
                            WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H,t,x,y,z)+lamb
da*feval(h2,y,z,t)+lambda*feval(h3,x,z,t)+lambda*feval(h5,x,y,t);
                        else
                            WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H,t,x,y,z)+lamb
da*feval(h3,x,z,t)+lambda*feval(h5,x,y,t);
                        end
                    end
                else if j==n-1
                        if i==1
                            WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H,t,x,y,z)+lambda*feval(h1,y,z,t)+l
ambda*feval(h4,x,z,t)+lambda*feval(h5,x,y,t);
                        else if i==n-1
                            WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H,t,x,y,z)+lambda*feval(h2,y,z,t)+l
ambda*feval(h4,x,z,t)+lambda*feval(h5,x,y,t);
                        else
                            WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H,t,x,y,z)+lambda*feval(h4,x,z,t)+l
ambda*feval(h5,x,y,t);
                        end
                    end
                else
                    if i==1
```

```
                                WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H,t,x,y,z)+lambda*feval(h1,y,z,t)+l
ambda*feval(h5,x,y,t);
                            else if i==n-1
                                    WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H,t,x,y,z)+lambda*feval(h2,y,z,t)+l
ambda*feval(h5,x,y,t);
                                else
                                    WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+k*feval(H,t,x,y,z)+lambda*feval(h5,x,y,t);
                                end
                            end
                        end
                    end

                else if p==n-1
                        if j==1
                            if i==1
                                WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h1,y,z,t)+lambda*feval(h3,x,z,t)+
lambda*feval(h6,x,y,t);
                            else if i==n-1
                                    WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h2,y,z,t)+lambda*feval(h3,x,z,t)+
lambda*feval(h6,x,y,t);
                                else
                                    WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h3,x,z,t)+lambda*feval(h6,x,y,t);
                                end
                            end
                        else if j==n-1
```

```
                                    if i==1
                                        WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j-1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h1,y,z,t)+lambda*feval(h4,x,z,t)+
lambda*feval(h6,x,y,t);
                                    else if i==n-1
                                            WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-1,j,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h2,y,z,t)+lambda*feval(h4,x,z,t)+
lambda*feval(h6,x,y,t);
                                        else

WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j-1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h4,x,z,t)+lambda*feval(h6,x,y,t);
                                        end
                                    end
                                else
                                    if i==1
                                        WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h1,y,z,t)+lambda*feval(h6,x,y,t);
                                    else if i==n-1
                                            WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h2,y,z,t)+lambda*feval(h6,x,y,t);
                                        else

WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-1,p,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h6,x,y,t);
                                        end
                                    end
                                end
                            end


                        else
                            if j==1
```

```
                                if i==1
                                    WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(
i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h1,y,z,t)+lambda*feval(h3,x,z,t);
                                else if i==n-1
                                        WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h2,y,z,t)+lambda*feval(h3,x,z,t);
                                    else
                                        WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h3,x,z,t);
                                    end
                                end
                            else if j==n-1
                                    if i==1
                                        WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h1,y,z,t)+lambda*feval(h4,x,z,t);
                                    else if i==n-1
                                            WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-1,j,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h2,y,z,t)+lambda*feval(h4,x,z,t);
                                        else

WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j-1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h4,x,z,t);
                                        end
                                    end
                                else
                                    if i==1
                                        WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-
6*lambda)*W(i,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h1,y,z,t);
```

```
                                    else if i==n-1
                                                WL(i,j,p)=(1-
6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(i,j,p-
1,l)+k*feval(H,t,x,y,z)+lambda*feval(h2,y,z,t);
                                            else

WL(i,j,p)=lambda*W(i+1,j,p,l)+(1-6*lambda)*W(i,j,p,l)+lambda*W(i-
1,j,p,l)+lambda*W(i,j+1,p,l)+lambda*W(i,j-
1,p,l)+lambda*W(i,j,p+1,l)+lambda*W(i,j,p-1,l)+k*feval(H,t,x,y,z);
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
    end
    W(:,:,:,l+1)=WL;
end
```

Ahora hacemos la llamada al algoritmo. Pondremos p=200 para que nos calcule la temperatura en t=0,2 segundos.

Los archivos f.m, H.m, h1.m, h2.m, h3.m, h4.m, h5.m, y h6.m son, respectivamente:

f.m

```
function z=f(x,y)
z=0;
```

H.m

```
function z=H(t,x,y)
z=0;
```

h1.m

```
function z=h1(y,t)
z=1000;
```

h2.m

```
function z=h2(y,t)
```

```
z=0;
```

h3.m

```
function z=h3(x,t)
z=0;
```

h4.m

```
function z=h4(x,t)
z=0;
```

h5.m

```
function z=h5(x,t)
z=0;
```

h6.m

```
function z=h6(x,t)
z=0;
```

>> W=metodoprogresivo3d('f','H','h1','h2','h3','h4','h5','h6',200)

En este caso es más complicado mostrar los datos, puesto que para el instante t=0.2 segundos existen 9 matrices cuadradas de orden (n-1). Cada una de esas matrices representa a los distintos valores que toma z.

Por ejemplo, el valor de la temperatura en t=0.2 y en el plano z=0.1 sería:

>> W(:,:,1,201)

```
ans =
  Columns 1 through 7
  332.7420  428.6939  460.3059  471.7276  474.7025  471.7276  460.3059
  141.2284  205.9982  238.4636  252.6241  256.6679  252.6241  238.4636
   63.6480  108.0385  133.5643  146.0737  149.8794  146.0737  133.5643
   30.6600   60.2670   78.4258   88.0646   91.1138   88.0646   78.4258
   15.7535   34.9782   47.2769   54.1574   56.3868   54.1574   47.2769
    8.5006   20.7181   28.7643   33.4158   34.9459   33.4158   28.7643
    4.7031   12.2218   17.2720   20.2511   21.2406   20.2511   17.2720
    2.4921    6.8198    9.7513   11.4993   12.0833   11.4993    9.7513
    1.0243    3.0452    4.3947    5.2003    5.4700    5.2003    4.3947

  Columns 8 through 9
  428.6939  332.7420
  205.9982  141.2284
```

```
  108.0385   63.6480
   60.2670   30.6600
   34.9782   15.7535
   20.7181    8.5006
   12.2218    4.7031
    6.8198    2.4921
    3.0452    1.0243
```

El valor de la temperatura en t=0.2 y en el plano z=0.5 sería:

>> W(:,:,5,201)

```
ans =
  Columns 1 through 7
  474.6691  648.1052  718.2845  747.0961  755.0360  747.0961  718.2845
  256.6161  410.2195  491.7447  530.5183  541.9196  530.5183  491.7447
  149.8483  259.4356  327.5615  363.4710  374.5748  363.4710  327.5615
   91.1193  164.4156  214.5075  242.8022  251.8714  242.8022  214.5075
   56.4207  104.0975  138.6054  159.0057  165.7077  159.0057  138.6054
   34.9916   65.3679   88.1347  101.9958  106.6249  101.9958   88.1347
   21.2840   40.0435   54.4027   63.3086   66.3144   63.3086   54.4027
   12.1157   22.8854   31.2308   36.4647   38.2426   36.4647   31.2308
    5.4871   10.3857   14.2060   16.6161   17.4377   16.6161   14.2060
  Columns 8 through 9
  648.1052  474.6691
  410.2195  256.6161
  259.4356  149.8483
  164.4156   91.1193
  104.0975   56.4207
   65.3679   34.9916
   40.0435   21.2840
   22.8854   12.1157
   10.3857    5.4871
```

El valor de la temperatura en t=0.2 y en el plano z=0.9 sería:

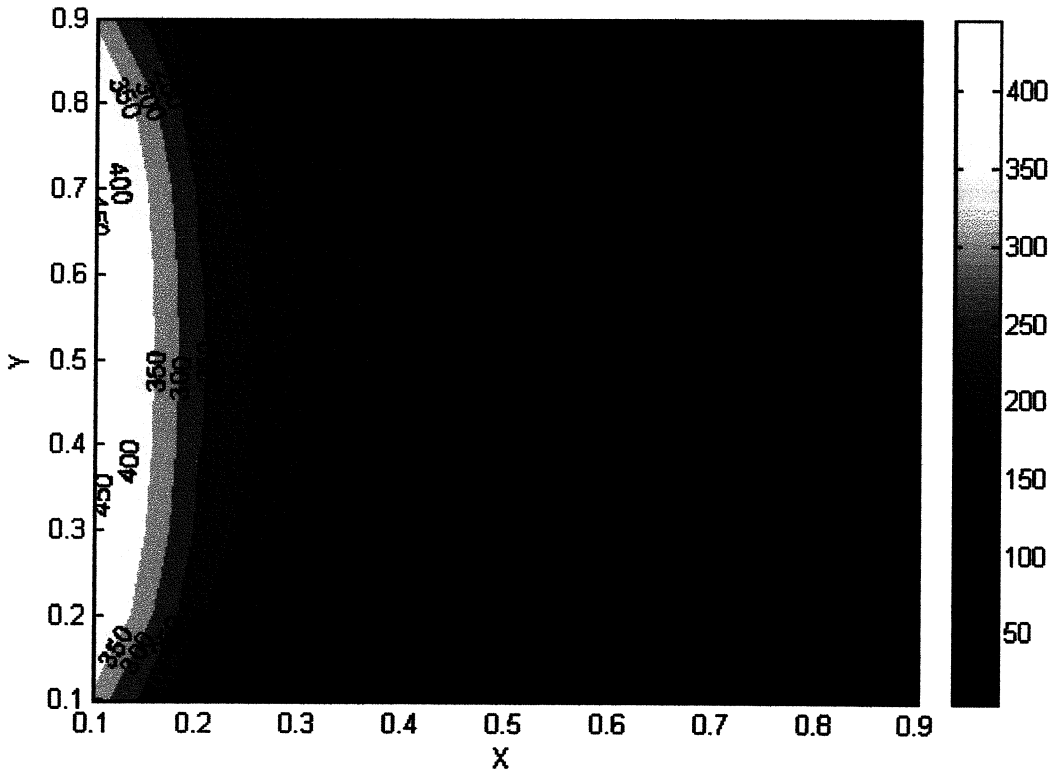
>> W(:,:,9,201)

```
ans =
  Columns 1 through 7
  331.5414  427.6160  459.9050  471.6037  474.6357  471.6037  459.9050
  134.0237  204.3576  237.7180  252.4079  256.5643  252.4079  237.7180
   63.9000  107.6078  133.0981  145.9376  149.8171  145.9376  133.0981
   34.1809   60.8065   78.3863   88.0640   91.1247   88.0640   78.3863
   19.5973   35.9052   47.4962   54.2519   56.4545   54.2519   47.4962
```

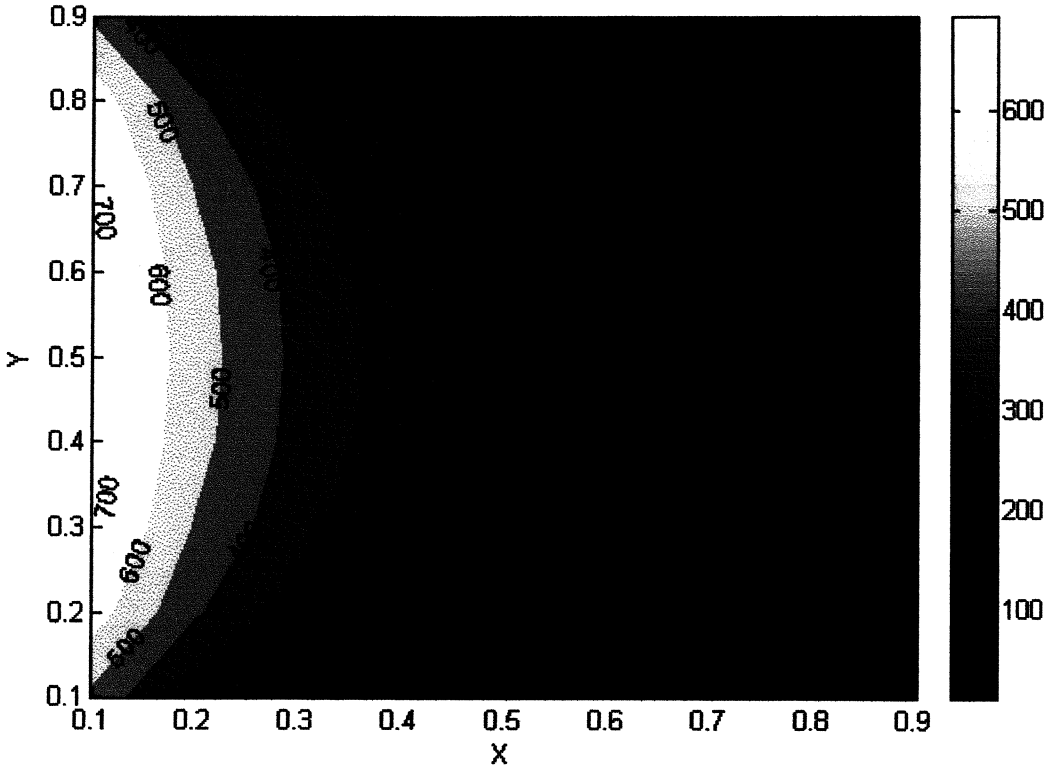
```
   11.6193   21.6282   29.0573   33.5450   35.0373   33.5450   29.0573
    6.8874   12.9310   17.5311   20.3713   21.3274   20.3713   17.5311
    3.8649    7.2898    9.9339   11.5873   12.1481   11.5873    9.9339
    1.7379    3.2855    4.4889    5.2463    5.5042    5.2463    4.4889
  Columns 8 through 9
  427.6160  331.5414
  204.3576  134.0237
  107.6078   63.9000
   60.8065   34.1809
   35.9052   19.5973
   21.6282   11.6193
   12.9310    6.8874
    7.2898    3.8649
    3.2855    1.7379
```

Para verlo mejor, podemos dibujar gráficamente los tres ejemplos anteriores:

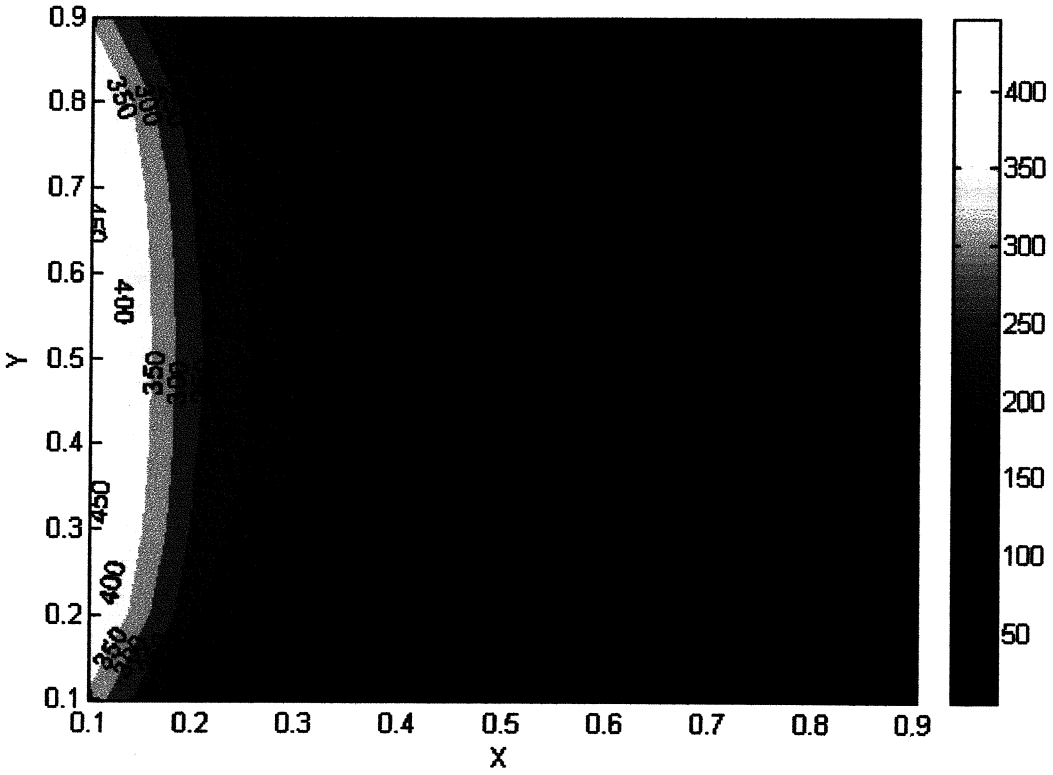
t=0.2 segundos y z=0.1



t=0.2 segundos y z=0.5

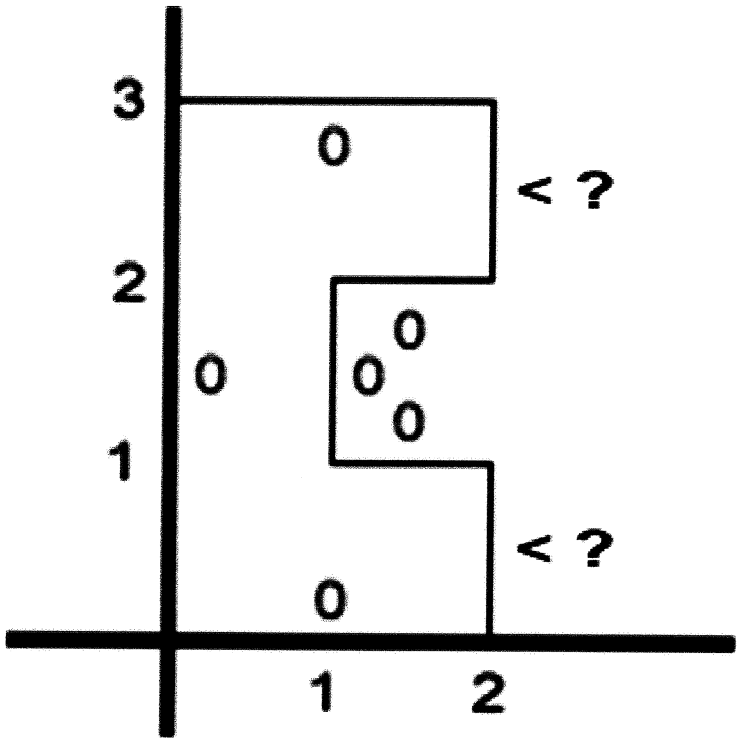


t=0.2 segundos y z=0.9



PROBLEMA 4. PROBLEMA DE CONTROL

Tenemos una plancha como la del dibujo:



Las temperaturas son los números en rojo. Queremos conocer qué temperatura hay que ponerle en los contornos marcados con "< ?" para que en el punto (0'7,1'3) la temperatura sea la que nosotros determinemos, en un instante dado.

Tenemos, por tanto, lo siguiente:

$$U_t=U_{xx}+U_{yy},$$

Con las siguientes condiciones de contorno:

$$u(0,y,t) = 0$$
$$u(1,y,t) = 0,\ 1 < y < 2$$
$$u(x,0,t) = 0$$
$$u(x,1,t) = 0,\ 1 < x < 2$$
$$u(x,2,t) = 0,\ 1 < x < 2$$
$$u(x,3,t) = 0$$

Y la condición inicial:

$$u(x,y,0) = 0$$

Las condiciones de contorno que desconocemos son:

$$u(2,y,t) = ¿?,\ 0 < y < 1$$
$$u(2,y,t) = ¿?,\ 2 < y < 3$$

Para solucionar este problema, voy a implementar un algoritmo que recibe unos valores iniciales de temperatura para los contornos donde no la conocemos, y va calculando las temperaturas para los instantes t correspondientes hasta llegar a un t determinado. Una vez hallados esos valores, que rellenarían la matriz W, comprobaríamos qué valor tiene el punto que queremos controlar. Si está dentro de un margen de error determinado, el algoritmo ha terminado. Si no, incrementamos o decrementamos (según el caso) la temperatura en los contornos donde no las conocemos, y volvemos a calcular las temperaturas.
El algoritmo termina bien cuando se llega a una buena aproximación de la temperatura deseada en el punto que queremos o bien cuando se sobrepasan las 100 iteraciones.
El algoritmo es el siguiente:

ALGORITMO [A5.4]

Datos que recibe el algoritmo:

- H: función H(t,x,y,z)
- h1: condición de contorno
- h2: condición de contorno
- h3: condición de contorno
- h4: condición de contorno
- h5: condición de contorno
- h6: condición de contorno
- h7: condición de contorno
- h8: condición de contorno
- p: valor final de t
- d: temperatura deseada en el punto (0'7,1'3)
- h3ini: valor inicial que le vamos a dar al contorno h3
- h4ini: valor inicial que le vamos a dar al contorno h4

Datos que devuelve el algoritmo:

- h3: valor que necesitamos poner en el contorno h3 para que el punto (0'7,1'3) tenga una temperatura de $(d \pm 0.01)$
- h4: valor que necesitamos poner en el contorno h4 para que el punto (0'7,1'3) tenga una temperatura de $(d \pm 0.01)$
- it: número de iteraciones realizadas

```
function [h3,h4,it] =
metodoprogresivo(f,H,h1,h2,h5,h6,h7,h8,p,d,h3ini,h4ini)
x0=0;
y0=0;
h=0.2;
k=0.001;
lambda=k/(h^2);
```

```
W=zeros(9,14,p);
W0=zeros(9,14);
W(:,:,1)=W0;
m1=0;
m2=0;
h3=h3ini;
h4=h4ini;
a=1;
it=0;

while(a==1)
if (it<=100)
    it=it+1;
    for l=1:p
        WL=zeros(9,14);
        t=l*k;
        for j=1:4
            y=y0+j*h;
            for i=1:9
                x=x0+i*h;
                if j==1
                    if i==1
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h1,
y,t)+lambda*feval(h5,x,t);
                    else if i==9
                        WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*h3+lambda*feval(h5,x,
t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+k*feval(H,t,x,y)+lambda*feval(h5,x,t);
                        end
                    end
                else
                    if j==4
                        if i>4
                            if i==9
```

```
                                    WL(i,j)=(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*h3+lambda*feval(h6,x,t);
                                else
                                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h6,x,t);
                                end
                            else
                                if i==1
                                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                                else
                                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);

                                end
                            end
                        else
                            if i==1
                                WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                            else if i==9
                                    WL(i,j)=(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*h3;
                                else
                                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                                end
                            end
                        end
                    end
                end
            end
        end
        for j=5:10
            y=y0+j*h;
```

```
            for i=1:4
                x=x0+i*h;
                if i==1
                    WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                else if i==4
                    WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h2,y,t);
                    else
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                    end
                end
            end
        end
        for j=11:14
            y=y0+j*h;
            for i=1:9
                x=x0+i*h;
                if j==14
                    if i==1
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t)+lambda*feval(h8,x,t);
                    else if i==9
                        WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*h4+lambda*feval(h8,x,t);
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h8,x,t);
                        end
                    end
                else
                    if i==1
```

```
                        WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y)+lambda*feval(h1,y,t);
                    else if i==9
                        WL(i,j)=(1-4*lambda)*W(i,j,l)+lambda*W(i-
1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-1,l)+k*feval(H,t,x,y)+lambda*h4;
                        else
                            WL(i,j)=lambda*W(i+1,j,l)+(1-
4*lambda)*W(i,j,l)+lambda*W(i-1,j,l)+lambda*W(i,j+1,l)+lambda*W(i,j-
1,l)+k*feval(H,t,x,y);
                        end
                    end
                end
            end
        end
        W(:,:,l+1)=WL;
    end
    punto1=(W(3,6,p+1)+W(4,6,p+1))/2;
    punto2=(W(3,7,p+1)+W(4,7,p+1))/2;
    punto=(punto1+punto2)/2;
    if (abs(punto-d)<0.01)
        a=2;
    else if (punto>d)
            if (h3>h3ini)
                m1=m1+1;
            end
            h3=h3-h3ini/(5+m1);
            h4=h4-h4ini/(5+m1);
        else if (punto<d)
                if (h3<h3ini)
                    m2=m2+1;
                end
                h3=h3+h3ini/(5+m2);
                h4=h4+h4ini/(5+m2);
            end
        end
    end
else
    if(punto<d)
        warning('No hay solución, ponga valores iniciales mayores');
    else if (punto>d)
```

```
            warning('No hay solución, ponga valores iniciales
menores');
        end
    end
    return;
end
end
```

Vamos a hacer una llamada a este algoritmo.

>> [h3,h4,it]=metodoprogresivo('f','H','h1','h2','h5','h6','h7','h8',200,0.9,300,300)

```
h3 =
   910

h4 =
   910
it =
    13
```

El resultado nos dice que, para obtener una temperature de 0.9 en el punto (0'7,1'3) tenemos que poner 910 tanto en h3 como en h4. El algoritmo ha hecho 13 iteraciones para dar con la solución.

Puede suceder que pase lo siguiente:

>> [h3,h4,it]=metodoprogresivo('f','H','h1','h2','h5','h6','h7','h8',200,0.9,30,30)

```
Warning: No hay solución, ponga valores iniciales mayores
> In metodoprogresivo at 127
```

Esto quiere decir que el algoritmo ha dado más de 100 iteraciones y no ha encontrado una solución lo suficientemente aproximada. Él mismo nos indica que necesita unos valores iniciales mayores para poder aproximar la solución. Por tanto, probemos con valores mayores:

>> [h3,h4,it]=metodoprogresivo('f','H','h1','h2','h5','h6','h7','h8',200,0.9,100,100)

```
h3 =
   920
h4 =
   920
it =
    42
```

Lógicamente, el algoritmo ofrece unos valores muy parecidos a los que daba antes (910), pero ha hecho 42 iteraciones frente a las 13 que hizo antes. El valor inicial que le demos al algoritmo es muy importante, pues de él dependerá la calidad de la solución dada.

Puede pasar que le demos al algoritmo unos valores muy grandes y que con ellos no sea capaz de llegar a la solución deseada.

Otra opción que tenemos es exigir al algoritmo un error más pequeño. Para hacerlo, tenemos que cambiar el algoritmo y poner:

Donde pone esto:

```
if (abs(punto-d)<0.01)
    a=2;
```

poner

```
if (abs(punto-d)<error)
    a=2;
```

donde error es el error que deseamos.

Apéndice: Formas de discretización

Primera derivada

- *Forma 1*

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$

- *Forma 2*

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

- *Forma 3*

$$f'(x) = \frac{f(x-h) - f(x)}{-h} + O(h)$$

Segunda derivada

$$f''(x) = \frac{f(x+h) - f(x) + f(x-h)}{h^2} + O(h^2)$$