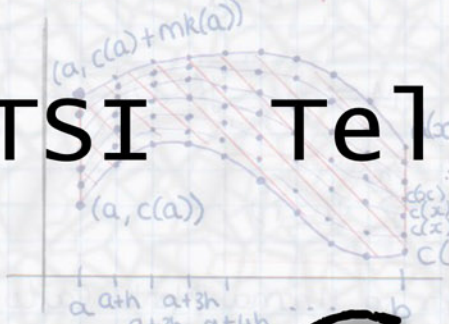
ETSI Telecomunicación

Cálculo Numérico



Apuntes de Pak

Cálculo Numérico

Apuntes de Pak (Francisco José Rodríguez Fortuño)
ETSI Telecomunicación. Universidad Politécnica de Valencia.
Segundo cuatrimestre de 2º curso
Curso 2004/2005

Contenido:

Apuntes extensos de la asignatura.

Fecha de última actualización: 30 Julio 2007

CÁLCULO NUMÉRICO

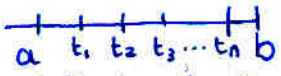
TEMA 1. SISTEMAS DE ECUACIONES NO LINEALES

Surgen frecuentemente:

ej. Discretización de problemas de frontera.

$$\begin{cases} u'' = f(t,u) & t\in[a,b] \\ u(0) = \alpha \\ u(1) = \beta \end{cases}$$

1. Discretización de la variable independiente $t_j = j\cdot h + a$



h ↔ paso $h = \frac{b-a}{n+1}$

2. Aproximación de la derivada:

$$u'(x) \simeq \frac{u(x+h)-u(x)}{h}$$

$$u''(x) \simeq \frac{u'(x)-u'(x-h)}{h} = \frac{u(x-h)-2u(x)+u(x+h)}{h^2}$$

3. Sustitución en el problema de frontera:

$$\frac{u(t_{j-1})-2u(t_j)+u(t_{j+1})}{h^2} = f(t_j, u(t_j)) + \underbrace{r(t_j,h)}_{\text{error}}$$

despreciando el error y pasando h^2 mult:

$$\begin{cases} x_{j-1} - 2x_j + x_{j+1} = h^2 f(t_j, x_j) & j = 1,2,\dots,n \\ x_0 = \alpha \\ x_{n+1} = \beta \end{cases}$$

Queda el sistema de ecuaciones no lineal

$$\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} -f(t_1,x_1) + \frac{\alpha}{h^2} \\ -f(t_2,x_2) \\ \vdots \\ -f(t_{n-1},x_{n-1}) \\ -f(t_n,x_n) + \frac{\beta}{h^2} \end{pmatrix} \cdot h^2$$

$$A \cdot x = \phi(x)$$

ej. ajuste mínimo cuadrático

Año:	1993	1994	1995	···	a_j
Abonados (millones):	0'1	0'2	0'4	···	b_j

ajuste: $y = ke^{bt}$

queremos minimizar el error cuadrático $g(k,b) = \sum (b_j - ke^{ba_j})^2$

para ello hallamos $\begin{cases} f_1(k,b) = \frac{\partial g}{\partial k} = 0 \\ f_2(k,b) = \frac{\partial g}{\partial b} = 0 \end{cases}$

$$\Rightarrow \left.\begin{aligned} &\sum[(b_j - ke^{ba_j})\cdot e^{ba_j}] = 0 \\ &\sum[(b_j - ke^{ba_j})\cdot ka_j e^{ba_j}] = 0 \end{aligned}\right\} \text{ sistema no lineal}$$

ej. problemas de optimización

Dos productos: ① precio p_1 cantidad q_1
② precio p_2 cantidad q_2

Demanda: $q_1 = 680 - 5p_1 - 3p_2$
$q_2 = 430 - 3p_1 - 2p_2$

Coste de producción: $C(q_1, q_2) = \frac{1}{6} q_1^3 - 4q_2^2 - 10q_1 + 90$

Beneficios = Ingresos - coste
$B(q_1, q_2) = I(q_1, q_2) - C(q_1, q_2)$
↳ $I(q_1, q_2) = p_1 q_1 + p_2 q_2$

1. Despejar p_1 y p_2 en función de q_1 y q_2

$$p_1 = 70 + 3q_2 - 2q_1$$
$$p_2 = 110 + 3q_1 - 5q_2$$

2. Sustituir en $I(q_1, q_2) = 70q_1 - 2q_1^2 + 6q_1q_2 + 110q_2 - 5q_2^2$

$\Rightarrow B(q_1, q_2) = I(q_1, q_2) - C(q_1, q_2)$

$$= -\frac{1}{6}q_1^3 - 2q_1^2 - q_2^2 + 6q_1q_2 + 80q_1 + 110q_2 - 90$$

maximizar beneficios.

$$\begin{cases} \frac{\partial B}{\partial q_1} = -0'5q_1^2 - 4q_1 + 80 + 6q_2 = 0 \\ \frac{\partial B}{\partial q_2} = 6q_1 + 110 - 2q_2 = 0 \end{cases}$$

sistema no lineal

Notación

$$\left.\begin{array}{l} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{array}\right\}$$

se abrevia:

$$F(x) = 0 \qquad \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \xrightarrow{F} \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{pmatrix}$$

Método del punto fijo (pag 16)

- **Punto fijo de una variable**

$f(x) = 0 \xrightarrow[x]{\text{despejar}} g(x) = x$ según $g(x)$ el método convergerá más o menos.

En cualquier caso podemos tomar $g(x) = kf(x) + x = x$

estimación inicial x_1

$g(x_1) = x_2 \to g(x_2) = x_3 \to g(x_3) = x_4 \to \dots \; g(x_k) \simeq x_k$

$\Rightarrow x^* \simeq x_k$ (punto fijo)

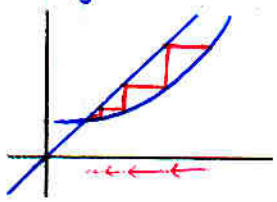
Teorema

Sea x^* punto fijo de $g(x)$ en $[a,b]$

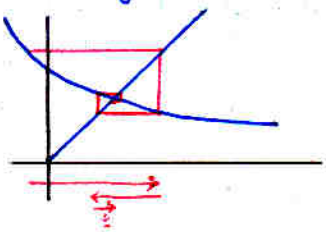
g continuam. diferenciable, $|g'(x)| < 1 \quad x \in]a,b[$ $\Rightarrow$ la iteración converge a x^* para cualquier estimación inicial

Casos.

$0 < g'(x^*) < 1$



$-1 < g'(x^*) < 0$



$0 < |g'(x^*)| < 1 \Rightarrow$ converge linealmente

$|g'(x^*)| = 0 \Rightarrow$ converge cuadraticamente

- **Generalización a $\mathbb{R}^n$**

$$\left\{ \begin{array}{l} f_1(x_1, x_2, x_3) = 0 \\ f_2(x_1, x_2, x_3) = 0 \\ f_3(x_1, x_2, x_3) = 0 \end{array} \right\} \qquad F(x) : \mathbb{R}^3 \to \mathbb{R}^3$$

$F(x) = 0$

se despeja cada una de las incógnitas de la ecuación que queramos

$$\left\{ \begin{array}{l} x_1 = g_1(x_1, x_2, x_3) \\ x_2 = g_2(x_1, x_2, x_3) \\ x_3 = g_3(x_1, x_2, x_3) \end{array} \right\}$$

$x = G(x)$

estimación inicial $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})$

$x^{(1)} = G(x^{(0)})$

$x^{(2)} = G(x^{(1)})$

hasta que

$\| x^{(k+1)} - x^{(k)} \| < tol$

habrá varias formas, algunas pueden ser divergentes y otras no $\Rightarrow$ (ir probando)

Convergencia

Para ello hay que calcular el error $e_k = \| x_k - x_{k-1} \|$

(MATLAB: diff(x))
(MATLAB: $\|\cdot\|$ = norm(x))

Lineal: $\frac{e_{k+1}}{e_k} \to K < 1 \equiv \|x_{k+1} - x_k\| < \|x_k - x_{k-1}\|$

Superlineal: $\frac{e_{k+1}}{e_k} \to 0$ un poco más que lineal

Cuadrática: $\frac{e_{k+1}}{e_k^2} \to K < 1 \equiv \|x_{k+1} - x_k\| < \|x_k - x_{k-1}\|^2$

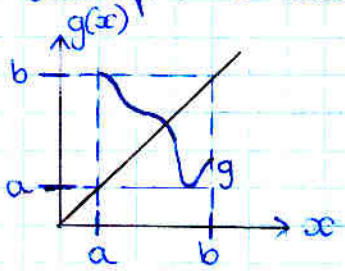
Punto Fijo: Existencia

$G: D \subset \mathbb{R}^n \to \mathbb{R}^n$ continua en D

$G(D) \to D$ la imagen está dentro del mismo subconjunto

$\Longrightarrow$ G tiene un punto fijo en D: $\exists p \in D \ / \ G(p) = p$

dem: no se puede dibujar una función continua de [a,b] en [a,b] sin cortar al menos una vez con la bisectriz del primer cuadrante



esto se puede extender a $\mathbb{R}^n$

Punto Fijo: Convergencia

si $\exists \ M<1$ tal que $\frac{\partial g_i(x)}{\partial x_j} \leq \frac{M}{n}$ para cada $i = 1,2,\dots,n$ con cada $j = 1,2,3,\dots,n$

hay que calcular n^2 derivadas

$\Longrightarrow$ la sucesión $\{x^{(k)}\}_{k=0}^{\infty}$ definida $x_{k+1} = g(x_k)$ y con $x^{(0)} \in D$ arbitrario converge al único punto fijo p

i.e. converge para cualquier estimación inicial y tiene UNICO punto fijo

además: $\|x^{(k)} - p\|_\infty \leq \frac{M^k}{1-M} \|x^{(1)} - x^{(0)}\|_\infty$

ejemplo punto fijo:

$$\left\{\begin{array}{l} x_1 x_2 + x_3 = 0 \\ x_1 \cos x_2 - e^{x_3} = 0 \\ x_1 x_3 - \ln x_2 = 0 \end{array}\right\} \quad F(x) = 0$$

↓ despejar cada incógnita en la ecuación que queramos

$$\left\{\begin{array}{l} x_1 = \frac{1}{3}\cos(x_2 x_3) + \frac{1}{6} = g_1(x_1, x_2, x_3) \\ x_2 = \frac{1}{9}\sqrt{x_1^2 + \operatorname{sen}(x_3) + 1'06} - 0'1 = g_2(x_1, x_2, x_3) \\ x_3 = -\frac{1}{20} e^{-x_1 x_2} - \frac{10\pi - 3}{3} = g_3(x_1, x_2, x_3) \end{array}\right\} \quad x = G(x)$$

si trabajamos en $D = \{ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3 \ / \ -1 \le x_i < 1 \}$

aplicamos teorema de existencia: - G continuas

¿ $G(D) \subset D$? Hay que comprobarlo para cada g_i

$$|g_1(x_1,x_2,x_3)| = \left|\frac{\cos(x_2 x_3)}{3} + \frac{1}{6}\right| \le \frac{1}{3}|\cos(x_2 x_3)| + \frac{1}{6} = \frac{1}{3} + \frac{1}{6} < 1$$

(desigualdad triangular)

$$|g_2(x_1,x_2,x_3)| = \left|\frac{1}{9}\sqrt{x_1^2 + \operatorname{sen}(x_3) + 1'06} - 0'1\right|$$

$$\le \frac{1}{9}\sqrt{1 + 1 + 1'06} + 0'1 = 0'29 < 1$$

(recuerda $D \equiv |x_i| < 1$)

cumple $G(D) \subset D$; $|G(|x|<1)| < 1$

$$|g_3(x_1,x_2,x_3)| = \ldots < 1$$

para aplicar el T^a de conv. ∀ estimación inicial y punto fijo único habría que calcular

$$\frac{\partial g_1}{\partial x_1}, \frac{\partial g_1}{\partial x_2}, \frac{\partial g_1}{\partial x_3}, \frac{\partial g_2}{\partial x_1}, \frac{\partial g_2}{\partial x_2}, \frac{\partial g_2}{\partial x_3}, \frac{\partial g_3}{\partial x_1}, \frac{\partial g_3}{\partial x_2}, \frac{\partial g_3}{\partial x_3} \le \frac{M<1}{n=3}$$

tomamos una estimación inicial ej. $x^{(0)} = \begin{pmatrix} 0.1 \\ 0.1 \\ -0.1 \end{pmatrix}$

$$x^{(1)} = G(x^{(0)}) = \begin{pmatrix} 0'499983 \\ 0'022220 \\ -0'523046 \end{pmatrix}$$

$$x^{(2)} = G(x^{(1)}) = \ldots$$

hasta que $\| G(x^{(k)}) - x^{(k)} \| < tol$

$\equiv \| x^{(k+1)} - x^{(k)} \| < tol$

↑ en MATLAB es norm

Variaciones en el método

→ Desplazamientos simultaneos

Es la forma lógica que hace $\vec{x}_k = \vec{G}(\vec{x}_{k-1})$

en ej.
$$\begin{cases} x_1^{(k+1)} = g(x_2^{(k)}, x_3^{(k)}, x_1^{(k)}) \\ x_2^{(k+1)} = g(x_1^{(k)}, x_3^{(k)}, x_2^{(k)}) \\ x_3^{(k+1)} = g(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}) \end{cases}$$

pero se puede introducir una mejoria que en general acelera la convergencia, aunque en algunos casos puede estropearla.

→ Desplazamientos sucesivos

se trata en ir utilizando las estimaciones ya obtenidas en el paso actual $x_1^{(k)}, x_2^{(k)}$... en lugar de utilizar las del paso anterior $x_1^{(k-1)}, x_2^{(k-1)}, \ldots$
Esto implica que en MATLAB haya que calcular cada g_i por separado sin poder hacerlo vectorialmente.

en ej.
$$\begin{cases} x_1^{(k+1)} = g_1(x_2^{(k)}, x_3^{(k)}, x_1^{(k)}) \\ x_2^{(k+1)} = g_2(x_1^{(k+1)}, x_3^{(k)}, x_2^{(k)}) \\ x_3^{(k+1)} = g_3(x_1^{(k+1)}, x_2^{(k+1)}, x_3^{(k)}) \end{cases}$$

en la primera evaluación no hay más remedio que usar las x_i del paso anterior

en esta segunda componente g_2 ya podemos usar el $x_1^{(k+1)}$ calculado en este mismo paso

en g_3 podemos usar $x_1^{(k+1)}$ y $x_2^{(k+1)}$ generadas en este ultimo paso.

Función en MATLAB

```
function x = ptijo (g, x0, tol, maxiter)

    iter = 0;
    incr = tol+1;

    while  incr > tol & iter < maxiter
           x = feval (g, x0);
           incr = norm(x-x0);
           iter = iter+1;
           x0 = x
    end
```

g estará definida en una función .m

para el método de desplazamientos sucesivos la función será la misma, lo que cambiará es la función de g.

Método de Newton

- una variable

El desarrollo de Taylor de $f(x)$ en torno a x_0 es:

$$f(x) = f(x_0) + (x-x_0) f'(x_0) + \frac{1}{2}(x-x_0) f''(\xi)$$

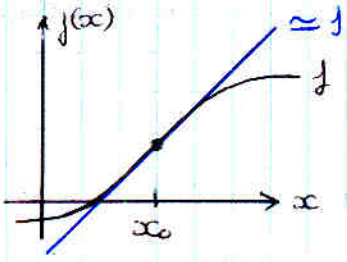
$\xi \in]x, x_0[$

si despreciamos el término de segundo orden, se obtien la aproximación a $f(x)$ como su recta tangente en x_0

$$f(x) \simeq f(x_0) + (x-x_0) f'(x_0)$$

$= \underbrace{b}_{\text{valor en } (x-x_0)=0} + \underbrace{(x-x_0)}_{\text{desplaz}} \underbrace{a}_{\text{pendiente}}$

ecuación de una recta tangente a f en x_0



lo interesante es cuando tomamos una raiz de $f(x)$ es decir $\bar{x}$ tal que $f(\bar{x}) = 0$

Sustituyendo en la fórmula:

$$0 \simeq f(x_0) + \underbrace{(\bar{x} - x_0)}_{\text{lo llamamos } h} f'(x_0)$$

$$h = -\frac{f(x_0)}{f'(x_0)}$$

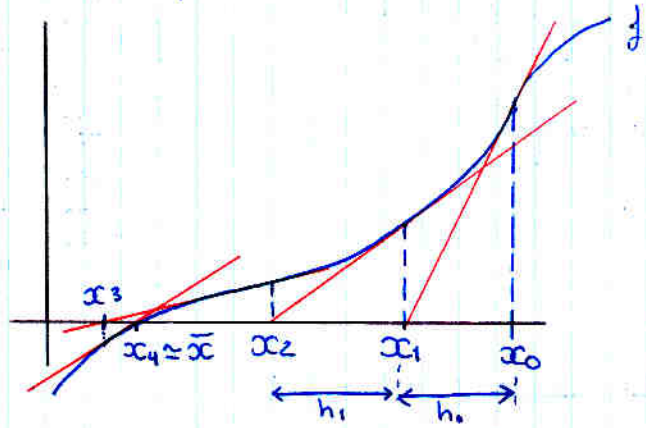
donde x_0 era una primera estimación de $\bar{x}$, ahora conociendo h podemos mejorarla

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

iterativamente:

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}$$

gráficamente, lo que estamos haciendo es:



Desventajas:
- además de evaluar la función, hay que evaluar su derivada
- la derivada no se puede anular en el conjunto en que trabajamos.

- generalización a sistemas (pag 20)

unidimensional NR 2 ??? p 26

con una variable hacíamos:

$$0 = f(x) \simeq f(x_0) + f'(x_0)(x - x_0)$$

ahora tomemos el ejemplo $\left.\begin{matrix} f(x,y)=0 \\ g(x,y)=0 \end{matrix}\right\} F(x,y)=0$

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} f(x_0,y_0) \\ g(x_0,y_0) \end{pmatrix} + \begin{pmatrix} \partial f/\partial x(x_0,y_0) & \partial f/\partial y(x_0,y_0) \\ \partial g/\partial x(x_0,y_0) & \partial g/\partial y(x_0,y_0) \end{pmatrix} \begin{pmatrix} x_1 - x_0 \\ y_1 - y_0 \end{pmatrix}$$

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} h_x \\ h_y \end{pmatrix}$$

en general, con $F: \mathbb{R}^n \to \mathbb{R}^n$

$$0 = F(x) \simeq F(x_0) + J_F(x_0)(x_1 - x_0)$$

jacobiano · H

para la siguiente iteración queremos H.

$$x_{k+1} = x_k + H$$
$$H = J_F(x_k) \backslash -F(x_k)$$

si Newton converge, lo hace cuadráticamente, más rapido que el punto fijo.

Algoritmo de Newton en Matlab

```
Function [x, iter, incr, tcc] = newton(F, x0, tol, maxiter)

    % Inicialización de variables
        iter = 0; incr = tol + 1; x = [x0];

    % Iteraciones
        while iter<maxiter & incr > tol

            [Y, JFY] = feval(F, x0);
            H = JFY \ (-Y);
            xn = x0 + H;
            incr = norm(H);
            iter = iter + 1;
            x0 = xn;
        end while

    % otras salidas
        if incr > tol
            disp('El metodo no converge')

    % error y tasa de convergencia cuadrática
      e = diff(x)
      e(1:end) = norm(e(1:end));
      tcc = e(2:end) ./ e(1:end-1)^2;
```

$tcc = \frac{e_2}{e_1^2}, \frac{e_3}{e_2^2}, \frac{e_4}{e_3^2}, \ldots$

la función F.m devuelve $F(x)$ y el jacobiano en x.

```
function [Y, JFY] = F(x)
    Y(1)
    Y(2)
    JFY(1,1)
    JFY(1,2)
    JFY(2,1)
    JFY(2,2)
```

$F = \begin{pmatrix} g_1(x) \\ g_2(x) \end{pmatrix}$

$\begin{pmatrix} \partial g_1/\partial x & \partial g_1/\partial y \\ \partial g_2/\partial x & \partial g_2/\partial y \end{pmatrix}$

Newton modificado:

- Para no tener que calcular el Jacobiano cada iteración, se puede calcular sólo para x_0 y usarlo para todo x_k

$$H = (J_F(x_0) \backslash -F(x_k))$$

- O bien calcularlo cada p iteraciones (modificación en el programa)

$$H = (J_F(x_0) \backslash -F(x_k)) \quad k = 0, 1, \ldots, p-1$$
$$= (J_F(x_p) \backslash -F(x_k)) \quad k = p, p+1, \ldots, 2p-1,$$
$$\ldots$$

esto se puede hacer: añadiendo parámetro p dentro del while poner

```
if (iter/p) = ceil(iter/p)    % es un multiplo de p
    [Y, JFY] = jeval(F, x0);
else
    [Y] = jeval(F, x0);
```

y que la función F.m calcule el Jacobiano sólo si ...

if nargout = 2 (nº de argumentos de salida)

TEMA 2. INTEGRACION NUMERICA

- no conocemos primitiva de $f(x)$
- tenemos valores tabulados de $f(x)$

$$I(f) = \int_a^b f(x)\,dx$$

La idea consiste en aproximar mediante una suma finita:

$$I_n(f) = \sum_{j=0}^{n} \beta_j f(x_j)$$

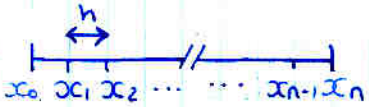
$x_0, x_1, \dots, x_n$ nodos
$\beta_0, \beta_1, \dots, \beta_n$ pesos

El error $E_n(f) = I(f) - I_n(f)$ debe ser lo mas pequeño posible

Orden del error: n $\quad E_n(f) = O(h^n)$ con $h = \frac{b-a}{n}$

Grado de precisión m Tal que $E_n(f) = 0$ si integramos un polinomio de grado $\leq m$

Fórmulas de cuadratura mediante interpolación



n subintervalos
$n+1$ nodos

Si Interpolamos con polinomios de Lagrange ⇒ Fórmulas de Newton-Cotes

$$f(x) \simeq p_n(x) = \sum_{j=0}^{n} f(x_j) L_j(x)$$

L_i cumple:
$L_i(x_i) = 1$
$L_i(x_j) = 0 \quad i \neq j$

$$L_i(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})\cancel{(x-x_{i+1})}\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}$$

nos saltamos $(x-x_i)$

$$f(x) = p_n(x) + \underbrace{\frac{f^{(n+1)}(\xi)}{n+1!}(x-x_0)(x-x_1)\dots(x-x_n)}_{\text{error de interpolación}} \qquad \xi \in [a,b]$$

integrando a ambos lados y trabajando se llega a:

$$I(f) = I_n(f) + E_n(f)$$

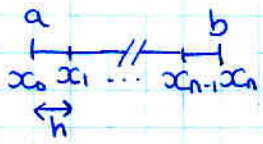
$$I(f) = \int_a^b f(x)\,dx \qquad I_n(f) = \sum_{j=0}^{n} \beta_j f(x_j)$$

siendo $\beta_j = \int_a^b L_j(x)\,dx$

$$E_n(f) = \frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi(x)) \prod_{j=0}^{n} (x-x_j)\,dx$$

- Fórmulas cerradas de Newton-Cotes

tomando $x_0 = a$, $x_n = b$



$x_j = x_0 + jh$

$h = \frac{b-a}{n}$

Teorema:

$$\int_a^b f(x)\,dx = \sum_{j=0}^{n} \beta_j f(x_j) + E_n(f) \qquad \beta_j = \int_a^b L_j(x)\,dx$$

- si n es par
$$E_n(f) = \frac{h^{n+3}}{(n+2)!} f^{(n+2)}(\eta) \int_0^n s^2(s-1)\dots(s-n)\,ds$$
- si n es impar
$$E_n(f) = \frac{h^{n+2}}{(n+1)!} f^{(n+1)}(\eta) \int_0^n s(s-1)\dots(s-n)\,ds$$

Observa: si n es par → grado de precisión $n+1$ ($E_n = 0$ para pol grado $\leq n+1$)
si n es impar → grado de precisión n ($E_n = 0$ para pol grado $\leq n$)

⇒ para aumentar el grado de precisión hay que aumentar n en 2.

de estas fórmulas podemos obtener los métodos vistos el año pasado.

$$\int_a^b f(x)\,dx = \underbrace{I_n(f)}_{\text{aproximación}} + \underbrace{E_n(f)}_{\text{error}}$$

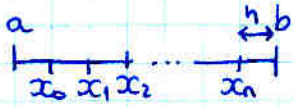
$n=1$ → trapecios
$$\int_a^b f(x)\,dx = \frac{h}{2}(f(x_0) + f(x_1)) - \frac{h^3}{12} f''(c)$$

$n=2$ → simpson
$$\int_a^b f(x)\,dx = \frac{h}{3}(f(x_0) + 4f(x_1) + f(x_2)) - \frac{h^5}{90} f^{(IV)}(c)$$

$n=3$ → simpson 3/8
$$\int_a^b f(x)\,dx = \frac{3h}{8}(f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)) - \frac{3h^5}{80} f^{(IV)}(c)$$

para más casos ver libro pag 107

- Formulas abiertas de Newton Cotes



$x_0 = a + h$, $x_n = b - h$

$x_j = a + (j+1)h$ $\quad h = \frac{b-a}{n+2}$ (los dos subintervalos de los extremos)

- El Teorema es el mismo salvo los límites de la integral en s del error $\int_{-1}^{n+1}$
- para valores de n se obtienen también métodos vistos el año pasado. ej: punto medio (ver pag 108)

- Formulas de cuadratura compuestas

Dividimos el intervalo $[a,b]$ en varios subintervalos y se aplica a cada intervalo individualmente alguno de los métodos anteriores (para un método de grado n, agrupamos los subintervalos de n en n ya que el método necesitará n subintervalos)

Trapecios para N subintervalos
(se obtiene un trapecio para cada subintervalo)

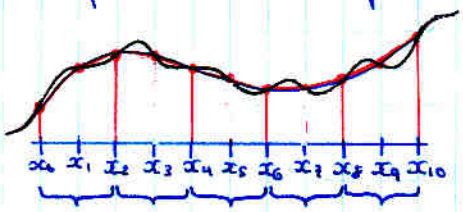
$$\int_a^b f(x)\,dx = \sum_{k=0}^{N-1} \underbrace{\int_{x_k}^{x_{k+1}} f(x)\,dx}_{\text{se aplica trapecios a esta integral.}}$$

(la integral no será más que la suma de cada trapecio)

Desarrollando la expresión sale:

$$\int_a^b f(x)\,dx = \underbrace{\frac{h}{2}\left[f(x_0)+2\sum_{k=1}^{N-1} f(x_k)+f(x_N)\right]}_{I_T(f)\text{ aproximación fórmula de cuadratura}} \underbrace{-\frac{h^2}{12}(b-a)f''(c)}_{E_T(f)\text{ error cometido}} \quad c\in[a,b]$$

Simpson para N subintervalos
(aplicamos simpson a cada pareja de subintervalos)



se necesita n par. $\quad m = 2n$

$$h = \frac{b-a}{2m}$$

$$x_k = a + kh$$

se obtiene (simplemente sumando simpon varias veces)

$$\int_a^b f(x)\,dx = \underbrace{\frac{h}{3}\left[f(x_0)+2\sum_{k=1}^{m-1} f(x_{2k})+4\sum_{k=1}^{m} f(x_{2k-1})+f(x_{2m})\right]}_{I_S(f)} \underbrace{-\frac{h^4}{180}(b-a)f^{(IV)}(c)}_{E_S(f)}$$

- Integración de Romberg (aplicar extrapolación de Richardson a lo anterior)

Sabiendo que el error en simpson es: $E_S = -\frac{h^4}{180}(b-a)f^{IV}(\xi) = k\cdot h^4$

si evaluamos simpson con h: $I \cong I_S[h] + k\cdot h^4$
y con $\frac{h}{2}$ $\quad I \cong I_S[\frac{h}{2}] + k\cdot \frac{h^4}{16}$

} podemos estimar k (sistema de ecuaciones) y por tanto el error, refinando por tanto la aproximación

Esto se puede repetir varias veces con las sucesivas aproximaciones.
se obtiene la tabla de Romberg

$$R_{kj} = \frac{4^{j-1}R_{kj-1} - R_{k-1j-1}}{4^{j-1}-1}$$

R_{11}
$R_{12} \to R_{22}$
$R_{13} \to R_{23} \to R_{33}$
R_{14} R_{24} R_{34} R_{44} ⋱
⋮ ⋮ ⋮ ⋮

las columnas y las diagonales de la tabla de Romberg convergen a $I(f)$

(algoritmo pagina 113)

Cuadratura de Gauss

Tenemos

$$I(f) = \int_a^b f(x)dx \simeq c_1 f(x_1) + c_2 f(x_2) + \dots + c_n f(x_n)$$

$$E(f) = \frac{1}{n!} \int_a^b f^{(n)}(c) \prod_{i=1}^{n} (x - x_i)dx$$

En general, con n nodos, el error es cero para polinomios de grado n. (grado de precisión $m = n$)
¿Podemos conseguir un grado de precisión $m = n + r$ con r máximo?
eligiendo los nodos $x_1, x_2, \dots$ adecuadamente (sólo es posible si conocemos la expresión de $f(x)$) y los coeficientes $c_1, c_2, \dots$ se puede conseguir que el error sea cero para polinomios de grado hasta $2n-1$

[$2n$ parámetros $x_1, x_2, \dots, x_n$ / $c_1, c_2, \dots, c_n$ pueden ajustar $2n$ incógnitas ≡ los coeficientes de un polinomio de grado $2n-1$
se puede hacer un sistema de ecuaciones para obtener x_i, c_i

El caso óptimo:
grado de precisión $m = n + r$ maximizando r
$m = 2n - 1$

Teorema
se obtiene si y sólo si se verifica:
(a) la fórmula es de tipo interpolatorio
(b) Los nodos de la fórmula son las raices del n-ésimo polinomio ortogonal respecto del producto escalar $\langle f, g \rangle = \int_a^b f(x)\, g(x)\, w(x)\, dx$ (función peso)

se obtienen unos polinomios u otros según $w(x)$

ejemplo: $w(x) = 1$ y tomamos $n = 2$
antes de hacer uso del teorema, resolvamos el sistema de ecuaciones de x_i y c_i que hace que la integración sea exacta para polinomios de grado $2n-1$, y veremos que coinciden con las raices de los polinomios ortogonales para $w(x) = 1$ (Legendre)

$$n=2,\ [a,b]=[-1,1] \Rightarrow \int_{-1}^{1} f(x)\,dx = c_1 f(x_1) + c_2 f(x_2)$$

la integral debe ser exacta para pol. de grado 0, 1, 2 y 3 ($\leq 2n-1$)

$$\begin{aligned} \int_{-1}^{1} dx &= 2 \\ \int_{-1}^{1} x\,dx &= 0 \\ \int_{-1}^{1} x^2 dx &= \tfrac{2}{3} \\ \int_{-1}^{1} x^3 dx &= 0 \end{aligned} \Rightarrow \begin{cases} c_1 \cdot 1 + c_2 \cdot 1 = 2 \\ c_1 x_1 + c_2 x_2 = 0 \\ c_1 x_1^2 + c_2 x_2^2 = \tfrac{2}{3} \\ c_1 x_1^3 + c_2 x_3^3 = 0 \end{cases}$$

se obtiene
$c_1 = c_2 = 1$
$x_1 = -\sqrt{1/3} = -\frac{\sqrt{3}}{3}$
$x_2 = \sqrt{1/3} = \frac{\sqrt{3}}{3}$

x_1 y x_2 son precisamente las raices de los polinomios de Legendre (familia ortogonal en $[-1,1]$ con $w(x) = 1$) lo cual verifica el teorema

Según sea $w(x)$ y $[a,b]$ obtenemos:

→ **Gauss-Legendre** ↳ $w(x)=1$, ↳ $[a,b]=[-1,1]$

polinomios de Legendre:

$p_0(x)=1$

$p_1(x)=x$

$p_{n+1}(x)=\frac{1}{n+1}\left[(2n+1)x\,p_n(x)-n\,p_{n-1}(x)\right]$

$p_n(x)$ tiene n raíces:

$x_k=\left[1-\frac{1}{8n^2}+\frac{1}{8n^3}\right]\cos\frac{4k-1}{4n+2}\pi+O(n^{-4})$

$k=1,2,3,\dots,n$

además, si se calculasen los coeficientes c_i en el caso general (sistema de ecuaciones como el ej. anterior pero n cualquiera) se tiene

$$c_i=\frac{2}{(1-x_i^2)(p_n'(x_i))^2}\qquad i=1,2,\dots,n$$

en MATLAB: Legendre(n)

① Programa que calcule polinomios de Legendre (recuerda: $p(x)\cdot x$ es añadir un cero al final del vector que le representa)

② Calcular raíces de los polinomios roots(p)

③ Calcular c_i (para derivar hacer polyder(p))

Cambio de intervalo: $[a,b]\to[-1,1]$

$$x=mt+n \qquad m=\frac{b-a}{2},\quad n=\frac{a+b}{2}$$

→ **Gauss-Chebyshev**

↳ $w(x)=\frac{1}{\sqrt{1-x^2}}$

↳ $[a,b]=[-1,1]$

Se aplica para integrandos de la forma $\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}dx\simeq\sum_1^n c_i f(x_i)$

Polinomios:

$T_0(x)=1$

$T_1(x)=x$

$T_n(x)=2x\,T_{n-1}(x)-T_{n-2}(x)$

raíces de $T_n(x)$:

$x_k=\cos\frac{(2k-1)\pi}{2n}$

$k=1,2,\dots,n$

pesos $c_i=\frac{\pi}{n}$

→ **Gauss-Laguerre**

↳ $w(x)=e^{-x}$

↳ $[a,b]=[0,+\infty[$

$$\int_0^{+\infty}e^{-x}f(x)\,dx\simeq\sum_1^n c_i f(x_i)$$

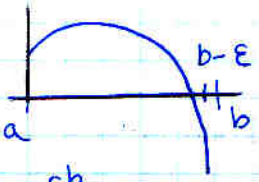
ver expresiones de c_i y x_i en transparencias

Integración Impropia

Def: integral impropia con asíntota en a o b

$\lim_{x\to a^+} |f(x)| = \infty$ $\lim_{x\to b^-} |f(x)| = \infty$

Si existen los límites, la integral es convergente. Si no, es divergente

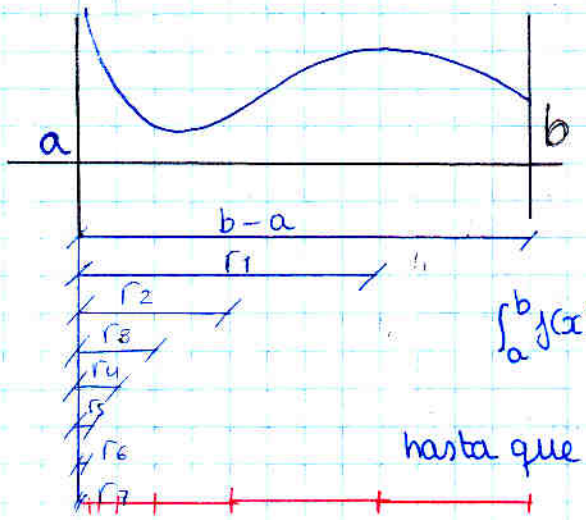


$$\int_a^b f(x)dx = \lim_{\varepsilon\to 0}\int_{a+\varepsilon}^b f(x)dx \qquad \int_a^b f(x)\,dx = \lim_{\varepsilon\to 0}\int_a^{b-\varepsilon} f(x)\,dx$$

Técnicas numéricas

(si la singularidad no está en un extremo, dividimos en subintervalos tal que lo esté)

(a) tomamos sucesión decreciente convergente a cero $b-a > r_1 > r_2 > r_3 > \dots$

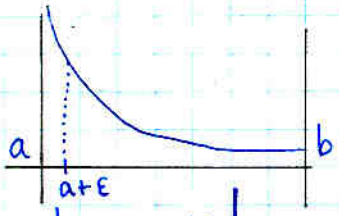


podemos descomponer

$$\int_a^b f(x)\,dx = \int_{a+r_1}^{b} \dots + \int_{a+r_2}^{a+r_1} + \int_{a+r_3}^{a+r_2} + \int_{a+r_4}^{a+r_3} + \dots$$

hasta que $\left|\int_{a+r_{k+1}}^{a+r_k} f(x)dx\right| < tol$

(b) aislar los puntos de discontinuidad



$$\int_a^b f(x)\,dx = \int_a^{a+\varepsilon} f(x)dx + \int_{a+\varepsilon}^b f(x)dx$$

↑ cuadratura de Gauss | debe aprox. a cero conforme $\varepsilon\to 0^+$

↑ cualquier método para integral propia

ejemplo: $\int_0^{\frac{\pi}{2}} \tan(x)\,dx$

para $\varepsilon = 0'01$

$$I = \int_0^{\frac{\pi}{2}-0'01} \tan(x)\,dx + \int_{\frac{\pi}{2}-0'01}^{\pi/2} \tan(x)\,dx$$

$\int_0^{\frac{\pi}{2}-0'01} \tan(x)\,dx \to 4'605187$

$\int_{\frac{\pi}{2}-0'01}^{\pi/2} \tan(x)\,dx$ por cuadratura gauss

$$= \frac{0'01}{2}\int_{-1}^{1} \tan\left[\frac{\pi}{2}-0'01+\frac{0'01}{2}(t+1)\right]dt$$

$= 4'566650$

para $\varepsilon = 0'0001$

$$I = \int_0^{\frac{\pi}{2}-\varepsilon} \tan(x)\,dx + \int_{\frac{\pi}{2}-\varepsilon}^{\pi/2} \tan(x)\,dx$$

$\int_0^{\frac{\pi}{2}-\varepsilon} \tan(x)\,dx \to 4'605153$

$\int_{\frac{\pi}{2}-\varepsilon}^{\pi/2} \tan(x)\,dx$ ↓ cuadratura gauss $\to 4'566667$

$\not\to 0 \Rightarrow$ no conv.

I. impropias $\xrightarrow{\text{conversión}}$ I propias

- Cambio de variable

ej: $\int_0^1 x^{-\frac{1}{n}} f(x)\,dx \xrightarrow[t^n = x]{} n\int_0^1 t^{n-2} f(t^n)\,dt$

- Desarrollo por series

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \dots$$

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

ej: $\int_{0'0001}^{1} \frac{e^x}{x^3}\,dx$ toma valores muy grandes cerca de 0'0001

conviene: $\int_{0'0001}^{1} \frac{e^x}{x^3}\,dx = \int_{0'0001}^{1} \frac{1+x+\frac{x^2}{2}}{x^3}\,dx + \int_{0'0001}^{1} \frac{e^x - 1 - x - \frac{x^2}{2}}{x^3}\,dx$

se resuelve analíticamente

ya no toma valores tan grandes

podemos llegar incluso a la:

- eliminación de la singularidad

$$\int_0^1 \frac{\cos x}{\sqrt{x}}\,dx = \underbrace{\int_0^1 \frac{1 - x^2/2}{\sqrt{x}}\,dx}_{\frac{1}{\sqrt{x}} - \frac{1}{2}\frac{x^2}{\sqrt{x}}} + \underbrace{\int_0^1 \frac{\cos x - 1 + \frac{1}{2}x^2}{\sqrt{x}}\,dx}_{\lim_{x\to 0} = 0 \text{ (l'hopital)}}$$

$\frac{1}{\sqrt{x}}$ → conv; $\frac{x^2}{\sqrt{x}}$ → conv; es continua

$$\int_0^x \frac{e^{-t}}{1-t}\,dt = \int_0^x \frac{e^{-1} + e^{-t} - e^{-1}}{1-t}\,dt$$

$$= \int_0^x \frac{e^{-1}}{1-t}\,dt + \int_0^x \frac{e^{-t} - e^{-1}}{1-t}\,dt$$

no se puede hacer

$\frac{e^{-1}}{1-t}$ → ~~$e^{-1}\ln(1-x)$~~ diverge

$\frac{e^{-t} - e^{-1}}{1-t}$ → continua

Integrales infinitas

Intervalo infinito.

ej $\lim_{b\to\infty} \int_a^b f(x)\,dx$ convergencia $\Leftrightarrow$ existe el límite y es un número real

ej $\int_1^\infty \frac{1}{x}\,dx = \lim_{b\to\infty} \int_1^b \frac{1}{x}\,dx = \lim_{b\to\infty} (\ln|x|)_1^b = \lim_{b\to\infty} (\ln b - \ln 1) = +\infty$

ej. $\int_{-\infty}^0 e^x\,dx = \lim_{a\to-\infty} \int_a^0 e^x\,dx = \lim_{a\to-\infty} [e^x]_a^0 = 1$

- Descomposición en suma de integrales

$\int_1^\infty = \int_1^2 + \int_2^4 + \int_4^8 + \int_8^{16} + \dots$ hasta que $\left|\int_{2^k}^{2^{k+1}}\right| < tol$

los intervalos pueden ser cualesquiera, pero conviene que se vayan doblando.

- Cambio de variable:

ej. $\int_0^{+\infty} \sqrt{x}\, e^{-x}\,dx = \left\{\begin{matrix} t = e^{-x} \\ dt = e^{-x}(-dx)\end{matrix}\right\} = \int_1^0 \sqrt{-\log t}\cdot(-dt) = \int_0^1 \sqrt{\log\tfrac{1}{t}}\,dt$

$= \underbrace{\int_0^{0'0001} \sqrt{\log\tfrac{1}{t}}\,dt}_{\text{tiene problema en 0}} + \underbrace{\int_{0'0001}^1 \sqrt{\log\tfrac{1}{t}}\,dt}_{\text{converge}}$

↳ Gauss-Legendre → Romberg

ej $\int_1^{+\infty} x^2 e^{-x^2}\,dx = \left\{\begin{matrix} x = 1/\sqrt{t} \\ dx = -\tfrac{1}{2}t^{-3/2}dt\end{matrix}\right\} = \frac{1}{2}\int_0^1 \frac{e^{-1/t}}{t^{5/2}}\,dt$

$\lim_{t\to0} \frac{e^{-1/t}}{t^{5/2}} = K$

$\lim_{t\to0} \ln\frac{e^{-1/t}}{t^{5/2}} = \ln K = \lim_{t\to0}\left\{-\frac{1}{t} - \frac{5}{2}\ln t\right\} = -\infty \to K = 0$

por tanto, se puede introducir $\frac{1}{2}\int_0^1 \frac{e^{-1/t}}{t^{5/2}}\,dt$ en Romberg, pero en la función poner que devuelva 0 en 0, es decir que busque ceros en el vector de entrada (for) y si algún elemento es cero que devuelva cero (if)

- Convertir en una ecuación diferencial

$F(x) = \int_0^x f(t)\,dt \Leftrightarrow \begin{cases} \frac{dF(x)}{dx} = f(x) \\ F(0) = 0 \end{cases}$

se resuelve por Runge-Kutta

• Integración multiple

$$\iint_R f(x,y)\,dA = \int_a^b \left[\int_c^d f(x,y)\,dy\right] dx$$

Tan simple como aplicar las fórmulas de cuadratura repetidas veces p ej. simpson

Recordemos por ej Simpson

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \ldots + 4f(x_{2n-1}) + f(x_{2n})\right]$$

siendo

$$h = \frac{b-a}{2n}$$

recuerda: se usa 2n porque debe ser un nº impar de nodos (contando el cero)

es decir $I = \frac{h}{3}\sum_{i=1}^{2n} C_i\, f(x_i)$

$C_i = [1\ 4\ 2\ 4\ 2\ 4 \ldots 2\ 4\ 1]$

En el caso de la integral doble de arriba tomariamos:

$x = a:h:b$ $\quad x_i = a + ih$ $\quad h = \frac{b-a}{2n}$ y $y = c:k:d$ $\quad y_j = c + jk$ $\quad k = \frac{b-a}{2m}$

y seria

$$I = \int_a^b \left[\int_c^d f(x,y)\,dy\right] dx$$

aplicando simpson a la integral de dentro

$$= \int_a^b \frac{k}{3}\left[\underbrace{f(x,y_0)}_{(1)} + 4\sum_{j=1}^{m} \underbrace{f(x,y_{2j-1})}_{(2)} + 2\sum_{j=1}^{m-1} \underbrace{f(x,y_{2j})}_{(3)} + \underbrace{f(x,y_{2m})}_{(4)}\right] dx$$

ahora aplicando de nuevo el método al integrando tan grande quedaria (separando en 4 integrales y aplicandolo a cada una)

$$= \frac{h}{3}\frac{k}{3}\Bigg[\left(f(x_0,y_0) + 4\sum_{i=1}^{n} f(x_{2i-1},y_0) + 2\sum_{i=1}^{n-1} f(x_{2i},y_0) + f(x_{2n},y_0)\right) \quad (1)$$

$$+ 4\sum_{j=1}^{m}\left(f(x_0,y_{2j-1}) + 4\sum_{i=1}^{n} f(x_{2i-1},y_{2j-1}) + 2\sum_{i=1}^{n-1} f(x_{2i},y_{2j-1}) + f(x_{2n},y_{2j-1})\right) \quad (2)$$

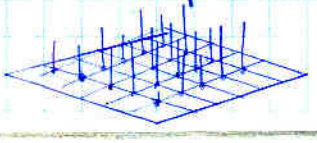
$$+ 2\sum_{j=1}^{m-1}\left(f(x_0,y_{2j}) + 4\sum_{i=1}^{n} f(x_{2i-1},y_{2j}) + 2\sum_{i=1}^{n-1} f(x_{2i},y_{2j}) + f(x_{2n},y_{2j})\right) \quad (3)$$

$$+ \left[f(x_0,y_{2m}) + 4\sum_{i=1}^{n} f(x_{2i-1},y_{2m}) + 2\sum_{i=1}^{n-1} f(x_{2i},y_{2m}) + f(x_{2n},y_{2m})\right] \quad (4)$$

que se puede escribir como:

$$I = \frac{hk}{9}\sum_{i=1}^{2m+1}\sum_{j=1}^{2n+1} p_i\, q_j\, f(x_i,y_j)$$

es decir, $(2m+1)\cdot(2n+1)$ nodos en el plano $[a,b]\times[c,d]$ cada uno con superso



en MATLAB

① Aplicar directamente la fórmula anterior con for anidados

② Aprovechar el tratamiento matricial de Matlab

ej:

y_3	1	4	2	4	1
y_2	4	16	8	16	4
y_1	1	4	2	4	1
q''					
$p=$	1	4	2	4	1
	x_0	x_1	x_2	x_3	x_4

$$w = q' \cdot p = \begin{pmatrix} 1 & 4 & 2 & 4 & 1 \\ 4 & 16 & 8 & 16 & 4 \\ 1 & 4 & 2 & 4 & 1 \end{pmatrix}$$

$$M = \begin{pmatrix} f(x_0,y_0) & f(x_1,y_0) & f_{20} & f_{30} & f_{40} \\ f(x_0,y_1) & f_{11} & f_{21} & f_{31} & f_{41} \\ f_{02} & f_{12} & f_{22} & f_{32} & f_{42} \\ f_{03} & f_{13} & f_{23} & f_{33} & f_{43} \end{pmatrix}$$

```
x = [x0 x1 x2 ...]
y = [y0 y1 y2 ]
[X,Y] = meshgrid(x,y)
M = feval(f,X,Y)
```

```
I = sum(sum( w .* M))
```

la suma de todos los elementos de la multiplic. elemento a elemento de los pesos por la función evaluada en los puntos

Lo mismo para integrales tripes $\int_a^b \int_c^d \int_e^f f(x,y,z)\,dx\,dy\,dz$

$$I = \frac{h\,k\,w}{27} \sum_{i=1}^{2n+1} \sum_{j=1}^{2m+1} \sum_{k=1}^{2r+1} p_i\, q_j\, l_k\, f(x_i, y_j, z_k)$$

$x_i = a : h : b$ — $2n+1$ elem

$y_i = c : k : d$ — $2m+1$ elem

$z_i = e : w : f$ — $2r+1$ elem

Deberíamos saber hacerlo aplicando otros métodos además de simpson

ej: $\int_a^b \int_c^d f(x,y)\,dy\,dx$ por cuadratura Gauss-Legendre

$$\int_a^b \int_c^d f(x,y)\,dy\,dx \xrightarrow[\substack{y = \frac{d-c}{2}t + \frac{d+c}{2} \\ dy = \frac{d-c}{2}dt}]{\text{cambiar a } [-1,1]} \frac{d-c}{2} \int_a^b \int_{-1}^{1} \frac{d-c}{2} f\left(x, \frac{d-c}{2}t + \frac{d+c}{2}\right) dt$$

obtener t_i y C_i

```
[nodos, pesos] = legendre(n)
```

$$\xrightarrow{} \frac{d-c}{2} \int_a^b \sum_{i=1}^{n} C_i\, f\left(x, \frac{d-t}{2}t_i + \frac{d+c}{2}\right) dx$$

y ahora volver a aplicar gauss-legendre para la integral de fuera.

$$\xrightarrow[x = \frac{b-a}{2}u + \frac{b+a}{2}, \; dx = \frac{b-a}{2}du]{\text{cambiar a } [-1,1]} \frac{d-c}{2}\frac{b-a}{2} \int_a^b \sum_{i=1}^{n} C_i\, f\left(\frac{b-a}{2}u + \frac{b+a}{2}, \frac{d-t}{2}t_i + \frac{d+c}{2}\right) dx$$

$$= \frac{d-c}{2}\frac{b-a}{2} \sum_{i=1}^{n} C_i \int_a^b f\left(\frac{b-a}{2}u + \frac{b+a}{2}, \frac{d-t}{2}t_i + \frac{d+c}{2}\right) dx$$

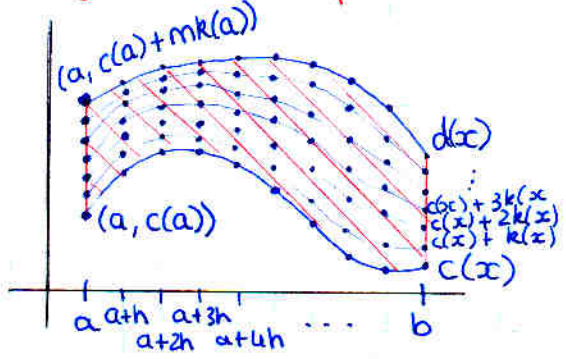
obtener u_j y d_j

```
[nodos, pesos] = legendre(m)
```

$$\xrightarrow{} \frac{d-c}{2}\frac{b-a}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} C_i\, d_j\, f\left[\frac{b-a}{2}u_j + \frac{b+a}{2}, \frac{d-t}{2}t_i + \frac{d+c}{2}\right]$$

de nuevo es la multiplicación matricial de los vectores de pesos multiplicado elemento a elemento por la evaluación de la función en los nodos

• Integración múltiple sobre recintos no rectangulares



$$h = \frac{b-a}{n}$$

$$k(x) = \frac{d(x)-c(x)}{m}$$

los nodos son

$$x_{ij} = (a+ih,\ c(x)+jk(x)) \qquad (x_i, y_j) \qquad \begin{matrix} i = 0 \dots n \\ j = 0 \dots m \end{matrix}$$

a estas integrales se les puede aplicar cualquiera de los métodos ya vistos.

ej: método de Simpson

(en MATLAB no se puede empezar con x_0)

$$\begin{cases} h = \dfrac{b-a}{2n} & x_i = a+(i-1)h \quad i = 1,2,3,\dots,2n+1 \\ k(x) = \dfrac{d(x)-c(x)}{2m} & y_j = c(x_i)+(j-1)k(x_i) \quad j = 1,2,3,\dots,2m+1 \end{cases}$$

pesos: $p_i = [1\ 4\ 2\ 4\ 2 \dots 4\ 1]_{2n+1}$

$q_j = [1\ 4\ 2\ 4\ 2 \dots 4\ 1]_{2m+1}$

$$\int_a^b \int_{c(x)}^{d(x)} f(x,y)\,dy\,dx = \frac{h}{q} \sum_{i=1}^{2n+1} \sum_{j=1}^{2m+1} p_i\, q_j\, k(x_i)\, f(x_i, y_j)$$

$\frac{h}{3} \cdot \frac{k(x_i)}{3}$

function I = simpsonvar(f, a, b, c, d, n, m)

ej: método de Gauss-Legendre hay que convertir el intervalo en $[-1,1]\times[-1,1]$

$$\int_a^b \int_{c(x)}^{d(x)} f(x,y)\,dy\,dx \qquad \begin{cases} y = \dfrac{d(x)-c(x)}{2}\,t + \dfrac{d(x)+c(x)}{2} \\ dy = \dfrac{d(x)-c(x)}{2} \\ x = \dfrac{b-a}{2}\,z + \dfrac{b+a}{2} \\ dx = \dfrac{b-a}{2} \end{cases}$$

$$= \int_{-1}^{1} \int_{-1}^{1} \frac{d(x)-c(x)}{2} \cdot \frac{b-a}{2} \cdot f\left(\frac{b-a}{2}z + \frac{b+a}{2},\ \frac{d(x)-c(x)}{2}t + \frac{d(x)+c(x)}{2}\right) dt\,dz$$

($d(x)$ y $c(x)$ evaluados en $x(z)$)

$[c, T] = \text{legendre}(n) \rightarrow \begin{matrix} c_1, c_2, c_3, \dots c_n \\ t_1, t_2, t_3 \dots t_n \end{matrix} \quad i = 1,2,\dots,n$

$[P, Z] = \text{legendre}(m) \rightarrow \begin{matrix} p_1, p_2, p_3, \dots p_m \\ z_1, z_2, z_3, \dots z_m \end{matrix} \quad j = 1,2,\dots,m$

$\frac{b-a}{2}z_j + \frac{b+a}{2} = x_j$

$$= \sum_{j=1}^{n} \sum_{j=1}^{m} \frac{d\left(\frac{b-a}{2}z_j+\frac{b+a}{2}\right) - c\left(\frac{b-a}{2}z_j+\frac{b+a}{2}\right)}{2} \cdot \frac{b-a}{2} \cdot p_j \cdot c_i \cdot f\left(\frac{b-a}{2}z_j+\frac{b+a}{2},\ \frac{d(x_j)-c(x_j)}{2}t + \frac{d(x_j)+c(x_j)}{2}\right)$$

• Método de Monte Carlo

media de $f(x)$ en $[a,b]$: por definición $M = \dfrac{\int_a^b f(x)\,dx}{b-a}$

tomando muestras x_i $\quad M \simeq \dfrac{1}{n}\sum_{i=1}^{n} f(x_i)$

despejando

$$\boxed{I = \int_a^b f(x)\,dx \simeq \frac{b-a}{n}\sum_{i=1}^{n} f(x_i)}$$

si los puntos x_i son aleatorios el método se llama Monte Carlo

En el caso de una integral múltiple

$$\int_a^b\int_c^d f(x,y)\,dy\,dx \simeq \frac{(b-a)(c-d)}{n}\sum_{i=1}^{n} f(x_i,y_i)$$

(área del recinto)

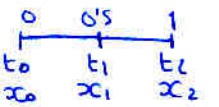
(x_i, y_i) son ptos del plano aleatorios (no hace falta doble sumatorio)

ej: $\int_0^3\int_5^7 e^{x+3y}\,dy\,dx = \frac{6}{100}\sum_{i=1}^{100}\exp(x_i+3y_i)$ siendo $(x_i,y_i) = (\text{rand}(0,3), \text{rand}(5,7))$

Problema

$$u(x) = \frac{2-x^2}{3} + \int_0^1 (x^2+t)\,u(t)\,dt \qquad x \in [0,1]$$

Lo resolvemos por simpson y discretizando así:



$$u(x) = \frac{2-x^2}{3} + \frac{h}{3}\left[(x^2+t_0)u(t_0) + 4(x^2+t_1)u(t_1) + (x^2+t_2)u(t_2)\right]$$

siendo:

$$\left.\begin{array}{ll} x=t_0 & u_1 = \frac{2-t_0^3}{3} + \frac{h}{3}\left[(t_0^2+t_0)u_1 + 4(t_0^2+t_1)u_2 + (t_0^2+t_2)u_3\right] \\ x=t_1 & u_2 = \frac{2-t_1^3}{3} + \frac{h}{3}\left[(t_1^2+t_0)u_1 + 4(t_1^2+t_1)u_2 + (t_1^2+t_2)u_3\right] \\ x=t_2 & u_3 = \frac{2-t_2^3}{3} + \frac{h}{3}\left[(t_2^2+t_0)u_1 + 4(t_2^2+t_1)u_2 + (t_2^2+t_2)u_3\right] \end{array}\right\}$$

obtenemos u_1, u_2, u_3 resolviendo el sistema

$$\begin{pmatrix} \frac{h}{3}(t_0^2+t_0)-1 & \frac{4h}{3}(t_0^2+t_1) & \frac{h}{3}(t_0^2+t_2) \\ \frac{h}{3}(t_1^2+t_0) & \frac{4h}{3}(t_1^2+t_1)-1 & \frac{h}{3}(t_1^2+t_2) \\ \frac{h}{3}(t_2^2+t_0) & \frac{4h}{3}(t_2^2+t_1) & \frac{h}{3}(t_2^2+t_2)-1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} -\frac{2-t_0^3}{3} \\ -\frac{2-t_1^3}{3} \\ -\frac{2-t_2^3}{3} \end{pmatrix}$$

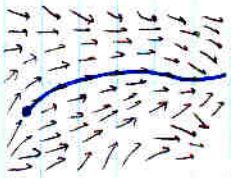
en matlab:

$$A(:,i) = \frac{h}{3}T.^{\wedge}2 + T(i)$$

TEMA 3 - PROBLEMAS DE VALOR DE FRONTERA

Recordemos primero: PVI : problemas de valor inicial

Nos dan una ec. diferencial y el valor de las incognitas y sus derivadas en el inicio.



definiendo el punto inicial y su derivada (dirección inicial) se obtiene una única curva

Los PVI, bajo hipótesis adecuadas, tienen solución única

se resuelven mediante runge kutta

ej: PVI (orden 3)
$$\begin{cases} y''' = 2y'' - 3y'x\cos x + \operatorname{sen} x \\ y(0) = 1 \\ y'(0) = 0 \\ y''(0) = 2 \end{cases}$$

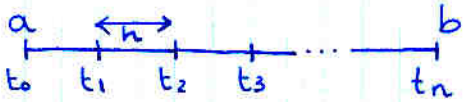
↓ lo transformamos en un sistema

c.v. $y_1 = y$, $y_2 = y'$, $y_3 = y''$

sistema equiv. PVI de orden 1
$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ y_3' = 2y_3 - 3y_2 x\cos x + \operatorname{sen} x \\ \text{c.I.} \quad y_1(0) = 1 \\ \qquad\quad y_2(0) = 0 \\ \qquad\quad y_3(0) = 2 \end{cases}$$

matricialmente: $y' = f(t, y)$

Runge-Kutta:



$$\int_{t_0}^{t_1} y'(t)\,dt = \int_{t_0}^{t_1} f(t,y)\,dt$$

$$y(t_1) - y(t_0) = \int_{t_0}^{t_1} f(t,y)\,dt$$

$$y(t_1) = y(t_0) + \underbrace{\int_{t_0}^{t_1} f(t,y)\,dt}$$

aproximar ej por Simpson necesitamos 3 puntos, paso $h/2$

$x_0, x_{1/2}, x_1$

$$\frac{h}{6}\Big[\underbrace{f(t_0,y_0)}_{k_1} + \underbrace{4f(t_{1/2}, y_{1/2})}_{2k_2+2k_3} + \underbrace{f(t_1,y_1)}_{k_4}\Big]$$

$k_1 = f(t_0, y_0)$

$k_2 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}k_1)$

$k_3 = f(t_0 + \frac{h}{2}, y_0 + \frac{h}{2}k_2)$

$k_4 = f(t_1, y_0 + hk_3)$

↑ ya que k_1 y k_2 son aprox de $f(t,y) = y'$ que es la derivada

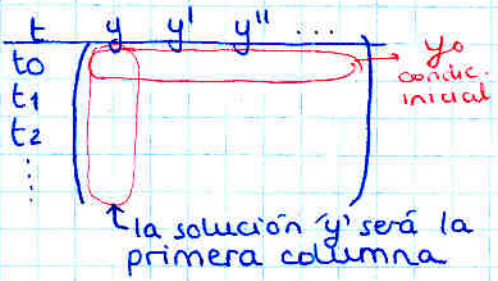
$$y(t_1) = y(t_0) + \frac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

y así avanzando por los k's

en MATLAB sería

```
function [t,y] = rungekut (f, a, b, y0, n)
h = (b-a)/n;
t = a : h : b;
y = zeros (size(t));
y(1) = y0;
for k = 1 : n
    k1 = f(t(k), y(k));
    tk2 = t(k) + h/2;
    k2 = f(tk2, y(k) + h/2 * k1);
    k3 = f(tk2, y(k) + h/2 * k2);
    k4 = f(t(k+1), y(k) + h * k3);
    y(k+1) = y(k) + h/6 * (k1 + 2*k2 + 2*k3 + k4);
end
```

la salida 'y' será una matriz



En un problema de Valor de frontera nos dan la ecuación diferencial

$$y'' = f(x, y, y')$$

y además:

(a) Condiciones de Dirichlet: $y(a) = \alpha$, $y(b) = \beta$

(b) Condiciones naturales: $\alpha_a y'(a) + \beta_a y(a) = \gamma_a$, $\alpha_b y'(b) + \beta_b y(b) = \gamma_b$

(c) Condiciones mixtas: $\alpha_a y'(a) + \beta_a y(a) = \gamma$, $y(b) = \gamma_b$

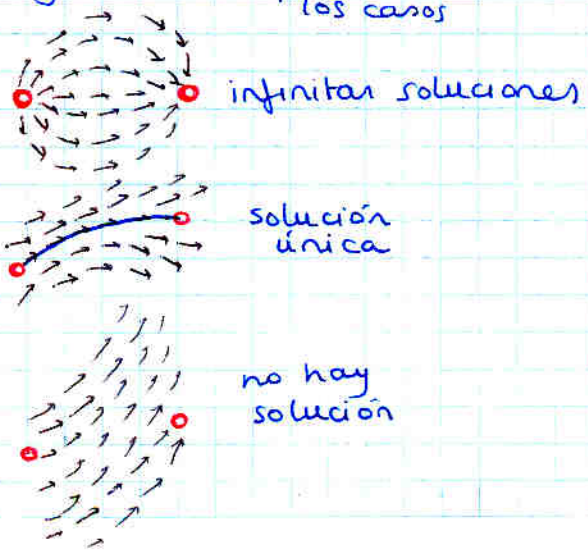
pag 190

- En un PVI, conociendo el punto de partida (valor y derivada) sabemos que son valores razonables, y es fácil continuar con valores aceptables a lo largo del dominio.
- Sin embargo, en un PVF o PC (problema contorno), la condición en el punto de partida no determina una única solución (sino ∞) y la elección al azar de una de ellas (suponiendo una derivada inicial al azar) con certeza conduce a una solución que no satisface la condición en el otro extremo.

ej PVI

ej PC

pueden darse todos los casos



Método de disparo lineal

Se aplica a problemas de contorno lineales; es decir:

$$(1)\quad \begin{cases} y'' = p(x)y' + q(x)y + r(x) \\ y(a) = \alpha \\ y(b) = \beta \end{cases} \qquad x \in [a,b]$$

El método es muy simple. En lugar de resolver el PF anterior, se resuelven los dos PVI siguientes por el método que se quiera (normalmente Runge-Kutta)

$$(2),(3)\quad \begin{cases} y_1'' = p(x)y_1' + q(x)y_1 + r(x) \\ y_1(a) = \alpha \\ y_1'(a) = 0 \end{cases} \qquad \begin{cases} y_2'' = p(x)y_2' + q(x)y_2 + \cancel{r(x)} \\ y_2(a) = 0 \\ y_2'(a) = 1 \end{cases}$$

resolver los PVI

$y_1(x)$ $\qquad$ $y_2(x)$

Entonces la solución del problema inicial vendrá dada por:

$$(4)\quad \boxed{y(x) = y_1(x) + \underbrace{\frac{\beta - y_1(b)}{y_2(b)}}_{\text{cte}} y_2(x)}$$

para comprobarlo,
- deriva 2 veces ambos lados
- sust y_1'' e y_2'' por sus expr
- agrupa $p(x)$ y $q(x)$

queda:

$$y'' = p(x)\underbrace{\left[y_1' + \frac{\beta - y_1(b)}{y_2(b)}\right]}_{y'} + q(x)\underbrace{\left[y_1'' + \frac{\beta - y_1(b)}{y_2(b)} y_2''\right]}_{y} + r(x)$$

Teorema
El sistema tiene solución única si:
- $p(x), q(x), r(x)$ continuas en $[a,b]$
- $q(x) > 0 \quad \forall x \in [a,b]$

en MATLAB

```
function  y = displin (f1, f2, a, b, α, β, n)
                          extremos   c.c.   nodos que usará runge-kutta
    [t, y1] = rungekut (f1, a, b, [α, 0], n)
    [t, y2] = rungekut (f2, a, b, [0, 1], n)
                                  c.I. del PVI
```

resolvemos los 2 PVI y obtenemos y_1 e y_2

$y \to y(:,1) = y_1(:,1) + \dfrac{\beta - y_1(end,1)}{y_2(end,1)} y_2(:,1)$

$y' \to y(:,2) = y_1(:,2) + \dfrac{\beta - y_1(end,1)}{y_2(end,1)} y_2(:,2)$

y_1, y_1', $y_1(b)$, $y_2(b)$, y_2, y_2'

obtenemos $y, y', y'', \ldots$ deberia ser un bucle for

podríamos calcular k previamente

siendo f_1 y f_2 las funciones adecuadas para que runge-kutta resuelva los PVI (2) y (3)
(es decir, convertir cada PVI en un sistema de orden 1 → cambio de variable)

Caso en que las C.C. no son las de Dirichlet

ej:
$$\begin{cases} y'' = p(x)y' + q(x)y + r(x) \quad \text{en } [a,b] \\ y(a) = \alpha \\ 3y(b) + y'(b) = \beta \end{cases}$$

En lugar de resolver eso, resolvemos dos PVI (los mismos que antes)

$$\begin{cases} y_1'' = p(x)y_1' + q(x)y_1 + r(x) \\ y_1(a) = \alpha \\ y_1'(a) = 0 \end{cases} \qquad \begin{cases} y_2'' = p(x)y_2' + q(x)y_2 \\ y_2(a) = 0 \\ y_2'(a) = 1 \end{cases}$$

pero esta vez no podemos construir directamente $y = y_1 + ky_2$

esta vez suponemos:

$$\boxed{y = k_1 y_1 + k_2 y_2}$$

hallamos k_1 y k_2 con las CC

$$\begin{cases} y(a) = \alpha \\ 3y(b) + y'(b) = \beta \end{cases}$$

$$\downarrow$$

$$\begin{cases} k_1 y_1(a) + k_2 y_2(a) = \alpha \\ 3[k_1 y_1(b) + k_2 y_2(b)] + [k_1 y_1'(b) + k_2 y_2'(b)] = \beta \end{cases}$$

$y_1(a), y_1'(a), y_2(a), y_2'(a)$ conocidos $\downarrow$

$$\begin{cases} k_1 \alpha + k_2 \cdot 0 = \alpha \\ 3[k_1 y_1(b) + k_2 y_2(b)] + [k_1 y_1'(b) + k_2 y_2'(b)] = \beta \end{cases}$$

$$\downarrow$$

$$\begin{cases} k_1 = 1 \\ k_2 = \dfrac{\beta - 3y_1(b) - y_1'(b)}{3y_2(b) + y_2'(b)} \end{cases}$$

a veces, mejor usar disparo no lineal ←

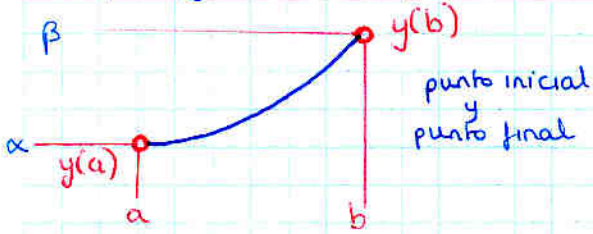
en nuestro ej ha salido fácil. si las 2 C.C. hubieran sido naturales $\alpha y(a) + \beta y'(a) = \gamma$ el sistema habría sido más complicado, pero igualmente tendría solución.

si sale indeterminado, tomar $k_1 = 1$ ya que es la única forma de que $y = k_1 y_1 + k_2 y_2$ incluya el término $r(x)$

Método de disparo no lineal

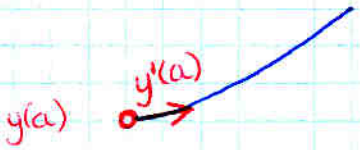
Sabemos que:

la solución a un problema de contorno (q. tenga sol única)



punto inicial y punto final

será también la solución a un PVI



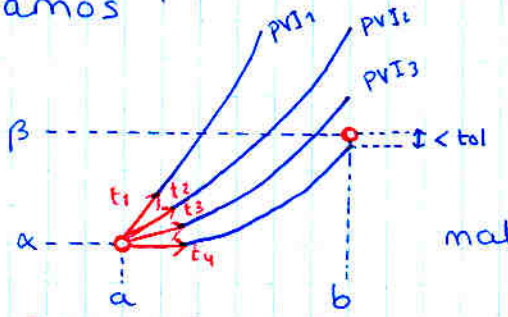
punto inicial y derivada inicial

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha \\ y(b) = \beta \end{cases}$$

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha \\ y'(a) = t \rightarrow \end{cases}$$

t es en principio desconocida

el sistema consiste en ir resolviendo el PVI para distintos t hasta que la solución en b este tan cerca de β como queramos



obtenemos la aproximación a la solución del PC con una sucesión de PVI

es como ir disparando en ciertos ángulos, (t) corigiendo el ángulo según los resultados anteriores, hasta acertar en la diana (β)

matematicamente:
Hay que tomar una sucesión $\{t_1, t_2, t_3, \dots\}$
tal que $\lim_{k\to\infty} y(b, t_k) = \beta$

Todo esto se cumplirá si el problema original tiene solución unica

Teorema

Supongamos: $f(x,y,y')$, $\frac{\partial f}{\partial y}$, $\frac{\partial f}{\partial y'}$ } continuas en D

$D = \{(x,y,y') ;\ x \in [a,b],\ y(a)=\alpha,\ y(b)=\beta\}$

- $\frac{\partial f}{\partial y}(x,y,y') > 0$ en D
- $\exists M > 0 : \left|\frac{\partial f}{\partial y}(x,y,y')\right| \le M$

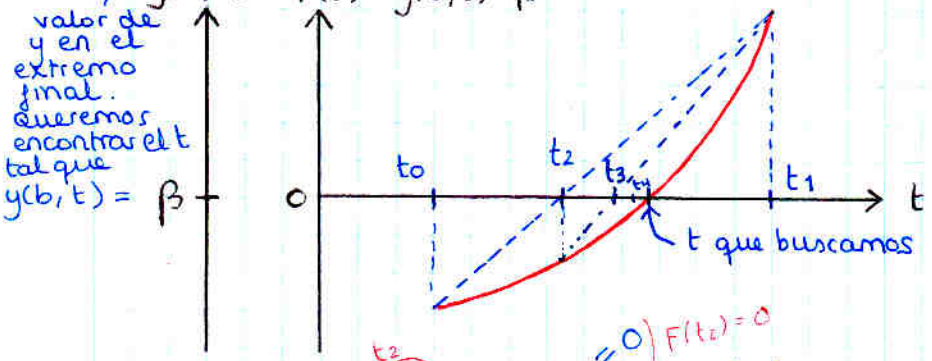
$\Rightarrow \exists!$ P.C.(f)

La siguiente cuestión es cómo elejir los parámetros t_k
Hay varios métodos: método de la secante, método de Newton

- método secante

$y(b,t)$ valor de y en el extremo final. Queremos encontrar el t tal que $y(b,t) = \beta$

$F(t) = y(b,t) - \beta$ → dejamos F tal que coincida con lo que queremos anular



gráficamente: tomamos un t_0, tal que la sol del PVI esté por debajo de β en b y un t_1 tal que la sol del PVI esté por encima de β en b.
Sabemos entonces que el t que buscamos está entre ambos.
Suponemos que $y(b,t)$ varía linealmente con t para obtener t_2 donde $y(b,t) = \beta$ (si variara lin) y así sucesivamente $t_3, t_4, \dots$

suponiendo que es recta: $\frac{t - t_1}{t_0 - t_1} = \frac{F(t) - F(t_1)}{F(t_0) - F(t_1)}$ (t_2, $F(t_2) = 0$)

despejando t.

$$t_k = t_{k-1} - \frac{y(b,t_{k-1}) - \beta}{y(b,t_{k-1}) - y(b,t_{k-2})} \cdot (t_{k-1} - t_{k-2})$$

así obtenemos la sucesion t_k

recuerda que lo que haremos será resolver

Para cada t_k el PVI (mediante Runge-Kutta)

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha \\ y'(a) = t_k \end{cases}$$

hasta que la solución en b esté cerca de β

$$t_k = t_{k-1} - \frac{[y(b, t_{k-1}) - \beta]}{y(b, t_{k-1}) - y(b, t_{k-2})}(t_{k-1} - t_{k-2})$$

- Método Newton

convertiremos el PC en un PVI por ejemplo

$$PC \begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha \\ y(b) = \beta \end{cases}$$

$$PVI \begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha \\ y'(a) = t_k \end{cases}$$

queremos ir hallando t_k hasta que $y(b) = \beta$ es decir $y(b) - \beta = F(t) = 0$

$$t_k = t_{k-1} - \frac{F(t_{k-1})}{F'(t_{k-1})}$$

POR DEFINICIÓN del método de Newton para hallar t que anule a $F(t)$: $F(t) = 0$

$F(t)$ es lo que queremos que sea cero cuando: t_k $k \to \infty$ (depende de las cc)

en este caso (caso típico): $F(t) = y(b) - \beta$

$$F'(t) = \frac{dy(b)}{dt}$$

PROBLEMA: No conocemos $\frac{dy(b)}{dt}$ ya que solo tenemos valores de $y(b)$ para $t_0, t_1, \ldots$

NOTA: recalcaremos que $y(b)$ depende de t_k, $y(b, t_k)$

Definimos $z = \frac{\partial y}{\partial t}$

$$z(x, t) = \frac{\partial y}{\partial t}(x, t)$$

derivando la ec. dif:

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha \\ y(b) = \beta \\ y'(a) = t_k \end{cases}$$

derivando se obtiene

$$\frac{d}{dt} y'' = \frac{d}{dt} f(x, y, y')$$

$$= \frac{d}{dx} f(x, y, y') \frac{dx}{dt}^{0} + \frac{d}{dy} f(x, y, y') \frac{dy}{dt}^{z} + \frac{d}{dy'} f(x, y, y') \frac{dy'}{dt}^{z' = \frac{dz}{dx}}$$

$$\frac{d}{dt} y(a) = 0$$

$$\frac{d}{dt} y(b) = 0 \qquad \frac{d}{dt} y'(a) = 1$$

se obtiene

$$\begin{cases} z'' = \frac{\partial f}{\partial y}(x, y, y') z + \frac{\partial f}{\partial y'}(x, y, y') z' \\ z(a) = 0 \\ z'(a) = 1 \end{cases}$$

Este PVI sirve para el caso de CC de Dirichlet. En otros casos habría que modificar el programa

y ya podremos usar $F'(t) = \frac{dy(b)}{dt} = z(b)$ resolviendo z como un PVI adicional

Conclusión: Disparo no lineal. Método Newton. CC Dirichlet

Se resolverían los dos PVI:

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha \\ y'(a) = t_k \end{cases} \qquad \begin{cases} z'' = \dfrac{df}{dy} z + \dfrac{df}{dy'} z' \\ z(a) = 0 \\ z'(a) = 1 \end{cases}$$

que se resolverían de una sola vez con un sistema

$$\left.\begin{aligned} y_1 &= y \\ y_2 &= y' \\ y_3 &= z \\ y_4 &= z' \end{aligned}\right\} \longrightarrow \left\{\begin{aligned} y_1' &= y_2 \\ y_2' &= f(x, y_1, y_2) \\ y_3' &= y_4 \\ y_4' &= \frac{\partial f}{\partial y_1} y_3 + \frac{\partial f}{\partial y_2} y_4 \end{aligned}\right.$$

C.I. $\quad y_1(a) = \alpha$, $y_2(a) = t_k$, $y_3(a) = 0$, $y_4(a) = 1$

PVI: $y = \text{rungekut}('f_s', a, b, [\alpha, t_k, 0, 1])$ (↑ función del sistema)

Iterar dicho PVI modificando t_k $\qquad t_{k+1} = t_k - \dfrac{F(t)}{F'(t)}$

$$t_{k+1} = t_k - \frac{y(b, t_k) - \beta}{z(b, t_k)} = t_k - \frac{y(end, 1) - \beta}{y(end, 3)}$$

Algoritmo en Matlab:

Entrada: f_s, a, b, α, β, n, tol, maxiter

Salida: y

Inicializar: h, $t = \frac{\beta - \alpha}{b - a}$ (típico valor), incr = tol + 1, k = 1

```
while incr > tol & k < maxiter
    y = rungekut('fs', [a,b], [α, tk, 0, 1], n)
    incr = abs(y(end,1) - β);
    tk+1 = tk - (y(end,1) - β) / y(end,3)
    k = k + 1
end
```

el programa funcionará para las CC $y(a) = \alpha$, $y(b) = \beta$

En cuanto estas cambien, todo para a ser distinto

veamos ejemplos:

ejemplo:

$$y'' = -\frac{1}{x}y' + \frac{1}{x^2}y + x \qquad x \in [1,2] \qquad \text{C.C. } \begin{matrix} y(1)=0 \\ y(2)+y'(2)=0 \end{matrix}$$

NOTA: es lineal ¿y si usamos disparo lineal?

$$\text{PVI (1)} \begin{cases} y'' = -\frac{1}{x}y' + \frac{1}{x^2}y + x \\ y(1) = 0 \\ y'(1) = 0 \end{cases} \qquad \text{PVI (2)} \begin{cases} y'' = -\frac{1}{x}y' + \frac{1}{x^2}y \\ y(1) = 0 \\ y'(1) = 1 \end{cases}$$

$$y = k_1 y_1 + k_2 y_2$$

$$\begin{cases} y(1) = k_1 \underbrace{y_1(1)}_{0} + k_2 \underbrace{y_2(1)}_{0} = 0 \\ y(2) + y'(2) = k_1 y_1(2) + k_2 y_2(2) + k_1 y_1'(2) + k_2 y_2'(2) \end{cases} \Big\} \text{ sistema para hallar } k_1 \text{ y } k_2$$

sale indet → forzamos $k_1 = 1$ (siempre debe ser lo) y sacamos k_2.

Será más fácil el método disparo no lineal

sigamos por el método no lineal

$$\begin{cases} y'' = -\frac{1}{x}y' + \frac{1}{x^2}y + x \qquad x \in [1,2] \\ y(1) = 0 \\ y(2) + y'(2) = 0 \end{cases}$$

Nos planteamos para ello el PVI

$$(1) \begin{cases} y'' = -\frac{1}{x}y' + \frac{1}{x^2}y + x \\ y(1) = 0 \\ y'(1) = t_k \end{cases}$$

queremos ir hallando t_k hasta que se cumpla $F(t) = y(2) + y'(2) = 0$

y para hallar la sucesión t_k por Newton necesitaremos t_b.

$$(2) \begin{cases} z'' = \underbrace{+\frac{1}{x^2}}_{\frac{\partial f}{\partial y}} z \underbrace{-\frac{1}{x}}_{\frac{\partial f}{\partial y'}} z' \\ z(1) = 0 \\ z'(1) = 1 \end{cases}$$

lo que queremos que se anule cuando $k \to \infty$ esta vez en lo que conocemos (esta vez no es $y(b)$ sino $y(b) + y'(b)$)

por tanto:

$$\boxed{F(t) = y(b,t) + y'(b,t)}$$

Newton

$$\boxed{t_{k+1} = t_k - \frac{F(t)}{F'(t)}} = t_k - \frac{y(2,t_k) + y'(2,t_k)}{\frac{dy}{dt}(2,t_k) + \frac{dy'}{dt}(2,t_k)}$$

para converge a $F(t) = 0$ que es lo que queremos

$$= t_k - \frac{y(2,t_k) + y'(2,t_k)}{z(2,t_k) + z'(2,t_k)}$$

(se utilizarán las 4 columnas de salida de rungekut. y, y', z, z') al meterle el sistema que incluye a (1) y (2)

ejemplo:
$$\begin{cases} y'' = \frac{1}{2y}(y')^2 - 2y & x \in [\frac{\pi}{4}, \frac{\pi}{2}] \\ y'(\frac{\pi}{4}) = 1 \\ y'(\frac{\pi}{2}) = 0 \end{cases}$$

1. Lo convertimos a un PVI

$$\begin{cases} y'' = \frac{1}{2y}(y')^2 - 2y & x \in [\frac{\pi}{4}, \frac{\pi}{2}] \\ y(\frac{\pi}{4}) = t_k \\ y'(\frac{\pi}{4}) = 1 \end{cases}$$

extrañamente esta vez lo que no conocemos es la "posición inicial" y sí que conocemos el "ángulo de tiro"

Además esta vez lo que hay que acertar variando t_k no es "el centro de la diana" $y(\frac{\pi}{2}) = \beta$ sino el "ángulo con que damos en la diana" $y'(\frac{\pi}{2}) = 0$

2. Introducimos $z = \frac{dy}{dt}$ →
$$\begin{cases} z'' = \frac{\partial f}{\partial y} z + \frac{\partial f}{\partial y'} z' \\ z(\pi/4) = 1 \\ z'(\pi/4) = 0 \end{cases}$$

3. Hallamos la fórmula de Newton correspondiente para la sucesión t_k.

¿Qué pretendemos conseguir usando el t_k cuando $k \to \infty$?

pretendemos que se cumpla $y'(\frac{\pi}{2}) = 0$

llamamos $F(t) = y'(\frac{\pi}{2})$

y queremos $F(t) = 0$

por Newton

$$t_{k+1} = t_k - \frac{F(t)}{F'(t)} = t_k - \frac{y'(\frac{\pi}{2}, t_k)}{z'(\frac{\pi}{2}, t_k)}$$

4. convertimos los sistemas (1) y (2) en un sistema de orden 1

$$\begin{aligned} y_1 &= y \\ y_2 &= y' \\ y_3 &= z \\ y_4 &= z' \end{aligned} \qquad \begin{cases} y_1' = y_2 \\ y_2' = \frac{1}{2y_1}(y_2)^2 - 2y_1 \\ y_3' = y_4 \\ y_4' = \frac{\partial f}{\partial y_1} y_3 + \frac{\partial f}{\partial y_2} y_4 \end{cases} \quad \left(= \frac{\partial f}{\partial y} z + \frac{\partial f}{\partial y'} z' \right)$$

PVI
$$\begin{cases} y_1(\frac{\pi}{4}) = t_k \\ y_2(\frac{\pi}{4}) = 1 \\ y_3(\frac{\pi}{4}) = 1 \\ y_4(\frac{\pi}{4}) = 0 \end{cases}$$

Vamos resolviendo el PVI para cada t_k hasta que $y'(\pi/2)$ esté tan cerca de cero como queramos

ejemplo: orden 3

$$\begin{cases} y''' = f(x, y, y', y'') & x \in [a, b] \\ y(a) = \alpha \\ y'(a) = \beta \\ y(b) = \gamma \end{cases}$$

lo convertimos en un PVI con parámetro (método disparo)

$$(1)\quad \begin{cases} y''' = f(x, y, y', y'') \\ y(a) = \alpha \\ y'(a) = \beta \\ y''(a) = t_k \end{cases}$$

por Newton

$$t_{k+1} = t_k - \frac{F(t)}{F'(t)}$$

$F(t)$ siempre es lo q. queremos q. se anule

$$F(t) = y(b, t_k) - \gamma$$

por tanto:

$$t_{k+1} = t_k - \frac{y(b, t_k) - \gamma}{\frac{d}{dt} y(b, t_k)}$$

Para conocer el denominador, como siempre, definimos:

$$z(x, t) = \frac{\partial}{\partial t} y(b, t)$$

$$(2)\quad \begin{cases} z''' = \frac{\partial f}{\partial y} z + \frac{\partial f}{\partial y'} z' + \frac{\partial f}{\partial y''} z'' \\ z(a) = 0 \ (= \partial\alpha/\partial t) \\ z'(a) = 0 \ (= \partial\beta/\partial t) \\ z''(a) = 1 \ (= \partial t_k/\partial t) \end{cases}$$

Si convertimos los sistemas (1) y (2) a un unico sistema de orden 1, podemos ir variando t_k usando:

$$t_{k+1} = t_k - \frac{y(b, t_k) - \gamma}{z(b, t_k)}$$

---

ejemplo: orden 3 : Disparo hacia atrás

$$\begin{cases} y''' = f(x, y, y', y'') \\ y(a) = \alpha \\ y(b) = \beta \\ y'(b) = \gamma \end{cases} \quad x \in [a, b]$$

si lo convirtieramos a un PVI en a, tendriamos dos parametros. Habria que hacer Newton de dos variables (sistema de ecs)

mejor evitarlo y hacer el PVI en b, queda:

$$\begin{cases} y''' = f(x, y, y', y'') \\ y(b) = \beta \\ y'(b) = \gamma \\ y''(b) = t_k \end{cases} \quad x \in [b, a]$$

¿Funcionará en nuestros programas?

Si, ya que $h = \frac{a-b}{2} < 0$ (negativa)

seguimos como arriba:

Errata del libro p 267 b)

$$\begin{cases} u'' + u' = 0 \\ u(1) = 540 \\ u'(2) = 0'0f3\,(u(2) - 20) \end{cases} \longrightarrow$$

pasamos a PVI

$$\begin{cases} u'' = -f u' \\ u(1) = 540 \\ u'(1) = t_k \end{cases}$$

definimos

$$z(x, t) = \frac{\partial}{\partial t} u(b, t)$$

$$z'' = \frac{\partial f}{\partial u} z + \frac{\partial f}{\partial u'} z'$$

$$z'' = -\frac{1}{f} z'$$

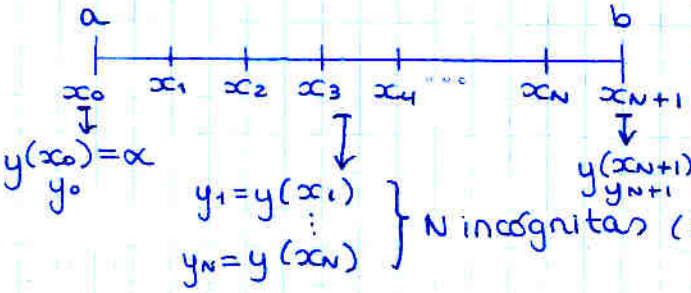
$$t_{k+1} = t_k - \frac{u'(2) - 0'0f3\,(u(2) - 20)}{z'(2) - 0'0f3\, z(2)}$$

Método de diferencias finitas para problemas lineales

Tenemos el problema lineal:

$$\begin{cases} y'' = p(x)y' + q(x)y + r(x) & x \in [a,b] \\ y(a) = \alpha \\ y(b) = \beta \end{cases}$$

- Discretizamos el intervalo $[a,b]$ en $N+1$ subintervalos, $N+2$ puntos



$$x_i = a + ih \qquad i = 0, 1, \dots, N+1$$

$$h = \frac{b-a}{N+1}$$

} N incógnitas (saldrá sistema $N \times N$)

- Ahora hay que sustituir y' e y'' por un cociente incremental.

Podríamos usar: $y'(x) \simeq \underbrace{\frac{y(x+h)-y(x)}{h}}_{\text{aprox hacia delante}} \simeq \underbrace{\frac{y(x)-y(x-h)}{h}}_{\text{aprox. hacia atrás}}$

pero el error es de orden h.

Podemos obtener expresiones mejores del siguiente modo:

Desarrollando $y(x)$ por Taylor hasta grado 3

$$y(x+h) \simeq y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \frac{h^3}{6}y'''(\alpha) \qquad \alpha \in [x, x+h]$$
$$y(x-h) \simeq y(x) - hy'(x) + h^2/2\, y''(x) - h^3/6\, y'''(\alpha')$$

que podemos expresar con notación más sencilla como:

$y(x_i) = y_i$, $y(x_i+h) = y_{i+1}$

$$① \quad y_{i+1} \simeq y_i + h y'_i + h^2/2\, y''_i + h^3/6\, y'''(\alpha)$$
$$② \quad y_{i-1} \simeq y_i - h y'_i + h^2/2\, y''_i - h^3/6\, y'''(\alpha')$$

restando ① − ② y despejando y' se obtiene:

$$y'_i \simeq \frac{y_{i+1} - y_{i-1}}{2h} - \frac{h^2}{6}y'''(\mu_i)$$

Los errores son de orden 2 $E(h^2)$

sumando ① + ② y despejando y'' se obtiene

$$y''_i = \frac{2y_i - y_{i+1} - y_{i-1}}{h^2} - \frac{h^2}{24}[\dots]$$

por tanto podemos sustituir estas aproximaciones en la ecuación inicial

$$\frac{2y_i - y_{i+1} - y_{i-1}}{h^2} = p(x_i)\frac{y_{i+1} - y_{i-1}}{2h} + q(x_i)y_i + r(x_i) \qquad i = 1, 2, \dots, N$$

reorganizando adecuadamente se tiene

$$-\left[1+\frac{h}{2}p(x_i)\right]y_{i-1} + \left[2+h^2q(x_i)\right]y_i - \left[1-\frac{h}{2}p(x_i)\right]y_{i+1} = -h^2 r(x_i) \qquad i = 1, 2, \dots, N$$

(en $i=1$: $y_{i-1}=\alpha$ término indep; en $i=N$: $y_{i+1}=\beta$)

Matricialmente:

$$\begin{pmatrix} 2+h^2q(x_1) & -1+\frac{h}{2}p(x_1) & 0 & \cdots & 0 \\ -1-\frac{h}{2}p(x_2) & 2+h^2q(x_2) & -1+\frac{h}{2}p(x_2) & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \\ & & & & \\ & & & & \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} = \begin{pmatrix} -h^2r(x_1)+\left[1+\frac{h}{2}p(x_1)\right]\alpha \\ -h^2r(x_2) \\ \vdots \\ -h^2r(x_{N-1}) \\ -h^2r(x_N)+\left[1+\frac{h}{2}p(x_N)\right]\beta \end{pmatrix}$$

A matriz tridiagonal · $y = b$

$Ay = b$

Bajo ciertas condiciones el sistema tiene solución única:

Teorema: $p(x), r(x), q(x)$ continuas en $[a,b]$; $q(x) \geq 0$ en $[a,b]$; $h < \frac{2}{L}$ siendo $L = \max|p(x)|$ ⟹ El sistema tridiagonal tiene solución única

se puede aplicar cualquier algoritmo de resolución de sistemas lineales directo o iterativo.

Primero veamos en Matlab como crearíamos la matriz:

```
function y = diflin(p, q, r, a, b, alpha, beta, N)
% p, q, r: p(x), q(x), r(x)
% [a,b]
% alpha = y(a), beta = y(b)

h = (b-a)/(N+1);
x = a:h:b;          (N+2 puntos)
x = x(2:N+1);       (eliminamos ptos inicial y final, conocidos)

qx = feval(q, x);
px = feval(p, x);
rx = feval(r, x);

a = 2 + h^2 .* qx          (b es menos largo que a, tiene N-1 elementos)
b = -1 + (h/2) px(1:end-1)
c = -1 - (h/2) px(2:end)

d = -h^2 rx
d(1) = d(1) + [1 + (h/2) px(1)] * alpha
d(end) = d(end) + [1 + (h/2) px(N)] * beta
```

$$\begin{pmatrix} a & b & & \\ c & a & b & \\ & \ddots & \ddots & \ddots \\ & & & \end{pmatrix} \begin{pmatrix} \\ \\ \\ \\ \end{pmatrix} = \begin{pmatrix} d \\ \\ \\ \\ \end{pmatrix}$$

ahora no hace falta que creemos la matriz en sí?
El algoritmo de Crout está hecho para sistemas tridiagonales (en ellos es mucho mejor que Gauss) y le basta con saber los vectores a, b, c, d.
Veamos como funciona el...

Algoritmo de Crout (sistemas tridiagonales)

Idea general: tenemos: $A \cdot x = D$

(1) hacemos: $A = L \cdot U$ (triang inf · triang sup)

por tanto: $L \cdot U \cdot x = D$

(2) resolvemos: $L \cdot Z = D$ $\xrightarrow{\text{sust}}$ obtenemos Z

(3) y sabiendo: $U \cdot x = Z$ $\xrightarrow{\text{sust}}$ obtenemos x

(1) Descomposición LU $(A = L \cdot U)$

$$U \begin{pmatrix} 1 & u_1 & & & \\ & 1 & u_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & u_{N-1} \\ & & & & 1 \end{pmatrix}$$

$$L \begin{pmatrix} l_1 & & & & \\ c_1 & l_2 & & & \\ & c_2 & l_3 & & \\ & & \ddots & \ddots & \\ & & & c_{N-1} & l_N \end{pmatrix} \begin{pmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & c_2 & a_3 & b_3 & \\ & & \ddots & \ddots & \ddots \\ & & & c_{N-1} & a_N \end{pmatrix} A$$

se ve claramente:

1ª columna: $l_1 = a_1$

2ª columna: $u_1 l_1 = b_1 \rightarrow u_1 = b_1/l_1$

$u_1 c_1 + l_2 = a_2 \rightarrow l_2 = a_2 - u_1 c_1$

3ª columna: $u_2 l_2 = b_2 \rightarrow$

$u_2 c_2 + l_3 = a_3 \rightarrow$

en general se obtiene:

$$\begin{cases} u_{i-1} = \dfrac{b_{i-1}}{l_{i-1}} \\ l_i = a_i - u_{i-1} c_{i-1} \end{cases}$$

(2) $LZ = D$.

$$\begin{pmatrix} l_1 & 0 & & & \\ c_1 & l_2 & & & \\ & c_2 & l_3 & & \\ & & \ddots & \ddots & \\ & & & c_{N-1} & l_N \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{pmatrix}$$

$z_1 l_1 = d_1 \rightarrow z_1 = d_1/l_1$

$c_1 z_1 + l_2 z_2 = d_2 \rightarrow z_2 = \dfrac{d_2 - c_1 z_1}{l_2}$

$\rightarrow$ $$z_i = \frac{d_i - c_{i-1} z_{i-1}}{l_i}$$

(3) $UX = Z$

$$\begin{pmatrix} 1 & u_1 & & & \\ & 1 & u_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & u_{N-1} \\ & & & & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{N-1} \\ z_N \end{pmatrix}$$

$x_N = z_N$

$x_{N-1} + u_{N-1} x_N = z_{N-1}$

$\hookrightarrow x_{N-1} = z_{N-1} - u_{N-1} x_N$

en general

$$x_{i-1} = z_{i-1} - u_{i-1} x_i$$

Mirando las fórmulas recuadradas, vemos que se puede solucionar el sistema sin mas que saber los vectores a, b, c, d

en MATLAB:

function x = cout (a, b, c, d)

recuerda:

$$\begin{pmatrix} a & b & & \\ c & \ddots & \ddots & \\ & \ddots & \ddots & b \\ & & c & a \end{pmatrix} \begin{pmatrix} x \\ \vdots \\ \vdots \\ x \end{pmatrix} = \begin{pmatrix} d \\ \vdots \\ \vdots \\ d \end{pmatrix}$$

inicializar variables:

$l_1 = a_1$

$z_1 = d_1 / l_1$

$x_1 = z_1$

$u_1 = b_1 / l_1$

bucle ($i = 2 : N$)

$l_i = a_i - u_{i-1} c_{i-1}$

$u_i = b_i / l_i$

$z_i = \frac{1}{l_i} (d_i - c_{i-1} z_{i-1})$

bucle de sust inversa ($j = n : -1 : 1$)

$x_j = z_j - u_j x_{j+1}$

Con Extrapolación de Richardson (pagina 210)

Diferencias finitas: Caso no lineal

$$\begin{cases} y'' = f(x, y, y') \\ x \in [a,b] \\ y(a) = \alpha \\ y(b) = \beta \end{cases}$$

dividimos el intervalo en $N+1$ subintervalos:

$x_i = a + ih \quad i = 0, 1, \dots, N+1$

$h = \frac{b-a}{N+1}$

queda:

$$y''(x_i) = f(x_i, y(x_i), y'(x_i))$$

sustituyendo y' e y'' por sus aprox.

$$\frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} = f\left(x_i, y(x_i), \frac{y(x_{i+1}) - y(x_{i-1})}{2h}\right) + O(h^2)$$

$i = 1, \dots, N$

denotamos $y(x_i) := y_i$

$y_0 = \alpha$

$y_{N+1} = \beta$

$$\boxed{\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} = f\left[x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}\right] \quad i = 1, 2, \dots, N}$$

con esto obtenemos un sistema no lineal NxN

$$\begin{cases} -\alpha + 2y_1 - y_2 + h^2 f(x_1, y_1, \frac{y_2 - \alpha}{2h}) = 0 \\ -y_1 + 2y_2 - y_3 + h^2 f(x_2, y_2, \frac{y_3 - y_1}{2h}) = 0 \\ -y_2 + 2y_3 - y_4 + h^2 f(x_3, y_3, \frac{y_4 - y_2}{2h}) = 0 \\ \vdots \\ -y_{N-1} + 2y_N - \beta + h^2 f(x_N, y_N, \frac{\beta - y_{N-1}}{2h}) = 0 \end{cases}$$

Este sistema no lineal tiene solución y es única bajo ciertas condiciones (ver página 211)

Lo resolvemos mediante el método de Newton:

aprox inicial $\begin{pmatrix} y_1^0 \\ y_2^0 \\ y_3^0 \\ \vdots \\ y_N^0 \end{pmatrix} = Y_0$

$Y_1 = Y_0 + V_0$

$\begin{pmatrix} y_1^1 \\ y_2^1 \\ y_3^1 \\ \vdots \\ y_N^1 \end{pmatrix} = \begin{pmatrix} y_1^0 \\ y_2^0 \\ y_3^0 \\ \vdots \\ y_N^0 \end{pmatrix} + \begin{pmatrix} V_1^0 \\ V_2^0 \\ V_3^0 \\ \vdots \\ V_N^0 \end{pmatrix}$

$Y_2 = Y_1 + V_1$

$Y_3 = Y_2 + V_2$

$\vdots \quad \vdots$ y así sucesivamente formando una sucesión convergente a Y exacta

Y

para ello, necesitamos obtener en cada iteración v_i, que se obtiene resolviendo el sistema:

derivada $\frac{\partial f}{\partial y}$; $\frac{\partial f}{\partial z} = \frac{\partial f}{\partial y'}$

$$\begin{pmatrix} 2+h^2 f_y(x_1,y_1,z_1) & -1+\left(\frac{h}{2}\right) f_z(x_1,y_1,z_1) & 0 & \cdots \\ -1+\left(\frac{h}{2}\right) f_z(x_2,y_2,z_2) & 2+h^2 f_y(x_2,y_2,z_2) & 0 & \cdots \\ 0 & \ddots & \ddots & \ddots \\ & & & 2+h^2 f_y(x_N,y_N,z_N) \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_N \end{pmatrix} = - \begin{pmatrix} -\alpha + 2y_1 - y_2 + h^2 f(x_1,y_1,z_1) \\ -y_1 + 2y_2 - y_3 + h^2 f(x_2,y_2,z_2) \\ \vdots \\ -y_{N-1} + 2y_N + h^2 f(x_N,y_N,z_N) - \beta \end{pmatrix}$$

$J\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$ jacobiano (matriz tridiagonal)

resolvemos el sistema por Crout

siendo $z_i = \dfrac{y_{i+1} - y_{i-1}}{2h} \simeq y_i'$

en MATLAB

function y = difnol(f, fy, fz, y0, a, b, α, β, N, tol, maxiter)

% Partición: $h = \frac{b-a}{N+1}$ x = a:h:b (N+2 puntos) x = x(2:N+1) (N puntos, hemos quitado α y β)

(*) y = [α, y0, β] % [α, y1, y2, y3, ..., yN, β] (1 2 3 4 ... N+1 N+2)

cuidado, a diferencia del vector x e z hemos hecho que el vector y posea los extremos y0 e yN+1

incr = tol+1 k=0

while incr > tol & k < maxiter,

z = (y(3:N+2) - y(1:N)) / 2h

a = 2 + h^2 feval(fy, x, y(2:N+1), z) % diag principal

b = -1 + (h/2) feval(fz, x(1:N-1), y(2:N), z(1:N-1))

c = -1 - (h/2) feval(fz, x(2:N), y(3:N+1), z(2:N))

d = -(y(1:N) + 2y(2:N+1) - y(3:N+2) + h^2 feval(f, x, y(2:N+1), z)) % term. indep

v = crout(a, b, c, d);

y(2:N+1) = y(2:N+1) + v;

incr = norm(v);

k = k+1;

end

para aclararse

y = [α, y1, y2, y3, ..., yN-1, yN, β] (1 2 3 4 ... N N+1 N+2)

x = [x1, x2, x3, ..., xN-1, xN]

z = [z1, z2, z3, ..., zN-1, zN] (1 2 3 ... N-1 N)

(*) NOTA: para no tener que introducir y0, podemos hacer que sea

$y0 = \alpha + i\frac{\beta-\alpha}{b-a}h$ $i = 1,2,\ldots,N$ $\Leftrightarrow$ $k = \frac{\beta-\alpha}{N+1}$ y = α:k:β y = y(2:N+1)

ejemplo:

$$\begin{cases} x^3 y'' = (y - xy')^2 \\ y(1) = 0 \\ y(2) = 2 \end{cases} \qquad x \in [1,2]$$

nuestro programa lo hará bien

$$f(x, y, y') = \frac{1}{x^3}(y - xy')^2$$
$$f(x, y, z) = \frac{1}{x^3}(y - xz)^2$$

$$f_y = \frac{\partial f}{\partial y} = \frac{1}{x^3} 2(y - xy')$$
$$f_z = \frac{\partial f}{\partial z} = -\frac{x}{x^3} 2(y - xy')$$

en este caso aplicamos el algoritmo directamente.

---

casos mas complicados:

- las condiciones no son las de Dirichlet

$$\text{ej:} \begin{cases} y'' - yy' = e^{-2x} - x + xe^{-x} & x \in [0,1] \\ y(0) + y'(0) = 1 \\ y(1) = 1 + 1/e \end{cases}$$

- Habria que 'seguir' el método de Newton (jacobiano, ...)
- Además cuando nos aparezca y_{-1} (al sust $i = 0$) (ya que ahora $y(0)$ tb es incognita)

$$y_0 + y'_0 = 1$$
$$y_0 + \frac{y_1 - y_{-1}}{2h} = 1 \quad \rightarrow \quad y_{-1} = -2h(1 - y_0) + y_1$$

---

otro caso mas complicado:

si aparece y'''

$$y'''_i = (y_i'')' = \left(\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}\right)'$$
$$= \frac{1}{h^2}(y_{i+1}' - 2y_i' + y_{i-1}') \quad \Big\} \; y_i = \frac{y_{i+1} - y_{i-1}}{2h}$$
$$= \frac{1}{2h^3}[y_{i+2} - 2y_{i+1} + 2y_{i-1} - y_{i-2}]$$

$O(h^2)$, la matriz ya no será tridiagonal sino pentadiagonal

Otra opción es hallar una aprox de y''' que diera una matriz tridiagonal, pero su precisión ya no sera $O(h^2)$

Métodos variacionales Rayleigh-Ritz

Creérselo:

[Bajo ciertas condiciones de $p(x)$, $q(x)$, $f(x)$]

La solución al problema de frontera

$$\left\{ -\frac{d}{dx}\left[p(x)\frac{dy}{dx}\right] + q(x)y = f(x) \right\}_{x\in[0,1]} \qquad y(0)=y(1)=0$$

es la función $y(x)$ que minimiza la integral

$$I(y(x)) = \int_0^1 \left[p(x)(y'(x))^2 + q(x)[y(x)]^2 - 2f(x)y(x) \right] dx$$

La integral I se minimiza en el subespacio generado por las funciones base $\{\phi_1, \phi_2, \phi_3, \dots\}$

- linealmente independientes
- verifican las c.c. $\phi_i(0) = \phi_i(1) = 0$

Hallaremos una aproximación a la solución $y(x)$ mediante la función combinación lineal de $\phi_1, \phi_2 \dots$ que minimize la integral I.

$$y(x) \simeq \phi(x) = \sum_{i=1}^{\tilde{n}} c_i \phi_i(x)$$

Hay que hallar los coeficientes que minimizan $\phi(x)$

Haciéndolo se obtiene el sistema lineal:

$$A \cdot c = b \rightarrow b_i = \int_0^1 f(x)\phi_j(x)\,dx$$

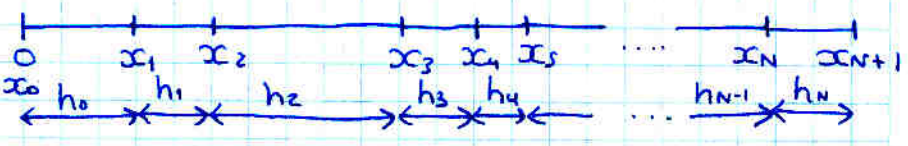
$$a_{ij} = \int_0^1 \left[p(x)\phi_i'(x)\phi_j'(x) + q(x)\phi_i(x)\phi_j(x) \right] dx$$

se obtiene matriz simétrica.

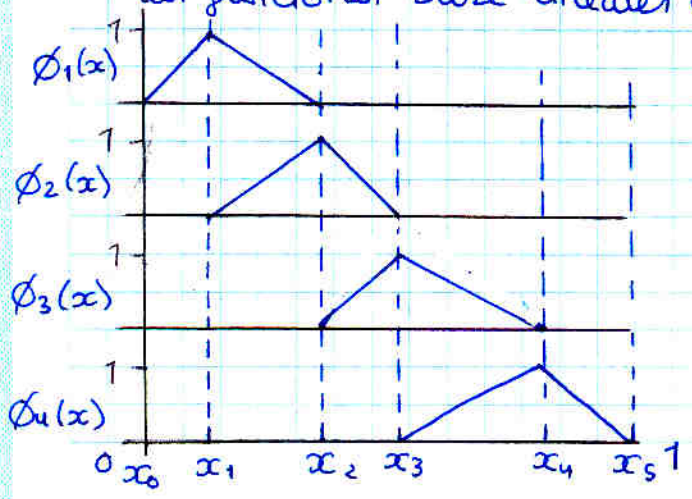
¿Cuales son las funciones ϕ_i?

Tomamos las funciones base lineales a trozos

si tenemos los puntos (no tienen porque ser equiespaciados)



las funciones base lineales a trozos son:



$$\text{i.e. } \phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{h_{i-1}} & x \in [x_{i-1}, x_i] \\ \frac{x_{i+1} - x}{h_i} & x \in [x_i, x_{i+1}] \\ 0 & \text{resto} \end{cases}$$

y por tanto:

$$\phi_i'(x) = \begin{cases} \frac{1}{h_{i-1}} & x \in [x_{i-1}, x_i) \\ -\frac{1}{h_i} & x \in [x_i, x_{i+1}] \\ 0 & \text{resto} \end{cases}$$

que cumplen las c.c. y son independientes $\phi_i\phi_j = 0 \quad \forall j \neq i-1, i, i+1$

$\phi_i'\phi_j' = 0$

Conociendo ϕ_i y ϕ_i' no hay mas que crear las matrices y resolver el sistema

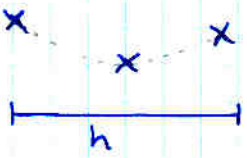
queda: $A \cdot c = b$ (será un sistema tridiagonal)

$$a_{ii} = \int_{x_{i-1}}^{x_i} \left(\frac{1}{h_{i-1}}\right)^2 p(x)\,dx + \int_{x_i}^{x_{i+1}} \left[-\frac{1}{h_i}\right]^2 p(x)\,dx + \ldots$$ (ver página 219)

Convendria, para calcular los valores de A, construir la función simpson vectorial en lugar de simpson (f, a, b, n)

simpsonv (y, h) $\rightarrow h = \frac{b-a}{1}$

↳ vector con $y(a)$ e $y(b)$ e $y(\frac{a+b}{2})$



ejemplo: $-\frac{d}{dx}(p(x)y') + q(x)y = f(x) \qquad x \in [0,1]$

con $p(x) = 1 \qquad y(0) = y(1) = 0$

$q(x) = \pi^2$

$f(x) = 2\pi^2 \operatorname{sen} \pi x$

al evaluar en los puntos de la partición como $\boxed{\phi_i(x_i) = 1}$

$$\boxed{\phi(x_j) = \sum_{i=1}^{\tilde{N}} c_i \phi_i(x_j) = c_j}$$

gracias a usar la base lineal a trozos

ejemplo: intervalo no es $[0,1]$

$$\begin{cases} -\frac{d}{dx}(xy') + 4y = g(x) & x \in [2,4] \\ y(2) = y(4) = 0 \end{cases}$$

C.V. hacemos: $x = mt + n \qquad t \in [0,1]$

$2 = 0 + m \}\ n = 2$
$4 = m + 2 \}\ m = 2$ $\rightarrow \quad t = \frac{x-2}{2}$

aplicando el C.V queda ... $g(mt+n)$

$$-\frac{d}{dt}\frac{dt}{dx}\left(x \frac{dy}{dt}\frac{dt}{dx}\right) + 4y = g(t)$$

$$-\frac{d}{dt}\frac{1}{2}\left((2t+2)\frac{1}{2}\frac{dy}{dt}\right) + 4y = g(t)$$

ya podemos aplicar el método

ejemplo: las C.C. no son nulas

$$-\frac{d}{dx}(p(x)y') + q(x)y = f(x) \qquad x \in [0,1]$$

$$y(0) = \alpha$$
$$y(1) = \beta$$

hacemos un C.V tal que $z(0) = z(1) = 0$

este es:

$$z = y - \beta x - (1-x)\alpha \qquad \text{[idea feliz]}$$
$$z' = y' - \beta + \alpha$$

aplicando el C.V.

$$-\frac{d}{dx}\left(p(x)\left(z' + \beta - \alpha\right)\right) + q(x)\left(z + \beta x + (1-x)\alpha\right) = f(x)$$

despejando adecuadamente:

$$-\frac{d}{dx}(p(x)z') + q(x)z = \underbrace{\frac{d}{dx}(p(x)(\beta-\alpha) - q(x)(\beta x + (1-x)\alpha) + f(x)}_{\text{nuestra nueva } f(x)}$$

ejemplo: el problema 8 tiene las dos pegas

- intervalo no es $[0,1]$
- C.C. no nulas

TEMA 4 DIFERENCIAS FINITAS PARA ECUACIONES EN DERIVADAS PARCIALES

Introducción

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu = G$$

Resolvemos ecuaciones EDP.

Notación: $Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G$

En particular: $\Delta = B^2 - 4AC$

Ec. de Ondas	$u_{tt} - c^2 u_{xx} = 0$	$\Delta > 0$ - hiperbólica
Ec. del Calor	$u_t - c u_{xx} = 0$, $c > 0$	$\Delta = 0$ - parabólica
Ec. de Laplace	$u_{xx} + u_{yy} = 0$	$\Delta < 0$ - elíptica

Siempre tenemos dos métodos:

- Explícitos: las incognitas se van calculando sucesivamente en terminos de valores conocidos o ya calculados
  - sencillo
  - Inestable: debido a la acumulación de errores de redondeo.
- Implícitos: se resuelve un sistema de ecuaciones en cada paso para calcular las incógnitas
  - Más complejos
  - Estables.

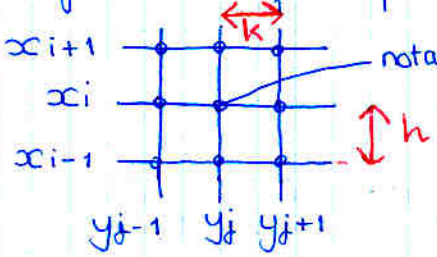
Diferencias Finitas

- Diferencias primeras
  - hacia delante: $u_{x\,i,j} \simeq \frac{u_{i+1,j} - u_{i,j}}{h}$
  - hacia atrás: $u_{x\,i,j} \simeq \frac{u_{i,j} - u_{i-1,j}}{h}$

  Error $O(h)$
  - diferencias simétricas: $u_{x\,i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2h}$ Error $O(h^2)$
- Diferencias segundas
  - diferencias simetricas: $u_{xx\,i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$ Error $O(h^2)$

Para demostrar el orden del error se hace el desarrollo de Taylor (p. 283)

Se reduce el dominio a un número finito de puntos y se sustituyen las derivadas parciales por los cocientes de diferencias que aparecen arriba



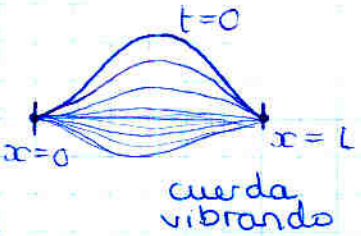
notación: $u(x_i, y_j) := u_{i,j}$

Convergencia y estabilidad

- convergencia: la solución por diferencias finitas $\xrightarrow{h,k \to 0}$ solución exacta
- Estabilidad: si podemos controlar el error de redondeo

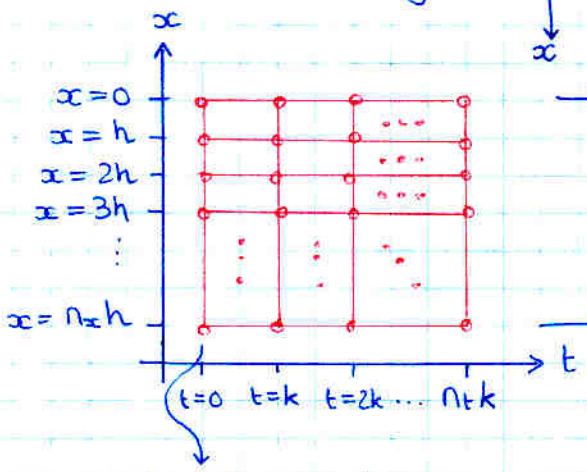
- Ecuaciones Hiperbólicas (Ec de Ondas)

$$\begin{cases} u_{tt} = c^2 u_{xx} & 0 < x < L \\ u(x,0) = f(x) & \text{posición inicial} \\ u_t(x,0) = g(x) & \text{velocidad inicial} \\ u(0,t) = l(t) \quad (\text{típico } l(t)=0) & \text{extremos de las cuerdas} \\ u(L,t) = r(t) \quad (\text{típico } r(t)=0) & \text{típicamente fijos en cero} \end{cases}$$



cuerda vibrando

- Discretizamos x y t



$\rightarrow u(1,:) = f(t)$ feval(f, t)

$\rightarrow u(n_x+1, :) = r(t)$ feval(r, t)

$u(:,1) = f(x)$ feval(f, x)

con las CI obtenemos $u(1,:)$, $u(n_x+1,:)$, $u(:,1)$

(aun no hemos aplicado la CI) $u_t(x,0) = g(x)$

- convertimos la ecuación en cocientes de diferencias

$$\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = c^2 \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

podemos despejar los valores de u en t_{j+1} en función de los valores en t_j y t_{j-1} (ya habrán sido calculados en pasos anteriores)

(1) $$\boxed{u_{i,j+1} = \alpha^2(u_{i+1,j} + u_{i-1,j}) + 2(1-\alpha^2)u_{i,j} - u_{i,j-1}} \qquad \alpha = \frac{ck}{h}$$

Para comenzar a calcular toda la malla necesitaríamos las 2 primeras columnas, ya tenemos $u(:,1) = f(x)$ en $t=0$ pero, y la segunda $u(:,2)$? en $t=k$

- Falta aplicar la C.I.

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

$$\frac{u_{i,1} - u_{i,-1}}{2k} = g_i$$

$$-u_{i,-1} = 2kg_i - u_{i,1}$$ (aprox $-u_{i-1}$ que sería $u(x_i, -k)$)

al hacer $j=0$ en (1) y sustituir $u_{i,0} = f_i$
y sustituir $-u_{i,-1} = 2kg_i - u_{i,1}$
obtenemos los valores de la 2ª columna (despejando $u_{i,1}$)

$$\boxed{u_{i,1} = \alpha^2 (f_{i-1}, f_{i+1})/2 + (1-\alpha^2) f_i + kg_i}$$

Teniendo las 2 primeras columnas obtenemos las sucesivas con (1)
$u_{i,j+1} = \text{función}(u_{i,j-1}, u_{i,j})$

Método explícito (en MATLAB)

```
u(1, :) = ft j
u(:, 1) = fx j
u(nx+1, :) = rt j

% Indices
c =   2 : nx   % i
r =   3 : nx+1 % i+1
l =   1 : nx-1 % i-1

% Paso previo para columna 1
u(c, 2) = α²/2 [f(R) + f(L)] + (1-α²) f(c) + k·gx

for j = 1 : nt
    u(c, j+1) = α² (u(R,j) + u(L,j)) + 2(1-α²) u(c,j)
                - u(c, j-1)
end
```

El programa es muy específico y habría que modificarlo según el problema.

Modificación en la ecuación

$$u_{tt} = c^2 u_{xx} + 2$$ (sigue siendo hiperbólica, discriminante mayor q cero)

La fórmula cambiaría:

$$\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = c^2 \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + 2$$

despejando $u_{i,j+1}$

(1) $$u_{i,j+1} = \alpha^2 (u_{i+1,j} + u_{i-1,j}) + 2(1+\alpha^2) u_{i,j} - u_{i,j-1} + 2k^2$$

y además para el paso previo de la columna 1

$j=0$ $$u_{i,1} = \alpha^2 (u_{i+1,1} + u_{i-1,1}) + 2(1+\alpha^2) u_{i,0} - u_{i,-1} + 2k^2$$

$u_{i,-1}$ fuera del intervalo. Aplicamos c.c.

$$u_t(x,0) = g(x) \rightarrow \frac{u_{i,1} - u_{i,-1}}{2k} \simeq g_i$$

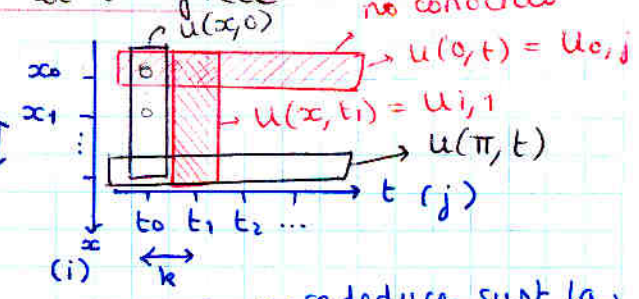
$$-u_{i,-1} \simeq 2kg_i - u_{i,1}$$

queda, para el paso previo a (1)

$$\frac{2}{2} u_{i,1} = \frac{\alpha^2}{2} (u_{i+1,1} + u_{i-1,1}) + \frac{2}{2}(1+\alpha^2) u_{i,j} + \frac{2kg_i}{2} + \frac{2k^2}{2}$$

ej. Modificación: no conozco la 1ª fila

$$u_{tt} = c^2 u_{xx} + 2 \qquad 0 \le x \le \pi$$
$$u(x,0) = 0$$
$$u_t(x,0) = 0$$
$$u_x(0,t) = t \leftarrow \text{sólo conocemos la derivada}$$
$$u(\pi,t) = 2t$$



la fórmula general será: (como ej anterior) (se deduce sust la ec por cocientes de diferencias)

(1) $$\boxed{u_{i,j+1} = \alpha^2(u_{i+1,j} + u_{i-1,j}) + 2(1-\alpha^2)u_{i,j} - u_{i,j-1} + 2k^2}$$

Para poderla aplicar necesitamos saber $u_{i,1}$ e $u_{0,j}$ (incluyendo $u_{0,1}$)

→ **Hallar $u_{i,1}$**

hacemos $j=0$ en (1)

$$u_{i,1} = \alpha^2(u_{i+1,0} + u_{i-1,0}) + 2(1-\alpha^2)u_{i,0} - u_{i,-1} + 2k^2$$

(se sale del otro)

aplicamos c.c. $u_t(x,0) = 0 \rightarrow \dfrac{u_{i,1} - u_{i,-1}}{2k} = 0$

$$-u_{i,-1} = -u_{i,1}$$

queda:

(2) $$\boxed{u_{i,1} = \frac{\alpha^2}{2}(u_{i+1,0} + u_{i-1,0}) + (1-\alpha^2)u_{i,0} + k^2}$$

→ **Hallar $u_{0,1}$**

hacemos $i=0$ en (2)

$$u_{0,1} = \frac{\alpha^2}{2}(u_{1,0} + u_{-1,0}) + (1-\alpha^2)u_{0,0} + k^2$$

(se sale)

aplicamos c.c.

$$u_x(0,t) = t \rightarrow \frac{u_{1,j} - u_{-1,j}}{2h} = t_j$$

$$u_{-1,j} = u_{1,j} - 2ht_j$$

en particular: $u_{-1,0} = u_{1,0} - 2ht_0$

queda

(3) $$\boxed{u_{0,1} = \frac{\alpha^2}{2}(u_{1,0} + u_{1,0} - 2ht_0) + (1-\alpha^2)u_{0,0} + k^2}$$

→ **Hallar $u_{0,j}$**

hacemos $i=0$ en (1)

$$u_{0,j+1} = \alpha^2(u_{1,j} + u_{-1,j}) + 2(1-\alpha^2)u_{0,j} - u_{0,j-1} + 2k^2$$

(se sale)

c.c. $u_x(0,t) = t \rightarrow$ (ver arriba) $\rightarrow + u_{-1,j} = u_{1,j} - 2ht_j$

queda:

(4) $$\boxed{u_{0,j+1} = \alpha^2(u_{1,j} + u_{1,j} - 2ht_j) + 2(1-\alpha^2)u_{0,j} - u_{0,j-1} + 2k^2}$$

Ya podemos por tanto aplicar (2), (3) y (4) como pasos previos y luego la fórmula general (1)

este ejemplo en MATLAB

```
% Rellenamos lo que ya sabemos de la malla
u(:,1) = fx;
u(nx+1,:) = rt;

% subindices
c = 2:nx      % i
p = 3:nx+1    % i+1
l = 1:nx-1    % i-1

% Pasos previos (1), (2)
u(1,2) = ...      % u0,1
u(c,2) = ...      % ui,1

% Formula general (4) con paso previo (3)   (3) dentro del for
for j = 1:nt-1
    u(1,j+1) = ...      % u0,j+1
    u(c,j+1) = ...      % ui,j+1
end
```

- **Método Implícito**

$$u_{tt} = c^2 u_{xx}$$

DEM:

$$\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = c^2 \left[\frac{(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}) + (u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1})}{2h} \right]$$

$\alpha = \frac{ck}{h}$

$u_{xx} \simeq$ media ↑ de las diferencias centrales en las filas $j+1$, $j-1$

$$u_{i,j+1} - 2u_{i,j} + u_{i,j-1} = \frac{\alpha^2}{2}\left[(u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}) + (u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1})\right]$$

agrupando a la izquierda $j+1$ y a la derecha $j-1$

$$(1+\alpha^2)u_{i,j+1} - \frac{\alpha^2}{2}(u_{i+1,j+1} + u_{i-1,j+1}) = 2u_{i,j} + \frac{\alpha^2}{2}(u_{i+1,j-1} + u_{i-1,j-1}) - (1+\alpha^2)u_{i,j-1}$$

Es un sistema de ecuaciones lineales donde obtenemos u en t_{j+1}
Se puede simplificar aun mas si hacemos

$$-\frac{\alpha^2}{4}(u_{i-1,j-1} + u_{i-1,j+1}) + \frac{(1-\alpha^2)}{2}(u_{i,j-1} + u_{i,j+1}) - \frac{\alpha^2}{4}(u_{i+1,j-1} + u_{i+1,j+1}) = u_{i,j}$$

y llamamos $w_i = u_{i,j-1} + u_{i,j+1}$

$$-\frac{\alpha^2}{4}w_{i-1} + \frac{(1-\alpha^2)}{2}w_i - \frac{\alpha^2}{4}w_{i+1} = u_{i,j} \qquad i = 1, \dots, nx-1$$

$$A w = u$$

se resuelve w y se despeja: $u_{j+1} = w - u_{j-1}$

Veamos exactamente como queda la matriz:

$$
\begin{cases}
i=1 & -\frac{\alpha^2}{4}\omega_0 + \frac{(1+\alpha^2)}{2}\omega_1 - \frac{\alpha^2}{4}\omega_2 \quad \ldots & = u_{1,j} \\
i=2 & \text{c.c.} \quad -\frac{\alpha^2}{4}\omega_1 + \frac{(1+\alpha^2)}{2}\omega_2 - \frac{\alpha^2}{4}\omega_3 & = u_{2,j} \\
 & \ddots & \\
i=nx-1 & -\frac{\alpha^2}{4}\omega_{nx-2} + \frac{(1+\alpha^2)}{2}\omega_{nx-1} - \frac{\alpha^2}{4}\omega_{nx} & = u_{nx-1,j}
\end{cases}
$$

c.c.

pasando las c.c. a los term indep ....

$$
\underbrace{\begin{pmatrix}
\frac{1+\alpha^2}{2} & -\frac{\alpha^2}{4} & & & & \\
-\frac{\alpha^2}{4} & \frac{1+\alpha^2}{2} & -\frac{\alpha^2}{4} & & & \\
 & -\frac{\alpha^2}{4} & \frac{1+\alpha^2}{2} & -\frac{\alpha^2}{4} & & \\
 & & -\frac{\alpha^2}{4} & \frac{1+\alpha^2}{2} & -\frac{\alpha^2}{4} & \\
 & & & \ddots & &
\end{pmatrix}}_{A}
\underbrace{\begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \vdots \\ \omega_{nx-1} \end{pmatrix}}_{\omega}
=
\underbrace{\begin{pmatrix} u_{1,j} + \frac{\alpha^2}{4}\omega_0 \\ u_{2,j} \\ u_{3,j} \\ \vdots \\ u_{nx-2,j} \\ u_{nx-1,j} + \frac{\alpha^2}{4}\omega_{nx} \end{pmatrix}}_{v}
$$

Como vemos habría que resolver este sistema $\forall j$
Para cada j el sist. de ecs que se obtiene tiene la misma matriz tridiagonal y de coeficientes constantes

Todo ello sugiere utilizar LU y crout.

$A\omega = v$
$A = LU$ ↓
$LU\omega = v$
$U\omega = z$ ↓
$\{ Lz = v \}$
$\{ U\omega = z \}$

Una vez se tiene ω
$u_{j+1} = \omega - u_{j-1}$
para $j=0$, habrá que usar MISMO método que en el caso explícito

Programación en MATLAB

```
diag_principal = (1+α^2)/2 * ones(1, nx-1);
diag_secund = -α^2/4 * ones(1, nx-2);
[l, u] = cLU(diag_principal, diag_secund, diag_secund);
i = 2:nx
% C.C. conocidas  u(1,:)
                  u(:,1)
                  u(nx+1,:)
% Paso previo (igual q en explícito)
    u(i,2) = ...
% Bucle
  for  j = 2:nt
       v = u(c,j)
       v(1) = v(1) + α^2/4 (l(j-1) + l(j+1))
       v(nx-1) = v(nx-1) + α^2/4 (r(j-1) + r(j+1))
                                              } vector v de term indep
       u(i, j+1) = croutlu(l, u, i, v) - u(i, j-1)
                   programa nuevo, devuelve w
  end
```

4. Modificación Método Implícito para resolver

$$\begin{cases} u_{tt} = c^2 u_{xx} + 2 \\ u(x,0) = 0 \\ u_t(x,0) = 0 \\ u_x(0,t) = t \\ u(\pi,t) = 2t \end{cases}$$

no conocemos $u(0,t) \equiv (i=0)$

Si sigues los pasos llegas a

$$-\frac{\alpha^2}{4} w_{i-1} + \frac{(1-\alpha^2)}{2} w_i - \frac{\alpha^2}{4} w_{i+1} = u_{i,j} + k^2 \cdot \frac{2}{2}$$

esta vez al hacer $i=1$ aparece w_0 que esta vez no es una C.C. conocida. Hay que utilizar w_0 como una incognita mas → esto requerirá una ecuación mas.

$i=0 \rightarrow$ $-\frac{\alpha^2}{4} w_{-1} + \frac{1-\alpha^2}{2} w_0 - \frac{\alpha^2}{4} w_1 = u_{0,1} + k^2$

$i=1 \rightarrow$ ⋮

$$w_{-1} = u_{-1,j-1} + u_{-1,j+1}$$

utilizando C.C. $u_x(0,t) = t \rightarrow \frac{u_{1,j} - u_{-1,j}}{2h} = t_j$

$$u_{-1,j} = u_{1,j} - 2h t_j$$

$$w_{-1} = u_{1,j-1} - 2h t_{j-1} + u_{1,j+1} - 2h t_{j+1}$$

$$w_{-1} = w_1 - 2h(t_{j-1} + t_{j+1})$$

se une al w_1 que ya hay — pasa al term indep

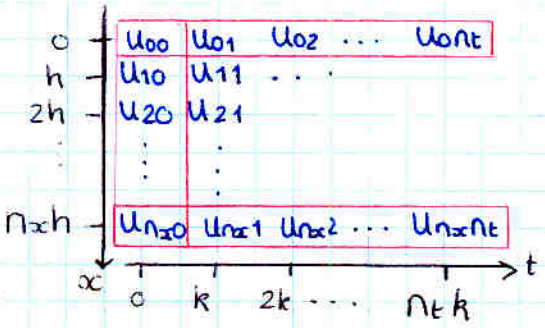
$i=0$ $\frac{1+\alpha^2}{2} w_0 - \frac{\alpha^2}{4} 2 w_1 = u_{0,j} + k^2 - \frac{\alpha^2}{4} 2h(t_{j-1} + t_{j+1})$ ← dudas

$i=1$ $-\frac{\alpha^2}{4} w_0 + \frac{(1+\alpha^2)}{2} w_1 - \frac{\alpha^2}{4} w_2 = u_{1,j} + k^2$

⋮ ⋱

- Ecuaciones PARABOLICAS

Ecuación del calor
$$\begin{cases} u_t = c\,u_{xx} & 0<x<L \\ u(x,0) = f(x) & t>0 \\ u(0,t) = T_0 \\ u(L,t) = T_L \end{cases}$$



se sustituye la ecuación por cociente de diferencias divididas según el caso.

Explicito: $t \to$ diferencias progresivas $O(k)$; $x \to$ aprox. central $O(h^2)$ } $O(k+h^2)$

Implicito: $t \to$ diferencias regresivas; x – aprox. central } $O(k+h^2)$

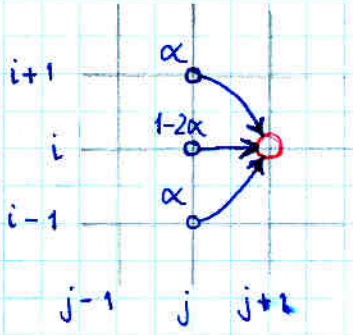
Crank-Nicholson: t_j – dif progresivas; t_{j+1} – dif regresivas; x – aprox central } se hace la media } $O(k^2+h^2)$

· Método EXPLICITO

t – dif. progresivas
x – dif centrales

$$\frac{u_{i,j+1} - u_{i,j}}{k} = c\,\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

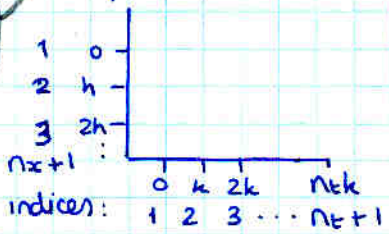
$$\boxed{u_{i,j+1} = \alpha(u_{i+1,j} + u_{i-1,j}) + (1-2\alpha)u_{i,j}}$$



Esta vez no utiliza la columna $j-1$ por lo que se puede aplicar la fórmula directamente.

en MATLAB

```
k=... h=... xi=... tj=...   %hacemos partición
u(1,:) = t0 * ones(1, nt+1)
u(nx+1,:) = tl * ones(1, nt+1)
u(:,1) = fx                 % Rellenamos u con las cond. que nos dan
```




```
c = 2 : nx      % central
l = 1 : nx-1    % left
r = 3 : nx+1    % right

for j = 1 : nt
    u(c,j+1) = α*(u(r,j) + u(l,j)) + (1-2*α)*(u(c,j));
end
```

Convergencia

$$A = \begin{pmatrix} 1-2\alpha & \alpha & 0 & 0 & \cdots \\ \alpha & 1-2\alpha & \alpha & 0 & \cdots \\ 0 & \alpha & 1-2\alpha & \alpha & \cdots \\ & & \ddots & & \end{pmatrix}$$

si el 1er valor propio inicial de A (llamado radio espectral) menor o igual que uno

$\rho(A) \leq 1 \Rightarrow$ convergencia

El programa anterior sólo sirve para el caso mas general, hagamos un ejemplo cambiando la ec. y las condiciones.

ejemplo: modificación ecuación parabólica

$$u_t = c u_{xx} + 3u_x$$
$$u(x,0) = \cos \pi x$$
$$u_x(0,t) = t+1$$
$$u_x(l,t) = t^2$$

$$\frac{u_{i,j+1} - u_{i,j}}{k} = c\,\frac{u_{i+1} - 2u_{i,j} + u_{i-1,j}}{h^2} + 3\,\frac{u_{i+1,j} - u_{i-1,j}}{2h}$$

• despejando $u_{i,j+1}$

$$\boxed{u_{i,j+1} = \left(\alpha + \tfrac{3}{2}\tfrac{k}{h}\right)u_{i+1,j} + (1-2\alpha)u_{i,j} + \left(\alpha - \tfrac{3}{2}\tfrac{k}{h}\right)u_{i-1,j}} \quad (1)$$

• pasos en los que tendremos problemas:

$\boxed{i=0} \Rightarrow$ en (1) $\quad u_{0,j+1} = \left(\alpha + \frac{3k}{2h}\right)u_{1,j} + (1-2\alpha)u_{0,j} + \left(\alpha - \frac{3k}{2h}\right)u_{-1,j}$

utilizo la C.C (aprox central: en x)

$$u_x(0,t) = t+1 \rightarrow \frac{u_{1,j} - u_{-1,j}}{2h} = t_j + 1 \xrightarrow[u_{-1,j}]{\text{despejo}} u_{-1,j} = u_{i,j} - 2h(t_j+1)$$

sustituir (y agrupar)

$$\boxed{u_{0,j+1} = (2\alpha)u_{i,j} + (1-2\alpha)u_{0,j} - \left(\alpha - \tfrac{3}{2}\tfrac{k}{h}\right)(2h(t_j+1))} \quad (2)$$

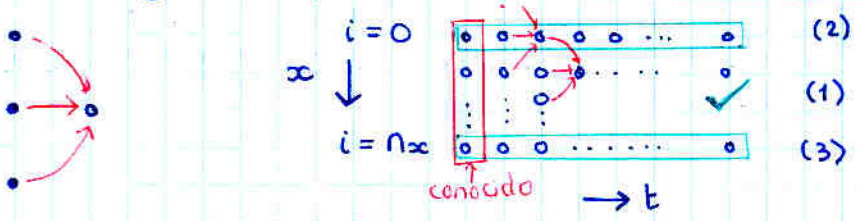
$\boxed{i = nx} \overset{(1)}{\Rightarrow} \quad u_{nx,j+1} = \left(\alpha + \frac{3}{2}\frac{k}{h}\right)u_{nx+1,j} + (1-2\alpha)u_{nx,j} + \left(\alpha - \frac{3}{2}\frac{k}{h}\right)u_{nx-1,j}$

utilizo la C.C.

$$u_x(l,t) = t^2 \rightarrow \frac{u_{nx+1,j} - u_{nx-1,j}}{2h} = t_j^2 \xrightarrow[u_{nx+1,j}]{\text{despejo}} u_{nx+1,j} = 2ht_j^2 + u_{nx-1,j}$$

$$\boxed{u_{nx,j+1} = (2\alpha)u_{nx-1,j} + (1-2\alpha)u_{nx,j} + \left(\alpha - \tfrac{3}{2}\tfrac{k}{h}\right)(2ht_j^2)} \quad (3)$$

Es lógico tener problemas en esos pasos, pues un punto necesita, para ser calculado



En el programa, dentro del for:

```
for j = 1 : nt
    (2)
    (1)
    (3)
end
```

el orden debe ser este.

• método IMPLICITO

t → diferencias regresivas
x → diferencias centrales

$$u_t = c\,u_{xx}$$

$$\frac{u_{i,j} - u_{i,j-1}}{k} = c\,\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

despejar $u_{i,j-1}$

$$\boxed{(1+2\alpha)\,u_{i,j} - \alpha(u_{i-1,j} + u_{i+1,j}) = u_{i,j-1}}$$

C.C.

$i=1$: $-\alpha u_{0,j} + (1-2\alpha)u_{1,j} - \alpha u_{2,j} = u_{1,j-1}$

$i=2$: $-\alpha u_{1,j} + (1+2\alpha)u_{2,j} - \alpha u_{3,j} = u_{2,j-1}$

$\vdots$

$i = nx-1$: $-\alpha u_{nx-2,j} + (1+2\alpha)u_{nx-1,j} - \alpha u_{nx,j} = u_{nx-1,j-1}$

C.C

queda: $A \cdot x = b$

$$\begin{pmatrix} 1-2\alpha & -\alpha & & & \\ -\alpha & 1-2\alpha & -\alpha & & \\ & -\alpha & 1-2\alpha & -\alpha & \\ & & \ddots & & \\ & & & -\alpha & 1-2\alpha \end{pmatrix} \begin{pmatrix} u_{1,j} \\ u_{2,j} \\ u_{3,j} \\ \vdots \\ u_{nx-1,j} \end{pmatrix} = \begin{pmatrix} u_{1,j-1} + \alpha u_{0,j} \\ u_{2,j-1} \\ u_{3,j-1} \\ \vdots \\ u_{nx-1,j-1} + \alpha u_{nx-1,j} \end{pmatrix}$$

sistema tridiagonal que hay que resolver $\forall j = 1 \dots n_t$
→ utilizar desc. LU

$Ax = b$
$LUx = b \longrightarrow Lz = b$ (sacar z) $\longrightarrow Ux = z$ (sacar x)

en MATLAB se programaría muy similar a las hiperbolicas, y las modificaciones se harían igual (añadir las filas que darán problema (tipico i=0) y sustituir el termino problematico segun la c.c. adecuada)

Converge para todo α

- Método de CRANK - NICHOLSON

La idea se basa en hacer la media de las formulas de diferencias progresivas y regresivas. en t_j : dif progresivas; en t_{j+1} : dif regresivas

$$u_t = c\,u_{xx}$$

en t_j: $\dfrac{u_{i,j+1} - u_{i,j}}{k} = c\,\dfrac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$

en t_{j+1}: $\dfrac{u_{i,j+1} - u_{i,j}}{k} = c\,\dfrac{u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}}{h^2}$

esto se puede demostrar que mejora el error $O(k, h^2)$

$$2(u_{i,j+1} - u_{i,j}) = \alpha\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i+1,j+1} - 2u_{i,j+1} + u_{i-1,j+1}\right]$$

despejando los términos del paso $j+1$

$$\boxed{-\alpha u_{i-1,j+1} + 2(1+\alpha)u_{i,j+1} - \alpha(u_{i+1,j+1}) = 2(1-\alpha)u_{i,j} + \alpha(u_{i+1,j} + u_{i-1,j})}$$

que podemos escribir como un sistema (pasando los términos apropiados a la columna de términos independientes → igual que caso implícito)

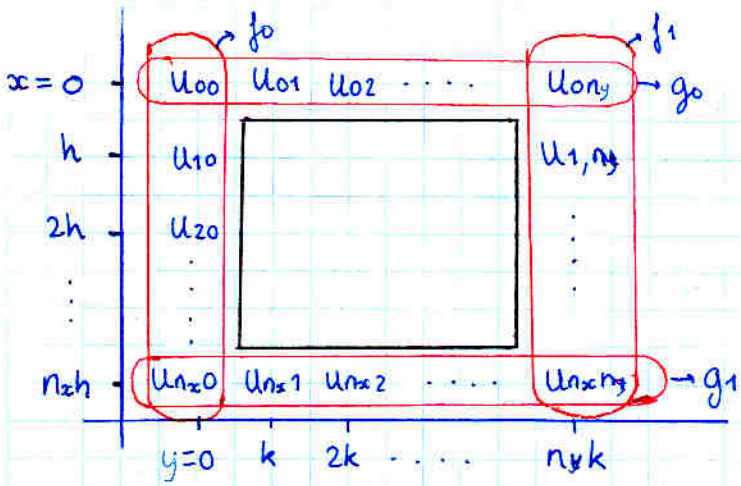
$$\begin{matrix} i=1 \\ i=2 \\ \vdots \\ \vdots \\ i=n_x \end{matrix}\begin{pmatrix} 2(1+\alpha) & -\alpha & & & \\ -\alpha & 2(1+\alpha) & -\alpha & & \\ & -\alpha & 2(1+\alpha) & -\alpha & \\ & & \ddots & \ddots & \ddots \\ & & & -\alpha & 2(1+\alpha) \end{pmatrix} \cdot \begin{pmatrix} u_{1,j+1} \\ u_{2,j+1} \\ u_{3,j+1} \\ \vdots \\ u_{n_x,j+1} \end{pmatrix} = \begin{bmatrix} 2(1-\alpha)u_{1,j} + \alpha(u_{2,j} + u_{0,j}) \\ \quad + \alpha u_{0,j+1} \\ \vdots \\ 2(1-\alpha)u_{i,j} + \alpha(u_{i+1,j} - u_{i-1,j}) \\ \vdots \\ 2(1-\alpha)u_{n_x,j} + \alpha(u_{n_x+1,j} - u_{n_x-1,j}) \\ \quad + \alpha(u_{n_x+1,j+1}) \end{bmatrix}$$

en MATLAB, como siempre

- diag-princip
- diag-secund
- hallar LU
- definir c, l, r

```
for j = 1 : nt
    b = 2(1-α) u(c,j) + α(u(r,j) + u(l,j)) j
    b(1) = b(1) + α hl(j+2) ;
    b(nx) = b(nx) + α hr(j+1) ;
    u(c,j+1) = croutlu(l, u, c, b) ;
end
```

- Ecuaciones Elípticas $\Delta = 0$



Ecuación de laplace

$u_{xx} + u_{yy} = 0 \qquad 0<x<a \quad 0<y<b$

$u(x,0) = f_0(x)$
$u(x,b) = f_1(x)$
$u(0,y) = g_0(y)$
$u(a,y) = g_1(y)$

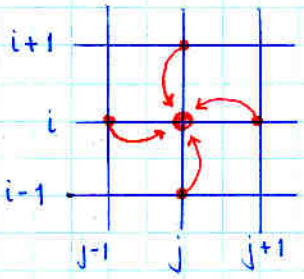
$$u_{xx} + u_{yy} = 0$$

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{k^2} = 0$$

$$\alpha = \frac{k}{h}$$

$$\alpha^2 (u_{i-1,j} + u_{i+1,j}) + u_{i,j-1} + u_{i,j+1} - 2(\alpha^2+1)u_{i,j} = 0$$

un sólo punto necesita de las 4 incognitas de su alrededor



No tenemos mas opción que resolver todas las incógnitas de un sólo paso en un único sistema muy grande y con muchos ceros.

Seria muy costoso resolver mediante Gauss asi que empleamos ...

nota: para $\alpha=1$ es simplem. la media

$\alpha = 1 \Leftrightarrow h = k$

Métodos Iterativos [convergen si diagonal principal dominante]

- **Método de Jacobi**

$$u_{i,j}^{(k)} = \left[\alpha^2\left(u_{i-1,j}^{(k-1)} + u_{i+1,j}^{(k-1)}\right) + u_{i,j-1}^{(k-1)} + u_{i,j+1}^{(k-1)}\right] \Big/ (2+2\alpha^2)$$

se entiende con un ejemplo:

$$\begin{cases} 3x_1 - x_2 + x_3 = 5 \\ x_1 + 5x_2 - 3x_3 = -1 \\ 2x_1 + 2x_2 + 7x_3 = 2 \end{cases} \rightarrow \begin{cases} x_1 = \frac{1}{3}(5 - x_3 - x_2) \\ x_2 = \frac{1}{5}(-1 + 3x_3 - x_1) \\ x_3 = \frac{1}{7}(2 - 2x_2 - 2x_1) \end{cases}$$

Estimación inicial $x^{(0)} = (0,0,0)$

paso 1 $x^{(1)} = (x_1(x_2^{(0)}, x_3^{(0)}), x_2(x_1^{(0)}, x_3^{(0)}), x_3(x_1^{(0)}, x_2^{(0)}))$
$= (5/3, -1/5, 2/7)$

$x^{(k)} = \dots$

hasta que $\max \left| u_{i,j}^{(k)} - u_{i,j}^{(k-1)} \right| < \varepsilon$

- **Método de Gauss-Seidel**

$$u_{i,j}^{(k)} = \left[\alpha^2\left(u_{i-1,j}^{(k)} + u_{i+1,j}^{(k-1)}\right) + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k-1)}\right] \Big/ (2+2\alpha^2)$$

Modificación del anterior: las incógnitas que vamos hallando en un paso las vamos utilizando en ese mismo paso

$$x^{(k)} = (x_1^{(k)}(x_2^{(k-1)}, x_3^{(k-1)}), x_2^{(k)}(x_1^{(k)}, x_3^{(k-1)}), x_3^{(k)}(x_1^{(k)}, x_2^{(k)}))$$

- **Método de sobrerrelajación**: la incognita que usamos es una ponderación entre la de este paso y la anterior

$$u_{i,j}^{(k)} = (1-\omega)u_{i,j}^{(k-1)} + \omega \hat{u}_{i,j}^{(k)}$$

$\omega = 0 \Rightarrow$ Jacobi $\qquad \omega = 1 \Rightarrow$ Gauss-Seidel

Método de Jacobi en MATLAB:

Entrada: a, b, f_0, f_1, g_0, g_1

Proceso:

$nx = length(f_0) - 1;$
$ny = length(g_0) - 1;$
$h = a/nx;$
$k = b/nx;$
$u_0 = \left[\frac{\sum f_0}{nx+1} + \frac{\sum f_1}{nx+1} + \frac{\sum g_0}{ny+1} + \frac{\sum g_1}{ny+1}\right]\frac{1}{4}$ Estimación inicial
$u = u_0 * ones(nx+1, ny+1)$

$u(1,:) = g_0;$
$u(nx+1,:) = g_1;$ condiciones de contorno
$u(:,1) = f_0;$
$u(:,ny+1) = f_1;$

Inicializamos { incr, iter, $v = u$

while incr > tol & k < maxiter
  for i = 2 : nx
    for j = 2 : ny

$$v(i,j) = \frac{[\alpha^2(u(i-1,j)+u(i+1,j)) + u(i,j-1) + u(i,j+1)]}{2 + 2\alpha^2}$$

    end
  end

  iter = iter + 1;
  incr = max(max(u - v))
  u = v;
end

---

para Gauss-Seidel:
todo igual, pero:

for
 for
  v(i,j) = ... [misma expr de antes]
  incr = max(incr, abs(u(i,j) - v(i,j)))
  u(i,j) = v(i,j)

otra posibilidad es en
$v(i,j) = (\alpha^2(v(i-1,j) + u(i+1,j)) + v(i,j-1) + u(i,j+1)] / (2 + 2\alpha^2)$
cambiar la u por v donde se usen incognitas calculadas y ya no haria falta ningun otro cambio

ej: Modificación : no conocemos todo el contorno

$$u_{xx} + u_{yy} = 0 \qquad \begin{matrix} 0 \le x \le 2 \\ 0 \le y \le 2 \end{matrix}$$

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

$u_x(2,y) = u(2,y)$

ec. Gauss Seidel

$$u_{i,j}^{(k)} = \frac{1}{4}\left[u_{i-1,j}^{(k)} + u_{i+1,j}^{(k-1)} + u_{i,j-1}^{(k)} + u_{i,j+1}^{(k+1)}\right]$$

tendremos problemas en

$i = n_x$: $$u_{n_x,j}^{(k)} = \frac{1}{4}\left[u_{n-1,j}^{(k)} + \underbrace{u_{n+1,j}^{(k+1)}}_{\text{no lo tenemos}} + u_{n,j-1}^{(k)} + u_{i,j+1}^{(k-1)}\right]$$

aplicamos C.C.

$$u_x(2,y) = u(2,y) \rightarrow u_{x_{n,j}} = u_{n,j} \rightarrow \frac{u_{n+1,j} - u_{n-1,j}}{2h} = u_{n,j}$$

$$u_{n+1,j} = 2h\,u_{n,j} + u_{n-1,j}$$

sustituyendo:

$$u_{n,j}^{(k)} = \frac{1}{4}\left[2u_{n-1,j}^{(k)} + 2h\,u_{n,j}^{(k)} + u_{n,j-1}^{(k)} + u_{n,j+1}^{(k-1)}\right]$$

en MATLAB será $u(n+1, j)$

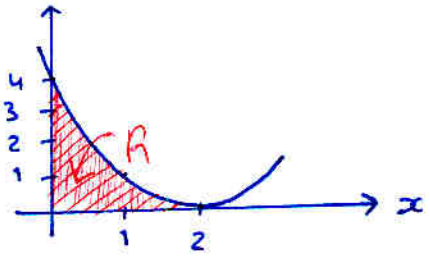
$$u_{n,j} = \frac{1}{4}\frac{1}{1+h/2}\left(2u_{n-1,j}^{(k)} + u_{n,j-1}^{(k)} + u_{n,j+1}^{(k)}\right).$$

TEMA 2. INTEGRACION NUMERICA

Problema 2. Septiembre 2003

$$I = \iint_R \sqrt{x+y+e^x}\, dA$$

siendo R la región entre $y = (x-2)^2$ y los ejes coorderados



- Simpson doble variable

$c(x) = 0$
$d(x) = (x-2)^2$
$f(x) = \sqrt{x+y+e^x}$

para 8 subintervalos pasarle $m = n = 4$

- Gauss legendre doble variable : igual
- Montecarlo. Primero calculo area

$$\int_0^2 \int_0^{(x-2)^2} 1 \, dy\, dx = \int_0^2 [(x-2)^2]\, dx$$
$$= \int_0^2 x^2 - 4x + 4\, dx$$
$$= \left[\frac{x^3}{3} - 2x^2 + 4x \right]_0^2$$
$$= \frac{8}{3} - 8 + 8 = 8/3$$

Problema 3.

Construir la formula de cuadratura

$$\int_0^1 f(x)\, dx = \sum_{j=0}^{10} w_j\, f\left(\frac{j}{10}\right)$$

siendo w_j los pesos que hacer la fórmula exacta para polinomios de grado ≤ 10

$$= w_0 f(0) + w_1 f(0'1) + w_2 f(0'2) + w_3 f(0'3) + w_4 f(0'4) + w_5 f(0'5) + w_6 f(0'6) + w_7 f(0'7) + w_8 f(0'8) + w_9 f(0'9) + w_{10} f(1)$$

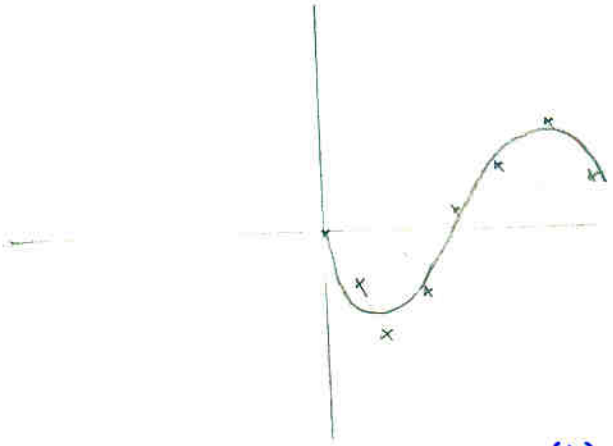
debe cumplirse para

$\int_0^1 dx = 1 = w_0 \cdot 1 + w_1 \cdot 1 + w_2 \cdot 1 + \dots$
$\int_0^1 x\, dx = \frac{1}{2} = w_0 \cdot 0 + w_1 \cdot 0'1 + w_2 0'2 + \dots$
$\int_0^1 x^2 dx = \frac{1}{3} = w_0 \cdot 0 + w_1 \cdot 0'1^2 + w_2 0'2^2 + \dots$
$\int_0^1 x^3 dx = \frac{1}{4} = w_0 \cdot 0 + w_1 \cdot 0'1^3 + w_2 \cdot 0'2^3 + \dots$
$\vdots$

El sistema es

$$\begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 0 & 0'1 & 0'2 & 0'3 & \dots & 1 \\ 0 & 0'1^2 & 0'2^2 & 0'3^2 & \dots & 1^2 \\ \vdots & \vdots & \vdots & \vdots & & \\ 0 & 0'1^{10} & 0'2^{10} & 0'3^{10} & \dots & 1^{10} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{10} \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/3 \\ \vdots \\ 1/11 \end{pmatrix}$$

Problema minimos cuadrados inventado



$t = 0'1 \quad 0'2 \quad 0'3 \quad 0'4 \quad 0'5 \quad 0'6 \quad 0'7$

$y = -0'1 \quad -0'3 \quad -0'2 \quad 0'05 \quad 0'2 \quad 0'3 \quad 0'21$

$2\pi \rightarrow 0'6$

$\frac{0'6}{2\pi} \quad x$

aproximar con: $y(t) = A\,\text{sen}(Bt) + Ct$

recuerda sen = sin

minimizar: $E = \sum (y(t) - A\,\text{sen}(Bt) - Ct)^2$

$$\Rightarrow \begin{cases} \frac{\partial E}{\partial A} = 2\sum (\underbrace{-\text{sen}(Bt)}_{dA})(y - A\,\text{sen}(Bt) - Ct) = 0 \\ \frac{\partial E}{\partial B} = 2\sum (\underbrace{-At\cos(Bt)}_{dB})(\underbrace{y - A\,\text{sen}(Bt) - Ct}_{F}) = 0 \\ \frac{\partial E}{\partial C} = 2\sum (\underbrace{-t}_{dC})(\underbrace{y - A\,\text{sen}(Bt) - Ct}_{F}) = 0 \end{cases}$$

$$J = \begin{pmatrix} 2\Sigma(dA * dA) & 2\Sigma((-t\cos Bt) * F + dA * dB) & 2\Sigma(dA * dC) \\ 2\Sigma((-t\cos(Bt)) * F + dB * dA) & 2\Sigma((At^2\,\text{sen}(Bt)) * F + dB * dB) & 2\Sigma(dB * dC) \\ 2\Sigma(dC * dA) & 2\Sigma(dC * dB) & 2\Sigma(dC * dC) \end{pmatrix}$$

estimación inicial: $[A, B, C] = [0'25,\ 2\pi/0'9,\ 0'1]$

tolerancia: $1 \cdot 10^{-8}$

resultado $A = -0'26526$ $\quad k = 4$

$B = 7'86956$

$C = 0'04333$