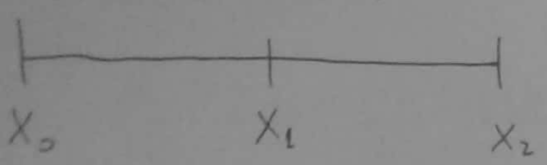
Ayudantía 7

1.- Recordemos que el polinomio interpolador de Lagrange es de la forma:

$$P_n(x) = \sum_{i=1}^{n}\left[y_i \cdot \prod_{\substack{j=1 \\ j\neq i}}^{n}\left(\frac{x-x_j}{x_i-x_j}\right)\right]$$

Con n el número de puntos.

Luego, si se desea realizar una interpolación de una función cualquiera $u(x)$, $x\in\mathbb{R}$ en tres puntos equiespaci



Donde:

$$u(x_i) = u_i$$

El polinomio interpolador corresponde a:

$$P_3(x) = \sum_{i=0}^{2}\left[u_i \prod_{\substack{j=0 \\ j\neq 1}}^{2}\left(\frac{x-x_j}{x_i-x_j}\right)\right]$$

$$= u_0\cdot\left(\frac{x-x_1}{x_0-x_1}\right)\cdot\left(\frac{x-x_2}{x_0-x_2}\right) + u_1\cdot\left(\frac{x-x_0}{x_1-x_0}\right)\cdot\left(\frac{x-x_2}{x_1-x_2}\right)$$

$$+ u_2\cdot\left(\frac{x-x_0}{x_2-x_0}\right)\cdot\left(\frac{x-x_1}{x_2-x_1}\right)$$

Entonces, si definimos h como el espacio entre puntos.

$$h = (x_1-x_0) = (x_2-x_1) \qquad 2h = (x_2-x_0)$$

$$-h = (x_0-x_1) = (x_1-x_2) \qquad -2h = (x_0-x_2)$$

Reemplazando:

$$P_3(x) = \frac{u_0}{2h^2}\cdot(x-x_1)\cdot(x-x_2) + \frac{u_1}{(-h^2)}(x-x_0)\cdot(x-x_2) + \frac{u_2}{2h^2}\cdot(x-x_0)(x-x_1)$$

Luego, derivando respecto a x.

$$\frac{dP_3(x)}{dx} = \frac{u_0}{2h^2}\left[x-x_1+x-x_2\right] - \frac{u_1}{h^2}\left[x-x_0+x-x_2\right] + \frac{u_2}{2h^2}\left[x-x_0+x-x_1\right]$$

Evaluando en el pto. de la izquierda (x_0).

$$\frac{dP_3(x_0)}{dx} = \frac{u_0}{2h^2}\left[x_0-x_1+x_0-x_2\right] - \frac{u_1}{h^2}\left[x_0-x_0+x_0-x_2\right] + \frac{u_2}{2h^2}\left[x_0-x_0+x_0-x_1\right]$$

$$= \frac{u_0}{2h^2}(-3h) - \frac{u_1}{h^2}(-2h) + \frac{u_2}{2h^2}(-h) = \frac{-3u_0}{2h} + \frac{2u_1}{h} - \frac{u_2}{2h}$$

$$\frac{dP_3(x_0)}{dx} = \frac{-3u_0+4u_1-u_2}{2h}$$

Considerando que:

$$u(x) = P_3(x) + error_3(x) \rightarrow \frac{du(x)}{dx} = \frac{dP_3(x)}{dx} + \frac{d(error_3(x))}{dx}$$

$$\frac{du(x_0)}{dx} = \frac{dP_3(x_0)}{dx} + \frac{d(error_3(x_0))}{dx}$$

$$\downarrow$$

$$u'(x_0) \approx \frac{-3u_0+4u_1-u_2}{2h}$$

Ahora, respecto al error se tiene que el error para n puntos:

$$error_n(x) = \frac{f^{(n)}(c)}{n!}\prod_{i=1}^{n}(x-x_i),$$ con $c \in [x_1, x_n]$ valor que maximiza $f^{(n)}$.

Scanned by CamScanner

Entonces, el error con 3 puntos corresponde a:

$$\text{error}_3(x) = \frac{f^{(3)}(c)}{3!} \cdot (x-x_0)\cdot(x-x_1)(x-x_2)$$

La derivada:

$$\frac{d(\text{error}_3(x))}{dx} = \frac{f^{(3)}(c)}{3!}\left[(x-x_1)\cdot(x-x_2)+(x-x_0)\cdot(x-x_1)+(x-x_0)(x-x_2)\right]$$

En $x=x_0$:

$$\frac{d(\text{error}_3(x_0))}{dx} = \frac{f^{(3)}(c)}{3!}\left[(x_0-x_1)\cdot(x_0-x_2)+(x_0-x_0)(x_0-x_1)+(x_0-x_0)\cdot(x_0-x_2)\right]$$

$$= \frac{f^{(3)}(c)}{6}(-h)(-2h) = \frac{h^2}{3}f^{(3)}(c)$$

Luego, el método derivado es de segundo orden.

$$u'(x_0) = \frac{-3u_0+4u_1-u_2}{2h} + \frac{h^2}{3}f^{(3)}(c)$$

De manera general, el método:

$$u'(x_i) = \frac{-3u_i+4u_{i+1}-u_{i+2}}{2h} + \frac{h^2}{3}f^{(3)}(c)$$

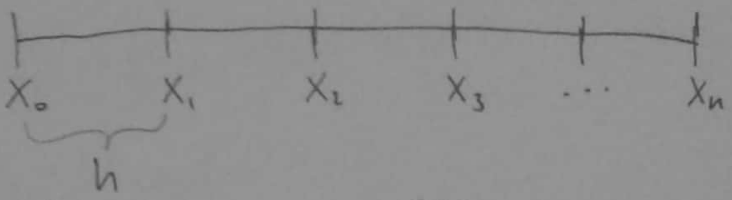
Scanned by CamScanner

2.- a) Se utilizarán aproximaciones de diferencias finitas.

$$\Delta u = \exp(-x^2)$$

$$\frac{d^2 u(x)}{dx^2} = \exp(-x^2) \quad (1)$$

$u(x_i) = u_i$, $x_i = ih$ Con h el espaciado en x.



Luego, la discretización de la ecuación (1).

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = \exp(-(ih)^2)$$

$$u_{i+1} - 2u_i + u_{i-1} = h^2 \exp(-(ih)^2)$$

En el borde izquierdo del intervalo solo se tiene el valor de la derivada (Condición de Neumann). Por esto se realizará la siguiente aproximación:

$$u'_i = \frac{-3u_i + 4u_{i+1} - u_{i+2}}{2h} + \frac{h^2}{3} f^{(3)}(c)$$

Sabiendo del enunciado que $u'(0) = -1$:

$$u'_0 \approx \frac{-3u_0 + 4u_1 - u_2}{2h} \rightarrow -1 \approx \frac{-3u_0 + 4u_1 - u_2}{2h}$$

De la expresión se puede despejar u_0.

$$u_0 \approx \frac{4u_1 - u_2 + 2h}{3}$$

Scanned by CamScanner

Con la información recopilada es posible expresar un sistema de ecuaciones, con incógnitas los valores de u en $x_1, \dots, x_{n-1}$.

Las ecuaciones son las siguientes:

$$i=1 \rightarrow u_2 - 2u_1 + u_0 = h^2 \exp(-h^2)$$

Para u_0 utilizamos la aproximación anterior.

$$u_2 - 2u_1 + \frac{4u_1}{3} - \frac{u_2}{3} + \frac{2h}{3} = h^2 \exp(-h^2)$$

$$u_1 \cdot \left(-\frac{2}{3}\right) + u_2 \cdot \left(\frac{2}{3}\right) = h^2 \exp(-h^2) - \frac{2h}{3}$$

$$i=2 \rightarrow u_3 - 2u_2 + u_1 = h^2 \exp(-(2h)^2)$$

$$i=3 \rightarrow u_4 - 2u_3 + u_2 = h^2 \exp(-(3h)^2)$$

$$\vdots$$

$$i=n-1 \rightarrow u_n - 2u_{n-1} + u_{n-2} = h^2 \exp(-((n-1)h)^2)$$

Pero u_n es conocido: $u_n = 1$

$$-2u_{n-1} + u_{n-2} = h^2 \exp(-((n-1)h)^2) - 1$$

Así el sistema matricial:

$$\begin{bmatrix} -\frac{2}{3} & \frac{2}{3} & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & & 0 \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-1} \end{bmatrix} = \begin{bmatrix} h^2 \exp(-h^2) - \frac{2h}{3} \\ h^2 \exp(-(2h)^2) \\ h^2 \exp(-(3h)^2) \\ \vdots \\ h^2 \exp(-((n-1)h)^2) - 1 \end{bmatrix}$$

Scanned by CamScanner

Luego, un algoritmo para resolver el sistema:

```
def Poisson_solver(a,b,n
    A = tri_diag((1,-2,1), n-1)
    A[0][0] = -2/3
    A[0][1] = 2/3
    b = vector_zeros(n-1)  <——  h=(b-a)/n
    for i in range(1,n):
        b[i-1] = h^2 · exp(-(ih)^2)
    end for.
    b[0] -= 2h/3
    b[n-1] -= 1
    u = Linear_Solve(A, b)
    return u
```

b) Para estimar el borde i39. se utilizó un método de segundo orden. Por lo que el método en general debería ser de segundo orden también, ya que la segunda derivada se aproximó con un método de 2do orden.

* Se aconseja utilizar siempre métodos del mismo orden.
· fuente: math.stackexchange.com

Scanned by CamScanner

3.- a) Ignorando el roce la expresión (5) queda:

$$\ddot{\theta}(t) = -\frac{g}{L}\,\text{sen}(\theta(t)), \quad \theta(0) = \theta_0, \; \theta(T) = \theta_T$$

Luego, para resolver el problema se pueden considerar dos caminos:

> $\text{sen}(\theta(t)) \approx \theta(t)$, $\theta(t) \to 0$

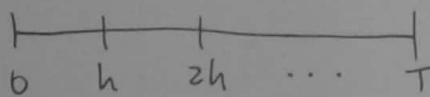
En este caso el problema a resolver es:

$$\ddot{\theta}(t) = -\frac{g}{L}\theta(t)$$

Lo cual es un BVP genérico.

> $\text{sen}(\theta(t)) = \text{sen}(\theta(t))$, $\theta(t) \to 0$.

Este es el caso que consideraremos, por lo que la discretización:



$\to t_i = ih \to \theta(t_i) = \theta_i$

Con el método de diferencias finitas la expresión (5) se aproxima como:

$$\frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{h^2} = \frac{-g}{L}\,\text{sen}(\theta_i) \to \theta_{i+1} - 2\theta_i + \theta_{i-1} = \frac{-gh^2}{L}\,\text{sen}(\theta_i)$$

$$\to \theta_{i+1} - 2\theta_i + \frac{gh^2}{L}\,\text{sen}(\theta_i) + \theta_{i-1} = 0 \quad / \quad \frac{gh^2}{L} = \sigma$$

Luego, las ecuaciones son $n-1$ de la forma:

$i=1 \quad \theta_2 - 2\theta_1 + \sigma\,\text{sen}(\theta_1) + \theta_0 = 0$

$i=2 \quad \theta_3 - 2\theta_2 + \sigma\,\text{sen}(\theta_2) + \theta_1 = 0$

$\vdots \qquad \vdots$

$i=n-1 \quad \theta_T - 2\theta_{n-1} + \sigma\,\text{sen}(\theta_{n-1}) + \theta_{n-2} = 0$

Scanned by CamScanner

Como hemos visto en ayudantías anteriores este tipo de problemas se pueden resolver con el método de Newton Multivariado.

De la forma:

$$F(\vec{\theta}) = \begin{bmatrix} \theta_2 - 2\theta_1 + \tau \operatorname{sen}(\theta_1) + \theta_0 \\ \vdots \\ \theta_T - 2\theta_{n-1} + \tau \operatorname{sen}(\theta_{n-1}) + \theta_{n-2} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \vec{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n-1} \end{bmatrix}$$

$$DF(\vec{\theta}) = \begin{bmatrix} -2+\tau\cos(\theta_1) & 1 & 0 & 0 & \cdots & 0 \\ 1 & -2+\tau\cos(\theta_2) & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2+\tau\cos(\theta_3) & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & & 1 & -2+\tau\cos(\theta_{n-1}) \end{bmatrix}$$

Luego, el método de newton multivariado:

$$DF(\vec{\theta}_k)\cdot\Delta\vec{\theta} = -F(\vec{\theta}_k), \quad \Delta\vec{\theta} = \vec{\theta}_{k+1} - \vec{\theta}_k$$

Esto se resuelve con un solve lineal. Es un sistema de ecuaciones.

Una vez que se tenga un $\vec{\theta}$ lo suficientemente preciso, se debe calcular la derivada en t_0, para estimar la velocidad angular inicial.

$$\theta_0' = \frac{-3\theta_0 + 4\theta_1 - \theta_2}{2h}$$

Un pseudo código:

```
def péndulo(θ₀, θ_T, T, n, g, L):
    h = T/n
    σ = gh²/L
    def DF(v):
        A = tri_diag(1,0,1)
        for i in range(1,n):
            A[i-1][i-1] = -2 + σ cos(v[i-1])
        end for
        return A.

    def F(v):
        b = vector(n-1)
        b[0] = v[1] - 2v[0] + σ sen(v[0]) + θ₀
        b[n-2] = θ_T - 2v[n-2] + σ sen(v[n-2]) + v[n-3]
        for i in range(1,n-2):
            b[i] = v[i+1] - 2v[i] + σ sen(v[i]) + v[i-1]
        end for.
        return b.

    θ = vector_ones(n-1)
    A = DF(θ)   b = F(θ)
    Solve_linear(A,-b) → Δθ
    while Δθ.modulo > tolerancia:
        θ += Δθ
        A = DF(θ)   b = F(θ)
        Solve_linear(A,-b) → Δθ
    end while.
```

Scanned by CamScanner

$\theta'_0 = (-3\theta_0 + 4\theta[0] - \theta[1])/2h$

return θ'_0.

Scanned by CamScanner