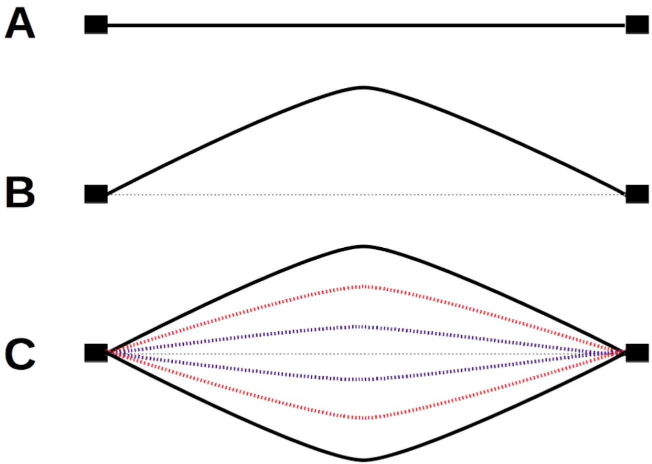
Contenido

MODELO

CONCEPTUAL

Ecuación de Onda: Modelo Conceptual

Objetivo: Modelar el movimiento de una cuerda sujeta en ambos extremos y con una condición inicial como la que se muestra en la figura B.



MODELO
MATEMÁTICO

Ecuación de Onda: Modelo Matemático

- La ecuación de onda está dada por la ecuación diferencial

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

  donde α es una constante que depende de las condiciones físicas del problema.

- Condiciones de frontera (la cuerda está fija en ambos extremos):

$$\begin{aligned} u(0,t) &= 0 \text{ para } t > 0, \\ u(l,t) &= 0 \text{ para } t > 0, \end{aligned}$$

- Condiciones iniciales:

$$\begin{aligned} u(x,0) &= f(x), \text{ para } 0 \le x \le L \quad \text{(Forma inicial de la onda)} \\ \frac{\partial u}{\partial t}(x,0) &= g(x), \text{ para } 0 \le x \le L \quad \text{(Velocidad inicial de la onda)} \end{aligned}$$

MODELO NUMÉRICO

Ecuación de Onda: Modelo Numérico

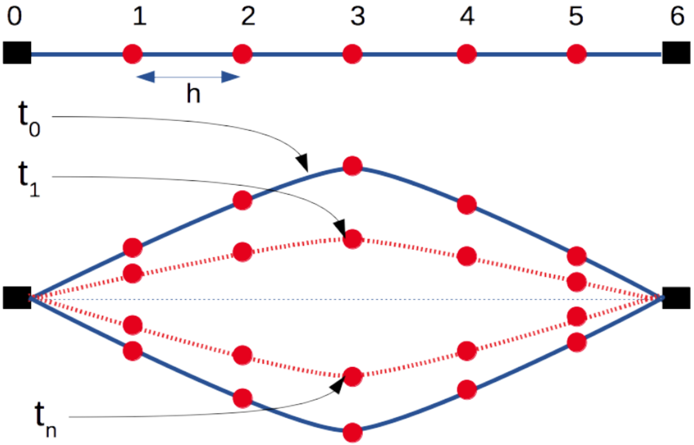
Definimos lo siguiente:

- Partición del dominio físico (construcción de la malla):
  - Definimos un entero $N > 0$ que representa el número de incógnitas en el eje x
  - Calculamos el espaciamiento mediante: $h = L/(N+1)$.
- Paso de tiempo y el número de pasos de la simulación:
  - Definimos $h_t > 0$ como paso de tiempo[1].
  - Definimos N_t como el número total de pasos de simulación.
  - Calculamos $T_{max} = h_t * N_t$ que es el tiempo total de simulación.
- Podemos entonces calcular:
  - $x_i = i * h$ para $i = 0, 1, \ldots, N+1$ y
  - $t_n = n * h_t$ para $n = 0, 1, \ldots, N_t$
- Notación:
  - $u(x_i, t_n) \equiv u_{i,n}$ para $i = 0, 1, \ldots, N+1$ y $n = 0, 1, \ldots, N_t$.

[1] Este número debe cumplir ciertos criterios de estabilidad del método.

Ecuación de Onda: Modelo Numérico

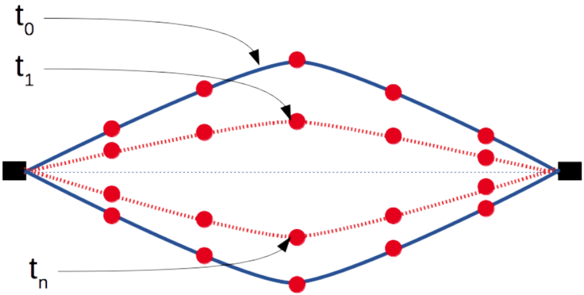
$N = 5 \qquad h = 1 / (N+1) \qquad h_t = t_{n+1} - t_n$



Ecuación de Onda: Modelo Numérico

- La ecuación de onda en un punto de la malla (espacial y temporal) se escribe como:

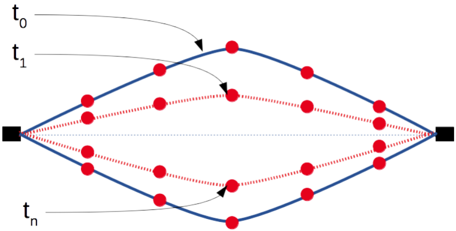
$$\frac{\partial^2 u}{\partial t^2}(x_i, t_n) - \alpha^2 \frac{\partial^2 u}{\partial x^2}(x_i, t_n) = 0,$$



Ecuación de Onda: Modelo Numérico

- Cada término de la ecuación lo aproximamos como sigue:

$$\begin{aligned}
\frac{\partial^2 u}{\partial t^2}(x_i, t_n) &= \frac{u(x_i, t_{n+1}) - 2u(x_i, t_n) + u(x_i, t_{n-1})}{h_t^2} + O(h_t^2) \\
&= \frac{u_{i,n+1} - 2u_{i,n} + u_{i,n-1}}{h_t^2} + O(h_t^2) \\
\frac{\partial^2 u}{\partial x^2}(x_i, t_n) &= \frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{h^2} + O(h^2) \\
&= \frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{h^2} + O(h^2)
\end{aligned}$$



Ecuación de Onda: Modelo Numérico

- Sustituyendo las ecuaciones en diferencias en la ecuación diferencial y despejando $u_{i,n+1}$ se obtiene:

$$\boxed{u_{i,n+1} = 2(1-\lambda^2)u_{i,n} + \lambda^2(u_{i+1,n} + u_{i-1,n}) - u_{i,n-1}} \quad (1)$$

donde $\lambda = \alpha h_t/h$ con $i = 1, \dots, N$ y $n = 1, \dots, N_t - 1$

- La ecuación (1) requiere de condiciones iniciales y de frontera:

$$\begin{aligned} u_{i,0} &= f(x_i) \quad \text{para } i = 1, \dots, N && \text{(cond. inicial)} \\ u_{0,n} &= 0 \quad \text{para } j = 1, \dots, N_t && \text{(cond. de frontera)} \\ u_{N+1,n} &= 0 \quad \text{para } j = 1, \dots, N_t && \text{(cond. de frontera)} \end{aligned}$$

- El algoritmo empieza en $n = 1$ para calcular $u_{i,2}$ para todas las i's:
  - $\boxed{u_{i,2} = 2(1-\lambda^2)u_{i,1} + \lambda^2(u_{i+1,1} + u_{i-1,1}) - u_{i,0}}$
  - Es decir, necesitamos conocer $u_{i,0}$ y $u_{i,1}$, para calcular $u_{i,2}$.
  - $u_{i,0}$ está dado por las condiciones de iniciales.
  - **¿Cómo obtenemos $u_{i,1}$?**

Ecuación de Onda: Modelo Numérico

- Para calcular $u_{i,1}$ definimos el siguiente problema de valor inicial:

$$\frac{\partial u}{\partial t}(x,0) = g(x), \text{ para } 0 \le x \le L$$

Obsérvese que este problema representa la velocidad inicial de la onda y es una condición inicial del problema.

1. Usando Forward Euler ($O(h_t)$) tenemos que:

$$\frac{u_{i,1} - u_{i,0}}{h_t} = g_i \Longrightarrow \boxed{u_{i,1} = u_{i,0} + h_t g_i}$$

2. Otra forma que proporciona una precisión de $O(h_t^3)$ es:

$$\begin{aligned} u_{i,1} &= (1-\lambda^2)u_{i,0} + \frac{\lambda^2}{2}\left[u_{i+1,0} + u_{i-1,0}\right] + h_t g_i \\ \Longrightarrow u_{i,1} &= (1-\lambda^2)f(x_i) + \frac{\lambda^2}{2}\left[f(x_{i+1}) + f(x_{i-1})\right] + h_t g_i \end{aligned}$$

donde $f(x)$ es la condición inicial (forma inicial de la onda).

En ambos casos para $i = 1, \ldots, N$.

Ejercicio 12: Ecuación de onda

Implementar la solución numérica de la ecuación de onda, presentada antes, con los siguientes parámetros:

- Longitud del dominio, $L = 1$.
- Número de incógnitas $N = 20$.
- Tiempo máximo de simulación $T_{max} = 1$.
- Paso de tiempo 0.05.
- Parámetro α igual a 2.
- Condiciones de frontera tipo Dirichlet igual a cero.
- Condición inicial: $u(x, 0) = f(x) = \sin(\pi x)$.
- Velocidad inicial: $g(x) = 0$

Este ejemplo tiene la siguiente solución analítica: $u(x, t) = \sin(\pi x)\cos(2\pi t)$

Entrege su código documentado en una notebook de nombre `E12_EcuacionDeOnda.ipynb`.

Ejercicio 12: Solución

Definición de los parámetros del problema:

- Tamaño del dominio L y número de incógnitas N.
- Tamaño de paso del tiempo, tiempo total de simulación T_{max} y número total de pasos N_t.
- Datos físicos: α.
- Cálculo de algunas constantes: h, N_t y λ.

```
L = 1          # Longitud del dominio
N = 9          # Numero de incognitas internas
Tmax = 1.0     # Tiempo maximo de simulacion
ht = 0.05      # Paso de tiempo
alpha = 2      # Dato fisico
h = L / (N+1) # Tamanio de la malla espacial

Nt = int(Tmax / ht)     # Numero total de pasos
lamb = alpha * ht / h   # Parametro lambda
Tmax = Nt * ht          # Tiempo total de simulacion
```

Ejercicio 12: Solución

Definición de algunas funciones:

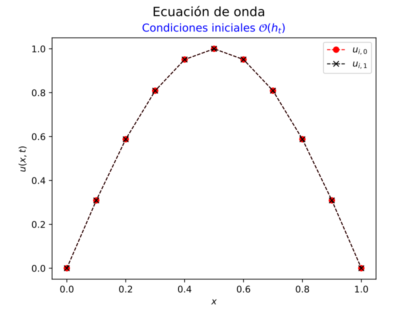
- Condición inicial: $f(x)$.
- Velocidad inicial de la onda: $g(x)$.
- Solución exacta.
- Cálculo del error.
- Condiciones iniciales.

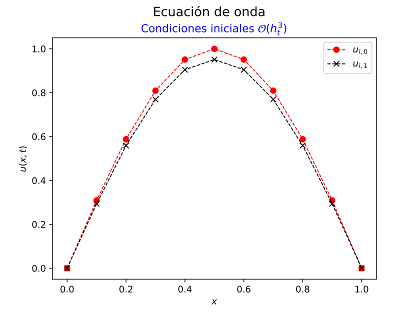
```python
def f(x):
    return np.sin(np.pi * x)

def g(x):
    return 0

def solExacta(x, t):
    return np.sin(np.pi * x) * np.cos(2 * np.pi * t)

def calcError(sol_n, sol_e):
    return np.abs(sol_n-sol_e)

def condicionesIniciales(l, ht, u, x, op=1):
    N = len(u)
    w = np.zeros(N)
    for i in range(1,N-1):
        if op == 1:
            w[i] = u[i] + ht * g(x[i])
        else:
            w[i] = (1 - l**2) * u[i] + 0.5 * l**2 * (u[i+1] + u[i-1]) + ht * g(x[i])
    return w
```

Ejercicio 12: Solución

Checar las condiciones iniciales

```
x = np.linspace(0,L,N+2)  # Coordenadas de la malla
u = f(x)                  # Condicion inicial
w = condicionesIniciales(lamb, ht, u, x, op=2) # Euler :op = 1
plt.suptitle('Ecuacion de onda', fontsize=14)
plt.plot(x, u,'ro--', lw = 1, label = "$u_{i,0}$")
plt.plot(x, w,'kx--', lw = 1, label = "$u_{i,1}$")
plt.title('Condiciones iniciales $\mathcal{O}(h_t)$', color='blue', fontsize=12)
plt.ylabel('$u(x,t)$')
plt.xlabel('$x$')
plt.savefig('condicion_03.pdf')
```





Ejercicio 12: Solución

- Cálculo de la solución numérica:

$$u_{i,n+1} = 2(1-\lambda^2)u_{i,n} + \lambda^2(u_{i+1,n} + u_{i-1,n}) - u_{i,n-1}$$

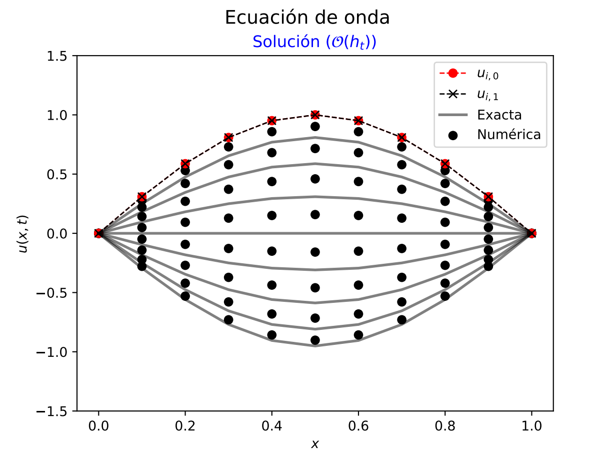
para $i = 1, \ldots, N$ y $n = 1, \ldots, N_t - 1$

```
def solver(u, w, N, x, Nt, l):
    s = np.zeros(N)
    for n in range(1,Nt):
        for i in range(1,N+1):
            s[i] = 2 * (1 - l**2) * w[i] + l**2 * (w[i+1] + w[i-1]) - u[i]
        u = w.copy()
        w = s.copy()

        plt.plot(x,s,'--')
    return s
```

Ejercicio 12: Solución

Cálculo de la solución con condición inicial $\mathcal{O}(h_t)$:

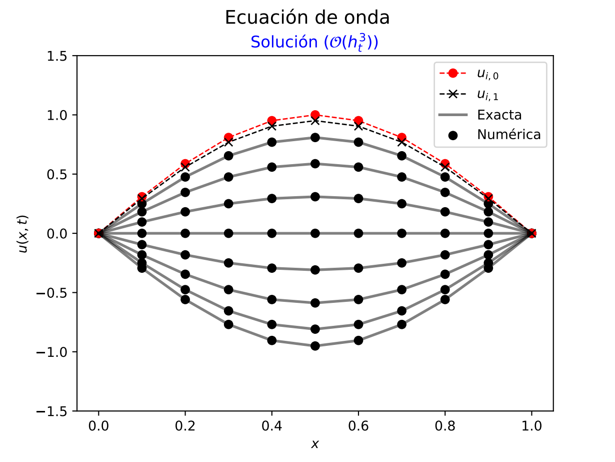
```
w = condicionesIniciales(lamb, ht, u, x, op = 1) # Euler :op = 1
s = solver(u, w, N, x, Nt, lamb)
```



Ejercicio 12: Solución

Cálculo de la solución con condición inicial $\mathcal{O}(h_t^3)$:

```
w = condicionesIniciales(lamb, ht, u, x, op = 2) # O(h^3) :op = 2
s = solver(u, w, N, x, Nt, lamb)
```



Ecuación de Onda: Estabilidad del algoritmo

Ejercicio 12: Solución

- El método es de orden $O(h^2 + h_t^2)$.
- El método converge cuando las funciones f y g son suficientemente diferenciables.
- El método explícito presentado aquí es condicionalmente estable.
- Se requiere que $\lambda = \alpha h_t / h \leq 1$ para obtener estabilidad.
- En nuestro ejemplo de calibración se tiene que $\alpha = 2$, $h_t = 0.05$ y $h = 0.1$ lo que implica que $\lambda = 1$.

Contenido

[1] Richard Burden and J. Douglas Faires
Numerical Analysis
Ninth Edition, 2011 Brooks/Cole, Cengage Learning

[2] R.J. Leveque,
Finite Difference Method for Ordinary and Partial Differential Equations: Steady State and Time-Dependent Problems ,
Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2007.

Contenido