Ecuación de Onda

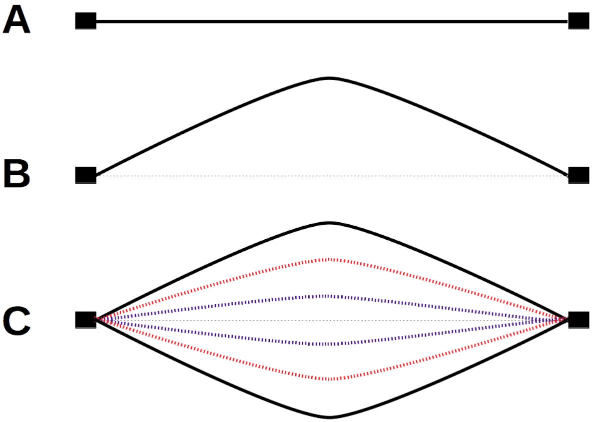
Diferencias Finitas

Luis Miguel de la Cruz Salas

Instituto de Geofísica
UNAM

Junio 2018

Ecuación de Onda: Modelo Conceptual



Ecuación de Onda: Modelo Matemático

- La ecuación de onda está dada por la ecuación diferencial

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

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

- Condiciones iniciales y de frontera:

$$u(0,t) = u(L,t) = 0 \text{ para } t > 0.$$

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

Ecuación de Onda: Modelo Matemático

- La ecuación de onda está dada por la ecuación diferencial

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

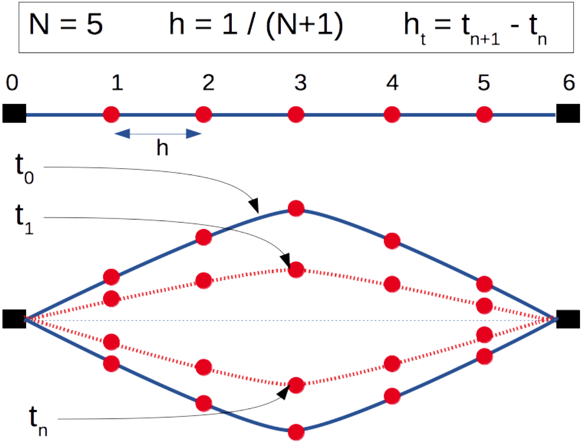
  donde α es una constante que depende de las condiciones físicas del problema.

- Condiciones iniciales y de frontera:

$$u(0,t) = u(l,t) = 0 \text{ para } t > 0,$$

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

Ecuación de Onda: Modelo Numérico



Ecuación de Onda: Modelo Numérico

- 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 $n = 0, 1, \ldots, N_t$

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

Ecuación de Onda: Modelo Computacional

Ejemplo de calibración

Aproximar la solución a la ecuación de onda con los siguientes parámetros:

- Longitud del dominio, L, igual a 1.
- Número de incógnitas $N = 9$.
- 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 solución analítica igual a: $u(x, t) = sin(\pi x)cos(2\pi t)$

Ecuación de Onda: Modelo Computacional

- Datos de entrada:
  - Tamaño del dominio L, número de incógnitas N.
  - Tamaño de paso del tiempo y número total de pasos N_t (o tiempo total de simulación T_{max}).
  - 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
Nt = int(Tmax / ht)      # Numero total de pasos
lamb = alpha * ht / h    # Parametro lambda
Tmax = Nt * ht           # Tiempo total de simulacion
```

Ecuación de Onda: Modelo Computacional

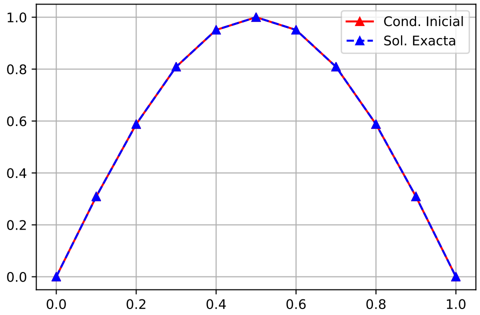
- Definición de Funciones
  - $f(x), g(x)$, solución exacta y cálculo del error.

```
import numpy as np
import matplotlib.pyplot as plt

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)
```

Ecuación de Onda: Modelo Computacional

- Prueba de Funciones

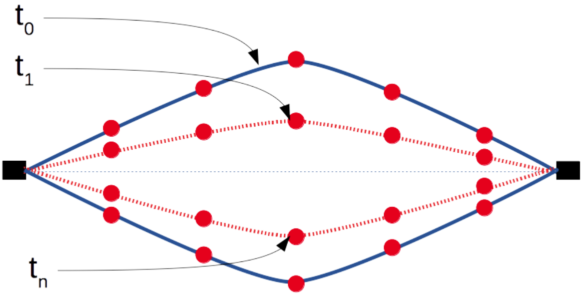
```
x = np.linspace(0,L,N+2)   # Definicion de la malla
u = f(x)                   # Condicion inicial

plt.plot(x, u,'^r-', label = "Cond. Inicial")
plt.plot(x, solExacta(x,Tmax),'^b--', label = "Sol. Exacta")
```



Ecuación de Onda: Modelo Numérico

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

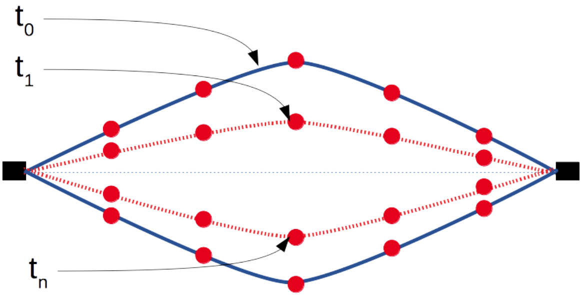
$$\frac{\partial^2 u}{\partial t^2}(x_i, t_n) - \alpha^2 \frac{\partial^2 u}{\partial x^2}(x_i, t_n) = 0,$$



- Cada término de la ecuación lo aproximamos como sigue:

$$\frac{\partial^2 u}{\partial t^2}(x_i, t_n) = \frac{u(x_i, t_{n+1}) - 2u(x_i, t_n) + u(x_i, t_{n-1})}{h_t^2} + O(h_t^2)$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_n) = \frac{u(x_{i+1}, t_n) - 2u(x_i, t_n) + u(x_{i-1}, t_n)}{h^2} + O(h^2)$$



Ecuación de Onda: Modelo Numérico

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

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

  donde $\lambda = \alpha h_t/h$ con $i = 1, \dots, N$ y $n = 1, \dots, N_t - 1$
- En la ecuación (1) se observa que :
  - El algoritmo empieza en $n = 1$ para calcular $u_{i,2}$.
  - En general, el cálculo de $u_{i,n+1}$ requiere de u en los instantes n y $n-1$, es decir, en dos tiempos previos.
- Condiciones iniciales y de frontera:

$$\begin{aligned} u_{i,0} &= f(x_i) \qquad \text{para} \quad i = 1, \dots, N \\ u_{0,n} = u(N+1, n) &= 0 \qquad \text{para} \quad j = 1, \dots, N_t \end{aligned}$$

Ecuación de Onda: Modelo Numérico

- Tenemos que $u_{i,0}$ está dado por las condiciones de iniciales.
- Problema de valor inicial (dado por las condiciones iniciales):

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

  - Euler ($O(h_t)$):

$$\boxed{u_{i,1} = u_{i,0} + h_t g(x_i)}$$

  - Otra opción ($O(h_t^3)$) :

$$\boxed{u_{i,1} = (1-\lambda^2) f(x_i) + \frac{\lambda^2}{2}\left[f(x_{i+1}) + f(x_{i-1})\right] + h_t g(x_i)}$$

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

Ecuación de Onda: Modelo Computacional

- Cálculo de las condiciones iniciales:
  - $u_{i,1} = u_{i,0} + h_t g(x_i)$
  - $u_{i,1} = (1-\lambda^2)f(x_i) + \frac{\lambda^2}{2}\left[f(x_{i+1}) + f(x_{i-1})\right] + h_t g(x_i)$

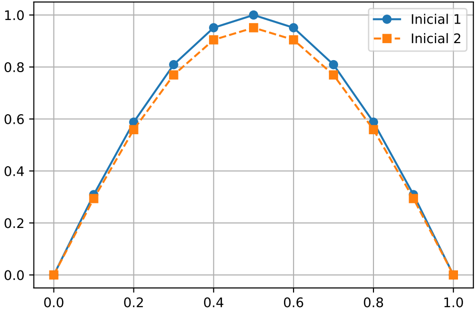
  (En ambos casos para $i = 1, \ldots, N$)

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

Ecuación de Onda: Modelo Computacional

- Prueba del cálculo de la condición inicial para $n = 1$.

```
w = condicionesIniciales(lamb, ht, u, x)

plt.plot(x,u,'o-', label="Inicial 1")
plt.plot(x,w,'s--', label="Inicial 2")
```



Ecuación de Onda: Modelo Computacional

- Cálculo de la solución:

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

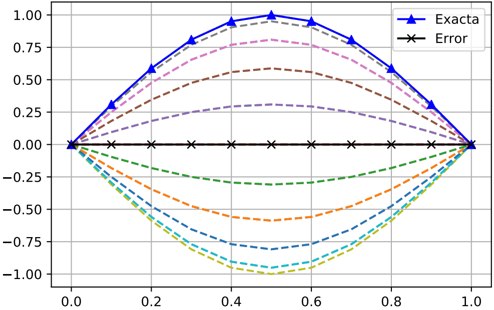
para $i = 1, \ldots, N$ y $n = 1, \ldots, N_t - 1$

```
def solver(u, w, N, x, Nt, l):
    s = np.zeros(N+2)
    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
```

Ecuación de Onda: Modelo Computacional

- Prueba del cálculo de la solución:

```
s = solver(u, w, N, x, Nt, lamb)

plt.plot(x,solExacta(x,Tmax),'^b-',label="Exacta")
plt.plot(x,calcError(s,solExacta(x,Tmax)),'xk-',label="Error")
```



Ecuación de Onda: Estabilidad del algoritmo

- 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 $\lambda = 1$.

Bibliografía I

[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.