

Método de Diferencias Finitas

Ecuaciones en derivadas parciales de segundo orden

Diego Cancio-Donlebún Fernández

16 de mayo de 2025

Contenido

EDPs de segundo orden

El Método de Diferencias Finitas

Esquema del método

Planteamiento del problema

Caso elíptico: La ecuación de Poisson

Planteamiento del problema

Forma matricial

Implementación en Mathematica

Caso parabólico: La ecuación de calor

Planteamiento del problema

Euler implícito

Crank-Nicolson

Caso hiperbólico: La ecuación de Ondas

Planteamiento del problema

Implementación en Mathematica

Bibliografía

Tipos de EDP

Ecuación general de segundo orden en dos variables:

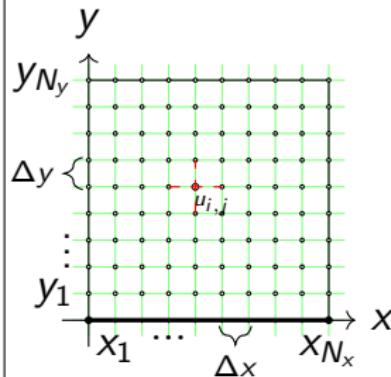
$$A(x,y) \frac{\partial^2 u}{\partial x^2} + B(x,y) \frac{\partial^2 u}{\partial x \partial y} + C(x,y) \frac{\partial^2 u}{\partial y^2} + F(x,y,u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}) = 0$$

Tipos:

- | | | |
|-----------------|--------------------|------------------------|
| $B^2 - 4AC < 0$ | Elíptica | La Ecuación de Poisson |
| $B^2 - 4AC = 0$ | Parabólica | La Ecuación de Calor |
| $B^2 - 4AC > 0$ | Hiperbólica | La Ecuación de Ondas |

Esquema de un Método de Diferencias finitas

Discretización del problema:



- ▶ Nodos en x, $x_i = i \cdot \Delta x$;
 N_x = número de nodos en x.
- ▶ Nodos en y, $y_j = j \cdot \Delta y$;
 N_y = número de nodos en y.
- ▶ $u_{i,j} \equiv$ aprox. numérica de $u(x_i, y_j)$

Discretización de los operadores diferenciales:

$$\frac{\partial u}{\partial y} \approx \frac{u_{i,j+1} - u_{i,j}}{\Delta y} \quad ; \quad \frac{\partial u}{\partial x} \approx \frac{u_{i+1,j} - u_{i,j}}{\Delta x} \quad ; \quad \frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

..., , ...

Planteamiento del problema

1. Las $u_{i,j}$ son las **incógnitas** del problema.
2. Al **discretizar** los operadores diferenciales, obtenemos **una ecuación lineal por cada $u_{i,j}$** . Se generan tantas ecuaciones como incógnitas.
3. En cada problema se especifican las **condiciones de contorno** para garantizar la unicidad de la solución.
Debemos incorporarlas en nuestras ecuaciones.

Reduce el problema a un **sistema de ecuaciones lineales**.

Notas:

- ▶ Los operadores diferenciales pueden discretizarse de distintas formas: diferencias centrales, progresivas, regresivas, etc. Elegir la más adecuada depende de cada problema y de la precisión requerida (como en EDOs).
- ▶ El sistema de ecuaciones resultante es inmenso; cualquier atajo que permita no enfrentarse directamente contra él es preferible.

Ejemplo 1, Caso elíptico: La ecuación de Poisson

Problema:

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y) & \text{en } \Omega = [0, 1] \times [0, 1] \subset \mathbb{R}^2, \\ u(\mathbf{x}) = g(\mathbf{x}) & \text{en } \partial\Omega \end{cases}$$

Discretizando los operadores diferenciales y sustituyendo:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = f_{i,j} \equiv f(x_i, y_j)$$

Para cada i, j obtenemos la ecuación:

$(\Delta x = \Delta y)$

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = \Delta x^2 \cdot f_{i,j}$$

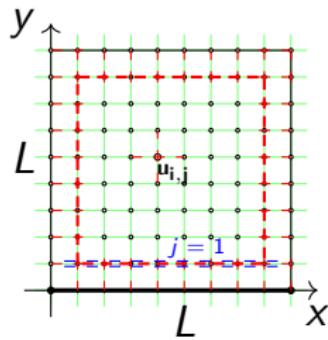
Condiciones de contorno

- ▶ Las condiciones de contorno fijan el valor de la solución en la frontera del cuadrado; tenemos que:

$$u_{i,0} = g(x_i, 0), \quad u_{i,n} = g(x_i, L) \quad i = 0, \dots, n$$

$$u_{0,j} = g(0, y_j), \quad u_{n,j} = g(L, y_j) \quad j = 0, \dots, n$$

- ▶ Estos valores deben ser sustituidos en las ecuaciones de los $u_{i,j}$ contiguos a la frontera:



Por ejemplo, cuando $j=1$:

$$u_{i+1,1} + u_{i-1,1} + u_{i,2} + u_{i,0} - 4u_{i,1} = \Delta x^2 \cdot f_{i,j}$$

Puesto que $u_{i,0} = g(x_i, 0)$, la ecuación es

$$u_{i+1,1} + u_{i-1,1} + u_{i,2} - 4u_{i,1} = \Delta x^2 \cdot f_{i,j} - g(x_i, 0)$$

Forma matricial

Para darle la forma de un sistema de ecuaciones lineales al uso, ordenando lexicográficamente las incógnitas, podemos renombrar: $u_{i,j} \equiv w_k$.

La correspondencia es:

$$(i,j) \mapsto k = (n-1)(j-1) + i \quad ; \quad k \mapsto \left(i(k) := [k - 1 \bmod (n-1)] + 1, \frac{k - i(k)}{n-1} \right)$$

Se comprueba que

$$(i+1,j) \mapsto k+1 \quad ; \quad (i-1,j) \mapsto k-1 \quad ; \quad (i,j+1) \mapsto k+n-1 \quad ; \quad (i,j-1) \mapsto k-n+1$$

Ahora la ecuación para $(i,j) \mapsto k$ se escribe como:

$$w_{k+1} + w_{k-1} + w_{k+n-1} + w_{k-n+1} - 4w_k = \Delta x^2 \cdot \tilde{f}_k$$

Forma matricial

Matriz del sistema ampliada, para unas condiciones de contorno y un termino independiente arbitrarios

$$\left(\begin{array}{cccccccccccccc|c} -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0,125 \\ 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0,457107 \\ 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0,3125 \\ 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0,1875 \\ 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0,457107 \\ 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0,707107 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -0,25 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0,3125 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0,707107 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & -4 & 1 & 0 & 0 & 1 & 0,5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 & -0,1875 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & -4 & 1 & 0 & -0,25 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & -4 & 1,49976 \times 10^{-32} \end{array} \right)$$

Implementación en Mathematica, parámetros

```
= L = 1.0; (*Longitud del lado*)
n = 15; (*Subdivisiones del lado*)
dx = L / (n - 1); (*Paso horizontal*)
Table[x[i] = i*dx, {i, 0, n}]; (*Nodos en x*)

$$[\text{tabla}]$$

Table[y[i] = i*dx, {i, 0, n}]; (*Nodos en y*)

$$[\text{tabla}]$$

f[x_, y_] := Sin[Pi x] Sin[Pi y]; (*Función f(x,y) *)

$$[\text{senc } \text{número } \cdot \text{senc } \text{número pi}]$$

(*Funciones índice*)
i[k_] := Mod[k - 1, n - 1] + 1

$$[\text{operación módulo}]$$

j[k_] := 1 + ((k - i[k]) / (n - 1))

(*Condiciones de frontera*)
u[x_, 0] := -x^2 + x; u[x_, 1] := -x^2 + x; u[0, y_] := -y^2 + y; u[1, y_] := -y^2 + y;

(*Condiciones de frontera*)
Table[U[i, 0] = u[x[i], 0]; U[i, n] = u[x[i], 1]; U[0, i] = u[0, y[i]]; U[n, i] = u[1, y[i]], {i, 0, n}];

$$[\text{tabla}]$$

```

Implementación en Mathematica, método

```
dim = (n - 1)^2; (*Dimensión de la matriz*)

A = Table[Coef[k, l] (*Matriz del sistema*)
  |tabla
  = If[k + 1 == 1, If[i[k] == n - 1, 0, 1],
    |si
    |si
    If[k - 1 == 1, If[i[k] == 1, 0, 1],
      |si
      |si
      If[k + n - 1 == 1, If[j[k] == n - 1, 0, 1],
        |si
        |si
        If[k - n + 1 == 1, If[j[k] == 1, 0, 1],
          |si
          |si
          If[k == 1, -4, 0]]]], {k, 1, (n - 1)^2}, {l, 1, (n - 1)^2}];

B = Table[ib[l] = (f[x[i[l]]], y[j[l]]) (*Términos independientes*)
  |tabla
  - If[i[l] == 1, u[0, j[l]], 0]
    |si
  - If[i[l] == n - 1, u[n, j[l]], 0]
    |si
  - If[j[l] == 1, u[i[l], 0], 0]
    |si
  - If[j[l] == n - 1, u[i[l], n], 0]), {l, 1, (n - 1)^2}];

(*Formamos la matriz del sistema aumentada*)

MatrizSistema = Join[A, Transpose[{B}], 2];
```

Implementación en Mathematica, resolución

```
k = 1;
While[k < dim, (*Operaciones de filas hasta convertirla en una matriz triangular superior*)
  [mientras
    If[MatrizSistema[[k, k]] ≠ 0,
      [si
        Table[MatrizSistema[[k + i, m]] = Round[MatrizSistema[[k + i, m]] - (MatrizSistema[[k + i, k]] / MatrizSistema[[k, k]]) * MatrizSistema[[k, m]], 0.0001],
          [tabla
            [entero más próximo
              {i, 1, dim - k}, {m, k, dim + 1}]];
        k++];
      ];
    ];
  ];
]

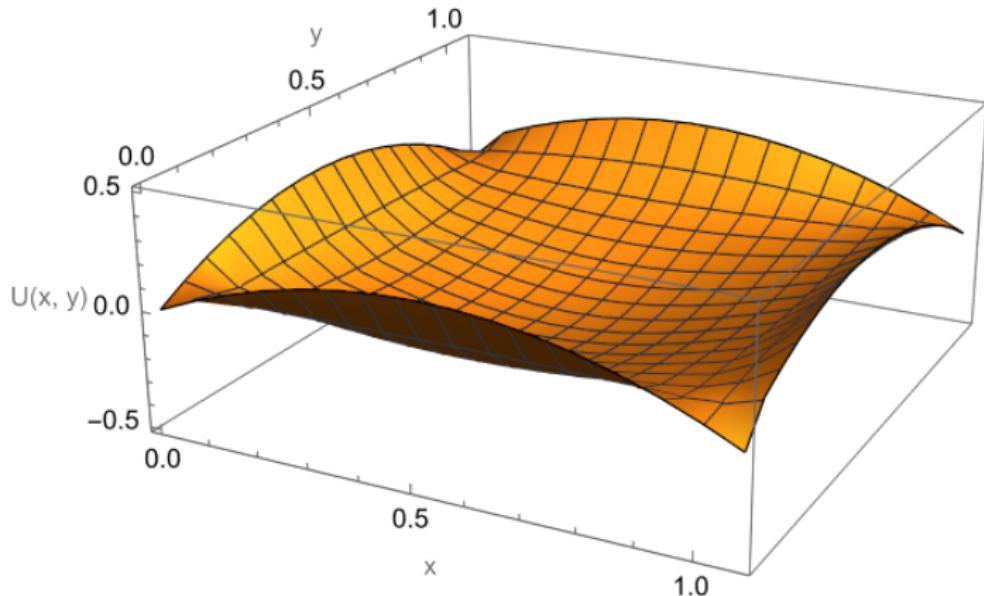
k = dim;
While[k > 1, (*Operaciones de filas hasta convertirla en una matriz diagonal*)
  [mientras
    If[MatrizSistema[[k, k]] ≠ 0,
      [si
        Table[MatrizSistema[[k - i, m]] = Round[MatrizSistema[[k - i, m]] - (MatrizSistema[[k - i, k]] / MatrizSistema[[k, k]]) * MatrizSistema[[k, m]], 0.0001],
          [tabla
            [entero más próximo
              {i, 1, k - 1}, {m, k, dim + 1}]];
        k--];
      ];
    ];
  ];
]

(*Normalizar las filas para obtener la matriz identidad en el lado izquierdo*)
Table[MatrizSistema[[k, dim + 1]] = Round[MatrizSistema[[k, dim + 1]] / MatrizSistema[[k, k]], 0.0001];
  [tabla
    [entero más próximo
      MatrizSistema[[k, k]] = 1, {k, 1, dim}]];
```

Implementación en Mathematica, representación

```
!:=  
Table[W[k] = MatrizSistema[[k, dim + 1]], {k, 1, dim}];  
[tabla  
Table[U[i[k], j[k]] = W[k], {k, 2, dim}];  
[tabla  
  
!:= ListPlot3D[  
[representación 3D de lista  
Flatten[Table[{x[i], y[j], U[i, j]}, {j, 0, dim}, {i, 0, dim}], 1],  
[aplana [tabla  
InterpolationOrder → 3,  
[orden de interpolación  
PlotRange → {All, All, {-0.5, 0.5}}, AxesLabel → {"x", "y", "U(x, y)"}, PlotLabel → "Solución de la ecuación de Poisson",  
[rango de repre... [todo [todo [etiqueta de ejes [etiqueta de representación  
PlotStyle → PointSize[0.01]]  
[estilo de repr... [tamaño de punto
```

Solución de la ecuación de Poisson



Ejemplo 2, Caso parabólico: La ecuación de calor

Problema: $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$; $u(x, 0) = f(x)$

Podemos discretizar la derivada espacial segunda de muchas formas distintas, dando lugar a distintos métodos. Veamos dos:

Euler implícito : $\frac{u_i^{n+1} - u_i^n}{\Delta t} = F_i^{n+1}$

Crank-Nicolson : $\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{1}{2}(F_i^{n+1} + F_i^n)$

Notación: $F_i^n \equiv F(u_n^i, x_i, t_n, \frac{u_{i+1}^n - u_i^n}{\Delta x}, \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2})$.

Ecuación de calor: Euler implícito

Aproximación numérica:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2}$$

Pasamos los términos en u^{n+1} a un lado y aquellos en u^n al otro

$$u_i^{n+1} - \alpha \frac{\Delta t}{\Delta x^2} (u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) = u_i^n$$

$$Denotamos r = \alpha \frac{\Delta t}{\Delta x^2} \Rightarrow u_i^{n+1} - ru_{i+1}^{n+1} + 2ru_i^{n+1} - ru_{i-1}^{n+1} = u_i^n$$

Ecuación de calor: Euler implícito

La aproximación numérica de la solución en el instante siguiente se obtiene como solución del sistema de ecuaciones lineales:

$$-ru_{i+1}^{n+1} + (1 + 2r)u_i^{n+1} - ru_{i-1}^{n+1} = u_i^n$$

En forma matricial:

$$\begin{pmatrix} (1 + 2r) & -r & 0 & \cdots & 0 \\ -r & (1 + 2r) & -r & \cdots & 0 \\ 0 & -r & \ddots & & \vdots \\ \vdots & \vdots & \ddots & -r & (1 + 2r) \\ 0 & \cdots & & -r & (1 + 2r) \end{pmatrix} \begin{pmatrix} u_0^{n+1} \\ \vdots \\ u_i^{n+1} \\ \vdots \\ u_{N_x}^{n+1} \end{pmatrix} = \begin{pmatrix} u_0^n \\ \vdots \\ u_i^n \\ \vdots \\ u_{N_x}^n \end{pmatrix}$$

$=\mathbf{A}$ $=\mathbf{u}^{n+1}$ $=\mathbf{u}^n$

Implementación en Mathematica, parámetros

```
L = 10; (*Longitud de la barra*)
T = 5; (*Tiempo total de la simulación*)
nx = 50; (*Número de nodos espaciales*)
nt = 300; (*Número de pasos temporales*)
alpha = 0.01; (*Difusividad térmica*)
dx = L / (nx - 1); (*Paso espacial*)
dt = T / nt; (*Paso temporal*)
r = alpha * dt / dx^2;
Table[t[i] = i * dt, {i, 0, nt}]; (*Nodos temporales*)
[tabla
Table[x[i] = i * dx, {i, 0, nx - 1}]; (*Nodos espaciales*)
[tabla
u0[x_] := Sin[x]; (*Temperatura inicial de la barra*)
[seno
V[0] = Table[u0[x[i]], {i, 0, nx - 1}];
[tabla
(*Guardamos en un vector los valores de la solución en los nodos en cada instante*)

A = Table[If[i == j, 1 + 2 r, If[i == j + 1 || j == i + 1, -r, 0]], {j, 1, nx}, {i, 1, nx}];
[tabla [si [si
(*Matriz del sistema*)]
```

Implementación en Mathematica, método

```
i = 0;
While[i < nt, (*Cada iteración proporciona la aproximación numérica en el instante siguiente*)
  mientras

  MatrizSistema = Join[A, Transpose[{V[i]}], 2]; (*Formamos la matriz del sistema aumentada*)
    |junta      |transposición

  m = 1;
  While[m < nx, (*Operaciones de filas hasta convertirla en una matriz triangular superior*)
    mientras
      Table[MatrizSistema[[m + 1, n]] = MatrizSistema[[m + 1, n]] - (MatrizSistema[[m + 1, m]] / MatrizSistema[[m, m]]) * MatrizSistema[[m, n]]
        |tabla
        , {n, 1, nx + 1}];
      m++;
    m = nx;

    While[m > 1, (*Operaciones de filas hasta convertirla en una matriz diagonal*)
      mientras
        Table[MatrizSistema[[m - 1, n]] = MatrizSistema[[m - 1, n]] - (MatrizSistema[[m - 1, m]] / MatrizSistema[[m, m]]) * MatrizSistema[[m, n]]
          |tabla
          , {n, 1, nx + 1}];
        m = m - 1];
    V[i + 1] = Table[MatrizSistema[[j + 1, nx + 1]] / MatrizSistema[[j + 1, j + 1]], {j, 0, nx - 1}]; (*Vector solución*)
    |tabla
  i++];

```

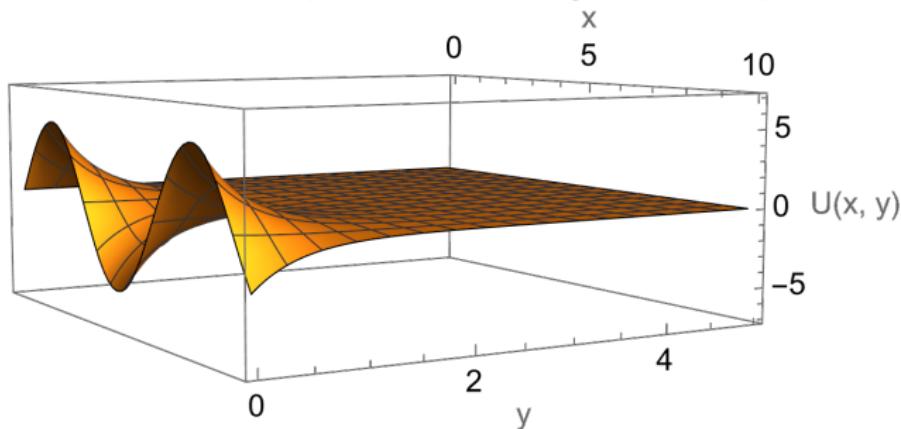
Implementación en Mathematica, solución

```
N[Table[U[t[j], x[i]] = V[j][[i + 1]], {j, 0, nt}, {i, 0, nx - 1}], 4];
 $\vdots$  [tabla
(*Damos forma a los vectores con los valores de la solución*)

ListPlot3D[
representación 3D de lista
Flatten[Table[{x[i], t[j], U[t[j], x[i]]}, {j, 0, nt}, {i, 0, nx - 1}], 1],
 $\vdots$  [aplana [tabla
InterpolationOrder → 3,
orden de interpolación
PlotRange → {All, All, {-7, 7}},
rango de repre... [todo ]todo
AxesLabel → {"x", "y", "U(x, y)"}, PlotLabel → "Evolución de la Temperatura a lo largo del Tiempo",
etiqueta de ejes etiqueta de representación
PlotStyle → PointSize[0.01]]
estilo de repr... [tamaño de punto]
```

Representación de la solución

Evolución de la Temperatura a lo largo del Tiempo



Ecuación de calor: Crank-Nicolson

Aproximación numérica:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\alpha}{2\Delta x^2} ((u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}) + (u_{i+1}^n - 2u_i^n + u_{i-1}^n))$$

Pasamos los términos en u^{n+1} a un lado y aquellos en u^n al otro

$$-ru_{i+1}^{n+1} + (1 + 2r)u_i^{n+1} - ru_{i-1}^{n+1} = ru_{i+1}^n + (1 - 2r)u_i^n + ru_{i-1}^n$$

Donde, ahora, $r = \alpha \frac{\Delta t}{2\Delta x^2}$

Ecuación de calor: Crank-Nicolson

En forma matricial:

$$\left(\begin{array}{cccc} (1+2r) & -r & \cdots & 0 \\ -r & (1+2r) & & \vdots \\ \vdots & & \ddots & -r \\ 0 & \cdots & -r & (1+2r) \end{array} \right) \begin{pmatrix} u_0^{n+1} \\ \vdots \\ u_{N_x}^{n+1} \end{pmatrix} = \left(\begin{array}{cccc} (1-2r) & r & \cdots & 0 \\ r & (1-2r) & & \vdots \\ \vdots & & \ddots & r \\ 0 & \cdots & r & (1-2r) \end{array} \right) \begin{pmatrix} u_0^n \\ \vdots \\ u_{N_x}^n \end{pmatrix}$$

$= \mathbf{A}$ $= \mathbf{u}^{n+1}$ $= \mathbf{B}$ $= \mathbf{u}^n$

La aproximación numérica de la solución en el siguiente instante se obtiene como solución del sistema de ecuaciones lineales:

$$B^{-1} A \mathbf{u}^{n+1} = \mathbf{u}^n$$

Implementación en Mathematica, parámetros

```
L = 10; (*Longitud de la barra*)
T = 5; (*Tiempo total de la simulación*)
nx = 50; (*Número de nodos espaciales*)
nt = 300; (*Número de pasos temporales*)
alpha = 0.5; (*Difusividad térmica*)
dx = L / (nx - 1); (*Paso espacial*)
dt = T / nt; (*Paso temporal*)
r = alpha * dt / (2 * dx^2);
Table[t[i] = i * dt, {i, 0, nt}]; (*Nodos temporales*)

$$t[i] = i \cdot \frac{T}{nt}$$

Table[x[i] = i * dx, {i, 0, nx - 1}]; (*Nodos espaciales*)

$$x[i] = i \cdot \frac{L}{nx - 1}$$

u0[x_] := 5 * Sin[x]; (*Temperatura inicial de la barra*)

$$u0[x] = 5 \cdot \sin(x)$$

(*Guardamos en un vector los valores de la solución en los nodos en cada instante*)
V[0] = Table[u0[x[i]], {i, 0, nx - 1}];

$$V[0] = \{u0[x[0]], u0[x[1]], \dots, u0[x[nx-1]]\}$$

(*Generamos las matrices A y B*)
A = Table[If[i == j, 1 + 2 r, If[i == j + 1 || j == i + 1, -r, 0]], {j, 1, nx}, {i, 1, nx}];

$$A = \begin{matrix} & \begin{matrix} 1 & 2r & 0 & \dots & 0 \end{matrix} \\ \begin{matrix} 1 & 0 & -r & \dots & 0 \end{matrix} & \begin{matrix} 1 & 2r & 0 & \dots & 0 \\ 0 & 1 & -r & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{matrix} \end{matrix}$$

B = Table[If[i == j, 1 - 2 r, If[i == j + 1 || j == i + 1, r, 0]], {j, 1, nx}, {i, 1, nx}];

$$B = \begin{matrix} & \begin{matrix} 1 & -2r & 0 & \dots & 0 \end{matrix} \\ \begin{matrix} 1 & r & 0 & \dots & 0 \end{matrix} & \begin{matrix} 1 & -2r & 0 & \dots & 0 \\ 0 & 1 & r & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{matrix} \end{matrix}$$

```

Implementación en Mathematica, invertir B

```
(*Formamos la matriz (B| I) para calcular la inversa de B*)
BI = Join[B, IdentityMatrix[Length[B]], 2];
  |junta      |matriz identidad      |longitud

m = 1;
While[m < nx, (*Operaciones de filas hasta convertir B en una matriz triangular superior*)
|mientras
  Table[BI[[m + 1, n]] = BI[[m + 1, n]] - (BI[[m + 1, m]] / BI[[m, m]]) * BI[[m, n]], {n, 1, 2*nx}];
|tabla
  m++];
m = nx;
While[m > 1, (*Operaciones de filas hasta convertir B en una matriz diagonal*)
|mientras
  Table[BI[[m - 1, n]] = BI[[m - 1, n]] - (BI[[m - 1, m]] / BI[[m, m]]) * BI[[m, n]], {n, 1, 2*nx}];
|tabla
  m = m - 1];
Table[BI[[m, n]] = BI[[m, n]] / BI[[m, m]], {m, 1, nx}, {n, 1, 2*nx}];
|tabla
(*La parte derecha de BI es ahora la inversa de B*)
InvB = Table[BI[[i, j]], {i, 1, nx}, {j, nx + 1, 2*nx}];
|tabla
```

Implementación en Mathematica, método

```
i = 0;
While[i < nt, (*Cada iteración proporciona la aproximación numérica en el instante siguiente*)
  mientras
    MatrizSistema = Join[InvB*A, Transpose[{V[i]}], 2]; (*Formamos la matriz ampliada del sistema*)
      Junta           |transposición
    m = 1;
    While[m < nx, (*Operaciones de filas hasta convertirla en una matriz triangular superior*)
      mientras
        Table[
          |tabla
          MatrizSistema[[m + 1, n]] = MatrizSistema[[m + 1, n]] - (MatrizSistema[[m + 1, m]] / MatrizSistema[[m, m]]) * MatrizSistema[[m, n]],
            {n, 1, nx + 1}];
        m++;
      m = nx;
      While[m > 1, (*Operaciones de filas hasta convertirla en una matriz diagonal*)
        mientras
          Table[
            |tabla
            MatrizSistema[[m - 1, n]] = MatrizSistema[[m - 1, n]] - (MatrizSistema[[m - 1, m]] / MatrizSistema[[m, m]]) * MatrizSistema[[m, n]],
              {n, 1, nx + 1}];
          m = m - 1];
      V[i + 1] = Table[MatrizSistema[[j + 1, nx + 1]] / MatrizSistema[[j + 1, j + 1]], {j, 0, nx - 1}]; (*Vector solución*)
        |tabla
      i++;
    ]
```

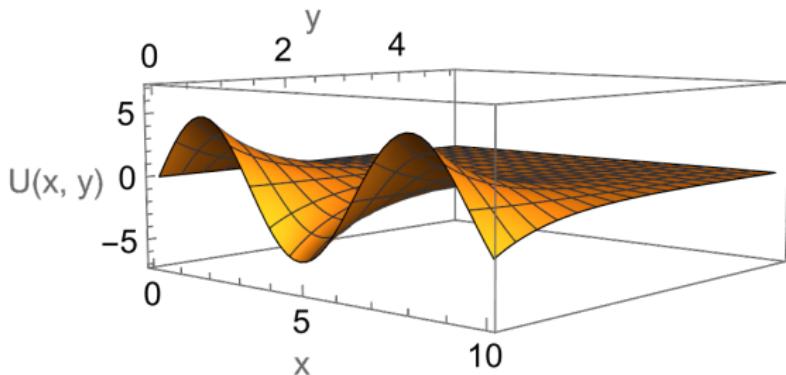
Implementación en Mathematica, solución

```
N[Table[U[t[j]], x[i]] = V[j][i + 1], {j, 0, nt}, {i, 0, nx - 1}], 4];
 $\vdots$  [tabla
(*Damos forma a los vectores con los valores de la solución*)

ListPlot3D[
 $\vdots$  [representación 3D de lista
Flatten[Table[{x[i], t[j], U[t[j], x[i]]}, {j, 0, nt}, {i, 0, nx - 1}], 1],
 $\vdots$  [aplaña [tabla
InterpolationOrder → 3,
 $\vdots$  [orden de interpolación
PlotRange → {All, All, {-7, 7}},
 $\vdots$  [rango de repre... [todo [todo
AxesLabel → {"x", "y", "U(x, y)"}, PlotLabel → "Evolución de la Temperatura a lo largo del Tiempo",
 $\vdots$  [etiqueta de ejes [etiqueta de representación
PlotStyle → PointSize[0.01]]
 $\vdots$  [estilo de repr... [tamaño de punto
```

Representación de la solución

Evolución de la Temperatura a lo largo del Tiempo



Ejemplo 3, Caso hiperbólico: La Ecuación de Ondas

Problema:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + f(x, y, t) \quad \text{en } \Omega = (0,1) \times (0,1) \subset \mathbb{R}^2$$

Condición de frontera $u(\mathbf{x}, t) = 0$ en $\partial\Omega$

Condición inicial $\begin{cases} u(x, y, 0) = 0 & \text{Posición} \\ \frac{\partial u}{\partial t}(x, y, 0) = 0 & \text{Velocidad} \end{cases}$ en Ω

Planteamiento del problema

Discretizando los operadores diferenciales y sustituyendo:

$$\frac{u_{i,j}^{k+1} + u_{i,j}^{k-1} - 2u_{i,j}^k}{\Delta t^2} = c^2 \left(\frac{u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k}{\Delta x^2} + \frac{u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k}{\Delta y^2} \right) + f_{i,j}^k$$

Para cada i, j, k obtenemos la ecuación:

$(\Delta x = \Delta y)$

$$u_{i,j}^{k+1} = 2u_{i,j}^k - u_{i,j}^{k-1} + \left(c \frac{\Delta t}{\Delta x}\right)^2 (u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k - 4u_{i,j}^k) + \Delta t^2 \cdot f_{i,j}^k$$

Esta fórmula actualiza la solución en el instante $k + 1$ en función de los dos pasos anteriores. Las condiciones iniciales proporcionan el punto de partida, y a partir de ahí, se aplica recursivamente esta fórmula para avanzar en el tiempo.

Iniciación del Algoritmo

- ▶ La condición inicial de posición, $u(x, y, 0)$, nos proporciona los valores $u_{i,j}^0$ de la solución en el instante inicial.
- ▶ La condición inicial de velocidad proporciona la velocidad inicial de la membrana, $\frac{\partial u}{\partial t}(x, y, 0)$, lo que nos permite dar una aproximación numérica razonada de la posición en el instante anterior (método de Euler), así obtenemos los valores u_{ij}^{-1} .

Esto nos permitiría inicializar el algoritmo en un contexto mucho más general que el que nos ocupa. En nuestro caso, como la membrana parte del reposo, sin hacer ningún cálculo:

$$u_{i,j}^0 = u_{i,j}^{-1} = 0 \quad \forall i, j$$

Implementación en Mathematica, parámetros

```
L = 1.0; (*Longitud del lado*)
nx = 15; (*Número de pasos espaciales*)
T = 2.0; (*Tiempo total*)
nt = 100; (*Número de pasos temporales*)
dx = L / (nx - 1); (*Paso espacial*)
dt = T / (nt - 1); (*Paso temporal*)
c = 1.0; (*Velocidad de propagación*)
r = (c * dt / dx)^2; (*Coeficiente r*)
Table[x[i] = i * dx, {i, 0, nx}]; (*Nodos en x*)

$$\text{tabla}$$

Table[y[j] = j * dx, {j, 0, nx}]; (*Nodos en y*)

$$\text{tabla}$$

Table[t[k] = k * dt, {k, 0, nt}]; (*Nodos en t*)

$$\text{tabla}$$

(*Fuerza externa (voz del cantante)*)
A = 50;
f0 = 261.63; (*Frecuencia del Do mayor en Hz*)
sigma = 0.1; (*Parámetro de dispersión*)
x0 = L / 2;
y0 = L / 2;

f[x_, y_, t_] := A * Sin[2 * Pi * f0 * t] * Exp[-((x - x0)^2 + (y - y0)^2) / (2 * sigma^2)]
```

Implementación en Mathematica, método

```
(*Condiciones de frontera*)
Table[U[i, 0, k] = 0; U[i, nx, k] = 0; U[0, i, k] = 0;
|tabla
U[nx, i, k] = 0;, {i, 0, nx}, {k, 0, nt}];

(*Condiciones iniciales*)
Table[U[i, j, -1] = 0; U[i, j, 0] = 0;, {i, 0, nx}, {j, 0, nx}];
|tabla

k = 0;
While[k < nt,
|mientras
  Table[U[i, j, k + 1] = 2*U[i, j, k] - U[i, j, k - 1] + r*(U[i + 1, j, k] + U[i - 1, j, k] + U[i, j + 1, k] + U[i, j - 1, k] - 4*U[i, j, k]) +
  |tabla
    dt^2*f[x[i], y[j], t[k]], {i, 1, nx - 1}, {j, 1, nx - 1}];
  k++];
|tabla

Table[{x[i], y[j], U[i, j, 50]}, {j, 0, nx}, {i, 0, nx}];
|tabla
```

Implementación en Mathematica, solución

```
Table[{x[i], y[j], U[i, j, 50]}, {j, 0, nx}, {i, 0, nx}];  
|tabla  
]:=  
ListAnimate[Table[ListPlot3D[(*Representamos la solución*)  
|anima lista |tabla |representación 3D de lista  
Flatten[Table[{x[i], y[j], U[i, j, k]}, {i, 0, nx}, {j, 0, nx}], 1],  
|aplana |tabla  
InterpolationOrder → 3, PlotRange → {All, All, {-0.04, 0.04}}, AxesLabel → {"x", "y", "U(x, y, t)"},  
|orden de interpolación |rango de repre... |todo |todo |etiqueta de ejes  
PlotLabel → "Tiempo = " <> ToString[t[k]], PlotStyle → PointSize[0.01]], {k, 0, nt}], AnimationRate → 5]  
|etiqueta de representación |convierte a cadena ... |estilo de repr... |tamaño de punto |velocidad de animación
```

Animación de Ondas

Bibliografía

- ▶ Morton, K. W., Mayers, D. F. (2005). *Numerical Solution of Partial Differential Equations: An Introduction*. Cambridge University Press.
- ▶ Wolfram Research. (2024). *Finite Differences Method*. Recuperado de <https://reference.wolfram.com/language/FiniteDifferences>.
- ▶ Wikipedia. (2023). *Finite difference method*. Recuperado de https://en.wikipedia.org/wiki/Finite_difference_method.
- ▶ Wolfram Research. (2024). *Mathematica Documentation*. Recuperado de <https://reference.wolfram.com/language/>.

Bibliografía

- ▶ Niemeyer, K. (2024). *Mechanical Engineering Methods*. Oregon State University. Recuperado de <https://kyleniemeyer.github.io/ME373-book/content/intro.html>
- ▶ Kværnø, A. (2020). *Partial differential equations and finite difference methods*. NTNU. Recuperado de https://wiki.math.ntnu.no/_media/tma4130/2020h/pde.pdf
- ▶ *Understanding the finite difference method*. Math Stack Exchange. Recuperado de <https://math.stackexchange.com/questions/2531050/understanding-the-finite-difference-methods>
- ▶ ChatGPT. (2024). *Asistencia en la confección del código*. OpenAI.