Libro: Métodos numéricos para ingenieros. Steven Chapra

Índice

08/08

Localización de raíces

Dada una función f(x), si x_0 satisface que f(x_0) = 0 entonces x_0 se llama raíz o cero de f(x).

Los métodos numéricos se diseñan para cálculos de raices demasiado complejos (donde el calculo analitico no es posible)
Para encontrar raíces hay varios métodos:

1- Método gráfico:

Graficamos f(x) y vemos donde dicha gráfica corta el eje x (siempre arrancamos usando este metodo para tener una noción inicial de donde se encuentran las raíces)
#comandos para graficar en Ubuntu:
1- sudo apt install gnuplot-qt
2- gnuplot
3- definir f(x) (ln es log, logaritmo decimal es log10)
4- plot f(x)
Casos posibles:
. f(x) no tiene raíces ⇒ rta: f(x) no tiene raíces
. f(x) tiene múltiples raíces ⇒ debo decidir cuales quiero encontrar. Para cada una de ellas determino un intervalo que solo encierre a ESA raíz
. f(x) raíz única ⇒ elijo cualquier intervalo (que contenga la raíz)

2- Métodos cerrados:

Parten de un intervalo [a,b] el cual encierra ÚNICAMENTE una raíz tal que f(a) . f(b) < 0 (ya que f(x) en algún punto entre a y b corta al eje x) siempre y cuando la función sea continua. Si f(a) . f(b) < 0 me aseguro que hay al menos una raíz, gráficamente comprobamos que haya una sola (en el caso de haber más, corresponderá un intervalo para cada una de ellas).

2.A- Método de Bisección:

Se basa en realizar iteraciones consistentes en tomar un valor

$$c = \frac{a+b}{2}$$

y evaluar si f(a) . f(c) < 0 o bien f(b) . f(c) < 0 para así “arrastrar” el extremo en el que la desigualdad sea falsa (es decir, >0) al valor de c (es decir, a = c o b = c). Estas iteraciones se realizan hasta un *Criterio de Corte.*
Criterio de Corte y Estimación de Error:
supongamos que queremos la raíz con cierta cantidad de decimales correctos (sea m ese número). El error es e < 10^{-m}

¿Cómo calculamos e?

a) Si tengo la raíz exacta x_0 ⇒ e = |c-x_0| (*error absoluto exacto*)

b) Otro criterio podría ser aproximar la raíz con cierto error porcentual

$\varepsilon = \frac{|c-x_0|}{|x_0|} * 100$ (*error porcentual exacto*)

Sin embargo, para esto deberíamos conocer x_0 (y de conocerlo, conviene hacer a)

⇒ Estimación de error

a) en cada iteración, la raíz siempre se encuentra entre a y b (ya que lo redefinimos, cambiando el valor de alguno por el de c) entonces podemos decir que el error es aproximadamente de $\frac{b-a}{2}$ (ya que siempre dividimos el intervalo por 2) (*error absoluto aproximado*)

b) (para otros métodos distintos de bisección, no es tan claro) la raíz siempre está más cerca del nuevo valor de c que del valor de c en la iteración anterior

⇒ $e_{aprox} = |cn - cv|$ (*error absoluto aproximado*)

donde cn = c de la iteración actual

cv = c de la iteración anterior (si es la primer iteración, se inicializa en a)

(requiere guardar el valor de c en cada iteración)

$\varepsilon = \frac{|cn-cv|}{|cn|} * 100$ (*error aproximado porcentual*)

Ejemplo: f(x) = 2x - 4

Sea [1,5 ; 4]

f(1,5) = -1

f(4) = 4

f(a) . f(b) < 0 ⇒ Hay al menos una raíz (2)

<u>1er iteración</u>

c = (1,5 + 4) / 2 = 2,75

e_{exacto} = |2,75 - 2| = 0,75

e_{aprox} = |2,75 - 1,5| = 1,25

$\varepsilon = \frac{1,25}{2,75} * 100$ = 45%

<u>2da iteración</u>

f(a) . f(c) < 0 ⇒ raíz entre a y c

b = c

c = (1,5 + 2,75) / 2 = 2,125

e_{exacto} = |2,125 - 2| = 0,125

e_{aprox} = |2,125 - 2,75| = 0,625

$\varepsilon = \frac{0,625}{2,125} * 100$ = 29%

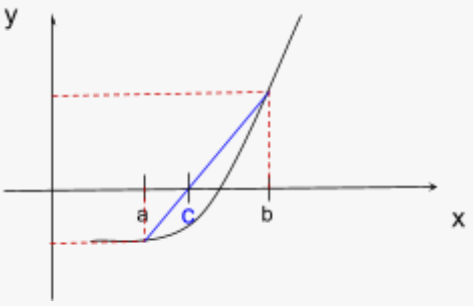
Y así continuariamos la cantidad de veces necesarias hasta satisfacer la tolerancia o error mínimo definida (es decir, la cantidad de cifras decimales de error que estamos dispuestos a considerar)

Algoritmo:
1- Definir f(x)
2- Determinar el intervalo que abarca la raíz de manera gráfica
3- Ingresar a y b al código (por teclado o hardcodeado)(usar double para todas las variables)
4- Definir el criterio de error y la tolerancia (error absoluto o porcentual)
5- Aplicar bisección iterativamente hasta satisfacer la tolerancia
6- Imprimir el valor de la raíz y el error

17/08

2.B- Método de Falsa Posición (regula-falsi):

Al igual que el metodo de biseccion, parte de la suposición que conocemos a y b tales que $x \in [a, b]$.
El procedimiento en este método es el siguiente: Tomamos los puntos (a, f(a)) y (b,f(b)), luego trazamos la recta que une ambos puntos y tomamos como aproximación de la raíz al punto de intersección de esta recta con el eje X (c en el gráfico)
Luego chequeamos al igual que en bisección si:
f(a) . f(c) < 0 ⇒ a = c
f(a) . f(c) > 0 ⇒ b = c
f(a) . f(c) = 0 ⇒ c es la raíz



Por semejanza de triángulos, podemos decir que el triángulo acf(a) y el triángulo bcf(b) son semejantes, por lo que:

$$\frac{c-a}{c-b} = \frac{f(a)}{f(b)} \Rightarrow f(b)\,(c-a) = f(a)\,(c-b)$$

$$f(b)\,.\,c - f(b)\,.\,a = f(a)\,.\,c - f(a)\,.\,b$$

$$f(b)\,.\,c - f(a)\,.\,c = f(b)\,.\,a - f(a)\,.\,b$$

$$c\,(f(b) - f(a)) = f(b)\,.\,a - f(a)\,.\,b$$

$$c = \frac{f(b)\,.\,a - f(a)\,.\,b}{f(b) - f(a)}$$

Obteniendo así una nueva forma de calcular c distinta a la forma vista en bisección. El resto del procedimiento es igual que el método de bisección, por lo que la única diferencia entre ambos es la forma de calcular c.

3- Métodos Abiertos:

A diferencia de los métodos cerrados, los métodos abiertos solo requieren UN valor de partida o inicial (o bien dos, pero no necesariamente deben encerrar a la raíz)
Los métodos abiertos *no siempre convergen* pero cuando lo hacen, lo hacen mucho más rápido que los métodos cerrados.

3.A- Método de Iteración de Punto Fijo:

Partamos de la idea de que queremos obtener un x tal que f(x) = 0. Esta última expresión *puede ser reordenada* de tal forma que obtengamos una ecuación del tipo x = g(x).

Por ejemplo:

$f(x) = x - x^2 + tg(x) = 0 \Rightarrow x = x^2 - tg(x)$

donde $g(x) = x^2 - tg(x)$

Sin embargo, no siempre es tan fácil despejar x, por lo que podemos hacer lo siguiente

$f(x) = e^{sen(x)} + ln(x) = 0$

Una forma de obtener una expresión x = g(x) sería sumar x en ambos miembros, obteniendo:

$x + e^{sen(x)} + ln(x) = x$

donde $g(x) = x + e^{sen(x)} + ln(x)$

Es decir, cualquiera sea la forma de f(x), siempre podemos reescribirla de forma tal que llevemos la expresión f(x) = 0 a una forma x = g(x)

Definición: **Punto Fijo de g(x)**

Sea g una función y x_p un valor tal que satisface

$x_p = g(x_p)$

Entonces decimos que x_p es un "punto fijo de g"

Por lo tanto, encontrar la raíz de f(x) es equivalente a encontrar el punto fijo de g(x).

Para esto, proponemos un método iterativo en el cual:

$x_{i+1} = g(x_i)$

y solo necesitamos un punto de partida x_0. Dicho punto se define de manera arbitraria.

Realicemos un ejemplo para ver cómo funciona el método.
Ejemplo:

$f(x) = e^{-x} - x$

$$f(x) = 0 \Rightarrow e^{-x} - x = 0 \Rightarrow x = e^{-x} \Rightarrow g(x) = e^{-x}$$

$x_{i+1} = e^{-x_i}$ tomemos $x_0 = 0$

$x_1 = 1$

$x_2 = e^{-1} = 0,367879$

$x_3 = e^{-0,367879} = -0,6922009$

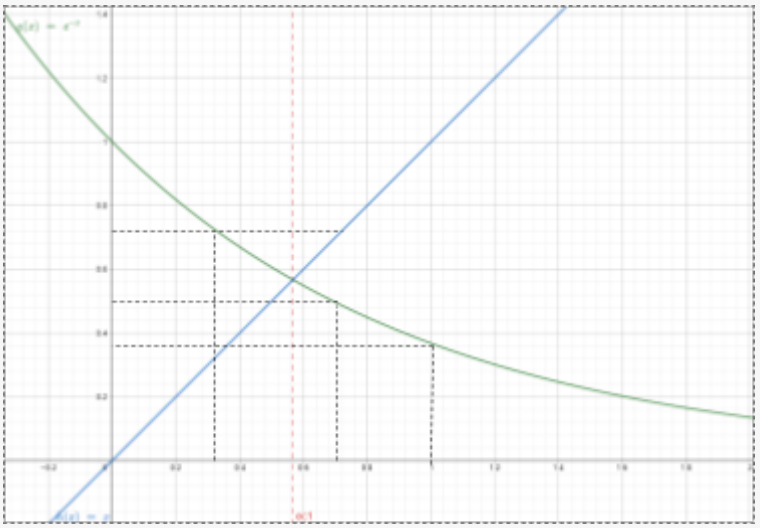
A su vez

$f(1) = e^{-1} - 1 = -0,36212$

$f(2) = e^{-2} - 2 = 0,324332$

$f(3) = e^{-3} - 3 = -0,191727$

Si marcamos $x_{i+1} = g(x_i)$ vemos como nos vamos acercando al punto fijo x = g(x) (líneas punteadas negras).



Error y Convergencia

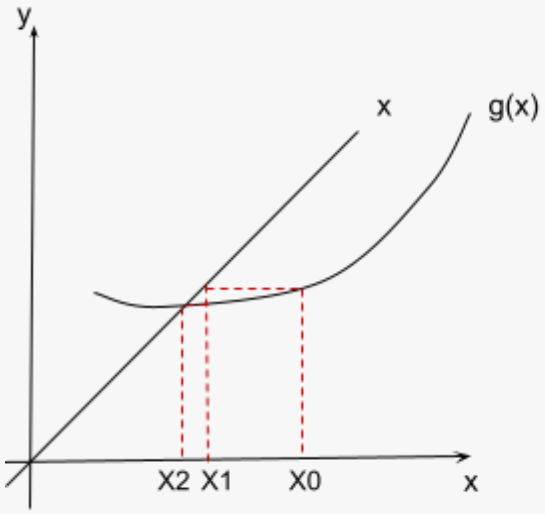
Al igual que con los métodos cerrados

$e_{aprox} = |x_{i+1} - x_i|$ (*error absoluto aproximado*)

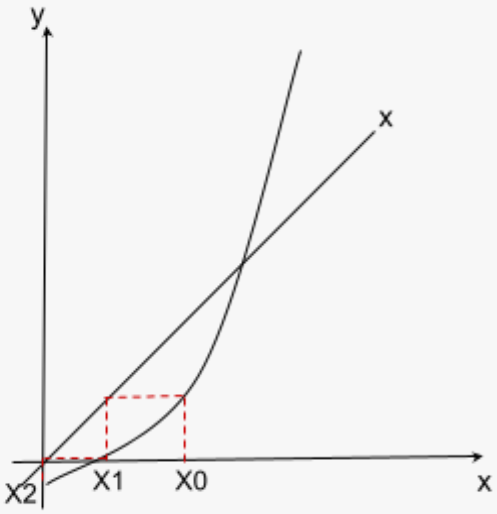
$\varepsilon = \frac{|x_{i+1} - x_i|}{|x_{i+1}|} * 100$ (*error aproximado porcentual*)

y definiremos una tolerancia como criterio de corte.

El método converge si y sólo si $|g'(x)| < 1$ en la región estudiada. Esto es fácil de ver gráficamente.



|g'(x)| < 1 ⇒ cumple
sta "acostada" cerca del punto fijo, los Xi convergen a Xp)



|g'(x)| > 1 ⇒ no cumple
(esta empinada cerca del punto fijo, los Xi se alejan cada vez más de Xp)

Para ver esto analiticamente, consideremos que x_p es el punto fijo real ⇒ $x_p = g(x_p)$.

Por nuestro método, $x_{i+1} = g(x_i)$ y si restamos ambas expresiones obtenemos que:

$$x_p - x_{i+1} = g(x_p) - g(x_i) \quad 1)$$

Y por el teorema del valor medio, existe un $c \in [x_p, x_i]$ tal que:

$$g'(c) = \frac{g(x_i) - g(x_p)}{x_i - x_p} \Rightarrow g'(c)\,(x_i - x_p) = g(x_i) - g(x_p)$$

$$-g'(c)\,(x_i - x_p) = g(x_p) - g(x_i)$$

$$g'(c)\,(x_p - x_i) = g(x_p) - g(x_i)$$

E igualando a 1)

$$g'(c)\,(x_p - x_i) = x_p - x_{i+1}$$

$$e_i \cdot |g'(c)| = e_{i+1}$$

Por lo que:

- Si $|g'(c)| < 1 \Rightarrow e_{i+1} < e_i \Rightarrow$ Converge
- Si $|g'(c)| > 1 \Rightarrow e_{i+1} > e_i \Rightarrow$ Diverge

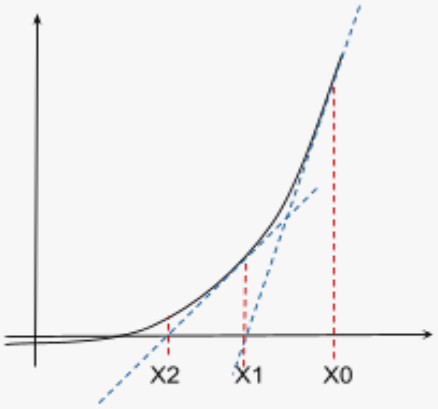
Nota: Es conveniente que el cálculo se realice despejando x (siempre que sea posible) ya que generalmente el sumar x de ambos lados nos da un resultado divergente.

Algoritmo:

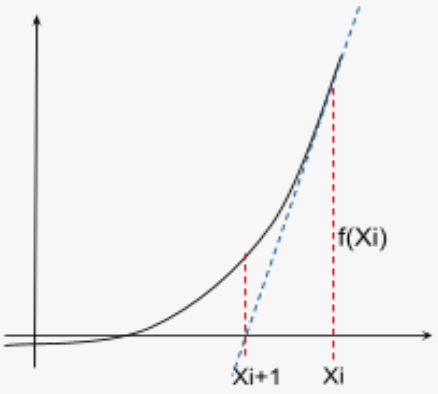
1- Definir f(x)
2- Construir g(x) (si es posible, despejando x)
3- Inicializar x_{viejo} (x_0)
4- Definir la tolerancia
5- Inicializar el error (mayor que la tolerancia) y un contador de iteraciones
6- Realizar el bucle do while con la condición que la tolerancia sea menor que el error. Dentro, aumentamos la iteración (iteracion++), verificamos que la g'(x_{viejo}) < 1 (esto lo hacemos mediante la definición de derivada, tomando el límite con un h lo suficientemente pequeño (0,01) y realizando $g'(x_{viejo}) = \frac{g(x_{viejo}+0{,}01)-g(x_{viejo})}{0{,}01}$, si dicho valor es mayor o igual a 1, entonces arrojamos un warning informando que el resultado es divergente), si dicha condición se satisface correctamente, entonces calculamos el x_{nuevo} como $x_{nuevo} = g(x_{viejo})$, el error y realizamos el cambio de x_{viejo} = x_{nuevo}. Finalmente, mostraremos los valores de la raíz, el error y la cantidad de iteraciones por pantalla.

3.B- Método de Newton-Raphson:

Si en la iteración i, el valor aproximado de la raíz es x_i, entonces podemos trazar la recta tangente a f(x) en x_i y ver en donde corta al eje X.



Vemos como nos vamos acercando a la raíz mediante las distintas rectas tangentes



También podemos observar que Xi Xi+1 f(Xi) forman un triángulo

$$tg(\alpha) = \frac{f(x_i)}{x_i - x_{i+1}} = f'(x_i)$$

$$f(x_i) = f'(x_i) \cdot (x_i - x_{i+1})$$

$$f(x_i) = f'(x_i) \cdot x_i - f'(x_i) \cdot x_{i+1}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Ahora bien, debemos tener en cuenta las siguientes advertencias:

1°) Tope de iteraciones: Podría suceder que $f'(x_i)$ fuese un número muy grande, por lo que $x_{i+1} \simeq x_i$ llevándonos a hacer muchísimas iteraciones para conseguir el resultado.

2°) Al finalizar las iteraciones, verificar que $f(x_{nuevo}) \simeq 0$

3°) Verificar que $f'(x_i)$ no sea demasiado pequeña (cercana a 0)

Algoritmo:

1- Definir f(x)

2- Construir f'(x)

3- Inicializar x_{viejo} (x_0)

4- Definir la tolerancia

5- Inicializar el error (mayor que la tolerancia) y un contador de iteraciones

6- Realizar el bucle do while con la condición que la tolerancia sea menor que el error o bien las iteraciones hayan llegado al máximo permitido. Dentro aumentaremos la iteración, evaluaremos las condiciones planteadas para f'(x_{viejo}) nombradas anteriormente, y de verificarse, calcularemos el valor de x_{nuevo} y el error, luego realizaremos el pasaje de x_{viejo} = x_{nuevo}.

Finalmente, mostraremos los valores de la raíz, el error y la cantidad de iteraciones por pantalla.

3.C- Método de la Secante:

Es básicamente Newton-Raphson donde aproximamos $f'(x_i)$

$$f'(x_i) \simeq \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

$$x_{i+1} = x_i - \frac{f(x_i) \cdot (x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

Algoritmo:

1- Definir f(x)

2- Inicializar x_{vv}, x_v, error, iteraciones

4- Definir la tolerancia

5- Realizar el bucle do while con la condición que la tolerancia sea menor que el error o bien las iteraciones hayan llegado al máximo permitido. Dentro aumentaremos la iteración, realizaremos el cálculo del x_{nuevo} mediante la fórmula propuesta, calcularemos el error y luego realizamos los pasajes x_{vv} = x_v y x_{nuevo} = x_v.

Finalmente, mostraremos los valores de la raíz, el error y la cantidad de iteraciones por pantalla.

24/08

Sistema de Ecuaciones Algebraicas Lineales

Hasta ahora, dada una f(x) encontramos un x_r tal que f(x_r) = 0, es decir, encontrabamos la raíz para una sola ecuación.
Ahora, tenemos que encontrar un conjunto de valores x_1, x_2,.....,x_n que de forma simultánea cumplan lo siguiente:

$$f_1(x_1, x_2,..., x_n) = 0$$
$$f_2(x_1, x_2,..., x_n) = 0$$
$$\vdots$$
$$f_n(x_1, x_2,..., x_n) = 0$$

Esto es lo que conocemos como un Sistema de Ecuaciones de n ecuaciones y n incógnitas (los métodos que vamos a estudiar solo son válidos para sistemas nxn)
En forma gráfica:

$$a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n$$

Por lo que podríamos armar la matriz $A\bar{x} = \bar{b}$.

Para resolver:
1. Armamos la matriz ampliada
2. Reducimos por Gauss o Gauss-Jordan
3. Despejamos (en caso de ser necesario) para obtener los valores de X

En ingeniería, siempre tenemos solución única, por lo que $det(A) \neq 0$. El programa debe chequear que dicha desigualdad se cumpla, es decir, multiplicar la diagonal principal de la matriz reducida con Gauss ($a_{11}.a_{22}. \dots . a_{nn}$) siendo esto válido ya que las operaciones de reducción son escalares distintos de 0 para el determinante final.

Método de Eliminación Gaussiana

Como hemos visto anteriormente, la reducción de matrices consiste en la suma de múltiplos de una fila con otra, de tal forma que la diagonal principal este formada por unos. Para realizar esto en nuestro código, deberemos realizar lo siguiente:

I. Sumar o restar k veces otra fila ⇒ Consideremos que tenemos un uno principal en la fila i, de manera general realizaremos esta operación como lo siguiente:

$$for\ j > i\ ,\ f_i . (-\ a_{ji}) + f_j$$

II. Multiplicar una fila por un múltiplo ⇒ Para obtener un uno principal, debemos multiplicar la fila por el inverso del primer elemento de esta, esto es:

$$f_i \Rightarrow f_i(\frac{1}{a_{ii}})$$

Podemos combinar I y II para obtener lo siguiente:

$$f_i . (\frac{a_{ji}}{a_{ii}}) + f_j$$

Triangulación:

do i = 1,....,n-1 ⇒ Recorre las filas

do j = i+1,n ⇒ Recorre los B

$$factor = \frac{-a_{ji}}{a_{ii}}$$

do k = i,...,n ⇒ Recorre las columnas

$$a_{jk} = a_{ik} . factor + a_{jk}$$

end do (índice k)

$$b_j = b_i . factor + b_j$$

end do (índice j)

end do (índice i)

$a_{11} . a_{22} . \ldots . a_{nn} \neq 0$ ⇒ Verificación del determinante (también debemos mostrar la matriz en este punto)

Retrosustitución

$$x_3 = \frac{b_3}{a_{33}}$$

$$x_2 = \frac{b_2 - a_{23} x_3}{a_{22}}$$

$$x_1 = \frac{b_1 - a_{12} x_2 - a_{13} x_3}{a_{11}}$$

y en forma general

$$x_i = \frac{b_i - \sum a_{ij} x_i}{a_{ii}}$$

Algoritmo

$$x_n = \frac{b_n}{a_{nn}}$$

do i = n-1, …., 1; (decrementa)
    suma = 0
    do j = i+1,n
        suma = suma + $a_{ij} x_j$
    end do
    $x_i = \frac{b_i - suma}{a_{ii}}$
end do
Mostrar x_i

Posibles riesgos:

➢ Algún a_{ii} demasiado pequeño:

$0,0001.\, x_1 + x_2 + x_3 = 1$
$2x_1 + 2x_2 - x_3 = 3$
$-5x_1 + x_3 = 5$

Lo que hacemos es cambiar las filas con *pivoteo parcial*

Pivoteo parcial: Mecanismo para evitar números muy pequeños. Consiste en, antes de dividir la fila por a_{ii}, verificar que a_{ii} es menor que una tolerancia a elección (10^{-3} como elección). Si pasa, buscamos el elemento más grande disponible e intercambiamos las filas.

Algoritmo

Para cada fila en triangulación (dentro del bucle i)
p=i
if($|a_{ii}| < \varepsilon$)
    do l = i+1, n
        if($|a_{li}| > |a_{ii}|$)
            p = l
    end do

do m = l, n

$swap = a_{pm}$

$b_{pm} = b_{im}$

$b_{im} = swap$

end do

$swap = b_p$

$b_p = b_i$

$b_i = swap$

end if

31/08

Métodos Iterativos para Sistemas de Ecuaciones Lineales

Método de Jacobi

Queremos resolver $A\bar{x} = \bar{b}$, es decir

$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$

$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$

.

.

.

$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n$

Podríamos despejar Xi de cada ecuación, es decir:

$x_1 = (b_1 - a_{12}x_2 - \ldots\ldots - a_{1n}x_n)/a_{11} \Rightarrow x_1 = (b_1 - \sum_{j!=n}(a_{1j}x_j))/a_{11}$

$x_2 = (b_2 - a_{21}x_1 - \ldots\ldots - a_{2n}x_n)/a_{22} \Rightarrow x_2 = (b_1 - \sum_{j!=2}(a_{2j}x_j))/a_{22}$

$x_n = (n_1 - a_{n1}x_1 - \ldots\ldots - a_{nn-1}x_{n-1})/a_{nn} \Rightarrow x_n = (b_1 - \sum_{j!=n}(a_{nj}x_j))/a_{nn}$

Y en general

$X_i = (b_i - \sum_{j \neq i} a_{ij}X_j)/a_{ii}$ con j != n

Para resolver, vamos a hacer un $X_n[\,]$ y un $X_v[\,]$ como en punto fijo, donde $X_v[\,]$ se va a inicializar en 0 y vamos a iterar para obtener los valores de $X_n[\,]$.

Es decir, Jacobi consiste en un método iterativo donde

$$X_i^{nuevo} = \frac{1}{a_{ii}}(b_1 - \sum_{j\neq i}^{n} a_{ij} X_j^{viejo})$$

Sin embargo, este método tiene el problema de que, cuando alguno de los valores de la diagonal principal es 0 o cercano a dicho valor, este diverge. Por lo que, antes de operar, debemos asegurarnos lo siguiente:

1) (Condición suficiente pero no necesaria, puede pasar que esta condición no se cumpla pero el método converja) Asegurarnos que ningún aii es cercano a 0. Para esto podríamos verificar que la matriz sea diagonalmente dominante. Esto es, que el valor absoluto de la diagonal principal de cada fila debe ser mayor que el valor absoluto de la suma de los demás elementos de dicha fila. Es decir:

$$|a_{ii}| > \sum_{j\neq i}^{n} |a_{ij}|$$ i = 1, 2, … , n

Ej:

1 2 2
0 1 5
3 8 2

No es diagonalmente dominante 1<4, 1<5, 2<11

5 1 2
-2 -6 1
0 -1 3

Es diagonalmente dominante 5>3, 6>3, 3>1

Si inicialmente la matriz no es diagonalmente dominante, pero mediante un cambio de filas puedo hacerla diagonalmente dominante, es válido.

#Nota: Podemos verificar esto, y si no se cumple dar un warning, pero no detener la ejecución y asignar un límite de iteraciones al ciclo

2) Inicializar X_{viejo} (usualmente se toma $x_{inicial}^{viejo}$ = {0,0,0….,0})

3) Establecer un criterio de corte en base a una tolerancia de error (A diferencia de Gauss, que daba el valor exacto del Xbarra, acá tenemos un cierto error)

Definimos el error:

$$e = ||X_{nuevo} - X_{viejo}|| \Rightarrow Error\ absoluto = \sqrt{\sum_{i=1} n(Xi^{nuevo} - Xi^{viejo})^2}$$

$$\varepsilon = \frac{||X_{nuevo} - X_{viejo}||}{||X_{nuevo}||} * 100 \Rightarrow Error\ porcentual$$

Algoritmo

- Definir la matriz A y el vector B (O bien la matriz ampliada)
- Verificar que la matriz sea diagonalmente dominante (si lo es, cheto, sino, tira un warning pero segui)

if (DD) ⇒ Continuar
else ⇒ warning (“La matriz no es DD”) pero el código continua

- Inicializar X^{viejo} (vector de size n) y además ingresar la tolerancia (10^{-5})
- do{

iteraciones++;
do i = 1, …. , n
suma = 0
do j = 1,.....,n
if(j != i) suma= suma + $a_{ij}x_j^{viejo}$

$x_i^{nuevo} = \frac{1}{aii}(b_i - suma)$

do i = 1, …. , n

error = error + $(x_i^{nuevo} - x_i^{viejo})^2$

$error = \sqrt{error}$
X_{viejo} = X_{nuevo};

if(iteraciones == 10000)
(“Número máximo de iteraciones alcanzadas”)
break;

}while(error>tolerancia | iteraciones > 10000)

mostrar X_{nuevo}

Método de Gauss-Seidel

$$x_1^{n} = (b_1 - a_{12}x_2^{v} - \ldots\ldots - a_{1n}x_n^{v})/a_{11}$$
$$x_2 = (b_2 - a_{21}x_1^{n} - \ldots\ldots - a_{2n}x_n^{v})/a_{22}$$

pero x_1 ya lo tenemos, por lo tanto podríamos reemplazarlo, y así consecutivamente para cada X obteniendo lo siguiente:

$$X^{nuevo} = (b_i - \sum_{j<i} a_{ij} x_j^{nuevo} - \sum_{j>i} a_{ij} x_j^{viejo}) / a_{ii}$$

Algoritmo

- ❖ Definir la matriz A y el vector B (O bien la matriz ampliada)
- ❖ Verificar que la matriz sea diagonalmente dominante (si lo es, cheto, sino, tira un warning pero segui)

if (DD) ⇒ Continuar
else ⇒ warning ("La matriz no es DD") pero el código continua

- ❖ Inicializar X^{viejo} (vector de size n) y además ingresar la tolerancia (10^{-5})
- ❖ do{

do i = 1,.....,n
    if(i == 1){
        suma = 0
        do j = 1,.....,n
            suma = suma + $a_{ij} x_j^{viejo}$
        end do
        $x_i = \frac{b_i - suma}{aii}$
    }
    else{
        suma = 0
        do j = 1, …… , i-1
            suma = suma + $a_{ij} x_j^{nuevo}$ ⇒ Primer sumatoria de la fórmula
        end do
        do j = i+1,.....,n
            suma = suma + $a_{ij} x_j^{viejo}$ ⇒ Segunda sumatoria de la fórmula
        end do
        $x_i^{nuevo} = \frac{b_i - suma}{aii}$
end do
do i = 1, …. , n
    error = error + $(x_i^{nuevo} - x_i^{viejo})^2$
    end do
$error = \sqrt{error}$
    $X_{viejo} = X_{nuevo}$

}while(error>tolerancia | iteraciones > 10000)

Relajación

Representa una pequeña modificación al método de Gauss-Seidel de la siguiente manera:

$x_i^{nuevo} = \omega * x_i^{nuevo} + (1 - \omega) * x_i^{viejo}, \quad \omega \in [0, 2]$

Este cálculo se hace luego de calcular x_i^{nuevo}[ ] y se calcula para cada i

ω = coeficiente de relajación ⇒ $0 \leq \omega \leq 2$

Su función es, cuando el método no converge (es decir, A no es diagonalmente dominante y las iteraciones superan el máximo establecido) una elección apropiada de omega permite que el método converja. Sin embargo, dicha elección se realiza experimentalmente mediante prueba y error, por lo que, definirlo es complicado.

- ➔ Si $\omega = 1$ ⇒ el método no se modifica
- ➔ si $\omega \in [0, 1)$ ⇒ se denomina subrelajacion y se usa para hacer que un sistema que no convergía, converja (Al darle más peso a los x_{viejo} hace que la suma de x_{nuevo} + x_{viejo} con la corrección se acerque a la solución).
- ➔ Si $\omega \in [1, 2)$ ⇒ se denomina sobrerrelajación. Se usa para hacer que un método que ya convergía, converja más rápido.

07/09

Ajuste de Curvas

Interpolación

A partir de un experimento, obtenemos valores (x_0, y_0), (x_1, y_1), … , (x_n, y_n) que, siendo graficados, conforman una cierta curva. Ahora bien, estos son puntos, pero no tenemos una función que nos ajuste estos valores a una gráfica. Entonces, supongamos que queremos obtener una función que reproduzca estos datos, es decir:

$$f(x_i) = y_i, \quad para\, i = 1, \ldots, n + 1$$

Sabemos que la función f(x) no es única (hay muchas funciones que pueden pasar por dichos puntos). La manera más sencilla es decir que f(x) sea un polinomio de grado n tal que:

$P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots + a_n x^n$, $forma\ general\ con\ (n + 1)\ coeficientes\ (a_i\ con\ i = 0, \ldots, n)$

Entonces, si decimos que el polinomio debe pasar por los puntos, esto es:

$P(x_0) = y_0$

$P(x_1) = y_1$

$P(x_n) = y_n$

Por lo que obtendremos n+1 ecuaciones (que podríamos resolver con los métodos ya propuestos). A esto llamamos interpolación polinomial.

Interpolación Polinomial

Dado un conjunto de n+1 datos, encontramos un polinomio de grado n que pase por dichos puntos. La idea de este polinomio (polinomio interpolador) es obtener una estimación de y para un valor de $\hat{x}$ intermedio, es decir, que no pertenezca al conjunto de datos ($\hat{x} \neq x_i$). Esto es:

$P(\hat{x}) = \hat{y} \Rightarrow$ **Interpolar la función en** $\hat{x}$

#Nota: Extrapolar la función es evaluar el polinomio en un valor que está por fuera de mi rango x_i (si extrapolamos demasiado fuera del rango, perdemos precisión)
Los x_i (datos) se llaman "nodos" del polinomio interpolador (nodo debido a que la función pasa por ese punto)

Propiedad: El polinomio interpolador de grado n que interpola (n+1) nodos, es ÚNICO, sin embargo, hay varias formas de calcularlo

Ahora bien, cómo obtenemos este polinomio?

Interpolacion de Lagrange

Dado un conjunto de (n+1) datos, es decir, (x_0, y_0), (x_1, y_1), … , (x_n, y_n), con todos los x_i distintos, entonces la forma del polinomio interpolador calculado por Lagrange se calcula como:

$$P_n(x) = \sum_{k=0}^{n} y_k C_{nk}(x) \Rightarrow con\ C_{nk} = \prod_{i \neq k} \frac{x - x_i}{x_k - x_i}$$

Queremos $P(x_i) = y_i$

$$C_{nk} = \frac{x - x_0}{x_k - x_0} * \frac{x - x_1}{x_k - x_1} \dots \frac{x - x_{k-1}}{x_k - x_{k-1}} * \frac{x - x_{k+1}}{x_k - x_{k+1}} \dots \frac{x - x_n}{x_k - x_n}$$

De manera que $C_{nk}(x_k) = 1$ y $C_{nk}(x_j) = 0$, por lo tanto, el $P_n(x_k)$ me devuelve únicamente y_k

Si ahora lo que queremos es interpolar en un punto intermedio llamado $\hat{x}$, esto es:

$$P_n(\hat{x}) = \hat{y} = \sum_{k=0}^{n} y_k C_{nk}(\hat{x})\ con\ C_{nk}(\hat{x}) = \prod_{i \neq k} \frac{\hat{x} - x_i}{x_k - x_i}$$

Algoritmo:

- Ingresar los datos (nodos) (x_i, y_i)
- Ingresar el $\hat{x}$
- Calcular $P_n(\hat{x})$

suma = 0
Do k = 0, …… , n
producto = 1
Do i = 0, ……, n ⇒ Calcula Cnk
if(i != k)
producto = producto * $(\hat{x}\text{-}x_i)/(x_k\text{-}x_i)$
suma = suma + (y_k*producto)

- Mostrar el valor de la suma (correspondiente a $\hat{y}$)

Desventaja: El método me devuelve el valor interpolado pero no el polinomio interpolador
#Nota: Este método es eficiente para poca cantidad de datos

Ejemplo: Usar interpolación de Lagrange en los nodos
$x_0 = 1, x_1 = 2, x_2 = 2,5$
para aproximar $f(x) = x + \frac{2}{x}$ en los puntos x=1,5 y x=1.2
$y_0 = f(x_0) \Rightarrow y_0 = f(1) = 3$
$y_1 = f(x_1) \Rightarrow y_1 = f(2) = 3$
$y_2 = f(x_2) \Rightarrow y_2 = f(2.5) = 3.3$

por lo tanto, los nodos son (1 ; 3), (2 ; 3), (2,5 ; 3,3)

Construyamos la interpolación de Lagrange

$$P_2(\hat{x}) = \hat{y} = \sum_{k=0}^{2} y_k C_{nk}(\widehat{x})$$

En general

$$C_{20} = \frac{x-x_1}{x_0-x_1} * \frac{x-x_2}{x_0-x_2} = \frac{(x-2)}{(1-2)} * \frac{(x-2,5)}{(1-2,5)}$$
$$C_{20} = \frac{(x-2)(x-2.5)}{1.5}$$
$$C_{21} = \frac{x-x_0}{x_1-x_0} * \frac{x-x_2}{x_1-x_2} = \frac{(x-1)}{(2-1)} * \frac{(x-2,5)}{(2-2,5)}$$
$$C_{21} = \frac{(x-1)(x-2.5)}{-0.5}$$
$$C_{22} = \frac{(x-1)(x-2)}{0.75}$$

Y en el polinomio:

$P_2(x) = 3\frac{(x-2)(x-2.5)}{1.5} - 3\frac{(x-1)(x-2.5)}{0.5} + 3.3\frac{(x-1)(x-2)}{0.75}$

Con lo que, quedaría evaluar dicha expresión en los valores solicitados, con un error de:

$|f(x) - P_2(x)|$

Ahora bien, como dijimos, este método no nos devuelve el polinomio interpolador, por lo que vamos a ver cómo calcularlo en su forma de ecuación.

Partiendo de la forma general para un polinomio de grado n:

$P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_nx^n$

Necesitamos que $P_n(x_i) = y_i$

$a_0 + a_1x_0 + a_2x_0^2 + a_3x_0^3 + \ldots + a_0x_0^n = y_0$

$a_0 + a_1x_1 + a_2x_1^2 + a_3x_1^3 + \ldots + a_0x_1^n = y_1$

$a_0 + a_1x_2 + a_2x_2^2 + a_3x_2^3 + \ldots + a_0x_2^n = y_2$

En forma matricial:

$$\begin{matrix} 1 & x_0 & x_0^2 & x_0^3 & . & . & x_0^n & a_0 & & y_0 \\ 1 & x_1 & x_1^2 & x_1^3 & . & . & x_1^n & a_1 & & y_1 \\ 1 & x_2 & x_2^2 & x_2^3 & . & . & x_2^n & a_2 & & y_2 \\ . & . & & . & . & . & & & & . \\ . & . & & . & . & & . & & = & . \\ . & . & & . & . & . & & . & & . \\ 1 & x_n & x_n^2 & x_n^3 & . & . & x_n^n & a_n & & y_n \end{matrix}$$

Es decir, es un sistema $Ax = b$ (sistema de ecuaciones lineales) donde los coeficientes a_i son las incógnitas.

Algoritmo:

➔ Ingresar los datos (nodos) (xi,yi)

➔ Armar las matrices A y B (el elemento ij de la matriz es x_i^j)

Para armar la matriz A:

Do i = 0, ……, n

Do j = 0, ……, n

$A[i][j] = pow(x_i, j)$

$b[i] = y_i$

➔ Resolver el sistema

Gauss(A, B, n, X) ⇒ Devuelve los ai

➔ Construir el polinomio (con un printf)(y evaluar, de ser necesario) $\sum_{i=0}^{n} a_i x^i$

Si evaluamos:

➢ Ingresar $\hat{x}$

➢ Calcular la suma

suma = 0

Do i = 0, ……, n

suma = suma+ai*pow($\hat{x}$, i)

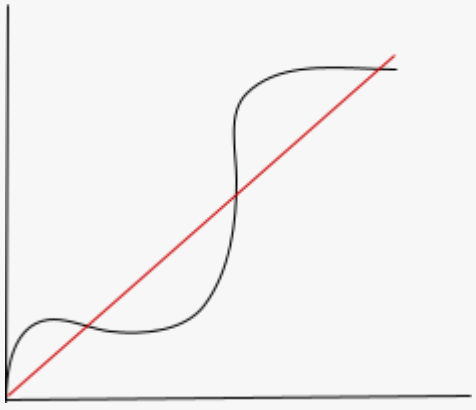
➢ Mostrar el resultado

Ejercicio: En el problema 1 de la guia 5, obtener P2(x) y graficarlo junto con f(x) (ambas curvas deberían cruzarse en los nodos)

14/09

Regresión por Cuadrados Mínimos

Hemos visto hasta ahora, el método de la interpolación polinomial. Sin embargo, dijimos que este método no era eficiente ni conveniente para una gran cantidad de datos. Para esto, lo que podemos hacer es aproximar el comportamiento de los datos mediante una función sencilla que se acerque a los puntos, en vez de pasar exactamente por cada uno de ellos.



Por lo que, ahora nuestro objetivo es hallar una función f(x) que ajuste los datos, pero no necesariamente de forma exacta. Por lo que, al aproximar cada punto, esto nos va a llevar a que:

⇒ En cada punto: $e_i = | y_i - f(x_i) |$

El metodo de minimos cuadrados consiste en minimizar:

Error total = $e = \sum_{i=0}^{n} e_i^2$ ⇒ queremos que este valor sea lo más pequeño posible

$$e = \sum_{i=0}^{n} (y_i - f(x_i))^2$$

Regresión Lineal

Este es el método más sencillo de regresión. Diremos que $f(x) = ax + b$. Por lo tanto

$e = \sum_{i=0}^{n} (ax_i + b - y_i)^2$ ⇒ queremos minimizar esto, por lo tanto debemos buscar el valor para a y b tal que la recta sea lo más cercana posible a los puntos.

Para conseguir esto, vamos a derivar e igualar a 0, es decir:

1) derivada de e respecto de a ⇒ $\frac{de}{da} = 0 = \sum_{i=0}^{n} 2(ax_i + b - y_i)x_i$

2) derivada de e respecto de b ⇒ $\frac{de}{db} = 0 = \sum_{i=0}^{n} 2(ax_i + b - y_i) * 1$

Por lo que tenemos dos ecuaciones con dos incógnitas, a y b

de 1) ⇒ $\sum(ax_i^2 + bx_i - y_i x_i) = 0$

$$a\sum_{i=0}^{n} x_i^2 + b\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} y_i x_i$$

de 2) ⇒ $\sum_{i=0}^{n} (ax_i + b - y_i) = 0$

$$a\sum_{i=0}^{n} x_i + b\sum_{i=0}^{n} 1 = \sum_{i=0}^{n} y_i$$

$$a\sum_{i=0}^{n} x_i + b(n+1) = \sum_{i=0}^{n} y_i$$

En forma matricial $Ax = B$

$\sum_{i=0}^{n} x_i^2$	$\sum_{i=0}^{n} x_i$		a		$\sum_{i=0}^{n} y_i x_i$
				=	
$\sum_{i=0}^{n} x_i$	$n+1$		b		$\sum_{i=0}^{n} y_i$

donde A son las sumatorias, x son el a y b y B las sumatorias de yi

Coeficiente de Correlación

Recordemos del laboratorio, si tenemos un conjunto de mediciones yi (i=0,.....,n)

$$\Rightarrow \bar{y} = (\sum_{i=0}^{n} y_i)/(n+1) \Rightarrow \sigma = \sqrt{(\sum_{i=0}^{n} (\bar{y} - y_i)^2)/n} = \sqrt{S_t/n}$$

con $S_t = \sum_{i=0}^{n} (\bar{y} - y_i)^2$

La pregunta a responder es: ¿Cómo sabemos que nuestros datos corresponden realmente a una relación lineal y no son simplemente varias mediciones de un mismo valor, con mucha dispersión?

$$r = \sqrt{(|e - S_t|)/S_t}$$

Calcula el error relativo de los errores, básicamente, verifica que, mientras más cercano sea r de 1, la regresión lineal es la herramienta correcta. En cambio, si r es cercano a 0, el ajuste lineal no necesariamente es una dependencia en x, sino que podría ser variaciones por dispersión estadística.

Veamos un ejemplo: Supongamos que tenemos una serie de mediciones arroja los siguientes datos.

x	y
0	1.2
1	2.7
1.5	3.9
3	7.1
5	10

$\sum_{i=0}^{n} x_i^2$	$\sum_{i=0}^{n} x_i$		a		$\sum_{i=0}^{n} y_i x_i$
				=	
$\sum_{i=0}^{n} x_i$	$n+1$		b		$\sum_{i=0}^{n} y_i$

37.25	10.5		a		79.85
				=	
10.5	5		b		24.9

a = 1.81 y b=1.17 ⇒ f(x) = 1.81x+1.17

$$e = \sum_{i=0}^{n} (y_i - f(x_i))^2 = \sum_{i=0}^{n} (y_i - 1.81x_i - 1.17)^2 = 0.377925$$

$$s_t = \sum_{i=0}^{n} (y_i - \overline{y})^2 = 50.348$$

$$r = \sqrt{\frac{|e - s_t|}{s_t}} = 0.99624$$

Regresión Polinomial

Cuando hablamos de regresión polinomial, hablamos de generalizar el método de regresión lineal, pero para ajustar los datos siguiendo un perfil polinomial, cuyo grado p es mayor que 1 y no necesariamente fijo, tal que ajuste los datos, es decir:

$$P(x) = a_0 + a_1 x_1 + a_2 x_2 + \ldots\ldots + a_p x_p$$

Buscando minimizar:

$$e = \sum_{i=0}^{n} (a_0 + a_1 x_i + \ldots + a_p x_i^p - y_i)^2$$

por lo que armamos el sistema de ecuaciones:

$$\frac{de}{da_0} = 0$$
$$\frac{de}{da_1} = 0$$
$$\frac{de}{da_2} = 0$$
.
.
.
$$\frac{de}{da_p} = 0$$

lo que es:

$$\frac{de}{da_k} = 0 \qquad \text{con k=0,.....,p}$$

$$0 = \sum_{i=0}^{n} (a_o x_i^k + a_1 x_i^{k+1} + a_2 x_i^{k+2} + a_3 x_i^{k+3} + \cdots + a_p x_i^{k+p} - y_i x_i^k)$$

distribuyendo el x_i^k y la sumatoria ⇒

$$\begin{vmatrix} \sum_{i=0}^{n} x_i^{0} & \sum_{i=0}^{n} x_i^{0+1} & \sum_{i=0}^{n} x_i^{0+2} & \dots \sum_{i=0}^{n} x_i^{0+p} \\ \sum_{i=0}^{n} x_i^{1} & \sum_{i=0}^{n} x_i^{1+1} & \sum_{i=0}^{n} x_i^{1+2} & \dots \sum_{i=0}^{n} x_i^{1+p} \\ & . & & \\ & . & & \\ \sum_{i=0}^{n} x_i^{p} & \sum_{i=0}^{n} x_i^{p+1} & \sum_{i=0}^{n} x_i^{p+2} & \dots \sum_{i=0}^{n} x_i^{p+p} \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ \\ \\ a_p \end{vmatrix} = \begin{vmatrix} \sum_{i=0}^{n} y_i x_i^{0} \\ \sum_{i=0}^{n} y_i x_i^{1} \\ \\ \\ \sum_{i=0}^{n} y_i x_i^{p} \end{vmatrix}$$

$$A_{lm} = \sum_{i=0}^{n} x_i^{(l+m)}$$

$$b_l = \sum_{i=0}^{n} y_i \cdot x_i^{l}$$

Algoritmo

➔ Ingresar los n+1 datos (xi, yi)
➔ Ingresar p
➔ Armar la matriz aumentada

```
Do l=0, … , p ⇒ Recorre las filas
        sumaxy = 0
        do i = 0, …., n
                sumaxy = sumaxy + pow (x(i),l)*y(i)
        b(l) = sumaxy
        Do m = 0, …., p ⇒ Recorre las columnas
                sumax = 0
                Do i = 0, …., n
                        sumax= sumax + pow(x(i),l+m)
                A(l,m)=sumax
```

➔ Llamar Gauss-Pivot(A,B,x,n)
➔ Mostrar el conjunto solución (x)

#Nota: Este método me permite hacer la regresión de cualquier cosa.

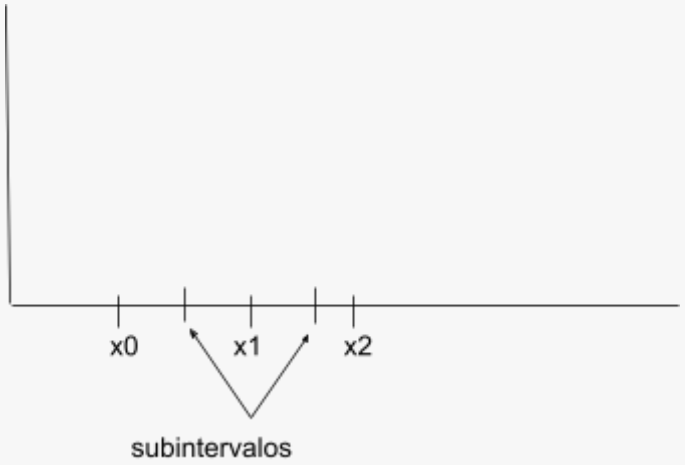
28/09

Interpolación Segmentaria (Curvas Spline)

En las clases anteriores lo que hacíamos era: tomar $n+1$ datos dados en la forma (x_0, y_0), (x_1, y_1) ..... (x_n, y_n) y con estos datos construimos $P_n(x) = \sum_{i=0}^{n} a_i x^i$

El problema es que cuando n es muy grande, el polinomio tiende a ser demasiado oscilante. Una forma de solucionar esto era con regresión. Otra alternativa que se usa mucho, es usar polinomios de grado más bajo por trozos (es decir, sub-intervalos determinados por los nodos). A estos polinomios se los llama **"trazadores"** o "**curvas spline"**

Si tenemos n+1 datos, entonces tenemos n sub-intervalos.



Spline lineal

El caso más sencillo consiste en tomar polinomios de grado 1 (rectas) en cada subintervalo (se fuerzan a que pasen por los nodos). Básicamente, si tenemos

$(n+1)\ datos\ (x_i, y_i)\ con\ i = 0, \cdots, n \rightarrow subintervalos$

$En\ el\ sub-intervalo\ k\colon [x_k, x_{k+1}]\ (k = 0, \cdots, n-1)$

$f_k(x) = f(x_k) + m_k(x - x_k)$

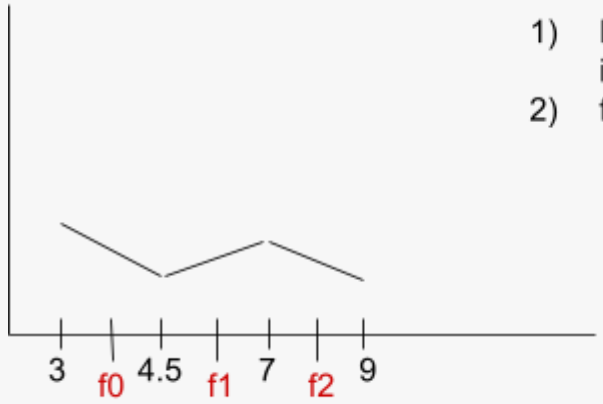
Con este procedimiento, si queremos interpolar en algún punto $\hat{x}$.

I. Determinar en qué subintervalo esta $\hat{x}$ (encontrar k).

II. Evaluar $f_k(\hat{x})$

Ejemplo: Dados los siguientes datos obtener, mediante interpolación spline lineal el valor de f(5)

x	y
3	2.5
4.5	1
7	2.5
9	0.5



1) El punto está en el intervalo 1 [x1, x2]
2) f1(x) = f(x1)+m1(x-x1)

2) $f_1(x) = f(x_1) + m_1(x - x_1)$

$m_1 = (y_2 - y_1)/(x_2 - x_1)$

$m_1 = (2.5 - 1)/(7 - 4.5) = 0.6$

$f_1(x) = 1 + 0.6 \cdot (x - 4.5)$

Evaluamos en $\hat{x} \rightarrow f(5) = 1.3$

Desventaja de spline lineal:

La función resultante no es suave (tiene picos en los nodos).

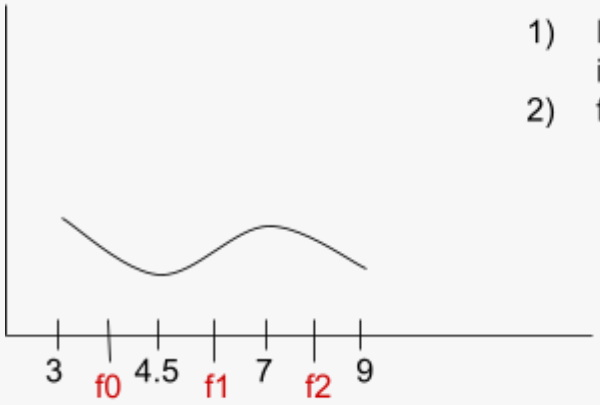
Posible solución:

Usar en cada sub-intervalo polinomios cuadráticos (o cúbicos) y pedir que “empalmen” de forma suave en los nodos.
Para estos casos, la función en cada sub-intervalo k, $f_k = a_k x^2 + b_k x + c_k \; con \; k = 0, \cdots, n - 1$

En este caso, buscamos a_k, b_k y c_k, por lo que tendríamos 3n coeficientes (3 por cada sub-intervalo, para n intervalos)

Spline Cuadrático

Manteniendo el ejemplo anterior, tenemos 9 incógnitas a determinar:



1) El punto está en el intervalo 1 [x1, x2]
2) f1(x) = f(x1)+m1(x-x1)

Sabemos que:

$$f_0(x_0) = y_0 \rightarrow a_0 \cdot 3^2 + b_0 \cdot 3 + c_0 = 2.5$$
$$f_0(x_1) = y_1 \rightarrow a_1 \cdot 4.5^2 + b_1 \cdot 4.5 + c_1 = 1$$
$$f_1(x_1) = y_1 \rightarrow a_1 \cdot 4.5^2 + b_1 \cdot 4.5 + c_1 = 1$$
$$f_1(x_2) = y_2 \rightarrow a_1 \cdot 7^2 + b_1 \cdot 7 + c_1 = 2.5$$
$$f_2(x_2) = y_2 \rightarrow a_2 \cdot 7^2 + b_2 \cdot 7 + c_2 = 2.5$$
$$f_2(x_3) = y_3 \rightarrow a_2 \cdot 9^2 + b_2 \cdot 9 + c_2 = 0.5$$

De acá ya obtenemos 6 ecuaciones de las 9 necesarias. Las otras tres restantes vienen de pedir que la derivada sea continua en los nodos interiores, esto es:

$$f_k(x) = a_k x^2 + b_k x + c_k$$
$$f'_k(x) = 2a_k x + b_k$$

$$f_0'(x_1) = f_1'(x_1) \rightarrow 2a_0 4.5 + b_0 = 2a_1 4.5 + b_1 \rightarrow 9a_0 + b_0 - 9a_1 - b_1 = 0$$
$$f_1'(x_2) = f_2'(x_2) \rightarrow 2a_1 7 + b_1 = 2a_2 7 + b_2 \rightarrow 14a_1 + b_1 - 14a_2 - b_2 = 0$$

La tercera ecuación se impone arbitrariamente. Esto es: $f''_0(x_0) = 0 \Rightarrow$ Condición Normal

$$2a_0 = 0 \Rightarrow a_0 = 0$$

Obteniendo así un sistema de ecuaciones lineales 8x8 dado por:

3	1	0	0	0	0	0	0		b_0		2.5
4.5	1	0	0	0	0	0	0		c_0		1
0	0	20.25	4.5	1	0	0	0		a_1		1
0	0	49	7	1	0	0	0		b_1	=	2.5
0	0	0	0	0	49	7	1		c_1		2.5
0	0	0	0	0	81	9	1		a_2		0.5
1	0	-9	-1	0	0	0	0		b_2		0
0	0	14	1	0	-14	-1	0		c_2		0

Por lo que, si ingresamos esta matriz a eliminación-gaussiana obtenemos el siguiente conjunto de solución:

$$f_0(x) = -x + 5.5$$

$$f_1(x) = 0.64x^2 - 6.76x + 18.46$$

$$f_2(x) = -1.6x^2 + 24.6x - 91.3$$

$$\widehat{x} = 5 : intervalo\ 1 \rightarrow f_1(5) = 0.66$$

Splines cúbicas

Son las más usadas (porque son más suaves que las cuadráticas) ya que permite obtener una función C^2 (continua hasta la 2da derivada)
Supongamos que tenemos n+1 puntos, es decir, n intervalos donde cada $f_k(x)$ va a ser de la forma:

$$f_k(x) = a_k x^3 + b_k x^2 + c_k x + d_k$$

por lo que tenemos 4n incógnitas (4 incógnitas por intervalo, n intervalos). Para que las funciones ajusten cada nodo, tenemos que:

$f_0(x_0) = y_0$
$f_0(x_1) = y_1$
$f_1(x_1) = y_1$
$f_1(x_2) = y_2$
$f_2(x_2) = y_2$
$f_2(x_3) = y_3$
.
.
.
$f_{n-2}(x_{n-1}) = y_{n-1}$
$f_{n-1}(x_{n-1}) = y_{n-1}$
$f_{n-1}(x_n) = y_n$

Teniendo así 2n ecuaciones (2 ecuaciones por cada punto interno y las 2 de los extremos). Para obtener las ecuaciones restantes tenemos que:

- Derivadas continuas en los nodos interiores:

$f_0'(x_1) = f_1'(x_1)$
$f_1'(x_2) = f_2'(x_2)$
$f_2'(x_3) = f_3'(x_3)$
$f_{n-2}'(x_{n-1}) = f_{n-1}'(x_{n-1})$

obteniendo n-1 ecuaciones

- Derivadas segundas continuas en los nodos interiores

$f_0''(x_1) = f_1''(x_1)$
$f_1''(x_2) = f_2''(x_2)$
$f_2''(x_3) = f_3''(x_3)$
.
.
.
$f_{n-2}''(x_{n-1}) = f_{n-1}''(x_{n-1})$

obteniendo n-1 ecuaciones

Faltando así 2 ecuaciones, que serán dadas de manera arbitraria diciendo que:
$f_0''(x_0) = 0$ y $f_{n-1}''(x_n) = 0$ ⇒ Condiciones Normales

Vamos a tener un sistema de ecuaciones 4n x 4n ⇒ A[4n][4n]

z[4n] = incógnitas: (a0, b0, c0, d0, a1, b1, c1, d1, …, an-1, bn-1, cn-1,dn-1)

$$a_k = z[4k]$$

$$b_k = z[4k + 1]$$

$$c_k = z[4k + 2]$$

$$d_k = z[4k + 3]$$

De las primeras 2n ecuaciones obtenemos que:

$$f_k(x_k) = y_k \Rightarrow a_k x_k^3 + b_k x_k^2 + c_k x_k + d_k = y_k$$

$$f_k(x_{k+1}) = y_{k+1} \Rightarrow a_k x_{k+1}^3 + b_k x_{k+1}^2 + c_k x_{k+1} + d_k = y_{k+1}$$

con k = 0, ……, n-1

Si reemplazamos en las expresiones de Z

$$f_k(x_k) = y_k \Rightarrow z(4k)x_k^3 + z(4k + 1)x_k^2 + z(4k + 2)x_k^1 + z(4k + 3)x_k^0 = y_k \Rightarrow 2k$$

$$f_k(x_{k+1}) = y_{k+1} \Rightarrow z(4k)x_{k+1}^3 + z(4k + 1)x_{k+1}^2 + z(4k + 2)x_{k+1}^1 + z(4k + 3)x_k{}^0 = y_{k+1} \Rightarrow 2k + 1$$

Es decir, si indexamos la matriz, la primera ecuación equivale a las filas pares de la matriz, y la segunda para las impares.

Do k=0,.....,n-1

  Do j=0,...,3

    $A(2k, 4k + j) = x_k^{3-j}$

    $A(2k + 1, 4k + j) = x_{k+1}^{3-j}$

  end Do

  $ind(2k) = y_k$

  $ind(2k + 1) = y_{k+1}$

end Do

Construyendo así las 2n primeras filas arrancando desde 0 hasta n-1 (correspondientes a que la función ajuste de manera exacta a los nodos).

Construyamos ahora las n-1 ecuaciones correspondientes a la igualdad de las derivadas:

$$f'_k(x_{k+1}) = f'_{k+1}(x_{k+1}) \quad con\, k = 0, \cdots, n-2$$

$$f_k(x) = a_k x^3 + b_k x^2 + c_k x + d_k \Rightarrow f'(x) = 3a_k x^2 + 2b_k x + c_k$$

Por lo tanto

$$f'_k(x_{k+1}) = f'_{k+1}(x_{k+1}) \Rightarrow 3a_k x_{k+1}{}^2 + 2b_k x_{k+1}{}^1 + c_k x_{k+1}{}^0 = 3a_{k+1} x_{k+1}{}^2 + 2b_{k+1} x_{k+1}{}^1 + c_{k+1} x_{k+1}{}^0$$

$$z(4k)3x_{k+1}{}^2 + z(4k+1)2x_{k+1}{}^1 + z(4k+2)x_{k+1}{}^0 - z(4(k+1))3x_{k+1}{}^2 - z(4(k+1)+1)x_{k+1}{}^1 - z(4(k+1)+2)x_{k+1}{}^0 = 0$$

Do i = 2n, ….., 3n-2
    Do k = 0, ….., n-2
        Do j = 0, …, 2
            $A(i, 4k+j) = (3-j) * x_{k+1}^{(2-j)}$
            $A(i, 4(k+1)+j) = (3-j) * x_{k+1}^{(2-j)}$
        end Do
    end Do
    $ind(i) = 0$
end Do

#Nota: filas que van desde 2n hasta 3n-2

Construyamos ahora las ecuaciones correspondientes a las derivadas segundas

$f_0''(x_1) = f_1''(x_1)$
$f_1''(x_2) = f_2''(x_2)$
$f_2''(x_3) = f_3''(x_3)$
.
.
.
$f_{n-2}''(x_{n-1}) = f_{n-1}''(x_{n-1})$

$$f_k(x) = a_k x^3 + b_k x^2 + c_k x + d_k \Rightarrow f''(x) = 6a_k x + 2b_k$$

Por lo tanto

$$6a_k x_{k+1}{}^1 + 2b_k x_{k+1}{}^0 = 6a_{k+1} x_{k+1}{}^1 + 2b_{k+1} x_{k+1}{}^0$$

$$z(4k)6x_{k+1}{}^{1} + z(4k+1)2x_{k+1}{}^{0} - z(4(k+1))6x_{k+1}{}^{1} + z(4(k+1)+1)2x_{k+1}{}^{0} = 0$$

Do i = 3n-1, …., 4n-3

Do k= 0, …., n-2

$A(i, 4k) = 6x_{k+1}$

$A(i,\ 4k+1) = 2$

$A(i,\ 4k+4) = -6x_{k+1}$

$A(i,\ 4k+5) = -2$

end Do

$ind(i) = 0$

end Do

#Nota: filas que van desde 3n-1 hasta 4n-3

Por último, las 2 condiciones restantes son:

$f_0''(x_0) = 0$ y $f_{n-1}''(x_n) = 0$

Esto es:

$f_0''(x_0) = 6a_0x_0 + 2b_0 = 0$ y $f_{n-1}''(x_n) = 6a_{n-1}x_{n-1} + 2b_{n-1} = 0$

$z(0)\cdot 6x_0 + z(1)\cdot 2b_0 = 0$ $z(4(n-1))\cdot 6x_{n-1} + z(4(n-1)+1)\cdot 2 = 0$

$A(4n-2,\ 0) = 6x_0$ $A(4n-1,\ 4n-4) = 6x_{n-1}$

$A(4n-2,\ 1) = 2$ $A(4n-1,\ 4n-3) = 2$

$ind(4n-2) = 0$ $ind(4n-1) = 0$

5/10

Métodos de Integración Numérica

Objetivo: calcular $\int_a^b f(x)\,dx = I$ para regiones complicadas.

Vemos los siguientes métodos para integración:

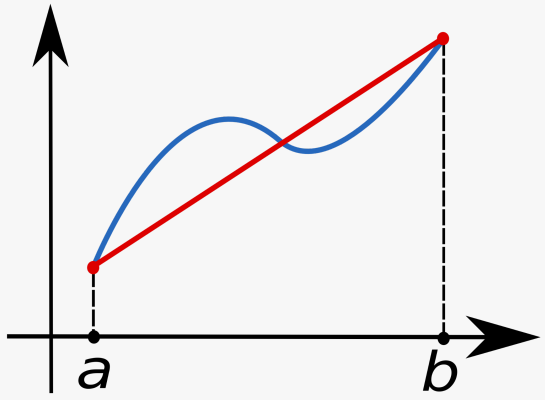
Fórmulas de Newton - Cotes

Se basa en reemplazar la función complicada o incluso un conjunto de datos, por un polinomio de aproximación que es fácil de integrar.

$$I \simeq \int_a^b P_n(x)\,dx \to P_n(x) = a_0 + a_1 x + \cdots + a_n x^n$$

I. **Regla del Trapecio**:

Es el más sencillo ya que corresponde a un polinomio de aproximación de grado uno (recta)



En donde la integral exacta es el área bajo la curva azul, y la integral aproximada será el área bajo la recta roja. Sin embargo, en ciertos casos esto puede llegar a ser bastante impreciso (dependiendo de la función)

$$I \simeq \int_a^b P_1(x)\,dx \to P_1(x) = recta\ que\ une\ (a, f(a))\ con\ (b,\ f(b))$$

$$P_0(x) = f(a) + \frac{f(b)-f(b)}{b-a}(x - a)$$

$$I \simeq \int_a^b f(a)\,dx + \int_a^b \frac{f(b)-f(a)}{b-a}(x - a)dx$$

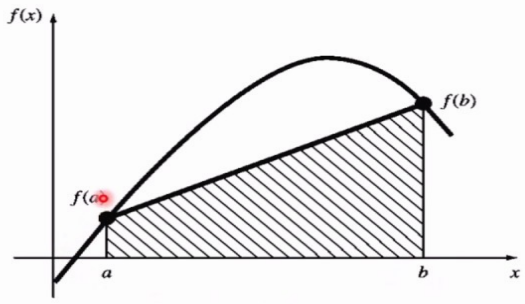
$$= f(a)\cdot x_a^b + \frac{f(b)-f(b)}{b-a} * \frac{(b-a)^2}{2}{}_a^b$$

$$I \simeq f(a)(b-a) + \frac{f(b)-f(b)}{b-a} * \frac{(b-a)^2}{2}$$

Operando obtenemos:

$$I \simeq \frac{2 \cdot f(a)(b-a)^2+(f(b)-f(a))(b-a)^2}{2(b-a)}$$

$$I \simeq (f(a) + f(b)) * \frac{(b-a)}{2} \Rightarrow \textbf{Fórmula del Trapecio}$$



$$I = (b-a)\left(\frac{f(a)+f(b)}{2}\right)$$

con un error de:

$$e = \left|I_{exacta} - I_{aproximada}\right|$$

Se puede demostrar que:

$$e = -\frac{1}{12} f''(\varrho)(b-a)^3 \qquad con\ \varrho \in (a,b)$$

Como el error depende de la derivada segunda, el método del trapecio calculara de manera exacta, las integrales para funciones hasta grado 1

#Nota: Cuanto más grande sea el intervalo, más inexacto es el método.
$f'' = 0 \rightarrow La\ regla\ del\ trapecio\ da\ el\ resultado\ exacto$

Ejemplo: Sea $f(x) = x^2 + 1$
Calcular

a) $\int_0^3 f(x)\,dx$
b) la aproximación utilizando la regla del trapecio
c) el error absoluto exacto y el aproximado

A) $$I = \frac{x^3}{3} + x\Big|_0^3 = 12 = I_{exacta}$$

B) $I \simeq (f(a) + f(b)) * \frac{(b-a)}{2} = 16.5 = I_{aproximada}$

C) $e_{exacto} = |I_{exacta} - I_{aproximada}| = 4.5$

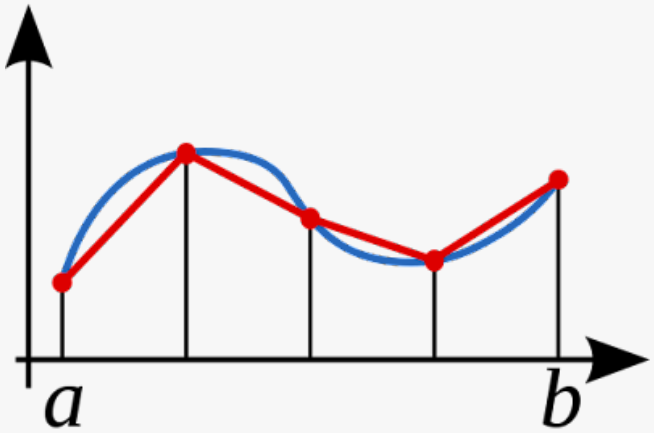
$e_{por} = \frac{e_{ex}}{I_{ex}} * 100 = 34.5\%$

D) $e_{aprox} = \left| -\frac{1}{12} f''(\varrho)(b-a)^3 \right| = 4.5$

El signo del error determina si estamos sobreestimando o subestimando el valor de la integral. De ser negativo, estamos sobreestimando, caso contrario, subestimamos.

II. Regla del Trapecio Compuesta

Una forma de mejorar la precisión es dividir el intervalo [a,b] en n subintervalos, y aplicar el método del trapecio en cada subintervalo.



Tomamos (n+1) puntos equiespaciados ($x_0, x_1, x_2, \ldots\ldots, x_n$) donde $x_0 = a$, $x_n = b$ y

$h = \frac{b-a}{n}$ (distancia entre cada x_i)

Obteniendo así:

$$I = \int_{x_0}^{x_1} f(x)\,dx + \int_{x_1}^{x_2} f(x)\,dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)\,dx$$

$$I \simeq \frac{x_1 - x_0}{2}(f(x_1) + f(x_0)) + \frac{x_2 - x_1}{2}(f(x_2) + f(x_1)) + \cdots + \frac{x_n - x_{n-1}}{2}(f(x_n) + f(x_{n-1}))$$

$$I \simeq \frac{h}{2}\left(f(x_0) + f(x_n) + \sum_{i=1}^{n-1} 2f(x_i)\right)$$

$$I \simeq \frac{b-a}{2n}\left(f(a) + f(b) + \sum_{i=1}^{n-1} 2f(a+ih)\right) \Rightarrow \textbf{Fórmula del Trapecio Compuesta}$$

#Nota: Si queremos usar el trapecio normal, n = 1

Con un error de:

$$e = \sum_{i=0}^{n} e_i = \sum_{i=0}^{n-1} -\frac{1}{12} f''(\zeta_i)(x_{i+1} - x_i)^3 = -\frac{1}{12} \sum_{i=0}^{n-1} f''(\varsigma_i) \left(\frac{b-a}{n}\right)^3 = -\frac{1}{12} \frac{(b-a)^3}{n^2} \overline{f''}$$

Retomando el ejemplo anterior: Si usamos 2 y 3 intervalos, obtenemos que

$I \simeq \frac{3}{4}(f(0) + f(3) + 2 \cdot f(1.5)$

$\simeq \frac{3}{4}(1 + 10 + 2 * 3.25 = 13.125$

$e_{exacto} = |12 - 13.125| = 1.125 = \frac{4.5}{2^2}$

Con 3 subintervalos

$h = \frac{b-a}{n} = 3/3 = 1$

$I \simeq \frac{3-9}{2 \cdot 3}(f(0) + f(3) + 2 \cdot f(1) + 2 \cdot f(2))$

$I \simeq 12.5$

$e_{exacto} = 0.5 = \frac{4.5}{3^2}$

$$I \simeq \frac{b-a}{2n}(f(a) + f(b) + \sum_{i=1}^{n-1} 2f(a + ih))$$

Algoritmo

- Definir f(x)
- Ingresar los límites de integración (a y b)
- Ingresar el número de subintervalos (n)
- Calcular I

  suma = f(a)+f(b)
  h = (b-a)/n
  do i = 1, ….., n-1 (incluidos)
  suma = suma + 2*f(a+ih)
  suma = suma * h/2
- Mostrar el resultado (suma)

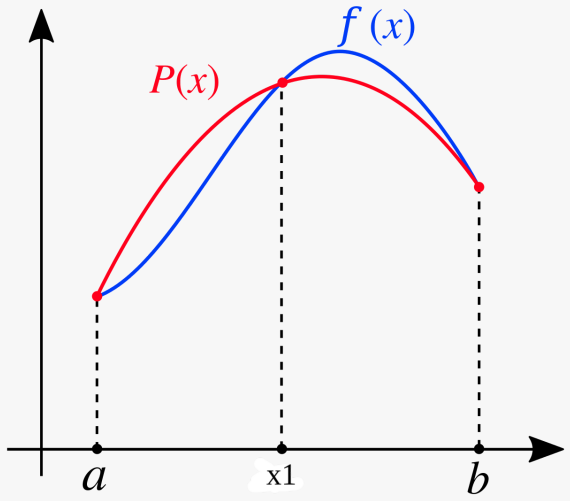
#Nota: Todavía no calculamos el error (por falta de herramientas) Podemos calcular error exacto en caso de que la integral exacta sea factible de calcular

III. Regla de Simpson 1/3

Otra forma de mejorar precisión es usar, en el intervalo de integración, un polinomio de grado mayor que 1.

Si usamos un polinomio de grado 2, queremos que dicho polinomio pase por los puntos a, b y un punto t intermedio llamado x_1

$$\int_a^b f(x)\,dx \simeq \int_a^b P_2(x)\,dx$$



Es decir, una parábola de nodos x_0, x_1, x_2

Si $P_2(x)$ es el polinomio de lagrange

$$f(x_0)\frac{x-x_1}{x_0-x_1} * \frac{x-x_2}{x_0-x_2} + f(x_1)\frac{x-x_0}{x_1-x_0} * \frac{x-x_2}{x_1-x_2} + f(x_2)\frac{x-x_0}{x_2-x_0} * \frac{x-x_1}{x_2-x_1}$$

Integrando y operando

$$I \simeq \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$$

En nuestro ejemplo: $\int_0^3 (x^2 + 1)\,dx$

$$I \simeq \frac{3}{2}\frac{1}{3}[f(x_0) + 4f(x_1) + f(3)]$$

$$I \simeq \frac{1}{2}[1 + 4 * 3.25 + 10] = 12$$

IV. Regla de Simpson Compuesta

Tomamos intervalos de 2 valores (lo que me da como condición que la cantidad de intervalos sea par y por ende la cantidad de puntos sea impar) y aplicamos Regla de Simpson ⅓ para cada intervalo.
Por lo que obtendremos:

$$I = \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx + \int_{x_4}^{x_6} f(x)\,dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)\,dx$$

$$= \frac{h}{3}(f(x_0) + 4\cdot f(x_1) + f(x_2)) + \frac{h}{3}(f(x_2) + 4\cdot f(x_3) + f(x_4)) + \frac{h}{3}(f(x_4) + 4\cdot f(x_5) + f(x_6))$$

$$I = \frac{b-a}{3n} \cdot [f(x_0) + f(x_n) + \sum_{i\ impar} 4 \cdot f(x_i) + \sum_{i\ pares} 2 \cdot f(x_i)]$$

Algoritmo

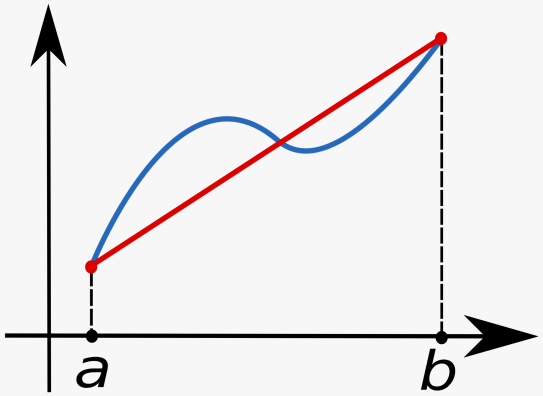
- Definir f(x)
- Ingresar los límites de integración (a y b)
- Ingresar el número de subintervalos (n) con un warning de que debe ser un número par
- Calcular I
  - h = (b-a)/n
  - suma = f(a) + f(b)
  - Do i = 1, ….., (n/2) - 1
    - x = a + 2ih
    - suma = suma + 2*f(x) + 4*f(x-h) ⇒ Llega hasta x_{n-2}
  - suma = h/3 * (suma+4*f(b-h))
- Mostrar resultado

17/10

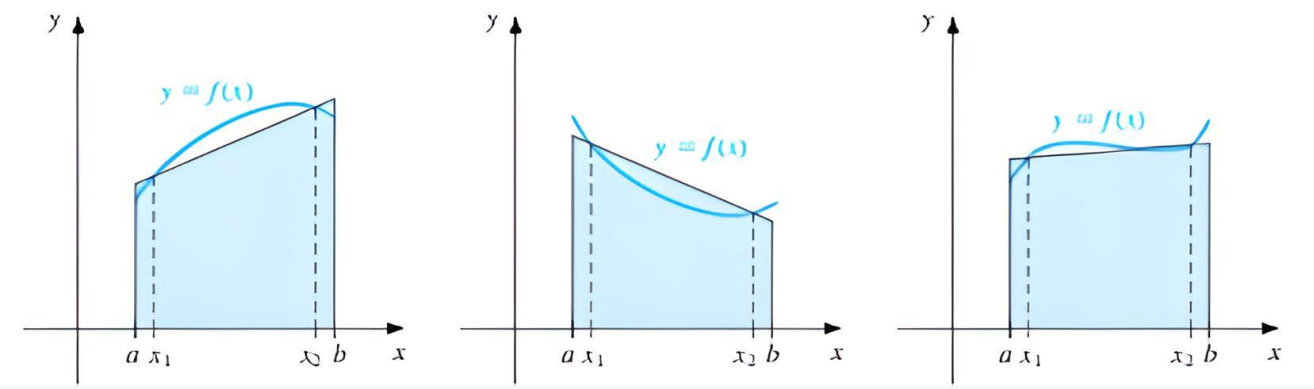
Integración por Cuadratura de Gauss

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

Por el trapecio sabemos que



Si flexibilizamos el trapecio, eligiendo 2 puntos $(x_0\ y\ x_1)$ y trazamos la recta entre a y b debajo de esta recta, compensando ligeramente los errores, obteniendo así un resultado más preciso (siempre y cuando la elección de $x_0\ y\ x_1$ sea buena)



Entonces, nuestro objetivo será elegir x_0 y x_1 de manera de ser lo más precisos posible. Para esto, debemos realizar la "fórmula de Gauss para 2 puntos".

➔ **Fórmula de Gauss-Legendre de dos puntos**

Consiste en aproximar la integral de la siguiente forma

$$I = \int_{-1}^{1} f(x)\,dx \simeq C_0 \cdot f(x_0) + C_1 \cdot f(x_1)$$

#Nota: Fórmula válida para integrales entre -1 y 1

¿Cómo determinamos c_0, c_1, x_0 y x_1? Para obtener dichos valores, le pedimos al método propuesto que ajuste de forma exacta para polinomios de orden máximo 3. Esto es,

$$\int_{-1}^{1} f_i(x)\,dx = resultado\ exacto$$

$$f_0(x) = 1 \Rightarrow \int_{-1}^{1} f_0(x)\,dx = x|_{-1}^{1} = 2 = c_0 + c_1$$

$$f_1(x) = x \Rightarrow \int_{-1}^{1} f_1(x)\,dx = 0 = c_0 x_0 + c_1 x_1$$

$$f_2(x) = x^2 \Rightarrow \int_{-1}^{1} f_2(x)\,dx = 2/3 = c_0 x_0^2 + c_1 x_1^2$$

$$f_3(x) = x^3 \Rightarrow \int_{-1}^{1} f_3(x)\,dx = 0 = c_0 x_0^3 + c_1 x_1^3$$

Formando un sistema no lineal de 4 ecuaciones con 4 incógnitas, que resolviendo da lo siguiente

$c_0 = c_1 = 1$

$x_0 = \frac{-1}{\sqrt{3}} = -0.5773503$

$x_1 = \frac{1}{\sqrt{3}} = \cdots$

Obteniendo así:

$$I = \int_{-1}^{1} f(x)\,dx \simeq f(\frac{-1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}})$$

Ejemplo:

$$I_{ex} = \int_{-1}^{1} e^x dx = e^x|_{-1}^{1} = e - \frac{1}{e} = 2.3504$$

$$I_{aprox} = \int_{-1}^{1} e^x dx = f(\frac{-1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}}) = e^{\frac{-1}{\sqrt{3}}} + e^{\frac{1}{\sqrt{3}}} = 2.3427$$

Viendo que el resultado es bastante cercano.

Ahora bien, ¿qué pasa si los límites de integración no son -1 y 1?

$$I = \int_{a}^{b} f(x)\,dx$$

Proponemos el siguiente cambio de variables:

$$x' = \alpha x + \beta$$

Siendo que, si $x = a \Rightarrow x' = -1$ y $x = b \Rightarrow x' = 1$

$$\int_{a}^{b} f(x)\,dx = \int_{-1}^{1} \bar{f}(x')\,dx'$$

$$-1 = \alpha a + \beta$$
$$1 = \alpha b + \beta$$

Restando ambas expresiones obtenemos:

$$2 = \alpha(b - a) \rightarrow \alpha = \frac{2}{b-a}$$

Reemplazamos alfa en la segunda expresión:

$$1 = \frac{2}{b-a} \cdot b + \beta \Rightarrow \beta = 1 - \frac{2b}{b-a} = \frac{b-a-2b}{b-a} = -\frac{b+a}{b-a}$$

Por lo que, el cambio de variables queda:

$$x' = \frac{2}{b-a}x - \frac{b+a}{b-a}$$

o bien

$$x = \frac{b-a}{2}(x' + \frac{b+a}{b-a})$$
$$\frac{b-a}{2}\,dx' = dx$$

Obteniendo así:

$$\int_a^b f(x)\,dx = \int_{-1}^{1} f(\frac{(b-a)x'+b+a}{2})\,\frac{(b-a)}{2}\,dx'$$

Con $\frac{(b-a)x'+b+a}{2} = \overline{f}(x')$

Por lo que podemos decir

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\int_{-1}^{1} f(\frac{(b-a)x'+b+a}{2})\,dx'$$

Ejemplo: $\int_0^{0.5} 2x^3 dx$

$$I_{ex} = \frac{2x^4}{4}|_0^{0.5} = 0.03125$$
$$I_{aprox} = \frac{0.5}{2}\int_{-1}^{1} 2(\frac{0.5x'+0.5}{2})^3 dx' = \frac{1}{4}(\overline{f}(\frac{1}{\sqrt{3}}) + \overline{f}(\frac{-1}{\sqrt{3}})) = 0.03125$$

➔ **Fórmula de Gauss-Legendre con más puntos**

Consiste en el mismo concepto que el anterior

$$I = \int_{-1}^{1} f(x)\,dx = c_0 f(x_0) + c_1 f(x_1) + \cdots + c_{n-1} f(x_{n-1})$$

Con $n = número\ de\ puntos\ que\ usamos$

Para calcular los valores de las c's y x's se determinan pidiendo que el método ajuste de forma exacta polinomios de grado máximo 2n-1. Obteniendo de tabla para hasta máximo 6 puntos

Tabla 22.1 Factores de ponderación c y argumentos de la función x usados en las fórmulas de Gauss-Legendre.

Puntos	Factor de ponderación	Argumentos de la función	Error de truncamiento
2	$c_0 = 1.0000000$	$x_0 = -0.577350269$	$\cong f^{(4)}(\xi)$
	$c_1 = 1.0000000$	$x_1 = 0.577350269$	
3	$c_0 = 0.5555556$	$x_0 = -0.774596669$	$\cong f^{(6)}(\xi)$
	$c_1 = 0.8888889$	$x_1 = 0.0$	
	$c_2 = 0.5555556$	$x_2 = 0.774596669$	
4	$c_0 = 0.3478548$	$x_0 = -0.861136312$	$\cong f^{(8)}(\xi)$
	$c_1 = 0.6521452$	$x_1 = -0.339981044$	
	$c_2 = 0.6521452$	$x_2 = 0.339981044$	
	$c_3 = 0.3478548$	$x_3 = 0.861136312$	
5	$c_0 = 0.2369269$	$x_0 = -0.906179846$	$\cong f^{(10)}(\xi)$
	$c_1 = 0.4786287$	$x_1 = -0.538469310$	
	$c_2 = 0.5688889$	$x_2 = 0.0$	
	$c_3 = 0.4786287$	$x_3 = 0.538469310$	
	$c_4 = 0.2369269$	$x_4 = 0.906179846$	
6	$c_0 = 0.1713245$	$x_0 = -0.932469514$	$\cong f^{(12)}(\xi)$
	$c_1 = 0.3607616$	$x_1 = -0.661209386$	
	$c_2 = 0.4679139$	$x_2 = -0.238619186$	
	$c_3 = 0.4679139$	$x_3 = 0.238619186$	
	$c_4 = 0.3607616$	$x_4 = 0.661209386$	
	$c_5 = 0.1713245$	$x_5 = 0.932469514$	

-

Algoritmo

1. Definir la función f(x)
2. Ingresar a y b
3. Calcular la integral (siempre teniendo en cuenta el cambio de variables)

```
printf("Ingrese la cantidad de puntos a usar (entre 2 y 6)")
scanf("%d",&puntos)
switch(puntos)
        case 2:
                c0 = 1
                c1 = 1
                x0 = -1/sqrt(3)
                x1 = 1/sqrt(3)
                I = (b-a)/2 * (c0*f((b-a)*x0+b+a)/2)+c1*f((b-a)*x1+b+a)/2
```

```
        break
case 3:
        c0=..
        c1=..
        c2=..
        x0=..
                .
                .
        I = …..
        .
        .
        .
case 6:
                .
                .

default:
        printf(“Cantidad de puntos entre 2 y 6)
```

4. Mostrar I

19/10

Diferenciación Numérica

Recordemos: Teorema de Taylor

Si la función f y sus primeras $p + 1$ derivadas son continuas en un intervalo que contiene a a y x, entonces:

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2}(x - a)^2 + \frac{f''(a)}{3!}(x - a)^3 + \cdots + \frac{f^{'(p)}(a)(x-a)}{p!} + R_p(x)$$

$$R_p(x) = \frac{f^{(p+1)}(\xi)}{(p+1)!}(x - a)^{p+1} = Residuo\ o\ Resto\ de\ Orden\ p \qquad \xi \in [a, x]$$

Operadores en Diferencias Finitas

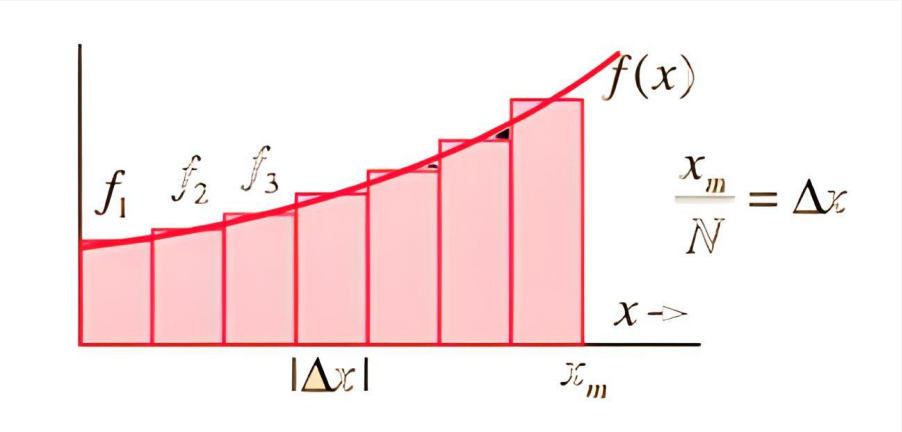
$$f'(p) = \lim_{h \to 0} \frac{f(x+h)-f(x)}{h} \simeq \frac{f(x+h)-f(x)}{h}\ con\ h\ pequeño$$

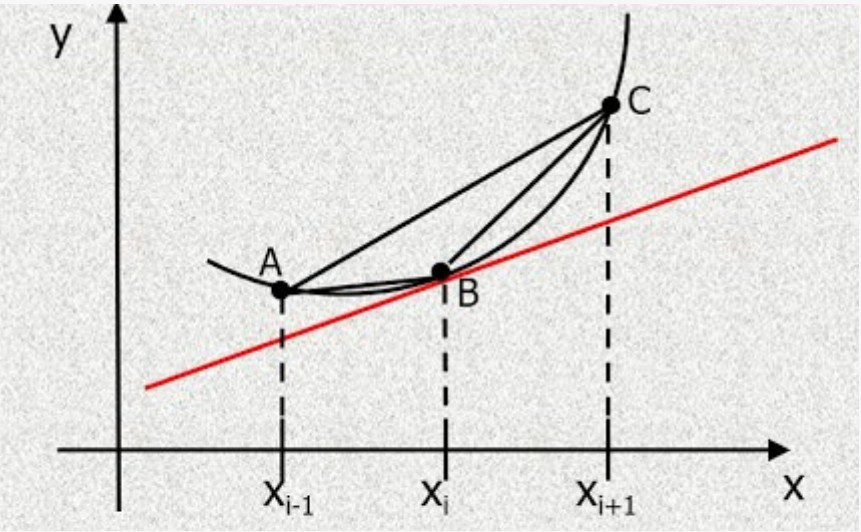
$$h = \frac{x_f - x_0}{n}$$

$$f'(x_i) \simeq \frac{f(x_i+h)-f(x_i)}{h} = \frac{f(x_{i+1})-f(x_i)}{h}$$

#Nota: Operador en diferencias finitas hacia adelante. Puedo calcularlo desde $[x_0, x_{x-1}]$

La idea es aproximar f'(x), donde f(x) es una función definida en un intervalo $[x_0, x_{final}]$ mediante una discretización del intervalo





$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2}(x_{i+1} - x_i)^2 + \frac{f''(x_i)}{3!}(x_{i+1} - x_i)^3 + O(x_{i+1} - x_i)^4$$

$$f(x_{i+1}) - f(x_i) = \frac{f'(x_i)h}{h} + \frac{f''(x_i)h}{2} \cdot \frac{h^2}{h} + \frac{f'''(x_i)h}{3} \cdot \frac{h^3}{h}$$

$$f'(x_i)_{aproximada} = f'(x_i)_{exacta} + ch$$

con un error de orden h, es decir, una constante por h

Ejemplo: $f(x) = ln(x) + \frac{1}{x}$ tomando $x_0 = 1.5$, $x_1 = 2$, $x_2 = 2.5$. Calcular $f'(1.5)$ con un operador de diferencias finitas hacia adelante, y calcular el error

$h = 0.5$

$f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_i)}{h}$

$f'(1.5) \simeq \frac{f(2) - f(1.5)}{0.5} \simeq \frac{1.1931 - 1.0721}{0.5} \simeq 0.24$

$f'_{exacta} = \frac{1}{x} - \frac{1}{x^2}|_{1.5} = 0.22$

$e = |f'_{exacta} - f'_{aprox}| = 0.02$

$e \simeq \frac{f''(x_i)}{2!} h \simeq \frac{0.15}{2} * 0.5 = 0.03$

Vemos que el aproximado y el exacto son del mismo orden.

Sin embargo, seguimos con el problema de no poder calcular para x_n. Utilizamos el siguiente método:

Esquema en Diferencias Finitas hacia atrás

$f(x_{i-1}) = f(x_i) + f'(x_i)(x_{i-1} - x_i) + \frac{f''(x_i)}{2}(x_{i-1} - x_i)^2 + O(x_{i-1} - x_i)^3$

$f(x_{i-1}) = f(x_i) + f'(x_i)(-h) + \frac{f''(x_i)}{2}(-h)^2 + O(-h)^3$

$-\frac{1}{h}[f(x_{i-1}) - f(x_i) = f'(x_i)(-h) + \frac{f''(x_i)}{2}(-h)^2 + O(-h)^3]$

$\frac{f(x_i) - f(x_{i-1})}{h} = f'(x_i) - \frac{f''(x_i)}{2} \cdot h + O(h^2)$

$f'(x_i) \simeq \frac{f(x_i) - f(x_{i-1})}{h} + O(h)$

#Nota: Esquema en diferencias finitas hacia atrás. No puedo calcular para x_0

En nuestro ejemplo: $f(x) = ln(x) + \frac{1}{x}$ tomando $x_0 = 1.5$, $x_1 = 2$, $x_2 = 2.5$

$f'(x_i) \simeq \frac{f(x_i) - f(x_{i-1})}{h}$

$$f'(x_1) \simeq \frac{f(x_1)-f(x_0)}{0.5} \simeq 0.24$$

Esquema de Diferencias Finitas Centradas

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2}h^2 + \frac{f''(x_i)}{3!}h^3 + O(x_{i+1} - x_i)^4$$
$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2}h^2 - \frac{f''(x_i)}{3!}h^3 + O(x_{i+1} - x_i)^4$$

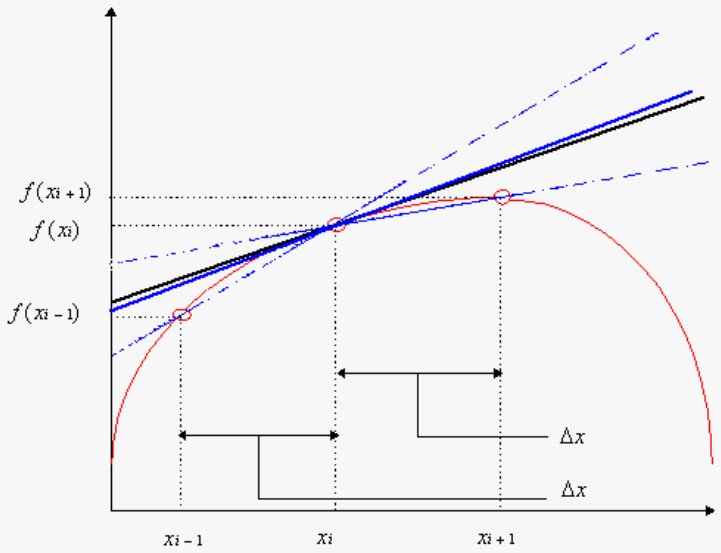
Restando ambas expresiones

$$f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)h + 2\frac{f''(x_i)}{3!}h^3$$

$$\frac{f(x_{i+1})-f(x_i)}{2h} = f'(x_i) + \frac{f''(x_i)}{3!}h^2$$

Duplicando la exactitud del error (error de orden h^2)

#Nota: Esquema de Diferencias Finitas Centrado. No puedo usar este esquema en los valores extremos (x_0 y x_1). Son más precisas que los esquemas hacia adelante y hacia atrás.



Algoritmo

- Definir f(x)
- Ingresar x_0 y x_{final}

- Ingreso h o n (mejor n) ($h = \frac{x_f - x_0}{n}$)
- Calcular f'(x_i)

Esquema hacia adelante

Do i = 0 , …… , $n - 1$

$x_i = x_0 + ih$

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

print f' en un archivo

Esquema hacia atrás

Do i = 1 , …… , n

$x_i = x_0 + ih$

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

print f' en un archivo

Esquema Centrado

Do i = 1 , …… , $n - 1$

$x_i = x_0 + ih$

$f'(x_i) = \frac{f(x_i+h) - f(x_i)}{2h}$

print f' en un archivo

Fórmulas de Diferenciación de Mayor Orden

La idea es usar más puntos y/o conservar más términos en la serie de Taylor para tener derivadas de orden más alto, o la derivada primera con mayor precisión.

Por ejemplo:

1) $f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2}h^2 + \frac{f'''(x_i)}{3!}h^3$

2) $f(x_{i+2}) = f(x_i) + f'(x_i)(x_{i+2} - x_i) + f''(x_i)(x_{i+2} - x_i)^2 + f'''(x_{i+2} - x_i)^3 \Rightarrow pero\ (x_{i+2} - x_i) = 2h$

$= f(x_i) + 2f'(x_i)h + 4\frac{f''(x_i)}{2!}h^2 + 8\frac{f'''(x_i)}{3!}h^3$

Si multiplicamos 1) por 4 y le restamos 2)

$4f(x_{i+1}) - f(x_{i+2}) = 3 \cdot f(x_i) + 2 \cdot f'(x_i) \cdot h - 4 \cdot \frac{f'''(x_i)}{3!}h^3$

$4f(x_{i+1}) - f(x_{i+2}) - 3 \cdot f(x_i) = 2 \cdot f'(x_i) \cdot h - 4 \cdot \frac{f'''(x_i)}{3!}h^3$

$$\frac{4f(x_{i+1})-f(x_{i+2})-3\cdot f(x_i)}{2h} = f'(x_i) + O(h^2)$$

Si multiplicamos 1) por 2 y le restamos 2)

$$2f(x_{i+1}) - f(x_{i+2}) = f(x_i) - 2\cdot\frac{f''(x_i)}{2!}\cdot h^2 - 6\cdot\frac{f'''(x_i)}{3!}\cdot h^3$$

$$-\frac{1}{h^2}[2f(x_{i+1}) - f(x_{i+2}) - f(x_i) = - f''(x_i)\cdot h^2 + O(h^3)]$$

$$\frac{f(x_i)+f(x_{i+2})-2f(x_{i+1})}{h^2} = f''(x_i) + O(h)$$

Ver más en Formulas de Diferenciacion.pdf

26/10

Ecuaciones Diferenciales Ordinarias

Una ecuación que involucra una función desconocida, y sus derivadas, se denomina ecuación diferencial.
Si dicha función depende de una sola variable y la ecuación involucra las derivadas hasta un orden p entonces se llama “Ecuación Diferencial Ordinaria de grado p”

Ejemplo: $m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0 \Rightarrow EDO\ de\ grado\ 2$
incógnita: $x(t)\ donde\ x\ es\ la\ variable\ dependiente\ y\ t\ la\ independiente$

⇒ Para una EDO de grado p, debemos dar p condiciones iniciales:

$$x(t_0) \qquad x'(t_0) \ldots\ldots x^{p-1}(t_0)$$

Ejemplo: Un cuerpo de 2kg de masa, se mueve dentro de un fluido experimentando una fuerza de resistencia dada por $F = k\cdot v$ donde v es la velocidad del cuerpo. Si el cuerpo tiene una $v_0 = 20\ m/s$ y una posición inicial de 5m. Calcular la posición x(t) con $k = 5\ \frac{Ns}{m}$:

Solución:
$F = m.a$
$v = \frac{dx}{dt}$
$a = \frac{d^2x}{dt^2}$

$$-k.v = m.a \Rightarrow -k.\frac{dx}{dt} = m.\frac{d^2x}{dt^2} \Rightarrow -k\cdot v = m\frac{dv}{dt} \Rightarrow \int -kdt = \int m\frac{dv}{v}$$

$$-kt + c = m \cdot ln|v| + c'$$

$$-kt = m \cdot ln|v| + c'' \Rightarrow para\ v(t = 0) \Rightarrow 0 = 2\,ln\,(20) + c'' \Rightarrow -2\,ln\,(20) = c''$$

$$-5t = 2\,ln\,(v) - 2\,ln\,(20) \Rightarrow -5t = 2(ln\,(v) - ln\,(20))$$

$$-\frac{5}{2} \cdot t = ln\,(\frac{v}{20}) \rightarrow e^{-\frac{5}{2}t} = \frac{v}{20} \Rightarrow v(t) = 20 \cdot e^{-\frac{5}{2}t} = \frac{dx}{dt}$$

$$\frac{dx}{dt} = 20e^{-\frac{5}{2}t} \Rightarrow dx = 20e^{-\frac{5}{2}t}dt$$

$$\int dx = \int 20 \cdot e^{-\frac{5}{2}t}$$

$$x = 20 \cdot e^{-\frac{5}{2}t} \cdot (-\frac{1}{\frac{5}{2}}) + c \Rightarrow 5 = 4 \cdot (-\frac{2}{5}) + c \Rightarrow 13 = c$$

$$x(t) = -8 \cdot e^{-\frac{5}{2}t} + 13$$

Ejemplo 3:

$$\frac{dy}{dx} + 2xy = 0 \Rightarrow\ y\ es\ dependiente,\ x\ es\ \ independiente,\ y(0) = 1$$

$$Queremos\ saber\ y(x)$$

$$\frac{dy}{dx} = -2xy \Rightarrow \frac{dy}{y} = -2x\,dx \Rightarrow integrando \Rightarrow ln(y) = -x^2 + c \Rightarrow y(0) \Rightarrow ln(1) = c = 0$$

$$ln(y) = -x^2 \Rightarrow y = e^{-x^2}$$

Nuestra idea ahora es ver métodos numéricos para resolver EDO's de primer orden
Supongamos una ecuación de la forma:

$$\frac{dy}{dx} = f(x, y)$$

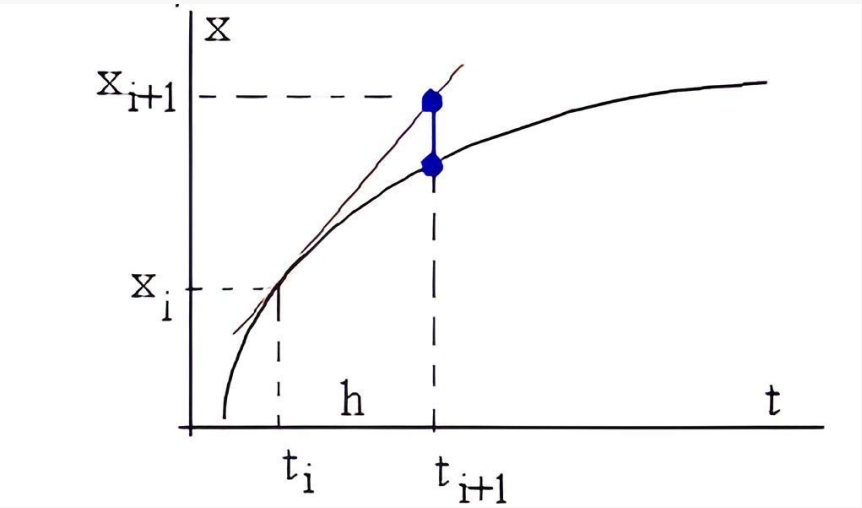
y supongamos que queremos encontrar la solución en un intervalo $[x_0, x_f]$ con un dato inicial $y(x_0) = y_0$

Método de Euler

Como todo método numérico, lo que vamos a hacer como primer paso es discretizar el intervalo (dividirlo) en n subintervalos, donde la distancia entre punto y punto va a estar dada por

$$h = \frac{x_f - x_0}{n}$$

La idea es obtener la solución aproximada en x_{i+1} sabiendo la solución en x_i



La recta tangente tiene una pendiente

$$y'(x_i) = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \Rightarrow Siendo\ \frac{dy}{dx} = f(x,y)\ \Rightarrow\ f(x_i, y_i) = \frac{y_{i+1} - y_i}{h}$$

$$h \cdot f(x_i, y_i) = y_{i+1} - y_i$$

$$y_{i+1} = y_i + h \cdot f(x_i, y_i) \Rightarrow \textbf{Método de Euler}$$

Ejemplo: Integrar usando el método de euler la EDO del ejemplo 3, en [0,1] con h=0.2.
Calcular $y_{real}(1)$, $y_{aprox}(1) \Rightarrow error(1)$?

$$\frac{dy}{dx} = -2xy \qquad\qquad y(0) = 1$$

Con $f(x,y) = -2xy$, $0 = x_o$, $1 = y_0$

$$x_1 = 0.2$$
$$y_1 = y_0 + h \cdot f(x_0, y_0) = 1 + 0.2 * (-2 \cdot 0 \cdot 1) = 1$$

$$x_2 = 0.4$$
$$y_2 = y_1 + h \cdot f(x_1, y_1) = 1 + 0.2 * (-2 \cdot 0.2 \cdot 1) = 0.92$$

$x_3 = 0.6$

$y_3 = y_2 + h \cdot f(x_2, y_2) = 0.92 + 0.2 * (-2 \cdot 0.4 \cdot 0.92) = 0.7728$

.

.

.

$x_5 = 1$

$y_5 = y_4 + h \cdot f(x_4, y_4) = 0.399383 \Rightarrow y_{aprox}(1)$

$y_{real}(1) = e^{-x^2}|_1 = \frac{1}{e} = 0.367879 \Rightarrow y_{real}(1)$

$error = |0.399383 - 0.367879| = 0.0315$

En la mayoría de los casos no vamos a conocer la función real, por lo que vamos a estimar el error con lo siguiente:

Error del Metodo de Euler

$y_{i+1} = y(x_i + h) = y(x_i) + y'(x_i) \cdot h + \frac{y''(x_i) \cdot h^2}{2} + \cdots$

$= y_i + f(x_i, y_i) \cdot h + \frac{f'(x_i, y_i)}{2} \cdot h^2 + O(h^3)$

donde a partir de $\frac{f'(x_i, y_i)}{2} \cdot h^2 + O(h^3)$ se denomina error de truncamiento

si h es pequeño, los términos en la serie de taylor decaen muy rápido, por lo tanto:

$e_{i+1} \simeq \frac{f'(x_i, y_i)}{2} \cdot h^2$

Es decir, el error es de orden (estimado) h^2

Algoritmo

- Definir f(x,y)
- Ingresar x_0, x_f
- Ingresar y_0
- Ingresar n ⇒ $h = \frac{x_f - x_0}{n}$
- Calcular la solución

Do i = 0, ….., n-1

$y_{i+1} = y_i + h \cdot f(x_i, y_i)$

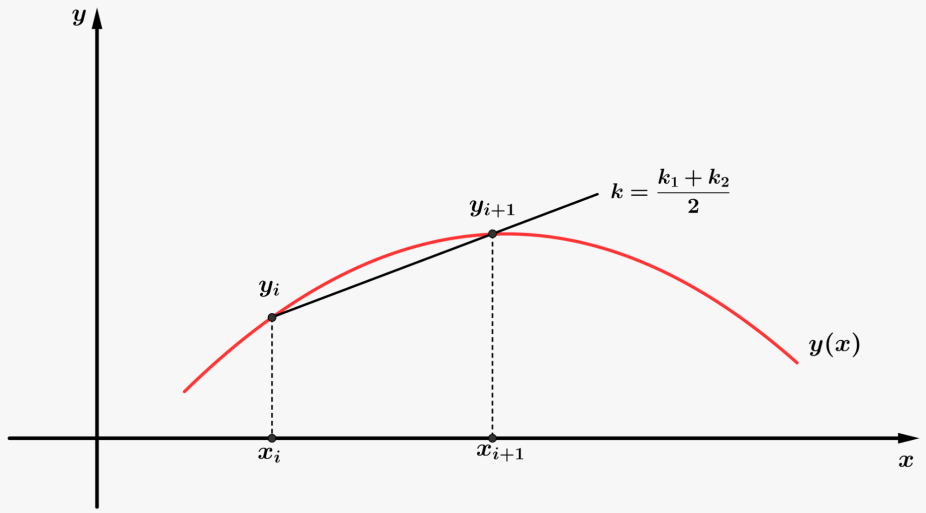
$x_{i+1} = x_i + h$

print en archivo

end do

Algunas mejoras al Metodo de Euler

Método de Heun



$pendiente \simeq \frac{y'(x_i)+y'(x_{i+1})}{2} = \frac{f(x_i, y_i)+f(x_{i+1}, y_{i+1})}{2}$

$y_{i+1} = y_i + h \cdot pendiente$

El problema es que no tenemos y_{i+1}. Para resolver este problema, vamos a estimar dicho valor como

$\hat{y}_{i+1} = y_i + h \cdot f(x_i, y_i) \Leftarrow predictor$

$y_{i+1} = y_i + h \cdot \frac{f(x_i, y_i)+f(x_{i+1}, \hat{y}_{i+1})}{2} \Rightarrow$ **Método de Heun**

Algoritmo

- ➢ Definir f(x,y)
- ➢ Ingresar x_0, x_f

- ➢ Ingresar y_0
- ➢ Ingresar n ⇒ $h = \frac{x_f - x_0}{n}$
- ➢ Calcular la solución

Do i = 0, ….., n-1
$x_{i+1} = x_i + h$
$y_t = y_i + h * f(x_i, y_i)$
$y_{i+1} = y_i + h \cdot \frac{f(x_i, y_i) + f(x_{i+1}, y_t)}{2}$
print en archivo
end do

Con un error del orden h^3

Método del Punto Medio

$y_{i+1} = y_i + h \cdot pendiente$

$pendiente = f(x_m, y_m)$

$Por\ Euler \Rightarrow y_m = y_i + \frac{h}{2} f(x_i, y_i)$
$x_m = x_i + \frac{h}{2}$
$y_{i+1} = y_i + h \cdot f(x_m, y_m)$

Algoritmo

- ➢ Definir f(x,y)
- ➢ Ingresar x_0, x_f
- ➢ Ingresar y_0
- ➢ Ingresar n ⇒ $h = \frac{x_f - x_0}{n}$
- ➢ Calcular la solución

Do i = 0, ….., n-1
$x_m = x_i + h/2$
$y_m = y_i + h/2 * f(x_i, y_i)$
$y_{i+1} = y_i + h \cdot f(x_m, y_m)$
$x_{i+1} = x_i + h$
print en archivo

end do

31/10

Métodos de Runge-Kutta

Existen varias variantes, pero todas tienen la forma general:

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

$\varphi(x_i, y_i, h)$ se denomina función incremental. Según sea $\varphi(x_i, y_i, h)$ es el método que se utiliza (Euler usa $\varphi(x_i, y_i, h) = f(x,y)$).
Esta función tiene la siguiente forma general:

$$\varphi = a_1k_1 + a_2k_2 + \ \ldots + a_nk_n$$
$$a_i = ctes$$
$$k_i = f(x_i, y_i)$$
$$k_2 = f(x_i + p_1h, y_i + q_{11}k_1h)$$
$$k_3 = f(x_i + p_2h, y_i + q_{21}k_1h + q_{22}k_2h)$$
.
.
.
$k_n = f(x_i + p_{n-1}h, y_i + q_{n-1,1}k_1h + q_{n-2,2}k_2h + \ldots\ldots + q_{n-1,n-1}k_{n-1}h)$ ⇒ **Método de RK de orden n**
$a_i, p_i, q_{ij} = ctes$
#Nota:

Euler: $y_{i+1} = y_i + h\,f(x_i, y_i) \quad con\ n = 1 \ \ a_1 = 1$
$RK \Rightarrow y_{i+1} = y_i + h\varphi(\ldots)$
$= y_i + ha_ik_i$
$= y_i + h.1.f(x_i, y_i)$

Para obtener todas las constantes a_i, p_i, q_{ij}, igualamos el método descrito (RK) al desarrollo de la serie de Taylor de y(x) y se comparan los términos de igual orden en h

Ejemplo: RK2 (Runge-Kutta de orden 2)

$$y_{i+1} = y_i + h(a_1k_1 + a_2k_2)$$

$k_1 = f(x_i, y_i)$

$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h)$

donde tenemos a a_1, a_2, p_1, q_{11} como incógnitas.

1) Serie de Taylor

$y(x_i + h) = y(x_i) + y'(x_i) + \frac{y''(x_i)h^2}{2} + O(h^3)$

$y_{i+1} = y_i + f(x_i, y_i)\, h + \frac{f'(x_i, y_i)}{2} h^2 + O(h^3)$

$f'(x,y) = \frac{\partial f}{\partial x}\frac{\partial x}{\partial x} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} f(x,y)$

$y_{i+1} = y_i + f(x_i, y_i)\, h + \frac{1}{2}[\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} f(x_i, y_i)]\, h^2 + O(h^3)$ (1)

2) Método

Recordemos el desarrollo de Taylor para dos variables.

Sea $g(x + r, y + s) = g(x, y) + \frac{\partial g}{\partial x} r + \frac{\partial g}{\partial y} s$ +.....

Si aplicamos esto a k_2

$k_2 = f(x_i, y_i) + \frac{\partial f}{\partial x} p_1 h + \frac{\partial f}{\partial y} q_{11} k_1 h +$ ...

Por lo tanto, el método queda

$y_{i+1} = y_i + h\,[a_1 f(x_i, y_i) + a_2 (f(x_i, y_i) + \frac{\partial f}{\partial x} p_1 h + \frac{\partial f}{\partial y} q_{11} k_1 h)]$ +....

$y_{i+1} = y_i + h\, f(x_i, y_i)(a_1 + a_2) + h^2 (a_2 \frac{\partial f}{\partial x} p_1 + a_2 \frac{\partial f}{\partial y} q_{11} k_1)$

$y_{i+1} = y_i + h(a_1 + a_2)\, f(x_i, y_i) + h^2 (a_2 \frac{\partial f}{\partial x} p_1 + \frac{\partial f}{\partial y} f(x_i, y_i)\, q_{11} a_2)$ (2)

Igualando 1 con 2, término a término, obtenemos que:

$a_1 + a_2 = 1$

$a_2 p_1 = \frac{1}{2}$

$q_{11} = \frac{1}{2}$

Nos genera un sistema de ecuaciones con infinitas soluciones

$a_i = 1 - a_2$, $p_1 = \frac{1}{2a_2}$, $q_{11} = \frac{1}{2a_2}$ $con\ a_2 = parámetro\ libre$

Para distintas elecciones de a_2 obtenemos distintos métodos.

Supongamos $a_2 = 1/2$

$a_1 = a_2 = 1/2$

$p_1 = q_1 = 1$

$$y_{i+1} = y_i + h(a_1 k_1 + a_2 k_2)$$
$$= y_i + h\left[\frac{1}{2} f(x_i, y_i) + \frac{1}{2} f(x_i + h, y_i + h f(x_i, y_i))\right]$$
$$= y_i + \frac{h}{2} f(x_i, y_i) + f(x_i + h, y_i + h f(x_i, y_i)) \Rightarrow \textit{Método de Heun}$$
$$predictor = \bar{y} = y_i + h f(x_i, y_i)$$

Ahora, si $a_2 = 1$

$a_1 = 0$

$p_1 = q_1 = 1/2$

$$y_{i+1} = y_i + h a_2 k_2$$
$$= y_i + h f(x_i + h, y_i + \frac{k_1 h}{2})$$
$$= y_i + h f(x_i + h, y_i + \frac{h}{2} f(x_i, y_i)) \Rightarrow \textit{Método de Punto Medio}$$

Método RK4

Al igual que en el caso de RK2, hay infinitas soluciones para los a, p, q. La más usada, que se denomina "método de RK4 clásico" está dado por:

$$RK4 \Rightarrow y_{i+1} = y_i + h(a_1 k_1 + a_2 k_2 + a_3 k_3 + a_4 k_4)$$
$$RK4\ clásico \Rightarrow y_{i+1} = y_i + \frac{1}{6} h(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(x_i, y_i)$$
$$k_2 = f(x_i + h, y_i + \frac{1}{2} k_1 h)$$
$$k_3 = f(x_i + \frac{h}{2}, y_i + \frac{1}{2} k_2 h)$$
$$k_4 = f(x_i + h, y_i + k_3 h)$$

Ejercicio: Resolver el siguiente problema

$$\frac{\partial y}{\partial x} - (2x + 1)\sqrt{y} = 0 \quad con\ y_0 = 1\ y\ x \in [0, 1]\ y\ h = 0.1$$

- ➢ Analiticamente
- ➢ Con Euler
- ➢ Con Heun y Punto Medio
- ➢ Con RK4

Graficar en un mismo plot todos los resultados

$$\frac{\partial y}{\partial x} = (2x + 1)\sqrt{y} \Rightarrow \frac{\partial y}{\sqrt{y}} = (2x + 1)dx \Rightarrow \int \frac{\partial y}{\sqrt{y}} = \int (2x + 1)dx$$

$$2\sqrt{y} = x^2 + x + C$$
$$\sqrt{y} = \frac{1}{2}(x^2 + x + C)$$
$$y = \frac{1}{4}(x^2 + x + C)^2$$

$$y(0) = 1 = \frac{C^2}{4}$$
$$4 = C^2$$
$$\pm 2 = C^2$$

$\Rightarrow y = \frac{1}{4}(x^2 + x \pm 2)^2$ ⇒ El método va a calcular con +C

Factor de Convergencia

Si decimos que un método converge con orden n, significa que el error es de orden h^n.

$\Rightarrow y_{aprox} = y_{exacto} + c.h^n$

Calculemos $y_1 = solución\ aproximada\ con\ un\ valor\ de\ h\ (\sim 0.01)$
Calculemos $y_2 = solución\ aproximada\ con\ un\ valor\ de\ h/2$
Calculemos $y_3 = solución\ aproximada\ con\ un\ valor\ de\ h/4$

$y_1 = y_{exacto} + c.h^n$
$y_2 = y_{exacto} + c.(\frac{h}{2})^n$
$y_3 = y_{exacto} + c.(\frac{h}{4})^n$

Definimos el “factor de convergencia” como:

$$Q = \frac{1}{ln(2)} ln\left|\frac{y_1 - y_2}{y_2 - y_3}\right| \simeq n$$

Si este coeficiente se aproxima a n, podemos decir que el error converge al valor deseado.

Algoritmo (para Euler)

I. Definir y[n+1], y2[n+1], y3[n+1], x[n+1]
II. Calcular y1

```
Do i = 0, ....., n-1
        y[i+1] = y[i] + h f(x[i], y[i])
```

x[i+1] = x[i] + h

III. Calcular y2

Do i = 0, ….., n-1
yp = y2[i] + h/2 f(x[i], y2[i])
xp = x[i] + h/2
y2[i+1] = yp + h/2 f(xp, yp)

IV. Calcular y3

Do i = 0, ….., n-1
yp = y3[i] + h/4 f(x[i], y[i])
xp = x[i] + h/4
yp = yp + h/4 f(xp, yp)
xp = xp + h/4
yp = yp + h/4 f(xp, yp)
xp = xp + h/4
y3[i+1] = yp + h/4 f(xp, yp)

V. Calcular Q

Do i = 0, …., n

$$Q[i] = \frac{1}{ln(2)} ln\left|\frac{y_1[i]-y_2[i]}{y_2[i]-y_3[i]}\right|$$

VI. Plotear Q(i) vs x(i)