MÉTODOS NUMÉRICOS

Iker Caballero Bragagnini

Tabla de contenido

Los fundamentos de los métodos numéricos

- Los métodos y el análisis numérico tratan de las matemáticas y las metodologías numéricas que subyacen la computación científica. Se discute la derivación y la implementación de los algoritmos, analizando los algoritmos matemáticamente

    - En el estudio de las técnicas se destacan dos elementos que normalmente compiten entre ellos: la exactitud y la eficiencia

        - Pocos cálculos dan una solución exacta para un problema, por lo que se tiene que entender cómo nace el error y cómo controlar o disminuir el error (aumentar la exactitud)

        - Aunque las computadoras son rápidas, hay maneras lentas y maneras rápidas de hacer cosas. Con todo igual, se prefieren aquellos algoritmos que son más rápidos que otros (más eficientes)

        - Se dice que ambas cosas entran en competición la una con la otra debido a que, generalmente hablando, los pasos que se pueden tomar para hacer que un algoritmo sea más exacto son más costosos en términos de eficiencia

    - Un tercer elemento importante en el estudio de los métodos numéricos que no es obvio como los otros: la estabilidad

        - La estabilidad se refiere a si el método produce resultados similares para datos similares, y si cambios pequeños en los datos causan grandes diferencias en los resultados. Si esto es así, se dice que el método es inestable, y los métodos inestables tienden a producir resultados no fiables

        - Es posible tener un método exacto que se implemente eficientemente pero que sea muy inestable

- Existen varios resultados del cálculo que son fundamentales para el análisis numérico, tales como el teorema de Taylor, el teorema del valor medio, el teorema del valor intermedio y otros

    - Siendo $f(x)$ con $n+1$ derivadas en $[a,b]$ para alguna $n \geq 0$, y siendo $x, x_0 \in [a,b]$, entonces se cumple la siguiente igualdad:

$$f(x) = p_n(x) + R_n(x) \qquad where$$

$$p_n(x) = \sum_{k=0}^{n} \frac{(x-x_0)^k}{k!} f^{(k)}(x_0) \;\; \& \;\; R_n(x) = \frac{1}{n!} \int_{x_0}^{x} (x-t)^n f^{(n+1)}(t) dt$$

- Además, existe un punto c entre x y x_0 tal que se cumple la siguiente igualdad:

$$R_n(x) = \frac{(x - x_0)^{n+1}}{(n+1)!} f^{(n+1)}(c)$$

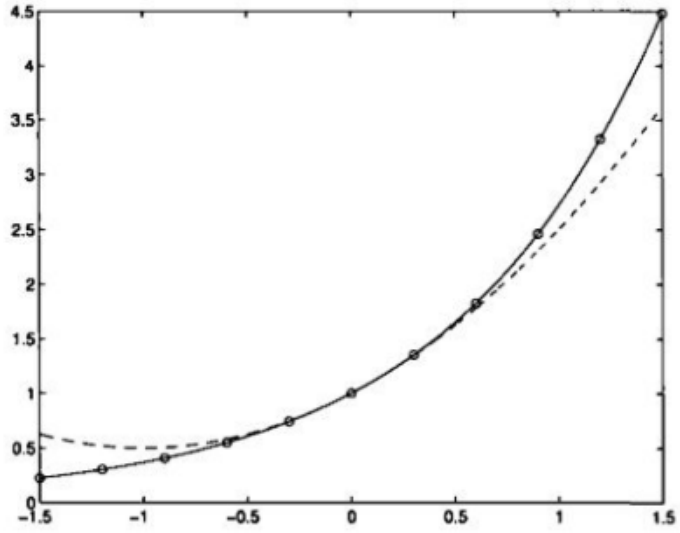
- El punto x_0 normalmente se escoge a discreción del usuario, y normalmente se toma como si fuera 0. Además, las dos formas del residuo son equivalentes: la forma por puntos (la última) se puede derivar de la forma integral (anterior)

- El teorema de Taylor es importante porque permite representar exactamente funciones generales en términos de polinomios con un error específico, conocido y acotable. Esto permite reemplazar las funciones generales con algo más simple que a su vez tiene una cota superior para el error que se comete

  - Un ejemplo clásico para poder entender la utilidad del teorema de Taylor es a través de la expansión de la función exponencial. Esta se puede representar de la siguiente manera:

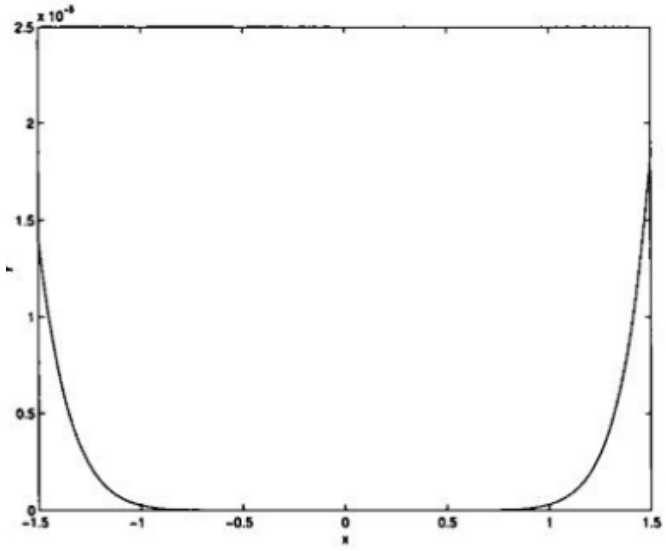
$$e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \cdots + \frac{1}{n!}x^n + \frac{1}{(n+1)!}x^{n+1}e^c =$$

$$= \sum_{k=1}^{n} \frac{x^k}{k!} + \frac{x^{n+1}}{(n+1)!}e^c = p_n(x) + R_n(x)$$



  - Si se quiere que una aproximación sea exacta dentro de un error absoluto de 10^{-t} para un intervalo $x \in [-1,1]$, entonces se tiene que respetar la siguiente desigualdad:

$$|e^x - p_n(x)| = |R_n(x)| \leq 10^{-t}$$



- A través de encontrar una cota superior simple para el valor absoluto del residuo, es posible obtener una desigualdad entre el error absoluto que se desea y esta cota, y después se puede obtener la n^* para la cual se tiene un error como mucho de 10^{-t}

$$|R_n(x)| = \left|\frac{x^{n+1}e^c}{(n+1)!}\right| = \frac{|x|^{n+1}e^c}{(n+1)!} \leq \frac{e^c}{(n+1)!} \leq \frac{e}{(n+1)!}$$

$$\Rightarrow |R_n(x)| \leq \frac{e}{(n+1)!} \leq 10^{-t} \Rightarrow \frac{e}{(n+1)!} \leq 10^{-t} \Rightarrow n^*$$

- Por lo tanto, se puede ver como el teorema de Taylor sirve para encontrar una aproximación a una función, utilizando un polinomio específico, tal que se tenga una exactitud concreta dentro de un intervalo (se construyen aproximaciones para cálculos difíciles y se sabe qué tan exactas son)

- Un tipo de expansión de Taylor especial que resultará útil es la expansión de $f(x+h)$ alrededor del punto $x = x_0$, donde h se considera un parámetro pequeño. La aplicación directa permite obtener los siguientes resultados:

$$f(x+h) = f(x) + [(x+h) - x]f'(x) + \frac{1}{2!}[(x+h) - x]^2 f''(x) +$$

$$+ \cdots + \frac{1}{n!}[(x+h) - x]^n f^{(n)}(x) + \frac{1}{(n+1)!}[(x+h) - x]^{n+1} f^{(n+1)}(c)$$

$$= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \cdots + \frac{1}{n!}h^n f^{(n)}(x) +$$

$$+\frac{1}{(n+1)!}h^{n+1}f^{(n+1)}(c)$$

- Este resultado resultará importante para realizar derivaciones e integraciones numéricas

- Siendo f una función continua en $[a,b]$ y diferenciable en (a,b), entonces existe un punto $c \in [a,b]$ tal que se cumple la siguiente igualdad (teorema del valor medio):

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

  - Desde el punto de vista del análisis numérico, este teorema es muy importante, dado que permite reemplazar diferencias de los valores de funciones con diferencias de sus argumentos, escalándolos por la derivada correspondiente

$$|\cos(x_2) - \cos(x_1)| = \sin(c)\,|x_2 - x_1| \leq |x_2 - x_1|$$

- Siendo f una función continua en $[a,b]$, y asumiendo que W es un valor entre $f(a)$ y $f(b)$ (de modo que $f(a) \leq W \leq f(b)$ o $f(b) \leq W \leq f(a)$, alternativamente), entonces existe un punto $c \in [a,b]$ tal que $f(c) = W$

  - Este teorema es la idea básica detrás del método de la bisección, y además permite ver que una función continua es una cuyo gráfico se puede dibujar sin ningún corte (existe siempre un valor intermedio)

- Adicionalmente, un resultado relacionado con este valor es el teorema del valor extremo

  - Siendo f una función continua en $[a,b]$, entonces existe un punto $m \in [a,b]$ tal que $f(m) \leq f(x)$ para toda $x \in [a,b]$ y un punto $M \in [a,b]$ tal que $f(M) \geq f(x)$ para toda $x \in [a,b]$. Además, f consigue su máximo y su mínimo en $[a,b]$ en los puntos extremos a o b, o en un punto crítico

- Siendo f y g dos funciones continuas en $[a,b]$, y asumiendo que g no cambia de signo en $[a,b]$, entonces existe un punto $c \in [a,b]$ tal que se cumple la siguiente igualdad (teorema del valor medio integral):

$$\int_a^b g(t)f(t)\,dt = f(c)\int_a^b g(t)\,dt$$

- Siendo f una función continua en $[a, b]$ y considerando la suma $S = \sum_{k=1}^{n} a_k f(x_k)$ donde cada punto $x_k \in [a, b]$, y los coeficientes satisfacen que $a_k \geq 0$ y que $\sum_{k=1}^{n} a_k = 1$, entonces existe un punto $c \in [a, b]$ tal que $f(c) = S$

$$f(c) = \sum_{k=1}^{n} a_k f(x_k)$$

- A partir de estos resultados, se discute sobre la definición del error que se usa para métodos numéricos, del concepto de igualdad aproximada y del concepto de orden asintótico, los cuales juegan un papel relevante

  - Si x es una cantidad que se quiere calcular y x_h es una aproximación a esta cantidad, entonces el error se define como $x - x_h$

    - El error absoluto es el valor absoluto del error, definido de la siguiente manera:

$$\xi_a(x) = |x - x_h|$$

    - El error absoluto permite medir qué tan lejos está el valor aproximado del valor real, proporcionando una escala de qué tan importante es el error

    - El error relativo, el cual normaliza esta diferencia, se define de la siguiente manera para $x \neq 0$:

$$\xi_r(x) = \left|\frac{x - x_h}{x}\right| = \frac{|x - x_h|}{|x|}$$

    - El error relativo es una medida que compara el tamaño del error con el tamaño del número a aproximar, permitiendo ver que tan significativo es el error realmente cuando se compara con el tamaño del número (no es lo mismo un error absoluto de 0.1 para 1000 que para 0.5)

    - Por lo tanto, el orden del error absoluto permite dar una visión sobre la exactitud de la aproximación al número deseado, mientras que el del error relativo da el número de dígitos correctos a través del exponente (negativo) de la base. Usar el error relativo permite no juzgar mal la exactitud de la aproximación debido a las escalas de los números a aproximar

$$\begin{cases} x = 0.1125351747 \times 10^{-6} \\ x' = 0 \end{cases} \Rightarrow \begin{cases} \xi_a(x) = 1.2 \times 10^{-7} \\ \xi_r(x) = 1 \end{cases}$$

$$\begin{cases} x = 0.8886110521 \times 10^7 \\ x' = 0.8886110517 \times 10^7 \end{cases} \Rightarrow \begin{cases} \xi_a(x) = 14 \times 10^{-3} \\ \xi_r(x) = 0.4501 \times 10^{-9} \end{cases}$$

- En las definiciones anteriormente vistas, se puede ver como se utiliza el suscrito h para sugerir que, por lo general, la aproximación depende de un parámetro. Para la mayor parte de los cálculos, estos estarán construidos de esta manera, teniendo un parámetro real h que tiende hacia cero o con un parámetro entero n el cual tiende hacia ∞

$$\lim_{h \to 0} A_h = A \qquad \lim_{n \to \infty} A_n = A$$

- En los problemas reales hay varios tipos de errores posibles, tales como los errores de medición, los errores de modelaje y otros. No obstante, el desarrollo se centra en los errores computacionales: los errores de truncamiento y los de redondeo, los cuales también se llaman errores matemáticos

- SI dos cantidades A y B son aproximadamente iguales, se usa la notación $A \approx B$ para denotar la relación. Aunque la noción de aproximación es vaga, se usa para uno de los dos contextos anteriores (un conjunto parametrizado de aproximaciones que en el límite tienden al número al que se aproxima)

  - Además, la aproximación satisface las propiedades de transitividad, simetría y reflexividad, de modo que son una relación de equivalencia:

$$A \approx B \ \& \ B \approx C \Rightarrow A \approx C$$

$$A \approx B \ \Leftrightarrow \ B \approx A$$

$$A \approx A$$

  - Consecuentemente, uno puede manipular las expresiones de equivalencia aproximada como las de equivalencia

- Otra notación que se suele utilizar de manera regular es la de la gran O o *Big O*, más formalmente llamada notación de orden asintótico

  - Suponiendo que se tiene un valor y y una familia de valores $\{y_h\}$, cada uno el cual aproxima el valor y (de modo que $y \approx y_h$ para valores pequeños de h). Si se puede encontrar una constante $C > 0$ independiente de h tal que $|y_h - h| \leq C\beta(h)$, donde $\beta(h)$ es una función tal que $\beta(h) \to 0$ cuando $h \to 0$, entonces se dice

que $y - y_h$ es del orden de $\beta(h)$, denotado de la siguiente manera:

$$y = y_h + O\big(\beta(h)\big) \quad as \quad h \to 0$$

- La utilidad de esta notación reside en que permite que uno se fije en un problema importante de la aproximación: la manera en que el error $y - y_h$ depende del parámetro h, que se determina por $\beta(h)$, mientras que se ignoran detalles como el tamaño preciso de C. Lo más importante es identificar en una expresión cuál es este parámetro que haría que la función tendiera a cero

- El uso es similar para secuencias $\{x_n\}$ que aproximan un valor x para valores grandes de n. Si $|x - x_n| \leq C\beta(n)$, donde $\beta(n)$ es una función tal que $\beta(n) \to 0$ cuando $n \to \infty$, entonces se dice que $x - x_n$ es del orden de $\beta(n)$, denotado de la siguiente manera:

$$x = x_h + O\big(\beta(n)\big) \quad as \quad n \to \infty$$

- Siendo $y = y_h + O\big(\beta(h)\big)$ y $z = z_h + O\big(\gamma(h)\big)$ cuando $b\beta(h) > \gamma(h)$ para toda h cerca de cero, entonces se cumplen las siguientes igualdades:

$$y + z = y_h + z_h + O\big(\beta(h) + \gamma(h)\big)$$

$$y + z = y_h + z_h + O\big(\beta(h)\big)$$

$$Ay = Ay_h + O\big(\beta(h)\big)$$

- Estos mismos resultados también aplican para el caso de una secuencia con parámetro entero n

- La demostración de la primera igualdad es la siguiente:

$$|(y + z) - (y_h + z_h)| = |(y - y_h) + (z - z_h)|$$

$$\leq |y - y_h| + |z - z_h| \leq C_1\beta(h) + C_2\gamma(h) \leq C\big(\beta(h) + \gamma(h)\big)$$

- La demostración de la segunda igualdad es la siguiente:

$$|(y + z) - (y_h + z_h)| = |(y - y_h) + (z - z_h)|$$

$$\leq |y - y_h| + |z - z_h| \leq C_1\beta(h) + C_2\gamma(h) \leq C\big(\beta(h) + b\beta(h)\big)$$

$$= C(1+b)\beta(h)$$

- La demostración de la tercera igualdad es la siguiente:

$$|Ay - Ay_h| = |A||y - y_h| \leq |A|C\beta(h)$$

El análisis de errores

- Los sistemas digitales como las calculadoras y computadores no suelen cometer errores, dado que siguen el orden programado al pie de la letra. No obstante, normalmente uno encuentra errores numéricos en los resultados de cálculos hechos con sistemas digitales

    - Evaluar la exactitud de los resultados de los cálculos es un objetivo primordial en el análisis numérico e importante para la aplicación de métodos numéricos. Uno puede distinguir varios tipos de errores que pueden limitar la exactitud, los cuales son errores en los datos de insumo, errores de redondeo o errores de aproximación

        - Los errores de insumo o de los datos están más allá del control humano en los cálculos. Estos pueden ser producidos, por ejemplo, por imperfecciones inherentes en las medidas físicas realizadas

        - Los errores de redondeo nacen si uno calcula con números cuya representación se restringe a un número finito de dígitos, como suele ser el caso

        - Muchos métodos no dan una solución exacta para un problema P, aunque los cálculos se lleven a cabo sin redondeo, sino que dan una solución de otro problema más simple $\tilde{P}$ que aproxima P. Un ejemplo puede ser el error de truncamiento que ocurre al usar una suma finita para aproximar una infinita, o el error de discretización que ocurre al aproximar una integral con una suma finita

    - Basándose en las maneras fundamentalmente diferentes de representar números, dos categorías de maquinaria computacional se pueden distinguir: las computadoras análogas y las computadoras digitales

        - Al utilizar las computadoras análogas, uno reemplaza números por cantidades físicas como la longitud de una barra o la intensidad del voltaje, de modo que se simula el problema matemático con uno físico, el cual se resuelve a través de la medición. Es obvio que la exactitud de los dispositivos análogos se limita por las medidas físicas que se emplean

- Las computadoras digitales expresan los dígitos de una representación numérica por una secuencia de cantidades físicas discretas, tales como calculadoras o computadoras digitales electrónicas. Cada dígito se representa con una cantidad física específica

- Debido a que hay un número finito pequeño de dígitos diferentes que se pueden codificar (en el sistema decimal), la representación de los dígitos en computadoras digitales no necesitar ser tan preciso como en el caso análogo. Por lo tanto, uno puede tolerar voltajes cercanos al número que se quiere representar, y, consecuentemente, la exactitud de las computadoras digitales no está limitada a la precisión de las medidas físicas

- Por razones técnicas, la mayoría de ordenadores digitales representan los números internamente en binario más que en forma decimal. En este caso, los coeficientes o bits α_i de una descomposición por potencias de 2 juegan el papel de dígitos en la representación de un número x

$$x = \pm(\alpha_n 2^n + \alpha_{n-1}2^{n-1} + \cdots + \alpha_0 2^0 + \alpha_{-1}2^{-1} + \alpha_{-2}2^{-2} + \cdots)$$

$$\alpha_i = 0 \quad or \quad \alpha_i = 1$$

  - Con tal de no confundir el sistema con el binario, la representación de los bits se hace con $\boldsymbol{L}$ y $\boldsymbol{0}$. Por ejemplo, el número $x = 18.5$ sería representado de la siguiente manera:

$$18.5 = 1 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 + 1 * 2^{-1}$$

$$\boldsymbol{L00L0.L}$$

  - La representación decimal de un número puede no ser única tanto en el sistema decimal como en el sistema binario, de modo que, para excluir estas ambigüedades, uno se refiere a esta representación como representación finita

  - Debido a como se representa un número en sistema binario, se puede ver como se tiene una serie infinita para representar un número, lo cual hará que el computador nunca sea capaz de guardar exactamente todos los números

- En general, las computadoras digitales deben conformarse con un número finito fijo de lugares o longitud de palabras (llamada *word length*) cuando representan internamente un número. Este número n se determina en el diseño de la máquina, aunque algunas máquinas tienen extensiones predeterminadas para múltiplos enteros $(2n, 3n, \ldots)$ de n

para ofrecer una mejor precisión si se necesita, y la longitud de palabras de n espacios se puede usar de diferentes maneras para representar

- La representación de punto fijo o *fixed-point representation* especifica un número n_1 de lugares antes del punto decimal o binario y un número n_2 después de este, de modo que $n = n_1 + n_2$ (normalmente $n_1 = 0$ o $n_1 = n$). En esta representación, la posición del decimal es fija

$$30.421 \rightarrow \boxed{0030}\boxed{421000}$$
$$0.0437 \rightarrow \underbrace{\boxed{0000}}_{n_1}\underbrace{\boxed{043700}}_{n_2}$$

- Las computadoras digitales suelen utilizar la representación de punto flotante o *floating-point representation*, en la cual el punto decimal o binario no está fijo, si no que su posición respecto al primer digito se indica para cada número separadamente. Esto se realiza especificando un exponente b que indica la posición del punto decimal con respecto a la *mantissa* a, que son los dígitos detrás del punto decimal (el número de desplazamientos del punto decimal para llegar a la representación original)

$$x = a \times 10^b \quad with \ |a| < 1 \ \& \ b \in \mathbb{Z} \quad (decimal)$$

$$x = a \times 2^b \quad with \ |a| < 1 \ \& \ b \in \mathbb{Z} \quad (binary)$$

- Rutishauser propuso la notación semilogarítmica para esta representación, que enseña la base del sistema numérico en el nivel de suscrito y mueve el exponente al nivel de la *mantissa*

$$x = 30.421 \ \Rightarrow \ 0.30421_{10}2 \quad (decimal)$$

$$x = 18.5 \quad \Rightarrow \quad \boldsymbol{0.L00L0L_2L0L} \quad (binary)$$

- En cualquier computadora hay solo un número fijo finito de dígitos en la *mantissa* t, de dígitos en el exponente e, y de dígitos disponibles totales o *word length* $n = t + e$

$$0\ \boxed{5420}_{10}\ \boxed{04} \quad \text{or more concisely} \quad \boxed{5420}\boxed{04}$$

$$0\ \boxed{0542}_{10}\ \boxed{05} \quad \text{or} \quad \boxed{0542}\boxed{05}$$

- La representación de punto flotante, no obstante, no necesita ser única, dado que hay diferentes combinaciones que harían posible la representación del mismo número

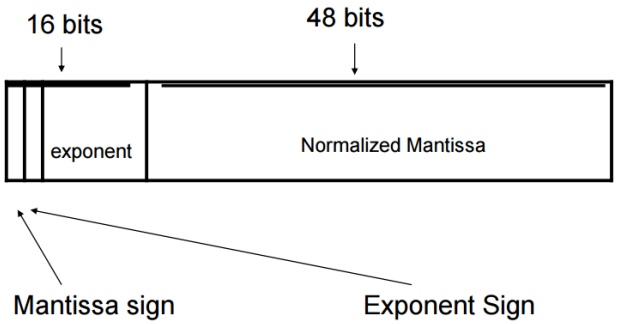
$$0.5420 \times 10^4 = 0.05420 \times 10^5 = \cdots$$

- Una representación de punto flotante se normaliza si el primer dígito o bit de la *mantissa* es diferente de cero o **0**, de modo que $|a| \geq 10^{-1}$ (porque $|a| < 10^{-1}$ comienza con un cero) o $|a| \geq 2^{-1}$ (porque $|a| < 2^{-1}$ comienza con **0**). Los dígitos o bits significativos de un número son los dígitos o bits de la *mantissa* sin contar los ceros líderes

$$(digits) \quad 10^{-1} \leq |a| < 1$$

$$(bits) \quad 2^{-1} \leq |a| < 1$$

- La representación detallada de los números de punto flotante dependerá en la arquitectura de la máquina que se esté utilizando. Asumiendo una arquitectura digital de 64 bits, el espacio se reparte de la siguiente manera:



Sign of q	1 bit
Sign of m	1 bit
Integer $\|q\|$	14 bits
Integer m	48 bits

- Con estas restricciones, el exponente máximo que se puede representar en una arquitectura de 64 bits es el 2^{14} (última posición posible para el exponente) menos 1

$$2^{14} - 1 = 16384 - 1 = 16383$$

- El valor absoluto del menor número que se puede representar por la máquina tiene la siguiente *mantissa*:

$$a = 2^{-48} \approx 0.3553 \times 10^{-14}$$

- Por lo tanto, hay números reales que no se pueden representar con esta *word structure* (se tienen que descartar dígitos a partir de la posición 48, la última permitida para la *mantissa* normalizada):

$$x = 0.\boldsymbol{L}\alpha_2 \ldots . \alpha_{48}\alpha_{49}\alpha_{50} \ldots \times 2^b \quad for \quad \alpha_t = \mathbf{0}\ or\ \boldsymbol{L}$$

$$\Rightarrow \quad x = 0.\boldsymbol{L}\alpha_2 \ldots . \alpha_{48} \times 2^b \quad for \quad \alpha_t = \mathbf{0}\ or\ \boldsymbol{L}$$

- El conjunto $A \subseteq \mathbb{R}$ de números reales que son representables en una maquina concreta (números de máquina o *machine nombres*) es finito, de modo que es interesante saber cómo aproximar un número $x \notin A$ con un número $g \in A$. Este problema se encuentra no solo al leer datos en una computadora, sino también en cálculos intermedios

    - Es natural postular que la aproximación de cualquier número $x \notin A$ por un número de máquina $rd(x) \in A$ debería satisfacer la siguiente desigualdad:

    $$|x - rd(x)| \leq |x - g| \quad for \quad \forall g \in A$$

        - Esta desigualdad expresa que el error absoluto $|x - rd(x)|$, tiene que ser menor o igual a la distancia entre x y cualquier otro número g

        - Esta aproximación $rd(x)$ se puede obtener en la mayoría de casos con el redondeo, en donde se redondea el número para que tengan s números decimales en la *mantissa*

        $$rd(0.14285_{10}0) = 0.1429_{10}0$$

        $$rd(3.14159_{10}0) = 0.3142_{10}1$$

        $$rd(0.142842_{10}2) = 0.1428_{10}0$$

        - Otra manera que se puede usar para aproximar el número x a otro x' es definiendo x' como un valor truncado, de modo que se trunca el valor a partir de un número de dígito en la *mantissa*:

        $$x = 0.\alpha_1\alpha_2 \ldots . \alpha_t\alpha_{t+1}\alpha_{t+2} \ldots \times 10^b \quad (decimal)$$

        $$\Rightarrow\ x' = 0.\alpha_1\alpha_2 \ldots . \alpha_t \times 10^b$$

        $$x = 0.\alpha_1\alpha_2 \ldots . \alpha_t\alpha_{t+1}\alpha_{t+2} \ldots \times 2^b \qquad (binary)$$

        $$\Rightarrow\ x' = 0.\alpha_1\alpha_2 \ldots . \alpha_t \times 2^b$$

    - En general, uno puede proceder de la siguiente manera para encontrar $rd(x)$ para una computadora de t dígitos (número de dígitos significativos en el número), las cuales sirven como normas para el truncamiento y el redondeo:

- El número $x \notin A$ se representa de forma normalizada $x = a \times 10^b$, de modo que $|a| \geq 10^{-1}$. Se supone que la representación decimal de $|a|$ viene dada por la siguiente expresión para una t predeterminada:

$$|a| = 0.\alpha_1\alpha_2 \ldots . \alpha_t\alpha_{t+1} \ldots \quad for \;\; 0 \leq \alpha_t \leq 9 \;\; \& \;\; \alpha_1 \neq 0$$

- A partir de la suposición anterior, se puede formar a' incrementando α_t por 1 si el dígito $(t+1)$ es mayor a 5 (a través del mecanismo que se explica abajo) y eliminando los dígitos después del dígito t, permitiendo definir $\widetilde{rd}(x)$

$$a' \equiv \begin{cases} 0.\alpha_1\alpha_2 \ldots . \alpha_t & if \;\; 0 \leq \alpha_{t+1} < 5 \\ 0.\alpha_1\alpha_2 \ldots \alpha_t + 10^{-t} & if \;\; \alpha_{t+1} \geq 5 \end{cases} \quad for\ a\ given\ t$$

$$\widetilde{rd}(x) \equiv sign(x) \times a' 10^b$$

$$\Rightarrow \;\; \widetilde{rd}(x) = sign(x)(0.\alpha_1\alpha_2 \ldots \alpha_t + 10^{-t})10^b$$

$$\Rightarrow \;\; \widetilde{rd}(x) = sign(x)(0.\alpha_1\alpha_2 \ldots \alpha_t 10^b + 10^{b-t})$$

- En binario, todo se define de manera análoga, obteniendo los mismos resultados con ligeras modificaciones:

$$|a| = 0.\alpha_1\alpha_2 \ldots . \alpha_t\alpha_{t+1} \ldots \quad for \;\; \alpha_t = \mathbf{0}\ or\ \boldsymbol{L} \;\; \& \;\; \alpha_1 = \boldsymbol{L}$$

$$a' \equiv \begin{cases} 0.\alpha_1\alpha_2 \ldots . \alpha_t & if \;\; \alpha_{t+1} = \mathbf{0} \\ 0.\alpha_1\alpha_2 \ldots . \alpha_t + 2^{-t} & if \;\; \alpha_{t+1} = \boldsymbol{L} \end{cases} \quad for\ a\ given\ t$$

$$\widetilde{rd}(x) \equiv sign(x) \times a' 2^b$$

$$\Rightarrow \;\; \widetilde{rd}(x) = sign(x)(0.\alpha_1\alpha_2 \ldots \alpha_t + 2^{-t})2^b$$

$$\Rightarrow \;\; \widetilde{rd}(x) = sign(x)(0.\alpha_1\alpha_2 \ldots \alpha_t 2^b + 2^{b-t})$$

- Como se puede deducir, por tanto, se cumple que $t = b + s$. El número de dígitos significativos t es determinado por el exponente b que indica el número de dígitos en la parte entera del número (a cuanto se debería elevar para tener el número redondeado) y por el número s de decimales que se incluyen en la representación redondeada

$$rd(42.876589) = 42.8766 = 0.428766 \times 10^2$$

$$\Rightarrow \; t = 6, b = 2, s = 4$$

- Cuando $\widetilde{rd}(x) \in A$ es un número de máquina, entonces $\widetilde{rd}$ cumple con la desigualdad de un proceso de redondeo correcto, y se puede definir lo siguiente:

$$\widetilde{rd}(x) \equiv rd(x) \quad for\ \forall x\ with\ \widetilde{rd}(x) \in A$$

  - Como solo un número finito e de lugares están disponibles para expresar el exponente en la representación de punto flotante, siempre habrá números $x \notin A$ con $rd(x) \notin A$. Los casos más comunes son los de desbordamiento de exponente o *exponent overflow* (el número es demasiado positivo y supera e) y los de flujo inferior de exponente o *exponent underflow* (el número es demasiado negativo y supera e)

  - Asumiendo que $t = 4$ y $e = 2$, los casos (a) y (b) hay desbordamiento (en este segundo caso, el desbordamiento solo ocurre después de redondear), mientras que (c) y (d) son casos de flujo inferior

$$(a)\ \widetilde{rd}(0.31794_{10}110) = 0.3179_{10}110 \notin A$$

$$(b)\ \widetilde{rd}(0.99997_{10}99) = 0.1000_{10}100 \notin A$$

$$(c)\ \widetilde{rd}(0.012345_{10} - 99) = 0.1235_{10} - 100 \notin A$$

$$(d)\ \widetilde{rd}(0.54321_{10} - 110) = 0.5432_{10} - 110 \notin A$$

- Las computadoras digitales tratan estas ocurrencias como irregularidades en el cálculo. En el caso en el que haya un flujo inferior, entonces $rd(x)$ se suele definir de manera alternativa (pero no cumple con la cota superior anteriormente vista), pero los cálculos se suelen parar en el caso de desbordamiento

  - Si hay flujo inferior, una manera de evitarlo es definiendo el redondeo de manera que el exponente se mantenga dentro de los límites, pero sacrificando la cota superior del error relativo (uno se puede equivocar más). Un ejemplo de cómo se podría definir el redondeo sería el siguiente:

$$rd(0.012345_{10} - 99) = 0.0123_{10} - 99 \notin A$$

$$rd(0.54321_{10} - 110) = 0 \notin A$$

  - En el resto de casos regulares, el redondeo se define de la siguiente manera:

$$rd(x) \equiv \widetilde{rd}(x)$$

- Los casos de desbordamiento y flujo inferior se pueden evitar en alguna medida con escalamiento adecuado de los datos de insumo e incorporando evaluaciones y escalamientos especiales durante los cálculos

- Debido a que no existe una descripción numérica exacta del error (si no, no sería necesaria la aproximación en ningún sentido), se suele utilizar la cota superior para el tamaño del error (ya sea absoluto o relativo) para poder analizar la exactitud o *accuracy* de la aproximación

    - Cuando se trunca un número x, se pueden establecer las siguientes cotas para los errores de truncamiento

        - El valor exacto se representa como $x = a \times 10^b$, siendo $a = 0.\alpha_1\alpha_2 \ldots . \alpha_t\alpha_{t+1} \ldots$ como antes, mientras que el valor aproximado por truncamiento se representa como $x' = a' \times 10^b$ donde $a' = 0.\alpha_1\alpha_2 \ldots . \alpha_t$ (se corta hasta el digito significativo t). Como la *mantissa* es una representación de punto fijo, se puede establecer la siguiente cota para el error absoluto de la *mantissa*:

$$\xi_a(a) = |a - a'| = |0.\alpha_1\alpha_2 \ldots . \alpha_t\alpha_{t+1} \ldots - 0.\alpha_1\alpha_2 \ldots . \alpha_t| =$$

$$= 0.00 \ldots 0\alpha_{t+1}\alpha_{t+2} \ldots$$

$$\Rightarrow \mathcal{E}_a(a) = |a - a'| \leq 0.00 \ldots 01 = 10^{-t}$$

        - En consecuencia, como $|x - x'| = |a - a'| \times 10^b$, se puede plantear la siguiente cota superior para el error absoluto entre x y x':

$$\xi_a(x) = |x - x'| = |a - a'| \times 10^b \leq 10^{b-t} = \mathcal{E}_a(x)$$

        - Lo mismo ocurre en el caso de utilizar una representación binaria, se obtienen resultados similares:

$$|a - a'| = |0.\alpha_1\alpha_2 \ldots . \alpha_t\alpha_{t+1} \ldots - 0.\alpha_1\alpha_2 \ldots . \alpha_t| =$$

$$= 0.\mathbf{00} \ldots \mathbf{0}\alpha_{t+1}\alpha_{t+2} \ldots$$

$$\Rightarrow \xi_a(a) = |a - a'| \leq 0.\mathbf{00} \ldots \mathbf{0L} = 2^{-t} = \mathcal{E}_a(a)$$

$$\Rightarrow \xi_a(x) = |x - x'| = |a - a'| \times 2^b \leq 2^{b-t} = \mathcal{E}_a(x)$$

        - Debido a que $|a| > 10^{-1}$, el error relativo absoluto de x' admite la siguiente cota superior:

$$\xi_r(x) = \left|\frac{x' - x}{x}\right| \sim \left|\frac{x' - x}{x'}\right|$$

$$\Rightarrow \xi_r(x) = \left|\frac{x' - x}{x'}\right| \leq \frac{10^{b-t}}{|a'| \times 10^b} = \frac{10^{-t}}{|a'|} \leq \frac{10^{-t}}{10^{-1}} = 10^{1-t} =$$

$$= \mathcal{E}_r(x)$$

- En el caso de una representación binaria, debido a que $|a| > 2^{-1}$, el error relativo absoluto de x' admite la siguiente cota superior:

$$\xi_r(x) = \left|\frac{x' - x}{x}\right| \sim \left|\frac{x' - x}{x'}\right|$$

$$\Rightarrow \xi_r(x) = \left|\frac{x' - x}{x'}\right| \leq \frac{2^{b-t}}{|a'| \times 2^b} = \frac{2^{-t}}{|a'|} \leq \frac{2^{-t}}{2^{-1}} = 2^{1-t} = \mathcal{E}_r(x)$$

- Si, en vez de truncar, se lleva a cabo el proceso de redondeo y truncamiento para los números que se ha visto antes, se obtienen los siguientes resultados:

  - El valor exacto se representa como $x = a \times 10^b$, siendo $a = 0.\alpha_1\alpha_2 \ldots . \alpha_t\alpha_{t+1} \ldots$ como antes, mientras que el valor aproximado por redondeo se representa como $x' = a' \times 10^b$ donde $a' = 0.\alpha_1\alpha_2 \ldots . \alpha_t$ (se corta hasta el digito significativo t). Como la *mantissa* es una representación de punto fijo, se puede establecer la siguiente cota para el error absoluto de la *mantissa*:

$$As \begin{cases} a \geq \widetilde{rd}(a) & if \ \ \alpha_{t+1} < 5 \\ a \leq \widetilde{rd}(a) & if \ \ \alpha_{t+1} \geq 5 \end{cases}$$

$$\Rightarrow \begin{cases} \left|a - \widetilde{rd}(a)\right| \geq 0.0 \ldots \alpha_{t+1} \ldots & if \ \ \alpha_{t+1} < 5 \\ \left|a - \widetilde{rd}(a)\right| \geq 0.0 \ldots (9 - \alpha_{t+1}) \ldots & if \ \ \alpha_{t+1} \geq 5 \end{cases}$$

$$\Rightarrow \xi_a(a) = \left|a - \widetilde{rd}(a)\right| \leq 0.00 \ldots 05 = 0.5 \times 10^{-t} =$$

$$= 5 \times 10^{-(t+1)} = \mathcal{E}_a(a)$$

  - En consecuencia, como $\left|x - \widetilde{rd}(x)\right| = \left|a - \widetilde{rd}(a)\right| \times 10^b$, se puede plantear la siguiente cota superior para el error absoluto entre x y x', que será la mitad que para el truncamiento:

$$\xi_a(x) = \left|x - \widetilde{rd}(x)\right| = \left|a - \widetilde{rd}(a)\right| \times 10^b$$

$$\leq \left(5 \times 10^{-(t+1)}\right)10^b =$$

$$= 5 \times 10^{b-(t+1)} = \mathcal{E}_a(x)$$

- Lo mismo ocurre en el caso de utilizar una representación binaria, se obtienen resultados similares, pero esta vez, el error absoluto es igual que el de truncamiento:

$$As \begin{cases} a \geq \widetilde{rd}(a) & if \ \alpha_{t+1} = \mathbf{0} \\ a \leq \widetilde{rd}(a) & if \ \alpha_{t+1} = \boldsymbol{L} \end{cases}$$

$$\Rightarrow \begin{cases} \left|a - \widetilde{rd}(a)\right| \geq 0.\mathbf{0} \ldots \mathbf{0}\alpha_{t+2} \ldots & if \ \alpha_{t+1} = \mathbf{0} \\ \left|a - \widetilde{rd}(a)\right| \geq 0.\mathbf{0} \ldots \mathbf{0}\alpha_{t+2} \ldots & if \ \alpha_{t+1} = \boldsymbol{L} \end{cases}$$

$$\Rightarrow \ \xi_a(a) = \left|a - \widetilde{rd}(a)\right| \leq 0.\mathbf{0}\mathbf{0} \ldots \mathbf{0}\boldsymbol{L} = 2^{-t} = \mathcal{E}_a(a)$$

$$\Rightarrow \xi_a(x) = \left|x - \widetilde{rd}(x)\right| = \left|a - \widetilde{rd}(a)\right| \times 2^b \leq 2^{-t}2^b = 2^{b-t} =$$

$$= \mathcal{E}_a(x)$$

- Debido a que $|a| > 10^{-1}$, el error relativo absoluto de $\widetilde{rd}(x)$ admite la siguiente cota (error de redondeo de la máquina), la cual se define como $\mathcal{E}_r(x) \equiv 5 \times 10^{-t}$ y se denomina la precisión de la máquina:

$$\xi_r(x) = \left|\frac{\widetilde{rd}(x) - x}{x}\right| \sim \left|\frac{\widetilde{rd}(x) - x}{\widetilde{rd}(x)}\right|$$

$$\Rightarrow \ \xi_r(x) = \left|\frac{\widetilde{rd}(x) - x}{x}\right| \sim \left|\frac{\widetilde{rd}(x) - x}{\widetilde{rd}(x)}\right| \leq \frac{5 \times 10^{b-(t+1)}}{\left|\widetilde{rd}(a)\right| \times 10^b} =$$

$$= \frac{5 \times 10^{-(t+1)}}{\left|\widetilde{rd}(a)\right|} \leq \frac{5 \times 10^{-(t+1)}}{10^{-1}} = 5 \times 10^{-t} \equiv eps$$

$$\Rightarrow \ \widetilde{rd}(x) \equiv x(1 + \varepsilon) \quad where \ \ |\varepsilon| \leq eps$$

- En el caso de una representación binaria, debido a que $|a| > 2^{-1}$, el error relativo absoluto de $\widetilde{rd}(x)$ admite la siguiente cota (error de redondeo de la máquina), la cual se define como $\mathcal{E}_r(x) = 2^{-t}$ y se denomina la precisión de la máquina:

$$\xi_r(x) = \left|\frac{\widetilde{rd}(x) - x}{x}\right| \sim \left|\frac{\widetilde{rd}(x) - x}{\widetilde{rd}(x)}\right|$$

$$\Rightarrow \xi_r(x) = \left|\frac{\widetilde{rd}(x) - x}{x}\right| \leq \frac{2^{b-(t+1)}}{\left|\widetilde{rd}(a)\right| \times 2^b} = \frac{2^{-(t+1)}}{\left|\widetilde{rd}(a)\right|}$$

$$\leq \frac{2^{-(t+1)}}{2^{-1}} = 2^{-t} \equiv \mathcal{E}_r(x)$$

$$\Rightarrow \ \widetilde{rd}(x) \equiv x(1+\varepsilon) \ \ where \ \ |\varepsilon| \leq eps$$

- Con una arquitectura de 64 bits, entonces $t = 48$, y se obtiene la siguiente desigualdad:

$$\xi_r(x) = \left|\frac{\widetilde{rd}(x) - x}{x}\right| \leq 2^{-48} \approx 0.36 \times 10^{14} \equiv \ eps$$

- Como cada método numérico requiere sus técnicas de protección especiales y los desbordamientos y flujos inferiores no suelen pasar frecuentemente, se suele hacer la suposición de que $e = \infty$, de modo que $rd \equiv \widetilde{rd}$ proporciona una regla que hace que se cumplan las siguientes condiciones:

$$rd : \mathbb{R} \to A$$

$$rd(x) = x(1+\varepsilon) \ \ with \ \ |\varepsilon| \leq \mathcal{E}_r(x) \ \ \ for \ \ \forall x \in \mathbb{R}$$

- En otros ejemplos se dará una longitud t de la *mantissa*, pero uno tiene que tener en mente que resultados subsiguientes de los errores de redondeo pueden ser inválidos si los desbordamientos y flujos inferiores de exponentes están permitidos

- Los resultados de las operaciones aritméticas $x \pm y$, x/y p $x \times y$ no son necesariamente números de máquina, aunque x e y lo sean, por lo que uno no puede esperar reproducir exactamente las operaciones aritméticas en una computadora

  - Uno se tiene que contentar con las operaciones sustitutas $+^*, -^*, \times^*$ y $/^*$, llamadas operaciones de punto flotante, que aproximan las operaciones aritméticas de la mejor manera posible. Estas operaciones se pueden definir, por ejemplo, con el mapa de redondeo rd de la siguiente manera:

$$\begin{cases} x +^* y \equiv rd(x+y) \\ x -^* y \equiv rd(x-y) \\ x \times^* y \equiv rd(x \times y) \\ x /^* y \equiv rd(x/y) \end{cases} \ \ for \ \ x, y \in A$$

$$\Rightarrow \begin{cases} x +^* y = (x+y)(1+\varepsilon_1) \\ x -^* y = (x-y)(1+\varepsilon_2) \\ x \times^* y = (x \times y)(1+\varepsilon_3) \\ x /^* y = (x/y)(1+\varepsilon_4) \end{cases} \ \ for \ \ |\varepsilon_i| \leq eps$$

- En muchos programas, las operaciones de punto flotante no se definen de esta manera, sino que se definen de modo que las implicaciones anteriores se mantienen para una cota débil como $|\varepsilon_i| \leq k(eps)$ para un entero pequeño $k \geq 1$. Debido a que pequeñas desviaciones de las implicaciones no son significativas para las exposiciones, se asume por simplicidad la definición anterior

- Para expresar el resultado de los cálculos de punto flotante, una notación conveniente pero un poco imprecisa que se usa es la siguiente: si el contexto deja claro como evaluar una expresión aritmética E, entonces $fl(E)$ denota el valor de la expresión obtenido por aritmética de punto flotante

$$fl(x \times y) \equiv x \times^* y$$

$$fl\big(a + (b + c)\big) \equiv a +^* (b +^* c)$$

$$fl\big((a + b) + c\big) \equiv (a +^* b) +^* c$$

- También se usa la notación $fl\big(\sqrt{x}\big)$, $fl(\cos(x))$ y otras cuando la computadora digital aproxime las funciones con los sustitutos de punto flotante correspondientes. Todas estas operaciones aritméticas y funciones básicas que tienen sustitutos de punto flotante se denominan operaciones elementales

- Cuando se sustraen dos números $\tilde{x}$ y $\tilde{y}$ que aproximan $x, y \in A$ con el mismo signo, hay que tener en cuenta la cancelación catastrófica, que es el fenómeno que ocurre cuando x e y coinciden en uno o más dígitos líderes con respecto al mismo exponente, haciendo que estos dígitos comunes desaparezcan y haciendo que el error de la aproximación a esta diferencia sea más grande que el real

$$\begin{cases} \hat{x} = x[1 + \xi_r(\hat{x})] \\ \hat{y} = y[1 + \xi_r(\hat{y})] \end{cases} \Rightarrow \hat{x} - \hat{y} = x - y + x\xi_r(\hat{x}) - y\xi_r(\hat{y})$$

$$\Rightarrow \hat{x} - \hat{y} = x - y + (x - y)\frac{x\xi_r(\hat{x}) - y\xi_r(\hat{y})}{x - y}$$

$$\Rightarrow \hat{x} - \hat{y} = (x - y)\left[1 + \frac{x\xi_r(\hat{x}) - y\xi_r(\hat{y})}{x - y}\right]$$

$$\Rightarrow \xi_r(\hat{x} - \hat{y}) = \left|\frac{(\hat{x} - \hat{y}) - (x - y)}{x - y}\right| = \left|\frac{x\xi_r(\hat{x}) - y\xi_r(\hat{y})}{x - y}\right|$$

$$\Rightarrow \lim_{x \to y} \left| \frac{x\xi_r(\hat{x}) - y\xi_r(\hat{y})}{x - y} \right| = \infty$$

- Lo que muestra el resultado es que, cuando los números x e y están muy cerca (varios dígitos significativos coinciden), entonces el error relativo tiende a ser arbitrariamente grande. Que el error relativo tienda a ser muy grande significa que el error de la aproximación, considerando el tamaño de la resta, es bastante grande, por lo que la aproximación es poco fiable

- El resultado exacto $x - y \in A$, de modo que no hay error de redondeo porque $x - y$ sería igual a $x -^{*} y$ al ser de máquina. La cancelación catastrófica solo ocurre cuando hay un error de redondeo o de aproximación para x e y (cuando se considera una resta de números aproximados)

- Este tipo de problemas es muy relevante cuando se necesitan resultados pequeños, dado que se estaría usando una aproximación "ruidosa" en vez de un valor preciso y alteraría los resultados de los cálculos subsiguientes

- Las mejores maneras de poder evitar la cancelación cuando se quieren utilizar expresiones complejas como fracciones racionales, logaritmos, funciones trigonométricas y otros suelen ser tres: racionalizar un número para evitar la resta de dos números parecidos, hacer expansiones de Taylor (apoyándose en el teorema de aproximación de Weierstrass) o expandir un polinomio o expresión polinómica

  - Racionalizando un número, se puede evitar la resta de dos números muy parecidos en alguna parte de la expresión:

$$f(x) = \sqrt{x^2 + 1} - 1 \quad for \quad x \approx 1$$

$$\Rightarrow \quad f(x) = \left(\sqrt{x^2 + 1} - 1 \right) \frac{\sqrt{x^2 + 1} + 1}{\sqrt{x^2 + 1} + 1} = \frac{x^2}{\sqrt{x^2 + 1} + 1}$$

  - Expandiendo un polinomio, se puede evitar la resta de dos números muy parecidos en la forma simple de la expresión:

$$f(x) = (x - 1)^7 \quad for \quad x \approx 1$$

$$\Rightarrow \quad f(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + \cdots$$

  - Cuando hay funciones que no son aritméticas, estas se pueden aproximar con una expansión de Taylor:

$$f(2) = \ln(2) \Rightarrow f(x) = \sum_{k=0}^{n} \frac{1}{k!}[\ln(x)]^{(k)}(x-1)^k + R_n(x)$$

$$where \quad R_n(x) = \frac{1}{n+1}[\ln(c)]^{(n+1)}(x-1)^{n+1}$$

- Con $R_n(x)$ es posible fijar una cota máxima para el error de precisión de t dígitos y también saber el número de términos necesarios para obtener esta precisión

$$|R_n(x)| = \frac{1}{n+1} f^{(n+1)}(c)(x-x_0)^{n+1} \leq \frac{1}{n+1} \leq 10^{-t}$$

$$\Rightarrow \quad n+1 \geq 10^t$$

- Uno de los conceptos más importantes, relacionado con la arquitectura de la computadora y con el sistema de números de punto flotante es el de la épsilon de la máquina

  - En un sistema computacional de punto flotante, existen muchos números $x \neq 0$ tales que $1 + x = 1$ dentro de la precisión de la computadora. Para una implementación de 64 bits, esto ocurre para cualquier $x < 2^{-48}$ porque $1 + 2^{-48}$ en binario requeriría el uso de 49 bits

  - Se define la épsilon de la máquina como aquel número más grande x que cumple esta propiedad (donde $1 + x$ no se puede distinguir de x para la computadora)

$$eps = \max\{x | 1 + x = 1 \ \ on\ computer\ arithmetic\}$$

  - Es posible calcular la épsilon de la máquina basándose en conocimiento del sistema de números de cosas flotantes de la computadora en cuestión

- La existencia de la épsilon de la máquina conlleva unas cuantas consecuencias interesantes para la aritmética computacional

  - Una de las consecuencias más relevantes es que el orden de las operaciones a veces importa. Desde un punto de vista técnico, esto quiere decir que la ley asociativa de la suma no se mantiene para sistemas numéricos aritméticos

$$1 + (eps + eps) = 1 + 2eps$$

$$(1 + eps) + eps = 1 + eps = 1$$

- Debido a esto, es posible estimar el valor la épsilon de la máquina construyendo un bucle que añade números a 1 cada vez más pequeños, y que solo termine cuando el resultado no puede distinguirse de 1

- Además de la propiedad asociativa, tampoco se cumple la propiedad distributiva. Esto significa que expresar de manera compacta una multiplicación puede afectar al resultado

$$eps(1 + eps) \neq eps + eps^2$$

$$eps(1 + eps) = eps$$

- Siendo $*$ una operación binaria básica (suma, resta, multiplicación, división), entonces existe una constante C tal que se cumple la siguiente desigualdad:

$$|(x * y) - fl(x * y)| \leq C\ eps|x * y|$$

- Por lo tanto, el valor que la computadora asigna a $x * y$ es relativamente preciso para $O(eps)$

$$x * y = fl(x * y) + O(eps)$$

- A partir de los errores absolutos y los errores relativos absolutos que se han visto anteriormente, es posible obtener los errores absolutos y absolutos relativos para las diferentes operaciones aritméticas y para funciones de una sola variable

  - Es posible encontrar las cotas superiores del error absoluto, denotadas por $\mathcal{E}_a$, y del error absoluto relativo, denotadas por $\mathcal{E}_r$, para las diferentes operaciones aritméticas. Para todas, se considera una aproximación con un error absoluto relativo ξ_a y con un error relativo ξ_r

    - Las operaciones de suma y resta tienen las siguientes cotas superiores absolutas y relativas:

$$\begin{cases} \hat{x} = x + \xi_a(\hat{x}) \\ \hat{y} = y + \xi_a(\hat{y}) \end{cases} \Rightarrow \hat{x} \pm \hat{y} = (x \pm y) + \left(\xi_a(\hat{x}) \pm \xi_a(\hat{y})\right)$$

$$\Rightarrow \xi_a(\hat{x} \pm \hat{y}) = |(\hat{x} \pm \hat{y}) - (x \pm y)| = (\hat{x} \pm \hat{y}) - (x \pm y)$$

$$= \xi_a(\hat{x}) \pm \xi_a(\hat{y}) \leq \xi_a(\hat{x}) + \xi_a(\hat{y}) \leq \mathcal{E}_a(\hat{x}) + \mathcal{E}_a(\hat{y}) =$$

$$= \mathcal{E}_a(\hat{x} \pm \hat{y})$$

$$\Rightarrow \xi_r(\hat{x} \pm \hat{y}) = \frac{\xi_a(\hat{x} \pm \hat{y})}{x \pm y} = \frac{\xi_a(\hat{x}) \pm \xi_a(\hat{y})}{x \pm y} \leq \frac{\mathcal{E}_a(\hat{x} \pm \hat{y})}{|x \pm y|}$$

$$\leq \frac{\mathcal{E}_a(\hat{x}) + \mathcal{E}_a(\hat{y})}{|x \pm y|} = \mathcal{E}_r(\hat{x} \pm \hat{y})$$

$$\begin{cases} \xi_a(\hat{x} \pm \hat{y}) = \xi_a(\hat{x}) \pm \xi_a(y); \ \mathcal{E}_a(\hat{x} \pm \hat{y}) = \mathcal{E}_a(\hat{x}) + \mathcal{E}_a(\hat{y}) \\ \xi_r(\hat{x} \pm \hat{y}) = \dfrac{\xi_a(\hat{x}) + \xi_a(\hat{y})}{x \pm y}; \ \mathcal{E}_r(\hat{x} \pm \hat{y}) = \dfrac{\mathcal{E}_a(\hat{x}) + \mathcal{E}_a(\hat{y})}{|x \pm y|} \end{cases}$$

- La operación de multiplicación tiene las siguientes cotas superiores absolutas y relativas:

$$\begin{cases} \hat{x} = x + \xi_a(\hat{x}) \\ \hat{y} = y + \xi_a(\hat{y}) \end{cases} \Rightarrow \hat{x}\hat{y} = \big(x + \xi_a(\hat{x})\big)\big(y + \xi_a(\hat{y})\big) =$$

$$= xy + x\xi_a(\hat{y}) + y\xi_a(\hat{x}) + \xi_a(\hat{x})\xi_a(\hat{y})$$

$$\Rightarrow \xi_a(\hat{x}\hat{y}) = |\hat{x}\hat{y} - xy| = \hat{x}\hat{y} - xy =$$

$$= x\xi_a(\hat{y}) + y\xi_a(\hat{x}) + \xi_a(\hat{x})\xi_a(\hat{y}) \leq$$

$$\leq |x\xi_a(\hat{y}) + y\xi_a(\hat{x}) + \xi_a(\hat{x})\xi_a(\hat{y})| =$$

$$\leq |x|\xi_a(\hat{y}) + |y|\xi_a(\hat{x}) + \xi_a(\hat{x})\xi_a(\hat{y}) \leq$$

$$\leq |x|\mathcal{E}_a(\hat{y}) + |y|\mathcal{E}_a(\hat{x}) + \mathcal{E}_a(\hat{x})\mathcal{E}_a(\hat{y}) = \mathcal{E}_a(\hat{x}\hat{y})$$

$$\Rightarrow \mathcal{E}_a(\hat{x}\hat{y}) \approx |x|\mathcal{E}_a(\hat{y}) + |y|\mathcal{E}_a(\hat{x})$$

$$\Rightarrow \xi_r(\hat{x}\hat{y}) = \frac{\xi_a(\hat{x}\hat{y})}{xy} = \frac{\xi_a(\hat{x})}{x} + \frac{\xi_a(\hat{y})}{y} + \frac{\xi_a(\hat{x})}{x}\frac{\xi_a(\hat{y})}{y} =$$

$$= \xi_r(\hat{x}) + \xi_r(\hat{y}) + \xi_r(\hat{x})\xi_r(\hat{y}) \leq \mathcal{E}_r(\hat{x}) + \mathcal{E}_r(\hat{y}) + \mathcal{E}_r(\hat{x})\mathcal{E}_r(\hat{y})$$

$$= \mathcal{E}_r(\hat{x}\hat{y})$$

$$\Rightarrow \mathcal{E}_r(\hat{x}\hat{y}) \approx \mathcal{E}_r(\hat{x}) + \mathcal{E}_r(\hat{y})$$

$$\begin{cases} \xi_a(\hat{x}\hat{y}) \approx x\xi_a(\hat{y}) + y\xi_a(\hat{x}); \ \mathcal{E}_a(\hat{x}\hat{y}) \approx |x|\mathcal{E}_a(\hat{y}) + |y|\mathcal{E}_a(\hat{x}) \\ \xi_r(\hat{x}\hat{y}) \approx \xi_r(\hat{x}) + \xi_r(\hat{y}); \ \mathcal{E}_r(\hat{x}\hat{y}) \approx \mathcal{E}_r(\hat{x}) + \mathcal{E}_r(\hat{y}) \end{cases}$$

- La operación de división tiene las siguientes cotas superiores absolutas y relativas:

$$\begin{cases}\hat{x} = x + \xi_a(\hat{x}) \\ \hat{y} = y + \xi_a(\hat{y})\end{cases} \Rightarrow \frac{\hat{x}}{\hat{y}} = \frac{x + \xi_a(\hat{x})}{y + \xi_a(\hat{y})}$$

$$\Rightarrow \xi_a\left(\frac{\hat{x}}{\hat{y}}\right) = \left|\frac{\hat{x}}{\hat{y}} - \frac{x}{y}\right| = \frac{\hat{x}}{\hat{y}} - \frac{x}{y} = \frac{x + \xi_a(\hat{x})}{y + \xi_a(\hat{y})} - \frac{x}{y}$$

$$= \frac{xy + \xi_a(\hat{x})y - xy - \xi_a(\hat{y})x}{y\left(y + \xi_a(\hat{y})\right)} = \frac{\xi_a(\hat{x})y - \xi_a(\hat{y})x}{y^2\left(1 + \frac{\xi_a(\hat{y})}{y}\right)}$$

$$\leq \frac{\mathcal{E}_a(\hat{x})|y| + \mathcal{E}_a(\hat{y})|x|}{y^2\left(1 + \frac{\mathcal{E}_a(\hat{y})}{|y|}\right)} \leq \frac{\mathcal{E}_a(\hat{x})|y| + \mathcal{E}_a(\hat{y})|x|}{y^2\left(1 - \frac{\mathcal{E}_a(\hat{y})}{|y|}\right)} = \mathcal{E}_a\left(\frac{\hat{x}}{\hat{y}}\right)$$

$$\Rightarrow \mathcal{E}_a\left(\frac{\hat{x}}{\hat{y}}\right) \approx \frac{\mathcal{E}_a(\hat{x})|y| + \mathcal{E}_a(\hat{y})|x|}{y^2}$$

$$\Rightarrow \xi_r\left(\frac{\hat{x}}{\hat{y}}\right) = \frac{\xi_a(\hat{x}/\hat{y})}{x/y} = \frac{\xi_a(\hat{x})y^2 - \xi_a(\hat{y})xy}{xy^2\left(1 + \frac{\xi_a(\hat{y})}{y}\right)} = \frac{\frac{\xi_a(\hat{x})}{x} - \frac{\xi_a(\hat{y})}{y}}{1 + \frac{\xi_a(\hat{y})}{y}}$$

$$= \frac{\xi_r(\hat{x}) - \xi_r(\hat{y})}{1 + \xi_r(\hat{y})} \leq \left|\frac{\xi_r(\hat{x}) - \xi_r(\hat{y})}{1 + \xi_r(\hat{y})}\right| \leq \frac{\mathcal{E}_r(\hat{x}) + \mathcal{E}_r(\hat{y})}{1 + \mathcal{E}_r(\hat{y})} = \mathcal{E}_r\left(\frac{\hat{x}}{\hat{y}}\right)$$

$$\Rightarrow \mathcal{E}_r\left(\frac{\hat{x}}{\hat{y}}\right) \approx \mathcal{E}_r(\hat{x}) + \mathcal{E}_r(\hat{y})$$

$$\begin{cases}\xi_a\left(\frac{x}{y}\right) = \frac{\xi_a(x)y' - \xi_a(y)x'}{(y')^2\left(1 + \frac{\xi_a(y)}{y'}\right)};\ \mathcal{E}_a\left(\frac{x}{y}\right) \approx \frac{\mathcal{E}_a(x)|y'| + \mathcal{E}_a(y)|x'|}{(y')^2} \\ \xi_r\left(\frac{\hat{x}}{\hat{y}}\right) = \frac{\xi_r(\hat{x}) - \xi_r(\hat{y})}{1 + \xi_r(\hat{y})};\ \mathcal{E}_r\left(\frac{\hat{x}}{\hat{y}}\right) \approx \mathcal{E}_r(\hat{x}) + \mathcal{E}_r(\hat{y})\end{cases}$$

- Estas derivaciones también se mantienen en el caso en el que $\hat{x} = x - \xi_a(\hat{x})$ y $\hat{y} = y + \xi_a(\hat{y})$, dado que es como cambiar x por $\hat{x}$ en los resultados anteriores (aplica también para aproximaciones menores a los números a aproximar)

- A través de esta aritmética para los errores, entonces se puede entender mejor la cancelación de términos

  - Considerando la resta de dos números muy cercanos, se pueden obtener los siguientes resultados:

$$\xi_r(\hat{x} - \hat{y}) = \frac{\xi_a(\hat{x}) + \xi_a(\hat{y})}{x - y} = \frac{x}{x - y}\xi_r(\hat{x}) + \frac{y}{x - y}\xi_r(\hat{y})$$

$$\leq \left|\frac{x}{x-y}\right| \mathcal{E}_r(\hat{x}) + \left|\frac{y}{x-y}\right| \mathcal{E}_r(\hat{y}) = \frac{\mathcal{E}_a(\hat{x}) + \mathcal{E}_a(\hat{y})}{|x-y|} =$$

$$= \mathcal{E}_r(\hat{x} - \hat{y})$$

- Como se puede ver, los factores que multiplican las cotas superiores para cada número permiten ver cómo, si la resta es muy cercana a cero (ambos valores son muy cercanos), entonces estos serán muy grandes, haciendo que la cota superior para el error crezca muchísimo

- Para una función de una sola variable, el error de la aproximación tiene las siguientes cotas superiores absolutas y relativas:

  - El error absoluto para una función se puede obtener a través fe una expansión de Taylor, la cual permite a su vez derivar la cota superior:

$$f(\hat{x}) = f\big(x + \xi_a(\hat{x})\big) = f(x) + f'(x)\xi_a(\hat{x}) + O(\xi_a(\hat{x})^2)$$

$$\approx f(x) + f'(x)\xi_a(\hat{x})$$

$$\Rightarrow \xi_a\big(f(\hat{x})\big) \approx |f'(\hat{x})|\xi_a(\hat{x}) \leq |f'(\hat{x})|\mathcal{E}_a(\hat{x}) \approx \mathcal{E}_a\big(f(\hat{x})\big)$$

  - En este caso, el error relativo y su cota superior son las siguientes:

$$\xi_r\big(f(\hat{x})\big) = \frac{|f(\hat{x}) - f(x)|}{f(x)} \approx \frac{|f(\hat{x}) - f(x)|}{f(\hat{x})} \approx \frac{|f'(\hat{x})|\xi_a(\hat{x})}{f(\hat{x})} =$$

$$= \frac{\hat{x}|f'(\hat{x})|\xi_a(\hat{x})}{f(\hat{x})\hat{x}} \approx \frac{\hat{x}|f'(x)|}{f(\hat{x})}\xi_r(\hat{x}) \leq \frac{\hat{x}|f'(\hat{x})|}{f(\hat{x})}\mathcal{E}_r(\hat{x}) =$$

$$= \mathcal{E}_r\big(f(\hat{x})\big)$$

La búsqueda de raíces

- El método de la bisección es una idea basada en poco más que la continuidad de una función f de variable real

  - Suponiendo que se sabe que $f(a)f(b) < 0$, entonces esto significa que f es negativa en un punto y positiva en el otro. Si se asume que f es una función continua, entonces, por el teorema del valor intermedio, tiene que haber un valor entre a y b en el cual f es cero (una raíz entre a y b)

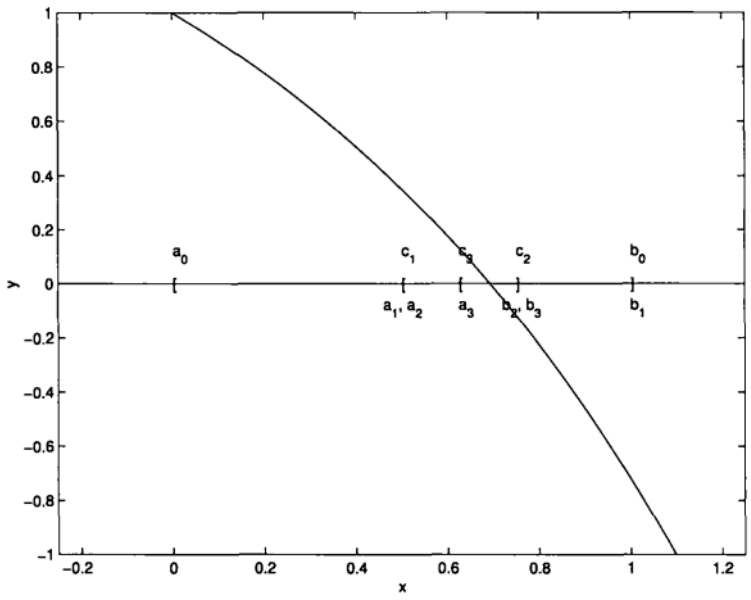
- Siendo $c = (a + b)/2$ el punto medio del intervalo $[a, b]$, y considerando el producto $f(x)$, entonces pueden ocurrir tres posibilidades que conciernen a la raíz α:

  $(1)\ f(a)f(c) < 0 \ \Rightarrow \ \alpha \in [a, c]$

  $(2)\ f(a)f(c) = 0 \ \Rightarrow \ \alpha = c \ \ as \ \ f(a) \neq 0$

  $(3)\ f(a)f(c) > 0 \ \Rightarrow \ \alpha \in [c, b]$

- Aunque solo se tenga una solución para α en el segundo caso, el primer y el tercer caso permite cortar a la mitad la longitud del intervalo original $[a, b]$, haciendo que se pueda buscar el valor en la mitad de posibilidades (dentro de una tolerancia concreta) y, por tanto, reduciendo el intervalo de incertidumbre (el intervalo en donde se supone que está la raíz)



- A partir de esto es posible construir un algoritmo para el método de bisección:

Algorithm 3.1 *Bisection Method (Outline Form).*

1. *Given an initial interval* $[a_0, b_0] = [a, b]$, *set* $k = 0$ *and proceed as follows:*
2. *Compute* $c_{k+1} = a_k + \frac{1}{2}(b_k - a_k)$;
3. *If* $f(c_{k+1})f(a_k) < 0$, *set* $a_{k+1} = a_k$, $b_{k+1} = c_{k+1}$;
4. *If* $f(c_{k+1})f(b_k) < 0$, *set* $b_{k+1} = b_k$, $a_{k+1} = c_{k+1}$;
5. *Update* k *and go to Step 2.*

- Siendo $[a_0, b_0] = [a, b]$ el intervalo inicial con $f(a)f(b) < 0$ y definiendo las raíces aproximadas como $x_n = c_n = (b_{n-1} + a_{n-1})/2$, entonces existe una raíz $\alpha \in [a, b]$ tal que se cumpla la siguiente desigualdad:

$$|\alpha - x_n| \leq \left(\frac{1}{2}\right)^n (b - a)$$

  - La demostración del teorema es la siguiente:

$$b_n - a_n = \frac{1}{2}(b_{n-1} - a_{n-1}) \Rightarrow b_n - a_n = \left(\frac{1}{2}\right)^n (b_0 - a_0)$$

$$\Rightarrow |\alpha - x_n| \leq \frac{1}{2}(b_{n-1} - a_{n-1})$$

$$\Rightarrow |\alpha - x_n| \leq \frac{1}{2}(b_{n-1} - a_{n-1}) \leq \frac{1}{2}\left(\frac{1}{2}\right)^{n-1} (b_0 - a_0) =$$

$$= \left(\frac{1}{2}\right)^n (b_0 - a_0)$$

  - Además, para conseguir una exactitud de $|\alpha - x_n| \leq \epsilon$ solo hace falta tomar n iteraciones que respeten la desigualdad:

$$n \geq \frac{\log(b - a) - \log(\epsilon)}{\log(2)}$$

  - La demostración para el número mínimo de iteraciones del algoritmo es la siguiente:

$$|\alpha - x_n| \leq \left(\frac{1}{2}\right)^n (b - a) \leq \epsilon$$

$$\Rightarrow n[\log(2)] + \log(b - a) \leq \log(\epsilon)$$

$$\Rightarrow n \geq \frac{\log(b - a) - \log(\epsilon)}{\log(2)}$$

- El algoritmo presentado anteriormente no tiene ningún criterio para parar, debido a que los resultados anteriores permiten ver cuántos pasos son necesarios para conseguir un nivel de exactitud deseado. Por lo tanto, se puede implementar un algoritmo más exacto:

Algorithm 3.2 *Bisection Method (Pseudo-code).*

```
input a,b,eps
external f
fa = f(a)
fb = f(b)
if f(a)*f(b) > 0 then stop
n  = fix((log(b-a) - log(eps))/log(2)) + 1
for i=1 to n do
     c  = a + 0.5*(b - a)
     fc = f(c)
     if fa*fc < 0 then
          b = c
          fb = fc
     else
          if fa*fc > 0 then
               a = c
               fa = fc
          else
               alpha = c
               return
          endif
     endif
endfor
```

- Se ha usado la fórmula $c = a + 0.5(b - a)$ y calcular c, más que la fórmula obvia $c = (a + b)/2$. La razón es que para valores grandes de a y b, el algoritmo puede comportar un problema de *overflow*, mientras que usando $c = a + 0.5(a + b)$ no

- Una implementación más descuidada llamaría a la función (la evaluación de la función) dos veces en el bucle, pero es posible con solo una vez. Esto es importante debido a que la función f es la parte que no se escoge y que puede ser la parte más costosa en del código en términos computacionales

- El método de la bisección es un ejemplo de un método global, lo cual quiere decir que siempre converge sin importar que tan lejos comiences de la raíz real

  - La mayoría de métodos siguientes solo convergen de manera localmente, lo cual quiere decir que se tiene que comenzar con una aproximación lo suficientemente buena

  - Una desventaja del método de la bisección es que no se puede usar cuando la función es tangente al eje y no lo cruza (el teorema del valor intermedio no aplica). Otra desventaja es que converge lentamente en comparación a otros métodos

  - No obstante, este método se puede usar junto a otros métodos locales que convergen localmente para hacer un algoritmo más efectivo

- El método de Newton es un algoritmo clásico para encontrar raíces de funciones, la cual se puede derivar geométricamente y analíticamente, y se puede estudiar su error

    - Se desea encontrar una raíz α de $y = f(x)$ dada una aproximación inicial de x_0. Desde el punto de vista geométrico, la idea fundamental del método de Newton es usar la línea tangente para aproximar la función f en el punto $\big(x_0, f(x_0)\big)$, la cual se expresa de la siguiente manera:

$$\frac{y - y_0}{x - x_0} = f'(x_0)$$

        - Por lo tanto, se tiene una ecuación de la recta de la siguiente manera:

$$y = \ell_0(x) = f(x_0) + f'(x_0)(x - x_0)$$

        - Para encontrar el punto x donde se cruza el eje horizontal, se fija $y = 0$ para obtener la siguiente expresión:

$$0 = f(x_0) + f'(x_0)(x - x_0) \;\Rightarrow\; x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

        - A partir de esta expresión, se puede generalizar la expresión fundamental del método de Newton:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

        - Este método, por tanto, se basa en una idea fundamental que se repite mucho en la derivación de los métodos numéricos: reemplazar la función general por una función más simple y hacer el cálculo requerido exactamente en la función más simple. En este caso, se reemplazo f por ℓ_0 y se encontró la raíz exacta de ℓ_0

    - Desde el punto de vista analítico, para derivar la expresión del método de Newton se tiene que utilizar el teorema de Taylor. Dado un valor $x_n \approx \alpha$, se expande f con una serie de Taylor alrededor de x_n, donde $\xi_n \in [x, x_n]$:

$$f(x) = f(x_n) + (x - x_n)f'(x_n) + \frac{1}{2}(x - x_n)^2 f''(\xi_n)$$

- Para poder obtener el algoritmo se iguala $f(x) = 0$ y se resuelve para x, de modo que si se sustituye $x_{n+1} = x$, se tiene la expresión general ignorando el residuo:

$$x = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{1}{2}(x - x_n)^2 \frac{f''(\xi_n)}{f'(x_n)}$$

$$\Rightarrow\ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- La derivación anterior permite sacar otra idea fundamental de los métodos numéricos: dada una expresión en términos de algo simple sumado a un residuo, se genera una aproximación numérica ignorando este residuo

- Experimentando con el método de Newton, se puede ver como el número de dígitos correctos (el número de ceros en la diferencia entre el valor aproximado y el real) se duplica en cada iteración

n	x_n	$\alpha - x_n$	$\log_{10}(\alpha - x_n)$
0	0.000000000000	0.693147180560	−0.1592
1	1.000000000000	0.306852819440	−0.5131
2	0.735758882343	0.042611701783	−1.3705
3	0.694042299919	0.000895119359	−3.0481
4	0.693147581060	0.000000400500	−6.3974
5	0.693147180560	0.000000000000	−13.0961

- En este caso, la cantidad $\log_{10}(\alpha - x_n)$ es una cota superior en el logaritmo de base 10 del término de error, y entonces da una estimación al número de dígitos correctos en la aproximación (una medida de exactitud de la aproximación)

- No obstante, hay funciones para las cuales no hay convergencia o es una convergencia mucho más lenta

- El método de Newton no es un método global, pero esto puede curbirse obteniendo una aproximación inicial mejor, pero a veces se tiene que tomar x_0 muy cerca de α con tal de obtener convergencia

- Si f, f' y f'' son continuas cerca de la raíz, y si f' no es igual a cero en la raíz, entones el método de Newton convergerá cuando la aproximación inicial esté lo suficientemente cerca de la raíz.

Además, la convergencia será muy rápida, con el número de dígitos correctos duplicándose en cada iteración

- Idealmente, se querría parar el método de Newton cuando el error $\alpha - x_n$ sea lo suficientemente pequeño. Por supuesto, no se puede usar esto porque no se tiene α, pero mientras f' no sea cero en la raíz, se puede relacionar este error con una cantidad calculable

  - El teorema del valor medio expresa que $f(\alpha) - f(x_n) = f'(c_n)(\alpha - x_n)$, donde $c_n \in [\alpha, x_n]$, por lo que se puede resolver para $\alpha - x_n$:

$$\alpha - x_n = \frac{f(\alpha) - f(x_n)}{f'(c_n)} = -\frac{f(x_n)}{f'(c_n)}$$

$$as\ \ x_{n+1} = x_n - \frac{f'(x_n)}{f(x_n)}\ \ \Rightarrow\ \ \alpha - x_n = (x_{n+1} - x_n)\frac{f'(x_n)}{f'(c_n)}$$

$$\Rightarrow \frac{\alpha - x_n}{x_{n+1} - x_n} = C_n \quad where \quad C_n = \frac{f'(x_n)}{f'(c_n)}$$

  - Por lo tanto, el error $\alpha - x_n$ es solamente un múltiplo de la cantidad calculable $x_{n+1} - x_n$ (se usan estimaciones calculables para estimar cantidades que no se pueden calcular fácilmente)

  - Además, si se asume que la convergencia está ocurriendo y que $f'(\alpha) \neq 0$, entonces la constante C_n cumple $\lim_{n\to\infty} |C_n| = 1$ y se obtienen los siguientes resultados, los cuales demuestran que la cantidad calculable $|x_{n+1} - x_n|$ mide la convergencia:

$$As\ \ c_n \in [\alpha, x_n]\ \ \&\ \ \lim_{n\to\infty} x_n = \alpha\ \ \Rightarrow\ \lim_{n\to\infty} c_n = \alpha$$

$$\Rightarrow \lim_{n\to\infty} \left|\frac{f'(x_n)}{f'(c_n)}\right| = \lim_{n\to\infty} \left|\frac{f'(\alpha)}{f'(\alpha)}\right| = \lim_{n\to\infty} |C_n| = 1$$

- Usualmente uno para la iteración cuando un múltiplo modesto de $|x_{n+1} - x_n|$ es pequeño, donde $\epsilon > 0$ es un parámetro de tolerancia definido por el usuario

$$M|x_{n+1} - x_n| \leq \epsilon$$

  - Hay veces en donde se tiene $|x_{n+1} - x_n|$ pequeño y x_{n+1} no está muy cerca de α. Esto puede pasar, por ejemplo, si $f'(x_n)$ es muy

grande comparado con $f(x_n)$, y eso hace que sea normal añadir un parámetro de error para ver si de verdad el valor de la función mismo es pequeño

$$|f(x_n)| + |x_{n+1} - x_n| \leq \epsilon/M$$

- Siendo $f \in C^2(I)$ una función dada para algún intervalo real $I \subset \mathbb{R}$ con $f(\alpha) = 0$ para alguna $\alpha \in I$, y definiendo $x_{n+1} = x_n - f(x_n)/f'(x_n)$ para $x_n \in I$, entonces existe un punto ξ_n entre α y x_n tal que se cumple la siguiente igualdad:

$$\alpha - x_{n+1} = -\frac{1}{2}(\alpha - x_n)^2 \frac{f''(\xi_n)}{f'(x_n)}$$

  - Expandiendo f en una serie de Taylor alrededor de $x = x_n$ y fijando después $x = \alpha$, se puede obtener la expresión anterior:

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2}f''(\xi_n)(x - x_n)^2$$

$$x = \alpha \;\Rightarrow\; 0 = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2}f''(\xi_n)(\alpha - x_n)^2$$

$$\Rightarrow\; \alpha - x_n + \frac{f(x_n)}{f'(x_n)} = -\frac{1}{2}(\alpha - x_n)^2 \frac{f''(\xi_n)}{f'(x_n)}$$

$$\Rightarrow\; \alpha - x_{n+1} = -\frac{1}{2}(\alpha - x_n)^2 \frac{f''(\xi_n)}{f'(x_n)}$$

  - A partir de este resultado, es posible establecer una cota superior para el error:

$$\|\alpha - x_{n+1}\|_\infty \leq \frac{1}{2}(\alpha - x_n)^2 \left\| \frac{f''(\xi_n)}{f'(x_n)} \right\|_\infty$$

  - Además, el resultado muestra que el error en un paso se comporta como el cuadrado del error en el paso previo, por lo que una vez el error se vuelve lo suficientemente pequeño, comienza a decrecer rápidamente. Si se asume que la convergencia ocurre, entonces se puede ver como de cerca x_0 tiene que estar de α para que ocurra la convergencia (aproximadamente):

$$As\;\; \xi_n \in [\alpha, x_n] \;\;\&\;\; \lim_{n\to\infty} x_n = \alpha \;\;\Rightarrow\;\; \lim_{n\to\infty} \xi_n = \alpha$$

$$\Rightarrow \lim_{n\to\infty} \alpha - x_{n+1} = -\frac{1}{2}(\alpha - x_n)^2 \frac{f''(\alpha)}{f'(\alpha)}$$

$$\Rightarrow \alpha - x_{n+1} \approx C(\alpha - x_n)^2 \quad for \quad C = -\frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}$$

- Si la convergencia ocurre, entonces se puede demostrar (por el mismo principio anteriormente visto para $|x_{n`+1} - x_n|$) que se cumple la siguiente igualdad:

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_{n+1})^2} = -\frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}$$

- Para caracterizar qué tan rápido una secuencia converge a un límite, se utiliza el concepto de orden de convergencia. Siendo x_n una secuencia de valores convergiendo a α tal que se cumple la siguiente igualdad para $C \neq \{0, \pm\infty\}$, entonces p se denomina orden de convergencia de la secuencia:

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^p} = C$$

  - El requerimiento de que $C \neq \{0, \pm\infty\}$ fuerza a que p sea un valor único. Si p es demasiado grande, el denominador se reduce muy rápido y la *ratio* crece sin cota superior, mientras que si p es muy pequeño, entonces la *ratio* tiende a cero

  - Existen algunos casos especiales tales como $p = 2$ (como en el método de Newton), llamado caso de convergencia cuadrática, o el caso $p = 1$ y $|C| < 1$ (como en el caso del método de la bisección). Otro caso interesante es cuando la serie es convergente superlinealmente, de modo que la serie converge más rápida que una serie que converge linealmente, pero no de manera cuadrática:

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = 0 \quad \& \quad \lim_{n\to\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = \pm\infty$$

  - Generalmente hablando, teniendo p más grande significa que la convergencia es más rápida, dado que $\alpha - x_{n+1} \approx C(\alpha - x_n)^p$

- Un problema del método de Newton es que requiere la fórmula para la derivada de f, pero a veces no se puede definir la función de manera analítica y simple.

Una manera obvia de superar este problema es usar otro método que no requiera del uso de derivadas, por lo que se concibe el método de la secante

- El método de Newton se deriva, geométricamente, de dibujar una recta tangente desde el punto aproximado actual hasta el eje horizontal, y la derivada se requiere por el uso de la tangente. Si, en cambio, se usara la línea secante (pasando entre dos puntos cualesquiera de la curva y no solo uno), entonces no se necesitaría ninguna derivada

    - Si x_0 y x_1 vienen dadas, se quiere construir una fórmula que pase a través de $\big(x_0, f(x_0)\big)$ y $\big(x_1, f(x_1)\big)$ y usa su raíz para definir la siguiente iteración x_2:

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

$$\Rightarrow \quad x_2 = x_1 - f(x_1)\left[\frac{x_1 - x_0}{f(x_1) - f(x_0)}\right]$$

$$\Rightarrow \quad x_n = x_n - f(x_n)\left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right]$$

    - Esto es consistente con el método de Newton, dado que se usa la aproximación de la derivada:

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

    - El error en esta aproximación es proporcional a $x_n - x_{n-1}$, por lo que si se asume que la iteración es convergente ($x_n - x_{n-1} \to 0$ cuando $n \to \infty$), el método de la secante se parece más y más al método de Newton. En consecuencia, se espera una convergencia rápida para x_0 cerca de α

- El método de la secante y el método de Newton sufren de problemas similares porque se basan en la misma idea geométrica. La teoría de convergencia fundamental es exactamente la misma: si la aproximación inicial está lo suficientemente cerca de la raíz, el método convergerá

    - En este caso, la fórmula de error para el método de la secante es la siguiente:

$$\alpha - x_{n-1} = -\frac{1}{2}(\alpha - x_n)(\alpha - x_{n-1})\frac{f''(\xi_n)}{f'(\eta_n)}$$

$$where \quad \min\{\alpha, x_n, x_{n-1}\} \leq \xi_n \quad \& \quad \max\{\alpha, x_n, x_{n-1}\} \geq \eta_n$$

- Además, este resultado permite obtener la siguiente aproximación y cuota superior:

$$|\alpha - x_{n+1}| = -\frac{1}{2}(\alpha - x_n)(\alpha - x_{n-1})\frac{f''(\xi_n)}{f'(\eta_n)}$$

$$\|\alpha - x_{n-1}\|_\infty \leq \frac{1}{2}|\alpha - x_n||\alpha - x_{n-1}| \left\|\frac{f''(\xi_n)}{f'(\eta_n)}\right\|_\infty$$

- Se puede construir un algoritmo para el método de la secante que sea eficiente:

Algorithm 3.3 *Secant Method*

```
input x0, x1, tol, n
external f
        f0 = f(x0)
        f1 = f(x1)
        for i=1 to n do
                x = x1 - f1*(x1 - x0)/(f1 - f0)
                fx = f(x)
                x0 = x1
                x1 = x
                f0 = f1
                f1 = fx
                if abs(x1 - x0) < tol then
                     root = x1
                     stop
                endif
        endfor
```

- El método de la secante requiere solo una evaluación de la función por iteración, mientras que el método de Newton necesita dos: uno para la función y otra para la derivada. Por lo tanto, el método de la secante cuesta por cada paso casi la mitad que el método de Newton

- Si f, f' y f'' son todas continuas cerca de la raíz, y si f' no iguala cero en la raíz, entonces el método de la secante convergerá cuando la aproximación inicial esté lo suficientemente cerca de la raíz. Además, la convergencia será superlineal

$$\lim_{n\to\infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = 0$$

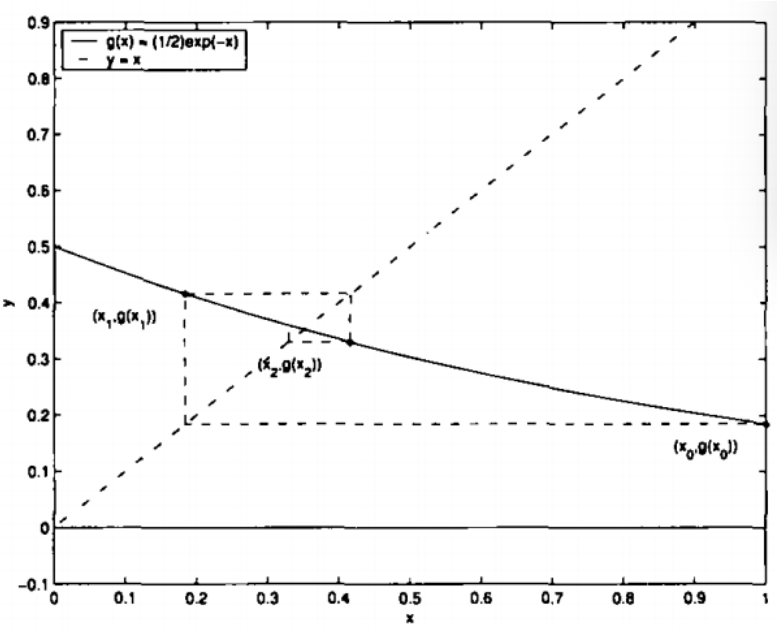
- Fundamentalmente, esto es casi lo mismo para el método de Newton, no obstante, no tiene una convergencia cuadrática

- Aunque se han visto métodos de iteraciones simples anteriormente, se puede estudiar la iteración simple, generalmente separado de cualquier consideración y conexión con problemas de encontrar raíces

  - Aplicando métodos iterativos como el de Newton sobre una función $f(x)$, y suponiendo que $x_n \to \alpha$ cuando $n \to \infty$, entonces se puede escribir de manera más abstracta la siguiente iteración:

$$x_{n+1} = g(x_n) \quad for \quad g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$$

    - Asumiendo que $x_n \to \alpha$ cuando $n \to \infty$ (hay convergencia), entonces $f(\alpha) = 0 \Leftrightarrow \alpha = g(\alpha)$. Esto define α como un punto donde una nueva función $y = f(x)$ cruza la línea $y = x$, de modo que es un punto fijo de la función g y una iteración de la forma de punto fijo para g

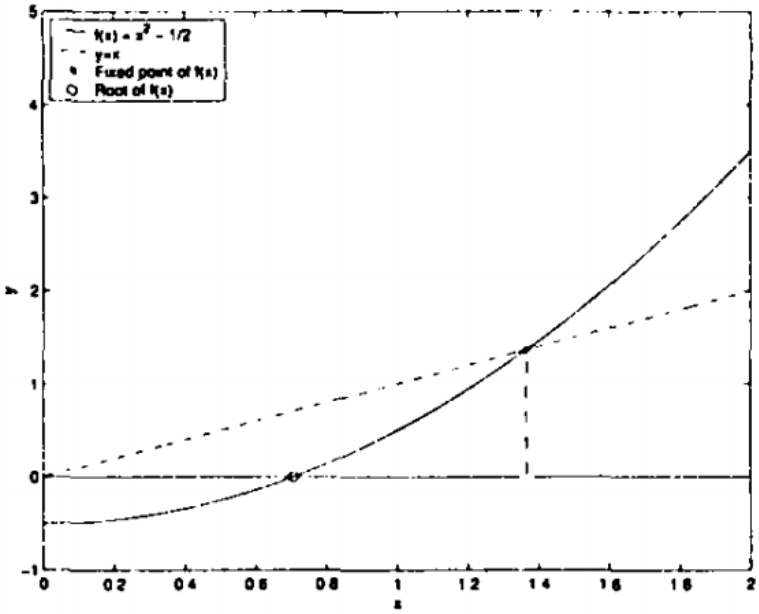


  - Aunque ahora se ha establecido una conexión con el problema de encontrar una raíz y el problema de un punto fijo. Para una función g, no obstante, se plantean preguntas como bajo qué condiciones existe un punto fijo, bajo cuales hay convergencia y su rapidez

    - Para ello, se utiliza la teoría y la práctica de la iteración de punto fijo, más de la perspectiva para aplicar al problema de encontrar raíces. De este modo, se usa la teoría de punto fijo para

informarse sobre ciertos aspectos de los problemas de encontrar raíces

- Generalmente, para cualquier función, las raíces y los puntos fijos no son lo mismo (si es que estos existen). Cuando se utiliza una iteración de punto fijo para encontrar una raíz de una función f, la raíz es un punto dijo de la función g, pero de f



- Antes de desarrollar resultados importantes, se puede hacer una investigación informal de la iteración de punto fijo. Teniendo $\alpha = g(\alpha)$ y $x_{n+1} = g(x_n)$, por lo que se obtiene el siguiente resultado y asumiendo que g es diferenciable para poder usar el teorema del valor medio, entonces:

$$\alpha - x_{n+1} = g(\alpha) - g(x_n) = g'(\xi_n)(\alpha - x_n)$$

- Si $|g'(x)| < 1$ cerca del punto fijo, entonces se puede demostrar que se cumple la siguiente desigualdad para $c < 1$:

$$|\alpha - x_{n+1}| \leq |g'(\xi_n)||\alpha - x_n|$$

$$\Rightarrow |\alpha - x_{n+1}| \leq \|g'(\xi_n)\|_\infty |\alpha - x_n|$$

$$if \max_{\xi_n \in [a,b]} |g'(\xi_n)| = c \Rightarrow |\alpha - x_{n+1}| \leq c|\alpha - x_n|$$

$$\Rightarrow |\alpha - x_n| < c^n |\alpha - x_0|$$

- Este resultado implica convergencia, de modo que, tomando el límite, se puede establecer que los métodos de punto fijo convergen linealmente:

$$\frac{\alpha - x_{n+1}}{\alpha - x_n} = g'(\xi_n) \to g'(\alpha) \quad as \quad n \to \infty$$

- Siendo $g \in C([a,b])$ con $a \leq g(x) \leq b$ para toda $x \in [a,b]$, entonces se cumplen las siguientes proposiciones:

  - La función g tiene al menos un punto fijo $\alpha \in [a,b]$

  - Si existe un valor $\gamma < 1$ tal que $|g(x) - g(y)| \leq \gamma|x - y|$ para toda la x y y en $[a,b]$, entonces se cumplen las siguientes condiciones:

    (a) $\alpha \; is \; unique$

    (b) $The \; iteration \; x_{n+1} = g(x_n) \; converges \; to \; \alpha \; for \; any \; initial \; guess \; x_0 \in [a,b]$

    (c) $The \; error \; estimate \; is \; |\alpha - x_n| \leq \frac{\gamma^n}{1-\gamma}|x_1 - x_0|$

  - Si g es continuamente diferenciable en $[a,b]$ con $\max_{x \in [a,b]} |g'(x)| = \gamma < 1$, entonces se cumplen las siguientes condiciones:

    (a) $\alpha \; is \; unique$

    (b) $The \; iteration \; x_{n+1} = g(x_n) \; converges \; to \; \alpha \; for \; any \; initial \; guess \; x_0 \in [a,b]$

    (c) $The \; error \; estimate \; is \; |\alpha - x_n| \leq \frac{\gamma^n}{1-\gamma}|x_1 - x_0|$

    (d) $The \; limit \; \lim_{n \to \infty} \frac{\alpha - x_{n+1}}{\alpha - x_n} = g'(\alpha) \; holds$

- Siendo g continuamente diferenciable en un intervalo abierto de un punto fijo α con $|g'(\alpha)| < 1$, entonces, para toda x_0 suficientemente cerca de α, la iteración $x_{n+1} = g(x_n)$ converge y se cumplen las siguientes relaciones para una $\gamma < 1$:

$$\lim_{n\to\infty}\frac{\alpha - x_{n+1}}{\alpha - x_n} = g'(\alpha) \quad \& \quad |\alpha - x_n| \leq \frac{\gamma^n}{1-\gamma}|x_1 - x_0|$$

- Como g es continuamente diferenciable en un intervalo abierto del punto $|g'(\alpha)| < 1$, se puede encontrar un intervalo cerrado J, centrado en el punto fijo α, tal que $|g'(x)| \leq \gamma < 1$ para toda $x \in J$. Por la definición de la iteración, se obtiene la siguiente desigualdad:

$$|\alpha - x_1| = |g(\alpha) - g(x_0)| \leq \gamma|\alpha - x_0|$$

- Por lo tanto x_1 está más cerca del punto fijo que x_0, y por tanto, todas las iteraciones siguientes. En ese caso, $g(x) \in J$ para toda $x \in J$, y se puede usar el teorema anterior para completar la demostración

La interpolación y la aproximación

- Antes de entrar en el tema de la interpolación, es necesario revisar métodos numéricos muy usados para polinomios. La mejor manera de evaluar las aproximaciones polinómicas para funciones dadas es la regla de Horner y la multiplicación anidada

  - La manera más eficiente de evaluar un polinomio es a través de la multiplicación anidada. Si se tiene $p_n(x) = a_0 + a_1x + \cdots + a_nx^n$, entonces se puede factorizar fuera cada una de las potencias de la siguiente manera:

$$p(x) = a_0 + x\big(a_1 + x(a_2 + \cdots + x(a_{n-1} + a_nx)\ldots)\big)$$

    - El cálculo con esta forma del polinomio tomaría $n+1$ multiplicaciones y n sumas, pero sin este método tomaría lo mismo más el coste de formar las nuevas potencias $x^2, x^3, \ldots, x^n$

  - Una forma algorítmica para la multiplicación se puede escribir de manera muy simple, tal como muestra el siguiente pseudo-código

Algorithm 2.1 *Horner's Rule for Polynomial Evaluation (pseudo-code).*

```
!------------------------------------------------
!
!     Assumes that the polynomial coefficients
!     are stored in the array a(j),j=0..n.
!     px = value of polynomial upon completion
!          of the code.
!
      px = a(n)
      for k = n-1 downto 0
           px = a(k) + px*x
      endfor
```

- Este algoritmo se conoce como la regla de Horner, y se puede modificar fácilmente para dar también las primeras derivadas.

$$p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

$$\Rightarrow\ p_n'(x) = a_1 + 2a_2x + \cdots + na_nx^{n-1}$$

$$\Rightarrow\ p_n'(x) = a_1 + x\big(2a_2 + \cdots + x\big((n-1)a_{n-1} + na_nx\big)\ldots\big)$$

Algorithm 2.2 *Horner's Rule for Polynomial Derivative Evaluation (pseudo-code).*

```
!------------------------------------------------
!
!     Assumes that the polynomial coefficients
!     are stored in the array a(j),j=0..n.
!     dp = value of derivative upon completion
!
      dp = n*a(n)
      for k=n-1 down to 1
           dp = k*a(k) + dp*x
      endfor
```

- Si los valores intermedios de los cálculos anteriores de $p(x)$ se guardan, entonces el cálculo subsiguiente de la derivada se puede hacer de manera más sencilla

  - Se definen los valores b_k para $k = 1,2,\ldots,n$ acorde a $b_k = xb_{k-1} + a_{n-k}$ con $b_0 = a_n$, de modo que $b_n = p(x)$. Además, se definen los valores c_k acorde a $c_k = xc_{k-1} + b_k$ con $c_0 = b_0$, por lo que ambos valores son funciones de x

  - Por lo tanto, se pueden obtener las siguientes igualdades:

$$b_0' = 0 \quad and \quad b_k' = b_{k-1} + xb_{k-1}' \ \Rightarrow\ b_1' = b_0$$

$$\Rightarrow\ c_1 = xc_0 + b_1 = xb_0 + b_1 = xb_1' + b_1 = b_2'$$

$$\Rightarrow\ c_2 = xc_1 + b_2 = xb_2' + b_2 = b_3'$$

$$\Rightarrow\ \ldots$$

- En general, es posible demostrar de manera inductiva que $c_k = b_{k+1}'$, por lo que $c_{n-1} = p'(x)$ dado que $b_n = p(x)$. Por lo tanto, un algoritmo más eficiente de la regla de Horner sería el siguiente:

Algorithm 2.3 *A More Efficient Implementation of Horner's Rule (pseudo-code).*

```
!
!     Assumes that the polynomial coefficients
!     are stored in the array a(j),j=0..n.
!     b(n) = value of polynomial upon completion
!          of the code.
!
      b(0) = a(n)
      for k = 1 to n
            b(k) = x*b(k-1) + a(n-k)
      endfor
!
!     c(n-1) = value of derivative upon completion
!            of the code
!
      c(0) = b(0)
      for k = 1 to n-1
            c(k) = x*c(k-1) + b(k)
      endfor
```

- Cuando se escribe la regla de Horner para un polinomio que tiene algunos coeficientes negativos, es importante guardar los coeficientes en la matriz directamente con el negativo, dado que es más difícil distribuir los signos negativos de manera correcta:

$$p(x) = 1 - 2x - x^2 + x^3$$

$$\Rightarrow\ p(x) = 1 + x\Big((-2) + x\big((-1) + x\big)\Big)$$

- Antes de la existencia de calculadoras, para estimar los valores intermedios de diversas funciones se usaba la interpolación, y aunque ahora esta aplicación particular no es tan importante ahora, sigue siendo importante para matemáticas computacionales

  - Dado que un conjunto de puntos x_k, típicamente llamado nodos, se dice que una función p interpola la función f en los nodos si $p(x_k) = f(x_k)$ para toda k

- A veces solo se tiene una función p y un conjunto de valores y_k, por lo que se dice que p interpola los datos si $p(x_k) = y_k$ para toda k. En el tipo más común de interpolación, la función p (el interpolante), es un polinomio

- Uno se interesa más por la medida en la que p aproxima f ($p \approx f$) o la medida en la que p representa los datos

- Uno de los problemas más antiguos de las matemáticas y uno de los más aplicados es el problema de construir una aproximación para una función dada f a partir de funciones más simples. Una variación del problema que también es de interés es la construcción de una función suave de un conjunto discreto de puntos

  - Dado un conjunto de nodos $\{x_i, 0 \leq i \leq n\}$ y valores de los datos correspondientes $\{y_i, 0 \leq i \leq n\}$, se tiene que encontrar un polinomio $p_n(x)$ de grado menor o igual a n, tal que se cumpla la siguiente igualdad:

$$p_n(x_i) = y_i \quad for \;\; 0 \leq i \leq n$$

  - Dado un conjunto de nodos $\{x_i, 0 \leq i \leq n\}$ y una función continua $f(x)$, se tiene que encontrar un polinomio $p_n(x)$ de grado menor o igual a n, tal que se cumpla la siguiente igualdad:

$$p_n(x_i) = f(x_i) \quad for \;\; 0 \leq i \leq n$$

  - En el primer caso se intenta ajustar un polinomio a los datos, mientras que en el segunda se intenta aproximar una función dada con el polinomio interpolador

  - Mientras que los dos casos son diferentes, siempre se puede considerar el primero como un caso especial del segundo (tomando $y_i = f(x_i)$ para toda i), por lo que se suele utilizar la segunda manera

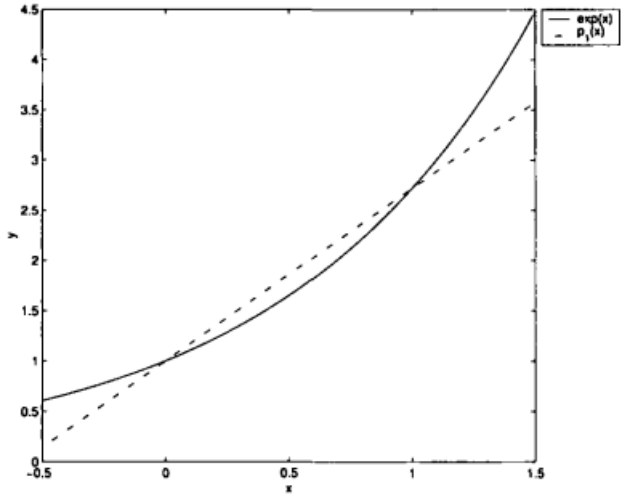
- La técnica más sencilla es la interpolación lineal, por la cual se usa una línea recta para aproximar una función dada. Como solo toma dos puntos determinar una línea recta, se asume que se dan solo dos nodos x_0 y x_1 y una función f, y se usa la siguiente ecuación de una línea recta que pasa a través de los dos puntos $\big(x_0, f(x_0)\big)$ y $\big(x_1, f(x_1)\big)$:

$$p_1(x) = \frac{x_1 - x}{x_1 - x_0} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

- Se puede ver como se cumplen las condiciones de interpolación dado que $p_1(x_0) = f(x_0)$ y $p_1(x_1) = f(x_1)$. Los coeficientes para $f(x_0)$ y $f(x_1)$ se conocen como funciones base nodales, y se podrían interpretar como una función λ_j definida de la siguiente manera:

$$\lambda_j(x_i) = \begin{cases} 1 & if \ \ i = j \\ 0 & if \ \ i \neq j \end{cases}, \qquad \forall i, j = 0,1,2, \dots, n$$

- La aproximación puede no ser muy precisa, pero si los nodos estuvieran más cerca, la aproximación podría ser más precisa entre nodos



- Para saber que tan exacta es una interpolación lineal, se utiliza el teorema de Rolle

  - Se definen los siguientes términos, en donde t es algún valor fijo en el intervalo (x_0, x_1) (aunque el valor t pueda estar en cualquier parte, esta restricción suele ser el caso común):

$$E(x) = f(x) - p_1(x) \qquad w(x) = (x - x_0)(x - x_1)$$

$$G(x) = E(x) - \frac{w(x)}{w(t)}E(t) \ \ \Rightarrow \ \ G(x_0) = 0, G(x_1) = 0, G(t) = 0$$

  - Entonces, el teorema de Rolle dice que debe existir un punto η_0 entre x_0 y t tal que $G'(\eta_0) = 0$ y, similarmente, un punto η_1 entre x_1 y t tal que $G'(\eta_1) = 0$. Se puede aplicar el teorema de Rolle a G' y decir que debe existir un punto ξ entre η_0 y η_1, tal que $G''(\xi) = 0$, obteniendo los siguientes resultados:

$$G''(x) = f''(x) - \frac{2}{w(t)}E(t)$$

$$\Rightarrow\ G''(\xi) = 0\ \ \Rightarrow\ \ 0 = f''(x) - \frac{2}{w(t)}E(t)$$

$$\Rightarrow\ \ f(t) - p_1(t) = \frac{1}{2}(t - x_0)(t - x_1)f''(\xi)\ \ for\ \ t \in [x_0, x_1]$$

- Para obtener una cota superior para el error de interpolación, se toman valores absolutos y el máximo de la segunda derivada (equivalente a usar la norma infinita o el peor caso para esta derivada)

$$|f(x) - p_1(x)| \leq \frac{1}{2}|(x - x_0)(x - x_1)| \max_{t\in[x_0,x_1]}|f''(t)| \leq$$

$$\leq \frac{1}{2}\left(\max_{t\in[x_0,x_1]}|(x - x_0)(x - x_1)|\right)\left(\max_{t\in[x_0,x_1]}|f''(t)|\right)$$

$$\Rightarrow\ \ |f(x) - p_1(x)| \leq \frac{1}{8}(x_1 - x_0)^2\left(\max_{t\in[x_0,x_1]}|f''(t)|\right)$$

- Siendo $f \in C^2([x_0, x_1])$ y p_1 un polinomio lineal que interpola f en x_0 y x_1, entonces para toda $x \in [x_0, x_1]$ se cumple la siguiente desigualdad:

$$|f(x) - p_1(x)| \leq \frac{1}{8}(x_1 - x_0)^2\left(\max_{t\in[x_0,x_1]}|f''(t)|\right)$$

- Un técnica común y relacionada para aproximar una curva es usar una interpolación lineal por piezas, en la que se rompe el intervalo de interés en subintervalos y se usa un interpolador lineal diferente en cada subintervalo. Para ello, primero hace falta entender la noción del espacio de funciones lineales y las funciones *hat*

  - Cuando se aplica el interpolador lineal en los diferentes subintervalos se fuerza la continuidad en los subintervalos adyacentes al imponer los grados de libertad en el punto inicial y final de los subintervalos (para que coincida inicio y final entre subintervalos adyacentes)

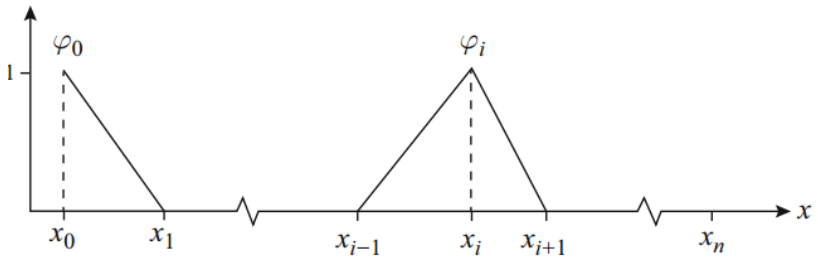
  - Siendo $I = [0, L]$ un intervalo y habiendo $\{x_i\}_{i=0}^n$ puntos, se define la partición o *mesh* $\mathcal{I}$: $0 = x_0 < x_1 < \cdots < x_{n-1} < x_n = L$

de I en n subintervalos $I_i = [x_{i-1}, x_i]$ para $i = 1,2,\dots,n$ de longitud $h_i = x_i - x_{i-1}$. En este *mesh* se define un espacio V_h de funciones lineales por piezas, donde $C^0(I)$ denota el espacio de funciones continuas en I y $P_1(I_i)$ denota el espacio de funciones lineales en I_i:

$$V_h = \{v : v \in C^0(I), v|_{I_i} \in P_1(I_i)\}$$

- Cada función en $v \in V_h$ se determina únicamente por sus valores nodales $\{v(x_i)\}_{i=0}^n$ y, conversamente, para cualquier conjunto de valores nodales $\{\alpha_i\}_{i=0}^n$, existe una función $v \in V_h$ con estos valores nodales. Motivándose por esta observación, se deja que los valores nodales definan los grados de libertad y se introduce una base $\{\varphi_j\}_{j=0}^n$ para V_h asociada con los nodos y tal que se cumple la siguiente definición para cada φ_j (llamada función sombrero o función *hat*):

$$\varphi_j(x_i) = \begin{cases} 1 & if \ \ i = j \\ 0 & if \ \ i \neq j \end{cases}, \qquad \forall i, j = 0,1,2,\dots,n$$



- Cada función *hat* es continua, lineal por piezas y toma un valor unitario en su propio nodo x_i y nulo para todos los otros. De este modo, φ_i solo es diferente a cero en los intervalos I_i y I_{i+1} conteniendo el nodo x_i (el soporte de φ_i es $I_i \cup I_{i+1}$), a excepción de los dos *half hats* φ_0 y φ_n en los nodos más a la izquierda y a la derecha (con soporte en I_0 para φ_0 y I_n para φ_n)

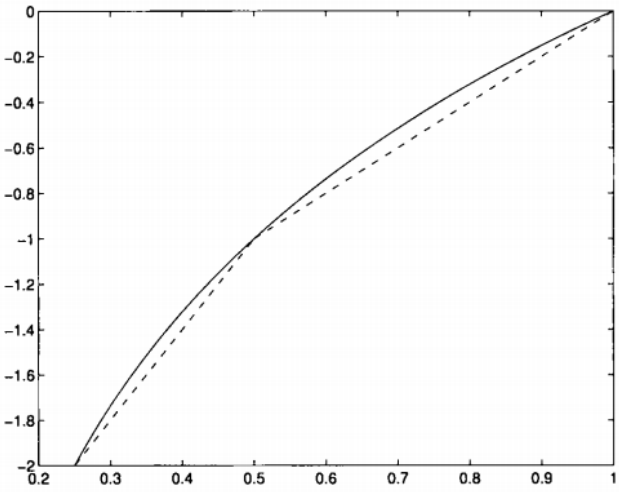
- Por construcción, cualquier función $v \in V_h$ se puede escribir como una combinación lineal de funciones *hat* $\{\varphi_j\}_{j=0}^n$ y los coeficientes correspondientes $\{\alpha_i\}_{i=0}^n$ con $\alpha_i = v(x_i)$ para $i = 0,1,\dots,n$

$$v(x) = \sum_{i=0}^{n} \alpha_i \varphi_i(x)$$

$$where \quad \varphi_i = \begin{cases} (x - x_{i-1})/h_i & if\ x \in I_i \\ (x_{i+1} - x)/h_{i+1} & if\ x \in I_{i+1} \\ 0 & otherwise \end{cases}$$

- Ahora es bastante sencillo extender el concepto de interpolación lineal en un solo intervalo a una interpolación lineal por piezas en un *mesh*. Dada una función continua f en el intervalo $I = [0, L]$, se define el interpolador lineal por piezas $p_{1p} \in V_h$ en el *mesh* $\mathcal{I}$ de I de la siguiente manera:

$$q_n(x) = \sum_{i=1}^{n} f(x_i)\varphi_i(x)$$



  - El interpolador $p_{1p}(x)$ satisface las siguientes desigualdades para el error de interpolación:

$$\|f(x) - q_n(x)\|_2^2 = \sum_{i=1}^{n} \|f(x_i) - q_n(x_i)\|_2^2 \leq C \sum_{i=1}^{n} h_i^4 \|f''(x_i)\|_2^2$$

$$\|f'(x) - q_n'(x)\|_2^2 = \sum_{i=1}^{n} \|f'(x_i) - q_n'(x_i)\|_2^2 \leq C \sum_{i=1}^{n} h_i^2 \|f''(x_i)\|_2^2$$

  - Los resultados muestran que el erro tiende a cero cuanto menor es el espaciado h_i

- Una vez vistos los problemas de interpolación y de aproximación y entendiendo los métodos más básicos de interpolación lineal, se puede pasar a un método de interpolación más complejo, que son los polinomios de Lagrange

- Siendo $x_i \in I$ para $0 \leq i \leq n$ los nodos dados, mientras cada uno de los nodos sean distintos ($x_i = x_j$ si, y solo si, $i = j$), entonces existe un único polinomio p_n de grado igual o menor a n que satisface las siguientes ecuaciones para un conjunto de datos $\{y_i\}$ o para una función $f \in C(I)$:

$$(for\ data\ values) \quad p_n(x_i) = y_i \quad for\ \ 0 \leq i \leq n$$

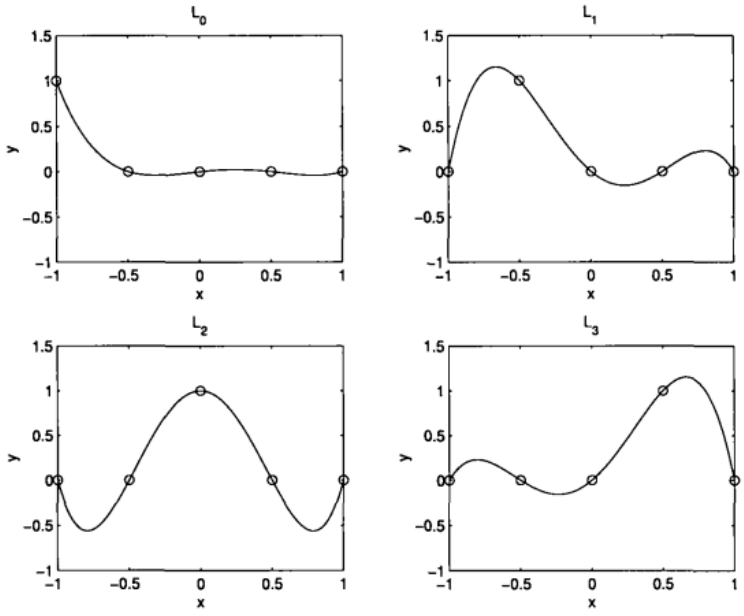
$$(for\ function\ f) \quad p_n(x_i) = f(x_i) \quad for\ \ 0 \leq i \leq n$$

  - Este teorema se denomina teorema de la existencia y unicidad de la interpolación polinómica debido a que demuestra la existencia y la unicidad de un polinomio interpolador para cada conjunto de datos o función $f \in C(I)$

  - Para demostrar el teorema, se comienza definiendo la familia de funciones $L_i^{(n)}(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n}(x - x_k)/(x_i - x_k)$ y se nota que son los polinomios de grado n y tiene la siguiente propiedad interesante, donde δ_{ij} es la delta de Kronecker:

$$L_i^{(n)}(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k} = \frac{(x - x_0) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

$$\Rightarrow L_i^{(n)}(x_j) = \delta_{ij} = \begin{cases} 1 & if \ \ i = j \\ 0 & if \ \ i \neq j \end{cases}$$



- Si uno se basa en la propiedad anterior, entonces si se define un polinomio como $p_n(x) = \sum_{k=0}^{n} y_k L_k^{(n)}(x)$, entonces se obtiene la siguiente igualdad:

$$p_n(x) = \sum_{k=0}^{n} y_k L_k^{(n)}(x) \;\Rightarrow\; p_n(x) = \sum_{k=0}^{n} y_k L_k^{(n)}(x_i) = y_i$$

- Por lo tanto, si se satisfacen las condiciones de interpolación, solo queda demostrar la unicidad del polinomio. Para ello, se asume que existe un segundo polinomio interpolador $q(x)$ y se define el siguiente polinomio:

$$r(x) = p_n(x) - q(x)$$

- Dado que p_n y q son polinomios de un grado menor o igual a n, su diferencia también lo es. No obstante, se cumple la siguiente igualdad para cada uno de los $n + 1$ nodos:

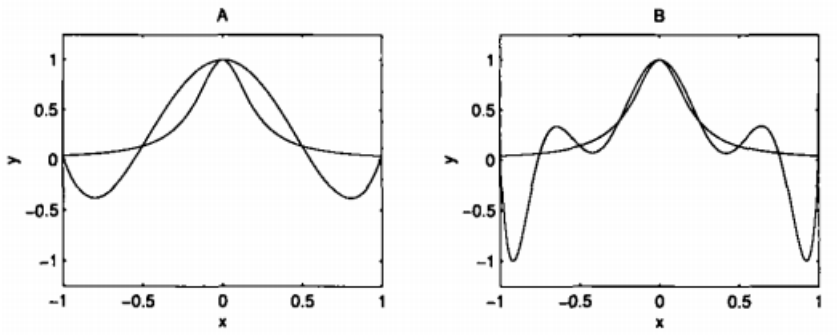
$$r(x_i) = p_n(x_i) - q(x_i) = y_i - y_i = 0$$

- En consecuencia, se tiene un polinomio de un grado menor o igual a n que tiene $n + 1$ raíces. Como el único polinomio que cumple eso es el cero, entonces p_n es único

$$r(x) = 0 \;\Rightarrow\; p_n(x) = q(x)$$

- La construcción presentada hasta ahora se denomina interpolación de Lagrange, y aunque otros enfoques se discutirán más adelante, este es el más básico y fundamental

    - Hay funciones para las cuales los polinomios interpoladores de Lagrange proporcionan una buena aproximación, mientras que hay otros casos en donde se tienen que aumentar los nodos o puntos (y, por lo tanto, el grado del polinomio) para que sea una mejor aproximación para un intervalo de valores completos



- No obstante, no se ha proporcionado ningún algoritmo para el cálculo de polinomios interpoladores de Lagrange debido a que la forma de Lagrange no es muy adecuada para cálculos actuales, y hay una manera alternativa para construir polinomios que es superior

- La forma del polinomio interpolador de Lagrange da una construcción ordenada, pero no permite usarse en cálculos reales, en parte debido a que cuando se decide añadir un punto en el conjunto de nodos, se tiene que recalcular todas las funciones $L_i^{(n)}$ y no se puede escribir fácilmente p_{n+1} en términos de p_n. Una forma alternativa es la forma del polinomio interpolador de Newton, la cual evita este problema y permite que escribir p_{n+1} en términos de p_n fácilmente

  - Siendo p_n un polinomio que interpola f en los nodos x_i para $0 \leq i \leq n$ y siendo p_{n+1} el polinomio que interpola f en los nodos x_i para $0 \leq i \leq n+1$, entonces p_{n+1} se da por la siguiente igualdad:

$$p_{n+1}(x) = p_n(x) + a_{n+1}w_n(x) \quad where$$

$$w_n(x) = \prod_{i=0}^{n}(x - x_i) \quad \& \quad a_{n+1} = \frac{f(x_{n+1}) - p_n(x_{n+1})}{w_n(x)}$$

$$and \quad p_0(x) \equiv a_0 = f(x_0)$$

    - Debido a que se sabe que el polinomio interpolador es único, solo se tiene que demostrar que p_{n+1} satisface las condiciones de interpolación. Para $k \leq n$, se tiene que $w_n(x_k) = 0$, de modo que se puede obtener la siguiente igualdad:

$$p_{n+1}(x_k) = p_n(x_k) + a_{n+1}w_n(x_k) = p_n(x_k) = f(x_k)$$

    - Por lo tanto, p_{n+1} interpola todo menos el último punto. Por lo tanto, calculando directamente para x_{n+1} se puede ver que p_{n+1}

interpola todos los nodos y es un polinomio de grado igual o menor a $n+1$

$$p_{n+1}(x_{n+1}) = p_n(x_{n+1}) + a_{n+1}w_n(x_{n+1}) =$$

$$= p_n(x_{n+1}) + f(x_{n+1}) - p_n(x_{n+1}) = f(x_{n+1})$$

- Para $\{a_k\}$ y w_n definidos anteriormente, se cumple la siguiente igualdad asumiendo que $w_{-1}(x) = 1$:

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots$$

$$+a_n(x - x_0) \dots (x - x_{n-1}) = \sum_{k=0}^{n} a_k w_{k-1}(x)$$

  - La demostración de este teorema se basa en un argumento totalmente inductivo, en donde se usa la recursión vista y las definiciones para calcular la expresión resultante

  - Es posible utilizar esta forma de Newton del polinomio interpolador para construir y evaluar polinomios interpoladores de una manera eficiente:

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots$$

$$+a_n(x - x_0) \dots (x - x_{n-1})$$

$$\Rightarrow p_n(x) = a_0 + (x - x_0)(a_1 + (x - x_1)(a_2 + \cdots + (x - x_{n-1})a_n) \dots))$$

- Los coeficientes a_k se denominan diferencias divididas, y se puede construir un algoritmo para poder calcularlas basándose en la definición. También es posible modificar ligeramente este algoritmo para permitir que el usuario pueda actualizar un conjunto existente de coeficientes de diferencias divididas, añadiendo nuevos puntos al conjunto de nodos

Algorithm 4.1 *Newton Interpolation (Construction) via Divided Difference Coefficients*

```
input n,x,y
a(0) = y(0)
for k=1 to n do
     w = 1
     p = 0
     for j=0 to k-1 do
          p = p + a(j)*w
          w = w*(x(k) - x(j))
     endfor
     a(k) = (y(k) - p)/w
endfor
```

Algorithm 4.2 *Newton Interpolation (Evaluation) via Divided Difference Coefficients*

```
input n,xx,x,a
!
!    n = degree of polynomial
!    xx = point at which the polynomial is
!         to be evaluated
!    x = array of nodes
!    a = array of divided difference coefficients
!
     px = a(n)
!
!    px = polynomial value at xx
!
     for k = n-1,0,-1
          xd = xx - x(k)
          px =  a(k) + px*xd
     endfor
```

- Una interpolación polinómica de un grado lo suficientemente alto reproducirá el polinomio original. Por lo tanto, una manera de poder comprobar si el código de interpolación funciona es comprobando si puede reproducir un polinomio de grado n cuando se intenta interpolarlo en $n + 1$ nodosç

- Una manera alternativa de poder llegar a los coeficientes de diferencias divididas, y la cual es la manera original, es a través de una tabla de diferencias divididas

  - Dado un conjunto de nodos y los valores de las funciones correspondientes, los valores de las funciones se denominan diferencias divididas iniciales y se denotan por $f_0(x_k)$

  - La primera diferencia dividida se forma a través de las diferencias divididas iniciales, y la segunda a partir de las primeras diferencias divididas. La fórmula general, por tanto, es la siguiente:

$$f_1(x_k) = \frac{f_0(x_{k+1}) - f_0(x_k)}{x_{k+1} - x_k} \;\Rightarrow\; f_1(x_k) = \frac{f_0(x_{k+1}) - f_0(x_k)}{x_{k+1} - x_k}$$

$$\Rightarrow \ldots \Rightarrow f_j(x_k) = \frac{f_{j-1}(x_{k+1}) - f_{j-1}(x_k)}{x_{k+j} - x_k}$$

- Usando los nodos iniciales $x_0, x_1, x_2, \ldots$, se calculan los valores de las funciones iniciales y se calculan las diferencias divididas para todos los pares de valores consecutivos hasta obtener todas diferencias dividas necesarias para un polinomio aproximador. Esto se resume en la tabla:

k	x_k	$f_0(x_k)$	$f_1(x_k)$	$f_2(x_k)$	$f_3(x_k)$
0	1	0			
			1		
1	2	1		$-\frac{1}{6}$	
			$\frac{1}{2}$		$\frac{1}{7}$
2	4	2		$-\frac{5}{21}$	
			$\frac{6}{7}$		
3	$\frac{1}{2}$	-1			

- Un polinomio interpolador, si se construye correctamente, tiene que reproducir los valores de los nodos exactamente, por lo que es fácil evaluar el código. Si los valores de los nodos se calculan correctamente, entonces el polinomio tiene que ser correcto, y si no, está mal

- Para poder evaluar qué tan buena aproximación da el polinomio interpolador se pueden usar varios resultados del error de una interpolación

  - Siendo $f \in C^{n+1}([a,b])$ y los nodos $x_k \in [a,b]$ para $0 \le k \le n$, entonces, para cada $x \in [a,b]$ existe una $\xi_x \in [a,b]$ tal que se cumple la siguiente:

$$f(x) - p_n(x) = \frac{w_n(x)}{(n+1)!} f^{(n+1)}(\xi_x)$$

$$where \;\; w_n(x) = \prod_{k=0}^{n} (x - x_k)$$

- Para cualquier $t \in [a, b]$ tal que $t \neq x_k$, se define la función auxiliar $G(x) = E(x) - [w(x)/w(t)]E(t)$ donde $E(x) = f(x) - p_n(x)$ y $w(x) = \prod_{k=0}^{n}(x - x_k)$. Por lo tanto, $G(x_k) = 0$ para $0 \leq k \leq n$, y $G(t) = 0$ para $t = x$, y así hay $n + 2$ puntos donde G es nula (en los $n + 1$ nodos y en $t = x$)

- Por lo tanto, por el teorema generalizado de Rolle, existe un punto $\xi \in [a, b]$ tal que $G^{n+1}(\xi) = 0$, de modo que se obtiene el siguiente resultado:

$$G^{(n+1)}(x) = E^{(n+1)}(x) - \frac{w^{(n+1)}(x)}{w(t)}E(t) =$$

$$= f^{(n+1)}(x) - \frac{(n+1)!}{w(t)}f^{(n+1)}(t)$$

$$\Rightarrow\ 0 = G^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \frac{(n+1)!}{w(t)}f^{(n+1)}(t)$$

$$\Rightarrow\ f^{(n+1)}(t) = \frac{w(t)}{(n+1)!}f^{(n+1)}(\xi)$$

- Al medir el error en ciertas aproximaciones, será útil medir el tamaño de una función, por lo que se vuelve a utilizar el concepto de norma, que es un mapeado de funciones de números reales no negativos que satisface ciertos axiomas. Una función es una norma es cualquier, denotado por $\|f\|$, que satisface las siguientes condiciones:

  $(1)\ \|f\| > 0\ \ for\ any\ function\ \ f\ \ that\ is\ not\ identically\ zero$

  $(2)\ \|\alpha f\| = |\alpha|\|f\|\ \ for\ any\ constant\ a$

  $(3)\ \|f + g\| \leq \|f\| + \|g\|\ \ \ for\ any\ two\ functions\ f\ \&\ g$

  - Algunas normas comunes son la norma infinita (definida solo si f es continua en el intervalo cerrado $[a, b]$) o la norma 2 (definida si f es tal que su cuadrado puede ser integrado sobre el intervalo $[a, b]$)

$$\|f\|_{2,[a,b]} = \left(\int_a^b [f(x)]^2\, dx\right)^{1/2} \quad \& \quad \|f\|_{\infty,[a,b]} = \max_{x\in[a,b]}|f(x)|$$

  - La más útil suele ser la norma infinita, dado que permite encontrar cotas superiores para funciones a través de encontrar

el máximo para el valor absoluto de la función (para el término de error que suele ser una función derivada k veces y evaluada en un punto ξ_x desconocido por el teorema de Rolle generalizado)

$$\left\|f^{(k)}(\xi_x)\right\|_\infty \Rightarrow \max_{\xi_x\in[a,b]}\left|f^{(k)}(\xi_x)\right|$$

$$\Rightarrow \; f^{(k)}(\xi_x) \le \left\|f^{(k)}(\xi_x)\right\|_\infty$$

- Ya se conoce el error de interpolación para $n = 1$ (la interpolación lineal), pero se puede extender para valores de n más grandes

  - Anteriormente, se ha que para visto $n = 1$ el error de interpolación está acotado de la siguiente manera:

$$|f(x) - p_1(x)| \le \frac{1}{8}(x_1 - x_0)^2 \max_{x_0\le x\le x_1}|f''(x)|, \qquad x \in [x_0, x_1]$$

$$\Rightarrow \|f(x) - p_1(x)\|_{\infty,I} \le \frac{1}{8}(x_1 - x_0)^2\|f''\|_{\infty,I}, \qquad I = [x_0, x_1]$$

  - Si $n = 2$ (la interpolación cuadrática) tiene la forma $f(x) - p_2(x) = (1/6)(x - x_0)(x - x_1)(x - x_2)f'''(\xi_x)$. Para poder obtener una cota superior como para la interpolación lineal, se asume que los nodos satisfacen $x_1 - x_0 = x_2 - x_1 = h$ para alguna $h > 0$, por lo que se puede obtener la siguiente cota:

$$(x - x_0)(x - x_1)(x - x_2) = \big((x - x_1) + h\big)(x - x_1)\big((x - x_1) - h\big)$$

$$= t(t^2 - h^2) \quad for \quad t = x - x_1$$

$$\Rightarrow \; |f(x) - p_2(x)| \le \frac{1}{6}\Big(\max_{-h\le t\le h}|t(t^2 - h^2)|\Big)\max_{x_0\le t\le x_2}|f'''(t)|$$

$$As \max_{-h\le t\le h}|t(t^2 - h^2)| = \frac{2}{3\sqrt{3}}h^3 :$$

$$\Rightarrow \; |f(x) - p_2(x)| \le \frac{1}{9\sqrt{3}}h^3 \max_{x_0\le t\le x_2}|f'''(t)|$$

$$\Rightarrow \; \|f - p_2\|_{\infty,I} \le \frac{1}{9\sqrt{3}}h^3\|f'''\|_{\infty,I}$$

  - Si $n = 3$ (la interpolación cúbica), se utiliza un argumento parecido al anterior para obtener la siguiente cota superior:

$$\|f - p_3\|_{\infty,I} \leq \frac{1}{24} h^4 \left\|f^{(4)}\right\|_{\infty,I}$$

- Un problema de la interpolación polinómica usando un grado alto del polinomio es que es un proceso potencialmente inestable, en el sentido de que los pequeños cambios en los datos pueden llevar a grandes cambios en el polinomio interpolador

  - Se considera $f(x)$ como la función que se quiere interpolar y $\hat{f}(x)$ como la función con un error de redondeo, y se asume que $f(x) - \hat{f}(x) = \epsilon(x)$ y que $\left\|f - \hat{f}\right\|_\infty = \|\epsilon\|_\infty = \epsilon_0$ para una ϵ_0 pequeña. El polinomio interpolador se calcula basado en la función con error de redondeo, de modo que tiene la forma lagrangiana y se quiere estimar el error $f(x) - \hat{p}(x)$

$$\hat{p}_n(x) = \sum_{i=0}^{n} L_i^{(n)}(x) \hat{f}(x_i)$$

  - Por conveniencia se define el interpolador ideal, basado en la función exacta f de la siguiente manera:

$$p_n(x) = \sum_{i=0}^{n} L_i^{(n)}(x) f(x_i)$$

  - El teorema anterior sobre el error de la interpolación da una expresión para el error entre la función a interpolar y la función interpoladora sin error de redondeo. No obstante, se puede obtener una expresión para el error entre la función a interpolar y el polinomio interpolador, en donde se divide el error de interpolación y el error de redondeo de los polinomios

$$f(x) - p_n(x) = \frac{1}{(n+1)!} \left(\prod_{i=0}^{n} (x - x_i) \right) f^{(n+1)}(\xi_x)$$

$$\Rightarrow f(x) - \hat{p}_n(x) = \big(f(x) - p_n(x)\big) + \big(p_n(x) - \hat{p}_n(x)\big)$$

  - A partir de esta expresión, se puede obtener una cota superior para el error de redondeo de los polinomios. Si se asume que el error de redondeo es pequeño y está acotado (lo cual es razonable), entonces se puede encontrar una cota superior que dependa de $O(u)$

$$|p_n(x) - \hat{p}_n(x)| = \left|\sum_{i=0}^{n} L_i^{(n)}(x)\left[f(x_i) - \hat{f}(x_i)\right]\right|$$

$$\leq \|\epsilon\|_\infty \sum_{i=0}^{n} \left\|L_i^{(n)}\right\|_\infty = \epsilon_0 \sum_{i=0}^{n} \Lambda_i \quad where \quad \left\|L_i^{(n)}\right\|_\infty = \Lambda_i$$

$$\|\epsilon\|_\infty = \epsilon_0 = O(u) \;\Rightarrow\; |p_n(x) - \hat{p}_n(x)| \leq O(u) \sum_{i=0}^{n} \Lambda_i$$

- Lo malo de esta cota superior es que la suma de la que depende puede aumentar mucho si n incrementa. Esto se puede mostrar sin la necesidad de un teorema concreto, sino con un ejemplo analítico de por qué incrementa

  - Asumiendo que los nodos son equidistantes en un intervalo $[a, b]$ con $x_0 = a$ y $x_n = b$, y con $x_{j+1} - x_j = h$ siendo el espaciado uniforme, entonces se pueden escribir las siguientes expresiones:

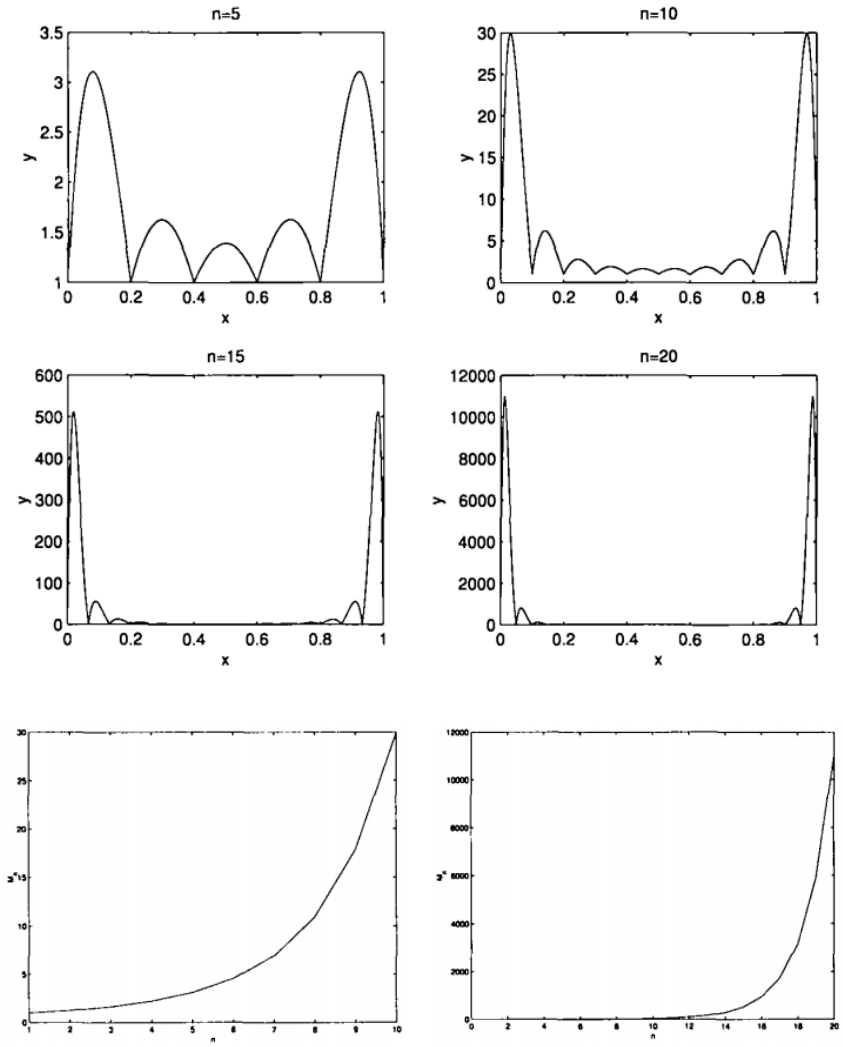
$$x_j = a + jh \quad for \;\; 0 \leq j \leq n$$

$$x = a + \eta_x h \quad where \;\; \eta_x = \frac{x - a}{h} \quad for \;\; 0 \leq \eta_x \leq n$$

  - Por lo tanto, las funciones de Lagrange (y, por tanto, las Λ_i) no dependen de la selección de a, b y h, sino que dependen enteramente de n, η_x (que depende de x) y de la distribución de los nodos

$$L_i^{(n)}(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k} = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{\eta_x - k}{i - k}$$

  - Esto significa que uno es capaz de graficar las funciones de Lagrange fácilmente y tener una idea de qué tan grande puede ser el término de la suma. Los primeros cuatro gráficos muestran $L(x) = \sum_{i=0}^{n} \left|L_i^{(n)}(x)\right|$, y los siguientes muestran los gráficos para $M_n = \sum_{i=0}^{n} \Lambda_i$ como una función de n



- Para el problema de interpolación numérico original, esto quiere decir que, a través del error de interpolación general con el polinomio interpolador con error de redondeo, se puede obtener la siguiente cota superior:

  - Como M_n puede crecer mucho, se sabe que la presencia de la ϵ_0 es pequeña, multiplicando el término de error de redondeo no es suficiente para garantizar que el error no será corrompido por el error de redondeo. Por otro lado, si se mantiene la interpolación polinómica por debajo de unos grados, entonces el error no se verá seriamente corrompido

$$\|f - \hat{p}_n\|_\infty \leq \|f - p_n\|_\infty + \|p_n - \hat{p}_n\|_\infty \leq \|f - p_n\|_\infty + \epsilon_0 M_n$$

  - No se está diciendo que la interpolación polinómica de un grado alto tiene que estar muy afectada por el error de redondeo necesariamente, sino que permite que haya un gran impacto de los errores de redondeo cuando n es lo suficientemente grande,

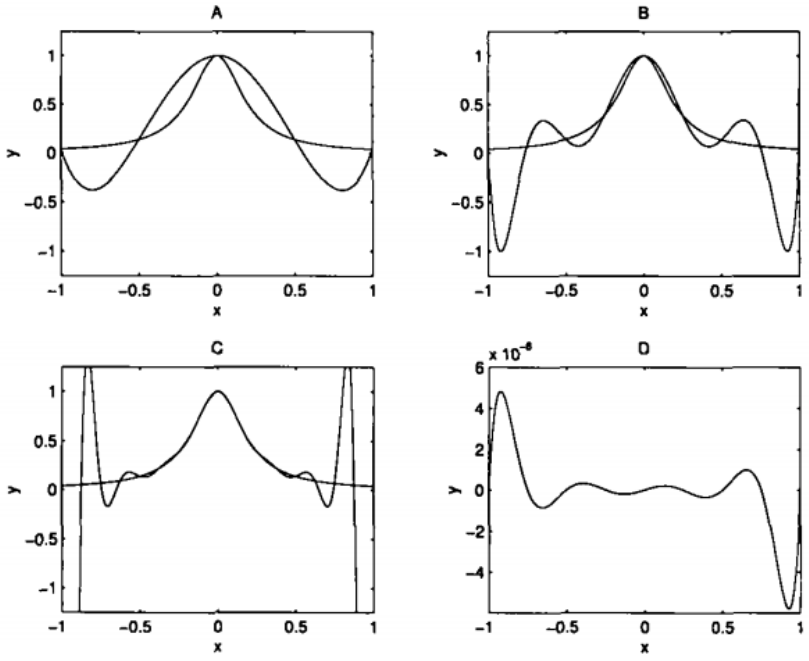
y que se puede evitar el problema potencial tomando una n tal que M_n sea pequeña

- En general, no es verdad que el error de interpolación tiende a ser nulo cuantos más grados n se utilizan para la aproximación, dado que este límite puede ser infinito para algunos puntos. A este fenómeno se le conoce como el fenómeno de Runge

$$\lim_{n\to\infty} \max_{x_0 \le x \le x_n} \|f - \hat{p}_n\| \neq 0$$

  - El fenómeno expresa que existe un problema de oscilación en los extremos del intervalo en donde se hace la interpolación polinómica si se usa un polinomio de grado alto con nodos equiespaciados

  - Esto se suele ejemplificar con el ejemplo de Runge. Un polinomio interpolador puede parecer suave y un buen aproximador para una función concreta para toda x, pero este puede no ser el caso cuando se deja que x tome valores imaginarios, haciendo que se causen problemas



- En el ejemplo analítico anterior para el efecto de un error de redondeo para un polinomio interpolador de un grado lo suficientemente alto, se puede ver como una distribución equidistante de los nodos puede ser una mala idea. Sin embargo, existe una elección alternativa para la distribución de nodos, llamados nodos de Chebyshev

- Se define la familia de funciones $T_n(x)$ para $n \geq 0$ por la siguiente fórmula:

$$T_n(x) = \cos(n \arccos(x)) \quad for \quad x \in [-1,1]$$

- Estas funciones son polinomios (polinomios de Chebyshev), y estas satisfacen las siguientes propiedades:

(1) $Each\ T_n\ is\ a\ polynomial\ of\ degree\ n$

(2) $For\ n \geq 1, T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$

(3) $T_n(x) = 2^{n-1}x^n + lower - order\ terms$

- Escribiendo $x = \cos(\theta)$, $T_n(x) = \cos(n\theta)$, por lo que, usando una identidad trigonométrica para el coseno, es posible obtener la siguiente expresión que demuestra (2):

$$\cos(n\theta)\cos(\theta) = \frac{1}{2}\left[\cos\big((n+1)\theta\big) + \cos\big((n-1)\theta\big)\right]$$

$$\Rightarrow \cos\big((n+1)\theta\big) = 2\cos(\theta)\cos(n\theta) - \cos\big((n-1)\theta\big)$$

$$\Rightarrow T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

- Para demostrar (1), se puede ver como $T_n(x) = 1$ para $n = 0$ y como $T_n(x) = x$ para $n = 1$. Asumiendo que $T_n(x)$ es un polinomio para todas las $n \leq k$ y a través de un argumento inductivo basado en la recursión en (2), se puede obtener el siguiente resultado que demuestra (1) y (3):

$$T_2(x) = 2xT_1(x) - T_0(x) = 2x^2 - 1$$

$$\Rightarrow T_3(x) = 2xT_2(x) - T_1(x) = 4x^3 - 3x \Rightarrow \cdots$$

- Es interesante ver que todas las familias polinómicas ortogonales satisfacen una recurrencia de tres términos, similar a la vista

- Asumiendo que uno solo está interesado en construir una aproximación en el intervalo $[-1,1]$, cada T_n tendrá exactamente n raíces (debido a que es un polinomio de grado n), las cuales serán distintas y todas estarán dentro del intervalo $[-1,1]$. Denotando las raíces del polinomio T_n de grado n como $z_k^{(n)}$, entonces se obtiene la siguiente igualdad:

$$z_k^{(n)} = \cos\left(\frac{(2k-1)\pi}{2n}\right) \quad for \ 1 \le k \le n$$

- Los nodos de Chebyshev son simplemente las raíces de los polinomios T_n. Para construir una interpolación polinómica de grado n, usando las $n+1$ raíces de T_{n+1}, se toman los nodos interpoladores de la siguiente manera:

$$x_k = z_{k+1}^{(n+1)}$$

- Los nodos de Chebyshev se definen de la siguiente manera:

$$x_k = \cos\left(\frac{(2k+1)\pi}{2(n+1)}\right) \quad for \ 0 \le k \le n$$

- El por qué estos nodos funcionan mejor requeriría mucha matemática adicional, pero una respuesta más simple pero rigurosa se puede dar a través del teorema básico del error de interpolación

  - Siendo $f \in C^{n+1}([-1,1])$ una función dada y p_n el polinomio de grado n para interpolar f, usando los nodos de Chebyshev se puede obtener la siguiente desigualdad:

$$\|f - p_n\|_\infty \le \frac{1}{2^n(n+1)!}\left\|f^{(n+1)}\right\|_\infty$$

  - Como los polinomios se definen únicamente para sus raíces (y multiplicados por un número constante), y como w_n y T_{n+1} tienen las mismas raíces, entonces se obtiene el siguiente resultado para una c_n constante:

$$f(x) - p_n(x) = \frac{w_n(x)}{(n+1)!}f^{(n+1)}(\xi_x) \quad where$$

$$w_n(x) = \prod_{i=0}^{n}(x - x_i)$$

$$\Rightarrow \ w_n = c_n T_{n+1}$$

  - Como T_{n+1} es un polinomio mónico (el coeficiente de orden más alto es 1), entonces c_n debe ser el recíproco del coeficiente líder para T_{n+1} (para que sea Mónico). Por lo tanto, por el teorema

anterior de los polinomios de Chebyshev, se pueden obtener el siguiente resultado:

$$w_n(x) = x^{n+1} + lower - order\ terms$$

$$\Rightarrow\ c_n = 2^{-n}$$

- No obstante, el polinomio de Chebyshev es un coseno en el intervalo $[-1,1]$, lo que permite obtener las siguientes desigualdades:

$$|T_{n+1}(x)| \leq 1 \quad for \quad -1 \leq x \leq 1$$

$$\Rightarrow\ |w_n(x)| = |c_n T_{n+1}(x)| \leq 2^{-n}$$

- Lo que muestra este teorema y su demostración es que la elección de los nodos de Chebyshev permite que se acote el factor que depende de $w_n(x)$ en el teorema del error de interpolación

  - Se puede demostrar que cualquier otra elección de nodos conllevará una $\|w_n(x)\|_\infty$ más grande, por lo que los nodos de Chebyshev son óptimos en este sentido

  - No obstante, no son perfectos, dado que se pueden encontrar funciones para las cuales los nodos de Chebyshev fallan, aunque estos ejemplos sean muy rebuscados

- Hasta ahora, todas las interpolaciones han involucrado únicamente el conocimiento de los valores de las funciones, pero no incluían información de, por ejemplo, las derivadas de estas funciones. La interpolación de Hermite permite incluir esta información a la hora de interpolar

  - La construcción de un polinomio interpolador como se ha visto no es un proceso interactivo: si la curva obtenida no es la que se desea, la única manera de mejorar potencialmente el resultado es añadir otro nodo

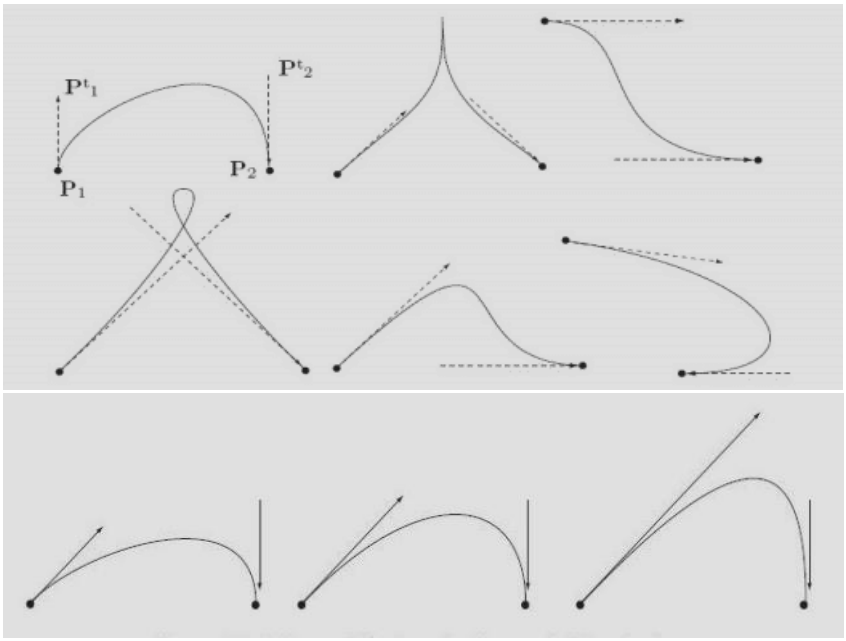
    - Esto incrementa el coste computacional del algoritmo de construcción del polinomio interpolador sin que haya ninguna garantía de que la curva será mejor

    - Un buen método de interpolación debería ser interactivo, dejando que el usuario pueda añadir nuevos datos para generar la curva de manera intuitiva. Esto se puede realizar a través de la interpolación de Hermite

- Para más especificidad, se quiere un polinomio $p(x)$ que interpola $f(x)$ en el sentido de que $p(x_i) = f(x_i)$ y $p'(x_i) = f'(x_i)$ para $1 \leq i \leq n$ (ahora ya no se comienza desde 0)

    - Este problema se conoce como el problema de interpolación de Hermite, y se puede conseguir tal polinomio interpolador. Esto se sustenta del teorema de interpolación de Hermite

    - En este caso, cambiando la magnitud de las derivadas, es posible obtener curvas con varias formas diferentes (teniendo puntas y hasta bucles en su forma). Las curvas también cambian con respecto a la dirección de las derivadas



    - Aunque no parezca intuitivo, es necesario tener en cuenta la magnitud de las derivadas debido a que, si no, entonces cualquier tangente (derivada) paralela a la tangente original sería equivalente

- Dados los n nodos x_i para $1 \leq i \leq n$ y siendo una función diferenciable $f(x)$, si los nodos son distintos, entonces existe un polinomio único H_n de grado menor o igual a $2n - 1$ tal que se cumplen las siguientes condiciones:

$$H_n(x_i) = f(x_i) \quad \& \quad H_n'(x_i) = f'(x_i) \quad for \ 1 \leq i \leq n$$

- Considerando $h_k(x) = \left[1 - 2\left(L_k^{(n)}\right)' x_k(x - x_k)\right]\left[L_k^{(n)}(x)\right]^2$ y $\tilde{h}_k(x) = (x - x_k)\left[L_k^{(n)}(x)\right]^2$, se pueden obtener los siguientes resultados:

$$h_k(x) = \delta_{kj} \quad \& \quad h_k'(x_j) = 0$$

$$\tilde{h}_k(x) = 0 \quad \& \quad \tilde{h}_k'(x_j) = \delta_{kj}$$

- Por lo tanto, el siguiente polinomio satisface las condiciones de interpolación del problema de Hermite y es claramente de grado menor o igual a $2n - 1$

$$H_n(x) = \sum_{k=1}^{n} \left[f(x_k)h_k(x) + f'(x_k)\tilde{h}_k(x)\right]$$

- Igual que la forma Lagrangiana forma una interpolación ordinaria, esta forma del polinomio de Hermite no conduce a una computación eficiente. Una variación del enfoque de Newton, incluyendo el uso de tablas de diferencias divididas, puede construirse

  - Se construye una tabla de diferencias divididas igual que antes, solo que se introducen los datos funcionales dos veces y se usa la derivada para la parte de las primeras diferencias divididas:

k	x_k	$f_0(x_k)$	$f_1(x_k)$
1	a	$f(a)$	
			$f'(a)$
1	a	$f(a)$	
2	b	$f(b)$	
			$f'(b)$
2	b	$f(b)$	

  - Después de eso, se crea la tabla de diferencias divididas de manera normal:

k	x_k	$f_0(x_k)$	$f_1(x_k)$	$f_2(x_k)$	$f_3(x_k)$
1	a	$f(a)$			
			$f'(a)$		
1	a	$f(a)$		B	
			A		D
2	b	$f(b)$		C	
			$f'(b)$		
2	b	$f(b)$			

$$A = \frac{f(b) - f(a)}{b - a} \qquad B = \frac{A - f'(a)}{b - a}$$

$$C = \frac{f'(b) - A}{b - a} \qquad D = \frac{C - B}{b - a}$$

- El polinomio de Hermite que interpola a f y f' para a y b, por tanto, es un polinomio cúbico y se puede definir de la siguiente manera:

$$H_2(x) = f(a) + f'(a)(x - a) + B(x - a)^2 + D(x - a)^2(x - b)$$

- Siendo $f \in C^{2n}([a, b])$ y teniendo nodos $x_k \in [a, b]$ para toda k cumpliendo $1 \leq k \leq n$, entonces cada $x \in [a, b]$, existe una $\xi_x \in [a, b]$ tal que se cumple la siguiente igualdad:

$$f(x) - H_n(x) = \frac{\psi_n(x)}{(2n)!} f^{(2n)}(\xi_x) \quad where \quad \psi_n(x) = \prod_{k=1}^{n} (x - x_k)^2$$

  - Para cualquier $t \in [a, b]$ tal que $t \neq x_k$, se define la función auxiliar $G(x) = E(x) - E(t)\big(\psi_n(x)/\psi_n(t)\big)$ donde $E(x) = f(x) - H_n(x)$. Igual que en el caso lagrangiano, en este caso $G(x_k) = 0$ para $0 \leq k \leq n$, y $G(t) = 0$ para $t = x$, de modo que hay $n + 2$ puntos donde G es nula

  - A través del teorema generalizado de Rolle, se permite obtener un punto $\xi \in [a, b]$ tal que se cumple la siguiente igualdad:

$$0 = G^{(2n)}(\xi) = f^{(2n)}(\xi) - \frac{(2n)!}{\psi_n(t)} E(t)$$

$$\Rightarrow \quad f(t) - H_n(t) = \frac{\psi_n(t)}{(2n)!} f^{(2n)}(\xi)$$

- Ahora que se ha introducido el polinomio interpolador de Hermite, se puede hacer una generalización algebraica de este, dando lugar a un

mejor entendimiento de lo que son las curvas de Hermite. Una curva de Hermite es un polinomio de tercer grado con la siguiente forma (forma algebraica de una curva paramétrica):

$$H_2(t) = \boldsymbol{a}^T \boldsymbol{T}(t) = [a \quad b \quad c \quad d] \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} = at^3 + bt^2 + ct + d$$

$$for \;\; t \in [0,1]$$

- La tangente de esta curva se puede escribir de la siguiente manera:

$$H_2'(t) = \boldsymbol{a}^T \frac{\partial \boldsymbol{T}(t)}{\partial t} = [a \quad b \quad c \quad d] \begin{bmatrix} 3t^2 \\ 2t \\ 1 \\ 0 \end{bmatrix} = 3at^2 + 2bt + c$$

- Con esta representación como una curva paramétrica, entonces se necesita un método para poder determinar los coeficientes de la curva. Si se consideran dos puntos $\boldsymbol{P_1}$ y $\boldsymbol{P_2}$ y sus vectores tangentes $\boldsymbol{\nabla_1}$ y $\boldsymbol{\nabla_2}$, se tienen cuatro ecuaciones vectoriales para poder determinar todas las constantes de la curva:

$$\begin{cases} H_2(t) = \boldsymbol{P_1} \\ H_2(t) = \boldsymbol{P_2} \\ H_2'(t) = \boldsymbol{\nabla_1} \\ H_2'(t) = \boldsymbol{\nabla_2} \end{cases}$$

- Resolviendo estas ecuaciones para cada una de las constantes, se obtiene la forma funcional de cada una de las constantes en función de los nodos o puntos y de las derivadas o vectores tangentes

$$\begin{cases} H_2(t) = \boldsymbol{P_1} \\ H_2(t) = \boldsymbol{P_2} \\ H_2'(t) = \boldsymbol{\nabla_1} \\ H_2'(t) = \boldsymbol{\nabla_2} \end{cases} \Rightarrow \begin{cases} a = g_1(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}) \\ b = g_2(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}) \\ c = g_3(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}) \\ d = g_4(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}) \end{cases}$$

- Una vez se tienen estas expresiones para los coeficientes, se pueden meter dentro del polinomio anterior y agrupar términos para $\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}$ y $\boldsymbol{\nabla_2}$ con tal de formar polinomios cúbicos $F(t)$ que servirán de coeficientes:

$$H_2(t) = at^3 + bt^2 + ct + d =$$

$$= g_1(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2})t^3 + g_2(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2})t^2$$

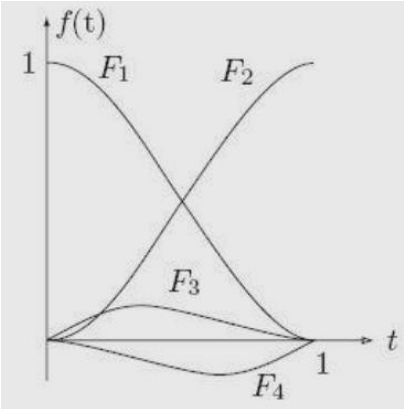
$$+g_3(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2})t + g_4(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}) =$$

$$= F_1(t)\boldsymbol{P_1} + F_2(t)\boldsymbol{P_2} + F_3(t)\boldsymbol{\nabla_1} + F_4(t)\boldsymbol{\nabla_2}$$

$$\Rightarrow \quad H_2(t) = \boldsymbol{b}^T\boldsymbol{F}(t) = [\boldsymbol{P_1} \quad \boldsymbol{P_2} \quad \boldsymbol{\nabla_1} \quad \boldsymbol{\nabla_2}] \begin{bmatrix} F_1(t) \\ F_2(t) \\ F_3(t) \\ F_4(t) \end{bmatrix} =$$

$$= F_1(t)\boldsymbol{P_1} + F_2(t)\boldsymbol{P_2} + F_3(t)\boldsymbol{\nabla_1} + F_4(t)\boldsymbol{\nabla_2}$$

- Las funciones $F_i(t)$ del vector $\boldsymbol{F}(t)$ se conocen como funciones de mezcla de Hermite o *blending functions*, y estas permiten generar la curva para cualquier punto cuando se saben los valores de las cuatro magnitudes $\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}$ y $\boldsymbol{\nabla_2}$



- El vector $\boldsymbol{F}(t)$ se puede reescribir como un producto entre el vector $\boldsymbol{T}(t)$ y una matriz de coeficientes $\boldsymbol{H}$, llamada matriz base de Hermite, haciendo posible reescribir también la curva de Hermite:

$$\boldsymbol{F}(t) = [\boldsymbol{T}(t)]^T\boldsymbol{H} = [t^3 \;\; t^2 \;\; t \;\; 1] \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

$$\Rightarrow H_2(t) = [\boldsymbol{T}(t)]^T\boldsymbol{H}\boldsymbol{b} = [t^3 \;\; t^2 \;\; t \;\; 1] \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix} \begin{bmatrix} \boldsymbol{P_1} \\ \boldsymbol{P_2} \\ \boldsymbol{\nabla_1} \\ \boldsymbol{\nabla_2} \end{bmatrix}$$

- A través de esta expresión y de darse cuenta que el único término que depende de t es $\boldsymbol{T}(t)$ (y, por tanto, $\boldsymbol{F}(t)$), se pueden encontrar las derivadas del polinomio interpolador de Hermite $H_2(t)$ derivando $\boldsymbol{T}(t)$ o $\boldsymbol{F}(t)$

$$H_2(t) = [\boldsymbol{T}(t)]^T \boldsymbol{H}\boldsymbol{b} \;\Rightarrow\; H_2'(t) = [\boldsymbol{T}'(t)]^T \boldsymbol{H}\boldsymbol{b}$$

$$\Rightarrow\; H_2''(t) = [\boldsymbol{T}''(t)]^T \boldsymbol{H}\boldsymbol{b} \;\Rightarrow\; \ldots$$

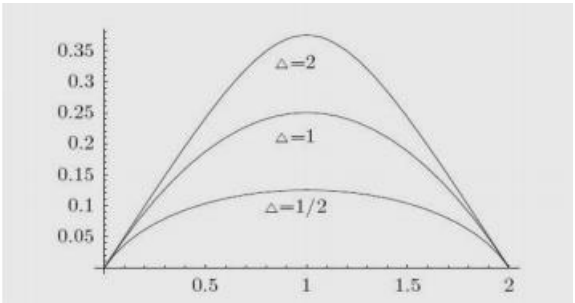
$$H_2(t) = \boldsymbol{F}(t)\boldsymbol{b} \;\Rightarrow\; H_2'(t) = \boldsymbol{F}'(t)\boldsymbol{b}$$

$$\Rightarrow\; H_2''(t) = \boldsymbol{F}''(t)\boldsymbol{b} \;\Rightarrow\; \ldots$$

- Los segmentos de la curva de Hermite con tensión son curvas de Hermite en la que se aplica un parámetro de tensión, generándose a partir de curvas no uniformes. Para poder hacer eso, se utiliza un parámetro t que varía en $[0, \Delta]$ donde $\Delta \in \mathbb{Z}^+$, obteniendo los siguientes resultados:

$$H_2(t) = [\boldsymbol{T}(t)]^T \boldsymbol{H}_{\boldsymbol{\Delta}}\boldsymbol{b} = [t^3 \;\; t^2 \;\; t \;\; 1] \begin{bmatrix} \frac{h_{11}}{\Delta^3} & \frac{h_{12}}{\Delta^3} & \frac{h_{13}}{\Delta^2} & \frac{h_{14}}{\Delta^2} \\ \frac{h_{21}}{\Delta^2} & \frac{h_{22}}{\Delta^2} & \frac{h_{23}}{\Delta} & \frac{h_{24}}{\Delta} \\ \frac{h_{31}}{\Delta} & \frac{h_{32}}{\Delta} & \frac{h_{33}}{1} & \frac{h_{34}}{1} \\ \frac{h_{41}}{1} & \frac{h_{42}}{1} & \frac{h_{43}}{1} & \frac{h_{44}}{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{P_1} \\ \boldsymbol{P_2} \\ \boldsymbol{\nabla_1} \\ \boldsymbol{\nabla_2} \end{bmatrix}$$

- En este caso, la matriz $\boldsymbol{H}_{\boldsymbol{\Delta}}$ con tensión se reduce a la matriz $\boldsymbol{H}$ cuando $\Delta = 1$. El efecto de incrementar Δ es el de crear más curvatura (elevar la curva), mientras que el de reducir Δ hace que se tenga menos curvatura, debido a que el incremento de Δ es equivalente a incrementar el módulo o norma de los vectores tangentes en los extremos de la curva



- Este caso de la curva de Hermite no uniforme es un caso especial de la curva de Hermite uniforme, dado que se puede cambiar de una a otra ajustando los valores de los vectores tangentes o ajustando el valor del parámetro t. Esta variación se entiende como un cambio en la tensión de la curva

- Las ideas de la curva de Hermite puede extenderse a polinomios de un grado más alto, aunque se necesitarán más datos iniciales para los coeficientes polinómicos. Con tal de incrementar el grado del polinomio, se usan datos de derivadas de un grado mayor (no de más vectores tangentes)

  - Usando los mismos puntos y vectores tangentes, se añade la segunda derivada para cada uno de los vectores tangentes, obteniendo así vectores de aceleración $\boldsymbol{\nabla_1^{(2)}}$ y $\boldsymbol{\nabla_2^{(2)}}$. De esta manera, se tiene suficiente información para construir un polinomio de grado 5:

$$H_4(t) = \boldsymbol{a}^T \boldsymbol{T}(t) = [a \quad b \quad c \quad d \quad e \quad f] \begin{bmatrix} t^5 \\ t^4 \\ t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} =$$

$$= at^5 + bt^4 + ct^3 + dt^2 + et + f$$

  - En este caso, los coeficientes polinómicos se obtienen de manera similar a la vista anteriormente:

$$\begin{cases} H_4(t) = \boldsymbol{P_1} \\ H_4(t) = \boldsymbol{P_2} \\ H_4'(t) = \boldsymbol{\nabla_1} \\ H_4'(t) = \boldsymbol{\nabla_2} \\ H_4'(t) = \boldsymbol{\nabla_1^{(2)}} \\ H_4'(t) = \boldsymbol{\nabla_1^{(2)}} \end{cases} \Rightarrow \begin{cases} a = g_1\left(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}, \boldsymbol{\nabla_1^{(2)}}, \boldsymbol{\nabla_2^{(2)}}\right) \\ b = g_2\left(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}, \boldsymbol{\nabla_1^{(2)}}, \boldsymbol{\nabla_2^{(2)}}\right) \\ c = g_3\left(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}, \boldsymbol{\nabla_1^{(2)}}, \boldsymbol{\nabla_2^{(2)}}\right) \\ d = g_4\left(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}, \boldsymbol{\nabla_1^{(2)}}, \boldsymbol{\nabla_2^{(2)}}\right) \\ e = g_5\left(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}, \boldsymbol{\nabla_1^{(2)}}, \boldsymbol{\nabla_2^{(2)}}\right) \\ f = g_6\left(\boldsymbol{P_1}, \boldsymbol{P_2}, \boldsymbol{\nabla_1}, \boldsymbol{\nabla_2}, \boldsymbol{\nabla_1^{(2)}}, \boldsymbol{\nabla_2^{(2)}}\right) \end{cases}$$

- La interpolación puede ser poco satisfactoria como herramienta de aproximación, sobre todo cuando se deja que el grado del polinomio interpolador crezca indefinidamente. No obstante, si se toma un orden fijo del

polinomio y se usan diferentes polinomios para diferentes intervalos que se van empequeñeciendo, esta puede ser muy exacta, lo cual motiva el uso de la interpolación por piezas y de las *splines*

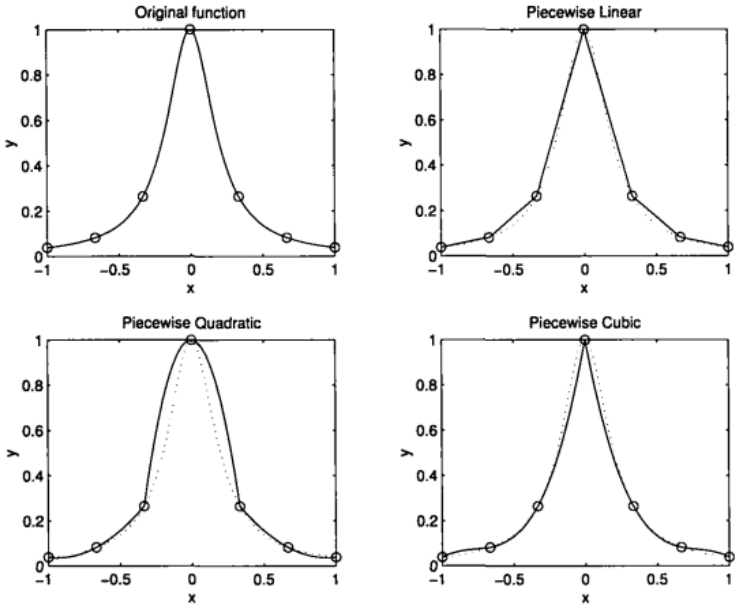
- Considerando el problema de construir una aproximación a la función $f(x)$ definida en n puntos en un intervalo I, si se usa la interpolaci´pn ordinaria, se puede construir un polinomio de $n-1$ grados, la cual se sospecha que tendrá resultados insatisfactorios. Pero, si se usa una interpolación lineal sobre cada subintervalo $[x_{k-1}, x_k]$ o una interpolación cuadrática en cada subintervalo $[x_{2k-2}, x_{2k}]$

    - Por lo tanto, se define la función aproximadora como un polinomio $p_{d,k}(x)$ de grado d:

$$q_d(x) = p_{d,k}(x) \quad for \quad x_{d(k-1)} \leq x \leq x_{dk}$$

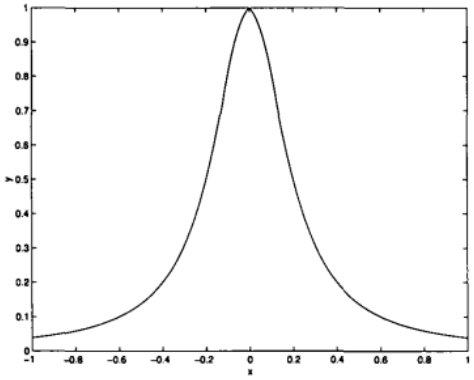
    - La exactitud de la aproximación proviene de las estimaciones de los errores vistas anteriormente, dado que la distancia entre nodos se asume como pequeña

- Un ejemplo claro es el de aproximar la función $f(x) = (1 + 25x^2)^{-1}$ sobre el intervalo $[-1,1]$ usando siete puntos equivalentemente espaciados $x_k = -1 + k/3$ para $k = 0,1, \dots ,6$. Para la aproximación, se usa interpolación lineal, cuadrática y cúbica por piezas



    - Las aproximaciones por piezas son más exactas que un polinomio interpolador de varios grados, y el contraste entre la

aproximación de varios grados con la aproximación por piezas es mayor cuantos más nodos se utilizan en la interpolación



- El uso de diferentes polinomios sobre diferentes intervalos, sin embargo, complica el proceso: ahora se tiene un conjunto de nodos x_k para $0 \leq k \leq n$ que define las condiciones de interpolación, y un segundo conjunto de puntos, llamados nudos o *knots*, que definen los subintervalos en los que se hace la separación de las piezas del polinomio)
    - En el desarrollo que se hace, se asume que los nudos son un subconjunto de los nodos, aunque este no tiene por qué ser el caso
    - Para un polinomio por partes de grado d, los nudos serán x_{dj} para $0 \leq j \leq m$, donde m es el número de piezas polinómicas en la aproximación. Para el ejemplo anteriormente visto, los nudos serían x_0, x_2, x_4 y x_6
- La aproximación polinómica es, de hecho, extraordinariamente útil para la aproximación, dado que usa diferentes polinomios en diferentes partes del dominio, lo cual permite imitar el rango del comportamiento de la función de manera más cercana
    - Para modelar exactamente las funciones más complicadas, es necesario asegurarse que las diferentes piezas se juntan de una manera que mantenga la mayor suavidad posible
    - Esto llevó naturalmente al concepto y la discusión de *splines*
- Un algoritmo de una aproximación polinómica por piezas generalmente consistirá de dos partes: una que construye la aproximación, usualmente en la forma de un arreglo o matriz de coeficientes polinómicos o de

coeficientes de diferencias divididas, y otra parte que usa el resultado de la primera parte para evaluar la aproximación

Algorithm 4.3 *Pseudo-code for Constructing Piecewise Polynomial Approximation*

```
input n, (x(i),y(i),i=0,n), d
!
     kount = 0
     for k=0 to n by d do
          kount = kount + 1
          for i=0 to d do
               xc(i) = x(k+i)
               yc(i) = y(k+i)
          endfor
!
!    compute divided difference coefficients
!    for the xc and yc arrays
!
          ac(0) = yc(0)
          for i=1 to d do
               w = 1
               p = 0
               for j=0 to i-1 do
                    p = p + ac(j)*w
                    w = w*(xc(i) - xc(j))
               endfor
               ac(i) = (yc(i) - p)/w
          endfor
          for i=0 to d do
               a(kount,i) = ac(i)
          endfor
!
!    This code evaluates the pcw poly at the point xx
!    First, find the subinterval containing the evaluation
!    point xx
!
     input n, d, kount, (x(i),i=0,n), (a(k,i),k=1,kount,i=0,d), xx
     j = search(x,xx)
!
!    Next, extract the correct nodes and DD coefficients
!
     for k=0 to d
          ac(k) = a(j,k)
          xc(k) = x(j+k)
     endfor
     yy = ac(j,d)
     for k = d-1 to 0 by -1
          xd = xx - xc(k)
          yy =  ac(k) + yy*xd
     endfor
```

- La evaluación de un polinomio por piezas requiere primero encontrar qué pieza del polinomio se usa para cada valor de x para los cuales se quiere calcular $q_d(x)$, y el pseudo-código asume que esto se hace en una rutina separada llamada $search()$

- Si el espaciado es uniforme, el cálculo $j = \lfloor (x - x_0)/h \rfloor$ garantiza que $x_j \leq x \leq x_{j+1}$, por lo que, a partir de esto, es fácil ver que pieza polinómica usar para calcular q_d. Si el espaciado no es uniforme, se tiene que usar un procedimiento de búsqueda más general para saber cuáles son los índices tales que $x_{(d-1)k} \leq x \leq x_{dk}$

- El teorema subyacente sobre el error es fácil de especificar y de demostrar. Siendo f una función lo suficientemente diferenciable en el intervalo $[a, b]$, y siendo q_d el polinomio interpolador por piezas de grado d para f en $[a, b]$, usando $n + 1$ con nodos igualmente espaciados x_k para $0 \leq k \leq n$ con $n = md$, entonces se cumple la siguiente desigualdad:

$$\|f - q_d\|_\infty \leq C_d h^{d+1} \left\| f^{(d+1)} \right\|_\infty \quad for \;\; d = 1,2,3$$

$$where \;\; C_1 = 1/8, C_2 = 1/9\sqrt{3} \;\; \& \;\; C_3 = 1/24$$

  - La demostración proviene directamente de las estimaciones de los errores anteriormente vistos para cada grado:

$$\|f - q_d\|_\infty = \max_{a \leq x \leq b} |f(x) - q_d(x)| =$$

$$= \max_{1 \leq k \leq n} \left(\max_{x_{k-1} \leq x \leq x_k} |f(x) - q_d(x)| \right) =$$

$$= \max_{1 \leq k \leq n} \left(\max_{x_{k-1} \leq x \leq x_k} \left| f(x) - p_{d,k}(x) \right| \right) =$$

$$= \max_{1 \leq k \leq n} \left\| f - p_{d,k} \right\|_{\infty,[x_{k-1},x_k]} \leq \max_{1 \leq k \leq n} C_d h^{d+1} \left\| f^{(d+1)} \right\|_{\infty,[x_{k-1},x_k]}$$

$$= C_d h^{d+1} \left\| f^{(d+1)} \right\|_{\infty,[a,b]} = C_d h^{d+1} \left\| f^{(d+1)} \right\|_\infty$$

- Mientras que la aproximación polinómica por piezas se usa mucho para una variedad de aplicaciones, es más importante como un precursor para el desarrollo de las aproximaciones de *splines*

  - Una aplicación interesante y una forma útil de una aproximación polinómica por piezas es el uso de la interpolación de Hermite cúbica por piezas: se usan polinomios cúbicos de Hermite como manera obvia de definir la aproximación polinómica entre cada par de nodos x_{k-1} y x_k

- Debido a que también se interpolan los valores de la derivada, la aproximación resultante es mucho más suave que la interpolación por piezas ordinaria y es menos probable que tengan puntas en el gráfico

- La aproximación polinómica por piezas permite construir aproximaciones muy exactas, pero no siempre produce aproximaciones suaves, dado que las juntas entre piezas separadas no son suaves (q_d no es continuamente diferenciable en todo su intervalo de aproximación). Las *splines* intentan solucionar este problema, cuya idea básica es construir un polinomio por piezas que no solo interpole los datos o valores de las funciones que se dan, sino que también sea suave (continuamente diferenciable hasta un grado concreto)

    - Suponiendo que se tiene un conjunto de nodos $\{x_i, 0 \leq i \leq n\}$ en el que se quiere inteprolar una función dada $f(x)$ con una *spline* de grado d, el problema es encontrar una función polinómica por piezas q_d tal que se satisfagan las siguientes condiciones:

      (1) $Interpolation$: $q_d(x_k) = f(x_k) \quad for \quad 0 \leq k \leq n$

      (2) $Smoothness$: $\lim_{x \to x_k^-} q_d^{(i)}(x) = \lim_{x \to x_k^+} q_d^{(i)}(x) \quad for \ 0 \leq i \leq N$

      (3) $Interval\ of\ definition$: $q_d\ is\ a\ polynomial\ of\ degree \leq d\ \ on\ each\ subinterval\ \ [x_{k-1}, x_k]$

        - En este caso, d se conoce como el grado de aproximación de la *spline*, mientras que N es el grado de suavidad de la *spline*

        - Se tiene que investigar la relación (si hay) entre el grado de aproximación d y el grado de suavidad N. El grado de los polinomios está relacionado al número de coeficientes desconocidos (los grados de libertad del problema), mientras que N se relaciona con el número de restricciones (se espera que los grados de libertad y el número de restricciones tienen que hacer un balance con tal de que la *spline* esté bien definida)

        - Como hay n subintervalos, cada uno siendo el dominio de definición para un polinomio de grado d, se tiene un total de $K_f = n(d+1)$ grados de libertad (parámetros del polinomio)

        - Por otro lado, hay $n+1$ condiciones de interpolación (para cada $f(x_0)$ hasta $f(x_n)$), $n-1$ puntos de unión (aunque en la construcción de una *spline*, ni x_0 ni x_n se consideran nudos) entre

subintervalos con $N+1$ condiciones de continuidad impuestos en cada uno de ellos (uno por cada nodo de x_0 hasta x_n). Por lo tanto, hay $K_c = n + 1 + (n-1)(N+1)$ restricciones

- Si se considera la diferencia entre K_f y K_c, que es la diferencia entre los grados de libertad y las restricciones para construir la *spline*, entonces se puede ver que hay $n(d-1-N)+N$ grados de libertad o restricciones faltantes

$$K_f - K_c = n(d+1) - [n+1+(n-1)(N+1)] =$$

$$= nd - n - nN + N = n(d-1-N) + N$$

  - Se puede hacer que el primer término sea nulo fijando que $N = d-1$, lo que establece una relación entre el grado del polinomio de la *spline* y del grado de suavidad. No obstante, no se tiene el mismo número de restricciones que de grados de libertad, porque $K_f - K_c = N$, haciendo necesaria la adición de N restricciones adicionales

  - En parte, por esa razón se suele preferir un número impar de orden de las *splines* se prefieren, porque si d es impar entonces $N = d-1$ es par y las restricciones adicionales se pueden imponer igualmente en los dos puntos extremos (lo cual es la elección típica) del intervalo

  - Es importante notar que se considera una *spline* con grado de suavidad $N = 0$ a una función lineal por piezas continua

- La construcción que se usa se para las *splines* cúbicas se basan en la *B-spline*: la idea usa una sola función ejemplar de la cual la base de *splines* se forma, y la aproximación se define en términos de esta base. La noción de *B-splines* es más general que la que se presenta, pero se restringe la discusión al caso de las *B-splines* cúbicas, con nodos y nudos coincidentes

  - Por simplicidad, se asume que los nodos se distribuyen uniformemente en un intervalo $[a,b]$ y que $x_k - x_{k-1} = h$ para toda $k \geq 1$. Además, se necesitarán definir 6 puntos extra y definir una función B que sea una *spline* cúbica (a través de un polinomio interpolador por partes)

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

$$x_{-3} = a - 3h \quad x_{-2} = a - 2h \quad x_{-1} = a - h$$

$$x_{n+1} = b + h \quad x_{n+2} = b + 2h \quad x_{n+3} = b + 3h$$

$$B(x) = \begin{cases} 0 & x \leq 2 \\ (x+2)^3 & -2 \leq x \leq 1 \\ 1 + 3(x+1) + 3(x+1)^2 - 3(x+1)^3 & -1 \leq x \leq 0 \\ 1 + 3(1-x) + 3(1-x)^2 - 3(1-x)^3 & 0 \leq x \leq 1 \\ (2-x)^3 & 1 \leq x \leq 2 \\ 0 & 2 \leq x \end{cases}$$

- La función $B(x)$ es una función *spline* cúbica porque se puede comprobar que es un polinomio cúbico por piezas, y entonces se calcula las primeras y segundas derivadas laterales en los nodos, y para los valores de las funciones también. Si los valores laterales son iguales, entonces la primera y la segunda derivada son continuas y B es una *spline* cúbica (si solo fuera la primera, entonces no sería una *spline* cúbica requiere que la segunda derivada sea continua también)

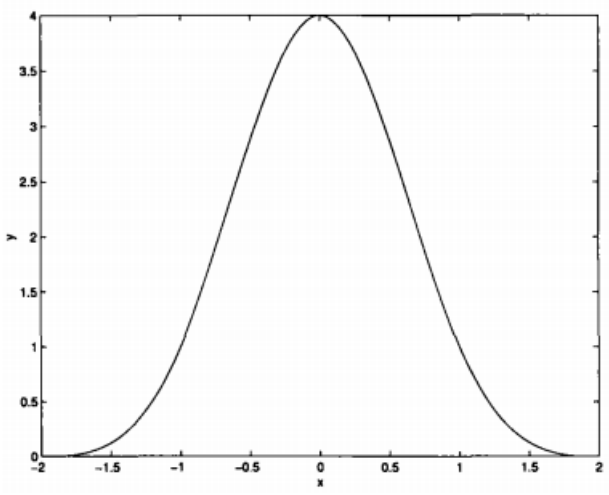
$$B'_-(x_k) = \lim_{x \to x_k^-} B'(x) \qquad B'_+(x_k) = \lim_{x \to x_k^+} B'(x)$$

- En este caso, la función particularmente definida tiene una forma de campana y sus valores son los siguientes:

$$B(0) = 4 \qquad B(\pm 1) = 1 \qquad B(\pm 2) = 0$$

$$B'(0) = 0 \qquad B'(\pm 1) = \mp 3 \qquad B'(\pm 2) = 0$$

$$B''(0) = -12 \qquad B''(\pm 1) = 5 \qquad B''(\pm 2) = 0$$



- La función B se define localmente, significando que es diferente de cero solo en un intervalo pequeño. Esta propiedad de definición local es importante en la utilidad de B-*splines* como una herramienta de aproximación

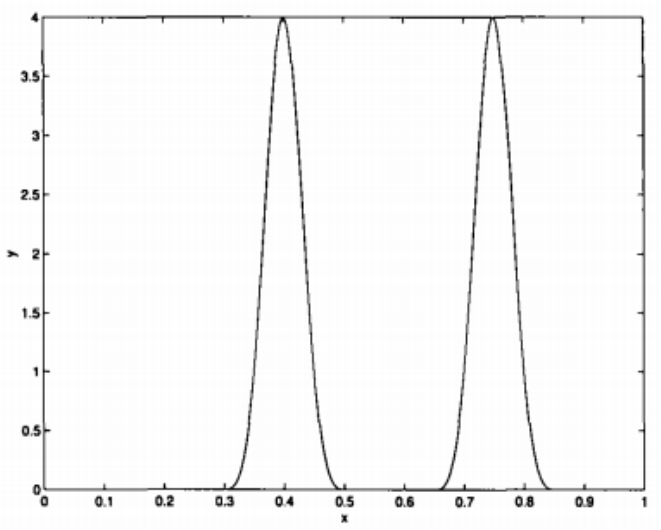
- Se puede usar B para construir una aproximación mediante una *spline* para una función arbitraria f usando las definiciones anteriores

  - Se define la secuencia de funciones $B_i(x) = B\big((x - x_i)/h\big)$ para $-1 \leq i \leq n + 1$, cada B_i es similar en forma a la función base B, pero se centra en x_i en vez de 0. Además, con esta definición se pueden plantear las siguientes igualdades:

$$B_i(x) = B(0) \qquad B_i(x_{i\pm1}) = B(\pm1) \qquad B_i(x_{i\pm2}) = B(\pm2)$$

$$B_i'(x) = 0 \qquad B_i'(x_{i\pm1}) = \mp\frac{3}{h} \qquad B_i'(x_{i\pm2}) = 0$$

$$B_i''(x) = -\frac{12}{h^2} \qquad B_i''(x_{i\pm1}) = \frac{6}{h^2} \qquad B_i''(x_{i\pm2}) = 0$$



  - Para construir un interpolador *spline* para una función dada f, se define la *spline* como una combinación lineal de funciones B_i y hacer que se busquen c_i para $-1 \leq i \leq n + 1$ tal que se cumpla $f(x_k) = \sum_{i=-1}^{n+1} c_i B_i(x_k)$

$$q_3(x) = \sum_{i=-1}^{n+1} c_i B_i(x)$$

$$with \quad c_i \quad s.t. \quad f(x_k) = \sum_{i=-1}^{n+1} c_i B_i(x_k) \quad for \quad 0 \leq k \leq n$$

- La definición local de B es muy útil ahora: para $k = 0$, se tendría la combinación $f(x_0) = c_{-1}B_{-1}(x_0) + c_0 B_0(x_0) + c_1 B_1(x_0)$ dado que $B_i(x_o) = 0$ para toda $i \geq 2$. En general, se tendría la siguiente expresión:

$$f(x_k) = c_{k-1}B_{k-1}(x_k) + c_k B_k(x_k) + c_{k+1}B_{k+1}(x_k)$$

- Por lo tanto, para cada ecuación en el sistema definido anteriormente $f(x_k) = \sum_{i=-1}^{n+1} c_i B_i(x_k)$ solo tiene tres términos que no son cero, haciendo que sea un sistema tridiagonal. Usando las igualdades anteriores, se pueden obtener los siguientes resultados:

$$f(x_k) = c_{k-1} + 4c_k + c_{k+1}$$

$$\begin{bmatrix} 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} c_{-1} \\ c_0 \\ \vdots \\ c_{n+1} \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

- En este caso, sigue siendo un sistema de $n + 1$ ecuaciones y $n + 3$ incógnitas, por lo que es necesario imponer dos restricciones adicionales con tal de eliminar las dos incógnitas. Las dos elecciones más comunes son la *spline* natural y la *spline* completa

  - En la *spline* natural se impone que $q_3''(x_0) = q_3''(x_n) = 0$, lo cual lleva a una construcción simple, pero causa que haya más error cerca de los puntos extremos del intervalo (la condición puede no ser adecuada siempre)

  - En la *spline* completa se impone que $q_3'(x_0) = f'(x_0)$ y que $q_3'(x_n) = f'(x_n)$, lo cual lleva a mejores propiedades de aproximación, y no requiere que realmente se sepan las derivadas en los puntos extremos

  - A partir de la forma funcional de B_i, es posible obtener $q_3''(x_0)$ y $q_3''(x_n)$ y, igualando ambos términos a cero, una ecuación que dependa de los coeficientes y así sustituirlos en el sistema anteriormente visto. Este sistema permitirá encontrar una expresión para los coeficientes en términos de las funciones en

los nodos x_0 y x_n, haciendo que el nuevo sistema (al sustituir las expresiones encontradas por los coeficientes) sea cuadrado (de $n-1$ incógnitas), tridiagonal y diagonalmente dominante, permitiendo que se aplique el algoritmo de solución para sistemas tridiagonales y así obtener los valores de los coeficientes de la *spline*

$$\begin{cases} q_3''(x_0) = h^{-2}[6(c_{-1}+c_1) - 12c_0] \\ q_3''(x_n) = h^{-2}[6(c_{n-1}+c_{n+1}) - 12c_n] \end{cases} \Rightarrow \begin{cases} c_{-1} = 2c_0 - c_1 \\ c_{n+1} = 2c_n - c_{n-1} \end{cases}$$

$$\begin{bmatrix} 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} c_{-1} \\ c_0 \\ \vdots \\ c_{n+1} \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

$$\Downarrow$$

$$\begin{cases} 6c_0 = f(x_0) \\ 6c_0 = f(x_n) \end{cases} \Rightarrow \begin{cases} c_0 = \dfrac{1}{6} f(x_0) \\ c_n = \dfrac{1}{6} f(x_n) \end{cases}$$

$$\Downarrow$$

$$\begin{array}{lllll} 4c_1 + c_2 & & & = & f(x_1) - \frac{1}{6} f(x_0), \\ c_1 + 4c_2 + c_3 & & & = & f(x_2), \\ \ddots & \ddots & & \vdots & \\ & c_{n-2} + 4c_{n-1} & & = & f(x_{n-1}) - \frac{1}{6} f(x_n). \end{array}$$

- A partir de la forma funcional de B_i, es posible obtener $q_3'(x_0)$ y $q_3'(x_n)$ y, igualando ambos términos a $f'(x_0)$ y $f'(x_n)$ respectivamente, una ecuación que dependa de los coeficientes y así sustituirlos en el sistema anteriormente visto. Este sistema permitirá encontrar una expresión para los coeficientes en términos de las derivadas en los nodos x_0 y x_n, haciendo que el nuevo sistema sea cuadrado (de $n+1$ incógnitas), tridiagonal y diagonalmente dominante, permitiendo que se aplique el algoritmo de solución para sistemas tridiagonales y así obtener los valores de los coeficientes de la *spline*

$$\begin{cases} q_3''(x_0) = -3h^{-1}(c_{-1}+c_1) \\ q_3''(x_n) = -3h^{-1}(c_{-1}+c_1) \end{cases} \Rightarrow \begin{cases} c_{-1} = c_1 - \dfrac{1}{3} h f'(x_0) \\ c_{n+1} = c_{n-1} - \dfrac{1}{3} h f'(x_n) \end{cases}$$

$$\begin{bmatrix} 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} c_{-1} \\ c_0 \\ \vdots \\ c_{n+1} \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

$$\Downarrow$$

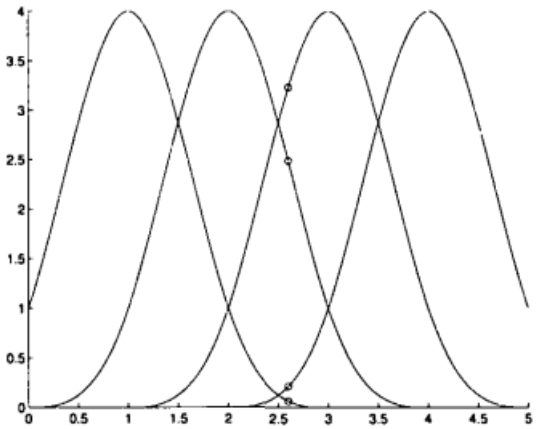
$$\begin{cases} c_{-1} = c_1 - \dfrac{1}{3}hf'(x_0) \\ c_{n+1} = c_{n-1} - \dfrac{1}{3}hf'(x_n) \end{cases}$$

$$\Downarrow$$

$$\begin{array}{ccccccccc} 4c_0 & + & 2c_1 & & \cdots & & & = & f(x_0) + \frac{1}{3}hf'(x_0), \\ \ddots & & \ddots & & \ddots & & \vdots & & \\ & c_{k-1} & + & 4c_k & + & c_{k+1} & = & & f(x_k), \\ & & \ddots & & \ddots & & \vdots & & \\ & & & 2c_{n-1} & + & 4c_n & = & & f(x_n) - \frac{1}{3}hf'(x_n), \end{array}$$

- Para cualquier $x \in [x_{k-1}, x_k]$, el hecho de que cada B_i solo sea diferente a cero de manera local significa que la evaluación de $q_3(x)$ requiere únicamente encontrar el índice k tal que $x_{k-1} \leq x \leq x_k$

$$q_3(x) = c_{k-2}B_{k-2}(x) + c_{k-1}B_{k-1}(x) + c_kB_k(x) + c_{k+1}B_{k+1}(x)$$



  - Para un intervalo dividido uniformemente, esto se puede conseguirse a través de $k = \lfloor (x - x_0)/h \rfloor + 1$, pero si se divide de manera no uniforme, entonces se necesitaría emplear una rutina de búsqueda más general

- Si $f \in C^4([a,b])$ y q es una *spline* cúbica de interpolación de f en un intervalo $a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$ con $\max(x_i - x_{i-1}) \leq h$ para toda i, entonces se cumple la siguiente desigualdad:

$$\left\| f^{(k)} - q^{(k)} \right\|_\infty \leq \frac{5}{384} h^4 \left\| f^{(4)} \right\|_\infty$$

  - Además, existen constantes C_k para $1 \leq k \leq 3$ para las cuales se cumple la siguiente igualdad:

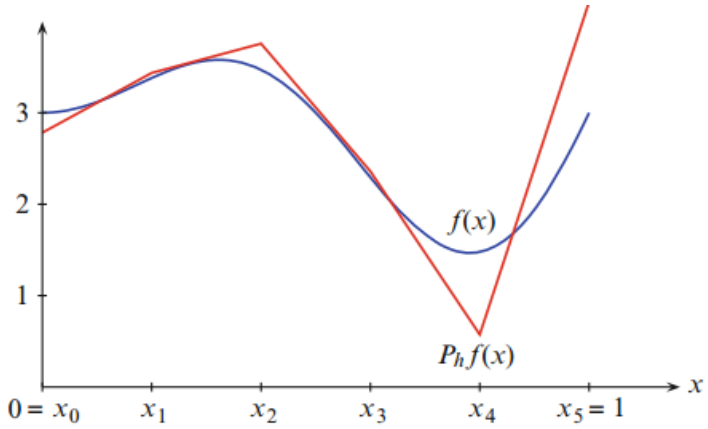
$$\left\| f^{(k)} - q^{(k)} \right\|_\infty \leq C_k h^{4-k} \left\| f^{(4)} \right\|_\infty$$

  - La interpolación por *splines* no es más precisa (en términos del exponente en el tamaño de paso h) que una aproximación polinómica por partes (la constante es menor). La ventaja de la interpolación por *splines* reside en la suavidad de la aproximación

- La noción de los mínimos cuadrados se puede extender más allá del problema de ajustar una función a los datos, y se puede introducir el concepto de las expansiones ortogonales, llevando al concepto de proyección ortogonal o proyección L_2

  - Considerando el problema de encontrar una aproximación a una función f dada en términos de un conjunto de funciones base $\{\phi_k, 1 \leq k \leq n\}$, se quieren encontrar los coeficientes de la siguiente expansión:

$$f(x) \approx q_n(x) = \sum_{k=1}^{n} c_k \phi_k(x)$$



  - Una manera de hacer esto es requerir que los coeficientes c_k produzcan una aproximación que minimiza el error en el sentido

de la norma L_2 cuadrada (mínimos cuadrados). Es por esto que al ejercicio de aproximar la función se le conoce como proyección L_2

$$R_n = \|r_n\|_2^2 = \|f - q_n\|_2^2$$

- Para poder obtener aquellos coeficientes, es necesario recurrir a la noción de producto interior

- Siendo f y g elementos del espacio vectorial real V, el producto interior es una operación $\langle f, g\rangle$ en f y g que cumple las siguientes condiciones:

  (1) $\langle f, f\rangle > 0, \qquad \forall f \in V \ \& \ \forall f \geq 0$

  (2) $\langle f, \alpha g_1 + \beta g_2\rangle = \alpha\langle f, g_1\rangle + \beta\langle f, g_2\rangle, \qquad \forall f, g \in V \ \& \ \forall \alpha, \beta \in \mathbb{R}$

  (3) $\langle f, g\rangle = \langle g, f\rangle, \qquad \forall f, g \in V$

  - Si se quiere considerar el espacio vectorial $C([a, b])$ (las funciones continuas en el intervalo $[a, b]$), entonces se puede establecer fácilmente que la integral de un producto de dos funciones sea un producto interior

  - Si w es una función tal que $\int_a^b w(x)\,dx$ está definida y $w(x) \geq 0$ para toda $x \in [a, b]$, para una f y g dadas en $C([a, b])$, se define el producto interior $\langle f, g\rangle_w$ en $C([a, b])$ de la siguiente manera:

$$\langle f, g\rangle_w = \int_a^b w(x) f(x) g(x)\,dx$$

  - Se puede ver que si $\langle f, g\rangle_w$ es un producto interior, entonces se cumple que $\|f\|_w = \langle f, g\rangle_w^{1/2}$ define una norma

- La definición del producto interior permite aplicar varias ideas del algebra lineal a la construcción de aproximaciones

  - Es posible usar la notación del producto interior para escribir el residual R_n de la siguiente forma:

$$R_n = \|f\|_w^2 - 2\sum_{k=1}^{n} c_k\langle f, \phi_k\rangle + \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j\langle \phi_i, \phi_j\rangle$$

- En este caso, se puede entender R_n como una función de las n variables c_k, y aplicar cálculo ordinario al problema de minimizar R_n. Después de aplicar una manipulación algebraica, se obtiene que para encontrar las c_k se tiene que resolver el siguiente sistema de ecuaciones de t:

$$\begin{bmatrix} \langle \phi_1, \phi_1 \rangle & \langle \phi_1, \phi_2 \rangle & \dots & \langle \phi_1, \phi_n \rangle \\ \langle \phi_2, \phi_1 \rangle & \langle \phi_2, \phi_2 \rangle & \dots & \langle \phi_2, \phi_n \rangle \\ \vdots & \vdots & \dots & \vdots \\ \langle \phi_n, \phi_1 \rangle & \langle \phi_n, \phi_2 \rangle & \dots & \langle \phi_n, \phi_n \rangle \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \langle f, \phi_1 \rangle \\ \langle f, \phi_2 \rangle \\ \vdots \\ \langle f, \phi_n \rangle \end{bmatrix}$$

$$\Rightarrow \boldsymbol{Mc} = \boldsymbol{b} \;\Rightarrow\; b_i = \sum_{k=0}^{n} M_{ik} c_k \quad for \;\; i = 1,2, \dots, n$$

- Por lo tanto, se obtiene un sistema de ecuaciones de tamaño $n \times n$, para los n coeficientes desconocidos. A la matriz $\boldsymbol{M}$ se la conoce como la matriz de masa o *mass matrix*, mientras que al vector $\boldsymbol{b}$ se le conoce como el vector de carga o *load vector*, y se suelen aproximar las integrales a través de integración numérica para obtener un algoritmo numérico para aproximar la proyección

- Es posible evitar la resolución de este sistema de ecuaciones si las funciones base que se utilizan satisfacen la condición de ortogonalidad, dado que, en ese caso, la matriz es diagonal y se tiene un resultado más simple:

$$\langle \phi_i, \phi_j \rangle = 0, \qquad \forall i \neq j \;\Rightarrow\; c_k = \frac{\langle f, \phi_k \rangle}{\langle \phi_k, \phi_k \rangle}$$

- Claramente, una de las maneras más sencillas (y en la que se basan varios métodos) de poder lograr esto es a través de usar la base de funciones *hat* vista anteriormente, de modo que $\phi_k = \varphi_k$

- Por lo tanto, el polinomio resultante será q_n con los coeficientes solución c_k para $k = 1,2, \dots, n$

$$q_n(x) = \sum_{k=1}^{n} c_k \phi_k(x)$$

- Como se ha visto, a partir de una base $\{\phi_k, 1 \leq k \leq n\}$ de funciones ortogonales se puede hacer una construcción sencilla de la expansión de

base que aproxima la función f. Lo importante es saber cuándo se puede encontrar una base que satisfaga la condición y qué tan bueno es el resultado de la aproximación

- La respuesta a esta pregunta es que siempre se puede encontrar esta base si se consideran funciones polinómicas para los elementos de la base, y las aproximaciones suelen ser bastante buenas

- Para cualquier $N \geq 0$, se define $\mathcal{P}_N$ como el espacio vectorial de polinomios de grado $\leq N$, que contiene una base estándar consistiendo de $\{1, x, x^2, \ldots, x^N\}$ y es un espacio $(n+1)$ - dimensional. Este espacio permite obtener el siguiente teorema para encontrar el tipo de base deseada

- Siendo w una función de ponderación no negativa en un intervalo $[a, b]$, y siendo $\langle \cdot, \cdot \rangle_w$ el producto interior asociado, entonces existe una familia de polinomios ortogonales $\{\phi_k : \phi_k \in \mathcal{P}_k,\ 0 \leq k \leq N\}$ tal que $\langle \phi_i, \phi_j \rangle_w = 0$ para $i \neq j$ y $\langle \phi_i, \phi_j \rangle_w > 0$ para $i \geq 0$. Además, ϕ_k satisface lo siguiente:

  - El conjunto $\{\phi_0, \phi_1, \ldots, \phi_N\}$ forma una base para $\mathcal{P}_N$

  - Si q_k es un elemento arbitrario de $\mathcal{P}_k$, para $k < N$, entonces $\langle q_k, \phi_N \rangle_w = 0$ para $N > k$. Por lo tanto, los polinomios ortogonales son ortogonales a todos los polinomios de un orden menor a k

  - Para $j \geq 1$, las raíces de cada ϕ_j están todas en $[a, b]$ y son todas distintas

La diferenciación numérica

- Uno de los usos más simples del teorema de Taylor como medios para construir aproximaciones involucra el uso de cocientes de diferencias para aproximar la derivada de una función conocida f. Esto es intuitivo de la definición de derivada, pero el reto está en saber que tan precisa es la aproximación en términos del paso h y encontrar maneras más exactas de aproximación

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

  - Aunque no parezca razonable usar métodos para aproximar la derivada para una función, dado que siempre se puede calcular la derivada y

después usar esa función, pero no siempre se tiene una fórmula fácil para f que permita calcular f' directamente

- La función f puede ser el resultado de un cálculo largo para el cual no existe una fórmula simple disponible, por lo que se necesita hacer una aproximación. Además, se pueden usar fórmulas desarrolladas en esta sección con tal de aproximar la derivada en un punto para funciones definidas por una tabla de valores

- También se pueden utilizar estas aproximaciones para reemplazar la derivada en otras ecuaciones con tal de producir esquemas de aproximación para otros problemas (como en el método de Euler)

- Para poder construir una fórmula en diferencias finitas para aproximar la derivada en un punto, primero se necesita un conjunto de puntos de interpolación en un intervalo alrededor de un punto x

$$\{(x_0, f(x_0)), (x_1, f(x_1)), \dots, (x_m, f(x_m))\}$$

- A través de utilizar una tabla para el cálculo de un polinomio interpolador, es posible obtener fórmulas para diferentes aproximaciones de la derivada. Considerando dos puntos $x_0 = x$ y $x_1 = x + h$, se obtiene la siguiente aproximación:

$$\begin{array}{c|c} a & f(a) \\ & \searrow \quad \dfrac{f(a+h)-f(a)}{h} \\ a+h & f(a+h) \quad \nearrow \end{array}$$

$$f'(x) \approx \frac{f(x+h)-f(x)}{h}$$

$$\Rightarrow \; P_1(x) = f(a) + \frac{f(a+h)-f(a)}{h}(x-a) \quad for \; f'(a)$$

- Si, en cambio, se consideran dos puntos $x_0 = x$ y $x_1 = x + h$, se obtiene la siguiente aproximación:

$$\begin{array}{c|c} a-h & f(a-h) \\ & \searrow \\ & \dfrac{f(a)-f(a-h)}{h} \\ & \nearrow \\ a & f(a) \end{array}$$

$$f'(x) \approx \frac{f(x)-f(x-h)}{h}$$

$$\Rightarrow \; P_1(x) = f(a) + \frac{f(a)-f(a-h)}{h}(x-a+h) \quad for \; f'(a)$$

- Si, en cambio, se consideran dos puntos $x_0 = x-h$ y $x_1 = x+h$, se obtiene la siguiente aproximación:

$$\begin{array}{c|c} a-h & f(a-h) \\ & \searrow \\ & \dfrac{f(a+h)-f(a-h)}{2h} \\ & \nearrow \\ a+h & f(a+h) \end{array}$$

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}$$

$$\Rightarrow \; P_1(x) = f(a-h) + \frac{f(a+h)-f(a-h)}{2h}(x-a+h)$$

$$for \; f'(a)$$

- La aproximación $[f(x+h)-f(x)]/h$ se suele llamar la aproximación por diferencia hacia adelante o *forward difference approximation*, $[f(x+h)-f(x-h)]/2h$ se suele llamar la aproximación por diferencia central o *central difference approximation*, y $[f(x)-f(x-h)]/h$ se suele llamar la aproximación por diferencia hacia atrás o *regressive difference approximation*

- La exactitud de la aproximación se determina por un cálculo muy simple involucrando el teorema de expansión de Taylor, que permite aislar el error de la derivada y obtenerlo en función de una derivada de mayor orden y de h

- El error de la aproximación para la derivada hacia adelante es el siguiente:

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(\xi_{x,h})$$

$$f'(x) - \frac{f(x+h) - f(x)}{h} = -\frac{1}{2}hf''(\xi_{x,h}) = O(h)$$

- El error de la aproximación para la derivada centrada es el siguiente:

$$\begin{cases} f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(\xi_1) \\ f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(\xi_2) \end{cases}$$

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{1}{6}h^3[f'''(\xi_1) + f'''(\xi_2)]$$

$$\Rightarrow f'(x) - \frac{f(x+h) - f(x-h)}{2h} = -\frac{1}{6}h^2 \frac{f'''(\xi_1) + f'''(\xi_2)}{2}$$

$$by\ the\ Discrete\ Average\ Value\ theorem:$$

$$\Rightarrow f'(x) - \frac{f(x+h) - f(x-h)}{2h} = -\frac{1}{6}h^2 f'''(\xi_{x,h}) = O(h^2)$$

- El error de la aproximación para la derivada hacia atrás es el siguiente:

$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(\xi_{x,h})$$

$$\Rightarrow \frac{f(x-h) - f(x)}{h} - f'(x) = \frac{1}{2}hf''(\xi_{x,h}) = O(h)$$

- Las aproximaciones cuestan más o menos lo mismo en términos computacionales: estas requieren dos evaluaciones de funciones y unas cuantas operaciones aritméticas adicionales. No obstante, los errores de estimaciones son muy diferentes

  - Debido a que la función es arbitraria, su coste puede ser grande, por lo que se ve esa como la parte más significativa del coste

- La primera y la última aproximación tiene un error que depende de f'' y es proporcional a h ($O(h)$), mientras que la segunda depende de f''' y es proporcional a h^2 ($O(h^2)$). Claramente, se querría usar una h pequeña para obtener la mejor exactitud, y por tanto la segunda estimación tenderá a ser mejor porque $h^2 < h$ tenderá a ser más importante que los diferentes tamaños de f'' y de f'''

h^{-1}	$D_1(h)$	$E_1(h) = f'(1) - D_1(h)$	$D_2(h)$	$E_2(h) = f'(1) - D_2(h)$
2	3.526814461	$-0.8085327148e+00$	2.832967758	$-0.1146860123e+00$
4	3.088244438	$-0.3699626923e+00$	2.746685505	$-0.2840375900e-01$
8	2.895481110	$-0.1771993637e+00$	2.725366592	$-0.7084846497e-02$
16	2.805027008	$-0.8674526215e-01$	2.720052719	$-0.1770973206e-02$
32	2.761199951	$-0.4291820526e-01$	2.718723297	$-0.4415512085e-03$
64	2.739639282	$-0.2135753632e-01$	2.718391418	$-0.1096725464e-03$
128	2.728942871	$-0.1066112518e-01$	2.718307495	$-0.2574920654e-04$

- Una pregunta que uno se hace al lidiar con estas aproximaciones es si es posible reducir el valor de h a placer para minimizar el valor de la derivada numérica

  - El error en las fórmulas de aproximación de la derivada depende del incremento h, de modo que pensar que minimizando h se puede minimizar el error es lógico

    - No obstante, cuando el paso h es muy pequeño, se puede producir un error numérico debido a la cancelación catastrófica que se produce cuando dos valores se parecen mucho y se restan

      $$f(x \pm h) \approx f(x) \quad for\ small\ h$$

    - Cuando se calculan números afectados por un cierto error de redondeo (como el caso en el que se redondean estas aproximaciones a la precisión de la máquina), nunca se mejorará la exactitud de los resultados debido a que los errores se incrementan por la propagación de errores

  - Anteriormente, los errores ignoraron el error de redondeo, lo cual es normal debido a que la mayoría de los estimadores de errores se harán de esta manera. No obstante, este error de redondeo puede entrometerse en los cálculos

    - Siendo $\tilde{f}(x)$ el cálculo de una función realmente hecha por la computadora (que contiene un error de redondeo) y definiendo $\epsilon(x) = f(x) - \tilde{f}(x)$ como el error entre la función calculado con precisión infinita y aquella realmente calculada por la máquina, el error será pequeño, pero no será nulo

- La derivada aproximada que se calcula se construye con $\tilde{f}(x)$, no con $f(x)$. Como ejemplo, se define $\widetilde{D}_2(h)$ de la siguiente manera:

$$\widetilde{D}_2(h) = \frac{\tilde{f}(x+h) - \tilde{f}(x-h)}{2h}$$

- Con tal de poder acotar el error $f'(x) - \widetilde{D}_2(h)$ (el error de la cantidad a calcular con la que actualmente se puede calcular), se puede realizar el siguiente desarrollo:

$$f'(x) - \widetilde{D}_2(h) = f'(x) - \frac{\tilde{f}(x+h) - \tilde{f}(x-h)}{2h}$$

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

$$+ \frac{f(x+h) - f(x-h)}{2h} - \frac{\tilde{f}(x+h) - \tilde{f}(x-h)}{2h} =$$

$$= -\frac{1}{6}h^2 f'''(\xi_{x,h}) + \frac{f(x+h) - f(x-h) - \tilde{f}(x+h) + \tilde{f}(x-h)}{2h} =$$

$$= -\frac{1}{6}h^2 f'''(\xi_{x,h}) + \frac{f(x+h) - \tilde{f}(x+h) - f(x-h) - \tilde{f}(x-h)}{2h} =$$

$$= -\frac{1}{6}h^2 f'''(\xi_{x,h}) + \frac{\epsilon(x+h) - \epsilon(x-h)}{2h}$$

- Este error se puede descomponer en dos partes: una parte del error que corresponde al error que hay al aproximar y otra parte del error debido al error de redondeo

$$f'(x) - \widetilde{D}_2(h) = -\frac{1}{6}h^2 f'''(\xi_{x,h}) + \frac{\epsilon(x+h) - \epsilon(x-h)}{2h}$$

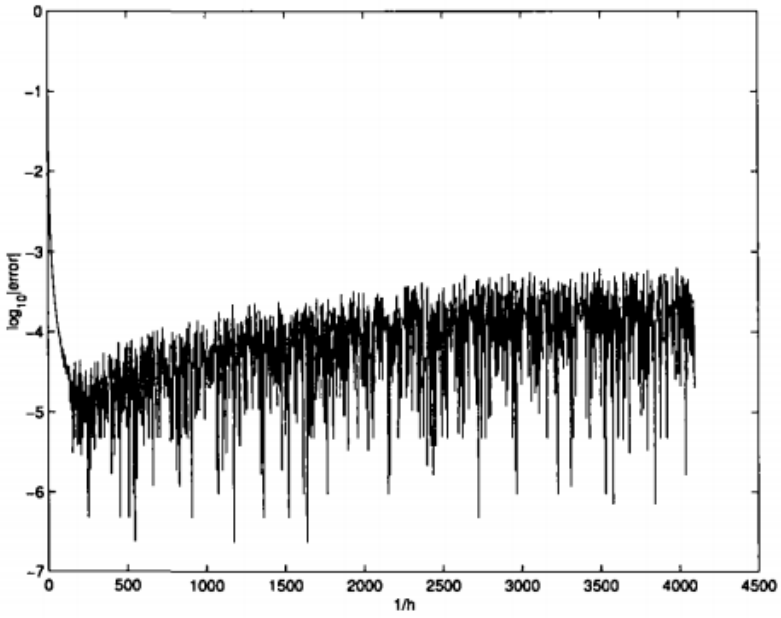
$$\begin{cases} Error\ due\ to\ approx.: & -\frac{1}{6}h^2 f'''(\xi_{x,h}) \\ Error\ due\ to\ rounding: & \frac{\epsilon(x+h) - \epsilon(x-h)}{2h} \end{cases}$$

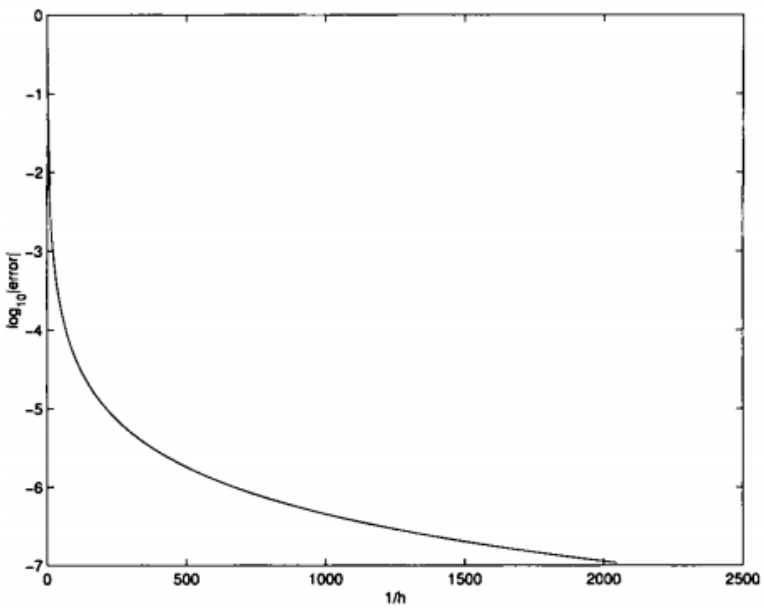
- Generalmente, el numerador de la fracción de la parte del error de redondeo no convergerá a cero cuando $h \to 0$, dado que siempre se tendrá una pequeña cuantía de error de redondeo presente. No obstante, como se está dividiendo por $2h$, se espera

que el término comience a crecer cuando h se hace pequeño (tiende a infinito), que se observa en la siguiente tabla:

h^{-1}	$D_2(h)$	$E_2(h) = f'(1) - D_2(h)$	$E_2(h)/E_2(h/2)$
2	2.832967758	$-0.1146860123e+00$	N/A
4	2.746685505	$-0.2840375900e-01$	4.038
8	2.725366592	$-0.7084846497e-02$	4.009
16	2.720052719	$-0.1770973206e-02$	4.001
32	2.718723297	$-0.4415512085e-03$	4.011
64	2.718391418	$-0.1096725464e-03$	4.026
128	2.718307495	$-0.2574920654e-04$	4.259
256	2.718292236	$-0.1049041748e-04$	2.455
512	2.718261719	$0.2002716064e-04$	-0.524

- Cuando hay error de redondeo, el error puede incrementar para valores incrementalmente pequeños de h y oscila salvajemente (por la cancelación catastrófica). En cambio, si no se tuviera este error de redondeo y se fuera más preciso, el error se reduciría para valores más pequeños de h y no habría oscilaciones salvajes





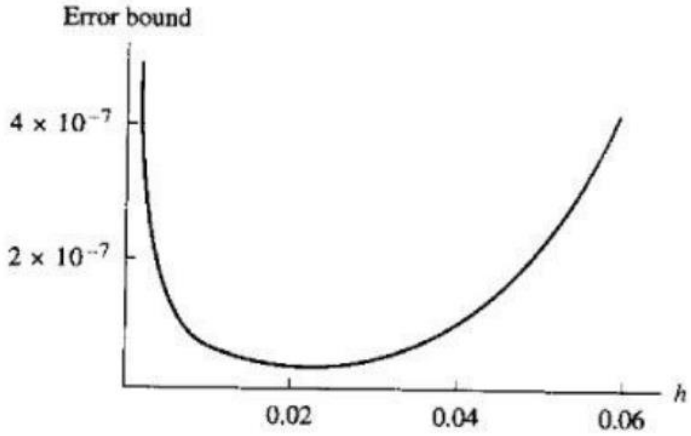
- Como se ha ejemplificado, el término del error de aproximación (teórico) si que debe desvanecerse cuando $h \to 0$, mientras que el término debido al error de redondeo tiende a ∞ cuando $h \to 0$. El valor óptimo de h se puede encontrar, entonces, minimizando el error total

  - En este caso, se utilizaría la expresión para el error y para el error de redondeo con tal de poder utilizar técnicas de optimización para encontrar la h^* que minimiza la expresión del error en valor absoluto (su cota superior)

$$f'(x) - \tilde{f}'(x) = approx.error + rounding\ error$$

$$\Rightarrow \left|f'(x) - \tilde{f}'(x)\right| \leq |approx.error + rounding\ error|$$

$$\Rightarrow \min_h |approx.error + rounding\ error|$$



- Con el ejemplo anterior y asumiendo que $x = 0$ y que $\epsilon(x+h) = 10^{-8}$ y $\epsilon(x-h) = -10^{-8}$, se obtiene el siguiente valor óptimo:

$$f'''(\xi_{x=0,h}) \approx 1 \Rightarrow \min_h \left|f'(x) - \widetilde{D}_2(h)\right| = \frac{1}{6}h^2 + \frac{2 \cdot 10^{-8}}{2h}$$

$$\frac{d}{dh}\left|f'(x) - \widetilde{D}_2(h)\right| = \frac{1}{3}h - \frac{10^{-8}}{h^2} = 0$$

$$\Rightarrow \quad h = (3 \cdot 10^{-8})^{1/3} \sim 0.003$$

- Combinaciones más complejas de $f(x \pm h)$ se puede usar para construir aproximaciones más precisas de f' y de derivadas de un orden mayor. Se puede utilizar la interpolación polinómica para derivar más fórmulas para aproximar la derivada, sobre todo útil para expresar el error de interpolación para derivadas

    - Esencialmente, como la teoría del error de interpolación muestra que los polinomios pueden aproximar más complicadas de manera razonablemente buena, se infiere que la derivada de una interpolación polinómica podría aproximar la derivada de la función razonablemente bien

        - Para poder aproximar una derivada de orden d con mayor precisión, se necesita construir un polinomio interpolador de grado $d \leq m - 1$ con más observaciones que su grado. Aumentando el número de observaciones, se puede obtener un polinomio interpolador más exacto y obtener mayor precisión en la interpolación

$$\{x_0 = a, \dots, x_m = a + mh\}, \{x_0 = a - mh, \dots, x_m = a\}$$

$$\left\{x_0 = a - \frac{mh}{2}, \dots, x_m = a + \frac{mh}{2}\right\}, \dots$$

        - En este caso, se plantea un polinomio interpolador de Newton con tal de poder obtener un polinomio p_m, de modo que, a partir de este polinomio, se puede derivar la expresión hasta el orden de la derivada d que se quiere aproximar. Por ejemplo, si se tiene $x_2 = a + 2h$, $x_1 = a + h$ y $x_0 = a$, se obtienen los siguientes resultados:

$$p_2(x) = p_1(x) + \frac{f(x_2) - p_1(x_2)}{(x_2 - x_1)(x_2 - x_0)}(x - x_1)(x - x_0)$$

$$where\ \ p_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0)$$

$$\Rightarrow p_2(x) = p_1(x) + \frac{f(a+2h) - p_1(a+2h)}{2h^2}(x - a - h)(x - a)$$

$$where\ \ p_1(x) = f(a) + \frac{f(a+h) - f(a)}{h}(x - a)$$

$$\Rightarrow\ p_2'(x) = p_1'(x) + \frac{f(a+2h) - 2f(a+h) + f(a)}{2h^2}(2(x - a) - h)$$

$$where\ \ p_1'(x) = \frac{f(a+h) - f(a)}{h}$$

$$\Rightarrow\ p_2'(a) = \frac{f(a+h) - f(a)}{h} - \frac{f(a+2h) - 2f(a+h) + f(a)}{2h} =$$

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

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

$$= \frac{-f(a+2h) + 4f(a+h) - 3f(a)}{2h}$$

$$\Rightarrow\ f'(x) \approx p_2'(x) = \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h}$$

- Si se buscan aproximaciones para derivadas de un orden superior a 1, es necesario utilizar un número de puntos mayor o igual al del máximo grado d del polinomio que se puede construir con m puntos (por lo tanto, $d \leq m - 1$). En este caso, se tiene que derivar el polinomio resultante hasta obtener una derivada de grado d deseada

- Debido a que se usa la derivada de un polinomio interpolador como una aproximación a la derivada, es posible utilizar el error de aproximación entre una función y su polinomio para encontrar el error de aproximación de la derivada y la derivada del polinomio

  - A partir de la ecuación para el teorema del error de interpolación polinómico, es posible obtener el error para la derivada a través de derivar directamente

$$f(x) - p_n(x) = \frac{1}{(n+1)!} w_n(x) f^{(n+1)}(\xi_x)$$

$$\Rightarrow\ f'(x) - p_n'(x) = \frac{1}{(n+1)!} \frac{d}{dx}\left[w_n(x) f^{(n+1)}(\xi_x)\right]$$

- Como no se sabe como ξ_x depende de x, entonces no se puede evaluar la derivada de $f^{(n+1)}(\xi_x)$. No obstante, si se evalúa en uno de los nodos x_i, entonces $w_n(x_i) = 0$ y se puede obtener una expresión cerrada:

$$\frac{d}{dx}\left[w_n(x) f^{(n+1)}(\xi_x)\right] = w_n'(x) f^{(n+1)}(\xi_x) + w_n(x) \frac{d}{dx}\left[f^{(n+1)}(\xi_x)\right]$$

$$for\ \ x = x_i:$$

$$\frac{d}{dx}\left[w_n(x_i) f^{(n+1)}(\xi_i)\right] = w_n'(x_i) f^{(n+1)}(\xi_i)$$

$$\Rightarrow\ f'(x_i) - p_n'(x_i) = \frac{w_n'(x_i)}{(n+1)!} f^{(n+1)}(\xi_i)$$

- En este caso, como el error depende de x_i, habrá una expresión diferente para cada i, y se tendrá una aproximación a la derivada en el punto x_i para cada valor de x_i y sus respectivos errores (obtenidos con la expresión derivada)

- No obstante, otra manera de poder obtener aproximaciones de la derivada y el error de la aproximación de la derivada es a través de la expansión polinómica de Taylor

  - Para poder construir estas aproximaciones, dados los puntos que se tienen y la condición de que no se pueden usar derivadas de ningún orden aparte del que se quiere aproximar, es posible usar el teorema de Taylor. Además, de este modo se puede encontrar directamente una expresión para el error de aproximación

  - Por ejemplo, si se tiene $x_2 = x + 2h$, $x_1 = x + h$ y $x_0 = x$, se obtienen los siguientes resultados:

$$\begin{cases} f(x+2h) = f(x) + 2hf'(x) + 2h^2 f''(x) + \frac{1}{6}h^3 f'''(\xi_{x,h}) \\ f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(\xi_{x,h}) \end{cases}$$

$$4f(x+h) - f(x+2h) = 3f(x) + 2hf'(x) + \frac{1}{3}h^3 f'''(\xi_{x,h})$$

$$4f(x+h) - f(x+2h) - 3f(x) = 2hf'(x) + \frac{1}{3}h^3 f'''(\xi_{x,h})$$

$$\frac{4f(x+h) - f(x+2h) - 3f(x)}{2h} = f'(x) + \frac{1}{6}h^2 f'''(\xi_{x,h})$$

$$by\ the\ Discrete\ Average\ Value\ theorem:$$

$$\Rightarrow\ f'(x) - \frac{4f(x+h) - f(x+2h) - 3f(x)}{2h} = -\frac{1}{6}h^2 f'''(\xi_{x,h})$$

- Una manera equivalente de hacer esto es plantear primero aproximaciones sencillas usando diferentes puntos $x+h, x+2h, x-3h, \ldots$ con tal de obtener una aproximación a una derivada concreta sencilla, y después sumar las expresiones de la derivada, de modo que en un lado de la ecuación se tiene un múltiplo de la derivada deseada y en la otra se tiene la expresión a multiplicar para obtener la aproximación final

$$f(x-h) = f(x) - f'(x)h + f''(x)\frac{h^2}{2!} - f'''(x)\frac{h^3}{3!} + O(h^4)$$

$$\Rightarrow\ f'(x) = \frac{f(x) - f(x-h)}{h} + O(h) \quad (regressive\ \ diff.)$$

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

$$\Rightarrow\ f'(x) = \frac{f(x+h) - f(x)}{h} + O(h) \quad (progressive\ \ diff.)$$

$$2f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{f(x+h) - f(x)}{h} - 2f'''(x)\frac{h^2}{3!} + O(h^4)$$

$$\Rightarrow\ f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \quad (centered\ \ diff.)$$

- Este método para encontrar aproximaciones realmente es equivalente, dado que la expresión del error de interpolación proviene del teorema de expansión polinómica de Taylor

- Con tal de evitar la cancelación de términos que aparecen en la derivación directa de las aproximaciones derivadas, se puede usar un algoritmo conocido como la extrapolación de Richardson

La integración numérica

- Para entrar en el tema de la integración numérica, es necesario revisar algunos conceptos de la integración definida

    - Para construir la integral definida de una función, se tienen que definir los puntos de corte o intermedios $x_i^{(n)}$ acorde a $a = x_0^{(n)} < x_1^{(n)} < \cdots < x_{n-1}^{(n)} < x_n^{(n)} = b$ y los puntos de evaluación η_i se toman para cada uno de los subintervalos ($\eta_i \in [x_{i-1}, x_i]$ para $1 \leq i \leq n$)

        - Si se asume que la distancia más grande entre puntos adyacentes tiende hacia cero cuantos más puntos se cogen, entonces, bajo condiciones no tan duras en f, se puede demostrar que el siguiente límite existe y que es independiente de la elección hecha para los puntos de corte $\left\{x_i^{(n)}\right\}$ y puntos de evaluación $\{\eta_i\}$:

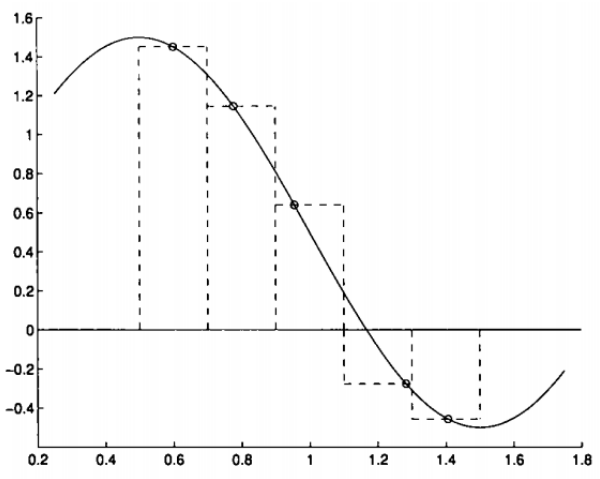
$$\lim_{n\to\infty}\left[\max_{1\leq i\leq n}\left(x_i^{(n)} - x_{i-1}^{(n)}\right)\right] = 0$$

$$\Rightarrow\ L = \lim_{n\to\infty}\sum_{i=1}^{n} f(\eta_i)\left(x_i^{(n)} - x_{i-1}^{(n)}\right)$$

        - Cuando esto ocurre, a este límite se le llama la integral definida de f y se denota como $L = I(f) = \int_a^b f(x)\,dx$

    - Para construir la integral se necesita muy poco: solo es necesario que la función f sea continua, que la secuencia $\left\{x_i^{(n)}\right\}$ sea tal que la distancia entre puntos adyacentes vaya disminuyendo y que los puntos de evaluación se tomen de cualquier parte en cada subintervalo

        - Una suma de la forma $\sum_{i=1}^{n} f(\eta_i)\left(x_i^{(n)} - x_{i-1}^{(n)}\right)$ se conoce como una suma riemanniana. Desafortunadamente, estas convergen demasiado lento, por lo que se requiere un valor de n grande para que la suma sea una buena aproximación del valor límite, por lo que no es práctica para aproximar la integral



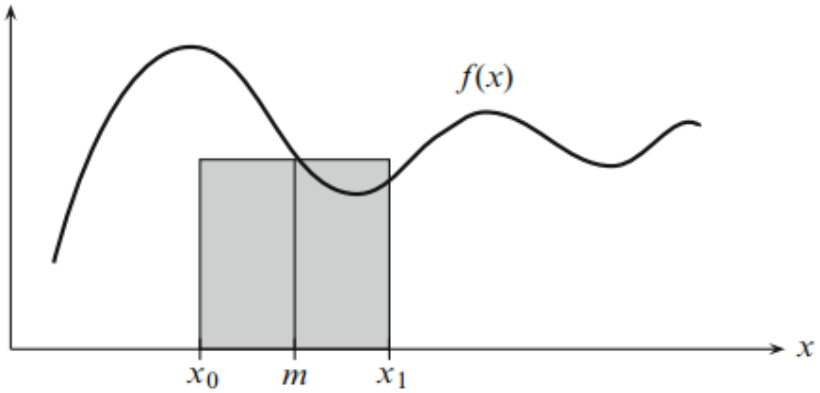
- Durante el desarrollo, normalmente se ignora el suscrito (n) (para las abscisas), de modo que se escribe x_i

- Como la integral es esencialmente el límite de la suma riemanniana (de valores de funciones), tiene sentido aproximar la integral definida tomando sumas finitas de valores de funciones, y los métodos de integración numérica se basan en esta idea

- A partir de la suma riemanniana se motiva una de las reglas de cuadratura más básicas, llamada la regla del punto medio. Asumiendo que $f(x)$ es continua $[a, b]$ y se divide el intervalo en n subintervalos de longitud $h_i = x_i^{(n)} - x_{i-1}^{(n)}$, la regla del punto medio es la siguiente:

$$M_n = \sum_{i=0}^{n} f(\eta_i) h_i \quad where \quad \eta_i = \frac{x_i^{(n)} + x_{i-1}^{(n)}}{2}$$

- En este caso, se puede ver como la aproximación para f es una suma riemanniana en donde η_i es el punto medio en el subintervalo $I_i = [x_{i-1}, x_i]$, en vez de cualquier punto $\eta_i \in [x_{i-1}, x_i]$



- Por lo tanto, en el límite, M_n tiende a la integral de Riemann cuando $n \to \infty$

$$\lim_{n\to\infty} M_n = \int_a^b f(x)\,dx$$

- La idea principal de la integración numérica es poder aproximarse a la integral definida a través de interpolaciones polinómicas de diferentes grados

  - Si se usa un polinomio de Lagrange con $n+1$ nodos, entonces la integral numérica de una función f se puede escribir de la siguiente forma:

$$\int_a^b f(x)\,dx = \int_a^b \sum_{i=0}^{n} f(x_i)L_i(x)\;dx + \int_a^b \frac{f^{(n+1)}(\xi_i)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)\;dx$$

$$= \sum_{i=0}^{n} w_i f(x_i) + \frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi_i)\prod_{i=0}^{n}(x-x_i)\;dx$$

$$where\;\; w_i = \int_a^b L_i(x)\,dx$$

  - En este caso, w_k son las ponderaciones de la fórmula de interpolación. Estas dependen solo del intervalo de integración $[a,b]$ y de los nodos x_i (nunca de la función f) y la fórmula resultante será exacta para cualquier polinomio de grado n o menor (siendo el polinomio interpolador único)

  - El error de cuadratura se puede estimar de la siguiente manera:

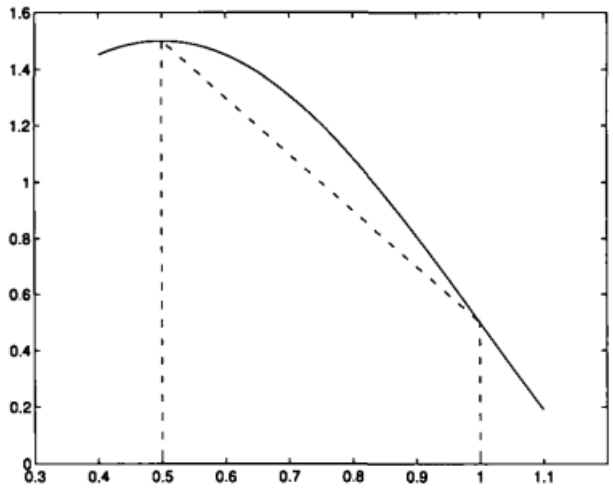
$$\int_a^b \big(f(x) - p_n(x)\big)\,dx = \int_a^b \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)\;dx$$

- Una de las aplicaciones más importantes de la interpolación lineal es la construcción de la regla del trapezoide o el método del trapezoide para aproximar integrales definidas

  - Se define la integración de interés como $I(f) = \int_a^b f(x)\,dx$, siendo $p_1(x) = \frac{x-a}{b-a}f(b) + \frac{b-x}{b-a}f(a)$, entonces la regla básica del trapezoide se define exactamente integrando $p_1(x)$:

$$T_1(f) = I(p_1) = \int_a^b p_1(x)\,dx = \frac{1}{2}(b-a)\big(f(b)+f(a)\big)$$

- Se construye una aproximación al reemplazar la función exacta f por una aproximación más simple para aproximarla p_1, y se realiza el cálculo deseado exactamente en esta función aproximadora



- Para saber que tan exacta es esta regla, se puede analizar el error porque la construcción de p_1 permite ver que se puede usar la teoría del error de la interpolación donde $\xi_x \in [a,b]$ depende de x:

$$I(f) - T_1(f) = I(f) - I(p_1) = \int_a^b f(x)\,dx - \int_a^b p_1(x)\,dx =$$

$$= \int_a^b f(x) - p_1(x)\,dx = \frac{1}{2}\int_a^b (x-a)(x-b)f''(\xi_x)\,dx$$

  - Debido a que $(x-a)(x-b)$ no cambia de signo en $[a,b]$, se puede aplicar el teorema del valor medio integral para obtener una estimación del error:

$$\int_a^b (x-a)(x-b)f''(\xi_x)\,dx = f''(\eta)\int_a^b (x-a)(x-b)\,dx =$$

$$= -\frac{1}{6}(b-a)^3 f''(\eta)$$

$$\Rightarrow\ I(f) - T_1(f) = -\frac{1}{12}(b-a)^3 f''(\eta)\ \ where\ \eta \in [a,b]$$

- Por lo tanto, siendo $f \in C^2([a,b])$ y p_1 interpolación f en a y b, y definiendo $T_1(f) = I(p_1)$, entonces existe una $\eta \in [a,b]$ tal que se cumple la siguiente igualdad:

$$I(f) - T_1(f) = -\frac{1}{12}(b-a)^3 f''(\eta)$$

- Aunque el resultado sea perfectamente válido, no parece que sea muy útil dado que el error solo será pequeño si el intervalo de integración $[a,b]$ es pequeño. Lo que se necesita es utilizar más puntos en la aproximación, por lo que se divide el intervalo $[a,b]$ en n subintervalos usando puntos intermedios x_i para $0 \leq i \leq n$:

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

  - Sobre cada subintervalo $[x_{i-1}, x_i]$ se aplica la regla básica del trapezoide para obtener una regla compuesta del trapezoide, más comúnmente referida como la regla del trapezoide de n subintervalos:

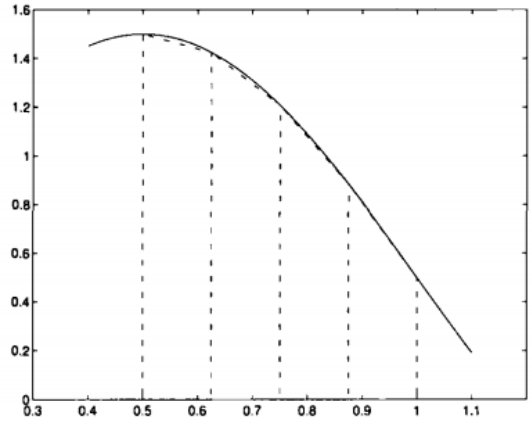
$$I(f) = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx \approx \sum_{i=1}^{n} \frac{1}{2}(x_i - x_{i-1})\big(f(x_{i-1}) + f(x_i)\big)$$

$$= T_n(f)$$

  - Si se usa una subdivisión uniforme, en donde los intervalos están equiespaciados de modo que $x_i - x_{i-1} = h$ para toda i, se simplifica a la siguiente igualdad:

$$T_n(f) = \frac{h}{2}\big(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\big)$$

  - Aún con pocos puntos, el error aparentemente parece sustancialmente reducido del caso anterior. Lo que se hace es integrar exactamente una interpolación lineal por trozos de f



- Siendo $f \in C^2([a,b])$ y $T_n(f)$ la regla del trapezoide de n subintervalos para $I(f)$, usando una segmentación uniforme, entonces existe $\xi_h \in [a,b]$ dependiendo de h tal que se cumple la siguiente igualdad:

$$I(f) - T_n(f) = -\frac{b-a}{12}h^2 f''(\xi_h)$$

  - Esto se demuestra a través del error para cada subintervalo y del teorema del valor promedio discreto:

$$I(f) - T_n(f) = \sum_{i=1}^{n}\left(\int_{x_{i-1}}^{x_i} f(x)\,dx - \frac{1}{2}(x_i - x_{i-1})\big(f(x_{i-1}) + f(x_i)\big)\right) =$$

$$= -\sum_{i=1}^{n}\frac{1}{12}h^3 f''(\xi_{i,h}) \quad for \quad \xi_{i,h} \in [x_{i-1}, x_i]$$

$$\Rightarrow -\frac{1}{12}h^3\sum_{i=1}^{n} f''(\xi_{i,h}) = -\frac{1}{12}h^2\frac{b-a}{n}\sum_{i=1}^{n} f''(\xi_{i,h}) =$$

$$= -\frac{b-a}{12}h^2\left(\frac{1}{n}\sum_{i=1}^{n} f''(\xi_{i,h})\right)$$

$by\ the\ Discrete\ Average\ Value\ theorem$:

$$\frac{1}{n}\sum_{i=1}^{n} f''(\xi_{i,h}) = f''(\xi_h) \quad for\ \xi_h \in [a,b]$$

$$\Rightarrow\ I(f) - T_n(f) = -\frac{b-a}{12}h^2 f''(\xi_h)$$

- Este teorema muestra que la aproximación numérica de convergerá al valor exacto siempre que $f \in C^2([a,b])$:

- Además, muestra qué tan rápido ocurre la convergencia, dado que el error se mueve como h^2. Una manera fácil de ver esto es entender que si el número de intervalos se duplica resulta en el error disminuyendo por un factor de 4 (el valor de ξ_h depende de h y se puede desplazar un poco si se cambia la longitud del intervalo)

- El valor de la estimación del error no permite predecir *a priori* el error, aunque a veces se pueda usar la teoría para estimarlo, sino que permite saber *a priori* qué tan rápido el error decrecerá en función de h

  - Esto es así porque en el término de error aparece la potencia de h, de modo que se puede ver qué tan rápido decrecería el error a partir de la notación $O(h^\alpha)$

- El mecanismo de duplicar el número de subintervalos y observar como el error se reduce en factores de 4 es un mecanismo que permite detectar errores en los algoritmos de la regla trapezoidal

n	$T_n(f)$	$I(f) - T_n(f)$	Error Ratio
2	1.753931092	$-0.356493E-01$	N/A
4	1.727221905	$-0.894008E-02$	3.9876
8	1.720518592	$-0.223676E-02$	3.9969
16	1.718841129	$-0.559300E-03$	3.9992
32	1.718421660	$-0.139832E-03$	3.9998
64	1.718316787	$-0.349584E-04$	4.0000
128	1.718290568	$-0.873962E-05$	4.0000
256	1.718284013	$-0.218491E-05$	4.0000
512	1.718282375	$-0.546227E-06$	4.0000
1024	1.718281965	$-0.136557E-06$	4.0000
2048	1.718281863	$-0.341392E-07$	4.0000

  - Si el programa no produce errores que se comportan de esta manera, entonces la función con la que se comprueba puede no ser $C^2([a,b])$ en todo el intervalo de integración, el código tiene un error o el ejemplo usado es de una clase especial de funciones (periódicas integradas por un múltiplo entero de su periodo) para la cual la regla del trapezoide es muy exacta, en donde el error comienza siendo muy pequeño y no varía de manera regular

- La suposición de que se hace una segmentación uniforme no es necesaria. Siendo $f \in C^2([a,b])$ y $T_n(f)$ la regla del trapezoide para n - subintervalos para $I(f)$ usando la una segmentación uniforme definida

por $a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$ con $h_i = x_i - x_{i-1}$ y $h = \max_i h_i$, entonces se cumple la siguiente desigualdad:

$$|I(f) - T_n(f)| = \frac{b-a}{12} h^2 \max_{\xi_h \in [a,b]} |f''(\xi_h)|$$

- La demostración de este teorema es similar a la anteriormente vista

- La implementación de la regla del trapezoide se da por el siguiente pseudo-código:

Algorithm 2.5 *Simple Trapezoid Rule (pseudo-code).*

```
input a, b, n; external f:
     sum = 0.5*(f(a) + f(b))
     h = (b - a)/n
     for i=1 to n-1 do
          x = a + i*h
          sum = sum + f(x)
     endfor
     trapezoid = h*sum
end code
```

- Cuando se implementa cualquier método de integración lo mejor es comprobarlo con una función la cual se conozca el valor de su derivada y verificar que las aproximaciones convergen al valor exacto al ritmo apropiado cuando h decrece

- Si el código funciona en un ejemplo, pero no en otros, el problema seguramente está en la implementación para los otros ejemplos (tales como equivocarse con el valor exacto)

- Es posible hacerlo mejor que con la regla del trapezoide utilizando un polinomio de mayor orden, y el método de la regla de Simpson no es más que el paso siguiente lógico a la regla del trapezoide

  - Siendo p_2 el polinomio cuadrático que interpola f en los tres puntos $x_0 = a$, $x_2 = b$, y $x_1 = c = (b+a)/2$ (el punto medio), la regla de Simpson se define de la siguiente manera:

$$S_2(f) = I(p_2) = \int_a^b \big(L_0(x)f(a) + L_1(x)f(c) + L_2(x)f(b)\big)\, dx$$

$$L_0(x) = \frac{(x-c)(x-b)}{(a-c)(a-b)}$$

$$L_1(x) = \frac{(x-a)(x-b)}{(c-a)(c-b)}$$

$$L_2(x) = \frac{(x-a)(x-c)}{(b-a)(b-c)}$$

- Por lo tanto, la regla de cuadratura viene dada de la siguiente manera:

$$S_2(f) = Af(a) + Cf(c) + Bf(b)$$

$$A = \int_a^b L_0(x)\,dx\,, C = \int_a^b L_1(x)\,dx \;\&\; B = \int_a^b L_2(x)\,dx$$

- Para calcular los parámetros, se asume que $h = c - a = b - c = (a+b)/2$, de modo que se obtiene el siguiente resultado:

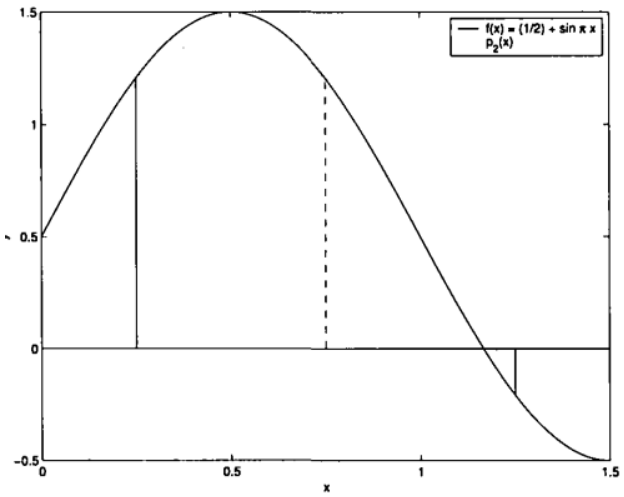
$$A = \int_a^{a+2h} L_0(x)\,dx = \frac{h}{3}$$

$$C = \int_a^{a+2h} L_1(x)\,dx = \frac{4h}{3}$$

$$B = \int_a^{a+2h} L_2(x)\,dx = \frac{h}{3} = A$$

- Por lo tanto, la regla de Simpson se convierte en la siguiente ecuación, en donde $h = (b-a)/2$ y por lo tanto es la distancia entre puntos en la discretización del intervalo $[a, b]$:

$$S_2(f) = \frac{h}{3}[f(a) + 4f(c) + f(b)] \quad where\ \ h = \frac{b-a}{2}$$



- La regla compuesta se construye de manera simple al sumar todas las instancias individuales de la regla de Simpson aplicada a pares de subintervalos (el método de Simpson requiere un número par de subintervalos)

  - Se obtiene la siguiente fórmula como generalización:

$$S_n(f) = \sum_{i=1}^{n/2} \frac{h_i}{3}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})]$$

$$where \quad h = \frac{x_{2i} - x_{2i-2}}{2}$$

  - Si se asume que el espaciado es uniforme en los puntos de corte, entonces $h_i = h$ para toda i y todo esto se simplifica a la siguiente fórmula:

$$S_n(f) = \frac{h}{3}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots +$$

$$+2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]$$

  - La regla de Simpson se puede construir sin asumir un espaciado uniforme, pero la regla resultante es mucho más complicada

- Si se registra el error de una aproximación con el método de Simpson, este decrece con un factor de 16 cuando se duplica el número de subintervalos, por lo que uno sospecharía que $I(f) - S_n(f) = O(h^4)$. No obstante, la teoría del error de interpolación muestra que el resultado debería ser del orden de $O(h^3)$

n	$S_n(f)$	$I(f) - S_n(f)$	Error Ratio
2	1.718861151877	–0.579323E–03	N/A
4	1.718318841922	–0.370135E–04	15.6517
8	1.718284154700	–0.232624E–05	15.9113
16	1.718281974052	–0.145593E–06	15.9777
32	1.718281837562	–0.910273E–08	15.9944
64	1.718281829028	–0.568969E–09	15.9986
128	1.718281828495	–0.355611E–10	15.9998
256	1.718281828461	–0.222178E–11	16.0057
512	1.718281828459	–0.137890E–12	16.1127
1024	1.718281828459	–0.910383E–14	15.1463
2048	1.718281828459	0.444089E–15	–20.5000

- Este error debería de ser del orden de $O(h^3)$ debido al siguiente resultado:

$$f - p_2 = O(h^3) \;\Rightarrow\; I(f) - S_n(f) = I(f) - I(p_2) =$$

$$= \int_a^b \big(f(x) - p_2(x)\big)\, dx = O(h^3)$$

- Por lo tanto, la regla de Simpson parece ser más precisa de lo que la teoría subyacente del error de interpolación predice. Aunque es posible derivar directamente el error de integración de la regla de Simpson (como con la del trapezoide), es más ilustrativo explicarlo desde el punto de vista de hasta qué punto este método integra polinomios de manera exacta, llevando a la noción de grado de precisión para una regla de cuadratura

- Siendo I_n una regla de cuadratura, y asumiendo que es exacta para polinomio de grado $\leq p$, entonces se dice que I_n tiene grado de precisión p

  - Se espera tener una mayor exactitud de métodos de integración que tengan un grado de precisión más alto

  - La regla del trapezoide tiene una precisión de $p = 1$ porque integra todos los polinomios lineales de manera exacta, mientras que se espera un grado de precisión de $p = 2$ para la regla de Simpson (porque integra los polinomios de segundo grado de manera exacta). No obstante, se puede demostrar que el método de Simpson tiene, en verdad, un grado de precisión $p = 3$, de modo que integra polinomios cúbicos de manera exacta (hay precisión extra inesperada de su construcción)

- Siendo q_3 un polinomio cúbico arbitrario de la forma $q_3 = Ax^3 + q_2(x)$, donde q_2 contiene todos los términos de menor grado, entonces el error al usar la regla de Simpson es el siguiente:

$$I(q_3) - S_2(q_3) = A\big(I(x^3) - S_2(x^3)\big) + \big(I(q_2) - S_2(q_2)\big)$$

  - La regla de Simpson claramente integrará la parte cuadrática de q_3 exactamente (porque nace de una integración exacta de un polinomio cuadrado), por lo que la regla será exacta para todos los polinomios cúbicos si, y solo si, es exacta para x^3. Un cálculo directo permite confirmar la hipótesis:

$$S_2(x^3) = \frac{(b-a)/2}{3}\Big(a^3 + 4\big((a+b)/2\big)^3 + b^3\Big) =$$

$$= \frac{b-a}{6}\left(a^3 + \frac{(a+b)^2}{2} + b^3\right) =$$

$$= \frac{b-a}{12}(2a^3 + (a+b)^2 + 2b^3) = \frac{1}{4}(b^4 - a^4) =$$

$$= \frac{1}{4}[x^4]_a^b = \int_a^b x^3\,dx = I(x^3)$$

  - Siendo $q_3(x)$ un polinomio arbitrario de grado menor o igual a 3, entonces la regla de Simpson integra q_3 exactamente para cualquier intervalo $[a, b]$

$$S_2(q_3) = \int_a^b q_3(x)\,dx$$

  - Asumiendo que $q_3 \approx f$, donde q_3 es una aproximación polinómica cúbica, y denotando el error por $R_3 = f - q_3$, entonces el error se puede expresar de la siguiente manera, debido a la linealidad de la integral y de la regla de Simpson:

$$I(f) - S_2(f) = I(q_3 + R_3) - S_2(q_3 + R_3)$$

$$\Rightarrow\ I(f) - S_2(f) = I(q_3) + I(R_3) - S_2(q_3) - S_2(R_3)$$

$$\Rightarrow\ I(f) - S_2(f) = I(q_3) + I(R_3) - I(q_3) - S_2(R_3)$$

$$\Rightarrow\ I(f) - S_2(f) = I(R_3) - S_2(R_3)$$

- Por lo tanto, la regla de Simpson es tan exacta para f como lo es para el error R_3 de cualquier aproximación polinómica cúbica p_3 para f. Debido a que se espera que la mejor aproximación cúbica sea mejor que la mejor aproximación cuadrática, el hecho de que se integren exactamente polinomios cúbicos hace que la regla tenga una mayor precisión

- Si $f \in C^4([a,b])$, entonces existe un punto $\xi \in [a,b]$ tal que se cumple la siguiente igualdad:

$$I(f) - S_2(f) = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)$$

  - Siendo p_3 un polinomio interpolador de f en los nodos $x_0 = a$, $x_1 = c - \epsilon$, $x_2 = c + \epsilon$ y $x_3 = b$, donde $\epsilon > 0$ es un parámetro pequeño, entonces para $x \in [a,b]$ se tiene el siguiente resultado para alguna $\xi_x \in [a,b]$:

$$f(x) - p_3(x) = R_3(x) = \frac{1}{4!}(x-x_0)(x-x_1)(x-x_2)(x-x_3)f^{(4)}(\xi_x)$$

  - Dado que la regla de Simpson integra polinomios exactamente, se tiene el siguiente resultado:

$$I(f) - S_2(f) = I(R_3) - S_2(R_3)$$

  - Como en $x = a$ y $x = b$ el residuo desaparece (se multiplica) y se obtiene la siguiente igualdad:

$$S_2(R_3) = \frac{b-a}{6}[f(a) + 4f(c) + f(b)] = 4\left(\frac{b-a}{6}\right)f(c)$$

$$= 4\left(\frac{b-a}{6}\right)\left(\frac{1}{4!}\right)(c-x_0)(c-x_1)(c-x_2)(c-x_3)f^{(4)}(\xi_c)$$

  - Debido a que x_1 y x_2 dependen del parámetro ϵ, entonces $x_1 \to c$ y $x_2 \to c$ cuando $\epsilon \to 0$, lo cual conlleva a que $R_3(c) \to 0$ cuando $\epsilon \to 0$ y entonces $S_2(R_3) \to 0$. Por otra parte, cuando $\epsilon \to 0$, se obtiene el siguiente resultado:

$$\lim_{\epsilon \to 0} I(R_3) = \frac{1}{4!}\int_a^b (x-a)(x-c)^2(x-b)f^{(4)}(\xi_x)\,dx$$

$$\Rightarrow I(f) - S_2(f) = \frac{1}{4!}\int_a^b (x-a)(x-c)^2(x-b) f^{(4)}(\xi_x)\, dx$$

- Debido a que $(x-a)(x-c)^2(x-b)$ no cambia de signo para $x \in [a,b]$, el teorema de valor medio integral puede usarse para simplificar la integral y obtener el resultado del teorema:

$$I(f) - S_2(f) = \frac{1}{4!} f^{(4)}(\xi_x) \int_a^b (x-a)(x-c)^2(x-b)\, dx$$

$$\Rightarrow I(f) - S_2(f) = -\frac{1}{90}\left(\frac{b-a}{2}\right)^5 f^{(4)}(\xi)$$

- Si $f \in C^4([a,b])$ y el espaciado de subintervalos es uniforme, entonces existe un punto $\xi \in [a,b]$ tal que se cumple la siguiente igualdad:

$$I(f) - S_2(f) = -\frac{(b-a)}{180} h^4 f^{(4)}(\xi) \quad for \quad h = \frac{b-a}{n}$$

- Como el espaciado es uniforme, entonces $h = (b-a)/n = x_i - x_{i-1}$ y permite obtener el siguiente resultado:

$$I(f) - S_n(f) =$$

$$= \sum_{i=1}^{n/2} \left(\int_{x_{2i-2}}^{x_{2i}} f(x)\, dx - \left(\frac{x_{2i} - x_{2i-2}}{6}\right)\left(f(x_{2i}) + 4f(x_{2i-1}) + f(x_{2i})\right) \right) =$$

$$= \sum_{i=1}^{n/2} \left(\int_{x_{2i-2}}^{x_{2i}} f(x)\, dx - \left(\frac{2(x_i - x_{i-1})}{6}\right)\left(f(x_{2i}) + 4f(x_{2i-1}) + f(x_{2i})\right) \right) =$$

$$= \sum_{i=1}^{n/2} \left(-\frac{1}{90}\left(\frac{2h}{2}\right)^5 f^{(4)}(\xi_i) \right) = -\frac{h^4}{90} \sum_{i=1}^{n/2} h\, f^{(4)}(\xi_i) =$$

$$= -\frac{h^4}{90} \sum_{i=1}^{n/2} \left(\frac{b-a}{n}\right) f^{(4)}(\xi_i) = -\frac{(b-a)h^4}{180} \sum_{i=1}^{n/2} \frac{2}{n} f^{(4)}(\xi_i)$$

- Por el teorema del valor medio discreto, es posible reemplazar la suma por una evaluación puntual de la cuarta derivada y obtener el siguiente resultado:

$$I(f) - S_n(f) = -\frac{(b-a)h^4}{180}\sum_{i=1}^{\frac{n}{2}}\frac{2}{n}f^{(4)}(\xi_i) = -\frac{(b-a)h^4}{180}f^{(4)}(\xi)$$

- Una de las ideas más importantes de la matemática computacional es que se puede tomar información de unas pocas aproximaciones y usar eso para estimar el error de aproximación y generar una mejor. A estos métodos de mejora se les conoce como métodos de extrapolación

    - Dada una integral $I(f)$ considerando una regla de cuadratura que se va a denotar por $I_n(f)$, se asume una relación del error de la forma de ley de potencias en función del número de subintervalos n:

    $$I(f) - I_n(f) \approx Cn^{-p}$$

        - Esta relación se mantiene para la regla del trapezoide ($p = 2$), la regla del punto medio ($p = 2$) y la regla de Simpson ($p = 4$), pero no la cuadratura gaussiana

        - Esta aproximación se quiere usar para estimar el valor de p, para construir una estimación del error calculable, y para construir una mejor aproximación para $I(f)$

    - La importancia de estimar p es que se puede usar para verificar que un programa computacional funciona correctamente, o para estimar tasas de convergencia cuando el integrante no es suave para aplicar la teoría del error desarrollada anteriormente

        - Aplicando la relación aproximada anterior a los casos n, $2n$ y $4n$, se considera una *ratio* la cual permite obtener un resultado en términos únicamente de p:

        $$\begin{cases} I(f) - I_n(f) = Cn^{-p} \\ I(f) - I_{2n}(f) = C(2n)^{-p} \\ I(f) - I_{4n}(f) = C(4n)^{-p} \end{cases} \quad \& \quad r_{4n} = \frac{I_n - I_{2n}}{I_{2n} - I_{4n}}$$

        $$\Rightarrow r_{4n} = \frac{I - Cn^{-p} + C(2n)^{-p} - I}{-C(2n)^{-p} + I + C(2n)^{-p} - I} =$$

        $$= \frac{C(2n)^{-p} - Cn^{-p}}{C(4n)^{-p} - C(2n)^{-p}} = \frac{2^{-p} - 1}{4^{-p} - 2^{-p}} = 2^p$$

- A partir de este resultado, es posible plantear un estimado para p:

$$2^p \Rightarrow \ln(2^p) = p\ln(2) \quad \Rightarrow \quad p \approx \frac{\ln(r_{4n})}{\ln(2)}$$

- Si la estimación de p no concuerda con lo esperado, entonces puede ser que la función no sea tan suave como se pensó previamente o que el programa no funcione correctamente. También puede darse el caso en que la función es de la clase especial de funciones para las cuales el método de cuadratura es muy exacto

n	$S_n(f)$	p
4	0.746855379791E+00	N/A
8	0.746826120527E+00	N/A
16	0.746824257436E+00	3.973123
32	0.746824140607E+00	3.995232
64	0.746824133300E+00	3.998911
128	0.746824132843E+00	3.999734
256	0.746824132814E+00	3.999953
512	0.746824132813E+00	3.999232
1024	0.746824132812E+00	4.002964
2048	0.746824132812E+00	4.063215

- Con tal de poder mejorar la aproximación y estimar el error, se utiliza un método similar al visto anteriormente:

  - A partir de una igualdad aproximada del error para I_{2n} con el error para I_n, es posible obtener una *ratio* conocida como el valor de extrapolación de Richardson

$$I - I_{2n} \approx C(2n)^{-p} = 2^{-p}(Cn^{-p}) \approx 2^{-p}(I - I_n)$$

$$\Rightarrow I - I_{2n} \approx 2^{-p}I - 2^{-p}I_n \quad \Rightarrow \quad I \approx \frac{2^p I_{2n} - I_n}{2^p - 1}$$

$$\Rightarrow \quad R_{2n} \approx \frac{2^p I_{2n} - I_n}{2^p - 1}$$

  - Por lo tanto, como se tiene un valor aproximado para la integral real I, es posible obtener una estimación calculable del error en I_{2n} como aproximación a $I = I(f)$ (el error de extrapolación)

$$E_{2n} = R_{2n} - I_{2n} = \frac{I_{2n} - I_n}{2^p - 1}$$

- También se puede ver al error de extrapolación E_{2n} como una estimación calculable en un cálculo, y entonces decidir cuando la aproximación es lo suficientemente exacta

n	$R_n(f)$	$\lvert I(f) - R_n(f)\rvert$	$\lvert I(f) - T_n(f)\rvert$	$\lvert E_n\rvert$
8	1.718272532150	0.929631E–05	0.223676E–02	0.224606E–02
16	1.718281246223	0.582236E–06	0.559300E–03	0.559882E–03
32	1.718281792050	0.364088E–07	0.139832E–03	0.139868E–03
64	1.718281826183	0.227585E–08	0.349584E–04	0.349607E–04
128	1.718281828317	0.142247E–09	0.873962E–05	0.873977E–05
256	1.718281828450	0.889089E–11	0.218491E–05	0.218492E–05
512	1.718281828458	0.553779E–12	0.546227E–06	0.546228E–06
1024	1.718281828459	0.364153E–13	0.136557E–06	0.136557E–06
2048	1.718281828459	0.266454E–14	0.341392E–07	0.341392E–07

Los métodos numéricos para ODEs

- El enfoque ahora es resolver problemas de ecuaciones diferenciales de manera numérica. De primeras, uno se concentra en el problema de valores iniciales o *initial value problems* (IVPs), y después se pasa a los problemas de valores de frontera o *boundary value problems* (BVPs), por lo que se comienza con los IVPs

    - Se considera la ecuación diferencial ordinaria $d\boldsymbol{y}/dt = \boldsymbol{f}\big(t, \boldsymbol{y}(t)\big)$, donde $\boldsymbol{y}(t_0) = \boldsymbol{y_0}$ y f es una función de $\mathbb{R}^{N+1}$ a $\mathbb{R}^N$ para alguna $N > 0$ (cuando $N = 1$, se obtiene una ecuación escalar, y si no, una ecuación vectorial)

        - Además, t_0 es un valor escalar dado (normalmente fijado a $t_0 = 0$) conocido como el punto inicial, y $\boldsymbol{y}_0 \in \mathbb{R}^N$ es un vector, conocido como el valor inicial

        - Se quiere encontrar la función desconocida $\boldsymbol{y}(t)$ que resuelve la ecuación diferencial, de modo que $\boldsymbol{y}'(t) - \boldsymbol{f}\big(t, \boldsymbol{y}(t)\big) = 0$ para toda $t > t_0$ y $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$

        - Para hacer el desarrollo más familiar, no obstante, se utilizará lenguaje y notación apropiada para ecuaciones escalares, pero se tiene que tener en cuenta que los resultados también se mantienen para ecuaciones vectoriales

    - La naturaleza de la solución de la ecuación diferencial (en verdad, la mera existencia de esta solución) y la habilidad para aproximarla con exactitud está muy conectada con la naturaleza de la función f

        - Básicamente, si f es lo suficientemente suave, entonces existe una solución única, y por tanto uno será capaz de aproximarla con

exactitud con varios métodos. No obstante, hay varias maneras de definir “suficientemente suave”

- Se suelen utilizar principalmente dos definiciones: la de función continua en el sentido de Lipschitz y la de función suave y uniforme y monótonamente decreciente. La primera definición es la condición normalmente utilizada para construir soluciones de IVPs, y es, generalmente, la condición más débil

- A continuación, se presentan las definiciones de continuidad anteriores y cómo se aplican estas a ecuaciones diferenciales

  - Siendo g una función de $\mathbb{R}$ a $\mathbb{R}$, se dice que la función g es continua en el sentido de Lipschitz en el intervalo I si existe una constante K tal que se cumpla la siguiente desigualdad para toda $x_1, x_2 \in I$:

$$|g(x_1) - g(x_2)| \leq K|x_1 - x_2|$$

  - En el caso de $f(t, y)$, se utilizan dos valores de y cualquiera y_1 e y_2 para calcular la diferencia $f(t, y_1) - f(t, y_2)$ y aplicar la desigualdad triangular. Uno se enfoca en y dado que se está analizando la continuidad de la misma función solución y

$$\frac{dy}{dt} = 4y + e^{-t} \Rightarrow f(t, y) = 4y + e^{-t}$$

$$\Rightarrow f(t, y_1) - f(t, y_2) = 4y_1 + e^{-t} - 4y_2 - e^{-t} = 4(y_1 - y_2)$$

$$\Rightarrow |f(t, y_1) - f(t, y_2)| \leq 4|y_1 - y_2|$$

  - Siendo g una función dada de $\mathbb{R}$ a $\mathbb{R}$, se dice que la función g es suave y uniformemente monótonamente decreciente si g es diferenciable y la derivada satisface la siguiente desigualdad para toda x, donde M y m son constantes positivas:

$$-M \leq \frac{d}{dx} g(x) \leq -m < 0$$

  - Es posible demostrar que la continuidad en el sentido de Lipschitz viene implícita por la suavidad y la uniformidad monótona decreciente a través del teorema del valor medio:

$$f(t, y_1) - f(t, y_2) = f(t, \xi)(y_1 - y_2)$$

$$\Rightarrow \ |f(t,y_1) - f(t,y_2)| \leq |f(t,\xi)||y_1 - y_2| \leq K|y_1 - y_2|$$

$$for \ f(t,\xi) = K$$

- En el caso de $f(t,y)$, se utiliza la derivada con respecto a y (dado que se está analizando la monotonía de la misma función solución y, no con respecto a t) y se intenta encontrar m y M, normalmente a través de ver las cotas para $\frac{d}{dy}f(t,y)$ usando el rango de t

$$\frac{dy}{dt} = -(1+t^2)y + \sin(t) \quad for \ t \in [0,1]$$

$$\Rightarrow f(t,y) = -(1+t^2)y + e^{-t}$$

$$\Rightarrow \frac{d}{dy}f(t,y) = -(1+t^2) \Rightarrow -2 \leq -t^2 - 1 \leq -1 < 0$$

- La relevancia de estas definiciones reside en cómo afectan a lo que uno sabe sobre las soluciones de las ecuaciones diferenciables

- Siendo $f(t,y)$ continua para todos los pares (t,y) en un rectángulo abierto $R = \{(t,y): a < t < b, c < y < d\}$ y continua en y en el sentido de Lipschitz, con constante K, entonces para todos los pares $(t_0, y_0) \in \mathbb{R}$ existe una única solución al IVP siguiente:

$$y' = f(t,y), \qquad y(t_0) = y_0$$

  - Además, si $z(t)$ es la solución para el mismo problema con los datos iniciales $z(t_0) = z_0$, entonces se cumple la siguiente desigualdad:

$$|y(t) - z(t)| \leq e^{K(t-t_0)}|y_0 - z_0|$$

- Siendo $f(t,y)$ continua para todos los pares (t,y) en un rectángulo abierto $R = \{(t,y): a < t < b, c < y < d\}$, suave y uniformemente y monótonamente decreciente en y, entonces para todos los pares $(t_0, y_0) \in \mathbb{R}$ existe una única solución al IVP siguiente:

$$y' = f(t,y), \qquad y(t_0) = y_0$$

- Además, si $z(t)$ es la solución para el mismo problema con los datos iniciales $z(t_0) = z_0$, entonces se cumple la siguiente desigualdad:

$$|y(t) - z(t)| \leq e^{-m(t-t_0)}|y_0 - z_0|$$

- Las desigualdades en ambos teoremas anteriores se conocen como los resultados de estabilidad, dado que miden que tan estables son las soluciones al IVP (qué tanto los pequeños cambios afectan conllevan pequeños cambios en la solución)

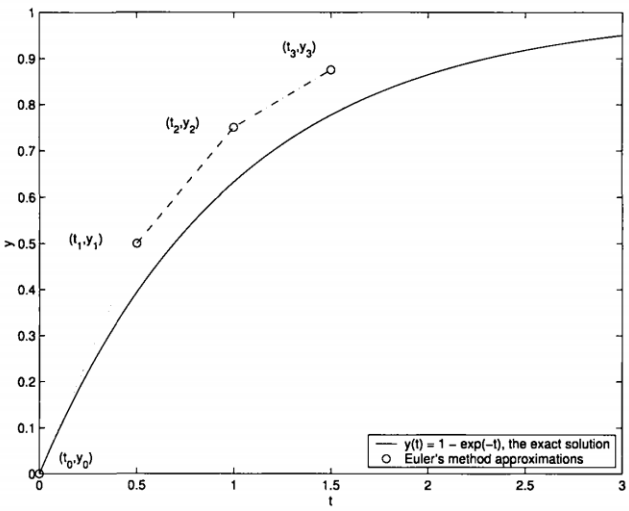
  - En el primer caso, si f es continua en el sentido de Lipschitz, entonces pequeños cambios en los datos iniciales pueden llevar a grandes cambios en las soluciones si se considera una t lo suficientemente grande. Aunque esto sea así, la condición expresa que las diferencias entre ambas soluciones y_1 e y_2 se puede acotar para toda $t \leq T$, donde T es un valor fijo y $z(t) \rightarrow y(t)$ en este intervalo, uniformemente, cuando $z_0 \rightarrow y_0$

  - En el segundo caso, si f es suave y uniformemente monótonamente decreciente, se sabe que la perturbación debido al error inicial en la solución decaerá a cero cuando $t \rightarrow \infty$. Esto tendrá implicaciones parecidas para el error asociado con los métodos numéricos aplicados a solucionar ecuaciones diferenciales

- El método de Euler es el punto de inicio natural para cualquier discusión de métodos numéricos para la solución de IVPs. Aunque no sea uno de los métodos más precisos, es el más simple, y mucho de lo que se desarrolla aquí se puede trasladar a otros métodos sin mucha dificultad

  - Existen dos derivaciones posibles para el método de Euler: una geométrica y la otra analítica. Lo natural es comenzar primero por la geométrica

    - El gráfico muestra la función solución $y(t)$ para el problema IVP $dy/dt = f(t, y)$ con $y(t_0) = y_0$ cerca del punto inicial t_0. Todo lo que se sabe es que la función pasa por el punto (t_0, y_0) y que tiene pendiente $f(t_0, y_0)$ en ese punto



- Por lo tanto, se puede aproximar $y(t_0 + h)$ para una h pequeña usando la aproximación de la línea tangente, la cual se puede extender para otros puntos $t_n = t_{n-1} + h$ y así derivar un método numérico de esta relación de recursividad:

$$y(t_0 + h) \approx y(t_0) + hf\big(t_0, y(t_0)\big)$$

$$\Rightarrow\ y(t_n + h) \approx y(t_n) + hf\big(t_n, y(t_n)\big)$$

$$\Rightarrow\ y_{n+1} \approx y_n + hf(t_n, y_n)$$

- Aunque esta derivación sea razonable, no admite mucho análisis del error y permite preguntarse aspectos como el de la validez de extender la línea tangente del primer punto calculado (t_1, y_1) al siguiente. Con el gráfico anterior, está claro que hay un error sustancial entre y_1 e $y(t_1)$, por lo cual este se extiende en los siguientes puntos

- La derivación analítica se basa en el teorema de aproximación de Taylor para una aproximación de primer grado

  - A través del teorema de Taylor, se hace una aproximación de primer grado para $f(t_0 + h)$ y, de la misma manera que antes, se obtiene una expresión recursiva definiendo $t_n = t_{n-1} + h$

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{1}{2}h^2 y''(\xi)$$

$$\Rightarrow\ y(t_0 + h) = y(t_0) + hf\big(t_0, y(t_0)\big) + \frac{1}{2}h^2 y''(\xi)$$

$$\Rightarrow\ y(t_n+h)=y(t_n)+hf\big(t_n,y(t_n)\big)+\frac{1}{2}h^2y''(\xi)$$

$$\Rightarrow\ y(t_n+h)\approx y(t_n)+hf\big(t_n,y(t_n)\big)$$

$$\Rightarrow\ y_{n+1}\approx y_n+hf(t_n,y_n)$$

- La ventaja de esta derivación es que relaciona la solución exacta con la aproximación numérica con un residuo preciso que permite obtener el error de aproximación del método de Euler. En este se pueden definir dos cantidades importantes: el residuo del método de Euler $R(t,h)$ y el error de truncación $\tau(t,h)$

$$R(t,h)=\frac{1}{2}h^2y''(\xi)\qquad \tau(t,h)=\frac{1}{h}R(t,h)$$

- Siendo f continua en el sentido de Lipschitz con constante K, y asumiendo que la solución $y\in C^2([t_0,T])$ para alguna $T>t_0$, entonces se cumple la siguiente desigualdad:

$$\max_{t_k\le T}|y(t_k)-y_k|\le C_0|y(t_0)-y_0|+Ch\|y''\|_{\infty,[t_0,T]}\quad where$$

$$C_0=e^{K(T-t_0)}\quad \&\quad C=\frac{C_0-1}{2K}$$

- La demostración de este teorema se basa en el hecho que la solución exacta satisface la misma relación que la aproximada excepto por el residuo. Por lo tanto, se pueden obtener los siguientes resultados:

$$\begin{cases}y(t_{n+1})=y(t_n)+hf\big(t_n,y(t_n)\big)+\frac{1}{2}h^2y''(\xi_n)\\ y_{n+1}=y_n+hf(t_n,y_n)\end{cases}$$

$$\Rightarrow\ y(t_{n+1})-y_{n+1}=$$

$$=y(t_n)-y_n+hf\big(t_n,y(t_n)\big)-hf(t_n,y_n)+\frac{1}{2}h^2y''(\xi_n)$$

$$\Rightarrow |y(t_{n+1})-y_{n+1}|\le$$

$$|y(t_n)-y_n|+h\big|f\big(t_n,y(t_n)\big)-f(t_n,y_n)\big|+\frac{1}{2}h^2y''(\xi_n)$$

$$\Rightarrow |y(t_{n+1}) - y_{n+1}| \leq$$

$$|y(t_n) - y_n| + hK|y(t_n) - y_n| + \frac{1}{2}h^2|y''(\xi_n)|$$

$$\Rightarrow e_{n+1} \leq \gamma e_n + R_n \quad where$$

$$e_n = |y(t_n) - y_n| \;\; , \;\; \gamma = 1 + hK \;\; \& \;\; R_n = \frac{1}{2}h^2|y''(\xi_n)|$$

- A partir de esta expresión, se puede resolver una expresión recursiva que permitirá obtener los siguientes resultados:

$$e_1 \leq \gamma e_0 + R_0$$

$$\Rightarrow \; e_2 \leq \gamma e_1 + R_1 \leq \gamma^2 e_0 + \gamma R_0 + R_1$$

$$\Rightarrow \; e_3 \leq \gamma e_2 + R_2 \leq \gamma^3 e_0 + \gamma^2 R_0 + \gamma R_1 + R_2$$

$$\Rightarrow \; \dots \; \Rightarrow \; e_n \leq \gamma^n e_0 + \frac{1}{2}h^2 \sum_{k=0}^{n-1} \gamma^k |y''(\xi_{n-1-k})|$$

- Con esta relación recursiva, lo único que hace falta es simplificar el término de la suma y obtener la desigualdad final:

$$|y''(t)| \leq \|y''\|_{\infty,[t_0,T]} \quad for \;\; all \;\; t \in [t_0, T]$$

$$\Rightarrow e_n \leq \gamma^n e_0 + \left(\frac{1}{2}h^2\|y''\|_{\infty,[t_0,T]}\right)\left(\sum_{k=0}^{n-1}\gamma^k\right)$$

$$\Rightarrow \; e_n \leq \gamma^n e_0 + \left(\frac{1}{2}h^2\|y''\|_{\infty,[t_0,T]}\right)\frac{\gamma^n - 1}{\gamma - 1}$$

$$\Rightarrow \; e_n \leq \gamma^n e_0 + \left(\frac{1}{2}h^2\|y''\|_{\infty,[t_0,T]}\right)\frac{\gamma^n - 1}{hK}$$

$$\Rightarrow \; e_n \leq \gamma^n e_0 + \frac{\gamma^n - 1}{2K}h\|y''\|_{\infty,[t_0,T]}$$

$$as \;\; \gamma^n = (1 + Kh)^n \leq e^{Knh} = e^{K(t_n - t_0)} \leq e^{K(T - t_0)}$$

$$\Rightarrow \; e_n \leq \gamma^n e_0 + \frac{e^{K(T-t_0)} - 1}{2K}h\|y''\|_{\infty,[t_0,T]}$$

- Esta estimación muestra que el método de Euler es de primer orden (el error es $O(h)$), pero también muestra que las constantes multiplicando los términos de error pueden volverse muy grandes. Si se asumiera que f es suave y uniformemente monótonamente decreciente, entonces se obtiene un estimador que involucra constantes menores

- Siendo f continua en el sentido de Lipschitz con constante K , y asumiendo que la solución $y \in C^2([t_0, T])$ para alguna $T > t_0$, entonces se cumple la siguiente desigualdad:

$$\max_{t_k \leq T} |y(t_k) - y_k| \leq C_0 |y(t_0) - y_0| + Ch \|y''\|_{\infty,[t_0,T]}$$

$$where \;\; C_0 \leq 1, C_0 \to 0 \;\; as \;\; k \to \infty \;\; \& \;\; C = \frac{1}{2m}$$

- Ambos teoremas de error muestran que el método de Euler es solo de exactitud de primer orden (por lo que $O(h)$). La diferencia entre ambos teoremas está en como las estimaciones dependen de f

  - Si f solo es continua en el sentido de Lipschitz, entonces las constantes pueden ser demasiado grandes y crecer rápidamente, pero si es suave y uniformemente monótonamente decreciente, entonces las constantes están acotadas para toda n

  - Si f es uniformemente monótonamente decreciente en y , entonces el error inicial se multiplica por γ^n, donde $0 \leq \gamma < 1$. Por lo tanto, el efecto de cualquier error inicial decrece rápidamente mientras la computación continua

- El método de Euler no es el único ni el mejor esquema para aproximar soluciones a problemas de valor inicial por lo que se necesita ver otros métodos. Muchas ideas se pueden considerar basadas en simples extensiones de una derivación del método de Euler

  - La tercera derivación del método de Euler, que también da el término residual, se basa en los métodos de diferencia para la aproximación de la derivada. Comenzando con la ecuación $y'(t) = f\big(t, y(t)\big)$ y reemplaza la derivada con el cociente diferencial simple, lo cual resulta en la siguiente ecuación:

$$\frac{y(t+h) - y(t)}{h} = f\big(t, y(t)\big) + \frac{1}{2} h y''\big(\theta_{t,h}\big)$$

- Los suscritos de $\theta_{t,h}$ recuerdan que el valor depende de t y h. El método de Euler entonces se obtiene simplemente sacando el residuo, reemplazando t con t_n y $y(t)$ con y_n y así

$$y(t+h) = y(t) + hf\big(t, y(t)\big) + \frac{h^2}{2} y''\big(\theta_{t,h}\big)$$

$$\Rightarrow\ y(t_{n+1}) = y(t_n) + hf\big(t_n, y(t_n)\big) + \frac{h^2}{2} y''\big(\theta_{t,h}\big)$$

$$\Rightarrow\ y_{n+1} = y_n + hf(t_n, y_n)$$

- Si se utilizan otras aproximaciones de la derivada, entonces se pueden obtener otros métodos relacionados con el método de Euler, pero no iguales

  - Si se usa la derivada hacia atrás, entonces se obtiene el método de Euler hacia atrás:

$$y'(t) = \frac{y(t) - y(t-h)}{h} - \frac{1}{2} h y''(\theta)$$

$$\Rightarrow\ \frac{y(t) - y(t-h)}{h} - \frac{1}{2} h y''(\theta) = f\big(t, y(t)\big)$$

$$\Rightarrow\ y(t) = y(t-h) + hf\big(t, y(t)\big) + \frac{1}{2} h^2 y''(\theta)$$

$$\Rightarrow\ y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

  - Si se usa la derivada del punto medio, entonces se obtiene el método del punto medio:

$$y'(t) = \frac{y(t+h) - y(t-h)}{2h} - \frac{1}{6} h^2 y'''(\theta)$$

$$\Rightarrow\ \frac{y(t+h) - y(t-h)}{2h} - \frac{1}{6} h^2 y'''(\theta) = f\big(t, y(t)\big)$$

$$\Rightarrow\ y(t+h) = y(t-h) + 2hf\big(t, y(t)\big) + \frac{1}{3} h^3 y'''(\theta)$$

$$\Rightarrow\ y_{n+1} = y_{n-1} + 2hf(t_n, y_n)$$

- Si se usa la aproximación de la derivada basada en la interpolación de Lagrange (derivándola), entonces se pueden obtener otros métodos numéricos:

$$\begin{cases} y'(t) \approx \frac{1}{2h}\big(-y(t+2h) + 4y(t+h) - 3y(t)\big) \\ y'(t+2h) \approx \frac{1}{2h}\big(3y(t+2h) - 4y(t+h) + y(t)\big) \end{cases}$$

$$\Rightarrow \begin{cases} y_{n+1} = 4y_n - 3y_{n-1} - 2hf(t_{n-1}, y_{n-1}) \\ y_{n+1} = \frac{4}{3}y_n - \frac{1}{3}y_{n-1} - \frac{2}{3}hf(t_{n+1}, y_{n+1}) \end{cases}$$

- Como se puede ver, los métodos serían exactos del orden $O(h)$ excepto el basado en la interpolación, el cual en el caso mostrado sería de $O(h^2)$ y serían más exactos que el método de Euler

- Otra manera de mejorar la exactitud de los métodos es a través del uso de más términos de la expansión de Taylor

  - Haciendo la expansión completa de Taylor para la solución en $y(t+h)$, se obtendría el siguiente resultado:

$$y(t+h) = y(t) + hf\big(t, y(t)\big) + \frac{h^2}{2}y''(t) + \cdots + \frac{h^n}{n!}y^{(n)}(t) + \frac{h^{(n+1)}}{(n+1)!}y^{(n+1)}(\xi)$$

$$\Rightarrow y(t_{n+1}) = y(t_n) + hf\big(t_n, y(t_n)\big) + \frac{h^2}{2}y''(t_n) + \cdots + \frac{h^n}{n!}y^{(n)}(t_n) + \frac{h^{(n+1)}}{(n+1)!}y^{(n+1)}(\xi_n)$$

  - Debido a que $y'(t) = f\big(t, y(t)\big)$, entonces $y''(t) = f'\big(t, y(t)\big)$, $y'''(t) = f''\big(t, y(t)\big)$ y así, permitiendo obtener el siguiente método de aproximación:

$$y(t_{n+1}) = y(t_n) + hf\big(t_n, y(t_n)\big) + \frac{h^2}{2}f'\big(t_n, y(t_n)\big) + \cdots + \frac{h^n}{n!}f^{(n-1)}\big(t_n, y(t_n)\big) + \frac{h^{(n+1)}}{(n+1)!}y^{(n+1)}(\xi_n)$$

$$\Rightarrow y(t_{n+1}) = y(t_n) + hf\big(t_n, y(t_n)\big) + \frac{h^2}{2}f'\big(t_n, y(t_n)\big) + \cdots$$

$$+\frac{h^n}{n!}f^{(n-1)}\big(t_n, y(t_n)\big) + \frac{h^{(n+1)}}{(n+1)!}y^{(n+1)}(\xi_n)$$

- Por lo tanto, si uno decide eliminar el término final del residuo, entonces se obtiene una exactitud del orden de $O(h^{n-1})$, siendo mucho más exactos (potencialmente) que el método de Euler

$$y_{n+1} = y_n + hf(t_n, y_n) + \frac{h^2}{2}f'(t_n, y_n) + \cdots + \frac{h^n}{n!}f^{(n-1)}(t_n, y_n)$$

- Finalmente, uno puede obtener otro conjunto de métodos se pueden derivar integrando la ecuación diferencial

  - La solución exacta a una ecuación siempre satisface la siguiente igualdad, por lo que se pueden aplicar varias aproximaciones numéricas de la integral con tal de obtener un método numérico:

$$y(t+h) = y(t) + \int_t^{t+h} f\big(s, y(s)\big)\, ds$$

  - Se puede aplicar la regla del trapezoide para obtener la siguiente igualdad, en donde $\theta_{t,h} \in [t, t+h]$:

$$y(t+h) = y(t) + \frac{h}{2}\big[f\big(t+h, y(t+h)\big) + f\big(t, y(t)\big)\big] - \frac{h^3}{12}y'''\big(\theta_{t,h}\big)$$

$$as \quad y'(t) = f\big(t, y(t)\big) \Rightarrow \frac{d^2}{dt^2}f\big(t, y(t)\big) = y'''(t)$$

$$\Rightarrow y_{n+1} = y_n + \frac{h}{2}[f(t_{n+1}, y_{n+1}) + f(t_n, y_n)]$$

  - Estos métodos tienen una exactitud del orden $O(h^2)$, por lo que también serán métodos más exactos que el de Euler

- A partir de los métodos numéricos derivados de extensiones del de Euler, es posible fijarse en los siguientes puntos:

  - En algunos métodos se han utilizado fórmulas para $y(t_{n+1})$ en términos de y_n y y_{n-1}. Estos métodos no son métodos de un solo paso o *single-step methods* como el de Euler, sino que son métodos de múltiples pasos o *multistep methods*, ya que dependen de la información de más de un valor aproximado de la función solución desconocida

- En el caso de los métodos de múltiples pasos, es necesario que se de más de un solo punto inicial para su implementación, dado que la ecuación diferencial solo da un solo valor inicial

- En métodos en donde se involucra el término $f(t_{n+1}, y_{n+1})$, uno no puede resolver de manera explícita para el valor aproximado de y_{n+1}, por lo que estos métodos se denominan métodos implícitos. Métodos como el de Euler, el del punto medio u otros se clasifican, en cambio, como métodos explícitos, dado que definen y_{n+1} explícitamente en términos de la información de pasos previos

- Uno está interesado en la exactitud de los métodos que se han desarrollado, por lo que es necesario tener claros conceptos como el error residual, el error de truncamiento y la consistencia

    - Todos los métodos numéricos para IVPs que se van a estudiar pueden escribirse de la siguiente manera general:

$$y_{n+1} = \sum_{k=0}^{p} a_k y_{n-k} + hF\left(y_{n+1}, y_n, \dots, y_{n-p}; f_{n+1}, f_n, \dots, f_{n-p}\right)$$

$$where \quad f_k = f(t_k, y_k)$$

        - Por ejemplo, para el método de Euler se tendría la siguiente forma:

$$y_{n+1} = \sum_{k=0}^{p} a_k y_{n-k} + hF\left(y_{n+1}, y_n, \dots, y_{n-p}; f_{n+1}, f_n, \dots, f_{n-p}\right)$$

$$where \begin{cases} a_0 = 1 \\ a_k = 0 \quad for \ k \geq 1 \\ F(y_{n+1}, \dots; f_{n+1}, \dots) = f_n \end{cases}$$

        - En cambio, para el método del trapezoide, se obtiene la siguiente forma:

$$y_{n+1} = \sum_{k=0}^{p} a_k y_{n-k} + hF\left(y_{n+1}, y_n, \dots, y_{n-p}; f_{n+1}, f_n, \dots, f_{n-p}\right)$$

$$where \begin{cases} a_0 = 1 \\ a_k = 0 \ for \ k \geq 1 \\ F(y_{n+1}, \dots; f_{n+1}, \dots) = \frac{1}{2}(f_n + f_{n+1}) \end{cases}$$

- Cómo la solución exacta $y(t)$ satisface el método numérico es crucial para la exactitud de estos métodos. Por lo tanto, se definen los conceptos importantes

- Para un método numérico escrito de la forma general, el residuo R_n y el error de truncamiento τ_n se define de la siguiente manera:

$$R_n = y(t_{n+1}) - \sum_{k=0}^{p} a_k y(t_{n-k}) + hF\big(y(t_{n+1}), \dots; f_{n+1}\big(t_n, y(t_n)\big), \dots\big)$$

$$\tau_n = \frac{1}{h} R_n$$

  - Generalmente, el residuo y el error de truncamiento se obtiene expandiendo $y(t_{n+1})$ en una serie de Taylor alrededor de t_n y hacer coincidir términos para que se eliminen, o aplicando un método de aproximación de la derivada o la integral

  - El método se dice que es consistente si se cumple la siguiente condición para las soluciones y lo suficientemente suave:

$$\lim_{h \to 0} \max_{t_n \leq T} |\tau_n| = 0$$

  - Para cualquier método razonable, el error $\max_{t_n \leq T} |y(t_k) - y_k|$ es proporcional al error de truncamiento si la solución es lo suficientemente suave. Nótese que el residuo y el error de truncamiento se han definido en términos de sustitución la solución exacta en el método numérico

  - Para la mayoría de métodos que se desarrollan, el residuo y el error de truncamiento aparecen naturalmente en los procesos de construcción. Para el método de Euler, por ejemplo, es consistente debido a la siguiente igualdad:

$$R_n = \frac{1}{2} h^2 y''(\theta_n) \quad \& \quad \tau_n = \frac{1}{2} h y''(\theta_n)$$

$$\Rightarrow \lim_{h \to 0} \max_{t_n \leq T} \frac{1}{2} h |y''(\theta_n)| = \left(\lim_{h \to 0} \frac{1}{2} h\right)\left(\max_{t \leq T} |y''(t)|\right) =$$

$$= \left(\lim_{h \to 0} \frac{1}{2} h\right)(t_{max}) = 0$$

- Si el error de truncamiento para un esquema numérico para la solución del IVP es $O(h^k)$, entonces se dice que el método tiene el orden de exactitud de k

  - El método del trapezoide tiene el orden de exactitud de $p = 2$, y el método de Euler tiene un orden de precisión de $p = 1$. Es común decir que la regla del trapezoide es exacta de segundo orden o *second-order accurate*, mientras que el método de Euler sería exacto de primer orden o *first-order accurate*

  - Es muy importante ver que la definición del orden de exactitud se basa en el error de truncamiento definido, no en el residuo. Por lo tanto, aunque el residuo sea del orden de $O(h^2)$ para el método de Euler, su error de truncamiento es del orden $O(h)$ y es exacto de primer orden

  - Esto solo se aplica para métodos numéricos en relación a IVP, y la exactitud de métodos en otras áreas se suele decir dependiendo del orden del residuo o error

- Debido a que dependen de más de un solo valor anterior, los métodos de múltiples pasos requieren valores iniciales. En esencia, se tiene que usar algún método para calcular aproximaciones y_1, y_2, …, y_p antes de que se use un método de múltiples pasos

  - Para muchos métodos presentados hasta ahora, solo era necesario un valor único y_1, pero para otros es necesario más valores iniciales adicionales

    - Una razón para el desarrollo de los métodos de predictor-corrector es precisamente para dar métodos de un solo paso que se pueden usar para producir valores iniciales precisos para los métodos de múltiples pasos

  - Cuando se quiere utilizar un método de múltiples pasos, se pueden usar diferentes métodos para estimar $y(t_1)$, que es necesario para proceder con el cálculo, y comparar los errores

n	y_1 computed by predictor–corrector		y_1 computed by Euler's method	
	y_n	Error	y_n	Error
1	0.521438685E+00	0.629350068E–05	0.521660849E+00	–0.215870648E–03
2	0.542442735E+00	–0.149082970E–04	0.542433042E+00	–0.521501377E–05
3	0.562913366E+00	–0.543392852E–05	0.563136000E+00	–0.228068587E–03
4	0.582876067E+00	–0.249892345E–04	0.582854530E+00	–0.345267290E–05
5	0.602241522E+00	–0.124030017E–04	0.602465395E+00	–0.236276553E–03
6	0.621050404E+00	–0.307978504E–04	0.621015069E+00	0.453741300E–05
7	0.639220651E+00	–0.152631109E–04	0.639446837E+00	–0.241449481E–03
8	0.656807255E+00	–0.330637366E–04	0.656756294E+00	0.178973669E–04
9	0.673732972E+00	–0.147732929E–04	0.673962850E+00	–0.244651773E–03
10	0.690066203E+00	–0.325779206E–04	0.689997851E+00	0.357749024E–04
11	0.705732009E+00	–0.117057153E–04	0.705967262E+00	–0.246958289E–03
12	0.720811392E+00	–0.301122659E–04	0.720723877E+00	0.574032598E–04
13	0.735229212E+00	–0.678027590E–05	0.735471822E+00	–0.249390296E–03
14	0.749078472E+00	–0.263680981E–04	0.748969952E+00	0.821511474E–04
15	0.762281388E+00	–0.625682149E–06	0.762533643E+00	–0.252880568E–03
16	0.774942633E+00	–0.219485394E–04	0.774811136E+00	0.109548790E–03

- Una pregunta que causa el uso de diferentes métodos para generar valores iniciales es si el uso de un método de menor orden para los valores iniciales afectará la exactitud del cómputo total. La explicación es que se puede demostrar que un método de orden $p-1$ puede ser usado para generar valores iniciales para un método de orden p sin usualmente afectar el orden de convergencia

- Por supuesto, sería mejor usar un método de un orden igual al que se usa para los valores iniciales, pero el orden neto de exactitud del método no debería verse afectado

- La familia de métodos de Runge-Kutta es una de las familias más populares de métodos de solución para IVPs. La derivación general puede volverse mucho más compleja, por lo que solo se desarrolla para el caso de segundo orden

  - Recordando la formulación usual del predictor-corrector del método del trapezoide, es posible realizar una sustitución para escribir una sola relación recursiva:

$$\begin{cases} \bar{y}_{n+1} = y_n + hf(t_n, y_n) \\ y_{n+1} = y_n + \frac{1}{2}h[f(t_{n+1}, \bar{y}_{n+1}) + f(t_n, y_n)] \end{cases}$$

$$\Rightarrow\ y_{n+1} = y_n + \frac{1}{2}h\big[f\big(t_{n+1}, y_n + hf(t_n, y_n)\big) + f(t_n, y_n)\big]$$

  - Como la ecuación diferencial implica que f da valores de y', se puede ver la relación de recursión como definiendo y_{n+1} desde y_n al avanzar a lo largo de la línea recta definida por una simple

media de dos pendientes $f(t_n, y_n)$ y $f(t_{n+1}, \bar{y})$, donde $\bar{y} = y_n + hf(t_n, y_n)$

- Esto hace que uno se pregunte si una media diferente de dos pendientes daría un método más exacto, por lo que se considera un método más general:

$$y_{n+1} = y_n + c_1 hf(t_n, y_n) + c_2 hf\big(t_n + \alpha h, y_n + \beta hf(t_n, y_n)\big)$$

  - En este caso, c_1, c_2, α y β son parámetros indeterminados. Uno quiere escoger estos para que la solución aproximada definida es lo más exacta posible, por lo que se quiere hacer al error de truncación lo más pequeño posible, en términos de las potencias de h

$$R = y(t+h) - y(t) - c_1 hf\big(t, y(t)\big)$$

$$-c_2 hf\left(t + \alpha h, y(t) + \beta hf\big(t, y(t)\big)\right)$$

  - Para reducir esto y poder inferir los valores correctos de c_1, c_2, α y β que darán el menor residuo R, se necesitará el teorema de Taylor para dos variables:

$$F(x+h, y+\eta) = F(x,y) + hF_x(x,y) + \eta F_y(x,y)$$

$$+\frac{1}{2}\left[h^2 F_{xx}(x,y) + h\eta F_{xy}(x,y) + \eta^2 F_{yy}(x,y)\right] + O(h^3 + \eta^3)$$

$$\Rightarrow f\left(t + \alpha h, y(t) + \beta hf\big(t, y(t)\big)\right) =$$

$$f\big(t, y(t)\big) + \alpha h f_t\big(t, y(t)\big) + \beta hf\big(t, y(t)\big) f_y\big(t, y(t)\big)$$

$$+\frac{1}{2}\Big[\alpha^2 h^2 f_{tt}\big(t, y(t)\big) + \beta^2 h^2 \big[f\big(t, y(t)\big)\big]^2 f_{yy}\big(t, y(t)\big) +$$

$$+\alpha\beta h^2 f\big(t, y(t)\big) f_{ty}\big(t, y(t)\big)\Big] + O(h^3)$$

  - A partir de sustituir expansiones de Taylor de las expresiones involucradas, se puede obtener una expresión derivada para el residuo (aunque no se ponga, todos los argumentos tienen $\big(t, y(t)\big)$):

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2}h^2y''(t) + O(h^3)$$

$$\Rightarrow y''(t) = f_t\big(t, y(t)\big) + f_y\big(t, y(t)\big)y'(t) =$$

$$= f_t\big(t, y(t)\big) + f_y\big(t, y(t)\big)f\big(t, y(t)\big)$$

$$\Rightarrow \; R = \left(y + hy' + \frac{1}{2}h^2y'' + O(h^3)\right) - y - c_1hf$$

$$-c_2h\left[f + \alpha hf_t + \beta hff_y + \frac{1}{2}\left(\alpha^2h^2f_{tt} + \alpha\beta h^2ff_{ty} + \beta^2h^2f^2f_{yy}\right) + O(h^3)\right]$$

$$\Rightarrow R = h(f - c_1f - c_2f) + h^2\left[\frac{1}{2}\left(f_t + f_yf\right) - c_2\alpha f_t - c_2\beta ff_y\right] + O(h^3)$$

$$= h(f - c_1f - c_2f) + h^2\left[\left(\frac{1}{2} - c_2\alpha\right)f_t + \left(\frac{1}{2} - c_2\beta\right)ff_y\right] + O(h^3)$$

- Finalmente, con este resultado, si uno quiere que $R = 0$, entonces se tienen que cumplir las siguientes condiciones, dando lugar a diferentes soluciones (hay 4 incógnitas y 3 ecuaciones) y métodos posibles:

$$\begin{cases} 1 - c_1 - c_2 = 0 \\ 1/2 - c_2\alpha = 0 \\ 1/2 - c_2\beta = 0 \end{cases} \Rightarrow \begin{cases} c_2 = 1 - c_1 \\ \alpha = \beta = 1/2c_2 \end{cases} \quad for\ c_1\ arbitrary$$

  - Cuando $c_1 = c_2 = 1/2$ y $\alpha = \beta = 1$, entonces se obtiene el método de predictor-corrector de la regla del trapezoide:

$$y_{n+1} = y_n + \frac{1}{2}h\big[f\big(t_{n+1}, y_n + hf(t_n, y_n)\big) + f(t_n, y_n)\big]$$

  - Cuando $c_1 = 0$, $c_2 = 1$ y $\alpha = \beta = 1/2$, entonces se obtiene el método de predictor-corrector de la regla del trapezoide:

$$y_{n+1} = y_n + hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(t_n, y_n)\right)$$

  - Cuando $c_1 = 1/4$, $c_2 = 3/4$ y $\alpha = \beta = 2/3$, entonces se obtiene el método de Heun:

$$y_{n+1} = y_n + \frac{1}{4}h\left[f(t_n, y_n) + 3f\left(t_n + \frac{2}{3}h, y_n + \frac{2}{3}hf(t_n, y_n)\right)\right]$$

- Todos los métodos anteriores tienen errores de truncamiento de segunda orden, y se puede demostrar como todos ellos tienen una exactitud de segunda orden, en el sentido que, para toda $T > 0$, se cumple la siguiente desigualdad, en donde C_T depende de T pero no de h:

$$\max_{t_k \leq T} |y(t_k) - y_k| \leq C_T h^2$$

- El método más utilizado es el método de Runge-Kutta de cuarto orden, derivado de manera similar a lo que se hizo anteriormente, pero usando cuatro valores de pendiente en vez de dos. El método normalmente se escribe de la siguiente manera:

$$\begin{cases} k_1 = hf(t_n, y_n) \\ k_2 = hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1\right) \\ k_3 = hf\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2\right) \\ k_4 = hf(t_n + h, y_n + k_3) \\ y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$$

- Un pseudo-código de la implementación de este método concreto es el siguiente:

Algorithm 6.2 *Pseudocode for Fourth-Order Runge–Kutta*

```
input t0, y0, h, n
external f
for k = 1 to n do
     t = t0+k*h
     t2 = t0 + 0.5*h
     v1 = f(t0,y0)
     v2 = f(t2,y0+0.5*h*v1)
     v3 = f(t2,y0+0.5*h*v2)
     v4 = f(t,y0+h*v3)
     y = (h/6.0)*(v1 + 2.0*(v2 + v3) + v4)
     y0 = y
     t0 = t
endfor
```

- La mayor desventaja de los métodos de Runge-Kutta es que requieren más evaluaciones de la función f que otros métodos

- Para funciones escalares simples no es mucho problema, pero si que lo es para sistemas más grandes

        - Para obtener métodos que sean tan exactos como los métodos de Runge-Kutta, pero menos costosos, entonces se tienen que revisar los métodos de múltiples pasos o *multi-step*

Los sistemas de ecuaciones lineales

- Uno de los mayores intereses de los métodos numéricos en álgebra lineal es el de poder aplicarlos para poder resolver sistemas de ecuaciones lineales y no lineales

    - Mucha parte del foco se centra en resolver sistemas de ecuaciones lineales de la forma $\boldsymbol{Ax} = \boldsymbol{b}$, donde $\boldsymbol{A}$ es una matriz de tamaño $K \times K$, $\boldsymbol{x}$ un vector columna K-dimensional y $\boldsymbol{b}$ un vector columna K-dimensional

        - Existen dos tipos de métodos que se pueden utilizar para resolver problemas lineales: los métodos directos y los métodos iterativos:

        - Los métodos directos son métodos que obtienen una solución después de un número de operaciones determinado. No hay posibilidad de controlar la precisión final

        - Los métodos iterativos son métodos que obtienen una solución después de un proceso iterativo. La precisión de la solución dependerá del número de iteraciones y se puede controlar de antemano

    - Otro tipo importante de problemas de álgebra lineal en los que se pueden aplicar métodos numéricos es para sistemas de ecuaciones no lineales, en donde para una función $\boldsymbol{F}$ se tiene que encontrar un vector $\boldsymbol{x}$ que cumpla $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$

        - Sobre todo, se consideran algoritmos iterativos a la hora de resolver este tipo de sistemas

- Dentro del conjunto de métodos directos para la resolución de sistemas de ecuaciones lineales, se pueden encontrar algoritmos como el de Crámer, el método de Gauss, y otros, los cuales permiten entender varios aspectos y problemas de este tipo de métodos

    - Un primer método de solución determinista es el método de Crámer, el cual se basa en la regla de Crámer: si $\boldsymbol{Ax} = \boldsymbol{b}$ es un sistema de ecuaciones y $\det(\boldsymbol{A}) \neq 0$, entonces la solución del sistema es $\boldsymbol{x} = \frac{\det(\boldsymbol{A_i})}{\det(\boldsymbol{A})}$, donde $\boldsymbol{A_i}$ es la matriz resultante de reemplazar la columna i por el vector columna $\boldsymbol{b}$

$$\boldsymbol{Ax} = \boldsymbol{b} \;\Rightarrow\; \boldsymbol{x} = \frac{\det(\boldsymbol{A_i})}{\det(\boldsymbol{A})} \quad for\ i = 1, \dots, K$$

- Con tal de calcular el determinante de tamaño $K \times K$, se necesita calcular $K!$ permutaciones de K elementos. Cada permutación necesita $K - 1$ multiplicaciones y se tiene un total de $K + 1$ determinantes (sumando el de la matriz original $\boldsymbol{A}$)

- Por lo tanto, se necesita un total de $K!\,(K-1)(K+1)$ operaciones. Para valores grandes de K , el número de operaciones es demasiado grande y requiere un coste computacional muy grande, mientras que otro problema importante que puede surgir es el de los errores de redondeo con un número tan grande de operaciones

- Por lo tanto, se prefieren métodos que sean más eficientes y que permitan tener menos inconvenientes

- Es posible poder construir un algoritmo general que permita resolver sistemas de ecuaciones lineales: la eliminación Gaussiana. Este tipo de algoritmo se basa en sistemáticamente eliminar elementos diferentes de cero por debajo de la diagonal principal de la matriz de coeficientes a través de operaciones elementales de fila

  - Este algoritmo se basa en reescribir el sistema de ecuaciones lineales en una matriz aumentada $\boldsymbol{A}' = [\boldsymbol{A}|\boldsymbol{b}]$ y operar a través de operaciones de fila elementales con tal de reducir el sistema en su forma triangular y resolverlo

  - Las operaciones elementales de fila son las comunes: multiplicar una fila por un escalar diferente de cero $c \neq 0$, intercambiar dos filas o multiplicar una fila por un escalar $c \neq 0$ y después sumar el resultado a otra fila (o restarla)

  - Si se puede manipular desde una matriz para obtener otra usando solo operaciones elementales de fila, entonces estas matrices son equivalentes en filas

- El teorema que conecta las operaciones elementales de fila a la solución de sistemas lineales es el siguiente: Siendo $\boldsymbol{A}'$ la matriz aumentada correspondiendo al sistema lineal $\boldsymbol{Ax} = \boldsymbol{b}$ y suponiendo que $\boldsymbol{A}'$ es equivalente en filas a $\boldsymbol{A}'' = [\boldsymbol{T}|\boldsymbol{c}]$, entonces dos sistemas lineales tienen precisamente los mismos conjuntos de solución

  - El objetivo, por tanto, es utilizar operaciones elementales de fila para reducir la matriz aumentada $\boldsymbol{A}'$ en una nueva matriz aumentada $\boldsymbol{A}'' = [\boldsymbol{U}|\boldsymbol{c}]$ en donde $\boldsymbol{U}$ es una matriz triangular

superior, lo cual hará que el sistema $\boldsymbol{Ux} = \boldsymbol{c}$ sea más sencillo de resolver

- El algoritmo general se aplica comenzando desde la primera columna a la izquierda y de arriba abajo. Primero se elimina (se convierte a cero) cada componente debajo de la diagonal principal, y se modifican el resto de los elementos de la fila correspondiente de manera apropiada

$$\begin{cases} 4x_1 + 2x_2 - x_3 = 5 \\ x_1 + 4x_2 + x_3 = 12 \\ 4x_1 - x_2 + 4x_3 = 12 \end{cases} \Rightarrow \begin{bmatrix} 4 & 2 & -1 \\ 1 & 4 & 1 \\ 4 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 12 \\ 12 \end{bmatrix}$$

$$\Rightarrow \boldsymbol{A}' = \left[\begin{array}{ccc|c} 4 & 2 & -1 & 5 \\ 1 & 4 & 1 & 12 \\ 4 & -1 & 4 & 12 \end{array}\right] \Rightarrow \boldsymbol{A}' = \left[\begin{array}{ccc|c} 4 & 2 & -1 & 5 \\ 0 & 7/2 & 5/4 & 43/4 \\ 2 & -1 & 4 & 12 \end{array}\right]$$

$$\Rightarrow \boldsymbol{A}' = \left[\begin{array}{ccc|c} 4 & 2 & -1 & 5 \\ 0 & 7/2 & 5/4 & 43/4 \\ 0 & -2 & 9/2 & 19/2 \end{array}\right]$$

$$\Rightarrow \boldsymbol{A}'' = \left[\begin{array}{ccc|c} 4 & 2 & -1 & 5 \\ 0 & 7/2 & 5/4 & 43/4 \\ 0 & 0 & 73/14 & 219/14 \end{array}\right]$$

$$\Rightarrow \begin{bmatrix} 4 & 2 & -1 \\ 0 & 7/2 & 5/4 \\ 0 & 0 & 73/14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 43/4 \\ 219/14 \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

- Una primera implementación de este algoritmo es la eliminación gaussiana ingenua, el cual es un algoritmo que es fácil de entender, pero no es completamente general. El pseudocódigo de este algoritmo es el siguiente:

Algorithm 7.1 *Naive Gaussian Elimination Algorithm for Ax = b (Pseudocode)*

```
for i=1 to n-1
     for j=i+1 to n
          m = a(j,i)/a(i,i)
          for k=i+1 to n
               a(j,k) = a(j,k) - m*a(i,k)
          endfor
          b(j) = b(j) - m*b(i)
     endfor
endfor
```

  - El primer bucle lo que hace es ir a través de cada una de las columnas de la matriz $\boldsymbol{A}$. Va de 1 a $n - 1$ debido a que en la última columna nunca se tiene que hacer una eliminación (no hay

elementos debajo de la diagonal), solo se tiene que cambiar a n cuando la matriz no es cuadrada y tiene más filas que columnas

- El bucle del medio va a los elementos por debajo de la diagonal de cada columna i, por lo que j va de $i+1$ a n. Primero se calcula el multiplicador m para cada fila (la constante que se usa para multiplicar la fila para eliminar el elemento a_{ji})

- En este último paso se sobreescriben los valores anteriores por nuevos valores y no se realiza realmente el cálculo que hace que a_{ji} sea cero. Además, en este bucle el vector de la parte derecha también se modifica para reflejar el paso de eliminación

- El bucle final va por los elementos de la fila j empezando desde la columna i, modificando cada elemento apropiadamente para reflejar la eliminación de a_{ji}

- Finalmente, uno tiene que ser consciente que el algoritmo no crea ceros en la mitad inferior triangular de $\boldsymbol{A}$, dado que sería una pérdida de tiempo computacional si estos ceros no se necesitan para que funcione el algoritmo

$$\boldsymbol{U} = \begin{bmatrix} 4 & 2 & -1 \\ 0 & 7/2 & 5/4 \\ 0 & 0 & 73/14 \end{bmatrix} \Rightarrow \boldsymbol{U}' = \begin{bmatrix} 4 & 2 & -1 \\ 1 & 7/2 & 5/4 \\ 2 & -2 & 73/14 \end{bmatrix}$$

- Este proceso solo funciona con matrices triangulares superiores, de modo que los elementos triangulares inferiores no necesitan ser referenciados

- Una vez se obtiene el sistema triangular, es posible utilizar el algoritmo de solución hacia atrás o *backsolve algorithm* a la matriz aumentada que resulta del paso de eliminación. Este se denomina paso de solución hacia atrás porque procede hacia atrás en la diagonal desde la de abajo hacia arriba

Algorithm 7.2 *Backward Solution Algorithm for Ax = b (Pseudocode)*

```
x(n) = b(n)/a(n,n)
for i=n-1 to 1
     sum = 0
     for j=i+1 to n
          sum = sum + a(i,j)*x(j)
     endfor
     x(i) = (b(i) - sum)/a(i,i)
endfor
```

- Este algoritmo avanza hacia atrás en la diagonal, calculando x_i en cada paso. Formalmente, se está haciendo el siguiente cálculo, necesario para resolver el sistema triangular:

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j\right) \; for \; i = K-1, K-2, \dots, 1$$

- También es posible llevar a cabo un proceso contrario si se llega a una matriz triangular inferior, el cual se conoce como el algoritmo de solución hacia delante o *forwardsolve algorithm*. No obstante, normalmente se usa el anterior

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j\right) \; for \; i = 2,3, \dots, K$$

- El algoritmo es una aplicación sistemática de eliminación gaussiana, pero hay un problema: en el cálculo del multiplicador, se divide por el valor corriente del elemento diagonal (los valores de la matriz cambian a lo largo del proceso de eliminación)

  - Si uno de esos valores de la diagonal es cero en algún punto del proceso, entonces no se puede proceder con el algoritmo. No obstante, una manera de poder solucionar esto es a través de intercambiar dos filas (operación elemental de filas) para poder hacer que la matriz no tenga elementos nulos en la diagonal

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \Rightarrow \boldsymbol{A}' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

  - Los elementos de la diagonal se llaman pivotes o elementos pivote, y el proceso de intercambiar filas para evitar un elemento nulo en la diagonal se denomina pivotaje. Sin embargo, existen varias maneras de hacer pivotaje

  - El pivotaje parcial, en donde las entradas en la misma columna por debajo de la diagonal actual se examinan, es el tipo más utilizado y el que se discute

  - El pivotaje completo, que busca no solo en la columna actual, sino también en todas las columnas subsiguientes, es más estable, pero también es mucho más costoso de implementar. A veces es necesario escalar las filas antes de pivotar, por lo que se usa el pivotaje escalado (pero completo a la vez)

- El algoritmo de pivotaje parcial es sencillo de describir, y solo se tienen que seguir los siguientes pasos:

  - Suponiendo que se está en la columna i, se busca una porción de la columna i por debajo e incluyendo la diagonal y se encuentra el elemento que tiene el mayor valor absoluto

$$|a_{pi}| = \max_{j \le i \le K} |a_{ji}| \quad \Rightarrow \quad a_{pi}$$

  - Después de encontrar ese valor a_{pi}, se intercambian las filas i y p y después se procede con la eliminación

  - Este algoritmo no solo busca para el primer elemento diferente de cero en la columna para hacerlo de pivote, sino que busca aquel más grande en la columna para ello

- Aunque el problema actual con la eliminación gaussiana es la potencial división entre un pivote cero, el algoritmo entero será menos susceptible a un error de redondeo si se escoge el elemento de pivote más grande (eliminación gaussiana con pivotaje parcial)

Algorithm 7.4 *Gaussian Elimination with Partial Pivoting (Pseudocode)*

```
for i=1 to n-1
     am = abs(a(i,i))
     p = i
     for j=i+1 to n
          if abs(a(j,i)) > am then
               am = abs(a(j,i))
               p = j
          endif
     endfor
     if p > i then
          for k = i to n
               hold = a(i,k)
               a(i,k) = a(p,k)
               a(p,k) = hold
          endfor
          hold = b(i)
          b(i) = b(p)
          b(p) = hold
     endif
     for j=i+1 to n
          m = a(j,i)/a(i,i)
          for k=i+1 to n
               a(j,k) = a(j,k) - m*a(i,k)
          endfor
          b(j) = b(j) - m*b(i)
     endfor
endfor
```

- Como ejemplo de por qué el error de redondeo se puede reducir al pivotar, se puede considerar el siguiente:

$$\begin{bmatrix}\varepsilon & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1\\ 2\end{bmatrix} \Rightarrow \begin{cases}x_1=\dfrac{1}{1-\varepsilon}=1+O(\varepsilon)\\ x_2=\dfrac{1-2\varepsilon}{1-\varepsilon}=1+O(\varepsilon)\end{cases}$$

$$Without\ pivoting:$$

$$\left[\begin{array}{cc|c}\varepsilon & 1 & 1\\ 1 & 1 & 2\end{array}\right] \Rightarrow \left[\begin{array}{cc|c}\varepsilon & 1 & 1\\ 0 & 1-\varepsilon^{-1} & 2-\varepsilon^{-1}\end{array}\right]$$

$$if\ small\ \varepsilon \Rightarrow \begin{cases}1-\varepsilon^{-1}=-\varepsilon^{-1}\\ 2-\varepsilon^{-1}=-\varepsilon^{-1}\end{cases}$$

$$\Rightarrow \left[\begin{array}{cc|c}\varepsilon & 1 & 1\\ 1 & -\varepsilon^{-1} & -\varepsilon^{-1}\end{array}\right] \Rightarrow \begin{cases}x_1\approx 1\\ x_2\approx 0\end{cases}$$

$$With\ pivoting:$$

$$\left[\begin{array}{cc|c}\varepsilon & 1 & 1\\ 1 & 1 & 2\end{array}\right] \Rightarrow \left[\begin{array}{cc|c}1 & 1 & 2\\ \varepsilon & 1 & 1\end{array}\right] \Rightarrow \left[\begin{array}{cc|c}1 & 1 & 2\\ 0 & 1-\varepsilon & 1-2\varepsilon\end{array}\right]$$

$$if\ small\ \varepsilon \Rightarrow \begin{cases}1-\varepsilon=1\\ 1-2\varepsilon=1\end{cases} \Rightarrow \left[\begin{array}{cc|c}1 & 1 & 2\\ 0 & 1 & 1\end{array}\right] \Rightarrow \begin{cases}x_1\approx 1\\ x_2\approx 1\end{cases}$$

- El pivotaje parcial puede ayudar a reducir los errores de redondeo incluso cuando hay algún pivote que no es cero estrictamente, sino que es cercano. Esto es especialmente importante cuando se considera que no se puede predecir el tamaño de los pivotes

- Una variante al algoritmo de eliminación clásico de Gauss es el método de Gauss-Jordan. Este método, en vez de transformar la matriz del sistema en una matriz triangular superior, se obtiene una matriz diagonal, la cual se puede resolver inmediatamente

  - En este caso, la única variación es que se sustrae a todas las filas, menos a la fila p del pivote, esta misma fila p multiplicada por un factor a_{pi}/a_{pp}

  - Esto hará que se obtenga una matriz diagonal adjunta al vector de constantes $\boldsymbol{b}$, haciendo que la solución se calcule de manera sencilla

$$x_i=\frac{b_i}{a_{ii}} \quad for \quad i=1,2,\dots,K$$

- En el caso particular de una matriz tridiagonal dominante, el método de Gauss se conoce como el algoritmo de Thomas

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & a_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{K-1} \\ 0 & \dots & 0 & a_K & b_K \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_K \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_K \end{bmatrix}$$

  - Si la condición $|b_j| > |a_j| + |c_j|$ se cumple, se dice que se tiene una matriz diagonal dominante. El valor absoluto del término diagonal tiene que ser mayor que la suma de los valores absolutos de los elementos de la misma fila j, garantizando la existencia de la solución

  - El algoritmo de Thomas funciona de manera recursiva. Primero se triangulariza la matriz del sistema con las siguientes transformaciones:

$$c_i' = \begin{cases} \dfrac{c_1}{b_1} & for\ \ i = 1 \\ \dfrac{c_i}{b_i - c_{i-1}' a_i} & for\ \ i = 2,3, \dots, K-1 \end{cases}$$

$$d_i' = \begin{cases} \dfrac{d_1}{b_1} & for\ \ i = 1 \\ \dfrac{d_i - d_{i-1}' a_i}{b_i - c_{i-1}' a_i} & for\ \ i = 2,3, \dots, K \end{cases}$$

  - Finalmente, como la matriz resultante es triangular superior, se puede usar la sustitución hacia atrás para resolver el sistema:

$$x_K = \frac{d_K'}{b_n}$$

$$x_i = d_i' - c_i' x_{i+1} \quad for\ \ i = K-1, \dots, 1$$

- Ahora que se ha mostrado la factorización LU, se puede estudiar cómo afectan los cambios en el problema en el proceso de solución. La razón principal para estudiar esto es llegar a un entendimiento de por qué algunos problemas de sistemas lineales son difíciles de resolver, lo cual tiene relación con el concepto de condicionamiento

  - Debido a que uno está interesado en medir errores en vectores y matrices, es necesario tener claros los conceptos de norma vectorial y matricial

- Una norma vectorial en $\mathbb{R}^K$ es una función $\|\cdot\|: \mathbb{R}^K \to \mathbb{R}$ con las siguientes propiedades:

  $(a)\quad \|\boldsymbol{x}\| \geq 0 \quad for\ all\ \boldsymbol{x} \in \mathbb{R}^K$

  $(b)\quad \|\boldsymbol{x}\| = 0 \quad if\ and\ only\ if\ \boldsymbol{x} = \boldsymbol{0}$

  $(c)\quad \|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\| \quad for\ all\ \alpha \in \mathbb{R}\ and\ \boldsymbol{x} \in \mathbb{R}^K$

  $(d)\quad \|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\| \quad for\ all\ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^K$

- Hay muchas posibilidades de definir una norma, aunque las más frecuentes son la Euclídea o L_2 y la infinita o L_∞

$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{K} x_i^2\right)^{1/2} \quad \& \quad \|\boldsymbol{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

- Una norma matricial definida en el conjunto matricial de matrices con coeficientes reales de dimensión $K \times K$ es una función $\|\cdot\|: \mathbb{R}^{K \times K} \to \mathbb{R}$ con las siguientes propiedades:

  $(a)\quad \|\boldsymbol{A}\| \geq 0 \quad for\ all\ \boldsymbol{A} \in \mathbb{R}^{K \times K}$

  $(b)\quad \|\boldsymbol{A}\| = 0 \quad if\ and\ only\ if\ \boldsymbol{A} = \boldsymbol{0}$

  $(c)\quad \|\alpha\boldsymbol{A}\| = |\alpha|\|\boldsymbol{A}\| \quad for\ all\ \alpha \in \mathbb{R}\ and\ \boldsymbol{A} \in \mathbb{R}^{K \times K}$

  $(d)\quad \|\boldsymbol{A} + \boldsymbol{B}\| \leq \|\boldsymbol{A}\| + \|\boldsymbol{B}\| \quad for\ all\ \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{K \times K}$

  $(e)\quad \|\boldsymbol{A}\boldsymbol{B}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{B}\| \quad for\ all\ \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{K \times K}$

- Si $\|\cdot\|$ es una norma vectorial en $\mathbb{R}^K$, entonces la siguiente relación define una norma matricial en el conjunto de matrices reales de dimensión $K \times K$ que se denominará norma natural:

$$\|\boldsymbol{A}\| = \max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\| = \max_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$$

$$\Rightarrow \|\boldsymbol{A}\|_2 = \max_{\|\boldsymbol{x}\|_2=1} \|\boldsymbol{A}\boldsymbol{x}\|_2 = \sqrt{\max \lambda_i(\boldsymbol{A}^T\boldsymbol{A})}$$

$$\Rightarrow \|\boldsymbol{A}\|_\infty = \max_{\|\boldsymbol{x}\|_\infty=1} \|\boldsymbol{A}\boldsymbol{x}\|_\infty = \max_{1 \leq i \leq K} \sum_{i=1}^{K} |a_{ij}|$$

- Suponiendo que se tiene que estudiar un sistema lineal $\boldsymbol{Ax} = \boldsymbol{b}$ y que se está interesando en entender como la solución de $\boldsymbol{x}$ cambia cuando el vector $\boldsymbol{b}$ cambia, se pueden obtener los siguientes resultados:

  - Considerando dos sistemas $\boldsymbol{Ax_1} = \boldsymbol{b_1}$ y $\boldsymbol{Ax_2} = \boldsymbol{b_2}$, y asumiendo que $\boldsymbol{A}$ no es singular, entonces se cumple la siguiente desigualdad para los errores relativos en $\boldsymbol{x_2}$ como una aproximación a $\boldsymbol{x_1}$:

$$\boldsymbol{x_1} - \boldsymbol{x_2} = \boldsymbol{A^{-1}}(\boldsymbol{b_1} - \boldsymbol{b_2}) \;\Rightarrow\; \|\boldsymbol{x_1} - \boldsymbol{x_2}\| = \left\|\boldsymbol{A^{-1}}(\boldsymbol{b_1} - \boldsymbol{b_2})\right\|$$

$$\Rightarrow\; \|\boldsymbol{x_1} - \boldsymbol{x_2}\| \leq \left\|\boldsymbol{A^{-1}}\right\| \|\boldsymbol{b_1} - \boldsymbol{b_2}\|$$

$$\Rightarrow\; \frac{\|\boldsymbol{x_1} - \boldsymbol{x_2}\|}{\|\boldsymbol{x_1}\|} \leq \left\|\boldsymbol{A^{-1}}\right\| \frac{\|\boldsymbol{b_1} - \boldsymbol{b_2}\|}{\|\boldsymbol{x_1}\|}$$

  - Se querría acotar el cambio en la solución a algo que no depende de esta solución (uno quiere deshacerse de $\boldsymbol{x_1}$ en el denominador a la derecha). Para hacer esto, se ve que $\|\boldsymbol{A}\|\|\boldsymbol{x_1}\| \geq \|\boldsymbol{b_1}\|$, por lo que se pueden obtener los siguientes resultados:

$$\frac{1}{\|\boldsymbol{x_1}\|} \leq \frac{\|\boldsymbol{A}\|}{\|\boldsymbol{b_1}\|} \;\Rightarrow\; \frac{\|\boldsymbol{x_1} - \boldsymbol{x_2}\|}{\|\boldsymbol{x_1}\|} \leq \|\boldsymbol{A}\|\left\|\boldsymbol{A^{-1}}\right\| \frac{\|\boldsymbol{b_1} - \boldsymbol{b_2}\|}{\|\boldsymbol{b_1}\|}$$

  - El coeficiente de multiplicación $\|\boldsymbol{A}\|\left\|\boldsymbol{A^{-1}}\right\|$ es interesante: depende enteramente de la matriz $\boldsymbol{A}$ del problema, pero es un amplificador al cambio relativo en el vector $\boldsymbol{b}$. A este factor se le denomina número de condicionamiento

- Para una matriz $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ y una norma matricial $\|\cdot\|$, el número de condicionamiento $\kappa(\boldsymbol{A})$ con respecto a la norma dada se define de la siguiente manera, usando por convención $\kappa(\boldsymbol{A}) = \infty$ si $\boldsymbol{A}$ es singular:

$$\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\left\|\boldsymbol{A^{-1}}\right\|$$

  - Siendo $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ una matriz no singular, para cualquier matriz singular $\boldsymbol{B} \in \mathbb{R}^{K\times K}$, se cumple la siguiente desigualdad (esta es la razón para usar esa convención):

$$\frac{1}{\kappa(\boldsymbol{A})} = \frac{1}{\|\boldsymbol{A}\|\|\boldsymbol{A^{-1}}\|} = \frac{1}{\|\boldsymbol{A}\|}\left(\frac{1}{\max\limits_{\boldsymbol{x}\neq\boldsymbol{0}} \|\boldsymbol{A^{-1}x}\|/\|\boldsymbol{x}\|}\right)$$

$$\leq \frac{1}{\|\boldsymbol{A}\|}\left(\frac{1}{\|\boldsymbol{A^{-1}x}\|/\|\boldsymbol{x}\|}\right) \leq \frac{1}{\|\boldsymbol{A}\|}\left(\frac{\|\boldsymbol{Ay}\|}{\|\boldsymbol{y}\|}\right) \;\; for \;\; \forall \boldsymbol{y} \in \mathbb{R}^K$$

$$If \;\; \boldsymbol{y} \;\; s.t. \;\; \boldsymbol{By} = \boldsymbol{0} \;\; then:$$

$$\Rightarrow \ \frac{1}{\kappa(\boldsymbol{A})} \leq \frac{\|(\boldsymbol{A}-\boldsymbol{B})\boldsymbol{y}\|}{\|\boldsymbol{A}\|\|\boldsymbol{y}\|} \leq \frac{\|\boldsymbol{A}-\boldsymbol{B}\|\|\boldsymbol{y}\|}{\|\boldsymbol{A}\|\|\boldsymbol{y}\|} = \frac{\|\boldsymbol{A}-\boldsymbol{B}\|}{\|\boldsymbol{A}\|}$$

- La importancia de este resultado es que cuando $\boldsymbol{A}$ es cercana a ser no invertible, entonces el recíproco del número de condicionamiento es cercano a cero ($\kappa(\boldsymbol{A})$ será muy grande). Por lo tanto, el número de condicionamiento mide que tan cercana es una matriz a ser singular o no invertible

- Resolver sistemas lineales que son muy cercanos a ser singulares es que se pueden producir grandes errores, dado que el proceso de solución se ve afectado el error de redondeo. Aunque cada instancia de error de redondeo será muy pequeña, si K es grande, entonces habrá un enorme número de redondeos individuales que suman

- Siendo $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ una matriz no singular y $\boldsymbol{b} \in \mathbb{R}^{K}$, y definiendo $\boldsymbol{x} \in \mathbb{R}^{K}$ como la solución del sistema lineal $\boldsymbol{Ax} = \boldsymbol{b}$, para un error de perturbación pequeño $\boldsymbol{\delta b} \in \mathbb{R}^{K}$ de $\boldsymbol{b}$ y la solución $\boldsymbol{x} + \boldsymbol{\delta x} \in \mathbb{R}^{K}$ al sistema $\boldsymbol{A}(\boldsymbol{x} + \boldsymbol{\delta x}) = \boldsymbol{b} + \boldsymbol{\delta b}$, entonces se cumple la siguiente desigualdad:

$$\frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \leq \kappa(\boldsymbol{A}) \frac{\|\boldsymbol{\delta b}\|}{\|\boldsymbol{b}\|}$$

  - La demostración de este resultado es la siguiente:

$$\boldsymbol{\delta x} = \boldsymbol{A}^{-1}\boldsymbol{\delta b} \ \Rightarrow \|\boldsymbol{\delta x}\| \leq \|\boldsymbol{A}^{-1}\|\|\boldsymbol{\delta b}\|$$

$$\Rightarrow \frac{\|\boldsymbol{\delta x}\|}{\|\boldsymbol{x}\|} \leq \frac{\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\|\boldsymbol{\delta b}\|}{\|\boldsymbol{A}\|\|\boldsymbol{x}\|}$$

$$as \ \|\boldsymbol{A}\|\|\boldsymbol{x}\| \geq \|\boldsymbol{b}\| \ \Rightarrow \ \frac{\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|\|\boldsymbol{\delta b}\|}{\|\boldsymbol{A}\|\|\boldsymbol{x}\|} \leq \kappa(\boldsymbol{A}) \frac{\|\boldsymbol{\delta b}\|}{\|\boldsymbol{b}\|}$$

  - La importancia de este teorema es mostrar que las perturbaciones en el problema afectan a la solución final en una forma que se amplifica por el número de condicionamiento de la matriz. Si la matriz tiene mal condicionamiento o es *ill-conditioned* (cuando $\kappa(\boldsymbol{A})$ es muy grande), entonces un pequeño cambio en los datos puede llevar a un gran cambio en la solución

- Un resultado similar involucra el residual de la solución: siendo $\boldsymbol{x_c}$ una solución calculada a $\boldsymbol{Ax} = \boldsymbol{b}$, entonces el vector residual es $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax_c}$ (la cantidad por la que $\boldsymbol{x_c}$ falla de ser la solución exacta). Si $\boldsymbol{r} = \boldsymbol{0}$, entonces $\boldsymbol{x_c}$ es exacta, por lo que se puede pensar que un vector

pequeño $\boldsymbol{r}$ indica que se está cerca de la solución exacta, pero esto no siempre es así

- Siendo $\boldsymbol{A} \in \mathbb{R}^{K \times K}$ una matriz no singular y $\boldsymbol{b} \in \mathbb{R}^{K}$ y definiendo $\boldsymbol{x_c} \in \mathbb{R}^{K}$ como la solución calculada al sistema lineal $\boldsymbol{Ax} = \boldsymbol{b}$ y $\boldsymbol{r} \in \mathbb{R}^{K}$ como el residual $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Ax_c}$, entonces se cumple la siguiente desigualdad:

$$\frac{\|\boldsymbol{x} - \boldsymbol{x_c}\|}{\|\boldsymbol{x}\|} \leq \kappa(\boldsymbol{A}) \frac{\|\boldsymbol{r}\|}{\|\boldsymbol{b}\|}$$

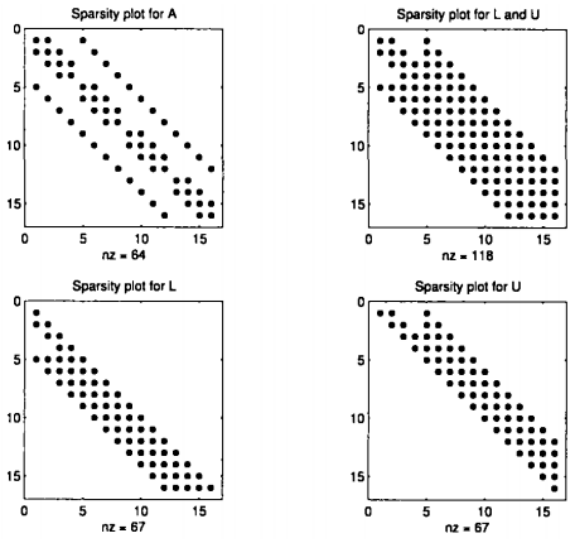
- La idea, por tanto, de esta desigualdad y de la anterior es que uno no puede confiar en la solución de un problema de una matriz mal condicionada a menos que se tenga especial cuidado con la solución. Otro resultado similar considera las perturbaciones en la misma matriz $\boldsymbol{A}$

- Si la matriz de coeficientes es muy grande y escasa o *sparse* (el número de elementos diferentes de cero es una fracción pequeña del total de elementos), entonces la eliminación gaussiana puede no ser la mejor manera de resolver un sistema lineal. Como alternativa, se utilizan métodos iterativos

  - En estas circunstancias, las técnicas de eliminación se ralentizarán drásticamente por los problemas de la gestión de memoria asociada con lidiar con una gran matriz, y también destruirán la escasez de la matriz original

$$A = \left[\begin{array}{cccc|cccc|cccc|cccc}
-4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 0 & 0 & 0 & 1 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -4 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4
\end{array}\right]$$

  - Aunque $\boldsymbol{A} = \boldsymbol{LU}$ sea escasa, los factores individuales $\boldsymbol{L}$ y $\boldsymbol{U}$ pueden no ser tan escasas como $\boldsymbol{A}$ lo es, haciendo que haya muchos números de relleno en la factorización que no corresponden con la matriz original. A través de los *sparsity plots* es posible comprobar cómo, a partir de $\boldsymbol{A}$, los factores $\boldsymbol{L}$ y $\boldsymbol{U}$ hacen que se tenga más celdas rellenas que la matriz original



- Es por los problemas de la eliminación gaussiana y los métodos de factorización que estos métodos no son la mejor elección. En vez de esto, se consideran métodos iterativos para los sistemas lineales

- Esto parece contraintuitivo debido a que se deja un modelo que da una solución exacta y que no da errores de redondeo por un método que da. Sin embargo, los métodos iterativos son eficientes en costes para problemas grandes y escasos, especialmente cuando $\boldsymbol{A}$ es simétrica y exhibe una fuerte estructura para el posicionamiento de sus elementos no nulos y nulos

- El uso de métodos iterativos permite obtener una solución al sistema lineal con un refinamiento progresivo, pero es necesario tener una medida de distancia entre dos vectores de iteraciones consecutivas (para evaluar qué tan buena es la respuesta). El concepto de norma vectorial y matricial, por tanto, es necesaria

- A partir de la norma, se puede definir la distancia entre vectores, que no es más que la norma de la diferencia entre vectores. Dada una distancia, se puede decidir si una sucesión vectorial se acerca a cierto límite, el cual se toma como vector solución

  - Una sucesión de vectores $\{\boldsymbol{x}^{(k)}\}$ es convergente a un vector $\boldsymbol{x}$ en $\mathbb{R}^K$ con respecto a cierta norma $\|\cdot\|$ si dado un valor $\varepsilon > 0$, existe un entero N_ε tal que se cumple la siguiente desigualdad:

$$\left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\right\| < \varepsilon \quad for\ all\ k \geq N_\varepsilon$$

- Una sucesión de vectores $\{\boldsymbol{x}^{(k)}\}$ es convergente a un vector $\boldsymbol{x}$ en $\mathbb{R}^K$ con respecto de la misma norma $\|\cdot\|$, si, y solo si, se cumple el siguiente límite:

$$\lim_{k\to\infty} x_i^{(k)} = x_i \quad for \ \ each \ \ i = 1,2,\dots,K$$

- Al detener los métodos iterativos, se tiene que considerar que la solución exacta es desconocida, y es necesario usar los valores calculados. Por lo tanto, primero es necesario encontrar una tolerancia $\varepsilon > 0$, que será una medida de precisión a la que se quiere llegar, y es necesario fijar un criterio para parar los cálculos:

$$\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\| < \varepsilon \quad or \quad \frac{\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\|}{\|\boldsymbol{x}^{(k)}\|} < \varepsilon$$

- Cuando se usa un método iterativo para resolver un sistema lineal de la forma $\boldsymbol{Ax} = \boldsymbol{b}$, se quiere encontrar una sucesión de vectores $\{\boldsymbol{x}^{(k)}\}$ con $k \geq 0$, verificando $\lim_{k\to\infty} x_i^{(k)} = x_i$, la cual sería entonces convergente

  - El sistema lineal $\boldsymbol{Ax} = \boldsymbol{b}$ se puede reescribir como $\boldsymbol{x} = \boldsymbol{Tx} + \boldsymbol{c}$ (de forma lineal). Esto permite escribir la siguiente regla recurrente, donde la matriz $\boldsymbol{T}$ se conoce como matriz iterativa:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{T}\boldsymbol{x}^{(k)} + \boldsymbol{c} \quad for \quad k \geq 0$$

  - Aunque $\boldsymbol{x}^{(0)}$ pueda ser un vector arbitrario, cuanto más cerca esté a la solución final, mejor, dado que la convergencia será más rápida

- Una clase importante de métodos iterativos pertenecen a la clase de métodos de separación o *splitting methods*, dado que se basan en la noción de separar la matriz de coeficientes en dos partes

  - Suponiendo que se quiere resolver un sistema $\boldsymbol{Ax} = \boldsymbol{b}$, donde $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ no es singular, se puede separar $\boldsymbol{A}$ en la diferencia $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$, en donde $\boldsymbol{M}$ es tal que los sistemas de la forma $\boldsymbol{Mz} = \boldsymbol{f}$ son fácilmente resolubles. Debido a que $\boldsymbol{Mz} = \boldsymbol{f}$ es fácilmente resoluble, entonces $\boldsymbol{M}^{-1}\boldsymbol{f}$ es fácil de calcular para cualquier $\boldsymbol{f}$, se puede obtener la siguiente expresión:

$$\boldsymbol{Ax} = \boldsymbol{b} \ \Rightarrow \ (\boldsymbol{M} - \boldsymbol{N})\boldsymbol{x} = \boldsymbol{b} \ \Rightarrow \ \boldsymbol{Mx} = \boldsymbol{Nx} + \boldsymbol{b}$$

$$\Rightarrow \ \boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{Nx} + \boldsymbol{M}^{-1}\boldsymbol{b}$$

- Esto sugiere que se puede realizar la siguiente iteración: dada una solución inicial $\boldsymbol{x}^{(0)}$, se puede calcular una secuencia de vectores acorde a la ecuación anterior hasta que converja

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{M}^{-1}\boldsymbol{N}\boldsymbol{x}^{(k)} + \boldsymbol{M}^{-1}\boldsymbol{b}$$

- Existe una gran literatura para estos esquemas generales de iteraciones, mostrando que la iteración converge si, y solo si, la matriz de iteración $\boldsymbol{T} = \boldsymbol{M}^{-1}\boldsymbol{N}$ es menor a 1 en un sentido concreto

  - Se puede reescribir la matriz de iteración de una manera en la que se ve que el método funciona mejor cuando $\boldsymbol{M}^{-1} \approx \boldsymbol{A}^{-1}$ o cuando $\boldsymbol{M} \approx \boldsymbol{A}$ (dado que de esta manera $\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A} \approx \boldsymbol{0}$)

$$\boldsymbol{T} = \boldsymbol{M}^{-1}\boldsymbol{N} = \boldsymbol{M}^{-1}(\boldsymbol{M} - \boldsymbol{A}) = \boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}$$

  - Siendo $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ y definiendo $\boldsymbol{T} = \boldsymbol{M}^{-1}\boldsymbol{N}$, donde $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$, entonces la iteración $\boldsymbol{x}^{(k+1)} = \boldsymbol{M}^{-1}\boldsymbol{N}\boldsymbol{x}^{(k)} + \boldsymbol{M}^{-1}\boldsymbol{b}$ converge para cualquier valor inicial $\boldsymbol{x}^{(0)}$ si, y solo si, existe una norma $\|\cdot\|$ tal que $\|\boldsymbol{T}\| < 1$

- Existe un resultado que relaciona la existencia de esta norma con una condición cuantitativa para los valores propios de la matriz $\boldsymbol{T}$, relacionado con el radio espectral de una matriz

  - Siendo $\boldsymbol{A} \in \mathbb{R}^{K\times K}$, entonces el radio espectral de $\boldsymbol{A}$, denotado como $\rho(\boldsymbol{A})$, es el mayor (en magnitud) de todos los valores propios de $\boldsymbol{A}$

$$\rho(\boldsymbol{A}) = \max_{\lambda_i} |\lambda_i|$$

  - Si $\boldsymbol{A}$ es una matriz real de tamaño $K \times K$, entonces para cualquier norma natural se cumplen las siguientes relaciones:

    $(a)\quad [\rho(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})]^{1/2} = \|\boldsymbol{A}\|_2$

    $(b)\quad \rho(\boldsymbol{A}) \leq \|\boldsymbol{A}\|$

  - Siendo $\boldsymbol{A} \in \mathbb{R}^{K\times K}$ una matriz cualquiera, entonces existe una norma $\|\cdot\|$ tal que $\|\boldsymbol{A}\| < 1$ si, y solo si, el radio espectral satisface $\rho(\boldsymbol{A}) < 1$

  - Por lo tanto, se concluye que el método de iteración anteriormente visto converge para cualquier valor inicial $\boldsymbol{x}^{(0)}$ si, y solo si, $\rho(\boldsymbol{T}) < 1$

- El método más simple que se basa en este desarrollo teórico, aunque sea el más lento en converger, es la iteración de Jacobi. Este método se basa en una inversión de la parte diagonal de $\boldsymbol{A}$ en cada paso de la iteración

  - En este caso, se usa la separación $\boldsymbol{A} = \boldsymbol{D} - (\boldsymbol{D} - \boldsymbol{A})$, donde $\boldsymbol{D}$ es la parte diagonal de $\boldsymbol{A}$, y la iteración es la siguiente:

$$\boldsymbol{D}\boldsymbol{x} - (\boldsymbol{D} - \boldsymbol{A})\boldsymbol{x} = \boldsymbol{b} \Rightarrow \boldsymbol{x} = \boldsymbol{D}^{-1}(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{x} + \boldsymbol{D}^{-1}\boldsymbol{b}$$

$$\Rightarrow \boldsymbol{x}^{(k+1)} = (\boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A})\boldsymbol{x}^{(k)} + \boldsymbol{D}^{-1}\boldsymbol{b} \quad where \ \boldsymbol{T}_J = \boldsymbol{I} - \boldsymbol{D}^{-1}\boldsymbol{A}$$

$$\Rightarrow x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}\right) \quad for \ i = 1,2,\dots,K$$

  - La mejor implementación de este método (y para todos los métodos de separación) depende de la estructura de la matriz de coeficientes, de modo que es difícil discutirla en general. Si se tiene una rutina especializada para formar productos de matriz y vectores involucrando $\boldsymbol{A}$ (usando su estructura para maximizar la eficiencia), entonces se puede tomar ventaja de esto y aplicar la siguiente iteración:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \boldsymbol{D}^{-1}\left(\boldsymbol{A}\boldsymbol{x}^{(k)}\right) + \boldsymbol{D}^{-1}\boldsymbol{b}$$

- Una extensión obvia de este método es invertir la parte triangular inferior de $\boldsymbol{A}$ completamente, no solo la parte diagonal. A este método se le llama la iteración de Gauss-Seidel

  - En este caso, se separa $\boldsymbol{A}$ en $\boldsymbol{A} = \boldsymbol{L} - (\boldsymbol{L} - \boldsymbol{A})$, de modo que la iteración es la siguiente:

$$\boldsymbol{L}\boldsymbol{x} - (\boldsymbol{L} - \boldsymbol{A})\boldsymbol{x} = \boldsymbol{b} \Rightarrow \boldsymbol{x} = \boldsymbol{L}^{-1}(\boldsymbol{L} - \boldsymbol{A})\boldsymbol{x} + \boldsymbol{L}^{-1}\boldsymbol{b}$$

$$\Rightarrow \boldsymbol{x}^{(k+1)} = (\boldsymbol{I} - \boldsymbol{L}^{-1}\boldsymbol{A})\boldsymbol{x}^{(k)} + \boldsymbol{L}^{-1}\boldsymbol{b} \quad where \ \boldsymbol{T}_{GS} = \boldsymbol{I} - \boldsymbol{L}^{-1}\boldsymbol{A}$$

$$\Rightarrow x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{K} a_{ij} x_j^{(k)}\right)$$

$$for \ i = 1,2,\dots,K$$

  - Igual que antes, esta iteración se puede implementar de manera alternativa:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \boldsymbol{L}^{-1}\left(\boldsymbol{A}\boldsymbol{x}^{(k)}\right) + \boldsymbol{L}^{-1}\boldsymbol{b}$$

- El algoritmo de Gauss-Seidel converge más rápido que el algoritmo de Jacobi, lo cual es consiste con la intuición matemática porque este primer método se basa en invertir "más" de la matriz original en cada caso. El resultado teórico formal viene dado por el teorema de convergencia de Jacobi y Gauss-Seidel

    - Si $\boldsymbol{A}$ es una matriz diagonalmente dominante, entonces tanto el método de Jacobi como el de Gauss-Seidel convergen, y el método de Gauss-Seidel converge más rápido en el sentido de que $\rho(\boldsymbol{T}_{GS}) < \rho(\boldsymbol{T}_J)$

    - Si $\boldsymbol{A}$ es una matriz semidefinida positiva (si $\boldsymbol{A} \in SPD$), entonces tanto el método de Jacobi como el de Gauss-Seidel convergen

$$\boldsymbol{z}'\boldsymbol{A}\boldsymbol{z} = [z_1 \ \dots \ z_K] \begin{bmatrix} a_{11} & \cdots & \cdots & a_{1K} \\ \cdots & a_{22} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ a_{K1} & \cdots & \cdots & a_{KK} \end{bmatrix} \begin{bmatrix} z_1 \\ \cdots \\ z_K \end{bmatrix} \geq 0$$

$$\Rightarrow \ \rho(\boldsymbol{T}_{GS}), \rho(\boldsymbol{T}_J) < 1$$

- Aunque no se sepa la solución exacta del problema, se puede poner un límite superior para el error después de k iteraciones, lo cual permite hacer un análisis del error para los métodos iterativos

    - A partir de la relación entre las iteraciones, es posible obtener una cota superior para el error relativo de la solución calculada contra la solución exacta del problema

$$\boldsymbol{x}^{(k+1)} - \boldsymbol{x} = \boldsymbol{T}\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}\right) = \cdots = \boldsymbol{T}^{k+1}\left(\boldsymbol{x}^{(0)} - \boldsymbol{x}\right)$$

$$\Rightarrow \ \left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\right\| = \left\|\boldsymbol{T}^k\left(\boldsymbol{x}^{(0)} - \boldsymbol{x}\right)\right\| \leq \|\boldsymbol{T}^k\|\left\|\boldsymbol{x}^{(0)} - \boldsymbol{x}\right\| \leq$$

$$\leq \|\boldsymbol{T}\|^k\left\|\boldsymbol{x}^{(0)} - \boldsymbol{x}\right\|$$

$$if \ \ \boldsymbol{x}^{(0)} = 0 \ \ \Rightarrow \ \frac{\left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\right\|}{\|\boldsymbol{x}\|} \leq \|\boldsymbol{T}\|^k$$

    - Es posible demostrar que esta desigualdad se puede expresar en términos del radio espectral de la matriz de iteración:

$$\left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\right\| \leq \rho(\boldsymbol{B})^k\left\|\boldsymbol{x}^{(0)} - \boldsymbol{x}\right\|$$

$$if \ \ \boldsymbol{x}^{(0)} = 0 \ \ \Rightarrow \ \frac{\left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\right\|}{\|\boldsymbol{x}\|} \leq \rho(\boldsymbol{B})^k$$

- Suponiendo que se sabe el radio espectral y que se quiere una estimación del número de iteraciones necesario para obtener un cierto error relativo del orden de 10^{-t}, se comienza de un vector nulo $\boldsymbol{x}^{(0)} = \mathbf{0}$ y se puede obtener una relación concreta para obtener un número mínimo de iteraciones:

$$\frac{\left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\right\|}{\|\boldsymbol{x}\|} \leq \rho(\boldsymbol{B})^k \leq 10^{-t} \;\Rightarrow\; k \geq \frac{t}{-\log_{10}[\rho(\boldsymbol{B})]}$$

- Cuanto menor sea el radio espectral, menor será el número de iteraciones k mínimo y más rápido será el método

- No es posible decidir *a priori* si el método de Jacobi será más rápido que el de Gauss-Seidel. Esto solo se puede saber con certeza a través del teorema de Stein-Rosenberg, usado en soluciones numéricas para ecuaciones diferenciales parciales elípticas

  - Siendo $\boldsymbol{T}_J$ la matriz de iteración del método de Jacobi y $\boldsymbol{T}_{GS}$ la del método de Gauss-Seidel, si $a_{ij} \leq 0$ para $i \neq j$ y $a_{ii} > 0$ para toda $i = 1,2,\ldots,K$ solo una de las siguientes proposiciones es verdad:

    $(a)\;\; 0 < \rho(\boldsymbol{T}_{GS}) < \rho(\boldsymbol{T}_J) < \mathbf{1}$

    $(b)\;\; 1 < \rho(\boldsymbol{T}_J) < \rho(\boldsymbol{T}_{GS})$

    $(c)\;\; \rho(\boldsymbol{T}_{GS}) = \rho(\boldsymbol{T}_J) = 0$

    $(d)\;\; \rho(\boldsymbol{T}_{GS}) = \rho(\boldsymbol{T}_J) = 1$

  - Si uno de los métodos es convergente, el otro será también convergente. En este caso, es mejor usar en el método de Gauss-Seidel, dado que tiene un radio espectral menor

  - Este resultado es general. El método más rápido será aquel con radio espectral más pequeño o con una norma más pequeña para la matriz de iteración

Los métodos numéricos para PDEs: diferencias finitas

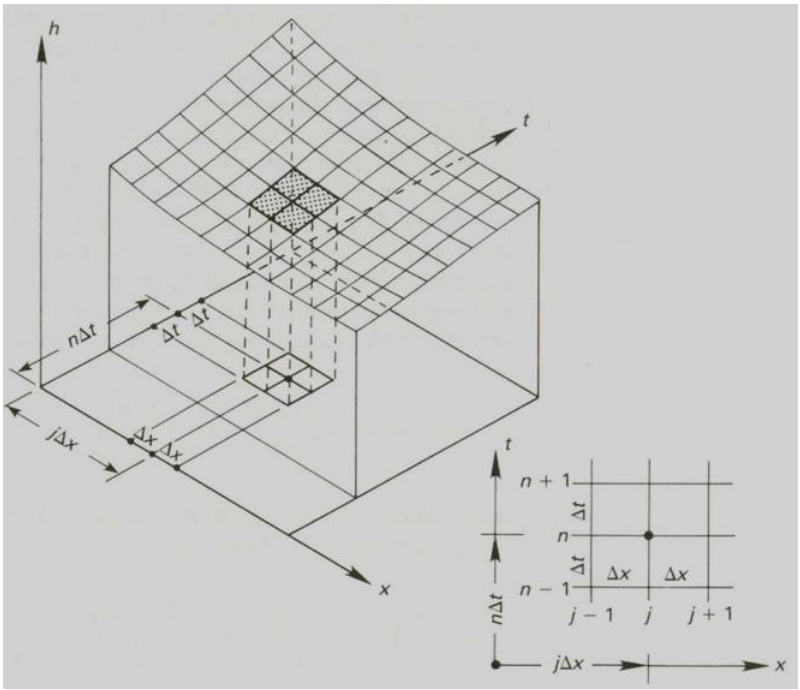
- Los métodos numéricos usados para ecuaciones diferenciales parciales suelen basarse en diferencias finitas, de modo que las derivadas en las PDEs se cambian por expresiones equivalentes en derivadas finitas para poder pasar de cálculos diferenciales a aritméticos

  - Se puede considerar una PDE como una caracterización de un problema en una región infinitesimal dentro de un dominio

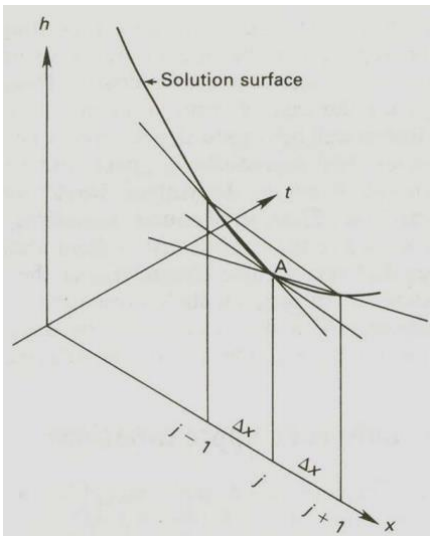
- Con el método de diferencias finitas, se extiende esta caracterización a una región finita del dominio usando una reja o *mesh* regular de puntos sobreimpuesta en el dominio

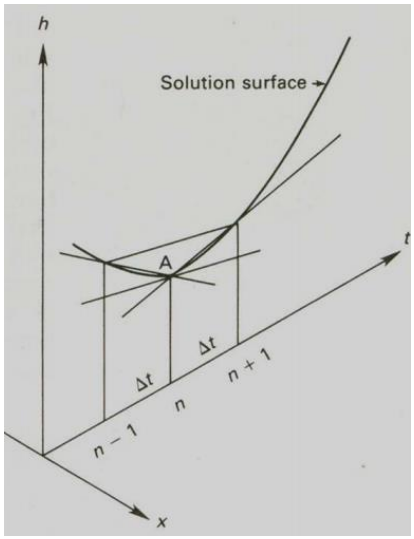
- Si se está usando una dimensión espacial y una dimensión temporal, los puntos en la reja se puede separar por cantidades Δx y Δt que son pasos constantes. Por lo tanto, cada punto se puede expresar como (x, t), y numéricamente se obtiene la siguiente equivalencia:

$$(x, t) = (i\Delta x, n\Delta t) \quad for \quad i, n = 1,2, \ldots$$

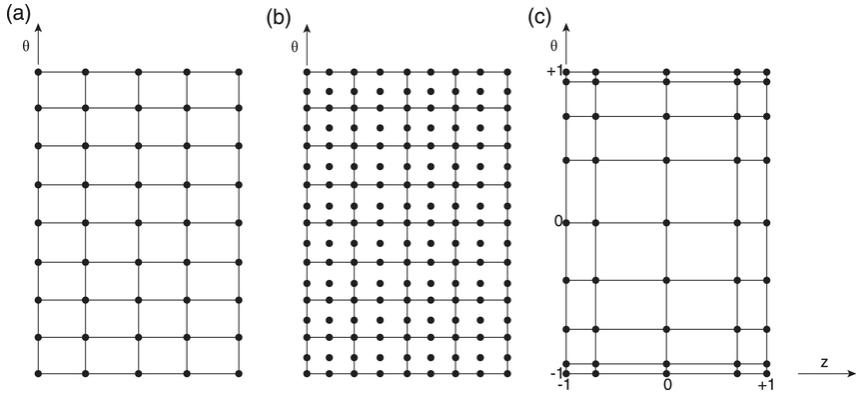


- Si se fija un nivel de una variable, como por ejemplo $n\Delta t$ para el tiempo t, entonces la solución no es más que una curva unidimensional para x, y la pendiente de esta curva será la derivada $\partial u/\partial x$ en cada punto. Del mismo modo, se puede fijar un valor espacial $j\Delta x$ para poder obtener una curva unidimensional del tiempo t, con pendiente $\partial u/\partial t$





- Las rejas de los métodos numéricos para ecuaciones diferenciales parciales suelen ser equidistantes: para cada variable, se define un paso para hacer los puntos o nodos equidistantes en ese eje y se obtiene una reja regular. No obstante, también se podrían plantear rejas irregulares



- Para dar una versión discreta de las ecuaciones, se usa una expansión de Taylor de la solución $u(t,x)$ alrededor de un punto particular del dominio, tal como se planteó anteriormente para la aproximación de la derivada

  - La derivación se podría obtener a través de usar polinomios interpoladores, pero en este caso no es fácil controlar la precisión de la aproximación. De este modo, el enfoque que se usó anteriormente se deja de lado y se usa el de expansiones de Taylor

  - A través de la expansión de Taylor univariante y denotando u_j^n como la función en el punto (j,n), se pueden obtener las siguientes aproximaciones para x (análogas para t):

$$u_{j-1}^n = u_j^n - \frac{\partial u}{\partial x}\Delta x + \frac{\partial^2 u}{\partial x^2}\frac{(\Delta x)^2}{2!} - \frac{\partial^3 u}{\partial x^3}\frac{(\Delta x)^3}{3!} + O[(\Delta x)^4]$$

$$\Rightarrow \frac{\partial u}{\partial x} = \frac{u_j^n - u_{j-1}^n}{\Delta x} + O(\Delta x) \quad (regressive\ \ diff.)$$

$$u_{j+1}^n = u_j^n + \frac{\partial u}{\partial x}\Delta x + \frac{\partial^2 u}{\partial x^2}\frac{(\Delta x)^2}{2!} + \frac{\partial^3 u}{\partial x^3}\frac{(\Delta x)^3}{3!} + O[(\Delta x)^4]$$

$$\Rightarrow \frac{\partial u}{\partial x} = \frac{u_{j+1}^n - u_j^n}{\Delta x} + O(\Delta x) \quad (progresive\ \ diff.)$$

$$2\frac{\partial u}{\partial x} = \frac{u_{j+1}^n - u_j^n}{\Delta x} + \frac{u_j^n - u_{j-1}^n}{\Delta x} - 2\frac{\partial^3 u}{\partial x^3}\frac{(\Delta x)^2}{3!} + O[(\Delta x)^4]$$

$$\Rightarrow \frac{\partial u}{\partial x} = \frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + O[(\Delta x)^2] \quad (centered\ \ diff.)$$

- De este modo, las ecuaciones diferenciales parciales más comunes se pueden expresar usando diferencias finitas con tal de obtener una expresión recursiva que use la función u:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \ \Rightarrow \ \frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_j^n - u_{j-1}^n}{\Delta x} = 0$$

$$\Rightarrow \ u_j^{n+1} = u_j^n(1 - C) + Cu_{j-1}^n \quad where \ \ C = c\frac{\Delta t}{\Delta x}$$

- Hay otras posibilidades para escribir una PDE en diferencias finitas, combinando derivadas progresivas, regresivas y centradas de diferentes maneras, o usando expresiones de mayor orden para las diferencias finitas (tal y como se ha visto anteriormente). Lo importante es escoger que el esquema seleccionado en diferencias finitas es apropiado para sustituir en la PDE

- Con tal de poder obtener una buena aproximación numérica de la solución de una PDE concreta, será necesario tener en cuenta los siguientes aspectos:

  - Seleccionar un buen esquema para las diferencias finitas, escogiendo el orden de la aproximación y su progresividad, regresividad y otros

  - Seleccionar un valor adecuado para los parámetros de la reja $\Delta x, \Delta t$, …, dado que determinarán los pasos y pueden hacer que el algoritmo converja o no

  - Usar algoritmos eficientes que usen pocos requerimientos de memoria y sean rápidos en términos de la CPU

- La expansión de Taylor permite evaluar el error de aproximación o de truncamiento de los errores de una ecuación en diferencias finitas, llevando a la noción de consistencia, convergencia y estabilidad

    - Si cuando $\Delta t \to 0$ y $\Delta x \to 0$, el esquema se aproxima a la PDE en el límite, entonces se dice que el esquema de diferencias finitas es consistente. Esto se comprueba llevando aquellos términos que se truncan de la expansión de Taylor a la ecuación:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$$

$$\Rightarrow \frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_j^n - u_{j-1}^n}{\Delta x} + \left[\frac{\partial^2 u}{\partial t^2}\frac{\Delta t}{2!} - c\frac{\partial^2 u}{\partial x^2}\frac{\Delta x}{2!} + \cdots\right] = 0$$

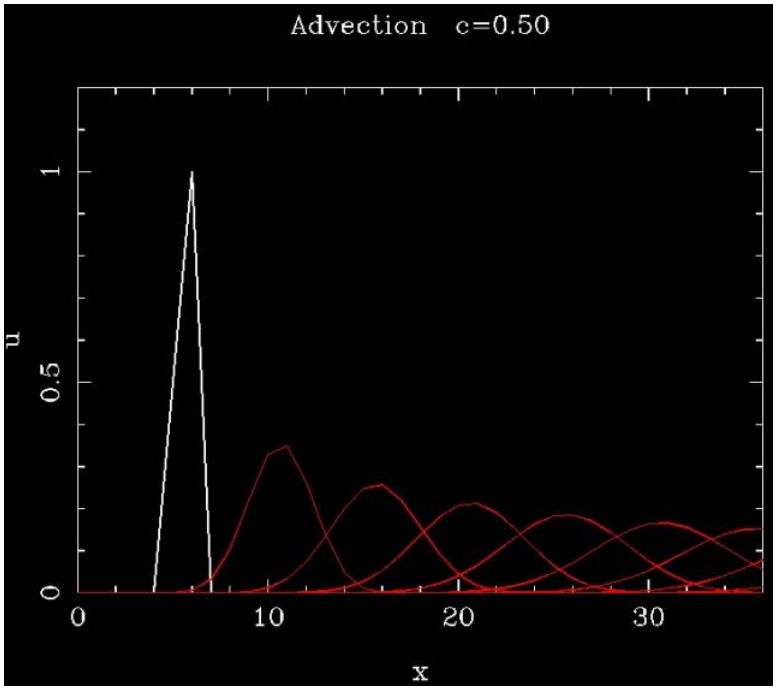
$$\Rightarrow \lim_{\Delta t, \Delta x \to 0} \frac{\partial^2 u}{\partial t^2}\frac{\Delta t}{2!} - c\frac{\partial^2 u}{\partial x^2}\frac{\Delta x}{2!} + \frac{\partial^3 u}{\partial t^3}\frac{(\Delta t)^2}{3!} - c\frac{\partial^3 u}{\partial x^3}\frac{(\Delta x)^2}{3!} + \cdots = 0$$

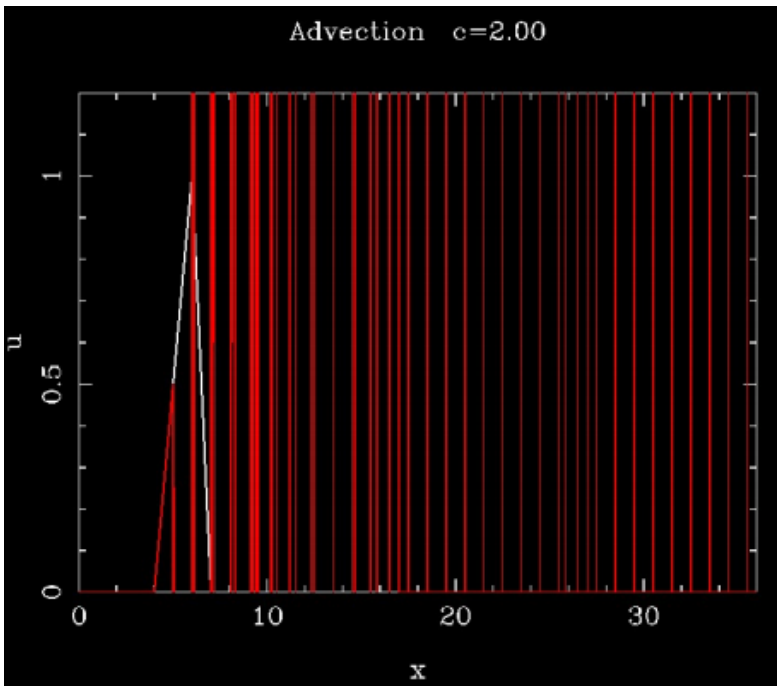
    - Cuando un esquema es consistente independientemente del camino de discretización que se haga para las derivadas, se dice que el esquema es incondicionalmente consistente. Cabe recordar, no obstante, que el concepto de convergencia solo se refiere a la relación entre la PDE original y la PDE expresada en diferencias finitas

    - El concepto de convergencia en este caso se refiere al hecho de que las soluciones de la ecuación en diferencias finitas se acerquen a la solución real exacta cuando $\Delta t \to 0$ y $\Delta x \to 0$. Por lo tanto, la convergencia se puede interpretar como la relación entre las soluciones del algoritmo y la real

    - Cuando se considera la convergencia de una solución, se quiere que esta convergencia sea suave, que es lo mismo que pedir que los errores de aproximación vayan a cero en todos los lugares (de modo que no haya cambios bruscos). Si se genera una secuencia de soluciones numéricas cuando $\Delta t \to 0$ y $\Delta x \to 0$ y los valores se mantienen finitos dentro de un dominio, se dice que el esquema es estable

- La convergencia y la estabilidad están relacionados, dado que la inestabilidad de un esquema no puede eliminarse simplemente usando una reja más fina, dado que se introducirían errores a través de los términos que se truncan





- El fenómeno de propagación de los errores de aproximación para las soluciones numéricas se conoce como dispersión y difusión numérica, la cual producirá oscilaciones en la solución numérica que pueden incluso amplificarse para llevar la solución numérica a la inestabilidad

- La relación entre consistencia, convergencia y estabilidad se resume en el teorema de equivalencia de Lax: dado un BVP bien condicionado y un esquema de diferencias finitas consistente, la

estabilidad de este esquema es una condición suficiente para la convergencia de la solución

- A partir de estas nociones básicas sobre los métodos de diferencias finitas, se pueden considerar métodos de resolución para cada una de las PDEs más importantes. Primero se considera el caso de la ecuación del calor o de difusión

  - En el BVP general para la ecuación del calor o difusión, se intenta obtener una función desconocida $u(x,t)$ que sea solución al siguiente problema:

$$\frac{\partial u}{\partial t} = c\frac{\partial^2 u}{\partial x^2} + f(x,t) \quad where \ \ x \in (0,1) \ \ \& \ \ t > 0$$

$$\begin{cases} u(0,t) = g_0(t) \\ u(1,t) = g_1(t) \\ u(x,t) = u_0(x) \end{cases} \quad where \ \ \ x \in [0,1] \ \& \ t \geq 0$$

    - En este caso, g_0 y g_1 son los datos de frontera conocidos, f es un término fuente o *source term* conocido y u_0 es el dato inicial. Es posible tener condiciones de frontera más complejas y generales, y se puede generalizar la ecuación, pero esta es la forma estándar

    - El parámetro c es positivo y se conoce como el término de difusividad térmica del material que se estudia. Si c es grande, significa que la difusión es más rápida, mientras que si c es menor, significa que la difusión es más lenta

    - Se estudian dos tipos de métodos numéricos básicos para resolver este tipo de problemas: los métodos explícitos y los métodos implícitos

  - El método explícito se construye usando una aproximación por diferencias directa para las derivadas en la PDE

    - Se comienza por definir una reja o *grid* en la variable espacial x, asumiendo un espaciado uniforme por simplicidad. No obstante, para la variable temporal no hay una cota superior: solo se escoge un intervalo equiespaciado Δt

$$\Delta x = \frac{1}{N} \ \Rightarrow \ x_j = j\Delta x \ \ for \ j = 0, \dots, N \ \ where \ \begin{cases} x_0 = 0 \\ x_N = 1 \end{cases}$$

$$t_n = n\Delta t \ \ for \ n \geq 0$$

    - En cada punto en la reja espacio-temporal se puede definir una aproximación por diferencias de las derivadas para obtener el siguiente resultado:

$$\frac{\partial u}{\partial t} \approx \frac{u_j^{n+1} - u_j^n}{\Delta t} \qquad \frac{\partial^2 u}{\partial x^2} \approx \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}$$

- La PDE de interés, por tanto, se puede aproximar con estas aproximaciones en diferencias finitas y obtener una ecuación recursiva:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = c\left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}\right) + f(x_j, t_n)$$

$$\Rightarrow \; u_j^{n+1} = \frac{c\Delta t}{(\Delta x)^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + \Delta t \cdot f(x_j, t_n)$$

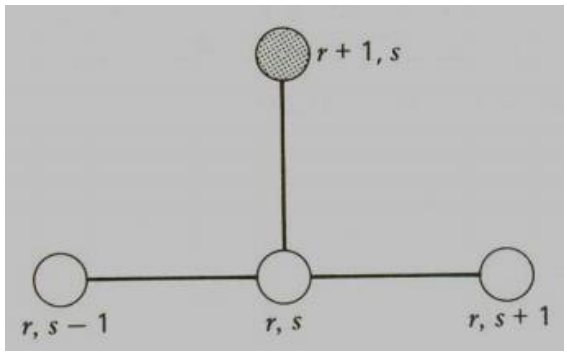
- A esto se le llama método explícito, dado que permite obtener el valor de la aproximación en t_{n+1} explícitamente en términos de la solución aproximada en t_n. Si $\rho = c\Delta t/(\Delta x)^2$, entonces se puede plantear el siguiente problema numérico:

$$u_j^{n+1} = \rho\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) + \Delta t \cdot f(x_j, t_n)$$

$$\begin{cases} u_M^{n+1} = g_0[(n+1)\Delta t] & for \;\; n \geq 0 \\ u_0^{n+1} = g_1[(n+1)\Delta t] & for \;\; n \geq 0 \\ u_j^0 = u_0(0, j\Delta x) & for \;\; j = 0, \dots, N \end{cases}$$

- Los valores u_0^0 y u_N^n no se pueden calcular usando la expresión explícita, sino que se obtienen a partir de las condiciones de frontera

- La molécula del método numérico que se usa es la siguiente:



- El error de la aproximación se puede calcular a través de las expansiones de Taylor de las aproximaciones (para las cuales se ha truncado los términos que no interesan):

$$\begin{cases} \dfrac{\partial u}{\partial t} = \dfrac{u_j^{n+1} - u_j^n}{\Delta t} - \dfrac{1}{2}\dfrac{\partial^2 u}{\partial t^2}\Delta t + O[(\Delta x)^2] \\ \dfrac{\partial^2 u}{\partial x^2} = \dfrac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} + \dfrac{1}{12}\dfrac{\partial^4 u}{\partial x^4}(\Delta x)^2 + O[(\Delta x)^3] \end{cases}$$

$$\Rightarrow \frac{u_j^{n+1} - u_j^n}{\Delta t} - c\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2} = \frac{1}{2}\frac{\partial^2 u}{\partial t^2}\Delta t + \frac{c}{12}\frac{\partial^4 u}{\partial x^4}(\Delta x)^2 + \cdots$$

$$\Rightarrow \ T_E(x,t) = \frac{1}{2}\frac{\partial^2 u}{\partial t^2}\Delta t + \frac{c}{12}\frac{\partial^4 u}{\partial x^4}(\Delta x)^2 + \cdots$$

- Los términos de menor orden constituyen los términos principales del error de truncamiento. Si las derivadas tienen una cota superior, entonces se puede obtener la siguiente desigualdad:

$$\frac{\partial^2 u}{\partial t^2} \le M_{tt} \ \& \ \frac{\partial^4 u}{\partial x^4} \le M_{xxxx}$$

$$\Rightarrow \ |T_E(x,t)| \le \frac{1}{2}M_{tt}\Delta t + \frac{1}{12}M_{xxxx}(\Delta x)^2$$

$$\Rightarrow \ |T_E(x,t)| \le \frac{1}{2}\Delta t\left(M_{tt} + \frac{1}{6\rho}M_{xxxx}\right)$$

- Debido a esta expresión, se puede ver como el error de truncación tiende a cero, independientemente de la relación entre el tamaño relativo de dos rejas consecutivas. Por lo tanto, el esquema de diferencias explícito es incondicionalmente consistente con la PDE

$$\lim_{\substack{\Delta t \to 0 \\ \Delta x \to 0}} T_E(x,t) = 0 \quad when \ \ x \in [0,1] \ \& \ t \in \left[0, t_f\right]$$

- Como $|T_E(x,t)|$ depende de Δt, entonces es $O(\Delta t)$ y se puede decir que es consistente hasta el primer orden o que el esquema tiene precisión de primer orden

- Aunque los resultados anteriores sean ciertos para el error de truncamiento, pueden existir problemas prácticos de convergencia, en donde el paso del tiempo Δt es demasiado grande para la velocidad de difusión c y el paso espacial Δx (expresados en ρ)

  - Esto significa que el método explícito requiere que el paso del tiempo sea lo suficientemente pequeño con tal de que el cálculo sea estable y exacto. Para ello, se puede plantear un resultado que da una condición para la exactitud y la estabilidad del método

- Siendo $u \in C^{4,2}(Q)$ para $Q = \{(x,t)|0 \leq x \leq 1{,}0 < t \leq T\}$ para alguna $T > 0$, si Δx y Δt son tales que $\Delta t \leq (\Delta x)^2/2c$, entonces existe una constante $A > 0$ independiente de Δx y Δt tales que se obtiene la siguiente desigualdad:

$$\max\left|u(x_j,t_n) - u_{\Delta x,\Delta t}(x_j,t_n)\right| \leq \max\left|u_0(x_j) - u_{\Delta x,\Delta t}(x_j,0)\right| + A(\Delta x)^2 M$$

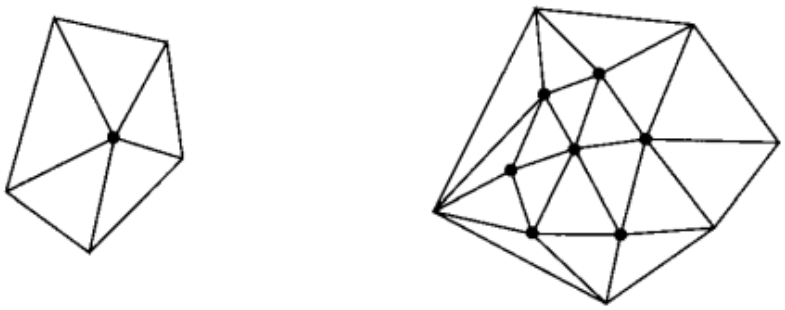
$$where\ M = \max_{(x,t)\in Q}\left\{\left|\frac{\partial^2 u}{\partial t^2}\right|, \left|\frac{\partial^4 u}{\partial x^4}\right|\right\}$$

- Si la condición $\Delta t \leq (\Delta x)^2/2c$ no se cumple, entonces se obtiene el siguiente resultado para $c = 4\lambda - 1 > 1$

Los métodos numéricos para PDEs: elementos finitos

- Una vez vistos los elementos teóricos necesarios que motivan el método, se puede introducir las ideas y elementos básicos del método de elementos finitos para la resolución de PDEs

  - La idea del método de los elementos finitos es dividir el dominio en piezas simples (polígonos) y aproximar la solución con funciones extremadamente simples de estas piezas (donde N es el número de puntos del dominio en el que la solución se construye explícitamente)



$N = 1$ $N = 7$

    - Este método fue creado por ingenieros con tal de manipular o gestionar dominios con formas curvas o irregulares. Por ejemplo, si el dominio es circular, entonces se tiene problemas de eficiencia con las diferencias finitas debido a que no se puede partir un círculo en rectángulos de manera precisa

    - El método está intrínsecamente conectado con los productos interiores y con la proyección L_2 vista anteriormente, dado que el objetivo final es poder obtener una aproximación de la PDE a

través de considerar una proyección con un sistema de ecuaciones lineales usando el producto interior con integrales

- La definición formal de un elemento finito consiste en el triple $(K, \mathcal{P}, \{L_i\}_{i=1}^n)$, de modo que $K \subset \mathbb{R}^d$ es un polígono d-dimensional, $\mathcal{P}$ es un espacio de funciones polinómicas en K y $\{L_i\}_{i=1}^n$ es el conjunto de funcionales lineales $L(\cdot)$ que definen los grados de libertad (con $n = \dim(P)$)

  - El polígono K es de un tipo diferente dependiendo de la dimensión del espacio (si $d = 1,2,3$). Los tipos más comunes de polígonos usados son las líneas, los triángulos, los cuadriláteros, los tetraedros y los ladrillos (usando ocasionalmente prismas)

  - Todo polígono nace de un *mesh* $\mathcal{K} = \{K\}$ del dominio computacional D. Las rejas triangulares y tetraédricas permiten representar fácilmente dominios con fronteras curvas, mientras que las rejas cuadrilaterales y de ladrillos son más fáciles de implementar en las computadoras

- Equipando P con una base $\{S_j\}_{j=0}^n$ de funciones base S_j que se conocen como funciones de forma, se dice que un elemento finito es unisolvente si los funcionales L_i pueden determinar únicamente las funciones de forma

  - La unisolvencia puede interpretarse como una condición de compatibilidad necesaria para $L(\cdot)$, $\mathcal{P}$ y K. Este requerimiento garantiza que el problema de encontrar las funciones de forma está bien planteado, de modo que existe una única solución S_j para las ecuaciones que definen la función de forma y los funcionales proporcionan la información suficiente para determinar estas funciones

  - Por definición, esto es equivalente a que la solución de $L_i(v) = 0$ para $v \in \mathcal{P}$ sea $v = 0$ para toda i. Esto significa que los funcionales L_i no son redundantes ni inconsistentes, y proporcionan una descripción completa del espacio de las funciones de forma

  - Aunque puede ser difícil establecer unisolvencia, el cálculo de las funciones de forma es fácil, dado que solo consiste en resolver un sistema lineal. Las funciones de forma S_j están determinadas a partir del sistema de n ecuaciones lineales siguiente, que permite identificar cada funcional L_i con una función de forma S_j:

$$L_i(S_j) = \delta_{ij}, \qquad i, j = 1,2, \dots, n$$

$$where \quad \delta_{ij} = \begin{cases} 1 & if\ i = j \\ 0 & if\ i \neq j \end{cases}$$

- Tomando una combinación lineal de las funciones de forma y los coeficientes, se tiene un polinomio o una función de elemento finito en $\mathcal{P}$ en cada polígono K, de modo que las funciones de forma son como bloques básicos para formar los polinomios necesarios para cada pieza de la reja o *mesh*

$$v = \sum_{j=1}^{n} c_j S_j \in \mathcal{P}$$

- Aparte de especificar las funciones de forma en cada polígono K, los funcionales también especifican el comportamiento de estas funciones entre polígonos adyacentes

- Por ejemplo, si se quiere funciones de elementos finitos que son continuas en la reja $\mathcal{K}$, entonces se debe tener cuidado especificando los funcionales, de modo que las funciones de forma resultantes sean continuas, sobre todo en la frontera del polígono ∂K. Los funcionales determinan tanto las propiedades locales como las globales del espacio de elementos finitos V_h

- La elección particular de los funcionales $L_i(\cdot)$ permiten dar pie a familias de elementos finitos que comparten propiedades similares, aunque puedan ser diferentes en otros aspectos. La familia de Lagrange es la más popular

  - Definiendo los funcionales como $L_i(v) = v(N_i)$ para $i = 1,2,\dots,n$, donde N_i son puntos de nodos seleccionados, los funcionales son los más simples, en el sentido de que cada uno consiste en una evaluación en el punto de v. Como cada función está asociada a un nodo particular, el conjunto de funciones de forma se conoce como base nodal

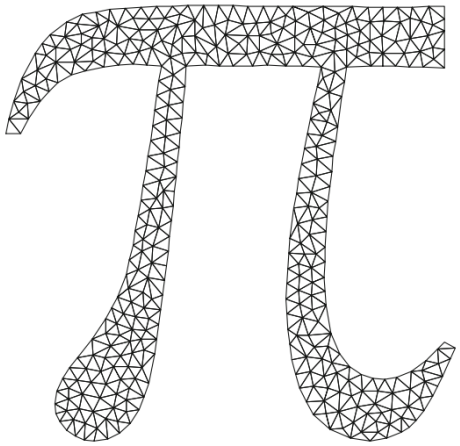
  - Para $d = 2$ con $\mathcal{P} = \mathcal{P}_1(K)$ en un triángulo K, los puntos nodales son los vértices del triángulo y las funciones S_j para $j = 1,2,3$ son las funciones *hat* familiares

  - Aunque estos elementos son continuos, tienen derivadas discontinuas a través de las fronteras de los elementos. Por lo tanto, son elementos continuos C^0, que sirven para aproximar funciones en el espacio de Hilbert H^1

- Para poder motivar el método de elementos finitos, se extiende la aproximación polinómica por piezas a más de una dimensión. Para ello,

se introduce el método de triangulación, que es un polígono muy usado cuando $d = 2$

- Siendo $D \in \mathbb{R}^2$ un dominio con una frontera o contorno ∂D suave o poligonal, una triangulación o *mesh* $\mathcal{K}$ de D es un conjunto $\{K\}$ de triángulos tales que $D = \bigcup_{K \in \mathcal{K}} K$ y que la intersección de dos triángulos sea un eje, un vértice o que esté vacía



- No se permite que ningún vértice de ningún triángulo esté colgando, de modo que no se permite que ningún punto en el extremo de un eje no sea vértice de los triángulos adyacentes. A los vértices de los triángulos se les llama nodos

- El conjunto de ejes E de los triángulos se denota por $\mathcal{E} = \{E\}$, y se distingue entre ejes que están dentro del dominio D, denotando su conjunto por $\mathcal{E}_I$, y aquellos que son parte de la frontera o contorno ∂D, denotando su conjunto por $\mathcal{E}_B$

- Para poder medir el tamaño de un triángulo K se introduce el tamaño local del *mesh* h_K, definido como la longitud del eje más largo en K. Además, para medir la calidad de K, se concibe un parámetro c_K llamado el parámetro de fragmentación o *chunkiness* c_K el cual es la *ratio* entre h_K y el diámetro d_K del círculo inscrito dentro del triángulo

$$h_K = \max\{E : E \in K, E \in \mathcal{E}\}$$

$$c_K = h_K / d_K$$

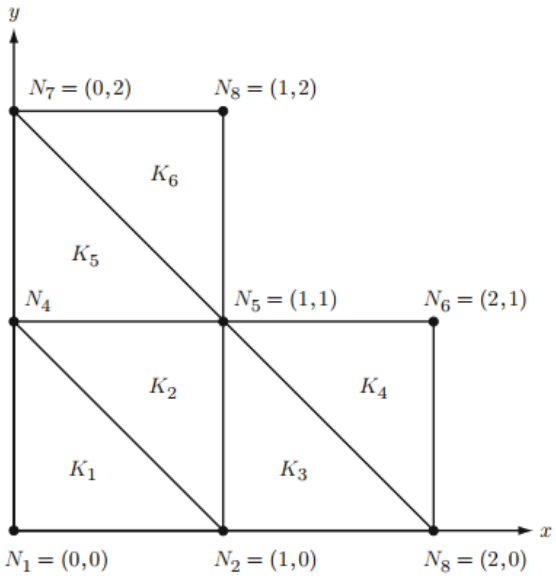
- Aparte del tamaño del *mesh* h_K, se puede introducir una medida del tamaño global considerando $h = \max_{K \in \mathcal{K}} h_K$

- Se dice que una triangulación $\mathcal{K}$ es regular en forma si existe una constante $c_0 > 0$ tal que $c_K \geq c_0$ para $\forall K \in \mathcal{K}$. Esta condición

significa que la forma de los triángulos no puede ser muy extrema (ángulos ni muy agudos ni muy obtusos), y siempre se asume que las triangulaciones con las que se trabaja son así (tiene consecuencias para las propiedades de aproximación)

- A veces, uno se referirá a un *mesh* casi-uniforme, que significa que h_K es casi lo mismo para todos los elementos de la triangulación. En particular, existe una constante $\rho > 0$ tal que $\rho < h_K/h_{K'} < \rho^{-1}$ para dos elementos K y K' cualesquiera

- La manera estándar de representar el *mesh* triangular con n_p nodos y n_t elementos o triángulos en la computadora es guardarlo como dos matrices P y T, llamadas matriz de puntos y matriz de conectividad respectivamente



$$P = \begin{bmatrix} 0.0 & 1.0 & 2.0 & 0.0 & 1.0 & 2.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 1.0 & 1.0 & 2.0 & 2.0 \end{bmatrix}, \qquad T = \begin{bmatrix} 1 & 2 & 5 & 3 & 4 & 5 \\ 2 & 5 & 2 & 6 & 5 & 8 \\ 4 & 4 & 8 & 5 & 7 & 7 \end{bmatrix}$$

- La matriz de puntos P es de tamaño $2 \times n_p$ y la columna j representa las coordenadas de cada variable para un nodo N_j. En cambio, la matriz de puntos P es de tamaño $3 \times n_p$ y la columna j representa el número de cada uno de los tres nodos en el triángulo K_j

- Se adopta la convención de que los nodos N_j están numerados de izquierda a derecha y de abajo a arriba, y los triángulos K_j también

- Se remarca que la representación del *mesh* a través de estas matrices es muy común y generaliza a casi cualquier tipo de elemento y dimensión (con tamaños $d \times n_p$ y $(d+1) \times n_t$)

- A partir de la representación de poligonal a través de la triangulación, es posible obtener la proyección L_2 para el caso bidimensional

  - Siendo K un triángulo y $\mathcal{P}_1(K)$ el espacio de funciones lineales en K, cualquier polinomio $v \in \mathcal{P}_1(K)$ se determina únicamente por sus valores nodales $\alpha_i = v(N_i)$ para $i = 1,2,3$. Haciendo esto, se obtiene el siguiente sistema lineal:

$$\begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

    - Al calcular el determinante de este sistema se obtiene que su valor absoluto es $2|K|$, donde $|K|$ es el área de K, lo que significa que el sistema tiene una solución única mientras que K no sea un triángulo degenerado

    - La base natural $\{1, x_1, x_2\}$ para $\mathcal{P}_1(K)$ no es adecuada, dado que se quiere usar los valores nodales como grados de libertad. Por lo tanto, se introduce la siguiente base nodal:

$$\lambda_j(N_i) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \quad \Rightarrow \quad v = \alpha_1\lambda_1 + \alpha_2\lambda_2 + \alpha_3\lambda_3$$

- Una vez se ha entendido la extensión bidimensional de la proyección L_2, entonces esta se puede usar para motivar la aproximación de la PDE bidimensional a través de la proyección

  - Para poder introducir las ideas del método del elemento finito, se plantea un BVP con una PDE de segundo grado no homogénea con condiciones de Dirichlet:

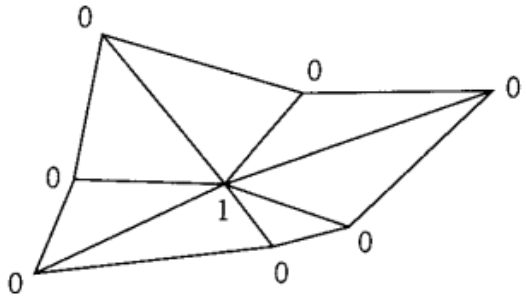
$$-\nabla \cdot (a\nabla u) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in D \quad where \;\; D \in \mathbb{R}^2$$

$$-\boldsymbol{n} \cdot a(\nabla u) = \kappa(u - g_D) - g_N \quad in \;\; \partial D$$

    - En este caso, la notación ∂D es el contorno o la frontera del dominio D para poder fijar las condiciones sobre esta frontera o contorno

    - El dominio D se triangula, de modo que se aproxima la región D_n con una unión de un número finito de triángulos, y se denotan los vértices interiores por $V_1, \dots, V_n$

- Después, se escogen N funciones de prueba $v_1(\boldsymbol{x})$, $v_2(\boldsymbol{x})$, …, $v_n(\boldsymbol{x})$, uno para cada vértice o punto interior. Cada función de prueba $v_i(\boldsymbol{x})$ se escoge para ser igual a 1 en su vértice V_i e igual a cero para todos los otros vértices



- Dentro de cada triángulo, cada función de prueba es una función lineal $v_i(\boldsymbol{x}) = a + bx_1 + cx_2$ (los coeficientes son diferentes para cada función de prueba y cada triángulo). Esta prescripción determina $v_i(\boldsymbol{x})$ únicamente

- Se quiere aproximar la solución $u(\boldsymbol{x})$ por una combinación lineal de funciones $v_i(\boldsymbol{x})$, y el objetivo es encontrar la manera de escoger los coeficientes $\xi_1, \xi_2, \dots, \xi_N$:

$$u(\boldsymbol{x}) = \sum_{k=1}^{N} \xi_k v_k(\boldsymbol{x}) = \xi_1 v_1(\boldsymbol{x}) + \cdots + \xi_N v_N(\boldsymbol{x})$$

- Para poder motivar la elección de los coeficientes, se tienen que usar resultados del cálculo multivariante para obtener la forma variacional del problema

  - En este caso, el teorema más importante es el de Green, pero en su forma vectorial con el vector normal $\boldsymbol{n}$ de la curva paramétrica que engloba el dominio:

$$\oint \boldsymbol{F} \cdot \boldsymbol{n}\, ds = \int\int_D \operatorname{div} \boldsymbol{F}(x,y)\, d\boldsymbol{S} \;\Rightarrow\; \int_{\partial D} \boldsymbol{F} \cdot \boldsymbol{n}\, ds = \int_D \operatorname{div} \boldsymbol{F}\, d\boldsymbol{S}$$

  - A partir de esta igualdad, si se consideran uno de los componentes, denotado por k, entonces se puede obtener la siguiente igualdad:

$$\boldsymbol{F}(x_1 \dots, x_n) = [f_1(x_1 \dots, x_n), \dots, f_n(x_1 \dots, x_n)] \;\Rightarrow\; f_k$$

$$\operatorname{div} \boldsymbol{F}(x_1 \dots, x_n) = \left[\frac{\partial f_1}{\partial x_1}, \dots, \frac{\partial f_n}{\partial x_n}\right] \;\Rightarrow\; \frac{\partial f_k}{\partial x_k}$$

$$\Rightarrow \int_{\partial D} f_k n_k \, ds = \int_D \frac{\partial f_k}{\partial x_k} \, dx_k \quad for \;\; k = 1,2, \dots, n$$

- Cambiando f_k por $f_k v$ (el producto de un componente del campo vectorial y la función de prueba v), entonces se puede usar la integración por partes para obtener el siguiente resultado:

$$\int_D \frac{\partial f_k}{\partial x_k} v \, dx_k = -\int_D f_k \frac{\partial v}{\partial x_k} \, dx_k + \int_{\partial D} f_k v n_k \, ds$$

- Si ahora se vuelven a considerar todos los componentes de los vectores y fijando $\boldsymbol{F} = -\nabla u$, se obtiene la fórmula de Green:

$$\int_D (\nabla \cdot \boldsymbol{F}) v \, d\boldsymbol{x} = -\int_D \boldsymbol{F} (\nabla \cdot v) \, d\boldsymbol{x} + \int_{\partial D} (\boldsymbol{F} \cdot \boldsymbol{n}) v \, ds$$

$$\Rightarrow \int_D -\nabla \cdot (\nabla u) v \, d\boldsymbol{x} = \int_D \nabla u \cdot \nabla v \, d\boldsymbol{x} - \int_{\partial D} \boldsymbol{n} \cdot (\nabla u) v \, ds$$

- Volviendo al problema con la ecuación de Poisson bidimensional, se pueden usar los resultados anteriores con tal de obtener una proyección o una aproximación para la PDE

  - A través del teorema de Green en forma vectorial derivado anteriormente se puede obtener la forma variacional de la ecuación de Poisson y obtener el siguiente resultado

$$\int_D f v \, d\boldsymbol{x} = \int_D -\nabla \cdot (a \nabla u) v \, d\boldsymbol{x}$$

$$\Rightarrow \int_D f v \, d\boldsymbol{x} = \int_D a \nabla u (\nabla \cdot v) \, d\boldsymbol{x} - \int_{\partial D} \boldsymbol{n} \cdot (a \nabla u) v \, ds$$

$$\Rightarrow \int_D f v \, d\boldsymbol{x} = \int_D a \nabla u \cdot \nabla v \, d\boldsymbol{x} + \int_{\partial D} [\kappa (u - g_D) - g_N] v \, ds$$

$$for \;\; \forall v \in V$$