Modelización y Simulación Computacional de Materiales
Clase 2

Brevísimo Repaso de cálculo numérico 1

Repaso de cálculo numérico

- Aproximaciones y errores
- Raíces de ecuaciones
- Ecuaciones algebraicas lineales
- Optimización
- Ajuste de funciones e interpolación
- Diferenciación e integración numérica
- Ecuaciones diferenciales der. ordinarias
- Ecuaciones diferenciales der. parciales

Errores

Siempre en un cálculo numérico hay un error

Debemos intentar reducirlo o, al menos, conocerlo

Hay dos fuentes de error típicos:

- **Error de redondeo**
- **Error de truncamiento**

Error de redondeo:

- **Se debe a que una computadora puede operar con una cantidad finita de dígitos**

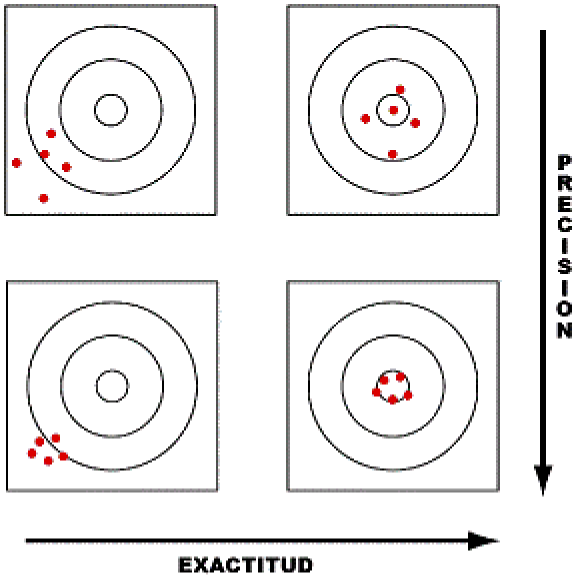
Error de truncamiento:

- **Representa la diferencia entre la formulación matemática exacta y una aproximación dada por un método numérico (por ej. basado en Taylor)**

Exactitud y precisión

Exactitud: qué tan cercano está un valor calculado o medido al valor verdadero.

Precisión: qué tan cercano está un valor individual medido o calculado con respecto a los otros valores.



Errores

Error Absoluto $\varepsilon_{abs} = Aproximación \text{ - } Valor\ Verdadero$

Error relativo $\varepsilon_{rel} = \dfrac{\varepsilon_{abs}}{Valor\ Verdadero}\ (100\%)$

Pero que pasa cuando no sé el valor verdadero o para métodos iterativos?

$$\varepsilon_{rel} = \frac{(Aprox.\ \ Anterior - Aprox.\ \ Actual)}{Valor\ \ Actual}\ (100\%)$$

Recordar **Validar y Verificar !!**

Sistemas de Ecuaciones

Muchos problemas se pueden llevar a hallar valores x_1, x_2, ..., x_N que satisfagan, en forma simultánea, un conjunto de ecuaciones:

$$
\begin{aligned}
f_1(x_1, x_2, \ldots, x_N) &= 0 \\
f_2(x_1, x_2, \ldots, x_N) &= 0 \\
\vdots \qquad &\quad \vdots \\
f_N(x_1, x_2, \ldots, x_N) &= 0
\end{aligned}
$$

estas ecuaciones pueden ser lineales o no-lineales.

Sistemas de Ecuaciones algebraicas lineales

Los sistemas lineales, donde las variables presentes no están elevadas a ninguna potencia, son los más fáciles de tratar y también los más comunes.

(incluso, muchos sistemas no-lineales se resuelven linealizándolos localmente)

$$\begin{aligned} A_{11}\,x_1 + A_{12}\,x_2 + \ldots + A_{1N}\,x_N &= b_1 \\ A_{21}\,x_1 + A_{22}\,x_2 + \ldots + A_{2N}\,x_N &= b_2 \\ \vdots \qquad\qquad\qquad\qquad & \quad \vdots \\ A_{N1}\,x_1 + A_{N2}\,x_2 + \ldots + A_{NN}\,x_N &= b_N \end{aligned}$$

Se puede escribir matricialmente como: $[A].\{x\} = \{b\}$

Donde pusimos, como siempre:

$$[A] = \begin{pmatrix} A_{11} & \cdots & A_{1N} \\ \vdots & \ddots & \vdots \\ A_{N1} & \cdots & A_{NN} \end{pmatrix}, \quad \{x\} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}, \quad \{b\} = \begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix}$$

Ejemplo:

$$\begin{cases} 2x_1 + 3x_2 - x_3 = 5 \\ 4x_1 + 4x_2 - 3x_3 = 3 \\ 2x_1 - 3x_2 + x_3 = -1 \end{cases} \qquad \begin{pmatrix} 2 & 3 & -1 \\ 4 & 4 & -3 \\ 2 & -3 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ -1 \end{pmatrix}$$

Ecuaciones algebraicas lineales

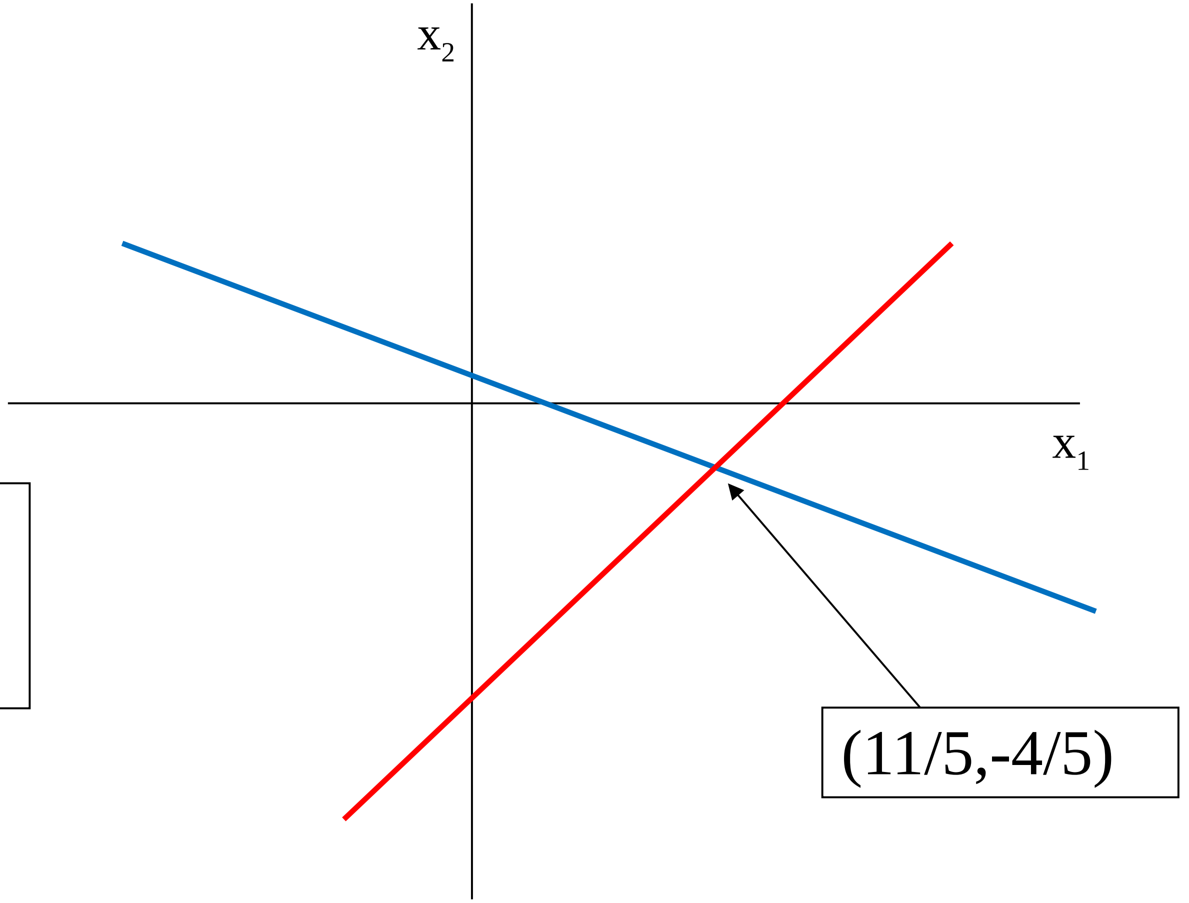
Ejemplo para dos ecuaciones

$$2x_1+3x_2=2$$

$$x_1-x_2=3$$

$$x_2=2/3-(2/3)x_1$$

$$x_2=x_1-3$$



Ecuaciones algebraicas lineales

El sistema tendrá solución única si $\det(A) \neq 0$

O sea si las filas y las columnas de [A] son linealmente independientes

Si, por otro lado $\det(A) = 0$

Pueden haber infinitas soluciones o ninguna

Una representación útil es definir la matriz aumentada:

$$[A \mid b] = \left(\begin{array}{ccc|c} A_{11} & \cdots & A_{1N} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ A_{N1} & \cdots & A_{NN} & b_N \end{array} \right)$$

Teorema de Rouché–Frobenius

Rango de una matriz: número máximo de columnas (filas) linealmente independientes

$$rg[A] = rg[A \mid b] = n \quad \text{existe solución y es única}$$

$$rg[A] = rg[A \mid b] < n \quad \text{existen muchas soluciones}$$

$$rg[A] \neq rg[A \mid b] \quad \text{no existe solución}$$

Ecuaciones algebraicas lineales

Formalmente: $\text{si } \det(A) \neq 0 \Rightarrow \{x\} = [A]^{-1}\{b\}$

Si el sistema es chico: Regla de Cramer

$$x_i = \frac{\det\begin{bmatrix} A_{1,i-1} & b_1 & A_{1,i+1} \\ A_{2,i-1} & b_2 & A_{2,i+1} \\ A_{3,i-1} & b_3 & A_{3,i+1} \end{bmatrix}}{\det[A]}$$

Pero esto en general es un método muy caro.

Ecuaciones algebraicas lineales

Métodos de solución numéricos

- Directos
- Indirectos

Métodos Directos

Eliminación de Gauss: $[A]\{x\} = \{b\} \Rightarrow [U]\{x\} = \{c\}$

Descomposición LU: $[A]\{x\} = \{b\} \Rightarrow [L][U]\{x\} = \{b\}$

Gauss-Jordan: $[A]\{x\} = \{b\} \Rightarrow [I]\{x\} = \{c\}$

Eliminación de Gauss simple

Es uno de las métodos más antiguos de resolución de ecuaciones lineales, pero continúa siendo uno de los más importantes. Veamos un caso simple.

$$a_{11}\ x_1 + a_{12}\ x_2 = b_1$$
$$a_{21}\ x_1 + a_{22}\ x_2 = b_2$$

Si a la primer ecuación la multiplico por a_{21}/a_{11} y las resto, me queda un sistema equivalente:

$$a_{11}\ x_1 + a_{12}\ x_2 = b_1$$
$$\left(a_{22} - \frac{a_{12} a_{21}}{a_{11}} \right) x_2 = b_2 - \left(\frac{a_{21}}{a_{11}} b_1 \right)$$

$$a_{11}\ x_1 + a_{12}\ x_2 = b_1$$
$$a_{22}^{'}\ x_2 = b_2^{'}$$

Eliminación de Gauss simple

Con esto obtengo x_2
y luego, sustituyendo,
obtengo x_1.

$$x_2 = b_2^{'} / a_{22}^{'}$$

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

Puedo hacer lo mismo con un sistema más grande.

Elimino hacia delante, hasta que me quede una matriz triangular, y luego sustituyo hacia atrás.

Ejemplo

$$\begin{aligned} x - 3y - 2z &= 6 \\ 2x - 4y - 3z &= 8 \\ -3x + 6y + 8z &= -5 \end{aligned}$$

Eliminación de Gauss simple, ejemplo

$$(-2)R_1 + R_2 \to R_2 \quad \left(\begin{array}{ccc|c} 1 & -3 & -2 & 6 \\ 2 & -4 & -3 & 8 \\ -3 & 6 & 8 & -5 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 1 & -3 & -2 & 6 \\ 0 & 2 & 1 & -4 \\ -3 & 6 & 8 & -5 \end{array}\right)$$

$$(3)R_1 + R_3 \to R_3 \quad \left(\begin{array}{ccc|c} 1 & -3 & -2 & 6 \\ 0 & 2 & 1 & -4 \\ -3 & 6 & 8 & -5 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 1 & -3 & -2 & 6 \\ 0 & 2 & 1 & -4 \\ 0 & -3 & 2 & 13 \end{array}\right)$$

$$3R_2 + 2R_3 \to R_3 \quad \left(\begin{array}{ccc|c} 1 & -3 & -2 & 6 \\ 0 & 2 & 1 & -4 \\ 0 & -3 & 2 & 13 \end{array}\right) \Rightarrow \left(\begin{array}{ccc|c} 1 & -3 & -2 & 6 \\ 0 & 2 & 1 & -4 \\ 0 & 0 & 7 & 14 \end{array}\right)$$

Eliminación de Gauss simple

OK, esto se bien, es muy simple, pero cuánto cuesta??

Se mide en FLOPs, operaciones de punto flotante por segundo. Básicamente debo contar las multiplicaciones y divisiones (las sumas y restas son más baratas)

(recordar que Frontier hacía 1102.10^{15} FLOPS)

Si uno hace la cuenta, sale que el número de operaciones crece como $\frac{n^3}{3} + O(n^2)$

La peor parte la hace la eliminación!!

Es un método que escalea muy mal.

La memoria necesaria va como n^2, el tiempo como $\frac{n^3}{3}$

¿Qué consecuencias tiene esto?

Supongamos que tengo que duplicar el tamaño de mi problema (vamos a ver que esto puede ser simplemente pasar de $\Delta x \rightarrow \Delta x/2$): n --> (2 n)

La **memoria necesaria** aumenta **4 veces**
y el **tiempo de cálculo 8 veces**!!

Por esto necesito Supercomputadoras!!

Hay técnicas que escalean mejor, por supuesto, pero no mucho, esto sirve como una buena estimación del costo de un cálculo.

Eliminación de Gauss simple

En general funciona, pero...

- Problemas con divisiones por cero
- Errores de redondeo (sistemas grandes)
- Sistemas mal condicionados
- Sistemas singulares

Algunas soluciones

- Pivoteo
- Más cifras significativas
- Escaleamiento
- Sistemas singulares

Ejemplo de mejora: Descomposición LU

Hasta ahora vimos como el método de eliminación de Gauss nos permite resolver sistemas de ecuaciones algebraicas lineales, de la forma:

$$[A].\{x\} = \{b\}$$

¿Qué pasa si lo que tengo ahora es la misma matriz [A], pero diferentes vectores {b}?

La idea es escribir [A]=[L].[U]

[L] es triangular inferior y [U] triangular superior

Descomposición LU

Supongamos que el sistema a resolver es

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} . \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

y que puedo encontrar dos matrices

$$L = \begin{pmatrix} 1 & 0 & 0 \\ f_{21} & 1 & 0 \\ f_{31} & f_{32} & 1 \end{pmatrix} \quad \text{y} \quad U = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a'_{33} \end{pmatrix}$$

$$\text{tal que } \left[A\right] = \left[L\right].\left[U\right]$$

Descomposición LU

Entonces lo que puedo hacer es hallar $\{\mathrm{d}\}$ tal que

$$[\mathrm{L}].\{d\} = \{b\}$$

y luego $\{x\}$ tal que

$$[\mathrm{U}].\{x\} = \{d\}$$

$$\Rightarrow [\mathrm{L}].\{d\} = \{b\} \rightarrow [\mathrm{L}].[\mathrm{U}].\{x\} = \{b\} \rightarrow [A].\{x\} = \{b\}$$

Se puede demostrar que la descomposición LU “cuesta” lo mismo que el método de eliminación de Gauss, pero se puede aplicar a muchos vectores {b} distintos.

Método iterativo de Gauss-Seidel

Es un método iterativo para acercarse lo más posible a la solución. Se usa para sistemas MUY grandes

Veamos un ejemplo para un sistema de 3x3.

Tengo $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} . \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$, formalmente puedo despejar x_1, x_2 y x_3 :

$$x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3}{a_{11}}, \quad x_2 = \frac{b_2 - a_{21}x_1 - a_{23}x_3}{a_{22}}, \quad x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}}$$

Puedo tomar valores iniciales y luego iterar hasta convergencia:

$$\text{abs}\,(x_i^n - x_i^{n-1}) < Error$$

Para mejorar la convergencia, las iteraciones se toman relajadas (con algún β)

$$x_i^{\text{prox}} = \beta x_i^{\text{pred}} + (1-\beta)x_i^{\text{ant}}$$

Resolución de problemas usando *bibliotecas*

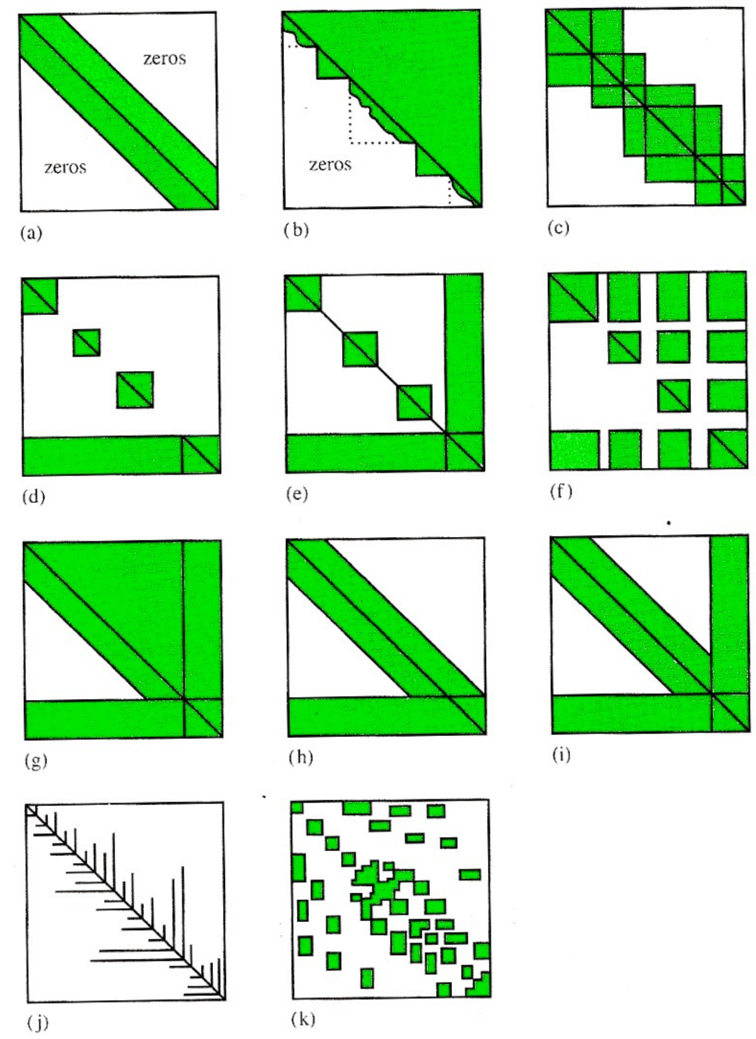
La mayoría de las veces, la solución de un sistema de ecuaciones se realiza usando bibliotecas, optimizadas para ese fin.

- Matchad / MATLAB / Excel
- IMSL
- LAPACK/BLAS
- SCALAPACK

Referencia importante: Numerical Recipes (http://www.nr.com/)

Sistemas lineales especiales

- Matrices tridiagonales
- Matrices diagonales por bandas
- Problemas con matrices ralas (*sparse*)



- Es importante usar las simetrías si existen
- “Reglas de selección”

- Aproximaciones y errores
- Raíces de ecuaciones
- Ecuaciones algebraicas lineales
- Optimización
- Ajuste de funciones e interpolación
- Diferenciación numérica
- Ecuaciones diferenciales ordinarias
- Ecuaciones diferenciales parciales

Tenemos un conjunto de puntos $(x_1,f(x_1)),\ (x_2,f(x_3)),\ \ldots.,\ (x_N,f(x_N))$.

Podemos tener dos puntos de vista alternativos:

a) Los datos se asumen sin error o no se conoce *a priori* su dependencia funcional. Se intentará, entonces, ajustar una curva que pase por todos los puntos.

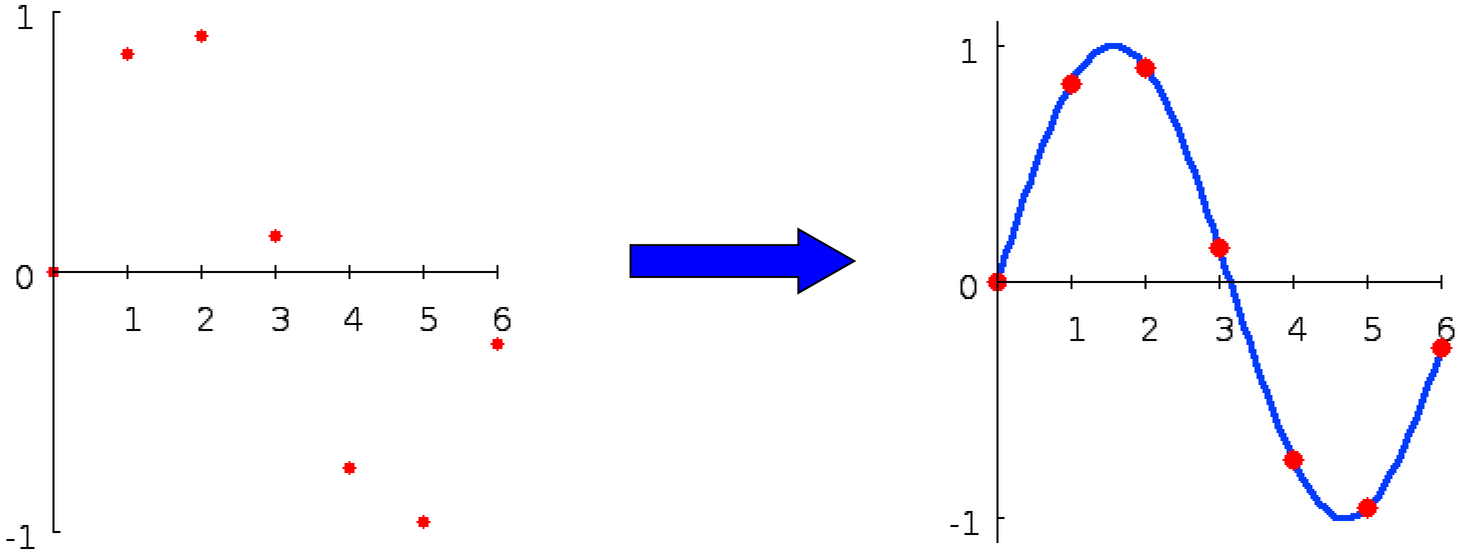
Interpolación

Tenemos un conjunto de puntos
$(x_1,f(x_1)), (x_2,f(x_3)), \ldots., (x_N,f(x_N))$.

Podemos tener dos puntos de vista alternativos:

b) Los datos se conocen con un cierto error o se asume una cierta forma funcional y se necesitan calcular los parámetros intervinientes en dicha función.

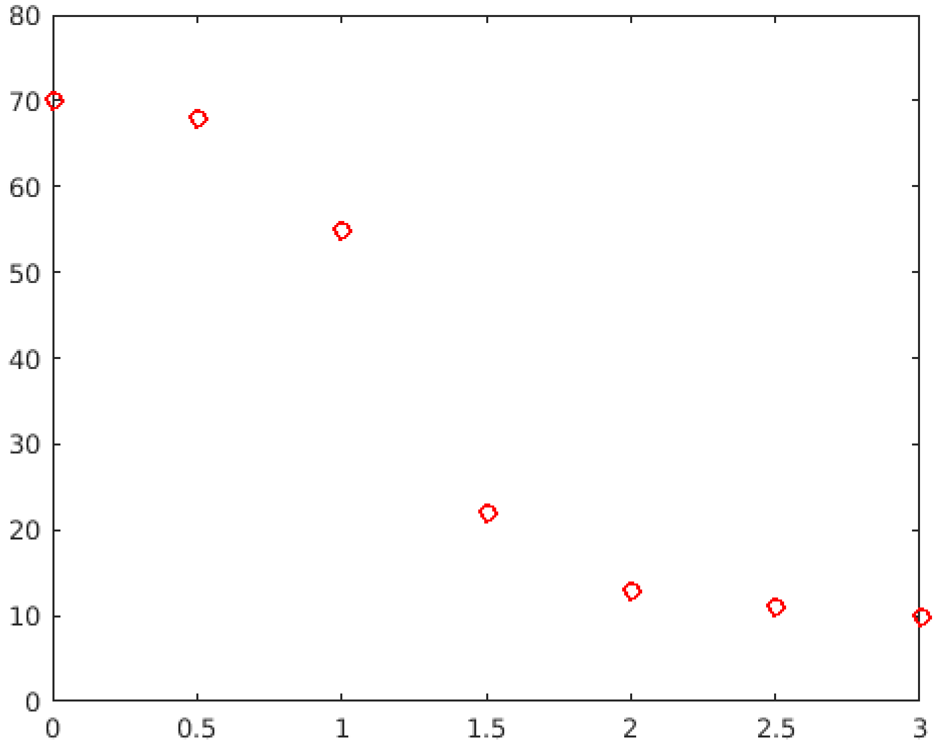
Ajuste de funciones



- **Existen muchos tipos de formas funcionales útiles para interpolar:**
  - » **Polinomios**
  - » **Funciones trigonométricas**
  - » **Exponenciales**

Interpolación, ejemplo

Supongamos que quiero una curva que pase por los siguientes puntos



A las funciones de interpolación se les pide que sean fáciles de determinar, de evaluar, derivar, integrar, etc.

Una de las formas más usuales es utilizando polinomios.

Si bien hay un solo polinomio de grado (*N-1*) que pasa por *N* puntos dados, hay varias maneras de expresarlo:

- **Polinomios de Lagrange**
- **Polinomios de Newton**

Pediré que los polinomios cumplan:

$$l_i(x_j) = \delta_{ij}$$

De esta manera puedo definir un interpolador que pase por todos los puntos dados como

$$p(x) = \sum_{i=1}^{N} y_i \, l_i(x), \qquad y_i = f(x_i)$$

Estos *$l_i(x)$* son de la forma

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{N} \frac{(x - x_j)}{(x_i - x_j)}$$

Polinomios de Lagrange

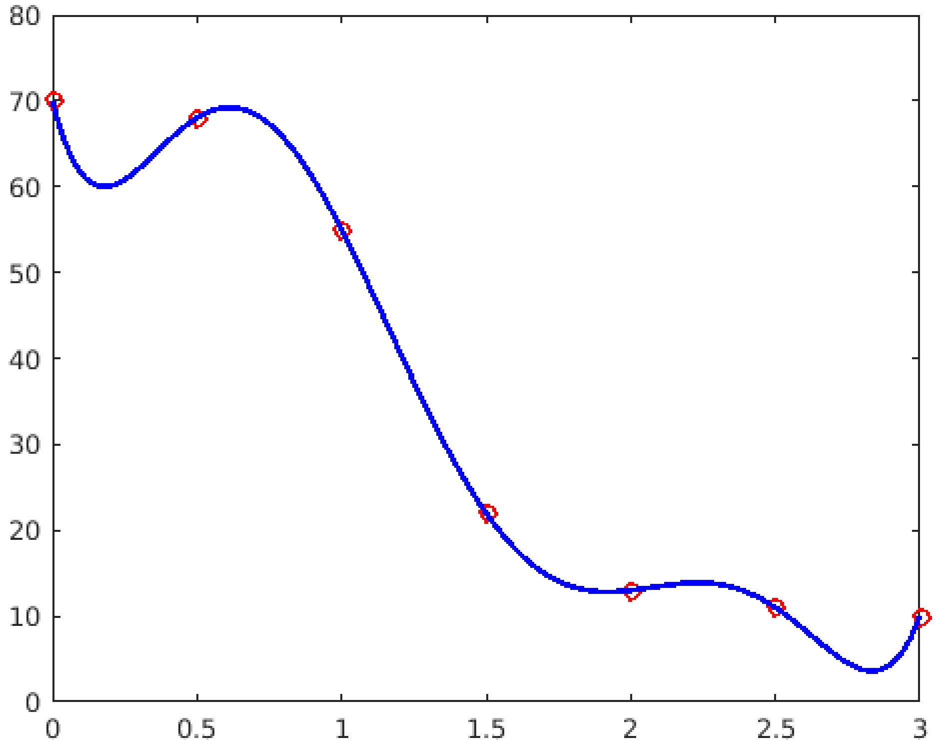
Por un lado son muy sencillos de escribir, sin embargo, tienen dos desventajas grandes:

- **Todo el trabajo se debe rehacer cuando se cambia el grado del polinomio.**
- **Todo el trabajo se debe rehacer para cada *x***

Para el primer punto hay formas recursivas de definir el polinomio: algoritmo de Neville

Interpolación, ejemplo

Veamos la curva con polinomios de Lagrange que pasa por los puntos anteriores



Es un método recursivo, fácil y con el error acotado.

Si aproximo por una recta:

$$f_1(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

Si aproximo por una parábola:

$$f_2(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2)$$

con $b_1 = f(x_1)$

$$b_2 = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

sale que: $b_3 = \dfrac{\dfrac{f(x_3) - f(x_2)}{x_3 - x_2} - \dfrac{f(x_2) - f(x_1)}{x_2 - x_1}}{(x_3 - x_1)}$

Se puede generalizar como

$$b_1 = f(x_1)$$

$$b_2 = f[x_2, x_1] = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

$$b_3 = f[x_3, x_2, x_1] = \frac{f[x_3, x_2] - f[x_2, x_1]}{(x_3 - x_1)}$$

en general

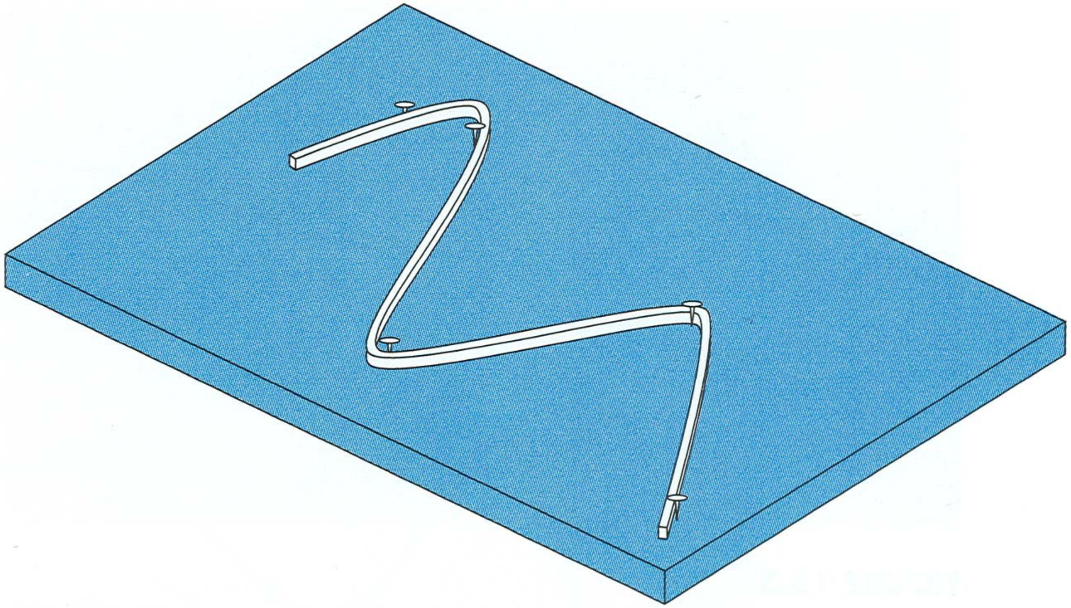
$$b_n = f[x_n, x_{n-1}, \ldots, x_1] = \frac{f[x_n, x_{n-1}, \ldots, x_2] - f[x_{n-1}, \ldots, x_1]}{(x_n - x_1)}$$

Si los polinomios son de grado muy alto, tendrán oscilaciones muy fuertes, y también problemas de redondeo y con puntos muy lejanos.

Una alternativa es aplicar polinomios de orden inferior a un subconjunto de puntos, con ciertas condiciones de pegado.

Funciones segmentarias o splines.

Splines



El grado del polinomio estará dado por cuantas condiciones le impondré a las funciones de interpolación, la *suavidad* de la curva

Entre cada par de puntos interpolo con una recta y pido continuidad de la función

$$f(x) = f(x_1) + m_1(x - x_1) \qquad x_1 \leq x \leq x_2$$

$$f(x) = f(x_2) + m_2(x - x_2) \qquad x_2 \leq x \leq x_3$$

....

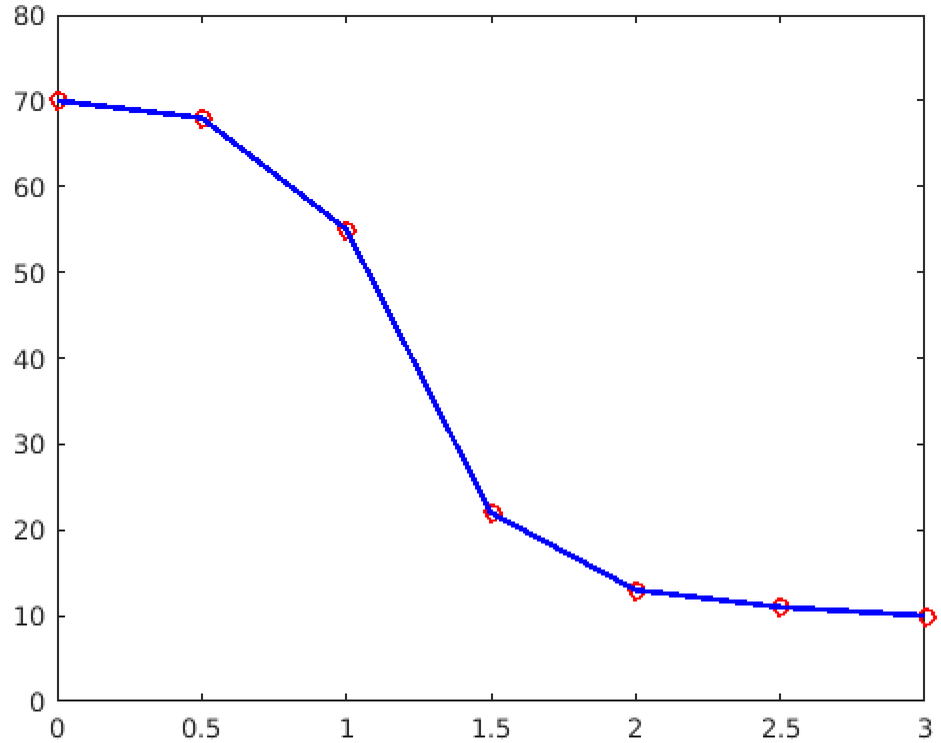
$$f(x) = f(x_{N-1}) + m_{N-1}(x - x_{N-1}) \qquad x_{N-1} \leq x \leq x_N$$

entonces

$$m_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \qquad 1 \leq i \leq N - 1$$

Interpolación, ejemplo

Veamos la curva con splines linealas que pasa por los puntos anteriores



Entre cada par de puntos interpolo con una parábola y pido continuidad de la función y de su derivada

$$f_i(x) = a_i(x - x_i)^2 + b_i(x - x_i) + c_i$$

$$1 \le i \le n - 1, \; x_i \le x \le x_{i+1}$$

- **Necesitaré *3(n-1) = (3n-3)* coeficientes**
- **Tengo:**
  - **$f(x_i)=y_i$: *n* datos**
  - **continuidad f(x): *n–2* datos**
  - **continuidad f'(x): *n–2* datos**

_3n–4_ datos

Necesitaré una condición externa!!

¿Cómo se opera?

Si defino $y_i = f(x_i),\quad h_i = x_{i+1} - x_i$

operando, me queda un sistema de ecuaciones para b_i

$$b_{i+1} = \frac{2(y_{i+1} - y_i)}{h_i} + b_i$$

Si conozco un b_i saco todos los demás y de allí los a_i. Como ademas $c_i = y_i$

$$\Rightarrow\ f_i(x) = \frac{(b_{i+1} - b_i)}{2h_i}(x - x_i)^2 + b_i(x - x_i) + y_i$$

¿Algo mejor? splines cúbicos

$$f_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$$

$$1 \le i \le n - 1, \quad x_i \le x \le x_{i+1}$$

Se pide continuidad en las funciones, las derivadas primera y segunda.

¿Se sigue mejorando?

En general no, con esto es suficiente para la mayoría de los casos.

Splines cúbicos

- **Necesitaré $4(n-1) = (4n-4)$ coeficientes**
- **Tengo:**
  - **$f(x_i)=y_i$: n datos**
  - **continuidad $f(x)$: $n–2$ datos**
  - **continuidad $f'(x)$: $n–2$ datos**
  - **continuidad $f''(x)$: $n–2$ datos**

$4n–6$ datos

Necesitaré dos condiciones extras!!

Splines cúbicos

Si defino $y_i = f(x_i)$, $h_i = x_{i+1} - x_i$ y opero como antes puedo poner todas las variables, a_i, c_i y d_i en función de los b_i, quedando un sistema de ecuaciones para b_i

$$h_{i-1}\, b_{i-1} + 2(h_{i-1} + h_i)\, b_i + h_i\, b_{i+1} = 3\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right)$$

¿Cómo se resuelve esto?

$\Rightarrow$ **es un sistema del tipo** $\overline{\overline{\mathrm{A}}}\,\bar{b} = \bar{t}$

Como me faltan 2 condiciones,

➡ tengo que imponerlas

Splines cúbicos

Condiciones más usuales

Spline Natural: $f_1''(x_1) = 0 \;\; \text{y} \;\; f_{N-1}''(x_N) = 0$

Bordes fijos: $f_1'(x_1) \;\; \text{y} \;\; f_{N-1}'(x_N) \quad \text{dadas}$

Not a knot:

$$f_1^{(3)}(x_2) = f_2^{(3)}(x_2)$$

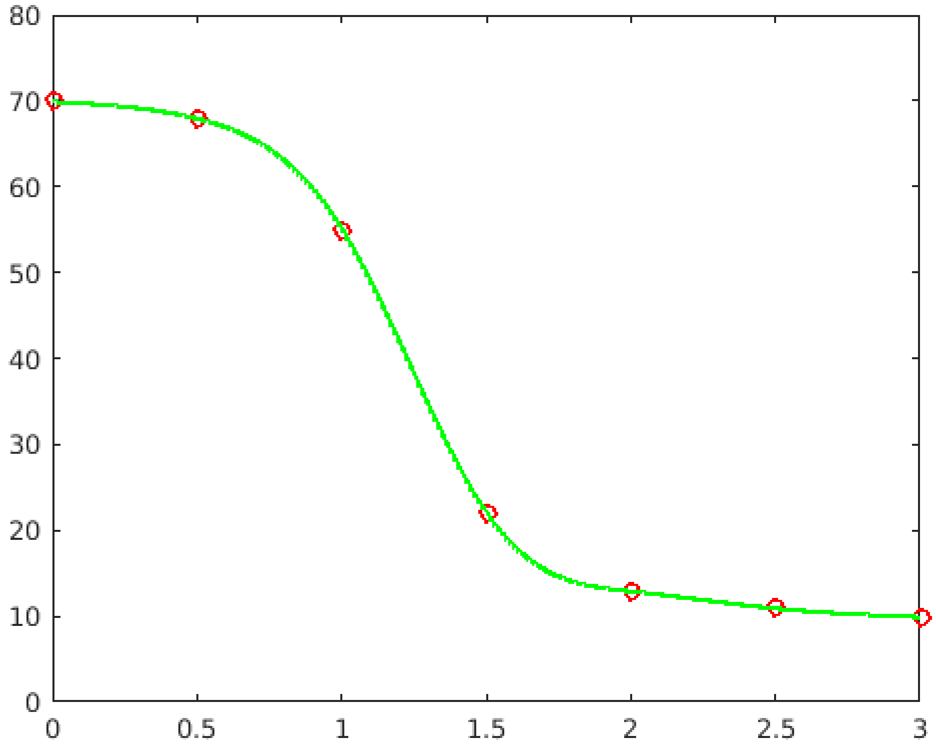
$$f_{N-2}^{(3)}(x_{N-1}) = f_{N-1}^{(3)}(x_{N-1})$$

Es un proceso en dos pasos

- **Se determinan los coeficientes de los distintos polinomios**
- **Se los usa cuando se necesitan (la evaluación es muy rápida porque el polinomio es de grado chico)**

Interpolación, ejemplo

Veamos la curva con splines cúbicos que pasa por los puntos anteriores



Ajuste de funciones

Si los datos que se disponen tienen errores y/o se conoce la forma funcional que deben seguir la interpolación polinomial es inapropiada

Supongo entonces una cierta forma funcional

$$y(x) = f(x; a_1, ..., a_M)$$

e intento determinar los parámetros intervinientes

La idea es minimizar el error

$$e = \sum_{i=1}^{N} (y_{medida} - y_{estimada})^2$$

Ajuste de funciones: cuadrados mínimos

Supongamos que tenemos una regresión lineal

$$y(x) = y(x;a,b) = a + b\ x$$

Debo entonces minimizar el error:

$$\frac{\partial e}{\partial a} = 0; \qquad \frac{\partial e}{\partial b} = 0$$

$$S_x = \sum_{i=1}^{N} x_i; \quad S_y = \sum_{i=1}^{N} y_i;$$

$$S_{xx} = \sum_{i=1}^{N} x_i^2; \quad S_{xy} = \sum_{i=1}^{N} y_i\ x_i$$

$$a = \frac{S_y\ S_{xx} - S_x\ S_{xy}}{N\ S_{xx} - (S_x)^2}; \quad b = \frac{N\ S_{xy} - S_x\ S_y}{N\ S_{xx} - (S_x)^2}$$

Ajuste de funciones: cuadrados mínimos

Esto se puede generalizar fácilmente:

$$y(x) = \sum_{k=1}^{M} a_k \, \varphi_k(x)$$

$\varphi_k(x)$ **puede ser cualquier tipo de función**

Minimizando el error, se obtiene un sistema del tipo

$$\sum_{i=1}^{N} A_{kj} \, a_j = B_k$$

con $A_{kj} = \sum_{i=1}^{N} \varphi_j(x_i) \, \varphi_k(x_i); \quad B_k = \sum_{i=1}^{N} y_i \, \varphi_k(x_i)$

Es un sistema lineal que se resuelve fácilmente.

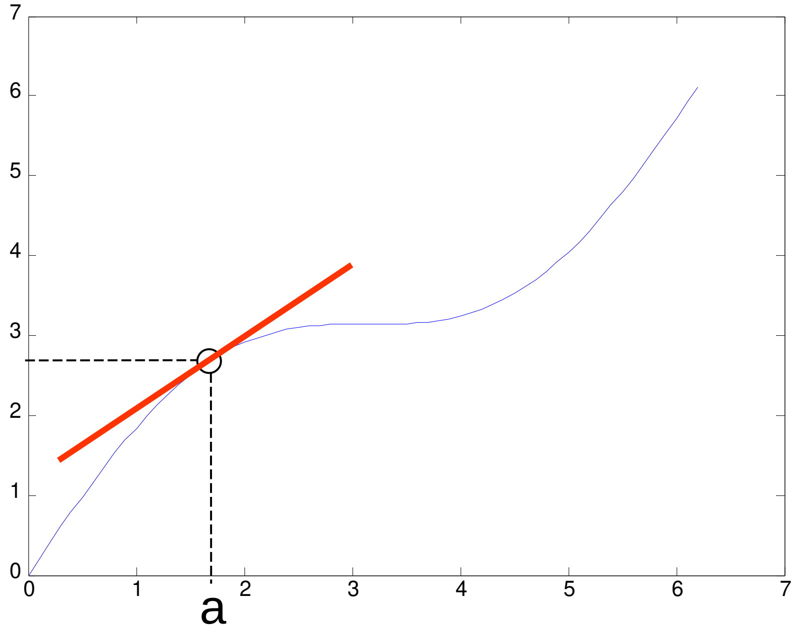
Elementos de cálculo numérico

- **Aproximaciones y errores**
- **Raíces de ecuaciones**
- **Ecuaciones algebraicas lineales**
- **Optimización**
- **Ajuste de funciones e interpolación**
- **Diferenciación e integración numérica**
- **Ecuaciones diferenciales ordinarias**
- **Ecuaciones diferenciales parciales**

Derivación numérica

Obviamente ya todos sabemos lo que son las derivadas, pero vamos a ver como se calculan numéricamente, ya que no podemos tomar el límite adecuado.

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$



Derivación numérica

En general nos basaremos en el desarrollo de series de Taylor de una función.

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

con lo que puede obtenerse

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

Derivación numérica

Uno por supuesto puede truncar aquí

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

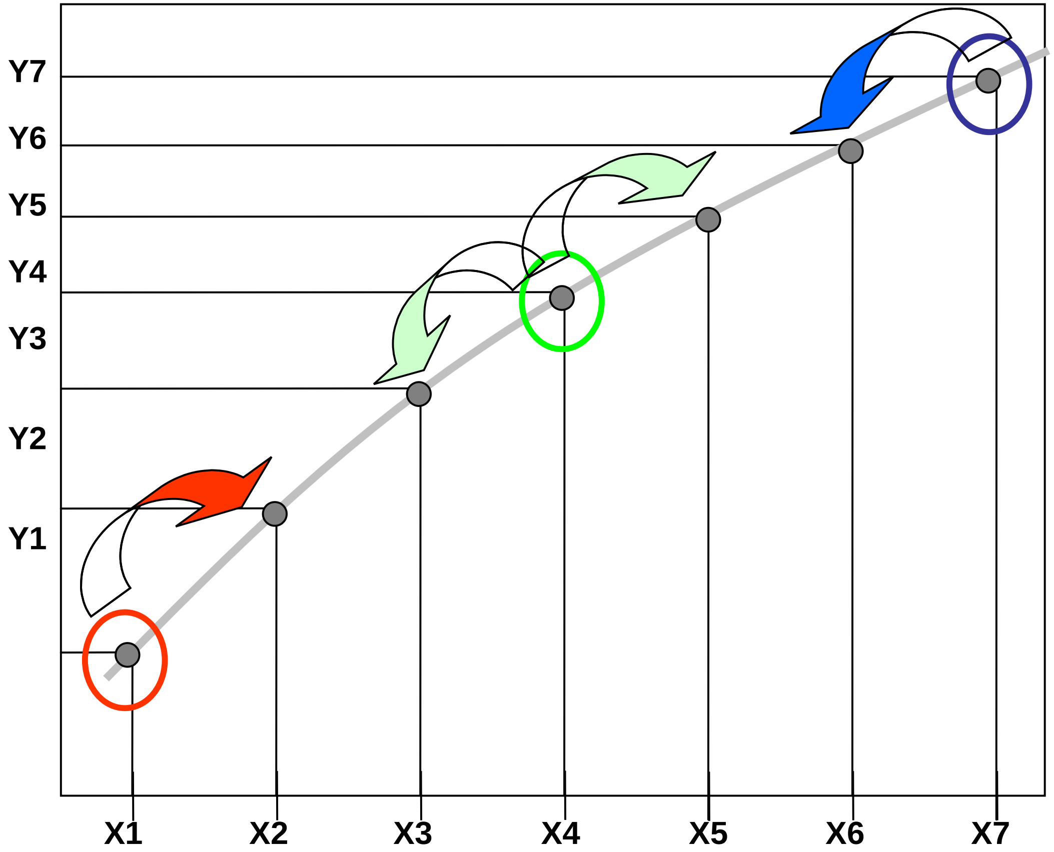
Esta es una aproximación hacia adelante.

También puedo calcularla hacia atrás o centrada

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

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

Derivación numérica



Derivación numérica

Y si quiero mas precisión?

Intento expresar la derivada segunda y reemplazarla en la ecuación anterior…

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

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

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

Derivación numérica

Así hay una serie de expresiones para todas las derivadas.

Por ejemplo para derivadas segundas:

$$f''(x_i) = \frac{-f(x_{i+3}) + 4f(x_{i+2}) - 5f(x_{i+1}) + 2f(x_i)}{h^2}$$

$$f''(x_i) = \frac{2f(x_i) - 5f(x_{i-1}) + 4f(x_{i-2}) - f(x_{i-3})}{h^2}$$

$$f''(x_i) = \frac{-2f(x_{i+2}) + 16f(x_{i+1}) - 30f(x_i) + 16f(x_{i-1}) - f(x_{i-2})}{12h^2}$$

Integración numérica

La idea es hallar la integral definida de una función cualquiera

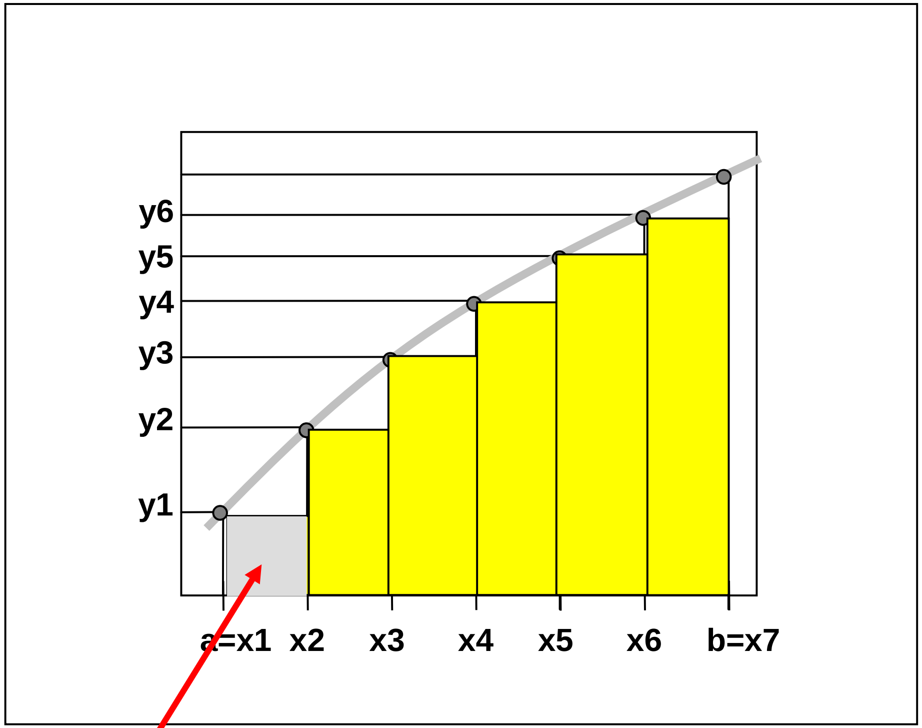
$$I = \int_a^b f(x)\,dx$$

En general se intenta aproximar la función real por polinomios y luego integrarla.

Son las fórmulas de Newton-Cotes.

Lo mas fácil es hacer:

Integración numérica

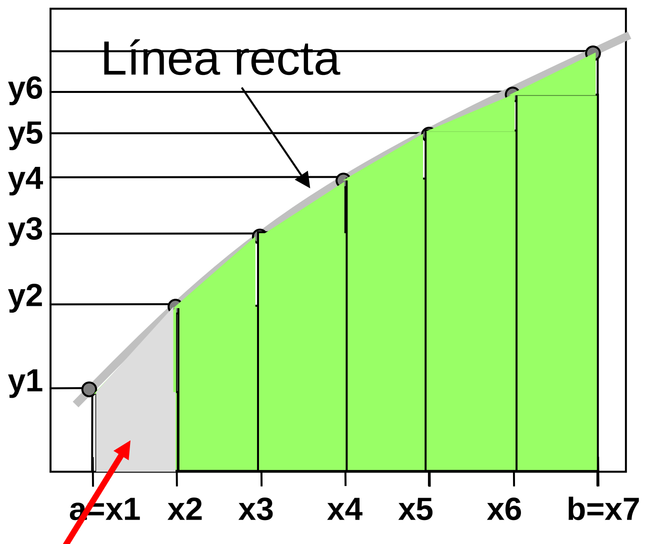


$$A = (x_2 - x_1) * y_1$$

$$I_L = \sum_{j=0}^{n-1} h\, f(a + j\, h), \quad \text{con} \quad n = \frac{b-a}{h}$$

Integración numérica, trapecios

Algo mejor es hacer trapecios…



A= $(x_2-x_1)*(y_2+y_1)/2$

$$I_1 = (b-a)\frac{f(b)+f(a)}{2}$$

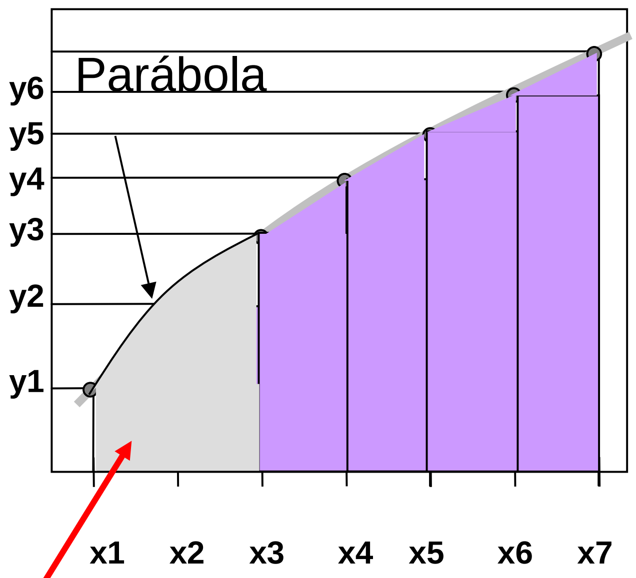
$$I_L = \left(\frac{b-a}{n}\right)\frac{f(x_0)+2\sum_{j=1}^{n-1} f(x_j)+f(x_n)}{2}$$

el error será $E_a = -\frac{(b-a)^3}{12\,n^2}\,\overline{f}\,''$

Integración numérica, Simpson

Y si seguimos con la idea, llegamos a la regla de Simpson 1/3

Regla de Simpson



A= $(x_3-x_1)*(y_1+4y_2+y_3)/6$

$$I_1 = \frac{(b-a)}{2}\,\frac{f(x_1)+4f(x_2)+f(x_3)}{3}$$

$$I_L = \left(\frac{b-a}{n-1}\right)\frac{f(x_1)+4\sum_{j=par}^{n-1} f(x_j)+2\sum_{j=impar}^{n-2} f(x_j)+f(x_n)}{3}$$

el error será $E_a = -\dfrac{(b-a)^5}{180\,(n-1)^4}\,\overline{f}^{(4)}$

Notar que *n* debe ser impar!!

Integración numérica, Newton-Cotes

La historia sigue con polinomios mayores

Fórmulas de Newton-Cotes

$$I = n\,\beta\,h\left(a_0\,f_0 + a_1\,f_1 + a_2\,f_2 + a_3\,f_3 + \ldots.\right)$$

n	β	a_0	a_1	a_2	a_3	a_4	a_5
1	½	1	1				
2	1/6	1	4	1			
3	1/8	1	3	3	1		
4	1/90	7	32	12	32	7	
5	1/288	19	75	50	50	75	19

Integración numérica, cuadratura de Gauss

Otra idea es usar cuadratura de Gauss:

En lugar de definir a priori la posición de los *n*+1 puntos de muestreo, se los determinará de manera de obtener el mayor orden de precisión para *n* dado.

Así se intentará determinar puntos t_0, t_1, …, t_n y números c_0, c_1,…,c_n tales que para todo polinomio p(t) de grado $\leq$ 2n-1,

$$\int_a^b p(t)\,dt = c_0\ p(t_0) + c_1\ p(t_1) + \ldots + c_n\ p(t_n)$$

Integración numérica, cuadratura de Gauss

Ejemplo, [a,b]=[-1,1] y dos puntos

$$c_0\,1 + c_1\,1 = \int_{-1}^{1} 1\,dt = 2$$

$$c_0\,t_0 + c_1\,t_1 = \int_{-1}^{1} t\,dt = 0$$

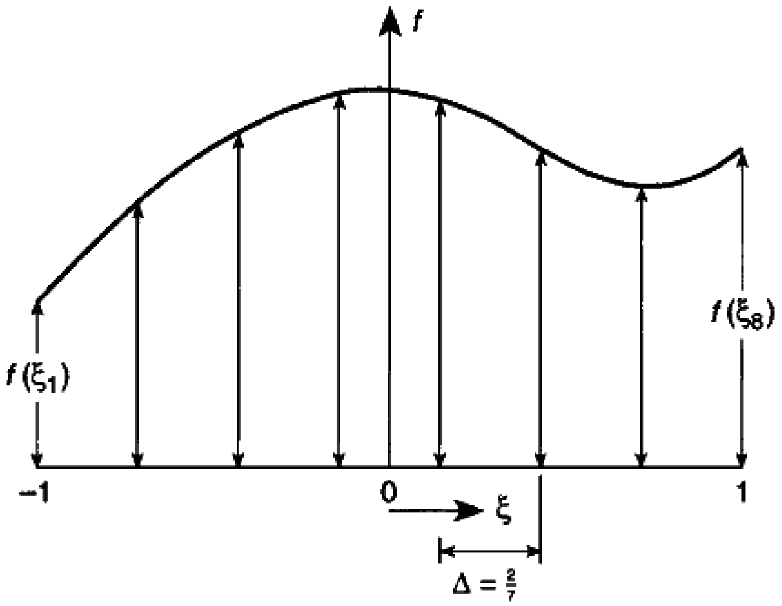
$$c_0\,t_0^{\,2} + c_1\,t_1^{\,2} = \int_{-1}^{1} t^2\,dt = \frac{2}{3}$$

$$c_0\,t_0^{\,3} + c_1\,t_1^{\,3} = \int_{-1}^{1} t^3\,dt = 0$$

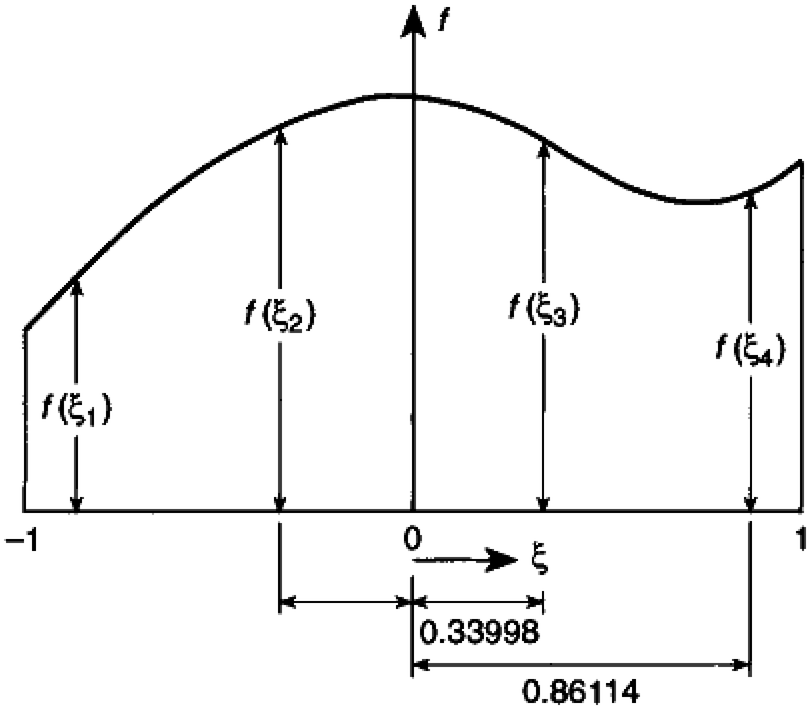
$$\Rightarrow \begin{cases} c_0 = c_1 = 1 \\ t_0 = -1/\sqrt{3} \\ t_1 = \;\;1/\sqrt{3} \end{cases}$$

Integración numérica, cuadratura de Gauss

Ej: Integración exacta de un polinomio de grado 7



Newton-Cotes



Gauss-Legendre

Integración numérica, cuadratura de Gauss

Table 9.1 Abscissae and weight coefficients of the gaussian quadrature formula $\int_{-1}^{1} f(x)\,\mathrm{d}x = \sum_{i=1}^{n} H_i f(a_i)$

$\pm a$		H
	$n = 1$	
0		2.000 000 000 000 000
	$n = 2$	
$1/\sqrt{3}$		1.000 000 000 000 000
	$n = 3$	
$\sqrt{0.6}$		5/9
0.000 000 000 000 000		8/9
	$n = 4$	
0.861 136 311 594 953		0.347 854 845 137 454
0.339 981 043 584 856		0.652 145 154 862 546
	$n = 5$	
0.906 179 845 938 664		0.236 926 885 056 189
0.538 469 310 105 683		0.478 628 670 499 366
0.000 000 000 000 000		0.568 888 888 888 889
	$n = 6$	
0.932 469 514 203 152		0.171 324 492 379 170
0.661 209 386 466 265		0.360 761 573 048 139
0.238 619 186 083 197		0.467 913 934 572 691
	$n = 7$	
0.949 107 912 342 759		0.129 484 966 168 870
0.741 531 185 599 394		0.279 705 391 489 277
0.405 845 151 377 397		0.381 830 050 505 119
0.000 000 000 000 000		0.417 959 183 673 469

Integración numérica, cuadratura de Gauss

Veamos como se usa

Quiero calcular $I = \int_a^b f(x)\, dx$ pero tengo $I_G = \int_{-1}^{1} F(t)\, dt$

Lo primero que debo hacer es transformar $t \rightarrow x$

Planteo: $x = mt + c$

Si $x = a \rightarrow t = -1$ y $x = b \rightarrow t = 1$

$$\Rightarrow \quad x = \left(\frac{b-a}{2}\right) t + \left(\frac{b+a}{2}\right), \quad dx = \left(\frac{b-a}{2}\right) dt$$

$$\Rightarrow \quad I = \int_a^b f(x)\, dx = \left(\frac{b-a}{2}\right) \int_{-1}^{1} F(t)\, dt = \left(\frac{b-a}{2}\right) \sum_{i=1}^{n} c_i\, F(t_i)$$

Integración numérica, cuadratura de Gauss

Quiero calcular, por ejemplo: $I = \int_{3.1}^{3.9} \frac{1}{x}\, dx$

Por lo anterior $m = \left(\frac{3.9-3.1}{2}\right) = 0.4, \quad \mathrm{c} = \left(\frac{3.9+3.1}{2}\right) = 3.5$

$$x = 0.4t + 3.5 \quad \Rightarrow \quad F(t) = \frac{1}{0.4t+3.5}$$

$$\Rightarrow \quad I = \int_{3.1}^{3.9} \frac{1}{x}\, dx = 0.4\int_{-1}^{1} \frac{1}{0.4t+3.5}\, dt = 0.4\sum_{i=1}^{n} c_i\, F(t_i)$$

Si tomo dos puntos:

$$\Rightarrow \quad I = \int_{3.1}^{3.9} \frac{1}{x}\, dx = 0.4\left([1]F\left(-1/\sqrt{3}\right) + [1]F\left(-1/\sqrt{3}\right)\right)$$

$$\Rightarrow \quad I = 0.4\left(\frac{1}{0.4(-1/\sqrt{3})+3.5} + \frac{1}{0.4(1/\sqrt{3})+3.5}\right) = 0.22957092$$

Comparado con el valor "real" = 0.22957444, el error es 0.0015%!!