MÉTODOS NUMÉRICOS

Víctor Jiménez López y Antonio Pallarés Ruiz

Capítulo 1

Números y Errores

Interrogantes centrales del capítulo

- Conocer la presencia de errores en cualquier procedimiento de cálculo numérico.
- Conocer cómo se representan los números en una «máquina» y los errores de redondeo que conlleva esta representación.
- Aprender a decidir entre uno y otro método numérico para resolver un mismo problema atendiendo a la «estabilidad» del método y al «condicionamiento» del problema.

Destrezas a adquirir en el capítulo

- Calcular utilizando aritméticas de punto fijo y flotante con un número pequeño de dígitos midiendo los errores absolutos y relativos que produce el redondeo.
- Encontrar alternativas «estables» a cálculos inestables.
- Medir «números de condición» de procesos sencillos.

En esta unidad vamos a analizar los distintos tipos de errores que se producen en los cálculos numéricos. Presentaremos los sistemas de numeración y la aritmética que utilizan las «máquinas». Mediremos como se propagan los errores de redondeo. Y, por último, estudiaremos los conceptos de «estabilidad» de un algoritmo y «condicionamiento» de un problema.

Desarrollo de los contenidos fundamentales

- Números y su representación.
- Errores.
- Números de máquina. Redondeo.
- Cálculos estables e inestables.
- Problemas mal condicionados. Número de condición.
- Prácticas I y II.

Temporalización: 6HTe + 2HPb + 6HLab = 14HPres

1.1. Introducción

Ejemplo 1.1.1 Probad a realizar la siguiente operación

$$\left(\sqrt{\pi}\right)^2.$$

Si hacemos *cálculo simbólico*, la misma definición de raíz cuadrada nos dice que $\left(\sqrt{\pi}\right)^2 = \pi$.

Pero si hacemos los cálculos con las representaciones numéricas de π en una calculadora, la realidad puede que sea distinta, y el resultado de la operación no coincida exactamente con la representación de π en nuestra calculadora.

¿Qué ocurre con el resultado en tu calculadora?

Ejemplo 1.1.2 Probad a realizar la siguientes operaciones en vuestras calculadoras utilizando las representaciones numéricas de las soluciones

(I) $2 \div 3$

(II) $\texttt{Ans} - 0.6666666667$

¿Qué ocurre con el resultado en tu calculadora?

Prueba ahora con

(I) $2 \div 3$

(II) $\texttt{Ans} - 0.666666666666666$

¿Qué ocurre ahora con el resultado en tu calculadora?

Ejemplo 1.1.3 Probad la siguientes operaciones en las calculadoras

(I) $10^{50} * 10^{40}$

(II) $10^{50} * 10^{50}/3$

(III) $10^{50} * (10^{50}/3)$

(IV) $(10^{19}+1)-10^{19}$

¿Qué ocurre con los resultados en tu calculadora?

Cualquier procedimiento numérico debe considerar el control de errores para medir cómo afectan al resultado

Es muy importante que en cada problema numérico hagamos un seguimiento de los errores cometidos con el fin de poder estimar el grado de aproximación de la solución que proporciona.

1.1.1. Fuentes del Error

Hay diferentes fuentes/tipos de error que pueden afectar a la solución numérica de un problema:

- Errores en los datos de entrada:
  (I) Errores en la medición de los datos iniciales, que pueden ser aleatorios y entonces hacen necesario acudir a técnicas estadísticas o de control de calidad, y errores sistemáticos de procedimiento que hay que descubrir y subsanar.
- Errores de redondeo en los datos y en el cálculo:
  (I) Los producidos al introducir los datos en las máquinas de cálculo. En ellas los números tienen una representación finita. Estos errores son INEVITABLES.
  (II) Los producidos al hacer cálculos en las máquinas. Otra vez chocamos con la finitud de la representación de los números. Estos errores también son INEVITABLES, aunque son PREVISIBLES y a veces, es posible esquivarlos sustituyendo los cálculos por otros equivalentes (ya iremos viendo cómo).
- Errores de truncamiento en el método:
  Como la resolución numérica de un problema consiste en obtener una aproximación numérica de la solución exacta. Siempre debemos asumir este error de aproximación, que recibe el nombre de error de truncamiento.
- Errores de inestabilidad y control.
  Se trata de analizar cómo afectan los errores en los datos de entrada o en los cálculos a la solución final. Si un método permite que pequeños errores en los datos de entrada o en los cálculos se traduzcan en errores grandes de la solución se dice que el método es inestable.
  En esta unidad analizaremos algunos cálculos que pueden ser causa de inestabilidad y propondremos cálculos alternativos.
  En algunas ocasiones, la inestabilidad de un método no proviene de los cálculos sino que es intrínseca al método, es decir al modelo empleado para resolver un problema. En estos casos se dice que el método, o el problema, está mal condicionado. Para evitar esta situación es necesario analizar el problema real e intentar sustituir el modelo o el método numérico empleado.
  También en la unidad, veremos como cuantificar (medir con números) el condicionamiento de un problema para poder predecir métodos mal condicionados y proponer su sustitución por otros.

1.1.2. Ejemplos

Ejemplo 1.1.4 Utilizando la calculadora[1] vamos a evaluar las expresiones de la igualdad

$$18817 - 10864\sqrt{3} = \frac{1}{18817 + 10864\sqrt{3}}$$

Si se quiere comprobar la igualdad anterior bastará recordar la identidad

$$a^2 - b^2 = (a - b)(a + b)$$

y poner $a = 18817$ y $b = 10864\sqrt{3}$ y constatar que $a^2 - b^2 = 1$.

Si hacemos los cálculos, nuestra calculadora proporciona los siguientes valores:

$$18817 - 10864\sqrt{3} = 2.65718\ 10^{-05}$$
$$\frac{1}{18817 + 10864\sqrt{3}} = 2.657171708\ 10^{-05}$$

Hemos obtenidos dos valores distintos, lo que nos plantea algunas cuestiones:

- ¿Cúal es la causa por la que la calculadora no respeta la igualdad?

  Las calculadoras están trucadas para fastidiarnos ☐

  Se han producido ERRORES ☐
- Si repitiéramos las operaciones con la calculadora, ¿obtendriamos los mismos valores?

  Si ☐ No ☐
- Si los resultados son aproximados ¿Cúal de los dos valores se aproxima más al verdadero valor de la expresión?

  Al finalizar la unidad debéis tener la respuesta.

Ejemplo 1.1.5 Consideremos el sistema de ecuaciones

$$\begin{cases} x + y = 2 \\ x + 1.00001y = 2.00001. \end{cases}$$

Su solución, es $x = 1$ e $y = 1$.

Consideremos ahora el sistema perturbando un poco los términos independientes.

$$\begin{cases} x + y = 2 \\ x + 1.00001y = 2. \end{cases}$$

Este sistema perturbado tiene como solución $x = 2$ e $y = 0$.

- Un error pequeño en los datos iniciales (en el término independiente) ha producido un error grande en la solución.
- En este caso, no se ha producido ningún error de cálculo.
- Este es un ejemplo de un problema (sistema lineal) **mal condicionado**. No podemos evitar que pequeños errores en los datos iniciales se conviertan en errores grandes en las soluciones.
- ¿Cómo afrontamos situaciones en las que aparezcan sistemas mal condicionados?

[1] Casio fx-991ES

1.2. Números y su representación

A la vista del ejemplo 1.1.4 , podemos hacer una primera afirmación: La aritmética con la que calcula un ordenador o una calculadora no es la que hemos aprendido en el colegio y en los cursos de álgebra (grupos, anillos, cuerpos, álgebras, etc...). Con nuestra calculadora hemos constatado que $(a-b)$ puede ser distinto de $(a^2-b^2)/(a+b)$.

La mayoría de los procesadores que realizan los cálculos en un ordenador trabajan con números que se escriben en binario (con 0 y 1) en registros de 8, 16, 32 o 64 dígitos (bits). Nosotros mismos, al representar los números en el sistema decimal (con las cifras 0, 1,...,9) utilizamos secuencias finitas de dígitos. Es decir, a la hora de trabajar con representaciones de «números» en una «máquina» sólo lo podemos hacer con una cantidad finita de ellos. Vamos a poder trabajar con exactitud cuando trabajemos con números enteros que puedan ser representados en la máquina, y también (cuando el procesador esté especialmente programado[2]) con algunos números racionales. Pero, a la hora de representar números reales cómo números de «máquina» siempre tendremos que utilizar «aproximaciones» .

1.2.1. Números enteros

1.2.1.1. *Representación de enteros en un sistema de base* b

Para representar enteros, usualmente utilizamos el sistema de numeración decimal (en base 10) que es un sistema posicional en el que cada número viene expresado por una sucesión finita de dígitos (cifras del 0 al 9) cuyos valores dependen de la posición que ocupan

Pon ejemplo $12879 = 9\ 10^0 + 7\ 10^1 + 8\ 10^2 + 2\ 10^3 + 1\ 10^4$,

Proposición 1.2.1 *Elegida una base natural $b \in \mathbb{N}$, $b \geq 2$, cualquier número entero positivo p admite una única representación en la forma*

$$p = a_n b^n + a_{n-1} b^{n-1} + \cdots + a_1 b^1 + a_0$$

con $a_j \in \mathbb{N}$, $0 \leq a_j \leq (b-1)$ y $a_n > 0$.

Se utiliza la representación:

$$p = a_n a_{n-1} \ldots a_1 a_{0(b)}$$

y se llama ***representación digital de*** p ***en base*** b.

Los coeficientes a_j se denominan ***dígitos*** *o cifras.*

¿Cuál es el algoritmo[3] para encontrar la representación en base b de un número entero? Ese algoritmo probará la proposición anterior.

Ejercicio 1.2.1 Describe los algoritmos para pasar de la representación en base 2 (binario) de un número entero a sus representaciones en las bases 4 y 8 (octal).

Describe también los reciprocos que pasen de las representaciones en las bases 4 y 8 a binario.

[2] Este es el caso para algunos números en la calculadora Casio fx-991ES. Revisad el manual.

[3] algoritmo: «Sucesión finita de instrucciones claras y precisas», especificando cuales son los **datos de entrada**, el **flujo** (la sucesión) de instrucciones, las **condiciones de parada** y los **datos de salida**.

1.2.1.2. *Tipos de enteros de «máquina»*

Como hemos dicho la mayoría de los procesadores que realizan los cálculos en un ordenador trabajan con números que se escriben en **binario** (base 2) utilizando registros de 8, 16, 32 o 64 dígitos (bits). En ellos se utiliza el primer dígito para señalar el signo del número.

- (`byte`) 8 bits
- (`short`) 16 bits
- (`int`) 32 bits
- (`large`) 64 bits

¿Cuáles son el mayor y el menor entero (`int`) de una «máquina»?

¿Qué errores puede producir la aritmética de la máquina al trabajar con enteros (`int`)?

Resp: Los cálculos con números enteros se pueden hacer de forma exacta. El único error que podemos tener al operar con ellos es que el resultado no quepa en los registros utilizados para guardarlo. En estos casos obtendremos **Errores por Desbordamiento (`Overflow`).**

Por cierto, ¿cómo se suman o se multiplican dos números en base 2?, por supuesto que con las tablas de sumar o multiplicar correspondientes, ¡descríbelas!

Ejercicio 1.2.2 Si tu calculadora tiene la función de representar números en distintas bases, encuentra los mayores enteros que se pueden escribir en hexadecimal (base 16), binario (base 2) u octal (base 8).

1.2.2. Números reales

También decíamos en la introducción que cuando el procesador esté especialmente programado, podemos representar algunos números racionales como pares de números enteros. Pero desde el punto de vista numérico, esto no es muy operativo: números racionales acotados pueden venir representados mediante fracciones de números enteros muy grandes.

Así, que para representar números racionales o reales en una «máquina» siempre tendremos utilizar «aproximaciones», escritas en base 2 y almacenadas en unos registros de tamaño prefijado.

Aquí el hecho de fijar el tamaño de los registros que han de ocupar los números en la máquina, nos anuncia la aparición de errores de aproximación o redondeo, errores que no podremos evitar. Por esta razón debemos aprender a controlarlos en el sentido de prever su efecto en un proceso numérico conociendo como se transmiten en los distintos pasos del mismo.

1.2.2.1. *Sistemas de representación de punto fijo*

Proposición 1.2.2 *Elegida una base natural $b \in \mathbb{N}$, $b \geq 2$, cualquier número real positivo x admite una representación en la forma*

$$x = a_n b^n + \cdots + a_1 b^1 + a_0 + a_{-1}\frac{1}{b} + a_{-2}\frac{1}{b^2} + \cdots + a_{-n}\frac{1}{b^n} + \ldots$$

con $a_j \in \mathbb{N}$, $0 \leq a_j \leq (b-1)$ *y* $a_n > 0$.

Se utiliza la representación:

$$x = a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-n} \dots_{(b)}$$

y se llama ***representación digital de punto fijo de*** x ***en base*** b.

¿Cuál es el algoritmo para encontrar una representación de punto fijo en base b de un número real?

¿Es única esta representación?

Resp: Si excepto para números racionales de la forma $x = \frac{k}{b^n}$.

Ejemplo 1.2.3

(I) $\frac{17}{3} = 5.6666\dots_{(10)}$

(II) $1_{(10)} = .99999\dots_{(10)} = .111111\dots_{(2)}$

(III) $.1_{(10)} = .0001\,1001\,1001\dots_{(2)}$

En base 10 el número .1 es decimal exacto (tiene una cantidad finita de dígitos no nulos) mientras que en base 2 ese mismo número tiene una representación digital "periódica mixta" (con infinitos dígitos no nulos que se van repitiendo de forma periódica).

(IV) $\pi = 3.141592....$

A la hora de trabajar en una máquina, tenemos que fijar el tamaño de los registros a usar para guardar los números. Esto se traduce inmediatamente en una limitación en el número de dígitos que podemos utilizar

Ejemplo 1.2.4 Supongamos que trabajamos con una máquina que utiliza números escritos en base 10, con un máximo de 6 dígitos además de la coma.

Consideremos los números: $x = 0.00001$, $y = 0.1$, $z = 10000$.

Vamos a simular como podríamos operar con ellos utilizando la máquina.

Para multiplicar $x \otimes y$:

- Suponemos que dentro de la máquina se puede realizar el producto exacto $p = x \times y = 0.000001$.
- Como este número tiene 7 dígitos, para guardarlo en un registro de la máquina, vamos a aproximarlo por el número más próximo de entre los de la máquina, es decir, de entre los que se pueden escribir con 6 dígitos. Esta aproximación es 0.00000.
- Así el resultado debe ser $x \otimes y = 0$.

Si multiplicamos $x \otimes y$ por z, tendremos $(x \otimes y) \otimes z = 0$.

Si hacemos lo mismo con $y \otimes z$, obtenemos $y \otimes z = y \times z = 1000$. Y si multiplicamos por x tendremos $x \otimes (y \otimes z) = 0.01$.

□

$$0 = (x \otimes y) \otimes z \neq x \otimes (y \otimes z) = 0.01$$

A la hora de trabajar con números en máquinas, este sistema de representación puede no ser conveniente al operar con números con muchos ceros.

Para evitar estos problemas que representan los ceros a la izquierda o a la derecha de la coma, se utiliza el sistema de representación de coma flotante.

1.2.2.2. *Representación de números reales en coma flotante*

Proposición 1.2.5 *Elegida una base natural $b \in \mathbb{N}$, $b \geq 2$, cualquier número real $x \neq 0$ admite una representación en la forma*

$$x = \pm b^N \sum_{n=1}^{\infty} a_{-n} \frac{1}{b^n}$$

con $\pm = +$ si x es positivo, y $\pm = -$ si x es negativo, $a_j \in \mathbb{N}$, $0 \leq a_j \leq (b-1)$ y $a_{-1} > 0$.

Se utiliza la representación:

$$x = \pm .a_{-1}a_{-2}\ldots a_{-n}\ldots \ E\ N$$

y se llama ***representación digital de punto flotante de*** x ***en base*** b.

Al número N se le llama exponente de la representación de x, y al número $m = \sum_{n=1}^{\infty} a_{-n}\frac{1}{b^n} = .a_{-1}a_{-2}\ldots a_{-n}\ldots_{(b)}$ se le llama mantisa.

¿Cuál es el algoritmo para encontrar una representación de punto flotante en base b de un número real?

¿Es única esta representación?

Resp: Si, excepto para números racionales de la forma $x = \frac{k}{b^n}$. O si se añade la condición de que para cada n existe un $k > n$ tal que $a_{-k} \neq b-1$

Ejemplo 1.2.6

(I) $\frac{17}{3} = 5.6666\ldots_{(10)} = 10^{-1}.56666\ldots_{(10)} = .56666\ldots E-1_{(10)}$

(II) $1_{(10)} = 2^0.111111\ldots_{(2)} = .111111\ldots E0_{(2)}$

(III) $.1_{(10)} = .0001\,1001\,1001\ldots_{(2)} = 2^{-3}.1\,1001\,1001\ldots_{(2)} = .1\,1001\,1001\ldots E-3_{(2)}$

(IV) $\pi = 3.141592... = .314151592..._{(10)}$

Observad que a la hora de guardar las representaciones de números en coma flotante no es necesario dejar sitio para la coma (el punto) porque siempre está en la misma posición. Si trabajamos con representaciones binarias tampoco es necesario dejar ningún sitio para a_1 que forzosamente sólo puede tomar el valor 1 (es distinto de cero y menor que 2).

Volvamos al problema planteado en el ejemplo 1.2.4 y veamos que operando con los números en coma flotante no se producen los mismos errores en el cálculo.

Ejemplo 1.2.7 Supongamos que trabajamos con una máquina que utiliza números escritos en coma flotante de base 10, con un máximo de 6 dígitos en la mantisa.

Consideremos los números: $x = 0.00001 = .1E-4$, $y = .1E0$, $z = 10000 = .1E5$.

Vamos a simular como podríamos operar con ellos utilizando la máquina.

Para multiplicar $x \otimes y$: Ahora dentro de la máquina se puede realizar el producto exacto $p = x \times y = .1E-5$.

Si multiplicamos por z, tendremos $(x \otimes y) \otimes z = .1E-1$.

Si hacemos lo mismo con $y \otimes z$, obtenemos $y \otimes z = y \times z = .1E4$. Y si multiplicamos por x volvemos a obtener $x \otimes (y \otimes z) = .1E-1$.

¿Cómo se multiplican dos números en coma flotante?

(no es necesario guardar el punto)

¿Cómo se suman?

¿Cómo se representan los números en tu calculadora?

1.2.2.3. *Tipos de reales en los «ordenadores»*

En la actualidad la mayoría de los procesadores que realizan los cálculos en un ordenador trabajan con números reales que se representan en coma flotante en **binario** (base 2) utilizando registros de 32 o 64 dígitos (bits). En ellos se utiliza el primer dígito para señalar el signo del número.

- (`float`) 32 bits
- (`double`) 64 bits

El "Institute for Electrical and Electronic Engineers´´ estableció en 1985 un protocolo de representación de números reales en los ordenadores que vienen observando desde entonces los fabricantes de los procesadores donde los ordenadores realizan las operaciones.

En el libro de Burden y Faires [2] encontraréis una breve descripción de este protocolo, conocido como el IEEE754-1985. ¡No pienso copiarla aquí!

¿Cuál es el menor real (`double`) positivo en un ordenador?

¿Cuál es el mayor real (`double`) positivo en un ordenador?

Ejercicio 1.2.3 Haz los ejercicios 1, 2 y 4 de la Hoja de problemas 1.

1.3. Errores

Definición 1.3.1 *Si el número real p^* es una aproximación del número p, para medir el tamaño del error cometido en esta aproximación podemos utilizar:*

- *Error exacto en la aproximación* $e = p - p^*$.
- *Error absoluto:* $e_a = |e| = |p - p^*|$.
- *Errores relativos (si* $p \neq 0$*):* $e_r = \frac{e_a}{|p|} = \frac{|p-p^*|}{|p|}$ *y* $\widetilde{e_r} = \frac{e_a}{|p^*|} = \frac{|p-p^*|}{|p^*|}$.

A la hora de expresar el error relativo hemos utilizado dos cuantificaciones porque no siempre estaremos en disposición de conocer el valor exacto de un número aunque si que podamos conocer aproximaciones y estimaciones del error absoluto cometido.

Utilizando la desigualdad triangular, $|p| \geq |p^*| - |p - p^*|$. Trasladando está acotación a la definición de e_r se obtiene la acotación:

$$e_r \leq \frac{\widetilde{e_r}}{1 - \widetilde{e_r}}.$$

De la misma forma, como $|p^*| \geq |p| - |p - p^*|$, también se tiene la acotación

$$\widetilde{e_r} \leq \frac{e_r}{1 - e_r}.$$

Así, para valores pequeños de e_r (respectivamente de $\widetilde{e_r}$) los valores de e_r y $\widetilde{e_r}$ están muy próximos ($\frac{e_r}{\widetilde{e_r}} \approx 1$).

En el caso de tener errores relativos pequeños podemos utilizar indistintamente e_r o $\widetilde{e_r}$.

No siempre (por no decir casi nunca) tendremos conocimiento exacto del valor real p y sí que lo tendremos de p^*. Por eso, será más fácil obtener estimaciones del valor de $\widetilde{e_r}$ a partir de estimaciones de e_a que intentar trabajar con e_r.

- Utilizando el error absoluto, podemos escribir:

$$p = p^* + e = p^* \pm e_a \text{ y también } p^* - e_a \leq p \leq p^* + e_a.$$

- Si tenemos una acotación de e_a, $e_a < \delta$, entonces podemos asegurar que

$$p^* - \delta < p < p^* + \delta.$$

- Utilizando el error relativo, si tenemos una acotación de $\widetilde{e_r}$, $\widetilde{e_r} < \varepsilon$, entonces podemos asegurar que $e_a = |p^*|\widetilde{e_r} < |p^*|\varepsilon$, y

$$p^* - |p^*|\varepsilon < p < p^* + |p^*|\varepsilon.$$

- En general, podemos escribir:

$$p = p^* + e = p^*(1 \pm \widetilde{e_r}) \text{ y también } p^* = p(1 \pm e_r).$$

Ejercicio 1.3.1

(I) Calcula el error relativo que se comete cuando el número 1.503 aparece redondeado a 1.5.

(II) Calcula los errores absoluto y relativo cuando el número $.abcE7$ aparece escrito como $a.bcE7$

(III) Supón que en un cálculo aparece el número 0.0001 cuando debía aparecer el 0. ¿Qué fórmula utilizarías para calcular el "error relativo"? Utilizando esa fórmula ¿qué valor obtienes?

(IV) Determina el mayor intervalo en el que debe estar p^* para aproximar a $p = \sqrt{2}$ con un error relativo de a lo sumo 10^{-6}.

1.4. Números de máquina. Redondeo

Definición 1.4.1 (Números de máquina) *Vamos a llamar «máquina» de t dígitos en la mantisa y k dígitos en el exponente en base b al conjunto de números que se pueden representar en coma flotante utilizando esta base y los referidos dígitos para la mantisa y el exponente.*

A estos números los llamaremos números de máquina de t dígitos en la mantisa y k dígitos en el exponente en base b.

(Salvo que expresemos lo contrario, no nos vamos a preocupar mucho por el exponente, que es entero y sólo puede producir errores de desbordamiento)

1.4.0.4. *Redondeo*

Por redondeo vamos a entender el proceso de introducir un número p en una «máquina », en el que se procurará utilizar el número de máquina p^* que esté más próximo a p (o uno de los que estén más próximos si es que hay varios).

Existen distintos tipos de redondeo, nosotros nos vamos a fijar en el siguiente:

Definición 1.4.2 (Redondeo) *Sean $b \geq 2$ un número natural, $t \in \mathbb{N}$ $t > 0$, y $x \in \mathbb{R} \setminus \{0\}$ descrito en coma flotante y base b por:*

$$x = \pm b^N \sum_{j=1}^{\infty} a_{-j} \frac{1}{b^j} = \pm .a_{-1}a_{-2}\dots EN_{(b)},$$

$a_{-k} \in \{0, 1, ..., (b-1)\}$.

Sean $x^* = \pm b^N \sum_{j=1}^{t} a_{-j} \frac{1}{b^j} = \pm .a_{-1}a_{-2}\dots a_{-t}EN_{(b)}$ el número de máquina que resulta al "truncar" la mantisa a los t primeros dígitos y $x^{*+} = \pm b^N \left(\left(\sum_{j=1}^{t} a_{-j} \frac{1}{b^j} \right) + \frac{1}{b^t} \right)$. Los números x^* y x^{*+} son los dos de la máquina de base b y t dígitos en la mantisa, que están más próximos a x.

Vamos a definir el número redondeado de x con t dígitos en la mantisa como el número real

$$fl_t(x) = \begin{cases} x^*. & si\ |x - x^*| < \frac{|x^* - x^{*+}|}{2} = 0.5b^{N-t} \\ x^{*+} & si\ |x - x^*| \geq \frac{|x^* - x^{*+}|}{2} = 0.5b^{N-t} \end{cases}$$

Cuando el número de dígitos t este fijado en el entorno de trabajo podemos escribir solamente $fl(x)$.

Observad que si b es par (por ejemplo si $b = 2$ o $b = 10$), entonces $|x - x^*| < \frac{|x^* - x^{*+}|}{2}$ si, y sólo si $a_{-t-1} < \frac{b}{2}$.

Podéis leer sobre otros tipos de redondeo en Kincaid-Cheney [3]:

(I) Redondeo por paridad («Roud to even»), redondeo equidistribuido

(II) Redondeo por truncamiento.

Proposición 1.4.3 (Errores de redondeo) *Sean $b \geq 2$ un número natural, $t \in \mathbb{N}$ $t > 0$, y $x \in \mathbb{R} \setminus \{0\}$. Entonces*

(I) $fl_t(x) = \pm b^{N'} \sum_{j=1}^{t} a'_{-j} \frac{1}{b^j} = \pm .a'_{-1}a'_{-2}\dots a'_{-t}EN_{(b)}$, con $N \leq N' \leq N + 1$.

(II) El error absoluto en el redondeo satisface la acotación

$$|x - fl_t(x)| \leq 0.5b^{N-t}$$

(III) Los errores relativoa en el redondeo satisfacen la acotación

$$\frac{|x - fl_t(x)|}{|x|} \leq 0.5b^{-t+1} \ y \ \frac{|x - fl_t(x)|}{|fl_t(x)|} \leq 0.5b^{-t+1}$$

El número $0.5b^{-t+1}$ se denomina la precisión relativa en la aritmética de t dígitos en coma flotante.

DEMOSTRACIÓN:

$$|x - fl_t(x)| = |x - x^*| < 0.5b^{N-t} \quad \text{si} \quad |x - x^*| < \frac{|x^* - x^{*+}|}{2} = \frac{b^N b^{-t}}{2} = 0.5b^{N-t}$$

$$|x - fl_t(x)| = |x - x^{*+}| \quad \text{si} \quad |x - x^*| \geq \frac{|x^* - x^{*+}|}{2} \quad \text{y en este caso}$$

$$|x - x^{*+}| = |(x - x^*) + (x^* - x^{*+})| \overset{(1)}{=} |x^* - x^{*+}| - |x - x^*| \leq \frac{|x^* - x^{*+}|}{2} = 0.5b^{N-t}.$$

(1) $x^* - x^{*+}$ y $x - x^*$ tienen distinto signo tanto si x es positivo como si es negativo. □

Ejercicio 1.4.1

(I) Mide los errores relativos del redondeo del número π en una máquina de cuatro dígitos de mantisa y base 10

(II) En una máquina de números escritos en coma flotante con 13 dígitos en la mantisa (en base 10), ¿cuál es el máximo error relativo posible?

Definición 1.4.4 (Precisión de k dígitos) *Se dice que el número p^* es una aproximación del número $p \neq 0$ con una precisión de, al menos, m cifras significativas en la base b, siempre que el error relativo*

$$\frac{|p - p^*|}{|p|} \leq .5 \times b^{-m+1}.$$

Cuando m es el mayor entero para el que se cumple la desigualdad anterior, se dice que p^* aproxima a p con m cifras significativas.

Ejemplo 1.4.5 Sea

$$e = .271828182845904523536028747135 27...E1$$

y $app_1(e) = .2718287E1$ una aproximación de e.

$$\frac{|e - app_1(e)|}{|e|} = \frac{.5171540\ldots E - 5}{.271828182E1} \approx 1.90250E - 6 = .19025E - 5,$$

$$.5E(-7+1) < .190250E - 5 < .5E(-6+1)$$

En este caso, $app_1(e)$ aproxima a e con 6 cifras significativas y en las mantisas de e y $app(e)$ coinciden las 6 primeras cifras.

No siempre sucede esto. Por ejemplo, sea $x = .1E0$ y $app(x) = .9999E0$, en este caso

$$\frac{|x - app(x)|}{|x|} = .1E - 3.$$

Con la definición anterior $app(x)$ es una aproximación de x con 4 cifras significativas, aunque no coinciden ninguna de las cifras de la mantisa.

Ejercicio 1.4.2 Una calculadora, A, trabaja en base 2 con mantisa de 22 bits y otra, B, trabaja en base 16 con 6 dígitos de precision (24 bits). ¿Cuál de las dos es más precisa?

1.4.0.5. *Distribución de Números de Máquina*

¿Cómo están distribuidos los números de máquina?

Resp: Hay muchos cerca del cero, muy pocos en el ∞, y suficientes alrededor de los números razonables (.1, 1, 10, ...).

Definición 1.4.6 (Épsilon de la Máquina) *Si M es una máquina ideal de números (t dígitos en mantisa y base b), se denomina "Épsilon de la Máquina" al menor número positivo $e \in M$ tal que $fl(1+e) > 1$. O en otros términos, al menor número positivo tal que $fl(1+e)$ es el número de máquina siguiente al 1,* $1^+ \frac{1}{b^t}$

En doble precisión (IEEE-754) el épsilon de la máquina es del orden de $\frac{1}{2^{52}} \approx 10^{-16}$. En prácticas lo precisaremos con exactitud.

El épsilon de la máquina nos indica la máxima precisión relativa que cabe esperar en la máquina. Así, a la hora de preguntar si dos números de máquina están próximos debemos de tener en cuenta no pedir que el error relativo sea menor que el épsilon de la máquina, porque en ese caso estaremos pidiendo que sean idénticos.

1.5. Aritmetica de la máquina. Propagación de errores

¿Cómo suma, resta, multiplica, divide, etc... una «máquina»?

Denotemos por $\square$ cualquiera de las operaciones $+, -, \times, /$. Si x e y son dos números de máquina con t dígitos en la mantisa, en general $x \square y$ no tiene porque ser un número de máquina con t dígitos en la mantisa:

Por ejemplo, supongamos que trabajamos en una máquina de 6 dígitos en la mantisa y que $x = .700001E1$ e $y = .600002E0$ son dos números de la máquina. Su suma, $x + y = .7600012E1$, es un número de 8 dígitos en la mantisa que debe ser redondeado por un número de la máquina si se quiere guardar en la misma, $x \oplus y := fl(x + y) = .760001E1$.

Así, para realizar operaciones aritméticas en una máquina ideal, pensaremos siempre que estas se realizan en dos etapas

(I) Primero se calcula $x \square y$ con la mayor precisión posible;

(II) después se redondea el resultado a un número de la máquina.

¿Cómo se multiplican dos números de máquina? ¿Cómo afectan los errores de redondeo en los factores en el producto?

¿Cómo se suman dos números de máquina del mismo signo? ¿Cómo afectan los errores de redondeo en los sumandos a la suma?

Ahora que sabemos como se deben realizar las operaciones dentro de nuestra calculadora, es buen momento para volver sobre el ejemplo 1.1.4 :

Ejemplo 1.5.1 Utilizamos la calculadora[4] para evaluar las expresiones de la igualdad

$$18817 - 10864\sqrt{3} = \frac{1}{18817 + 10864\sqrt{3}}.$$

Hicimos los cálculos,obteniendo dos valores distintos y nos preguntábamos cual de los dos era el más preciso.

Vamos a ir detallando las operaciones

(I) $fl(\sqrt{3}) = 1.732050808$, y $\sqrt{3} = 1.732050808(1 \pm e_{red})$

(II) $e_{red} \leq .5E - 10$

(III) $10864 \otimes fl(\sqrt{3}) = 18816.9997$,

[4]Casio fx-991ES: presenta 10 dígitos en pantalla, presupondremos que es una máquina de números de 10 dígitos en la mantisa.

(IV) $10864 \times \sqrt{3} = 10864 \times (fl(\sqrt{3}) \pm e_a) = (18816.9997 \pm e_{red}) \pm (10864 \times e_{red})$

(V) $e_1 = (10864 \times e_{red}) \pm e_{red} \leq .54E-6$, e_1 es una estimación del error en la operación $10864 \times \sqrt{3}$.

(VI) $x_1 = 18817 \ominus (10864 \otimes fl(\sqrt{3})) = 2.65718E-5$

(VII) $x = 18817 - (10864 \times \sqrt{3}) = 18817 - 18816.9997 \pm e_1$

(VIII) $e_1 = |x - x_1| \leq .54E-6$ y $\frac{|x-x_1|}{|x_1|} \leq .20E-1$. Así, sólo podemos afirmar que x_1 aproxima al valor de x con una precisión de dos dígitos.

De forma análoga podemos escribir

(IX) $d = 18817 \oplus (10864 \otimes fl(\sqrt{3})) = 37633.99997$, y $18817 + (10864 \times \sqrt{3}) = 37633.99997 \pm e_1$.

(X) $x_2 = 1 \oslash (18817 \oplus (10864 \otimes fl(\sqrt{3}))) = 2.657171708E-5$

(XI) $1/(18817 + (10864 \otimes fl(\sqrt{3}))) = 2.657171708E-5(1 \pm e_{red2})$

(XII) $e_{red2} \leq .5E(-10)$

(XIII)

$$\begin{aligned} x = 1/(18817 + (10864 \times \sqrt{3})) &= \frac{1}{18817 \oplus (10864 \otimes fl(\sqrt{3}))} \frac{18817 \oplus (10864 \otimes fl(\sqrt{3}))}{18817 + (10864 \times \sqrt{3})} \\ &= (2.657171708E-5)(1 \pm e_{red2}) \frac{37633.99997}{37633.99997 \pm e_1} \\ &\approx (2.657171708E-5)\,(1 + \frac{\pm(.54E-6)}{37633.99997 - (.54E-6)}) \\ &\approx 2.657171708E-5\,(1 \pm .1435E-10) \end{aligned}$$

(XIV) $\frac{|x-x2|}{|x_2|} \leq .1435E-10$. Ahora podemos asegurar que x_2 aproxima a x con los 10 dígitos de precisión.

¿Cuando se restan dos números de máquina próximos, cúal es la precisión en la diferencia?

Proposición 1.5.2 *Sea M una máquina de números de base b y k dígitos en la mantisa. Sean $x > y > 0$ dos números de M en cuyas representaciones coinciden el exponente y los p primeros dígitos de las mantisas. Entonces el número $x \ominus y$ a lo más tiene en su mantisa, $k - p$ dígitos significativos. Se pierden p dígitos de precisión.*

Ejercicio 1.5.1 Consideremos los números $p = 0.54619$ y $q = 0.54601$. Redondeando los números en una máquina de cuatro dígitos de precisión (base 10), mide el error relativo cometido al aproximar $p - q$ por $fl(p) - fl(q)$. ¿cúantas cifras significativas tiene $fl(p) - fl(q)$.

¡Analiza el resultado! ¿Podemos mejorarlo?

Ejercicio 1.5.2 Utilizando aritmética de cuatro dígitos (base 10), encuentra aproximaciones a las raíces x_1 y x_2 de la ecuación de segundo grado

$$x^2 - 200x + 1 == 0.$$

Comprueba la bondad de las aproximaciones reconstruyendo el polinomio de segundo grado $(x - x_1)(x - x_2)$.

¡Analiza el resultado! ¿Podemos mejorarlo?

1.6. Cálculos estables e inestables

Definición 1.6.1 *Diremos que un proceso numérico, o una operación, es* ***inestable*** *cuando pequeños errores en los datos de entrada, o errores de redondeo en alguna de las etapas el proceso, producen errores grandes en los datos de salida*

Diremos que un proceso numérico, es ***estable*** *cuando no es inestable.*

Un mismo algoritmo puede ser estable para algunos datos iniciales e inestable para otros. Entonces se dice que el algoritmo es ***condicionalmente estable***

Ejercicio 1.6.1 Evalúa la expresión $z = 1. - \dfrac{1.208}{x}$ en $x = 1.209$ y en $x = 3.1$, trabajando con 4 dígitos de precisión en base 10.

Analiza los algoritmos:

- $x \to (y = 1.208/x) \to (z = 1 - y)$,
- $x \to (u = x - 1.208) \to (z = u/x)$.

Después de ver cómo se realizan las operaciones aritméticas en una máquina de números, podemos quedarnos con el siguiente resumen:

- El producto $\otimes$ de dos números de máquina es un cálculo estable, sólo se pueden producir errores de desbordamiento ("overflow").
- La división $\oslash$ de dos números de máquina es un cálculo estable, también sólo se pueden producir errores de desbordamiento ("overflow").
- La suma $\oplus$ de dos números de máquina es estable cuando los dos números tienen el mismo signo, y puede ser inestable cuando los dos números tienen signo distinto.
- La resta $\ominus$ de dos números de máquina es inestable cuando los dos números están muy próximos, y es estable cuando los dos números tienen distinto signo (en este caso es una suma de números del mismo signo).
- Hay que prestar atención a las operaciones con números grandes o excesivamente pequeños para evitar errores de desbordamiento ("overflow").

Ejercicio 1.6.2 En relación con el último punto, ¿Cómo se puede realizar el cálculo

$$\frac{e^x}{e^x - 1}$$

para valores grandes de x $(x > 250)$?

¡Compruébalo con la calculadora!

1.6.1. Propagación de errores

¿Cómo afecta el número de operaciones al resultado final?

Si las operaciones son estables, lo menos que podemos esperar es que si en cada operación efectuamos un redondeo $e_r \leq 0.5b^{-t+1}$, después de n operaciones, tengamos un error acumulado del orden de ne_r, proporcional al número de operaciones.

Proposición 1.6.2 *Sean $x_i \geq 0$, números de máquina (base b y t-dígitos en la mantisa). Sean*

$$S_n = x_0 + x_1 + \cdots + x_n = S_{n-1} + x_n,$$

$$S_n^* = S_{n-1}^* \oplus x_n \ (S_0^* = x_0).$$

Entonces:

$$\frac{|S_n - S_n^*|}{|S_n|} \leq (1 + e_M)^n - 1 \approx n e_M,$$

donde $e_M = .5b^{-t+1}$ es la precisión de la máquina.

(ver prueba en el libro de Kincaid-Cheney [3])

Ejercicio 1.6.3 Sea n tal que $ne_M < \frac{1}{3}$. Utilizando el binomio de Newton comrueba que

$$1 + (0.5)ne_M \leq (1 + e_M)^n \leq 1 + (1.5)ne_M.$$

Ejercicio 1.6.4 Sean $x_i > 0$, números de máquina (base b y t-dígitos en la mantisa). Sean

$$P_n = x_0 x_1 \cdots x_n = P_{n-1} x_n,$$

$$P_n^* = P_{n-1}^* \otimes x_n \ (P_0^* = x_0).$$

Entonces:

$$\frac{|P_n - P_n^*|}{|P_n|} \leq (1 + e_M)^n - 1 \approx n e_M,$$

donde $e_M = .5b^{-t+1}$ es la precisión de la máquina.

Ejercicio 1.6.5 Para evaluar un polinomio $p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ disponemos de los dos algoritmos siguientes

(I) p=a_0; z=1;
para (k=1, hasta n, ↑)
z=z*x; p=p+a_k *x;

(II) (Horner)
p=a_n;
para (k=(n-1), hasta 0, ↓)
p=p*x +a_k;

Si ε es una cota del error relativo al hacer las operaciones aritméticas y no consideramos los errores de redondeo al introducir el valor de los parámetros, ¿Cuál de los dos algorimos acumula menos errores?

Ejercicio 1.6.6 Haz los ejercicios 7 y 10 de la hoja de problemas 1.

- A la hora de escribir un algoritmo o un proceso numérico, debemos intentar minimizar el número de operaciones para evitar acumulación de errores de redondeo.

1.6.2. Propagación de Errores en Procesos Iterativos

Un algoritmo o un método iterativo, es un procedimiento por el que se construye una sucesión x_n por etapas. Cada x_n dependerá de unos primeros datos iniciales a los que se van agregando en cada etapa los términos de la sucesión que ya se han calculado. En general,

$$x_n = f(x_0, x_1, ..., x_{n-1}) :$$

Normalmente, en los procesos iterativos esperamos que x_n sea una sucesión convergente hacia la solución de un problema $\overline{x}$.

En principio, desconocemos el valor exacto de $\overline{x}$ y, ante la presencia de errores inevitables (como los de redondeo), deberemos parar la iteración cuando creamos que x_n es una buena aproximación de $\overline{x}$.

En general, la condición de parada viene dada en términos de los errores absolutos o relativos pidiendo que :

$$\Delta_n = |x_n - x_{n-1}| < prec \text{ o que } \frac{\Delta_n}{|x_n|} = \left|1 - \frac{x_{n-1}}{x_n}\right| < prec,$$

donde $prec > 0$ es una precisión dada.

Normalmente, este tipo de condición de parada se complementa con un control sobre la ecuación o sistema de ecuaciones que modelizan el problema que se está intentando resolver. Por ejemplo si $F(\overline{x}) == 0$, podemos agregar a la condición de parada que $|F(x_n)| < prec$.

Recordemos que si $\overline{x} = 0$ sólo podemos usar errores absolutos, Y que si x_n es convergente la evaluación de Δ_n es inestable porque x_n y x_{n-1} pueden estar muy próximos.

Definición 1.6.3 *Supongamos que x_n está definida por un método iterativo y que $x_n \to \overline{x}$. Al implementar el método en una máquina, vamos construyendo una sucesión $\widetilde{x_n}$ que va acumulando los errores de redondeo y de propagación según las operaciones que intervienen.*

Denotamos $e_n = |x_n - \widetilde{x_n}|$ si $\overline{x} = 0$, y $e_n = \frac{|x_n - \widetilde{x_n}|}{|x_n|} = \left|1 - \frac{x_n}{\widetilde{x_n}}\right|$ si $\overline{x} \neq 0$, a los errores absoluto y relativo en cada etapa.

(I) *Se dice que la propagación de errores en la iteración es* ***lineal****, cuando existe $C > 0$ tal que*

$$e_n \approx Cne_0, \forall n.$$

(II) *Se dice que la propagación de errores en la iteración es de tipo exponencial cuando existen $C > 1$ y $M > 0$ tal que*

$$e_n \geq MC^n e_0, \forall n.$$

(III) *A veces encontraremos ejemplos en los que*

$$e_n \geq Kn!e_0, \forall n. \ (K > 0 constante)$$

En estos casos, si recordamos la fórmula de Stirling ($n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$), la propagación de errores es de tipo exponencial con cualquier constante $C > 1$.

- *Admitiremos que un proceso iterativo con propagación de error lineal va a ser estable si no requiere "demasiadas" iteraciones para pararse.*

  Debemos de establecer números máximos de iteraciones para evitar problemas de acumulación de errores que desvirtúen la solución aproximada propuesta.

- *Los procesos iterativos con propagación de errores exponencial van a ser muy inestables. Debemos intentar evitarlos.*

Observad los dos ejemplos siguientes que muestran dos procesos iterativos algo perversos. En la practica 2 os propongo un ejemplo más para que lo analicéis.

Ejemplo 1.6.4 Sea $y_n = n!(e^x - (1 + x + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}))$. Sabemos, por la expresión de Lagrange del resto de la fórmula de Taylor, que

$$y_n = n!\frac{e^{tx}}{(n+1)!}x^{n+1} = \frac{e^{tx}}{(n+1)}x^{n+1}.$$

De donde podemos concluir que para $x \leq 1$ $y_n \to 0$.

También sabemos que se cumple la fórmula de recurrencia lineal:

$$y_n = ny_{n-1} - x^n.$$

Si evaluamos los 21 primeros términos de esta iteración comenzando en $x_0 = 1$ e $y_0 = e - 1$, en una máquina que trabaja con reales de tipo `double` obtenemos:

y_0 =1.71828	y_{11}=0.090234
y_1 =0.718282	y_{12}=0.0828081
y_2 =0.436564	y_{13}=0.0765057
y_3 =0.309691	y_{14}=0.0710802
y_4 =0.238764	y_{15}=0.066203
y_5 =0.193819	y_{16}=0.0592478
y_6 =0.162916	y_{17}=0.00721318
y_7 =0.140415	y_{18}=−0.870163
y_8 =0.123323	y_{19}=−17.5331
y_9 =0.109911	y_{20}=−351.662
y_{10}=0.0991122	y_{21}=−7385.9

¿Qué ocurrió?

Si denotamos $\widetilde{y_n} = n \otimes \widetilde{y_{n-1}} \ominus 1$, podemos estimar que

$$\widetilde{y_n} - y_n = n(\widetilde{y_{n-1}} - y_{n-1}) = ... = n!(\widetilde{y_0} - y_0).$$

Aquí podemos observar como el pequeño error de redondeo al escribir el número e en la máquina y evaluar $\widetilde{y_0}$, se ha propagado de forma exponencial ($n! > M(3/e)^n$ si $n \geq 3$).

Ejemplo 1.6.5 Sea $x_n = \int_0^1 x^n e^x\, dx$. Sabemos que $0 \leq x_n \leq e\int_0^1 x^n\, dx = \frac{e}{n+1} \to 0$.

Integrando por partes se tiene que x_n viene dada por la fórmula de recurrencia lineal:

$$x_n = e - nx_{n-1},$$

comenzando con $x_0 = e - 1$.

Razonando como en el ejemplo precedente, si $\widetilde{x_n} = e \ominus n \otimes \widetilde{x_{n-1}}$ podemos estimar que

$$|\widetilde{x_n} - x_n| = n|\widetilde{x_{n-1}} - x_{n-1}| = ... = n!|\widetilde{y_0} - y_0)|,$$

y predecir que la sucesión $\widetilde{x_n}$ es muy inestable y no converge a 0, en absoluto.

En efecto, si trabajáis en doble precisión con números de tipo `double` podréis observar que con 25 iteraciones $x_25 \approx 8.2E8$ muy lejos del límite 0.

¡Evalúa los 25 primeros términos de la sucesión $\widetilde{x_n}$ en el ordenador!

1.7. Condicionamiento de un problema

Los términos "condicionamiento" y/o "problema bien o mal condicionado" se utilizan de manera informal para indicar la sensibilidad de la solución del problema con respecto a pequeños errores relativos en los datos de entrada.

Una posibilidad para hacer este tipo de apreciaciones es intentar "cuantificar" a priori esta sensibilidad, utilizando para ello un "modelo" apropiado del problema.

A continuación vamos a ver distintos "modelizaciones" de problemas en los que se pueden hacer esas cuantificaciones.

1.7.1. Funciones de una variable.

Este es el caso en el que el algoritmo consiste simplemente en dado un número x como dato de entrada, evaluar $f(x)$ como dato de salida, donde $f : \mathbb{R} \to \mathbb{R}$ es una función real.

Si además f es derivable, con derivada continua, podemos cuantificar el condicionamiento del algoritmo como sigue:

- Sea x el dato de entrada, y $\widetilde{x} = x + h$ una perturbación de x como dato de entrada.
- En la salida obtenemos $f(x + h)$ en lugar de $f(x)$.
- El error en salida se puede estimar en términos de la derivada:

  $$|f(x + h) - f(x)| = |f'(x + th)||h| \approx |f'(x)||h|$$

  donde $0 < t < 1$ y para h pequeño, $|f'(x + th)| \approx |f'(x)|$.
- El error relativo en el dato de salida es

  $$\frac{|f(x + h) - f(x)|}{|f(x)|} \approx \frac{|f'(x)||h|}{|f(x)|}$$

- Si medimos la razón entre los errores relativos en los datos de salida y los datos de entrada, tenemos

  $$\frac{e_r(f(x))}{e_r(x)} = \frac{\frac{|f(x+h)-f(x)|}{|f(x)|}}{\frac{|h|}{|x|}} \approx \frac{|f'(x)||x|}{|f(x)|}.$$

  Esta última estimación sólo depende de x y de f y se puede preveer antes de ejecutar el algoritmo de evaluación.

Definición 1.7.1 *Sea $f : [a, b] \to \mathbb{R}$ una función derivable con derivada continua. y sea $x \in [a, b]$ con $f(x) \neq 0$. El "número de condición de f en x" se define por la fórmula*

$$nc(f, x) := \frac{|f'(x)||x|}{|f(x)|}$$

y mide la razón entre los errores relativos en $f(x)$ y en x ante variaciones de este último valor.

Cuando el número de condición es pequeño, o menor que una cota razonable, pequeños errores en x se propagan a pequeños errores en $f(x)$ y el algoritmo es estable. Pero si los números de condición son muy grandes o tienden a infinito cerca de un valor de x el algoritmo será inestable cerca de este valor.

Ejemplo 1.7.2

(I) $f(x) = xe^{-x^2}$ $(f'(x) = e^{-x^2} - 2x^2e^{-x^2})$

$$nc(xe^{-x^2}, x) = \frac{|f'(x)||x|}{|f(x)|} = \frac{|(e^{-x^2} - 2x^2e^{-x^2})x|}{|xe^{-x^2|}} = |1 - 2x^2e^{-x^2|} \leq 1$$

La evaluación de $f(x) = xe^{-x^2}$ es estable en todo x: "los errores relativos en x y $f(x)$ son del mismo tamaño".

(II) $f(x) = \operatorname{arc\,sen}(x)$ $(f'(x) = \frac{1}{\sqrt{1-x^2}})$

$$nc(\operatorname{arc\,sen}(x), x) = \frac{|f'(x)||x|}{|f(x)|} = \frac{\frac{|x|}{\sqrt{1-x^2}}}{\operatorname{arc\,sen}(x)} = \frac{|x|}{\operatorname{arc\,sen}(x)\sqrt{1-x^2}}$$

$$\lim_{x \to 1^-} nc(\operatorname{arc\,sen}(x), x) = +\infty$$

La evaluación de $\operatorname{arc\,sen}(x)$ cerca de $x = 1$ $(\operatorname{arc\,sen}(1) = \frac{\pi}{2})$ es muy inestable: "pequeños errores en x producen errores grandes en $\operatorname{arc\,sen}(x)$".

(III) $f(x) = \log(x)$ $(f'(x) = \frac{1}{x})$

$$nc(\log(x), x) = \frac{|f'(x)||x|}{|f(x)|} = \frac{1}{|\log(x)|}$$

Si x está lejos de 1 la evaluación de $\log(x)$ es estable, pero si x esta cerca de $x = 1$ esta evaluación es inestable.

1.7.2. Funciones de varias variables

Este es el caso en el que el algoritmo consiste simplemente en dado un vector $\vec{x} \in \mathbb{R}^n$ como dato de entrada, evaluar $f(\vec{x})$ como dato de salida, donde $f : \mathbb{R}^n \to \mathbb{R}$ es una función real de varias variables.

Si además f es "diferenciable" con derivadas parciales[5] continuas podemos cuantificar el condicionamiento del algoritmo como sigue:

(Para simplificar las expresiones vamos a utilizar el caso $n = 3$)

- Sea $\vec{x} = (x_0, y_0, z_0)$ el dato de entrada, y $\widetilde{\vec{x}} = \vec{x} + \vec{h} = (x_0 + h_x, y_0, z_0)$ una perturbación en la primera coordena de $\vec{x}$ como dato de entrada.
- En la salida obtenemos $f(\vec{x} + \vec{h})$ en lugar de $f(\vec{x})$.

[5] Sea $f(x, y, x)$ una función de tres variables reales con valores en $\mathbb{R}$, si para cada par (y_0, z_0) fijo, la función de una variable $f_{y_0,z_0}(x)$ es derivable en x_0, se define la derivada parcial

$$\frac{\partial f(x_0, y_0, z_0)}{\partial x} = f'_{y_0,z_0}(x_0)$$

De forma análoga se definen las derivadas parciales de f en $\vec{x}_0$ con respecto a cada una de las coordenadas, derivando la función de una variable que resulta al fijar el resto de las coordenadas.

Si existen todas las derivadas parciales en $\vec{x}_0$ y son continuas, entonces f es "diferenciable en $\vec{x}_0$" en el sentido de que $f(\vec{x}) - f(\vec{x}_0)$ se aproxima "bien" por una aplicación lineal.

- El error en salida se puede estimar en términos de las derivadas parciales:

$$\begin{aligned}|f(\vec{x}+\vec{h})-f(\vec{x})| &= |f(x_0+h_x,y_0,z_0)-f(x_0,y_0,z_0)| \\ &= |\frac{\partial f(x_0+t_1h_x,y_0,z_0)}{\partial x}||h_x|\end{aligned}$$

  donde $0<t_1<1$ y para h_x pequeño, $|\frac{\partial f(x_0+t_1h_x,y_0,z_0)}{\partial x}| \approx |\frac{\partial f(x_0,y_0,z_0)}{\partial x}|$.

- El error relativo en el dato de salida es

$$\frac{|f(\vec{x}+\vec{h})-f(\vec{x})|}{|f(\vec{x})|} \approx \frac{|\frac{\partial f(x_0,y_0,z_0)}{\partial x}||h_x|}{|f(\vec{x})|}$$

- Si medimos la razón entre los errores relativos en los datos de salida y los datos de entrada, tenemos

$$\frac{e_r(f(\vec{x})),x_0}{e_r(x_0)} \approx \frac{|\frac{\partial f(x_0,y_0,z_0)}{\partial x}||x_0|}{|f(\vec{x})|}.$$

  Como antes, esta última estimación sólo depende de $\vec{x}$ y de f y se puede preveer antes de ejecutar el algoritmo de evaluación.

- De la misma forma se pueden obtener estimaciones de las razones entre los errores relativos cuando se van perturbando de forma independiente cada una de las coordenadas.

Definición 1.7.3 *Sea $f : A \subset \mathbb{R}^n \to \mathbb{R}$ una función diferenciable con derivadas parciales continuas continua. y sea $\vec{x} \in A$ con $f(\vec{x}) \neq 0$. Los números*

$$nc(f,\vec{x},i) := \frac{|\frac{\partial f(\vec{x})}{\partial x_i}||x_i|}{|f(\vec{x})|}, i = 1,2,...,n$$

se llaman números de condición f en $\vec{x}$" y mide la razón entre los errores relativos en $f(\vec{x})$ y en $\vec{x}$ ante variaciones en cada una de las coordenadas de este vector.

Cuando todos los números de condición son pequeño, o menores que una cota razonable, pequeños errores en x se propagan a pequeños errores en $f(x)$ y el algoritmo es estable. Pero si alguno de los números de condición es muy grande o tiende a infinito cerca de un valor de $\vec{x}$ el algoritmo será inestable cerca de este valor.

Ejemplo 1.7.4 Vamos a estimar el error máximo de $f(x,y) = x^2y$ cuando $x = 2 \pm 0.3$ e $y = 1.5 \pm 0.2$ y analizaremos la estabilidad de f en $(2, 1.5)$.

$\frac{\partial f(x,y)}{\partial x} = 2xy$ y $\frac{\partial f(x,y)}{\partial y} = x^2$,

Si $x \in [2-0.3, 2+0.3]$ e $y \in [1.5-0.2, 1.5+0.2]$, $|\frac{\partial f(x,y)}{\partial x}| \leq 2*2.3*1.7 = 7.82$ y $|\frac{\partial f(x,y)}{\partial y}| \leq 2.3^2 = 5.29$

$$\begin{aligned}|f(x,y)-f(2,1.5)| &\leq |f(x,y)-f(2,y)|+|f(2,y)-f(2,1.5)| \\ &= |\frac{\partial f(2+t(x-2),y)}{\partial x}(x-2)|+|\frac{\partial f(2,1.5+t(y-1.5))}{\partial y}(y-1.5)| \\ &\leq 7.28*|x-2|+5.29*|y-1.5| \\ &\leq 7.18*0.3+5.29*0.2 = 3.212\end{aligned}$$

El error máximo al evaluar $f(2, 1.5) = 6$ está acotado por 3.212 (el error relativo está acotado por 0.536).

Los números de condición con respecto a cada variable son:

$$nc(f,(x,y),1) := \frac{|\frac{\partial f(x,y)}{\partial x}||x|}{|f(x,y)|} = \frac{2x^2y}{x^2y} = 2, \text{ y}$$

$$nc(f,(x,y),2) := \frac{|\frac{\partial f(x,y)}{\partial y}||y|}{|f(x,y)|} = \frac{x^2y}{x^2y} = 1.$$

Podemos afirmar que el error en la coordenada x influye más que el error en la coordenada y al evaluar $f(x, y)$.

Observad también que los números de condición en este caso son independientes del vector (x, y).

1.7.3. Aplicaciones lineales

Cuando estudiemos los sistemas de ecuaciones lineales $A\vec{x} = \vec{b}$ (A matriz $n \times n$, consideraremos distintas normas para las matrices A y veremos que el número $nc(A) = \|A\| \, \|A^{-1}\|$ servirá como número de condición del sistema en el sentido de que mide la razón entre el error cometido en los datos iniciales del sistema $\|A\|$ o b y el error que se propaga a la solución del sistema.

Matrices como las de Hilbert

$$H_n = \left(\frac{1}{i+j-1}\right)_{\substack{1\le i\le n\\ 1\le j\le n}}$$

que aparecen de forma natural en ajustes funciones por polinomios de grado n, están muy mal condicionadas $nc(H_n) \approx e^{3.5n} \to \infty$.

1.7.4. Raíces de Polinomios

Vamos a analizar en el siguiente modelo (muy simple) cómo cuantificar el condicionamiento en el problema de aproximación de raíces de polinomios

Denotemos por $p(x)$ a un polinomio y supongamos que $algR(p)$ es un algoritmo que proporciona una raíz $r = algR(p)$ de $p(x)$ ($p(r) = 0$). Supongamos además que r es una raíz simple de p, e.d. $p'(r) \neq 0$.

Si se perturba el polinomio p de entrada, tomando una aproximación $\widetilde{p}(x) = p(x) + \varepsilon q(x)$, donde $q(x)$ es otro polinomio, y se aplica el algoritmo encontraremos una raíz $r + h = algR(\widetilde{p})$.

Si suponemos que h es pequeño, $h^2, h^3, ...$ serían mucho mas pequeños que h y podríamos despreciarlas frente a h. Así, utilizando el desarrollo de Taylor de p, como $p(r) = 0$, podemos modelizar

$$p(r+h) = p(r) + p'(r)h + \frac{p''(r)}{2}h^2 + ... + \frac{p^{(n)}(r)}{n!}h^2 \approx p'(r)h.$$

También si h es pequeño $q(r + h) \approx q(r)$. Así

$$0 = \widetilde{p}(r+h) = p(r+h) + \varepsilon q(r+h) \approx hp'(r) + \varepsilon q(r).$$

Despejando queda $h \approx -\varepsilon \frac{q(r)}{p'(r)}$.

Con este modelo, el número $\varepsilon \frac{q(r)}{p'(r)}$ puede servir para prever el error absoluto en el cálculo de la raíz en términos de la perturbación en el polinomio original.

Ejemplo 1.7.5 (Polinomios de Wilkinson) Cuando tengamos construidos e implementados algunos métodos de aproximación de raíces de polinomios, constataremos los hechos que ahora vamos a pronosticar en base al modelo propuesto más arriba.

Los polinomios de Wilkinson son los polinomios mónicos con raíces simples en los enteros $\{1, 2, ..., n\}$:

$$p_n(x) = (x-1)(x-2)...(x-n).$$

Consideremos la perturbación de p_{20} dada por $\widetilde{p_{20}}(x) = p_{20}(x) + \varepsilon x^{20}$.

Si $algR_{20}$ es un algoritmo que nos proporciona la raíz $r = 20$ de p_20, y $r + h_{20} = algR_{20}(\widetilde{p_{20}})$.

La estimación propuesta para el tamaño de h_{20} es

$$h_{20} \approx \varepsilon \frac{20^{20}}{p'_{20}(20)} = \varepsilon \frac{20^{20}}{19!} \approx \varepsilon 10^9.$$

Observad que una pequeña perturbación (ε) en el coeficiente de x^{20} se multiplica por 10^9 al intentar aproximar la raíz $r = 20$. Los polinomios de Wilkinson son un ejemplo de polinomios "mal condicionados" para la busqueda de raíces grandes.

Como ejercicio, podéis jugar con este polinomio y el modelo propuesto para estimar los errores que resultan al utilizar algoritmos $algR_{15}$ o $algR_{10}$ que proporcionen respectivamente las raíces $r = 15$ o $r = 10$.

1.8. Actividades complementarias del capítulo

Los documentos se pueden descargar de la Zona Compartida de SUMA.

- Hoja de problemas nº1 (25/09/2007).
- Practica nº1 . "Fenómenos extraños".
- Práctica nº2 " Máquinas ideales".

Bibliografía

[1] A. Delshams A. Aubanell, A. Benseny, *Útiles básicos de cálculo numérico*, Ed. Labor - Univ. Aut. de Barcelona, Barcelona, 1993.

[2] R.L. Burden and J.D. Faires, *Análisis numérico, 7^a edición*, Thomson-Learning, Mexico, 2002.

[3] D. Kincaid and W. Cheney, *Análisis numérico*, Ed. Addison-Wesley Iberoamericana, Reading, USA, 1994.

Capítulo 2

Algoritmos e Iteraciones

Interrogantes centrales del capítulo

- Conocer la estructura general de los algoritmos y establecer criterios para compararlos.
- Aproximar soluciones de ecuaciones no lineales $f(x) == 0.$ con métodos elementales:
  - Método de la bisección.
  - Métodos de Regula-Falsi y Newton.
- Conocer y analizar las iteraciones de punto fijo.
- Analizar la convergencia y el error en los métodos iterativos.
- Acelerar la convergencia de una sucesión.

Destrezas a adquirir en el capítulo

- Ser capaz de describir algoritmos sencillos y, en particular, los algoritmos correspondientes a los distintos métodos del capítulo.
- Saber implementar en el ordenador los programas de los métodos del capítulo.
- Conocer el enunciado y la demostración del teorema del punto fijo de Banach así como condiciones suficientes que garanticen sus hipótesis.
- Aplicar los programas construidos de forma efectiva en la localización y aproximación númerica de raíces de algunas ecuaciones no lineales.
- Analizar la convergencia de las aproximaciones proporcionadas por los distintos métodos y la conveniencia de utilizar uno u otro método numérico.

En la unidad anterior hemos trabajado con algunos algoritmos de una forma más o menos informal. Para escribir los algoritmos de forma que se puedan implementar fácilmente en un ordenador debemos ser más precisos, considerar siempre que tienen que construirse sobre una "estructura" clara y ordenada.

En este capítulo describiremos la forma general de los algoritmos, cómo se evalúan y cómo medir su complejidad para poder compararlos. Prestaremos especial atención a los algoritmos iterativos y al "orden de su convergencia". Trabajaremos con algunos algoritmos elementales de resolución de ecuaciones, estudiaremos el teorema del punto fijo de Banach y el algoritmo de Aitken para acelerar la convergencia de sucesiones.

Desarrollo de los contenidos fundamentales

- Algoritmos: forma, evaluación y complejidad.
- Algoritmos iterativos. Métodos iterativos elementales de resolución de ecuaciones no lineales:
  - Método de la bisección.
  - Método de regula-falsi.
  - Método de Newton.
- Iteración de punto fijo. Atractores y repulsores
- Orden de convergencia de una sucesión y constante asintótica del error.
- Aceleración de la convergencia: algoritmos de Aitken y Steffensen.

Temporalización: 8HTe + 2HPb + 2HLab = 12 H

2.1. Introducción

Algunos autores aseguran que la palabra "algoritmo" proviene del nombre del matemático persa Abu Jafar Mohamemed ibn Musa al-Khowarizmi, que trabajo en Bagdad alrededor del año 840. Aunque si acudimos al diccionario de la Real Academia Española encontramos la siguiente definición que se corresponde con nuestras intenciones:

algoritmo: (Quizá del lat. tardío "algobarismus", y este abrev. del ár. clás. " 'lisabu l'ubar", cálculo mediante cifras arábigas).

(I) m. Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

(II) m. Método y notación en las distintas formas del cálculo.

Comencemos este capítulo con una visita a un prototipo típico de un algoritmo:

El algoritmo de Euclides para encontrar el máximo común divisor de dos números naturales que fue descrito alrededor del año 350 a.C. y se puede visitar en el libro de los "Elementos de Euclides".

Algoritmo 2.1 **El algoritmo de Euclides**

Datos de entrada: m, $n \in \mathbb{N}$
Datos de salida: El máximo común divisor de m y n: $MCD(m,n) \in \mathbb{N}$
Variables:
$m^*; n^*$; // para operar sin cambiar los números.
$auxiliar$; // para hacer cambios.
Fujo del programa:
`if(m >= n){` $m^* = m;\ n^* = n;$ `}`
`else{` $m^* = n;\ n^* = m;$`}`
`while(`$n^* \neq 0$`){`
$auxiliar = n^*;$
$n^* = Resto(m^* \div n^*);$
$m^* = auxiliar$
`}`
Parada: $MCD(m,n) = m^*$

¿Cómo encuentra este algoritmo el máximo común divisor?

Simplemente, basta con observar que los divisores comunes de m y n ($m \geq n$)también son los divisores comunes de n y $Resto(m \div n)$, si además este resto es cero n es el máximo común divisor .

El algoritmo de Euclides trabaja reduciendo el tamaño de m y n hasta que el resto sea cero, momento en el que habremos encontrado el máximo común divisor.

Ejemplo 2.1.1 Utilizando el algoritmo de Euclides calcula el $MCD(18981, 34560)$.

El algoritmo de Euclides muestra varias propiedades típicas de los algoritmos:

- Los pasos del algoritmo están definidos con claridad.
- Un cierto número de etapas se repiten en un bucle (while) y está claro que sólo se realizan una cantidad finita de repeticiones.
- Una vez que se introducen los números m y n en el algoritmo los cálculos se realizan de forma automática hasta producir el MCD como respuesta.
- El algoritmo puede implementarse fácilmente en una máquina programable (ordenador).

2.2. Algoritmos: forma, evaluación y complejidad

El análisis del algoritmo de Euclides nos ha dada una idea de lo que es un algoritmo, que estructura tiene y cómo podemos juzgar razonable su implementación.

Definición 2.2.1 (Algoritmo) *Un algoritmo es un procedimento de cálculo que consiste en un conjunto de instrucciones precisas. Estas instrucciones especifican una* ***sucesión finita*** *de operaciones que una vez ejecutadas proporcionan la* ***solución a un problema*** *de entre los de una clase especial de problemas.*

Mirando el algoritmo de Euclides observamos que tiene una estructura (forma) que vamos a requerir a los demás algoritmos:

Estructura de un álgoritmo.(forma)

- Los **Datos de Entrada** son los valores de inicio que deben de aportarse al algoritmo antes de su ejecución. Estos datos iniciales deben de pertenecer a unos tipos de datos predeterminados.
- Los **Datos de Salida** son los datos esperados como respuesta de la ejecución del algoritmo. En algunos casos el algoritmo puede acabar enviando un mensaje de ERROR señalando la imposibilidad de alcanzar la solución esperada.
- El **Flujo del algoritmo** consiste en una sucesión de instrucciones aritméticas que deben ser realizadas. Estas instrucciones deben de cumplir las siguientes propiedades:

  (I) Cada etapa debe estar definida de forma precisa y sin ambigüedades. Todas las posibles situaciones deben de ser tenidas en cuenta.

  (II) El flujo debe terminar después de un número finito de etapas.

  (III) El flujo debería de ser eficaz para un conjunto amplio de problemas. Debe de tenerse en cuenta su aplicabilidad en situaciones generales.

  (IV) El flujo debe de poder ser implementado en una "máquina de cálculo".

Además de las operaciones aritméticas habituales $(+, -, \times, \div, ...)$ y las funciones elementales $(\exp, \text{sen}, \cos, \sqrt{}, ...)$ en los algoritmos también se utilizan operaciones lógicas (comparaciones, asignaciones y bucles). En función de las posibilidades del lenguaje de programación a utilizar en el proceso de implantación del algoritmo se puede simplificar, más o menos, la descripción del flujo.

Cuando un algoritmo está traducido en un lenguaje de programación concreto para ser implementado en la máquina, nos referiremos a él llamándolo "programa".

El proceso de **evaluar un algoritmo** consiste en seguir su desarrollo (etapa por etapa) para asegurarnos que cumple con las propiedades que acabamos de describir. En muchos casos, basta con realizar los distintos cálculos de forma ordenada con un ejemplo concreto para detectar posibles fallos.

Frecuentemente, existen distintos algoritmos que pueden usarse para resolver un mismo problema. Es necesario disponer de criterios para comparar distintos algoritmos. Estos criterios deben de ser independientes de implementaciones particulares en computadoras concretas y de ejemplos de aplicaciones con unas condiciones iniciales concretas. También deben poder aportar afirmaciones objetivas y concretas.

Existe una teoría en Matemáticas y Ciencias de la Computación, "la teoría de la Complejidad *(Complexity theory)*", cuyo objetivo es responder a cuestiones como:

- ¿Cómo se puede cuantificar la calidad y la ejecución de un algoritmo?
- ¿Qué criterios pueden construirse para comparar algoritmos?
- ¿Se pueden mejorar los algoritmos existentes?
- ¿En que sentido se puede afirmar que un algoritmo es el "mejor" posible?
- ¿Son útiles los "mejores" algoritmos?

En esta teoría se abordan dos tipos distintos de criterios para medir la complejidad de un algoritmo: criterios relativos a la "complejidad estática" como pueden ser la longitud del algoritmo, el número de ordenes que contiene o la estabilidad y el condicionamiento de las operaciones previstas; y criterios de "complejidad dinámica" relativos al tiempo de ejecución y a requerimiento de memoria en el sentido de la adecuación de la implantación del algoritmo a las condiciones reales de espacio y tiempo.

Nosotros nos fijaremos en las siguientes tres cualidades de los algoritmos para compararlos:

(I) el número de operaciones previstas en el algoritmo,

(II) la previsibilidad de la convergencia de los distintos métodos iterativos, y

(III) la velocidad de esa convergencia (orden de convergencia).

En los apartados siguientes vamos a analizar esa dos últimas cualidades de los procesos iterativos. Ahora, para finalizar esta sección vamos a recordar el algoritmo de Horner para evaluar polinomios comparándolo con las evaluaciones usuales. Utilizamos el número de operaciones como criterio de comparación.

El algoritmo de Horner

En el ejercicio 1.6.5 proponíamos contar el número de operaciones necesarias para evaluar un polinomio. En este apartado vamos a hacer los cálculos vamos a escribir correctamente el algoritmo de Horner.

Un polinomio complejo es una combinación lineal de potencias naturales de z:

$$p(z) = a_0 + a_1 z + \cdots + a_n z^n \tag{2.1}$$

donde los coeficientes $a_k \in \mathbb{C}$ son números complejos y $a_n \neq 0$. En estas condiciones se dice que el polinomio $p(z)$ tiene grado $n \in \mathbb{N}$.

Para calcular el valor del polinomio p en un punto z_0, si usamos la relación $z_0^{k+1} = z_0.z_0^k$ y la fórmula (2.1), necesitaremos hacer $2n-1$ multiplicaciones (1 en $a_1 z_0$ y 2 multiplicaciones en cada sumando $a_k z_0^k$ para $k \geq 2$) y n sumas. En total $3n-1$ operaciones.

Sin embargo, existe un procedimiento más económico, y por lo tanto más estable para el cálculo, conocido como Esquema de Horner que alguno conoceréis como la *Regla de Rufini*:

$$p(z_0) = (\ldots((a_n z_0 + a_{n-1})\, z_0 + a_{n-2})\, z_0 + \cdots + a_1)\, z_0 + a_0.$$

Los multiplicadores de z_0 en esta fórmula se van obteniendo de forma recursiva siguiendo el esquema

	a_n	a_{n-1}	a_{n-2}	$\ldots\ldots$	a_1	a_0
z_0		$z_0 b_n$	$z_0 b_{n-1}$	$\ldots\ldots$	$z_0 b_2$	$z_0 b_1$
	b_n	b_{n-1}	b_{n-2}	$\ldots\ldots$	b_1	b_0

con $b_n = a_n$ y, $b_k = b_{k+1} z_0 + a_k$ para $k = (n-1), \ldots, 1, 0$.

Con estas operaciones, sólo se necesitan $2(n-1)$ operaciones para calcular $b_0 = p(z_0)$.

Admás, los coeficientes b_k también determinan el cociente $q(z)$ de la división de $p(z)$ entre $(z - z_0)$,

$$p(z) = q(z)(z - z_0) + p(z_0). \tag{2.2}$$

En efecto, si escribimos

$$q(z) = b_n z^{n-1} + b_{n-1} z^{n-2} + \cdots + b_2 z + b_1.$$
$$q(z)(z - z_0) = b_n z^n + b_{n-1} z^{n-1} + \cdots + b_2 z^2 + b_1 z - \left(b_n z^{n-1} z_0 + \cdots + b_2 z_0 + b_1 z_0\right),$$
$$q(z)(z - z_0) + b_0 = b_n z^n + (b_{n-1} - b_n z_0) z^{n-1} + \cdots + (b_1 - b_2 z_0) z + (b_0 - b_1 z_0).$$

Tal y como tenemos definidos los coeficientes b_k, tenemos $b_n = a_n$ y $(b_k - b_{k+1} z_0) = a_k$, de donde resulta la igualdad 2.2.

El siguiente algoritmo reproduce el esquema de Horner que hemos expuesto, y lo implementaremos en el correspondiente programa de cálculo en las sesiones de prácticas.

Algoritmo 2.2 Esquema de Horner

Datos de entrada: $a_0, a_1, \ldots, a_n$ (coeficientes del polinomio p); z
Fujo del programa:
$b_n = a_n$
`for(k=n-1;k>=0;k--){`
$b_k = b_{k+1} z + a_k$
`}`
Datos de salida: (valor de $p(z)$) b_0; (coeficientes de $q(z)$) $b_1, \ldots, b_n$

Observad que si sólo necesitamos obtener el valor de $p(z)$ podemos acelerar el algoritmo utilizando una única variable b en lugar de una lista b_k. Se inicia b con el valor de a_n y en el bucle (`for`) se hace la asignación $b = bz + a_k$.

Ejemplo 2.2.2 $z = 2$ es raíz del polinomio $p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$

	1	-4	7	-5	-2
2		2	-4	6	2
	$\mathbf{b_4 = 1}$	$\mathbf{b_3 = -2}$	$\mathbf{b_2 = 3}$	$\mathbf{b_1 = 1}$	$\mathbf{b_0 = 0}$

$p(2) = 0$ y $p(z) = (z - 2)(\mathbf{1}z^3 - \mathbf{2}z^2 + \mathbf{3}z + \mathbf{1})$.

2.3. Métodos elementales de resolución de ecuaciones no lineales

Vamos a analizar tres algoritmos sencillos para la búsqueda de soluciones de ecuaciones de la forma

$$f(x) == 0,$$

donde $f : (a, b) \to \mathbb{R}$ es una función real de variable real

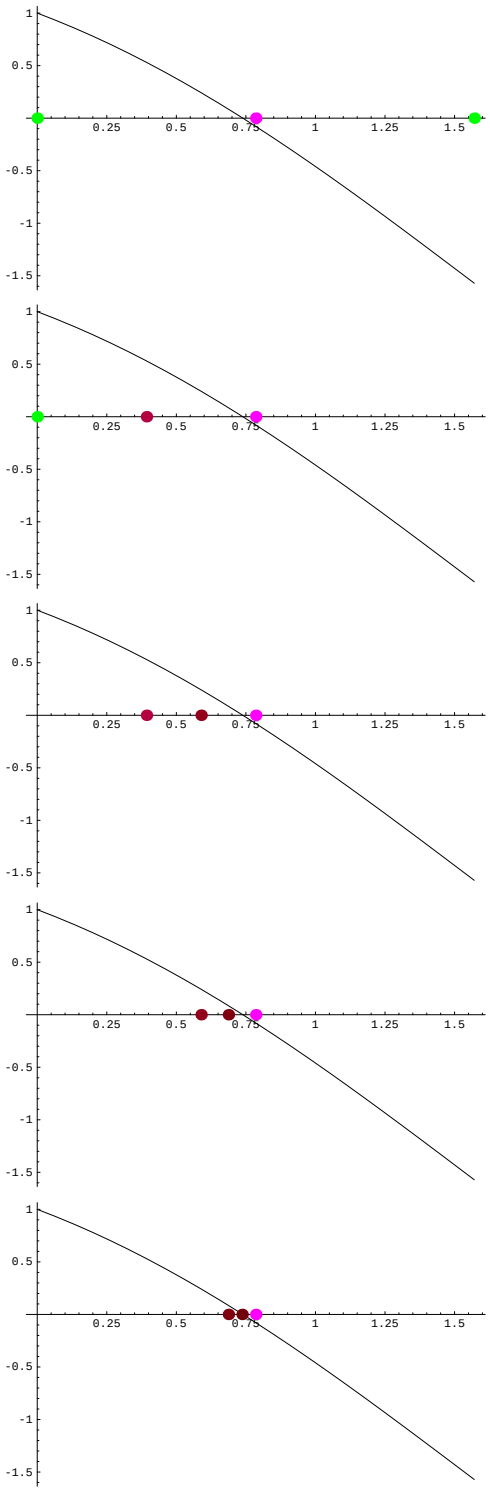
2.3.1. El Método de la Bisección

Teorema 2.3.1 (Bolzano) *Sea $f : [a, b] \to \mathbb{R}$ una función continua tal que*

$$f(a) * f(b) < 0.$$

Entonces existe $x \in [a, b]$ tal que $f(x) = 0$.

Recuerdas la demostración constructiva representada en el gráfico:



¡Escribe la demostración!

En si misma esa prueba es el algoritmo de la bisección:

Algoritmo 2.3 **Método de la bisección**

Datos de entrada: Funcion f; $a, b \in \mathbb{R}$; $precision \in \mathbb{R}$; $nmaxiter \in \mathbb{Z}$
Datos de salida: solución x $(f(x) = 0)$ o mensaje de error
Variables:
$aa, bb, h, c, u, v, w \in \mathbb{R}$ // auxiliares para operar sin cambiar los números.
Fujo del programa:
$aa = a$; $bb = b$; $u = f(aa)$; $v = f(bb)$;
`if(`$u * v > 0$`){` **Parada:** Error- No hay cambio de signo`}`
`for(k=1;k<=nmaxiter;k=k+1){`
$h = 0.5 * (bb - aa)$; $c = aa + h$; $w = f(c)$;
`if((`$|h| < precision$`) o (`$|w| < precision$`)){` **Parada:** Solución en $x = c$ `}`
`if(`$w * u > 0$`){` $aa = c$; $u = w$; `}`
`else{` $bb = c$; $v = w$;`}`
`}`
Parada: Error- No hay convergencia

- las sucesiones $a_k = aa$ y $b_k == bb$ (en la etapa k) son monótonas acotadas, encierran la sucesión $c_k = c$ (en la etapa k), y convergen hacia una solución $\overline{x}$ de la ecuación $f(x) == 0$.
- El algoritmo de la bisección proporciona en cada etapa k una aproximación $c_k = c$ a la raíz $\overline{x}$ de manera que

$$|\overline{x} - c_k| \leq \frac{b-a}{2^{k+1}}.$$

En el algoritmo de la bisección tenemos asegurada la convergencia del método aunque la velocidad de las iteradas es constante: en cada etapa se reduce el error en la aproximación a la mitad.

Ejemplo 2.3.2 En la tabla que sigue se pueden observar valores de las 37 primeras iteraciones del método de la bisección aplicado a la función $f(x) = e^x - x^2 - x - 1$ en el intervalo [1,2], tomando $precision = 10^{-10}$.

k= 1	aa= 1.0	bb= 2.0	u=-0.28171817154095447	v=0.3890560989306504
k= 2	aa= 1.5	bb= 2.0	u=-0.2683109296619355	v=0.3890560989306504
k= 3	aa= 1.75	bb= 2.0	u=-0.05789732399427017	v=0.3890560989306504
k= 4	aa= 1.75	bb= 1.875	u=-0.05789732399427017	v=0.13019412033011246
k= 5	aa= 1.75	bb= 1.8125	u=-0.05789732399427017	v=0.02808641188198635
k= 11	aa= 1.79296875	bb= 1.7939453125	u=-4.456149959981559E-4	v=9.443070728494263E-4
k= 12	aa= 1.79296875	bb= 1.79345703125	u=-4.456149959981559E-4	v=2.4886798603152016E-4
k= 13	aa= 1.793212890625	bb= 1.79345703125	u=-9.849297435415849E-5	v=2.4886798603152016E-4
k= 14	aa= 1.793212890625	bb= 1.7933349609375	u=-9.849297435415849E-5	v=7.515763303089784E-5
k= 15	aa= 1.79327392578125	bb= 1.7933349609375	u=-1.1675138180677891E-5	v=7.515763303089784E-5

k= 21	aa= 1.7932815551757812	bb= 1.7932825088500977	u=-8.21858566979472E-7	v=5.348177927189113E-7
k= 22	aa= 1.7932820320129395	bb= 1.7932825088500977	u=-1.4352084320989889E-7	v=5.348177927189113E-7
k= 23	aa= 1.7932820320129395	bb= 1.7932822704315186	u=-1.4352084320989889E-7	v=1.9564836062357926E-7
k= 24	aa= 1.7932820320129395	bb= 1.793282151222229	u=-1.4352084320989889E-7	v=2.606373072921997E-8
k= 25	aa= 1.7932820916175842	bb= 1.793282151222229	u=-5.8728563345766815E-8	v=2.606373072921997E-8
k= 32	aa= 1.7932821325957775	bb= 1.7932821330614388	u=-4.338631676148452E-10	v=2.2857626902350603E-10
k= 33	aa= 1.7932821328286082	bb= 1.7932821330614388	u=-1.0264278316185482E-10	v=2.2857626902350603E-10
k= 34	aa= 1.7932821328286082	bb= 1.7932821329450235	u=-1.0264278316185482E-10	v=6.296652088622068E-11
k= 35	aa= 1.7932821328868158	bb= 1.7932821329450235	u=-1.9838353182421997E-11	v=6.296652088622068E-11
k= 36	aa= 1.7932821328868158	bb= 1.7932821329159196	u=-1.9838353182421997E-11	v=2.156363976268949E-11

El método de encuentra $c = 1.7932821329013677$, $f(c) = 8.633094239485217E - 13$, como raíz, en 36 etapas (pasos).

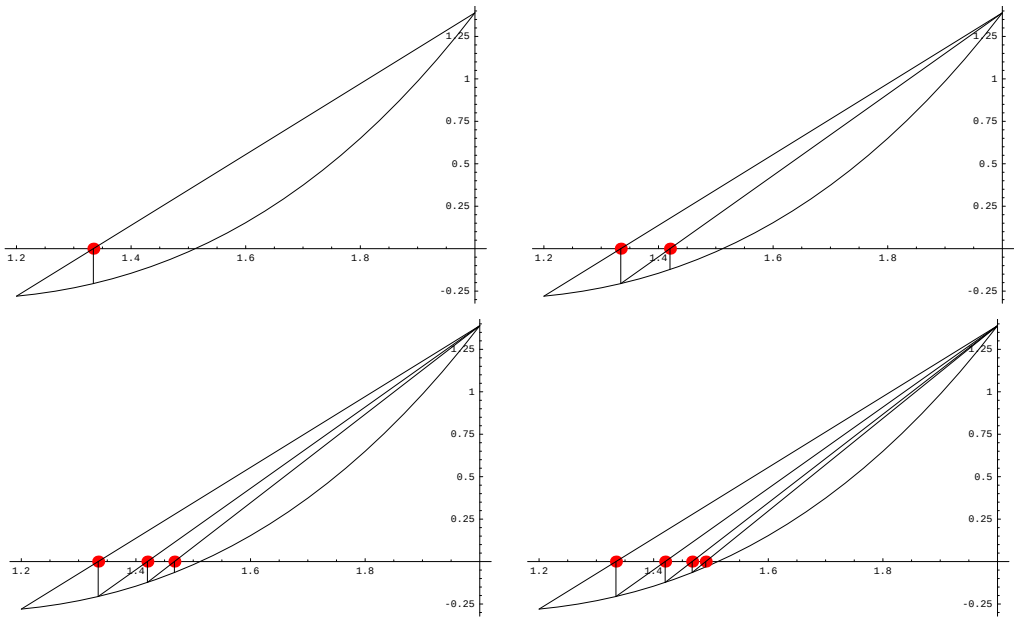
En los dos apartados siguientes vamos a ver dos intentos de aceleración de esta convergencia: los métodos de la "regla falsa" y de "Newton".

2.3.2. El método de la "Regula Falsi"

El método de la «**Regula Falsi** » (regla falsa), es una modificación del método de la bisección con la esperanza de obtener una aproximación de la raíz de forma más rápida.

Consiste en *utilizar el punto de corte c de la secante que une los extremos de la curva* $y = f(x)$ *en* a *y* b en vez del punto medio, para dividir el intervalo y después iterar el proceso quedándonos con los subintervalos en los que f cambie de signo.

En los siguientes gráficos podemos observar distintas etapas de este método:



Para describir el algoritmo, recordar que la ecuación de la recta que pasa por los puntos (a, u) y (b, v) es

$$y = u + \frac{v - u}{b - a}(x - a).$$

Si ahora hacemos $y = 0$ en esa expresión para determinar la abscisa de corte de la recta con el eje de abscisas se tiene

$$x = a - u\,\frac{b-a}{v-u}.$$

Modificando el algoritmo del método de la bisección en la definición de c tomando el corte de la secante a la curva $y = f(x)$ en los extremos de los intervalos con el eje de abscisas en vez de considerar el centro del intervalo, obtenemos el siguiente algoritmo:

(Enmarcada sobre fondo gris, está la modificación efectuada)

Algoritmo 2.4 Algoritmo de la "regula falsi"

Datos de entrada: Funcion f; $a, b \in \mathbb{R}$; $precision \in \mathbb{R}$; $nmaxiter \in \mathbb{Z}$
Datos de salida: solución x $(f(x) = 0)$ o mensaje de error
Variables:
$aa, bb, h, c, u, v, w \in \mathbb{R}$ // auxiliares para operar sin cambiar los números.
Fujo del programa:
$aa = a$; $bb = b$; $u = f(aa)$; $v = f(bb)$;
`if(`$u * v > 0$`){` **Parada:** Error- No hay cambio de signo`}`
`for(k=1;k<=nmaxiter;k=k+1){`
$h = -u * (bb - aa)/(v - u);$ $c = aa + h$; $w = f(c)$;
`if((`$|h| < precision$`) o (`$|w| <$`)){` **Parada:** Solución en $x = c$ `}`
`if(`$w * u > 0$`){` $aa = c$; $u = w$; `}`
`else{` $bb = c$; $v = w$;`}`
`}`
Parada: Error- No hay convergencia

Ejemplo 2.3.3 Retomando el ejemplo 2.3.2, $f(x) = e^x - x^2 - x - 1$ en [1,2], y aplicando el método de la regula falsi con la misma $precision = 10^{-10}$, tenemos

k= 1	aa= 1.0	bb= 2.0	u=-0.28171817154095447	v=0.3890560989306504
k= 2	aa= 1.4199895314155169	bb= 2.0	u=-0.29928267006193354	v=0.3890560989306504
k= 3	aa= 1.6721721622963694	bb= 2.0	u=-0.14461267362264119	v=0.3890560989306504
k= 4	aa= 1.7610064028149435	bb= 2.0	u=-0.043859961270654724	v=0.3890560989306504
k= 5	aa= 1.7852195260508816	bb= 2.0	u=-0.01133990799920781	v=0.3890560989306504
k= 11	aa= 1.7932804127143416	bb= 2.0	u=-2.447094586077725E-6	v=0.3890560989306504
k= 12	aa= 1.7932817129360898	bb= 2.0	u=-5.974324321922353E-7	v=0.3890560989306504
k= 13	aa= 1.793282030371081	bb= 2.0	u=-1.4585651553211676E-7	v=0.3890560989306504
k= 14	aa= 1.7932821078692915	bb= 2.0	u=-3.5609233561828546E-8	v=0.3890560989306504
k= 15	aa= 1.7932821267896093	bb= 2.0	u=-8.693593622766116E-9	v=0.3890560989306504
k= 16	aa= 1.793282131408792	bb= 2.0	u=-2.1224435542421816E-9	v=0.3890560989306504
k= 17	aa= 1.7932821325365136	bb= 2.0	u=-5.181701734358057E-10	v=0.3890560989306504
k= 18	aa= 1.7932821328118338	bb= 2.0	u=-1.26505694808543E-10	v=0.3890560989306504
k= 19	aa= 1.7932821328790503	bb= 2.0	u=-3.088485023283738E-11	v=0.3890560989306504

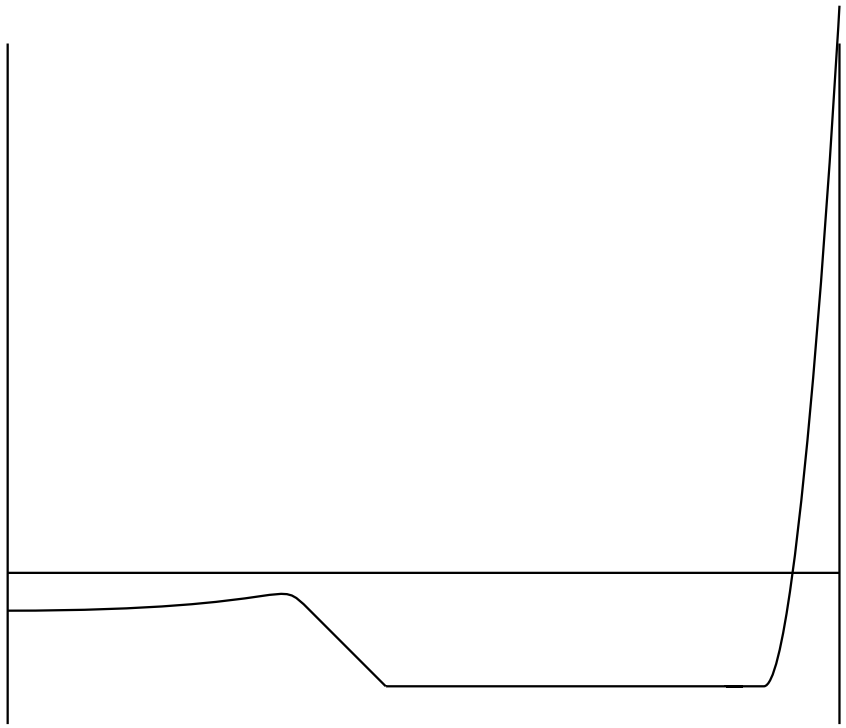
Se obtiene la misma solución que con el método de la bisección pero ahora sólo en 19 pasos.

A la vista de este ejemplo podríamos pensar que el algoritmo de la regula falsi es más rápido que el método de la bisección. En general, ese no es el caso

Ejemplo 2.3.4 Si consideramos la función $g(x) = x^3 - x - 1$ que también cambia de signo en [1,2], y tomamos la *precision* $= 10^{-10}$, el método de la bisección nos da la raíz $c = 1.324717957242683$ con $g(c) = -8.79785133633959E - 12$ en 37 etapas.

Si aplicamos la regula falsi a esta misma función, en el mismo intervalo, y con la misma precisión, obtenemos la misma raíz en 31 etapas, que sólo es unas cuantas iteraciones menos que el de la bisección.

Ejercicio 2.3.1 Trabajando sobre la siguiente gráfica. Dad una estimación del número de etapas que necesitan los métodos de la bisección y de la regula falsi para alcanzar buenas estimaciones de la raíz. Observad que en este caso el método de la bisección parece ser bastante más rápido que el de la regula falsi.



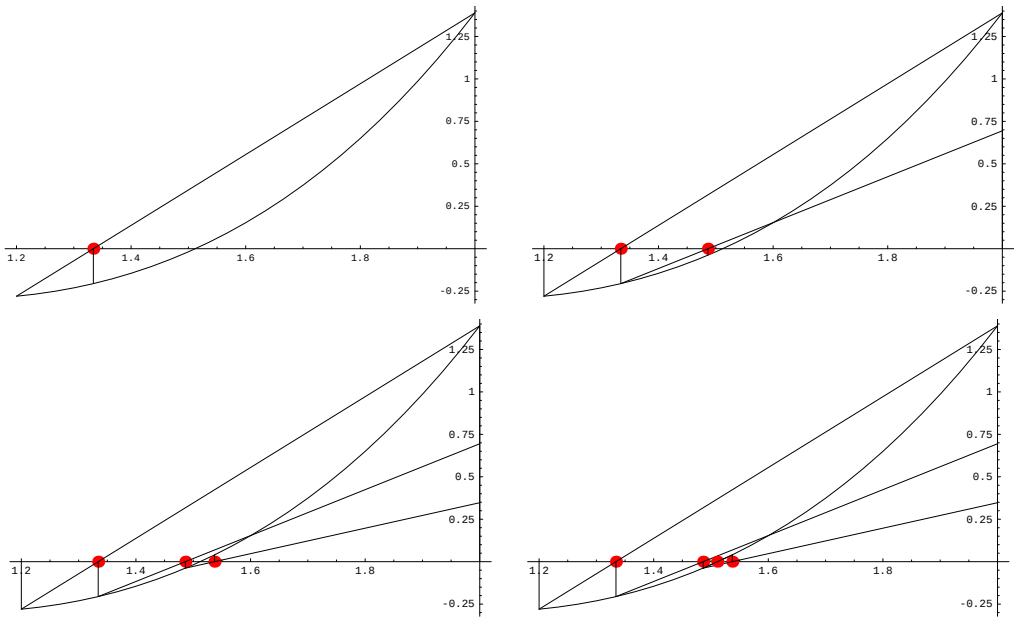
Observando la gráfica del ejercicio podemos conjeturar (sin equivocarnos demasiado) que el ralentizamiento de la regula falsi se debe a la presencia de regiones donde la curva tiene poca pendiente donde la secante aproxima mal a la curva, para pasar al final a una región con mucha pendiente donde las secantes aproximan bien a la curva.

Para evadir esta situación se puede utilizar el siguiente "truco" que consiste en reducir la pendiente de las secante en las etapas en las que se sea consciente de que podemos estar ralentizandonos.

La modificación esencialmente consiste en guardar memoria del signo de f en el punto de corte de la secante con el eje de abscisas y en caso de que el signo se repita en la siguiente iteración, se inclinará más la secante buscando una mejor aproximación del cero de f.

En la primera etapa, se utiliza $f(a)$ para comparar con el signo de $f(c)$ en el punto de corte, c, de la secante con el eje de abscisas.

Gráficamente hacemos lo siguiente:



(Enmarcadas sobre fondo gris, están la modificaciones efectuadas sobre el método de la regula falsi)

Algoritmo 2.5 **"Regula falsi" modificado**

Datos de entrada: Funcion f; $a, b \in \mathbb{R}$; $precision \in \mathbb{R}$; $nmaxiter \in \mathbb{Z}$
Datos de salida: solución x $(f(x) = 0)$ o mensaje de error
Variables:
$aa, bb, h, c, u, v, w,$ $w1 \in \mathbb{R}$ // auxiliares para operar sin cambiar los números.
Fujo del programa:
$aa = a$; $bb = b$; $u = f(aa)$; $v = f(bb)$;
`if(`$u * v > 0$`){` **Parada:** Error- No hay cambio de signo`}`
$w1 = u$
`for(k=1;k<=nmaxiter;k=k+1){`
$h = -u * (bb - aa)/(v - u)$; $c = aa + h$; $w = f(c)$;
`if((`$|h| < precision$`) o (`$|w| <$`)){` **Parada:** Solución en $x = c$ `}`
`if(`$w * u > 0$`){`
`if((`$w * w1 > 0$`)){` $v = 0.5 * v$`}`
$aa = c$; $u = w$; $w1 = w$ `}`
`else{`
`if((`$w * w1 > 0$`)){` $u = 0.5 * u$`}`
$bb = c$; $v = w$; $w1 = w$ `}`
`}`
Parada: Error- No hay convergencia

Ejemplo 2.3.5 Volviendo a retomar los ejemplos anteriores, para $f(x) = e^x - x^2 - x - 1$ en [1,2], el método de la regula falsi modificado proporciona la raíz $c = 1.7932821329006305$ $f(c) = -1.8562928971732617E - 13$ en sólo 8 etapas.

Para la función $g(x) = x^3 - x - 1$ que también cambia de signo en [1,2], la modificación de la regula falsi proporciona la raíz $c = 1.324717957244746$ $f(c) = 2.220446049250313E - 16$ en 7 etapas.

Con el "truco" propuesto, podemos pensar en que vamos a tener mejor convergencia.

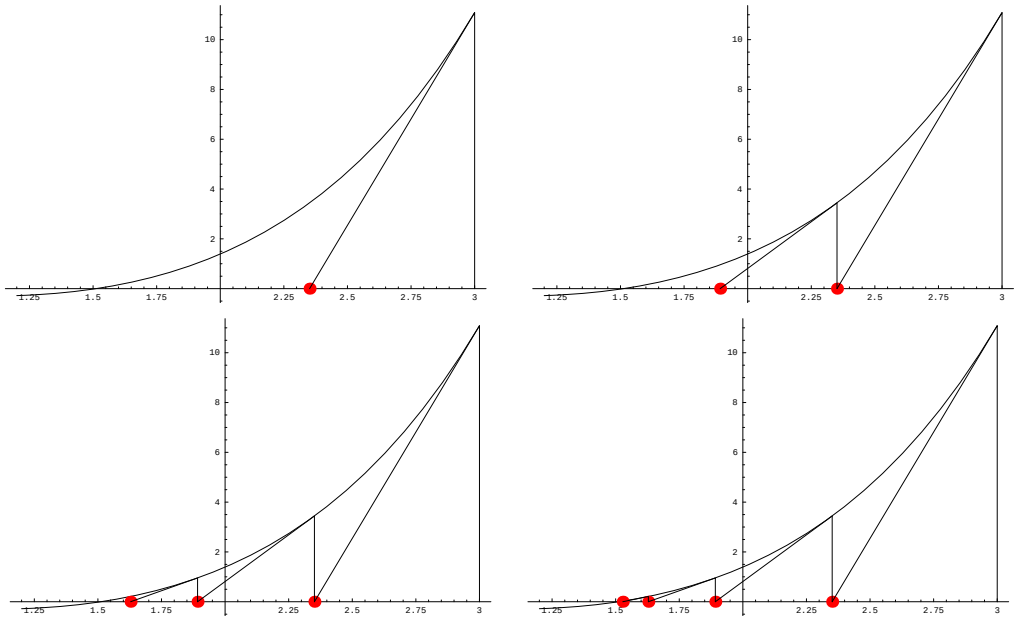
2.3.3. Método de Newton

Una variación a los métodos anteriores aparece cuando utilizamos rectas tangentes a la curva $y = f(x)$ para aproximarnos a la raíz. Este es el conocido como método de Newton.

Consiste en iterar el proceso siguiente:

- Dada una **aproximación** xi de la raíz de f, consideramos $g(x) = f(xi) + f'(xi)(x - xi)$ la ecuación de la tangente a $y = f(x)$ en xi.
- resolvemos la ecuación lineal $g(x) == 0$, tomando $h = -\frac{f(xi)}{f'(xi)}$ y $xf = xi + h$.
- la solución xf de la ecuación aproximada, será la **nueva aproximación** de la raíz de la ecuación inicial.

Gráficamente se puede representar como sigue:



El algoritmo es muy simple:

Algoritmo 2.6 **Método de Newton**

Datos de entrada: Función f; $x_0 \in \mathbb{R}$; $precision \in \mathbb{R}$; $nmaxiter \in \mathbb{Z}$
Datos de salida: solución x $(f(x) = 0)$ o mensaje de error
Variables:
$xi = x_0$ // la aproximación inicial.
fi; dfi;// Para almacenar el valor de f y el de f′ en la aproximación.
$hi = 1 + precision$; // Para almacenar el incremento de xi, con un valor inicial grande.
Fujo del programa:

```
for(k=1;k<=nmaxiter;k++){
  fi = f(xi);
  if((|fi| < precision) o (|hi| < precision)){   Parada: Raíz en xi }
  dfi = f'(xi);
  if(dfi==0){
    Parada:Error: derivada nula    }
  hi = -fi/dfi;
  xi = xi + hi;    // volvemos a iterar Newton con este nuevo valor de xi
  }
```

Parada: Error: No hay convergencia en $nmaxiter$ iteraciones

Terminamos la sección observando el comportamiento del algoritmo en algunos ejemplos.

Ejemplo 2.3.6 Volviendo a retomar los ejemplos anteriores, para $f(x) = e^x - x^2 - x - 1$ en [1,2], el método de Newton empezando en $xi = 1$ como aproximación inicial, proporciona la raíz $c = 1.793282132894812$ $f(c) = -8.462119893692943E-12$ en 30 etapas. En este caso el método más rápido ha sido el de la regula falsi modificado.

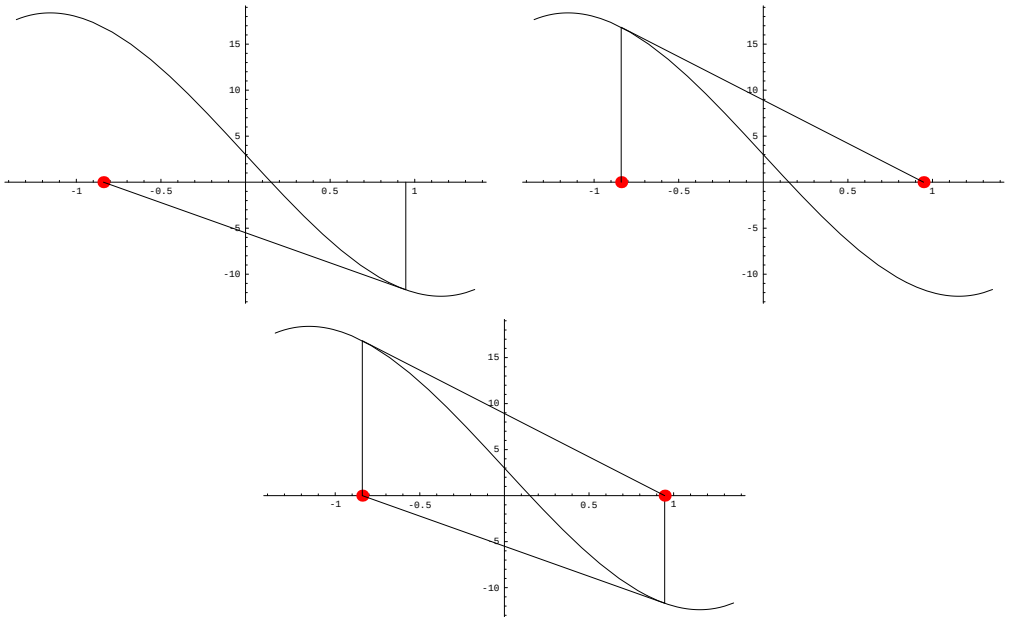
Con la función $g(x) = x^3 - x - 1$ y con $xi = 1$ como aproximación inicial, el método de Newton proporciona la raíz $c = 1.3247179572447898$ $f(c) = 1.865174681370263E - 13$ en 6 etapas. En este caso el método de Newton es el más eficaz de los cuatro estudiados.

- Los distintos métodos introducidos tienen diferente comportamiento según las ecuaciones propuestas en los ejemplos.
- En principio, debemos admitir la coexistencia de los diferentes métodos de resolución de ecuaciones estudiados, habida cuenta de que ninguno de los cuatro métodos estudiados se puede calificar de "mejor" método que los otros.

El método de Newton es un ejemplo de una iteración funcional, tal y como señalaremos en la siguiente sección. Para estas iteraciones vamos a tener resultados de convergencia local que nos dirán que **el método de Newton en general converge más rápido que los otros supuesto que partimos de una "buena aproximación" a la raíz $\overline{x}$ de f y que $f'(\overline{x}) \neq 0$.**

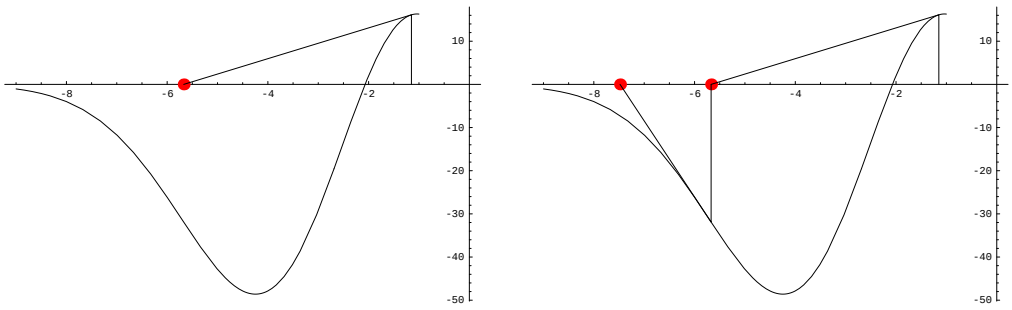
En los siguientes ejemplos gráficos podemos observar que las iteradas del método de Newton pueden no ser convergentes e incluso pueden dar lugar a sucesiones periódicas (sucesiones en las que a partir de un momento, todos los términos se repiten de forma periódica).

Ejemplo 2.3.7 Puntos periódicos

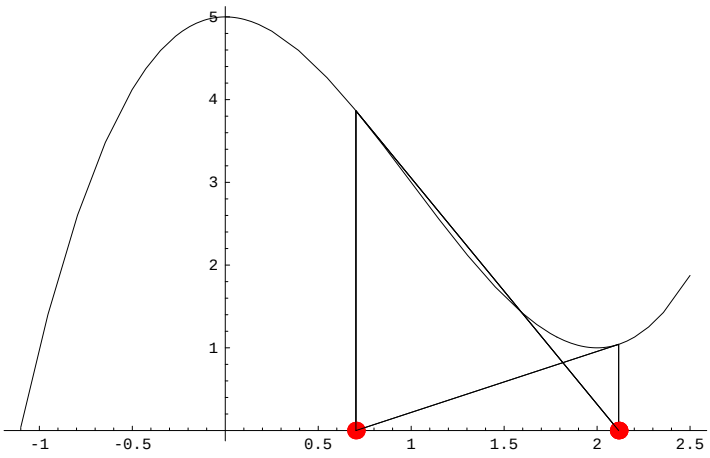


Cuando la derivada de f se anula, aparecen problemas:

Ejemplo 2.3.8 Sucesión divergente (asintotas)



Ejemplo 2.3.9 Extremos relativos (empezamos cerca del mínimo)



Referencias para la sección

- Método de la Bisección: Libro de Burden-Faires [2], sección 2.1, algoritmo 2.1.
- Método de la regula falsi: Libro de Burden-Faires [2], sección 2.3, algoritmo 2.5.
- Método de Newton: Libro de Burden-Faires [2], sección 2.3, algoritmo 2.3.

2.4. Iteración de punto fijo

Definición 2.4.1 *Un punto fijo de una función $f : C \to C$ es cualquier punto $\overline{x} \in C$ para el cual*

$$f(\overline{x}) = \overline{x}.$$

En esta sección vamos a estudiar los **problemas de punto fijo**, es decir, problemas de resolución de ecuaciones

$$f(x) == x,$$

donde $f : C \to C$ es una función del conjunto C en si mismo. Vamos a poner ejemplos de funciones reales de variable real aunque el resultado principal de la sección lo probaremos en el contexto de espacios métricos completos.

Los problemas de búsqueda de raíces de funciones $g(x) == 0$ son equivalentes a los de búsqueda de puntos fijos $f(x) == x$, considerando funciones auxiliares:

- Si $f(x)$ tiene un punto fijo en $\overline{x}$, entonces $g(x) = f(x) - x$ tiene una raíz en $\overline{x}$.
- Reciprocamente, si g tiene una raíz en $\overline{x}$, entonces $f(x) = h(x)g(x) + x$ tiene un punto fijo en $\overline{x}$.

Vamos a buscar puntos fijos viendo el comportamiento (la dinámica) de las iteradas de funciones, es decir, sucesiones de la forma

$$x_n = f(x_{n-1}).$$

A estas sucesiones se les llama **órbitas del punto $x[0]$ y la función f**,

Proposición 2.4.2 *Si f es una función continua y x_n converge, entonces su límite $\overline{x}$ es un punto fijo de f.*

La prueba de la proposición es la siguiente cadena de igualdades

$$f(\overline{x}) = \lim_n f(x_n) = \lim_n x_n = \lim_n x_{n+1} = \overline{x}.$$

Esta técnica de considerar sucesiones de iteradas, $x_n = f(x_{n-1})$, para buscar puntos fijos, recibe el nombre de **iteración de punto fijo** o **iteración funcional**. El algoritmo correspondiente es:

Algoritmo 2.7 **Iteración de punto fijo**

Datos de entrada: Función f; $x_0 \in \mathbb{R}$; $precision \in \mathbb{R}$; $nmaxiter \in \mathbb{Z}$
Datos de salida: solución x $(f(x) = x)$ o mensaje de error
Variables:
$xx = x_0$ // los puntos de la órbita.
$fx = f(xx)$; // el siguiente punto de la órbita f.
Fujo del programa:

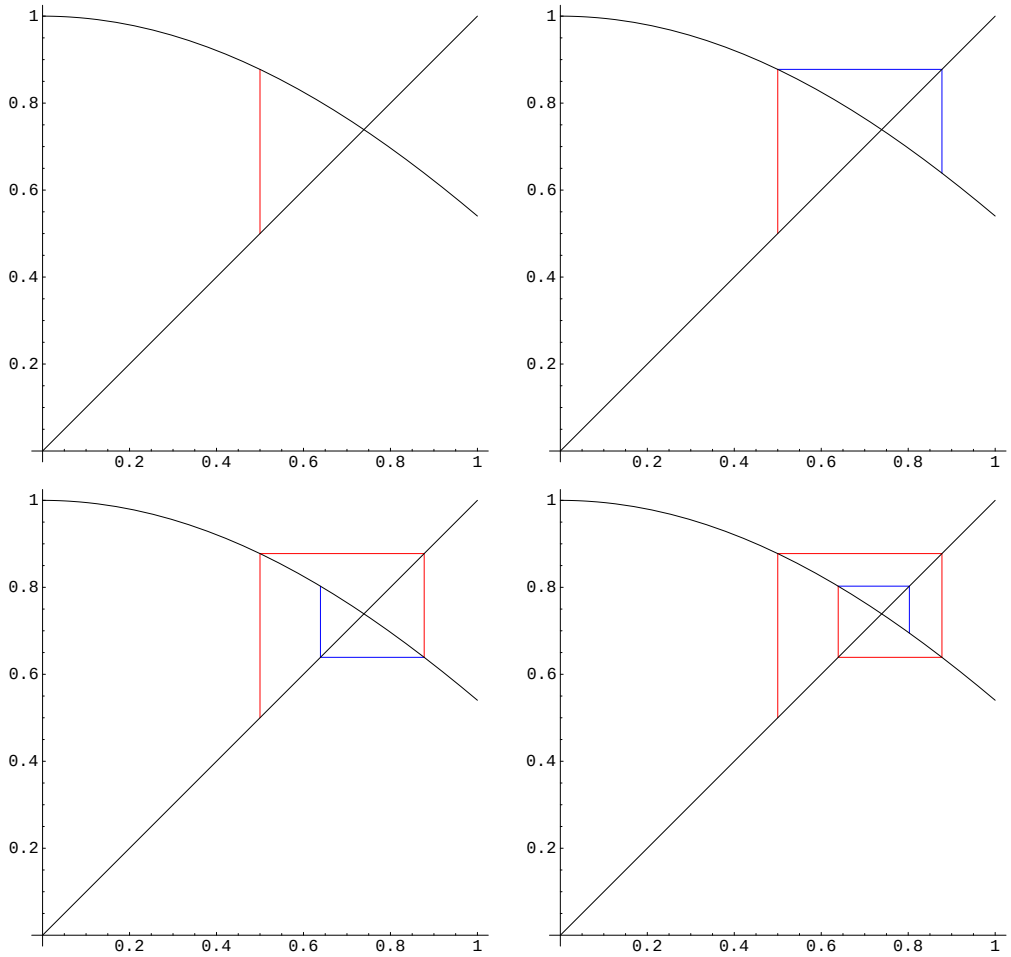
```
for(k=1;k<=nmaxiter;k++){
  if(|xx − fx| < precision){   Parada: Punto fijo en xx }
  xx = fx;
  fx = f(xx);    // el siguiente punto de la órbita f
  }
```

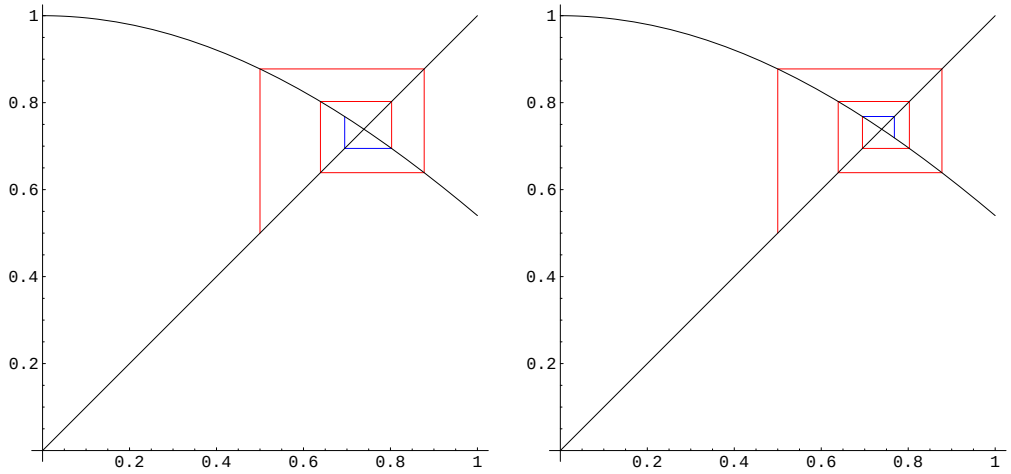
Parada: Error: No hay convergencia en $nmaxiter$ iteraciones

Ejemplo 2.4.3 Con la calculadora de mano, configurada para trabajar en radianes, si consideramos la función $\cos(x)$ y calculamos la órbita de cualquier número obtenemos un/el punto fijo del cos :

$$\cos(0.739085133) = 0.739085133.$$

Gráficamente, comenzando en 0.5 se tiene:





¿Recordáis resultados del curso "Analisis I" que garanticen la existencia de puntos fijos y su unicidad?

Echad un vistazo al Teorema 2.2 del libro de Burden-Faires [2] y seguro que recordáis una aplicación del teorema de Bolzano y otra de la fórmula de los incrementos finitos.

El siguiente teorema da una condición suficiente para la convergencia de las iteraciones de punto fijo. Vamos e enunciarlo para funciones definidas en espacios métricos completos (X, d), que contiene como caso particular a $\mathbb{R}$, $\mathbb{R}^n$, a los espacios de Banach (espacios normados completos) $(X, \|\ \|)$ y a sus subconjuntos cerrados.

Definición 2.4.4 *Sea (X, d) un espacio métrico completo, se dice que una función $f : X \to X$, es una función* ***contractiva*** *cuando existe una constante $c, 0 < c < 1$, verificando :*

$$d(f(x), f(y)) \leq c\, d(x, y) \qquad \text{para cada } x \text{ e } y \text{ en } X.$$

Teorema 2.4.5 (Teorema del Punto Fijo de Banach) *Sea (X, d) un espacio métrico completo y $f : X \to X$ una función* ***(contractiva)*** *tal que existe una constante $c, 0 < c < 1$, verificando :*

$$d(f(x), f(y)) \leq c\, d(x, y) \qquad \text{para cada } x \text{ e } y \text{ en } X.$$

Entonces la ecuación $f(x) == x$ tiene una única solución $xs \in X$.

Esta solución se encuentra como el límite de la sucesión de iteradas

$$xs = \lim_n x(n), \qquad x(n) = f(x(n-1)).$$

para un $x(0)$ en X arbitrario.

Además, el error en cada etapa está acotado por

$$e(n) = d(x(n), xs) \leq \frac{c^n}{1-c} d(x(1), x(0))$$

Demostración:

Ideas que intervienen

- *Como f es contractiva si tiene un punto fijo este es único.*

  En efecto, si $f(x) = x$ y $f(y) = y$, entonces

  $$d(x, y) = d(f(x), f(y)) \leq cd(x, y),$$

  y como $0 < c < 1$, esto sólo es posible si $d(x, y) = 0$, e. d. si $x = y$.

- *Como f es contractiva, f es continua. Si la iterada funcional $x_n = f(x_{n-1})$ converge, su límite es el punto fijo de f (Proposición 2.4.2).*
- *Como f es contractiva las iteradas funcionales x_n son sucesiones de Cauchy y en consecuencia son convergentes ya que X es un* ***espacio métrico completo****.*

(I) $d(x_n, x_{n-1}) \leq c\, d(x_{n-1}, x_{n-2}) \leq c^2\, d(x_{n-1}, x_{n-3}) \leq ... \leq c^{n-1} d(x_1, x_0)$.

(II) $d(x_{n+p}, x_n) \leq d(x_{n+p}, x_{n+p-1}) + d(x_{n+p-1}, x_n) \leq ...$
$d(x_{n+p}, x_{n+p-1}) + d(x_{n+p-1}, x_{n+p-2}) + ... + d(x_{n+1}, x_n) \leq$
$(c^{n+p-1} + c^{n+p-2} + ... + c^n) d(x_1, x_0) = \frac{c^n - c^{n+p}}{1-c} d(x_1, x_0) \to 0.$

□

La **fórmula de los incrementos finitos** nos ofrece condiciones para que una función sea contractiva y se cumpla el teorema del punto fijo:

$$|f(x) - f(y)| = |f'(x + t(y - x))||x - y| \leq c|x - y|,$$

donde $(0 < t < 1)$, y $|f'(s)| \leq c$ para todo $s \in [a, b]$ $(x + t(y - x) \in [a, b])$.

Corolario 2.4.6 *Sea $f : [a, b] \to [a, b]$ una función derivable para la que existe una constante $c, 0 < c < 1$ tal que:*

$$|f'(x)| \leq c \qquad \text{para todo } x \in [a, b].$$

Entonces la ecuación $f(x) == x$ tiene una única solución xs en $[a, b]$. Esta solución es el límite de la sucesión de iteradas

$$x_n = f(x_{n-1})$$

definida a partir de cualquier punto $x_0 \in [a, b]$. Además:

$$e_n = |x_n - xs| \leq \frac{c^n}{1 - c}|x_1 - x_0|.$$

Ejemplo 2.4.7 (Ejemplos 3 y 4 del apartado 2.2 de Burden-Faires [2]) La única raíz de $p(x) = x^3 + 4x^2 - 10$ en el intervalo $[1, 2]$ se puede convertir en punto fijo $x == g(x)$, de una función g de diferentes formas.

Vamos a relacionar 5 elecciones distintas de g, y describiremos el comportamiento de las iteradas funcionales empezando en $x_0 = 1.5$

(I) $x = g_1(x) = x - x^3 - 4x^2 + 10$

$x_n = g_1(x_{n-1})$ es divergente.

g_1 no es contractiva, $|g'(x)| > 1$ en todo $x \in [1, 2]$.

(II) $x = g_2(x) = \left(\frac{10}{x} - 4x\right)^{\frac{1}{2}}$

g_2 no lleva el intervalo [1,2] al intervalo [1,2]. La iterada x_3 no está definida.

(III) $x = g_3(x) = \frac{1}{2}(10 - x^3)^{\frac{1}{2}}$

$x_n = g_3(x_{n-1})$ es convergente.

g_3 es estrictamente decreciente en [1,2]. Lleva el intervalo [1,1.5] al intervalo [1.28,1.5]. Además como $|g_3'(x)| \leq |g_3'(1.5)| \approx 0.66 < 1$ para todo $x \in [1.28, 1.5]$, podemos concluir que g_3 es contractiva en ese intervalo.

(IV) $x = g_4(x) = \left(\frac{10}{4+x}\right)^{\frac{1}{2}}$

$x_n = g_4(x_{n-1})$ es convergente, y converge más rápido que la anterior.

g_4 es decreciente, lleva el intervalo [1,2] al intervalo [1,2] y $|g_4'(x)| = \left|\frac{5}{\sqrt{10}\,5^3}\right| < 0.15$ para todo $x \in [1,2]$.

g_4 es contractiva con constante menor que la de g_3. Esta es la razón por la iterada para g_4 converge más rápidamente que la de g_4.

(V) $x = g_5(x) = x - \frac{x^3+4x^2-10}{3x^2+8x}$.

La iterada funcional para g_5 es la iterada del método de Newton para $p(x)$ y es más rápida que las dos iteradas anteriores. Enseguida vamos a ver porqué.

2.4.1. Atractores y repulsores

Sea $f : [a,b] \to \mathbb{R}$ es una función derivable con derivada continua y $\overline{x} \in [a,b]$ un punto fijo de f $(f(\overline{x}) = \overline{x})$

El tamaño de la derivada $|f'(\overline{x})|$ indica el comportamiento de las iteradas funcionales en las próximidades de $\overline{x}$.

Proposición 2.4.8 *Si $|f'(\overline{x})| < 1$ entonces, en un entorno de $\overline{x}$ se cumplen las hipótesis del teorema del punto fijo. Se dice que $\overline{x}$ es un punto **atractor** (o **sumidero**) de f.*

Las iteradas $x_n = f(x_{n-1})$ convergen a $\overline{x}$, para x_0 próximo a $\overline{x}$.

DEMOSTRACIÓN:
Ideas que intervienen

- *$|f'(\overline{x})| < c < 1$ y f' continua $\Rightarrow$ que existe $\delta > 0$ tal que $|f'(x)| < c$ para $x \in [\overline{x}-\delta, \overline{x}+\delta]$.*
- *$f : [\overline{x}-\delta, \overline{x}+\delta] \to [\overline{x}-\delta, \overline{x}+\delta]$ En efecto:*

$$|f(x) - \overline{x}| = |f(x) - f(\overline{x})| = |f'(\xi)||x - \overline{x}| < c\,\delta < \delta.$$

(ξ está entre x y $\overline{x}$, $\xi \in [\overline{x}-\delta, \overline{x}+\delta]$)

- *Se cumplen las hipótesis del teorema del punto fijo para f definida en $[\overline{x}-\delta, \overline{x}+\delta]$.*

□

Ejemplo 2.4.9 (El método de Newton) El método de «**Newton** » consiste en iterar el proceso siguiente:

- Dada una raíz $\overline{x}$ de una función derivable f, $(f(\overline{x}) = 0)$. Si x_n **es una aproximación de** $\overline{x}$ consideramos $aux(x) = f(xi) + f'(xi)(x - xi)$ la ecuación de la tangente a $y = f(x)$ en x_n, resolvemos la ecuación lineal $aux(x) == 0$, y tomamos la solución como nueva aproximación $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$.

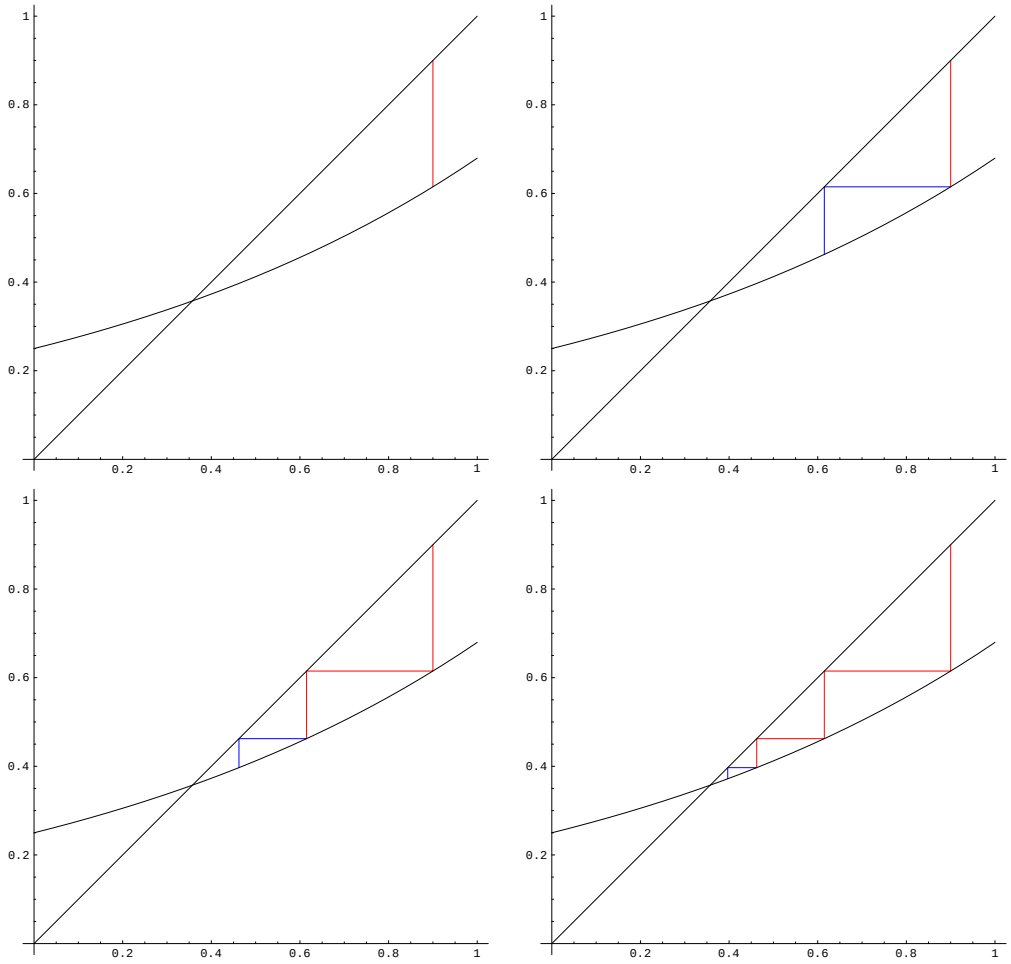
- Considerando la función
$$g(x) = x - \frac{f(x)}{f'(x)}.$$
Esperamos encontrar $\overline{x}$ como límite de $x_{n+1} = g(x_n)$, e.d., como punto fijo de g.

- Si f es dos veces derivable con f'' continua y $f'(\overline{x}) \neq 0$, g es derivable en un entorno de $\overline{x}$ y
$$g'(\overline{x}) = 0 (< 1) \text{ y } g''(\overline{x}) = \frac{f''(\overline{x})}{f'(\overline{x})}.$$
$\overline{x}$ es un atractor.

Como corolario de la proposición 2.4.8 tenemos el siguiente teorema de convergencia local para el método de Newton.

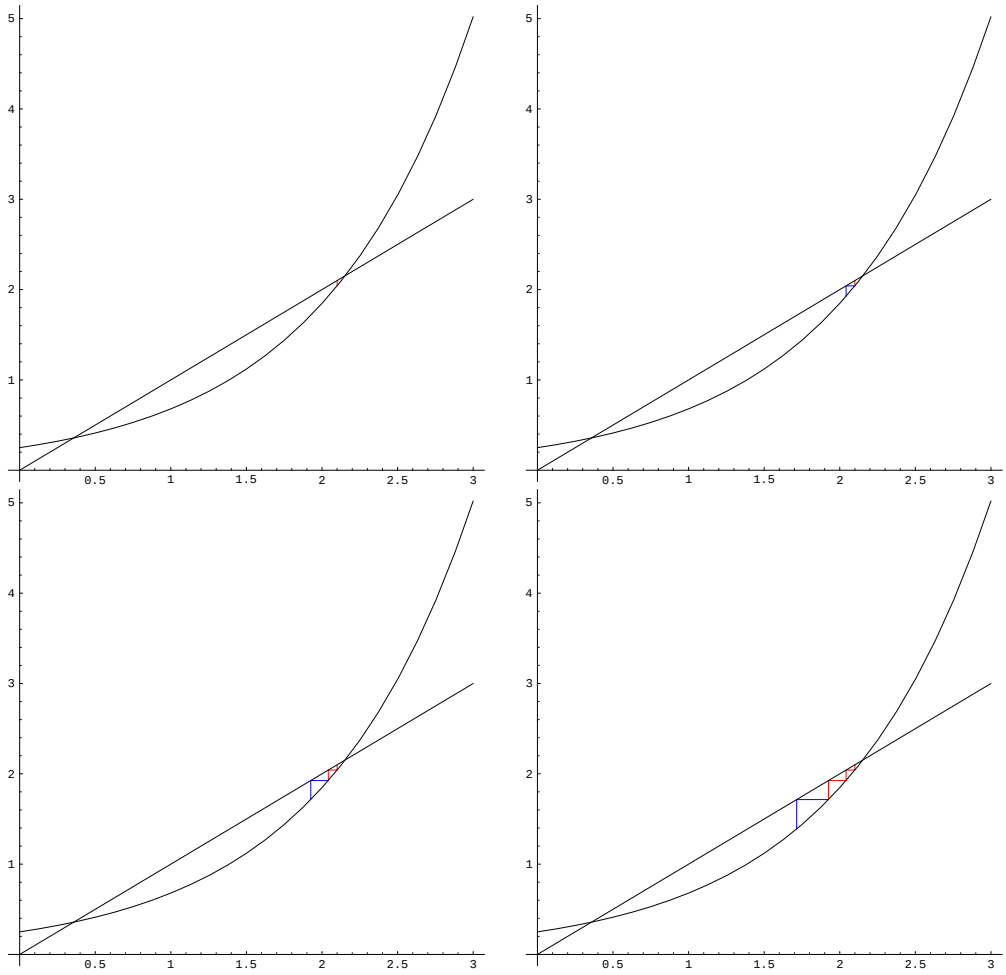
Proposición 2.4.10 *Sea $f : (a, b) \to \mathbb{R}$ una función dos veces derivable con f'' continua. Sea $\overline{x} \in (a, b)$ una raíz simple de f, e.d. $f(\overline{x}) = 0$ y $f'(\overline{x}) \neq 0$. Entonces existe $\delta > 0$ tal que si $x_0 \in (a, b)$, y $|x_0 - \overline{x}| < \delta$, la sucesión de iteradas de Newton, $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, está bien definida para todo $n \in \mathbb{N}$ y converge hacia $\overline{x}$.*

Ejemplo 2.4.11 $f(x) = 0.25 * e^x$ tiene una atractor cerca de 0.4, si hacemos las iteradas funcionales gráficamente desde $x_0 = 0.9$ tenemos:



Cuando la derivada de la función en el punto fijo es mayor que 1, el comportamiento de las iteradas funcionales cerca de este punto fijo es todo lo contrario a lo que ocurre con los atractores.

Ejemplo 2.4.12 $f(x) = 0.25 * e^x$, tiene punto fijo cerca de 2.15, si hacemos las iteradas funcionales gráficamente desde $x_0 = 2.1$ tenemos:



Proposición 2.4.13 *Si* $|f'(\overline{x})| > 1$ *entonces, en ningún entorno de* $\overline{x}$ *se cumplen las hipótesis del teorema del punto fijo. Se dice que* $\overline{x}$ *es un punto* ***repulsor*** *(o* ***fuente****) de* f*.*

Las iteradas $x_n = f(x_{n-1})$ *no convergen a* $\overline{x}$*, para* x_0 *próximo a* $\overline{x}$*, salvo que* $x_n = \overline{x}$ *para algún* n*.*

Demostración:

Ideas que intervienen

- $|f'(\overline{x})| > C > 1 + f'$ *continua* $\Rightarrow$ *que existe* $\delta > 0$ *tal que* $|f'(x)| > C$ ***para*** $x \in [\overline{x}-\delta, \overline{x}+\delta]$.
- *Si* $x_n = f(x_{n-1})$*, entonces* $x_n \not\to \overline{x}$ *salvo si para algún* n_0*,* $x_{n_0} = \overline{x}$ $(x_n = \overline{x}, \forall n > n_0)$.

En efecto: si fuese convergente existiría n_0 tal que $0 < |x_n - \overline{x}| < \delta$ para $n \geq n_0$. Pero

$$\begin{aligned} |x_n - \overline{x}| &= |f(x_{n-1}) - f(\overline{x})| \\ &= |f'(\xi)||x_{n-1} - \overline{x}| \\ &> C\,|x_{n-1} - \overline{x}| > ... \\ &> C^{n-n_0}|x_{n_0} - \overline{x}| \to \infty \end{aligned}$$

$(C > 1 \Rightarrow C^{n-n_0} \to \infty)$ **ABSURDO** □

2.5. Orden de convergencia de una sucesión

Para formalizar la idea de velocidad en la convergencia de una sucesión y poder cuantificar el que una sucesión converge hacia su límite más rápidamente que otra, vamos a considerar la siguiente definición:

Definición 2.5.1 *Si $x_n \to \overline{x}$, existen $C > 0$ y $m > 0$ tales que*

$$|x_n - \overline{x}| \leq C|x_{n-1} - \overline{x}|^m \quad \forall n,$$

*entonces se dice que $x_n \to \overline{x}$ con **convergencia de orden al menos** m*

Si $x_n \to \overline{x}$ con $x_n \neq \overline{x}$, existe $\lambda > 0$ y $m > 0$ tales que

$$\lim_n \frac{|x_n - \overline{x}|}{|x_{n-1} - \overline{x}|^m} = \lambda,$$

*entonces se dice que $x_n \to \overline{x}$ con **convergencia de orden** m. A λ se le llama la constante asintótica del error.*

Si x_n es una sucesión convergente con convergencia de orden $m = 1$, se dice que x_n es linealmente convergente.

Cuando x_n es una sucesión convergente con convergencia de orden $m = 2$, se de que la sucesión es cuadráticamente convergente.

Observad que para si n es grande x_n aproxima a $\overline{x}$ con k dígitos en el sentido que[1] $|x_n - \overline{x}| < 0.5 * 10^{-k}$, entonces

$$|x_{n+1} - \overline{x}| \approx \lambda * |x_n - \overline{x}|^2 < \lambda * 0.5^2 * 10^{-2k}$$

x_{n+1} aproximará a $\overline{x}$ con alrededor de $2k$ dígitos de precisión.

¿Cúal es el orden de convergencia de las iteraciones de punto fijo?

Teorema 2.5.2 (Orden de convergencia) *Si $f : [a, b] \to [a, b]$ es una función con un punto fijo en $\overline{x}$.*

(Caso $m = 1$) Si f es derivable con derivada continua, $|f'(\overline{x})| < 1$, y x_0 es un punto próximo a $\overline{x}$, entonces $x_n = f(x_{n-1})$ converge a $\overline{x}$ con convergencia de orden al menos 1 ($\overline{x}$ es un atractor).

(Caso $m \geq 2$) Si f es m-veces derivable en $[a, b]$, con derivada de orden m $f^{(m)}$ continua, se cumple:

$$f'(\overline{x}) = f''(\overline{x}) = ... = f^{(m-1)}(\overline{x}) = 0,$$

y x_0 es un punto próximo a $\overline{x}$, entonces $x_n = f(x_{n-1})$ converge a $\overline{x}$ con convergencia de orden al menos m.

Si además $f^{(m)}(\overline{x}) \neq 0$, la convergencia es de orden m con constante asintótica del error $|f^{(m)}(\overline{x})|/m!$.

[1] Permitid la licencia de que aquí consideremos errores absolutos en lugar de los errores relativos del capítulo anterior.

Demostración:
Ideas que intervienen

- *(caso m=1)* $|f'(\overline{x})| < 1$ *implica que* $\overline{x}$ *es un atractor. e.d. que las iteradas funcionales son convergentes.*
- *(caso m>1) Como* $f'(\overline{x}) = 0 < 1$ *la sucesión de iteradas* $x_{n+1} = f(x_n)$ *converge hacia* $\overline{x}$ *siempre que* x_0 *esté cerca de* $\overline{x}$.

  Considera la fórmula del desarrollo de Taylor de f *en* $\overline{x}$ *de orden* m*:*

$$f(x) - f(\overline{x}) = \frac{f^{(m)}(\overline{x})}{m!}(x - \overline{x})^m + o((x - \overline{x})^m).$$

  Así tenemos

$$\frac{x_{n+1} - \overline{x}}{(x_n - \overline{x})^m} = \frac{f(x_n) - f(\overline{x})}{(x_n - \overline{x})^m} = \frac{f^{(m)(\overline{x})}}{m!} + \frac{o((x_n - \overline{x})^m)}{(x_n - \overline{x})^m} \to \frac{f^{(m)(\overline{x})}}{m!}.$$

□

Como corolario de este resultado y de la proposición 2.4.10, se tiene el siguiente teorema de convergencia local del método de Newton,

Teorema 2.5.3 *Sea* $f : (a, b) \to \mathbb{R}$ *una función dos veces derivable con* f'' *continua. Sea* $\overline{x} \in (a, b)$ *una raíz simple de* f*, e.d.* $f(\overline{x}) = 0$ *y* $f'(\overline{x}) \neq 0$*. Entonces existe* $\delta > 0$ *tal que si* $x_0 \in (a, b)$*,* y $|x_0 - \overline{x}| < \delta$*, la sucesión de iteradas de Newton,* $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$*, está bien definida para todo* $n \in \mathbb{N}$ *y converge hacia* $\overline{x}$ ***con convergencia de orden al menos 2.***

¿Cúal es el orden de convergencia y la constante asintótica del error en el método de Newton si $f'(\overline{x}) \neq 0$ y $f''(\overline{x}) \neq 0$?

Volved al ejemplo 2.4.7 y comparar los ordenes de convergencia de los distintos procesos iterativos que se estudiaron allí.

2.6. Acelerando la convergencia: método Δ^2 de Aitken y método de Steffensen

No siempre tenemos la suerte de construir iteradas que converjan cuadráticamente. Pero si de antemano sabemos que nuestra sucesión de iteradas converge linealmente, podemos acelerar la convergencia de la sucesión con el método de aceleración de Aitken.

Supongamos que $x_n \to \overline{x}$ con convergencia de orden al menos 1, y que existe

$$\lim_{n\to\infty} \frac{x_{n+1} - \overline{x}}{x_n - \overline{x}} = A \neq 1.$$

Remarcad la hipótesis de que la constante asintótica $A \neq 1$. Esto ocurre en los puntos fijos que son atractores $A = |f'(\overline{x})| < 1$.

Entonces, para n suficientemente grande podemos escribir

$$\begin{cases} x_{n+1} - \overline{x} & \approx A(x_n - \overline{x}) \\ x_{n+2} - \overline{x} & \approx A(x_{n+1} - \overline{x}). \end{cases} \tag{2.3}$$

Restando esas dos expresiones tenemos

$$\begin{cases} x_{n+2} - x_{n+1} \approx A(x_{n+1} - x_n) \\ A \approx \frac{x_{n+2}-x_{n+1}}{x_{n+1}-x_n}, \text{ y} \\ 1 - A \approx -\frac{(x_{n+2}-x_{n+1})-(x_{n+1}-x_n)}{x_{n+1}-x_n} \end{cases} \tag{2.4}$$

Despejando $\overline{x}$ en (2.3) y sustituyendo $1 - A$ por la aproximación dada en (2.4) queda

$$\overline{x} \approx \frac{x_{n+1} - Ax_n}{1 - A} = x_n + \frac{x_{n+1} - x_n}{1 - A} \approx x_n - \frac{(x_{n+1} - x_n)^2}{(x_{n+2} - x_{n+1}) - (x_{n+1} - x_n)}. \tag{2.5}$$

Se utiliza la notación de Aitken:

$$\Delta x_n := (x_{n+1} - x_n)$$
$$\Delta^2 x_n := \Delta(\Delta x_n) = (x_{n+2} - x_{n+1}) - (x_{n+1} - x_n) = x_{n+2} - 2x_{n+1} + x_n.$$

Con lo que la estimación dada en (2.5) queda

$$\overline{x} \approx x_n - \frac{(x_{n+1} - x_n)^2}{(x_{n+2} - x_{n+1}) - (x_{n+1} - x_n)} = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}$$

El método de ***aceleración*** Δ^2 ***de Aitken****, consiste en ir construyendo la sucesión*

$$\tilde{x}_n = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}$$

con la esperanza de que $\tilde{x}_n$ *converja hacia* $\overline{x}$ *más rápidamente que* x_n*.*

Teorema 2.6.1 (Aceleración de Aitken) *Supongamos que* $x_n \to \overline{x}$ *con convergencia de orden al menos 1, que para todo* n $x_n - \overline{x} \neq 0$*, y que*

$$\lim_{n\to\infty} \frac{x_{n+1} - \overline{x}}{x_n - \overline{x}} = A \neq 1.$$

Entonces la sucesión del método de Aitken $\tilde{x}_n = x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n}$ *converge hacia* $\overline{x}$ *más rápidamente que* x_n *en el sentido de que*

$$\lim_{n\to\infty} \frac{\tilde{x}_n - \overline{x}}{x_n - \overline{x}} = 0$$

Demostración:

Ideas que intervienen

- *Expresar* $\tilde{x}_n - \overline{x}$ *en términos de* $x_n - \overline{x}$ *y* $A - 1 \neq 0$*.*

(I) $\delta_n := \dfrac{x_{n+1} - \overline{x}}{x_n - \overline{x}} - A \to 0$

(II) $x_{n+1} - \overline{x} = (x_n - x) + \Delta x_n = (x_n - x)(\delta_n + A)$

(III) $\Delta x_n := x_{n+1} - x_n = (x_{n+1} - \overline{x}) - (x_n - \overline{x}) = (\delta_n + (A - 1))(x_n - x)$.

(IV) $\Delta^2 x_n = \Delta x_{n+1} - \Delta x_n = (\delta_{n+1} + (A-1))(x_{n+1} - \overline{x}) - (\delta_n + (A-1))(x_n - x) = \Gamma_n(x_n - x)$, donde $\Gamma_n = (\delta_{n+1} + (A - 1))(\delta_n + A) - (\delta_n + (A - 1)) \to (A - 1)^2 \neq 0$.

(v) $$\tilde{x}_n - \overline{x} = (x_n - x) - \frac{(\Delta x_n)^2}{\Delta^2 x_n} = \frac{\Gamma_n (x_n - x)^2 - (\delta_n + (A-1))^2 (x_n - x)^2}{\Gamma_n (x_n - x)} = \\ = \frac{\Gamma_n (x_n - x) - (\delta_n + (A-1))^2 (x_n - x)}{\Gamma_n}.$$

(vi) $$\frac{\tilde{x}_n - \overline{x}}{x_n - x} = \frac{\Gamma_n - (\delta_n + (A-1))^2}{\Gamma_n} \to \frac{(A-1)^2 - (A-1)^2}{(A-1)^2} = 0.$$

□

Las hipótesis del teorema anterior se cumplen para iteraciones funcionales

$$x_n = f(x_{n-1})$$

que convergen hacia un punto fijo atractor $\overline{x}$, $|A| = |f'(\overline{x})| < 1$.

El algoritmo correspondiente a la aceleración de Aitken para iteraciones de punto fijo es el siguiente. En recuadros grises con texto en blanco tenéis las modificaciones al algoritmo de punto fijo:

Algoritmo 2.8 Iteración de punto fijo con Aceleración de Aitken

Datos de entrada: Función f; $x_0 \in \mathbb{R}$; $precision \in \mathbb{R}$; $nmaxiter \in \mathbb{Z}$
Datos de salida: solución x $(f(x) = x)$ o mensaje de error
Variables:
$xx = x_0$ // los puntos de la órbita.
$fx = f(xx)$; // el siguiente punto de la órbita f.
ffx; // para el siguiente punto de la órbita f(fx).
$xxacelerado$; // para guardar los terminos de la iteración acelerada por Aitken.
Fujo del programa:

```
for(k=1;k<=nmaxiter;k++){
  if(|xx − fx| < precision){   Parada: Punto fijo en xx }
    ffx = f(fx);
    xxacelerado = xx − ((fx − xx) ∗ (fx − xx))/(ffx − 2 ∗ fx + xx);
    if(|xxacelerado − f(xxacelerado)| < precision){
                                  Parada: Punto fijo en xxacelerado }
    xx = fx;
    fx = ffx;    // el siguiente punto de la órbita f
  }
  Parada: Error: No hay convergencia en nmaxiter iteraciones
```

Ejemplo 2.6.2 (véanse los ejemplos 5 y 6 del paquete `algoritmosIteraciones`) La función $g(x) = 1.6 + 0.99 \cos x$ es contractiva y tiene un único punto fijo.

La iteración de punto fijo para g, con inicio en $x_0 = 1.5707963267948966$ proporciona una aproximación al punto fijo $\overline{x} \approx 1.585521893915304$ en 560 iteraciones ($\overline{x} \approx x_{560}$). La derivada

de g en el punto fijo, $g'(\overline{x}) = -0.9898926649877118$, tiene valor absoluto menor pero próximo a 1. Esto confirma que el punto fijo es un atractor y que la convergencia no es de las más rápidas.

Si aceleramos la convergencia de la iteración de punto fijo iniciada en el mismo punto x_0, el método de Aitken proporciona la aproximación al punto fijo $\overline{x} \approx 1.585471801571905$ en 476 pasos.

Si consideramos la función $f(x) = 0.25e^x$ analizada en los ejemplos 2.4.11 y 2.4.12, y hacemos la iteración de punto fijo para f, con inicio $x_0 = 0.1$ aproximamos el punto fijo $\overline{x} = 0.35740295618138784$ en 33 pasos. Observad que en este ejemplo $f'(\overline{x}) = 0.3574029561813885$ es mucho menor que en el caso de la función anterior y que ahora la convergencia es mucho más rápida.

Si aceleramos la convergencia con el método de Aitken, iniciando también en $x_0 = 0.1$ aproximamos el mismo punto fijo en 15 pasos.

¡Cuanto menor es el tamaño de la derivada en el punto fijo, más rápido convergen las iteradas de punto fijo y sus aceleradas de Aitken!

2.6.1. Método de Steffensen

Si consideramos una iteración de punto fijo, $x_n = f(x_{n-1})$, que suponemos convergente hacia un punto fijo atractor $\overline{x}$, y la aceleramos por el método de Aitken, debemos esperar que la sucesión acelerada $\tilde{x}_n$ converja antes hacia $\overline{x}$. Esta esperanza, nos permite suponer que el término

$$y = \tilde{x}_n = x_n - \frac{(f(x_n) - x_n)^2}{f(f(x_n)) - 2f(x_n) + x_n}$$

se aproxima a $\overline{x}$ mejor que x_n. Con esta idea en mente, ¿Porqué seguir iterando f en x_n en lugar de considerar iterar las evaluaciones de y?

Eso es justo lo que propone el Método de Steffensen:

- dada y_n una aproximación a un punto fijo $\overline{x}$ de f,
- definimos

$$y_{n+1} = y_n - \frac{(f(y_n) - y_n)^2}{f(f(y_n)) - 2f(y_n) + y_n}.$$

Teorema 2.6.3 *Sea $f : [a,b] \to [a,b]$ una función de clase C^2 en [a,b], con un punto fijo en $\overline{x} \in (a,b)$ ($f(\overline{x}) = \overline{x}$), $f'(\overline{x}) \neq 1$. Entonces, existe $\delta > 0$ tal que $[\overline{x} - \delta, \overline{x} + \delta] \subset (a,b)$ y para cualquier $y_0 \in [\overline{x} - \delta, \overline{x} + \delta]$ el algoritmo de Steffensen*

$$y_{n+1} = y_n - \frac{(f(y_n) - y_n)^2}{f(f(y_n)) - 2f(y_n) + y_n},$$

está bien definido y converge hacia $\overline{x}$ con convergencia de orden al menos 2.

Demostración:

Ideas que intervienen

- *Escribir $y_{n+1} - \overline{x}$ en términos de $y_n - \overline{x}$ utilizando el desarrollo de Taylor de f en $\overline{x}$.*

- *Utilizar la notación* $z(x) = o((x-\overline{x})^k)$ *para designar a los infinitésimos* $\lim_{x\to\overline{x}} \frac{z(x)}{(x-\overline{x})^k} = 0$, *y simplificar las expresiones.*

  Recordar que $o((x-\overline{x})^k) + o((x-\overline{x})^{k'}) = o((x-\overline{x})^{\min\{k,k'\}}$, $(x-\overline{x})^k o((x-\overline{x})^{k'}) = o((x-\overline{x})^{k+k'}$, *etc.*

(I) $f(x) = \overline{x} + f'(\overline{x})(x-\overline{x}) + \frac{f''(\overline{x})}{2}(x-\overline{x})^2 + o((x-\overline{x})^2)$;

$f(x) - \overline{x} = f'(\overline{x})(x-\overline{x}) + \frac{f''(\overline{x})}{2}(x-\overline{x})^2 + o((x-\overline{x})^2)$

$(f(x)-\overline{x})^2 = f'(\overline{x})^2(x-\overline{x})^2 + o((x-\overline{x})^2)$

(II) $\Delta fx := f(x) - x = (f'(\overline{x})-1)(x-\overline{x}) + \frac{f''(\overline{x})}{2}(x-\overline{x})^2 + o((x-\overline{x})^2)$;

$(\Delta fx)^2 = (f(x)-x)^2 = (f'(\overline{x})-1)^2(x-\overline{x})^2 + (f'(\overline{x})-1)f''(\overline{x})(x-\overline{x})^3 + o((x-\overline{x})^3$;

(III) $f(f(x)) - f(x) = (f'(\overline{x})-1)(f(x)-\overline{x}) + \frac{f''(\overline{x})}{2}(f(x)-\overline{x})^2 + o((f(x)-\overline{x})^2) =$
$= (f'(\overline{x})-1)(f(x)-\overline{x}) + \frac{f''(\overline{x})}{2}(f(x)-\overline{x})^2 + o((x-\overline{x})^2)$;

(IV) Restando (III-II) tenemos:

$\Delta^2 fx := f(f(x)) - 2f(x) + x = (f'(\overline{x})-1)(f(x)-x) +$
$\frac{f''(\overline{x})}{2}[(f(x)-\overline{x})^2 - (x-\overline{x})^2] + o((x-\overline{x})^2) =$
$(f'(\overline{x})-1)^2(x-\overline{x}) + (f'(\overline{x})-1)\frac{f''(\overline{x})}{2}(x-\overline{x})^2 +$
$\frac{f''(\overline{x})}{2}(f'(\overline{x})^2-1)(x-\overline{x})^2 + o((x-\overline{x})^2) =$
$(f'(\overline{x})-1)^2(x-\overline{x}) + \frac{f''(\overline{x})}{2}(f'(\overline{x})-1)(1+(f'(\overline{x})+1))(x-\overline{x})^2 + o((x-\overline{x})^2)$;

(V) Consideramos $g(x) = x - \frac{(f(x)-x)^2}{f(f(x))-2f(x)+x}$.

$$g(x) - \overline{x} = \frac{\Delta^2 fx(x-\overline{x}) - (\Delta fx)^2}{\Delta^2 fx} =$$
$$= \frac{\frac{f''(\overline{x})}{2}(f'(\overline{x})-1)f'(\overline{x})(x-\overline{x})^3 + o((x-\overline{x})^3)}{(f'(\overline{x})-1)^2(x-\overline{x}) + \frac{f''(\overline{x})}{2}(f'(\overline{x})-1)(1+(f'(\overline{x})+1))(x-\overline{x})^2 + o((x-\overline{x})^2)} =$$
$$= \frac{\frac{f''(\overline{x})}{2}f'(\overline{x})(x-\overline{x})^2 + o((x-\overline{x})^2)}{(f'(\overline{x})-1) + \frac{f''(\overline{x})}{2}(1+(f'(\overline{x})+1))(x-\overline{x}) + o(x-\overline{x})};$$

(VI)

$$\lim_{x\to\overline{x}} \frac{g(x)-\overline{x}}{(x-\overline{x})^2} = \lim_{x\to\overline{x}} \frac{\frac{f''(\overline{x})}{2}f'(\overline{x}) + \frac{o((x-\overline{x})^2)}{(x-\overline{x})^2}}{(f'(\overline{x})-1) + \frac{f''(\overline{x})}{2}(1+(f'(\overline{x})+1))(x-\overline{x}) + o(x-\overline{x})} = \frac{f''(\overline{x})f'(x)}{2(f'(\overline{x})-1)}.$$

□

El algoritmo correspondiente a la aceleración de Steffensen para iteraciones de punto fijo es el siguiente. En recuadros grises con texto en blanco tenéis las modificaciones al algoritmo de punto fijo:

Algoritmo 2.9 **Iteración de punto fijo con Aceleración de Steffensen**

Datos de entrada: Función f; $x_0 \in \mathbb{R}$; $precision \in \mathbb{R}$; $nmaxiter \in \mathbb{Z}$
Datos de salida: solución x $(f(x) = x)$ o mensaje de error
Variables:
$xx = x_0$ // los puntos de la órbita.
fx; // el siguiente punto de la órbita f.
ffx; // para el siguiente punto de la órbita f(fx).
Fujo del programa:
`for(k=1;k<=nmaxiter;k++){`
$fx = f(xx)$;
$ffx = f(fx)$;
`if(`$|xx - fx| < precision$`){` **Parada:** Punto fijo en xx }
$xx = xx - ((fx - xx) * (fx - xx))/(ffx - 2 * fx + xx)$;
}
Parada: Error: No hay convergencia en $nmaxiter$ iteraciones

Notas:

- Observad que la hipótesis del teorema anterior, $f'(\overline{x}) \neq 1$, es más general que pedir que el punto fijo $\overline{x}$ sea un atractor, también puede darse en repulsores.
- La diferencia fundamental entre el método de aceleración de Aitken y el método de Steffensen aplicados a una iteración de punto fijo, radica en que en el método de Aitken la aceleración se produce sobre la sucesión de iteradas que hay que construir de entrada, mientras que en el método de Steffensen se construye directamente la sucesión acelerada.
- Si las iteradas de punto fijo convergen linealmente, la acelerada de Steffensen lo hace al menos cuadráticamente. Si las iteradas de punto fijo convergen con convergencia de orden $p > 1$, las de Steffensen convergen con convergencia de orden al menos $2p - 1$ [1]

Ejemplo 2.6.4 La función $f(x) = 0.25e^x$ considerada en los ejemplos 2.4.11, 2.4.12 y 2.6.2 , tiene dos puntos fijos, uno atractor y el otro repulsor.

Recordad que la iteración de punto fijo para f, con inicio $x_0 = 0.1$ aproximamos el punto fijo $\overline{x} = 0.35740295618138784$ en 33 pasos, y que acelerando con Aitken este número de iteraciones se reduce a 15 pasos.

El método de Steffensen con inicio en el mismo $x_0 = 0.1$ proporciona la aproximación a $\overline{x} = 0.35740295618138895$ en sólo 5 pasos.

Si consideramos la función $g(x) = f(x) - x$, que se anula en los puntos fijos de f, podemos comparar el método de Steffensen de búsqueda de puntos fijos de f con el de Newton para localizar los ceros de f.

La iteración de Newton para g, con inicio en $x_0 = 0.1$ proporciona $\overline{x} = 0.3573508313620166$ en 3 pasos.

El otro cero de g se puede encontrar haciendo la iteración de Newton con inicio en $x_0 = 2.1$, obteniendo $\overline{x} = 2.153299834108639$ en 3 iteraciones.

Si hacemos la iteración de Steffensen de f con inicio en $x_0 = 2.1$ obtenemos el punto fijo repulsor $\overline{x} = 2.1532923641103503$ en 5 pasos.

Observad que el método de Steffensen también permite localizar puntos fijos repulsores, tal y como señalábamos antes.

2.7. Actividades complementarias del capítulo

Los documentos se pueden descargar de la Zona Compartida de SUMA.

- Paquete de java: `algoritmosIteraciones`, con los métodos y los ejemplos de la unidad.
- Hoja de problemas nº2 (22/10/2007)
- Practica nº3

Bibliografía

[1] A. Delshams A. Aubanell, A. Benseny, *Útiles básicos de cálculo numérico*, Ed. Labor - Univ. Aut. de Barcelona, Barcelona, 1993.

[2] R.L. Burden and J.D. Faires, *Análisis numérico, 7^a edición*, Thomson-Learning, Mexico, 2002.

[3] D. Kincaid and W. Cheney, *Análisis numérico*, Ed. Addison-Wesley Iberoamericana, Reading, USA, 1994.

Capítulo 3

Introducción y complementos de análisis matricial.

Interrogantes centrales del capítulo

- Repaso de conceptos básicos del álgebra matricial.
- Triangulación y diagonalización de matrices
- Propiedades de las matrices simétricas y hermitianas. Radio espectral. Cocientes de Rayleigh.
- Normas matriciales.
- Condicionamiento de un sistema lineal
- Un problema de análisis numérico matricial.

Destrezas a adquirir en el capítulo

- Conocer y manejar distintas normas matriciales. Distinguir la mejor norma matricial.
- Saber definir el número de condición de una matriz o de un sistema lineal y saber calcularlo.
- Conocer problemas modelizados por sistemas lineales de dimensión grande.

Este es el primer capítulo de los cuatro dedicados al análisis numérico matricial que aborda las dos cuestiones siguientes

- *La resolución de sistemas de ecuaciones lineales:* Dada una matriz no singular A y un vector b, encontrar el vector solución x del sistema lineal

$$Ax = b.$$

- *El cálculo de valores propios y vectores propios de una matriz:* Dada una matriz cuadrada A, encontrar todos sus valores propios, o solo algunos significativos, y eventualmente los vectores propios correspondientes. En otras palabras, encontrar vectores $p \neq 0$ y escalares $\lambda \in \mathbb{R}$ o $\mathbb{C}$, tales que

$$Ap = \lambda p.$$

En los próximos tres capítulos vamos a estudiar métodos directos y métodos iterativos para la resolución de sistemas lineales y métodos de cálculo de valores y vectores propios, pero antes vamos a detenernos en observar un modelo que origina un sistema de ecuaciones lineales con el objetivo de que nos sirva de ejemplo del tipo de sistemas que pueden aparecer a la hora de buscar soluciones a problemas reales concretos.

Por otra parte, para poder establecer criterios que nos permitan comparar entre los distintos métodos, y analizar el condicionamiento y la estabilidad de los mismos, describiremos distancias entre matrices y aprenderemos a calcularlas.

Desarrollo de los contenidos fundamentales

- Repaso de conceptos básicos de álgebra matricial
- Origen de los problemas del análisis numérico matricial.
- Normas matriciales. Normas subordinadas y sus propiedades.
- Análisis del error y condicionamiento.

Temporalización: 7HTe + 1HPb + (Al finalizar, primer examen parcial) 3HEv = 11 H

3.1. Repaso de conceptos básicos de los sistemas de ecuacion es y el álgebra matricial.

3.1.1. Sistemas de Ecuaciones

Un **sistema** de m ecuaciones lineales (simultáneas) con n incógnitas es un conjunto de m ecuaciones con n incógnitas que se van a considerar relacionadas por la existencia de soluciones comunes.

Formalmente, se suele representar así:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

donde los **coeficientes**, a_{ij}, y los **términos independientes**, b_i, son escalares del cuerpo $\mathbb{K}$ ($\mathbb{R}$ o $\mathbb{C}$). El primer índice, i, indica la ecuación en la que se encuentran los coeficientes y los términos independientes ($1 \leq i \leq m$); mientras que el índice, j, de los coeficientes indica la incógnita, x_j, a la que multiplican ($1 \leq j \leq n$).

Las **soluciones** del sistema de ecuaciones son las listas de escalares $(x_1, \ldots, x_n)$ que son soluciones de todas las ecuaciones al mismo tiempo (simultáneamente).

Si un sistema de ecuaciones tiene soluciones se llama **compatible** y si no las tiene, **incompatible.** Un sistema compatible con una única solución se llama **determinado** y si tiene más soluciones se llama **indeterminado.**

Un sistema de ecuaciones donde todos los términos independientes son nulos ($b_1 = \cdots = b_m = 0$) se llama **homogéneo** y tiene siempre soluciones: al menos la solución trivial $x_1 = \cdots = x_n = 0$.

Se dice que dos sistemas de ecuaciones son **equivalentes** si tienen el mismo conjunto de soluciones. Recordad las siguientes operaciones que se utilizan para transformar los sistemas de ecuaciones en otros equivalentes que sean más fáciles de resolver:

Estas **operaciones elementales** para un sistema de ecuaciones son:

(I) Intercambiar dos ecuaciones de posición.

(II) Multiplicar una ecuación por un escalar no nulo.

(III) Sumar a una ecuación un múltiplo escalar de otra ecuación.

Teorema 3.1.1 *Al aplicar una operación elemental a un sistema, el sistema resultante es equivalente al original.*

En el capítulo siguiente utilizaremos operaciones elementales para recordar y describir el método de Gauss de resolución de sistemas de ecuaciones por «eliminación».

3.1.2. Matrices

La información de que disponemos acerca de un sistema de ecuaciones la proporcionan los coeficientes y los términos independientes, que podemos ordenar así:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{pmatrix}$$

Estas ordenaciones rectangulares se llaman **matrices** y están relacionadas con otros muchos problemas, además de con los sistemas de ecuaciones.

Si los elementos $a_{ij} \in \mathbb{K}$ con $1 \leq i \leq m$ y $1 \leq j \leq n$, se llama **matriz de m filas y n columnas** o matriz $m \times n$, a

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix},$$

También se suele decir que es una matriz de tipo (m, n), y se representa de forma abreviada mediante su **termino general** a_{ij},

$$(a_{ij})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}.$$

De forma más breve, también se utiliza una letra como por ejemplo

$$A = (a_{ij}),$$

si se sobrentiende el recorrido de los índices. Los índices de cada término a_{ij} nos dicen donde está colocado; el primero, i, indica la fila, y el segundo, j, la columna. A veces, interesará indicar el término general de la matriz A en función de los índices en la forma

$$A(i,j) = a_{ij}.$$

El conjunto de las matrices $m \times n$ de escalares se suele representar por $\mathcal{M}_{mn}(\mathbb{K})$, o simplemente por $\mathcal{M}_{mn}$ si el cuerpo $\mathbb{K}$ está claro.

Si el número de filas y el de columnas coinciden, $m = n$, se dice que la matriz es cuadrada, y se denota por $\mathcal{M}_n$ al conjunto de las matrices cuadradas.

Si $m = 1$ tenemos las matrices **fila**, y si $n = 1$, las matrices **columna.** A los conjuntos correspondientes se les suele denotar por $\mathbb{K}^n$ o $\mathbb{K}^m$ (aunque si fuese necesario distinguir entre filas o columnas se debe utilizar la notación $\mathcal{M}_{1n}$ o $\mathcal{M}_{m1}$) al identificarlos con los productos cartesianos de $\mathbb{K}$ consigo mismos, que son los conjuntos formados por las listas ordenadas de escalares dispuestas en filas o en columnas.

Dada una matriz M, si consideramos solamente y en el mismo orden, los elementos que están en la intersección de algunas filas y columnas apropiadas se obtiene una **submatriz.**

Ejemplo 3.1.2 Consideremos la matriz

$$M = \begin{pmatrix} -1 & 0 & 2 & \pi \\ 0 & 1 & -2 & 1 \\ 3 & 2 & 0 & -1 \end{pmatrix}.$$

Si tachamos la segunda fila y las columnas segunda y cuarta obtenemos la submatriz:

$$M' = \begin{pmatrix} -1 & 2 \\ 3 & 0 \end{pmatrix}.$$

Al contrario, si se pegan matrices de tamaños convenientes se puede obtener otra de tamaño mayor. Dicho al revés, la matriz grande se puede considerar descompuesta en **bloques**:

$$M = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right).$$

Ejemplo 3.1.3 Si $A = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, $B = \left(\begin{smallmatrix} 2 \\ 2 \end{smallmatrix}\right)$, $C = \left(\begin{smallmatrix} 3 & 3 \end{smallmatrix}\right)$ y $D = (4)$, podemos obtener una nueva matriz $(2+1) \times (2+1)$ descompuesta en bloques por

$$M = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) = \left(\begin{array}{cc|c} 0 & 1 & 2 \\ 1 & 0 & 2 \\ \hline 3 & 3 & 4 \end{array}\right) = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 3 & 3 & 4 \end{pmatrix}.$$

A veces es conveniente pensar en una matriz M descompuesta en bloques formados por cada una de sus filas, en otras palabras, pensar en M como una columna de filas. O pensar en M como en una fila de columnas. En este sentido se utiliza normalmente la notación :

$$M_i = (a_{i1}, \ldots, a_{in}) \in \mathbb{K}^n \text{ y } M^j = \begin{pmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{pmatrix}$$

para los vectores fila y columna respectivamente, representando

$$M = \begin{pmatrix} M_1 \\ \vdots \\ M_m \end{pmatrix} \text{ y } M = \left(M^1, \ldots, M^n\right).$$

Otro caso muy frecuente es el de la matriz del sistema que describíamos al principio. A la submatriz formada por las n primeras columnas se le llama **matriz de coeficientes**

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix},$$

y la submatriz dada por la última columna

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

se le llama columna de **términos independientes**. La matriz completa $(A \mid B)$ se le llama **matriz ampliada.** Cuando coinciden el número de ecuaciones y él de incógnitas, la matriz de coeficientes es una matriz cuadrada.

Ejemplo 3.1.4 La matriz ampliada del sistema

$$\begin{cases} 2x - y + z = -1 \\ -x + 2z = 3 \end{cases}$$

es $\left(\begin{smallmatrix} 2 & -1 & 1 & -1 \\ -1 & 0 & 2 & 3 \end{smallmatrix}\right)$. Su columna de términos independientes es $\left(\begin{smallmatrix} -1 \\ 3 \end{smallmatrix}\right)$ y la primera ecuación está representada por $(2,\ -1,\ 1,\ -1)$.

3.1.3. Espacios vectoriales. Aplicaciones lineales

Sea E un espacio vectorial de dimensión n sobre el cuerpo $\mathbb{K}$ ($\mathbb{R}$ o $\mathbb{C}$). Una base de E es un conjunto $\{v_1, \ldots, v_n\}$, de n vectores linealmente independientes de E y todo vector $x \in E$ admite una descomposición única de la forma

$$x = \sum_{i=1}^{n} x_i v_i.$$

Cuando la base está fijada sin ambigüedad, se identifican cada vector x con sus coordenadas $\{x_1, \ldots, x_n\}$ con respecto a esta base. Así se identifican E y $\mathbb{K}^n$.

En notación matricial representaremos siempre a x por el vector columna

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix},$$

y denotaremos por por x^t y x^* a los vectores fila

$$x^t = (x_1, \ldots, x_n) \qquad x^* = (\overline{x_1}, \ldots, \overline{x_n}),$$

donde $\overline{z}$ es complejo conjugado de $z \in \mathbb{C}$. El vector fila x^t es el **vector traspuesto** del vector columna x y x^* es el **vector adjunto** de x.

Si $\mathbb{K} = \mathbb{R}$, el producto escalar euclídeo en E se define por

$$(x, y) = x^t y = y^t x = \sum_{i=1}^{n} x_i y_i,$$

y cuando $\mathbb{K} = \mathbb{C}$ el producto escalar hermitiano en E se define por

$$(x, y) = y^* x = \overline{x^* y} = \sum_{i=1}^{n} x_i \overline{y_i}.$$

Asociada al producto escalar o hermitiano se tiene la norma

$$\|x\|_2 = \sqrt{(x, x)} = \sqrt{\sum_{i=1}^{n} |x_i|^2},$$

que define la distancia euclidea en E. Junto a esta noción de distancia también introduce la noción de ortogonalidad (ángulo), x e y se dicen ortogonales cuando $(x, y) = 0$. Observad que en relación a este producto escalar, los vectores de la base fijada v_i son dos a dos ortogonales y todos tienen norma 1. Las bases formadas por vectores ortogonales de norma 1 se denominan **bases ortonormales** con respecto al producto escalar.

Sean $f : V \to W$ una aplicación lineal, y $\{v_1, \ldots, v_n\}$, $\{w_1, \ldots, w_m\}$ bases de V y W respectivamente. Los n vectores $f(v_1), \ldots, f(v_n)$ están en W por lo que pueden escribirse de forma única como combinación lineal de la base $\{w_1, \ldots, w_m\}$:

$$\begin{aligned} f(v_1) &= a_{11} w_1 + a_{21} w_2 + \cdots + a_{m1} w_m \\ f(v_2) &= a_{12} w_1 + a_{22} w_2 + \cdots + a_{m2} w_m \\ &\cdots\cdots \quad \cdots \\ f(v_n) &= a_{1n} w_1 + a_{2n} w_2 + \cdots + a_{mn} w_m \end{aligned}$$

Se obtienen así $m \times n$ escalares a_{ij} $(i = 1, \ldots, m,\ j = 1, \ldots, n)$ en los que los subíndices (ij) expresan que se trata de la i-ésima (primer índice) coordenada de la imagen del j-ésimo (segundo índice) vector de la base de V. Estos $m \times n$ escalares determinan una matriz $m \times n$

$$M = (a_{ij}) = \begin{pmatrix} a_{11} & & a_{1n} \\ \cdots & \cdots & \cdots \\ a_{m1} & & a_{mn} \end{pmatrix}$$

que se llama matriz asociada a la aplicación lineal $f : V \to W$ con respecto de las bases $\{v_1, \ldots, v_n\}$, $\{w_1, \ldots, w_m\}$. Nótese que la columna j de M, $M^j = \begin{pmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{pmatrix}$, está formada por las coordenadas del $f(v_j)$ en la base $\{w_1, \ldots, w_m\}$.

Ejemplo 3.1.5 Sea $A = (a_{ij})$ una matriz $m \times n$ sobre un cuerpo $\mathbb{K}$. Definimos una aplicación $f_A : \mathbb{K}^n \to \mathbb{K}^m$ como sigue: dado un vector $(x_1, \ldots, x_n) \in \mathbb{K}^n$,

$$f_A(x_1, \ldots, x_n) = \left(\sum_{j=1}^n a_{1j}x_j, \sum_{j=1}^n a_{2j}x_j, \ldots, \sum_{j=1}^n a_{mj}x_j,\right).$$

la aplicación f_A así definida es lineal y nos referiremos a ella como la aplicación lineal asociada a la matriz A. Evidentemente la matriz asociada a la aplicación lineal f_A respecto de las bases canónicas de $\mathbb{K}^n$ y$\mathbb{K}^m$ es precisamente A.

Trataremos a continuación el problema de obtener la matriz asociada a una composición de aplicaciones lineales cuando se conocen las matrices asociadas a cada uno de los factores. Para ello, supongamos que tenemos dos aplicaciones lineales $g : U \to V$ y $f : V \to W$ tales que el espacio final de la primera coincide con el espacio inicial de la segunda. Fijadas las bases $\{u_1, \ldots, u_p\}$ de U, $\{v_1, \ldots, v_n\}$ de V y $\{w_1, \ldots, w_m\}$ de W, llamamos $N = (b_{hk})$ a la matriz $n \times p$ asociada a g, y $M = (a_{ij})$ a la matriz $m \times n$ asociada a f. La aplicación compuesta $f \circ g : U \to W$ también es lineal y con respecto a las bases $\{u_1, \ldots, u_p\}$ de U y $\{w_1, \ldots, w_m\}$ de W tendrá asociada una matriz $m \times p$ que denotaremos $P = (c_{rs})$, y vamos a tratar de encontrar el modo de relacionar la matriz P de $f \circ g$ con las matrices N, M de g y f respectivamente.

Comencemos fijando un vector u_s de la base de U. Para este vector se tiene

$$(f \circ g)(u_s) = c_{1s}w_1 + \cdots + c_{ms}w_m \tag{1}$$

y

$$g(u_s) = b_{1s}v_1 + \cdots + b_{ns}v_n \tag{2}$$

Además se tienen las relaciones

$$f(v_j) = a_{1j}w_1 + \cdots + a_{mj}w_m \tag{3}$$

Si en ambos miembros de (2) se toma la imagen por f resulta

$$f(g(u_s)) = b_{1s}f(v_1) + \cdots + b_{ns}f(v_n)$$

y sustituyendo la expresión (3) resulta

$$\begin{aligned} f(g(u_s)) &= b_{1s}(a_{11}w_1 + \cdots + a_{m1}w_m) \\ &+ b_{2s}(a_{12}w_1 + \cdots + a_{m2}w_m) \\ &+ \quad \cdots\cdots\cdots\cdots \\ &+ b_{ns}(a_{1n}w_1 + \cdots + a_{mn}w_m) \\ &= (b_{1s}a_{11} + b_{2s}a_{12} + \cdots + b_{ns}a_{1n})w_1 \\ &+ (b_{1s}a_{21} + b_{2s}a_{22} + \cdots + b_{ns}a_{2n})w_2 \\ &+ \quad \cdots\cdots\cdots\cdots \\ &+ (b_{1s}a_{m1} + b_{2s}a_{m2} + \cdots + b_{ns}a_{mn})w_m. \end{aligned}$$

Los paréntesis de esta última expresión son las coordenadas de $f(g(u_s)) = (f \circ g)(u_s)$ en la base $\{w_1, \ldots, w_m\}$ de W, y estas coordenadas son $(c_{1s}, \ldots, c_{ms}$ por (1). Como las coordenadas de un vector con respecto a determinada base son únicas, tenemos las igualdades

$$c_{rs} = b_{1s}a_{r1} + b_{2s}a_{r2} + \cdots + b_{ns}a_{rn} = \sum_{k=1}^{n} a_{rk}b_{ks} \tag{4}$$

$(r = 1, \ldots, m)$, y estas igualdades son válidas cualquiera que sea u_s $(s = 1, \ldots, p)$.

Hemos obtenido, por tanto, una fórmula que nos permite calcular la matriz P a partir de las matices N y M. Esta fórmula (4) se suele resumir diciendo que el elemento que ocupa la fila r-ésima y la columna s-ésima de P se obtiene multiplicando la fila r-ésima de M por la columna s-ésima de N. De acuerdo con la definición que estableceremos a continuación diremos que la matriz P de $f \circ g$ es el producto MN de la matriz de f por la matriz de g.

Dada una matriz $m \times n$, $M = (a_{ij})$, y una matriz $n \times p$, $N = (b_{hk})$, llamamos **matriz producto de** M **por** N y lo representaremos por MN a la matriz $m \times p$ cuyo elemento c_{rs} es

$$c_{rs} = \sum_{k=1}^{n} a_{rn}b_{ns}.$$

Es importante hacer notar que para que el producto MN esté definido el número de columnas de M ha de coincidir con el número de filas de N.

Observar que con esta notación, si identificamos los vectores $x \in \mathbb{K}^n$ con los vectores columna, para cada matriz cuadrada $f_A(x) = Ax$.

Dado el sistema de ecuaciones

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \quad \vdots \qquad \vdots \qquad\qquad \vdots \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

si A es la matriz de coeficientes y b es el vector columna de los términos independientes, podemos escribirlo en forma matricial como

$$Ax = b,$$

o utilizando operadores lineales, en la forma

$$f_A(x) = b.$$

Algunas matrices notables

Dada una matriz $M \in \mathcal{M}_{mn}(\mathbb{C})$, se define la **matriz adjunta** $M^* \in \mathcal{M}_{nm}$ de manera que

$$(Mu, v)_m = (u, M^*v)_n \text{ para todo } u \in \mathbb{C}^n \text{ y } v \in \mathbb{C}^m.$$

lo que implica que $(M^*)_{ij} = \overline{a_{ji}}$. M^* es la matriz obtenida al cambiar filas por columnas en A y tomar conjugados.

La **traspuesta** de la matriz $A \in \mathcal{M}_{mn}(\mathbb{R})$ es la matriz $A^t \in \mathcal{M}_{nm}$ definida de forma única por

$$(Mu, v)_m = (u, M^tv)_n \text{ para todo } u \in \mathbb{R}^n \text{ y } v \in \mathbb{R}^m,$$

y en consecuencia $(M^t)_{ij} = M_{ji}$.

Ejercicio 3.1.1 Comprobad que $(AB)^t = B^tA^t$ y que $(AB)^* = B^*A^*$.

De ahora en adelante y salvo que digamos lo contrario sólo vamos a considerar matrices cuadradas $A \in \mathcal{M}_n(\mathbb{K}) := \mathcal{M}_{nn}(\mathbb{K})$. **El espacio de las matrices cuadradas con la suma y el producto de matrices tiene estructura de anillo no conmutativo con elemento unidad.**

La matriz unidad con respecto al producto es $I \in \mathcal{M}_n(\mathbb{K})$, la matriz $I = (\delta_{ij})$ que tiene 1 en todos los términos de la diagonal principal y 0 fuera de la diagonal.

$$AI = IA \text{ para toda } A \in \mathcal{M}_n(\mathbb{K}).$$

Una matriz A se dice **inversible** cuando existe A^{-1} tal que $AA^{-1} = A^{-1}A = I$. En caso contrario se dice que la matriz es **singular**.

El cálculo de la matriz inversa de una matriz inversible A se puede identificar con el problema de resolver los sistemas lineales $Ax = e_j$ donde e_j es la base canónica de $\mathbb{K}^n$, pues sus soluciones son los vectores columna de A^{-1}. Así, por ejemplo, cuando en el siguiente capítulo escribamos el algoritmo de Gauss de resolución de sistemas, podremos modificarlo para invertir matrices fácilmente.

Ejercicio 3.1.2 Comprobad que si A y B son inversibles, entonces

$$(AB)^{-1} = B^{-1}A^{-1},\ (A^t)^{-1} = (A^{-1})^t), (A^*)^{-1} = (A^{-1})^*).$$

Una matriz A se dice

- **simétrica**, si $A \in \mathcal{M}_n(\mathbb{R})$ y $A = A^t$.
- **hermitiana**, si $A = A^*$.
- **ortogonal**, si $A \in \mathcal{M}_n(\mathbb{R})$ y $A^{-1} = A^t$, e.d. $AA^t = A^tA = I$.
- **unitaria**, si $A^{-1} = A^*$, e.d. $AA^* = A^*A = I$.
- **normal**, si $AA^* = A^*A$.

Ejercicio 3.1.3 Observad que una matriz es ortogonal-unitaria si, y solo si, los vectores fila (resp vectores columna) forman una base ortonormal de $\mathbb{K}^n$.

Una matriz A se dice **diagonal** cuando $a_{ij} = 0$ para $i \neq j$.

La **traza** de una matriz $A = (a_{ij})$ está definida por

$$tr(A) = \sum_{i=1}^{n} a_{ij}.$$

El **determinante** de una matriz A está definido por

$$det(A) = \sum_{\sigma \in \mathcal{G}_n} \varepsilon_\sigma a_{\sigma(1)1} a_{\sigma(2)2} ... a_{\sigma(n)n},$$

donde $\mathcal{G}_n$ es el grupo de las permutaciones de $\{1, 2, \ldots, n\}$ y ε_σ es la signatura de la permutación.

Recordad que $det(AB) = det(BA) = det(A)det(B)$.

Recordad que un sistema de ecuaciones $Ax = b$ es compatible determinado, si y sólo si, A es inversible, si y sólo si $det(A) \neq 0$.

Ejercicio 3.1.4 (Regla de Cramer) Si $A = (a_{ij})$ es una matriz no singular $n \times n$ y $b = (b_1, ..., b_n)^t \in \mathbb{K}^n$, la solución al sistema lineal $Ax = b$ tiene por coordenadas

$$x_j = \frac{det(C_j)}{det(A)}, \quad \text{donde } C_j = (c_{rs}) \ c_{rs} = \begin{cases} a_{rs} & \text{si } s \neq j \\ b_r & \text{si } s = j \end{cases}$$

Los **valores propios** de una matriz A son los complejos $\lambda \in \mathbb{C}$ para los que existen vectores no nulos p tales que $Ap = \lambda p$.También se dice que p es un vector propio de λ. Los valores propios de A son los ceros del **polinomio característico**

$$p_A(\lambda) = det(A - \lambda I)$$

de la matriz A.

El **espectro** de la matriz A es el conjunto de los vectores propios

$$\sigma(A) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}.$$

El radio espectral de A se define como

$$\rho(A) = \text{máx}\{|\lambda_1|, |\lambda_2|, \ldots, |\lambda_n|\}.$$

Recordad que $tr(A) = \sum_{i=1}^{n} \lambda_i$ y que $det(A) = \prod_{i=1}^{n} \lambda_i$.

Ejercicio 3.1.5 Comprobad que si A es simétrica o hermitiana, entonces todos sus valores propios son números reales.

Comprobad también que si A es ortogonal o unitaria, entonces todos sus valores propios tienen módulo 1.

3.1.4. Reducción de matrices

Sea $Ax = b$ un sistema de ecuaciones, sea una matriz inversible P (un cambio de base en $\mathbb{K}^n$) y $B = P^{-1}AP$. Si u es una solución del sistema de ecuaciones $Bu = P^{-1}b$, entonces $x = Pu$ es solución del sistema $Ax = b$, en efecto:

$$P^{-1}APu = P^{-1}b \rightarrow APu = b \rightarrow x = Pu.$$

Reducir una matriz A consiste en encontrar una matriz inversible P tal que $P^{-1}AP$ sea tan simple como sea posible.

Los casos más favorables en este sentido se tienen cuando $P^{-1}AP$ es una matriz diagonal o una matriz escalonada (triangular). En estos casos los elementos de la diagonal son los valores propios de A y la resolución de los sistemas de ecuaciones es muy simple.

Cuando existe P tal que $P^{-1}AP$ es diagonal, se dice que A es diagonalizable (en este caso los vectores columna de P forman una base formada por vectores propios de A).

Aunque no todas las matrices son diagonalizables (probad con $\left(\begin{smallmatrix} 1 & 1 \\ 0 & 1 \end{smallmatrix}\right)$), siempre se puede encontrar un buen cambio de base que las transforma en matrices triangulares:

Teorema 3.1.6

(I) *Dada una matriz cuadrada A, existe una matriz unitaria U tal que la matriz $U^{-1}AU$ es una matriz triangular superior, i.e. $U^{-1}AU(i,j) = 0$ si $i > j$.*

(II) *Dada una matriz normal A, existe una matriz unitaria U tal que la matriz $U^{-1}AU$ es una matriz diagonal.*

(III) *Dada una matriz simétrica A, existe una matriz ortogonal O tal que la matriz $O^{-1}AO$ es una matriz diagonal.*

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 1.2]
Ideas que intervienen

- *(1) Primero se prueba por inducción la existencia de un cambio de base (no necesariamente ortonormal) que transforma la matriz A en una triangular.*
- *Ortonormalizando la base anterior con el método de Gram-Schmidt se obtiene el cambio de base ortonormal que produce la matriz unitaria buscada.*
- *(2) $T = U^{-1}AU = U^*AU$. Si A es normal ($A^*A = AA^*$), T también es normal:*

  $$T^*T = U^*A^*UU^*AU = U^*A^*AU$$

  Si T es triangular superior y normal, entonces T es diagonal.
- *(3) Si A es simétrica, los pasos de la prueba de (1) y (2) se pueden hacer sin salir de $\mathbb{R}$, por lo tanto cambiando adjuntas por transpuestas y unitaria por ortogonal se obtiene la prueba.*

□

Se denominan valores singulares de una matriz A a las raíces cuadradas positivas de los valores propios de la matriz hermitiana A^*A (o A^tA si A es real) que siempre son positivos (comprobadlo como ejercicio). Los valores singulares son todos > 0 si A es no singular. En efecto:

$$Ap = 0 \rightarrow A^*Ap = 0 \rightarrow p^*A^*Ap = 0 \rightarrow (Ap)^*(Ap) = 0 \rightarrow Ap = 0.$$

Teorema 3.1.7 *Si A es una matriz real cuadrada, existen dos matrices ortogonales U y V, tales que $U^tAV = diagonal(\mu_i)$. Y si A es una matriz compleja cuadrada, existen dos matrices unitarias U y V, tales que $U^tAV = diagonal(\mu_i)$.*

En los dos casos los números $\mu_i \geq 0$ son los valores singulares de A.

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 1.2]
Ideas que intervienen

- *Diagonalizar la matriz normal A^*A, haciendo*

  $$V^*A^*AV = diagonal(\mu_i^2)$$

  con V unitaria/ortogonal.

- *si f_j es el vector columna de AV y $f_j \neq 0$ ponemos $u_j = \frac{1}{\mu_j} f_j$.*

  Los u_j así definidos tienen norma 1 y son 2-2 ortogonales.

  Se completa la lista u_j (en los casos $f_j = 0$) con vectores ortonormales para tener una base ortonormal.

- *la matriz U con vectores columna u_j es unitaria/ortogonal y cumple $U^tAV = diagonal(\mu_i)$.*

□

3.1.5. Cocientes de Rayleigh. Matrices simétricas y hermitianas

En este apartado vamos a enunciar los resultados para matrices complejas hermitianas aunque también se aplican a las matrices reales simétricas simplemente sustituyendo los adjetivos «hermitiana», «unitaria», «adjunta» y «compleja» por «simétrica», «ortogonal», «traspuesta» y «real».

Para caracterizar los valores propios de una matriz hermitiana (recordad que todos son números reales) vamos a hacer uso de la siguiente definición

Definición 3.1.8 (Cocientes de Rayleigh) *Sea A una matriz cuadrada de dimensión n. El cociente de Rayleigh de A es la aplicación $R_A : \mathbb{C} \setminus \{0\} \rightarrow \mathbb{C}$ definida por*

$$R_A(v) = \frac{(Av, v)}{(v, v)} = \frac{v^*Av}{v^*v}.$$

Si la matriz A es hermitiana, los cocientes de Rayleigh toman valores reales:

$$\overline{R_A(v)} = \frac{\overline{(Av, v)}}{(v, v)} = \frac{(v, Av)}{(v, v)} = \frac{(A^*v, v)}{(v, v)} = \frac{(Av, v)}{(v, v)} = R_A(v).$$

Observad también que R_A toma los mismos valores en toda la semirecta definida por v:

$$R_A(v) = R_A\left(\frac{v}{\|v\|_2}\right).$$

Teorema 3.1.9 *Sea A una matriz hermitiana de dimensión n, con valores propios*

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n,$$

y con una base ortonormal de vectores propios $p_1, ..., p_2$, e.d. tales que

$$(p_i, p_j) = \delta_{ij}, \ y \ Ap_i = \lambda_i p_i.$$

Para cada $k = 1, \ldots, n$ sea E_k el subespacio generado por $\{p_1, p_2, ..., p_k\}$, y denotemos por $\mathcal{S}_k$ a la familia de todos los subespacios de $\mathbb{C}^n$ con dimensión k. Pongamos también $E_0 = \{0\}$ y $\mathcal{S}_k = \{E_0\}$.

Los valores propios de A admiten las siguientes caracterizaciones

(I) $\lambda_k = R_A(p_k)$.

(II) $\lambda_k = \text{máx}\{R_A(v) : v \in E_k \setminus \{0\}\}$.

(III) $\lambda_k = \text{mín}\{R_A(v) : v \perp E_{k-1}\}$.

(IV) $\lambda_k = \text{mín}_{E \in \mathcal{S}_k} \text{máx}\{R_A(v) : v \in E \setminus \{0\}\}$. *(Courant-Fischer)*

(V) $\lambda_k = \text{máx}_{E \in \mathcal{S}_{k-1}} \text{mín}\{R_A(v) : v \perp E\}$. *(Courant-Fischer)*

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 1.3]
Ideas que intervienen

- *Considerar la matriz unitaria U cuyos vectores columna son los p_i, que diagonaliza A:*

$$U^*AU = diag(\lambda_i) =: D.$$

- *Poniendo $v = Uw$, e.d. tomando $w = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$ el vector columna de las coordenadas de v con respecto a la base $\{p_1, ..., p_2\}$, se tiene*

$$R_A(v) = \frac{v^*AV}{V^*V} = \frac{w^*U^*AUw}{w^*U^*Uw} = \frac{D}{w^*w} = \frac{\sum \lambda_i |w_i|^2}{\sum |w_i|^2}.$$

- *De la igualdad anterior se deducen (I), (II) y (III).*
- *Las otras dos identidades se deducen de las tres primeras.*

□

Como caso particular de este teorema se tiene que

$$\lambda_1 = \text{mín}\{R_A(v) : \|v\|_2 = 1\} \text{ y } \lambda_n = \text{máx}\{R_A(v) : \|v\|_2 = 1\}.$$

En la última sección del capítulo utilizaremos estas caracterizaciones para calcular números de condición de sistemas lineales.

Una matriz hermitiana se dice "definida positiva" (resp. positiva) si

$$(Av, v) = v^*Av > 0 \quad (\text{resp. } (Av, v) = v^*Av \geq 0) \text{ para todo } v \neq 0.$$

Una matriz hermitiana definida positiva tiene todos sus valores propios positivos ($\lambda_1 > 0$).

3.2. Origen de los problemas del análisis numérico matricial

Existen muchos problemas cuyas modelizaciones involucran la resolución de sistemas lineales o el cálculo de valores y vectores propios de matrices.

Dentro de los métodos numéricos objeto de este curso algunas soluciones a problemas no lineales pasan por aproximar el problema a uno lineal resolverlo y después iterando el proceso aproximarnos a la solución. Por ejemplo esta es la situación en el método de Newton para sistemas de ecuaciones no lineales. También aparecen en problemas de interpolación por splines o en problemas de aproximación por mínimos cuadrados que también forman parte de este curso.

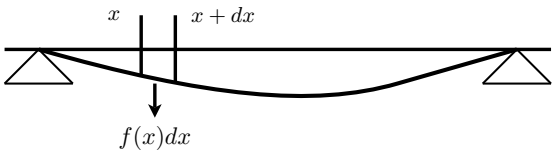
En el libro de Ciarlet [3] hay todo un capítulo (el capítulo 3) dedicado a mostrar aproximaciones lineales a las ecuaciones diferenciales en derivadas parciales de la Física (movimiento armónico, ecuación del calor, ...) que resultan al discretizar los problemas. En todos los casos aparecen sistemas con muchas ecuaciones y matrices de coeficientes con aspectos particulares donde los ceros aparecen dispuestos de forma especial (matrices tridiagonales o tridiagonales por bloques).

En este apartado vamos a reproducir el método de las diferencias finitas para una ecuación diferencial lineal de segundo grado en dimensión 1 con condiciones frontera [3, sección 3.1]:

Consideremos dos funciones continuas en [0,1], $c(x), f(x) \in \mathcal{C}([0,1])$, y dos constantes $a, b \in \mathbb{R}$. Problema: Encontrar $u(x) \in \mathcal{C}^2([0,1])$ tal que

$$\begin{cases} -u''(x) + c(x)u(x) = f(x), & 0 < x < 1, \\ u(0) = a, \ \ u(1) = b. \end{cases}$$

Un ejemplo de una situación física donde aparece este problema es el del estudio de los momentos $u(x)$ de la «flecha» de una viga de longitud 1, estirada por a lo largo de su eje por la acción de una fuerza (que determina c(x)) y sometida a la acción de una carga perpendicular $f(x)dx$ en cada punto de abcisa x.



Para obtener aproximaciones de $u(x)$ con el método de las diferencias finitas, se consideran particiones equidistribuidas de [0,1], $0 = x_0 < x_1 < ... < x_{N+1} = 1$, $x_i = i/(N+1)$, $h = x_i - x_{i-1} = 1/(N+1)$, y se hacen las aproximaciones:

$$\begin{cases} u'(x_i) \approx \frac{u(x_i)-u(x_{i-1})}{h}, & i = 1, 2, ..., N+1, \\ u''(x_i) \approx \frac{u'(x_{i+1})-u'(x_i)}{h} \approx \frac{u(x_{i+1})-2u(x_i)+u(x_{i-1})}{h^2}, & i = 1, 2, ..., N. \end{cases}$$

Después se trasladan estas aproximaciones a la ecuación diferencial y se tiene el sistema de N ecuaciones lineales

$$-\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} + c(x_i)u(x_i) = f(x_i), \quad i = 1, ..., N,$$

con incógnitas $u_i = u(x_i)$, $i = 1, ..., N$ ($u(x_0) = a$ y $u(x_{N+1}) = b$ son las condiciones frontera). El sistema de ecuaciones se puede escribir

$$\begin{cases} (c(x_1)h^2+2)u_1 & -u_2 & & & = h^2 f(x_1)+a \\ -u_1 & +(c(x_2)h^2+2)u_2 & -u_3 & & = h^2 f(x_2) \\ \ddots & \ddots & \ddots & & \\ -u_{N-2} & +(c(x_{N-1})h^2+2)u_{N-1} & -u_N & & = h^2 f(x_{N-1}) \\ & -u_{N-1} & +(c(x_N)h^2+2)u_N & & = h^2 f(x_N)+b \end{cases}$$

La matriz de coeficientes tiene una forma muy particular, es de las llamadas tridiagonales (sus coeficientes son cero fuera de la diagonal principal de la matriz y de las dos diagonales contiguas).

$$\begin{pmatrix} (c(x_1)h^2+2) & -1 & & & \\ -1 & (c(x_2)h^2+2) & -1 & & \\ & \ddots & \ddots & \ddots & \\ & -1 & (c(x_{N-1})h^2+2) & -1 & \\ & & -1 & (c(x_N)h^2+2) & \end{pmatrix}$$

En el capítulo siguiente veremos métodos para la resolución de sistemas lineales con matrices de coeficientes tridiagonales como la de este modelo.

Observad que para tener buenas aproximaciones numéricas a la solución $u(x)$ del problema hay que considerar sistemas con muchas ecuaciones correspondientes a considerar muchos puntos.

Ejemplos como este muestran la necesidad de disponer de métodos estables de resolución de ecuaciones capaces de trabajar con muchas ecuaciones y con ecuaciones con muchos ceros.

3.3. Normas matriciales

Sea E un espacio vectorial sobre el cuerpo $\mathbb{K}$. Una **norma** sobre E es una aplicación $\|\ \| : E \to [0,+\infty)$ que cumple las propiedades

(I) $\|x\| = 0$ si, y sólo si, $x = 0$.

(II) $\|x+y\| \leq \|x\| + \|y\|$ para todo $x, y \in E$ (desigualdad triángular).

(III) $\|ax\| = |a|\|x\|$ para todo $x \in E$ y todo $a \in \mathbb{K}$.

al par $(E, \|\ \|)$ se le llama espacio vectorial normado. La función $d(x,y) = \|x-y\|$ define una distancía en E, y la topología asociada a esta métrica.

Todas las normas definidas en un espacio de dimensión finita E son equivalentes en el sentido de que todas definen la misma topología.

En $\mathbb{K}^n$ las tres normas más utilizadas en la práctica son

- $\|x\|_1 = \sum_{i=1}^n |x_i|$,
- $\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2} = \sqrt{x^* x} = \sqrt{(x,x)}$,
- $\|x\|_\infty = \max\{|x_i| : i = 1, ..., n\}$.

Donde $x^t = (x_1, x_2, ..., x_n)$.

Una **norma matricial** en el anillo de las matrices $\mathcal{M}_n(\mathbb{K})$ de orden n es una norma $\|\ \|$ definida en $\mathcal{M}_n(\mathbb{K})$ que además cumple

$$\|AB\| \leq \|A\|\|B\|, \text{ para todo } A, B \in \mathcal{M}_n(\mathbb{K}).$$

($\mathcal{M}_n(\mathbb{K})$ es isomorfo a $\mathbb{K}^{n^2}$, ¿es una norma matricial la norma $\|A\| = \sum_{i,j=1}^{n} |A(i,j)|$?)

Dada una norma $\|\ \|$ en $\mathbb{K}^n$, utilizando la identificación de las matrices A con los operadores lineales que definen f_A, podemos considerar la norma de la aplicación lineal f_A para definir la **norma matricial subordinada** a la norma $\|,\|$ de $\mathbb{K}^n$. Esta norma se define por

$$\|A\| = \sup\{\frac{\|Ax\|}{\|x\|} : x \in \mathbb{K}^n - 0\} = \sup\{\frac{\|Ax\|}{\|x\|} : \|x\| \leq 1\} = \sup\{\frac{\|Ax\|}{\|x\|} : \|x\| = 1\}.$$

Observad que estas normas subordinadas tienen la propiedad siguiente:

$$\|Ax\| \leq \|A\|\|x\| \text{ para todo } A \in \mathcal{M}_n(\mathbb{K}) \text{ y } x \in \mathbb{K}^n.$$

Lo que las hace interesantes para intentar medir la estabilidad y el condicionamiento de los sistemas lineales $Ax = b$.

Existen normas matriciales que no están subordinadas a ninguna norma de $\mathbb{K}^n$ como mostraremos más adelante.

El siguiente teorema ofrece información sobre las normas subordinadas a las más usuales

Teorema 3.3.1 *Sea $A = (a_{ij}) \in \mathcal{M}_n(\mathbb{C})$ una matriz cuadrada. Entonces*

$$\|A\|_1 := \sup\left\{\frac{\|Ax\|_1}{\|x\|_1} : \|x\|_1 = 1\right\} = \max_j \sum_i |a_{ij}|$$

$$\|A\|_\infty := \sup\left\{\frac{\|Ax\|_\infty}{\|x\|_\infty} : \|x\|_\infty = 1\right\} = \max_i \sum_j |a_{ij}|$$

$$\|A\|_2 := \sup\left\{\frac{\|Ax\|_2}{\|x\|_2} : \|x\|_1 = 1\right\} = \sqrt{\rho(A^*A)} = \|A^*\|_2$$

La norma $\|\ \|_2$ es invariante por transformaciones unitarias

$$UU^* = I \Rightarrow \|A\|_2 = \|AU\|_2 = \|UA\|_2 = \|U^*AU\|_2.$$

Si además la matriz A es normal ($A^*A = AA^*$):

$$\|A\|_2 = \rho(A).$$

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 1.4]
Ideas que intervienen

- *Para las norma $\|\ \|_1$ y $\|\ \|_\infty$ sólo hay que utilizar la definición y buscar los supremos correspondientes.*
- *Para la $\|\ \|_2$ utilizaremos los cocientes de Rayleigh de la sección anterior*

$$\|A\|_2^2 = \sup\left\{\frac{(Ax, Ax)}{(x, x)} = \frac{(A^*Ax, x)}{(x, x)} = R_{A^*A}(x) : \|x\|_1 = 1\right\}.$$

□

Observad que

- La norma $\|A\|_2$ coincide con el mayor *valor singular* de A.

  Si U es una matriz unitaria/ortogonal se tiene $\|U\|_2 = 1$.

- Las normas $\| \|_1$ y $\| \|_\infty$ son mucho más fáciles de calcular que la norma $\| \|_2$

- Se puede probar (ver [3, 1.4.4]) que la norma euclídea en $\mathcal{M}_n \equiv \mathbb{K}^{n^2}$, $\|A\|_E = \sqrt{\sum_{i,j} |a_{ij}|^2}$ es una norma matricial no subordinada a ninguna norma en $\mathbb{K}^n$ pero que sirve para acotar la norma $\| \|_2$ y es más fácil de calcular que ésta.

$$\|A\|_2 \leq \|A\|_E \leq \sqrt{n}\|A\|_2.$$

Si A es normal $\|A\|_2 = \rho(A)$. Aunque esta identidad no es cierta para todas las matrices, el radio espectral tiene la siguiente propiedad:

Teorema 3.3.2 *Si A es una matriz cuadrada y $\| \|$ es una norma matricial (subordinada o no), entonces*

$$\rho(A) \leq \|A\|.$$

En sentido recíproco, si A es una matriz y $\varepsilon > 0$, existe una norma matricial subordinada tal que

$$\|A\| \leq \rho(A) + \varepsilon.$$

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 1.4]
Ideas que intervienen

- *Sea $p \neq 0$ un vector propio del valor propio λ, $|\lambda| = \rho(A)$, y sea q un vector tal que la matriz $pq^t \neq 0$.* ***(¡Escribe la matriz!)***

  $$\rho(A)\|pq^t\| = \|\lambda pq^t\| = \|(Ap)q^t\| = \|A(pq^t)\| \leq \|A\| \, \|pq^t\|,$$

  y despejando queda $\rho(A) \leq \|A\|$.

- *Para la segunda parte, la idea es considerar el cambio de base que triangula la matriz A e perturbándolo de forma que casi-diagonalize la matriz A. La norma del supremo al hacer este cambio de base será la que proporciona la norma subordinada del enunciado.*

En efecto, sea U la matriz unitaria tal que

$$U^{-1}AU = \begin{pmatrix} \lambda_1 & u_{12} & u_{13} & & \dots & u_{1n} \\ & \lambda_2 & u_{23} & & \dots & u_{2n} \\ & & \ddots & \ddots & & \vdots \\ & & & & \lambda_{n-1} & u_{n-1,n} \\ & & & & & \lambda_n \end{pmatrix}$$

En la diagonal aparecen los valores propios de A.

Para $\delta > 0$ se toma la matriz diagonal $D_\delta = diag(1, \delta, \delta^2, ..., \delta^{n-1})$. Entonces

$$(UD_\delta)^{-1} A (UD_\delta) = \begin{pmatrix} \lambda_1 & \delta u_{12} & \delta^2 u_{13} & & ... & \delta^{n-1} u t_{1n} \\ & \lambda_2 & \delta u_{23} & & ... & \delta^{n-2} u_{2n} \\ & & \ddots & \ddots & & \vdots \\ & & & & \lambda_{n-1} & \delta u_{n-1,n} \\ & & & & & \lambda_n \end{pmatrix}$$

Tomando δ suficientemente pequeño se tiene $\sum_{j=i+1}^{n} |\delta^{j-i} u_{ij}| < \varepsilon$ para $i = 1, 2, .., n-1$ y por lo tanto (teorema 3.3.1) que

$$\|(UD_\delta)^{-1} A (UD_\delta)\|_\infty \leq \rho(A) + \varepsilon.$$

Si consideramos la norma $\|v\|_* = \|UD_\delta)^{-1} v\|_\infty$, se tiene que la norma subordinada a ésta verifica la propiedad buscada:

$$\|A\|_* = \|(UD_\delta)^{-1} A (UD_\delta)\|_\infty \leq \rho(A) + \varepsilon.$$

□

Corolario 3.3.3

$$\rho(A) = \text{ínf}\{\|A\| : \|\ \| \text{ es una norma matricial } \}.$$

3.3.1. Convergencia de matrices

Recordad que en dimensión finita la convergencia de las sucesiones es independiente de la norma considerada porque todas las normas son equivalentes.

El siguiente teorema caracteriza las sucesiones de potencias B^k de matrices que convergen a 0 en términos del radio espectral. De este resultado deduciremos los métodos iterativos de resolución de sistemas de ecuaciones lineales.

Teorema 3.3.4 *Si B es una matriz cuadrada, entonces las siguientes condiciones son equivalentes:*

(I) $\lim_{k\to\infty} B^k = 0$,

(II) $\lim_{k\to\infty} B^k v = 0$, *para todo vector v.*

(III) $\rho(B) < 1$.

(IV) Existe una norma subordinada tal que $\|B\| < 1$.

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 1.5]
Ideas que intervienen

- *IV $\Rightarrow$ I. Basta con utilizar la desigualdad* $\|B^k\| \leq \|B\|^k$.
- *I $\Rightarrow$ II. Basta con utilizar la desigualdad* $\|B^k v\| \leq \|B^k\|\,\|v\|$, *válida para las normas subordinadas.*

- *II* $\Rightarrow$ *III*. *Sea* λ *un valor propio con* $|\lambda| = \rho(B)$ *y* p *un vector propio asociado, entonces* $B^k p = (\lambda)^p v \to 0$, $(\lambda)^p \to 0$, *y* $|\lambda| = \rho(B) < 1$.
- *III* $\Rightarrow$ *IV*. *Es el teorema 3.3.2* .

□

El radio espectral también servirá para medir la rapidez con que convergen los métodos iterativos.

Teorema 3.3.5 *Para cualquier norma matricial se cumple*

$$\lim_{k\to\infty} \|B^k\|^{\frac{1}{k}} = \rho(B).$$

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 1.5]
Ideas que intervienen

- $\rho(B) \leq \|B\|$ *y* $\rho(B^k) = \rho(B)^k$ *implica que*

$$\rho(B) = \rho(B^k)^{\frac{1}{k}} \leq \|B^k\|^{\frac{1}{k}}.$$

- *Para* $\varepsilon > 0$, *la matriz* $B_\varepsilon = \frac{1}{\rho(B)+\varepsilon} B$ *tiene radio espectral* $\rho(B_\varepsilon) < 1$.

  Por el teorema anterior $\lim_k B_\varepsilon^k = 0$, *por lo tanto para* k *suficientemente grande* $\|B_\varepsilon^k\| < 1$ *y entonces*

$$\rho(B) =\leq \|B^k\|^{\frac{1}{k}} = \|B_\varepsilon^k\|^{\frac{1}{k}} \rho(B) + \varepsilon \leq \rho(B) + \varepsilon.$$

□

3.4. Análisis del error. Condicionamiento

Recordad el ejemplo de la introducción:

Ejemplo 1.1.5 *Consideremos el sistema de ecuaciones*

$$\begin{cases} x & + & y = 2 \\ x & +1.00001 & y = 2.00001. \end{cases}$$

Su solución, es $x = 1$ *e* $y = 1$.

Consideremos ahora el sistema perturbando un poco los términos independientes.

$$\begin{cases} x & + & y = 2 \\ x & +1.00001 & y = 2. \end{cases}$$

Este sistema perturbado tiene como solución $x = 2$ *e* $y = 0$.

- *Un error pequeño en los datos iniciales (en el término independiente) ha producido un error grande en la solución.*

- *En este caso, no se ha producido ningún error de cálculo.*
- *Este es un ejemplo de un problema (sistema lineal)* ***mal condicionado****. No podemos evitar que pequeños errores en los datos iniciales se conviertan en errores grandes en las soluciones.*

Si en lugar de perturbar el término independiente se perturba la matriz

$$\begin{cases} 1.00001x & + & y = 2 \\ x & +1.00001y = 2.00001. \end{cases}$$

la solución pasa a ser $x = 0.9999E-5$ e $y = 1.99990001$.

- Un error pequeño en la matriz del sistema también ha producido un error grande en la solución.

Vamos a estimar el número de condición de un sistema lineal atendiendo a estos fenómenos.

- Supongamos que A es una matriz invertible (no singular) y que consideramos las soluciones de los sistemas
$$Ax = b \quad \text{y} \quad A(x + \Delta x) = b + \Delta b.$$
Se deduce fácilmente que $A\Delta x = \Delta b$ y que $\Delta x = A^{-1}\Delta b$.
Para cualquier norma subordinada se cumple:
$$\|\Delta x\| \leq \|A^{-1}\| \, \|\Delta b\| \quad \text{y} \quad \|b\| \leq \|A\|\|x\|.$$
Acotando el error relativo en x en función del error relativo en b se tiene
$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\| \, \|\Delta b\|}{\|x\|} = \|A^{-1}\|\|A\| \frac{\|\Delta b\|}{\|A\|\|x\|} \leq \|A^{-1}\|\|A\| \frac{\|\Delta b\|}{\|b\|}.$$
- Consideramos ahora las soluciones de los sistemas
$$Ax = b \quad \text{y} \quad (A + \Delta A)(x + \Delta x) = b.$$
Se deduce fácilmente que $A\Delta x = -\Delta A(x+\Delta x)$ y que $\Delta x = A^{-1}A(x+\Delta x)$ Para cualquier norma subordinada se cumple:
$$\|\Delta x\| \leq \|A^{-1}\|\|\Delta A\|\|x + \Delta x\|.$$
Acotando el error relativo en x (esta vez con respecto a $\|x + \Delta x\|$) en función del error relativo en A se tiene
$$\frac{\|\Delta x\|}{\|x + \Delta x\|} \leq \|A^{-1}\| \, \|\Delta A\| = \|A^{-1}\|\|A\| \frac{\|\Delta A\|}{\|A\|}.$$

Recordando la noción de condicionamiento de un problema y la de número de condición (ver 1.7), en el sentido de medir la sensibilidad de la solución de un sistema de ecuaciones lineales con respecto a perturbaciones en el término independiente o en la matriz de coeficientes, podemos establecer la siguiente definición:

Definición 3.4.1 *Dada una norma matricial subordinada $\|\ \|$ y una matriz cuadrada invertible A, se define el número de condición de la matriz A con respecto a esta norma por*

$$cond(A) = \|A\|\,\|A^{-1}\|.$$

Los dos teoremas siguientes prueban que el número de condición que acabamos de definir es la mejor acotación posible en las dos desigualdades que hemos encontrado para definirlo

Teorema 3.4.2 *Sea A una matriz invertible, x y $x + \Delta x$ las soluciones de los sistemas*

$$Ax = b \quad y \quad A(x + \Delta x) = b + \Delta b.$$

Si $b \neq 0$ entonces

$$\frac{\|\Delta x\|}{\|x\|} \leq cond(A)\frac{\|\Delta b\|}{\|b\|}.$$

Además esta desigualdad es la mejor posible en el sentido de que se pueden encontrar un vector $b \neq 0$ y un vector $\Delta b \neq 0$ para los que la desigualdad se convierte en igualdad.

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 2.2]
Ideas que intervienen

- *La desigualdad se probó con antelación.*
- *Para la igualdad, observad que la compacidad de la bola unidad de $\mathbb{K}$ proporciona $u \neq 0$ tal que $\|Au\| = \|A\|\|u\|$ $(b = Au)$ y $\Delta b \neq 0$ tal que $\|A^{-1}(\Delta b)\| = \|A^{-1}\|\|\Delta b\|$.*

□

Teorema 3.4.3 *Sea A una matriz invertible, x y $x + \Delta x$ las soluciones de los sistemas*

$$Ax = b \quad y \quad (A + \Delta A)(x + \Delta x) = b.$$

Si $b \neq 0$ entonces

$$\frac{\|\Delta x\|}{\|x + \Delta x\|} \leq \|A^{-1}\|\,\|\Delta A\| = \|A^{-1}\|\|A\|\frac{\|\Delta A\|}{\|A\|}.$$

Además esta desigualdad es la mejor posible en el sentido de que se pueden encontrar un vector $b \neq 0$ y una matriz $\Delta A \neq 0$ para los que la desigualdad se convierte en igualdad.

DEMOSTRACIÓN: [Prueba completa en Ciarlet [3] sección 2.2] □

La definición del número de condición depende de la norma subordinada considerada. Denotaremos $cond_2(A) = \|A^{-1}\|_2\|A\|_2$ al número de condición correspondiente a la norma euclidea.

Reuniendo los resultados que hemos ido obteniendo a lo largo del capítulo podemos obtener el siguiente teorema:

Teorema 3.4.4

(I) Para toda matriz A invertible se cumple

a) $cond(A) \geq 1$,

b) $cond(A) = cond(A^{-1})$.

c) $cond(\alpha A) = cond(A)$ *para todo escalar* α.

(II) Para toda matriz A

$$cond_2(A) = \frac{\mu_n(A)}{\mu_1(A)}$$

donde $\mu_1(A)$ y $\mu_n(A)$ son el menor y el mayor valor singular de A.

(III) Para toda matriz normal A

$$cond_2(A) = \rho(A)\rho(A^{-1}) = \frac{|\lambda_n(A)|}{|\lambda_1(A)|}$$

donde $|\lambda_1(A)|$ y $|\lambda_n(A)|$ son los valores propios de A de menor y mayor tamaño.

(IV) Para toda matriz unitaria U u ortogonal O se cumple $cond_2(U) = cond_2(O) = 1$.

(V) El número de condición $cond_2(A)$ es invariante por transformaciones unitarias (cambios de base ortonormales):

$$cond_2(A) = cond_2(UA) = cond_2(AU) = cond_2(U^{-1}AU).$$

Las afirmaciones (II y III) nos permiten pensar que el número de condición $cond_2(A)$ correspondiente a la norma euclídea es el *mejor* en el sentido de que proporciona la medida más pequeña del condicionamiento de los sistemas de ecuaciones lineales.

3.5. Actividades complementarias del capítulo

Los documentos se pueden descargar de la Zona Compartida de SUMA.

- Hoja de problemas nº3

Bibliografía

[1] A. Delshams A. Aubanell, A. Benseny, *Útiles básicos de cálculo numérico*, Ed. Labor - Univ. Aut. de Barcelona, Barcelona, 1993.

[2] R.L. Burden and J.D. Faires, *Análisis numérico, 7^a edición*, Thomson-Learning, Mexico, 2002.

[3] P.G. Ciarlet, *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson, Paris, 1990.

[4] D. Kincaid and W. Cheney, *Análisis numérico*, Ed. Addison-Wesley Iberoamericana, Reading, USA, 1994.

Capítulo 4

Métodos directos para resolver sistemas de ecuaciones

Interrogantes centrales del capítulo

- Analizar técnicas de resolución de sistemas de ecuaciones lineales.
- Aprender métodos de resolución :
  - Método de Gauss.
  - Factorización LU y Choleski.
  - Factorización QR, método de Householder.

Destrezas a adquirir en el capítulo

- Resolver sistemas de ecuaciones lineales utilizando los métodos directos del álgebra lineal.
- Implementar los métodos en el ordenador.
- Comparar su estabilidad y eficacia, y elegir el más adecuado a cada problema.
- Aplicar los programas construidos de forma efectiva en la búsqueda de la solución de sistemas lineales concretos.

En esta unidad se estudian distintos métodos directos de resolución de sistemas de ecuaciones lineales

$$Ax = b.$$

La complejidad numérica de cada uno de los métodos se va a medir en función del número de operaciones que requiere y la estabilidad en los cálculos se va a analizar en función del tipo de operaciones que se realizan.

En los tres métodos propuestos el número de operaciones para un sistema $n \times n$ es del mismo orden, $O(n^3)$. La elección entre uno u otro dependerá de la naturaleza del sistema y del control que tengamos sobre la estabilidad de los cálculos.

En los tres métodos propuestos se simplifica el problema de resolver un sistema de ecuaciones, reduciéndolo a resolver sistemas fáciles (triangulares). Usando estas reducciones, con cualquiera de los métodos descritos permite calcular el determinante de una matriz.

El problema de calcular la inversa A^{-1} de una matriz no singular A se reduce a la resolución de n sistemas lineales. Si $\{\vec{e_1}, ..., \vec{e_n}\}$ es la base canónica de $\mathbb{K}^n$,

$$A^{-1} = (\vec{u_1}, ..., \vec{u_n}) \quad \text{donde} \quad A\vec{u_k} = \vec{e_k}.$$

4.1. Sistemas fáciles de resolver

4.1.1. Sistemas diagonales

Si la matriz de coeficientes A es diagonal y no singular,

$$\begin{pmatrix} a_{11} & 0 & ... & 0 \\ 0 & a_{22} & ... & 0 \\ . & . & ... & . \\ 0 & 0 & ... & a_{\mathrm{nn}} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ . \\ b_n \end{pmatrix}$$

La solución se reduce a

$$\begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ . \\ \frac{b_n}{a_{\mathrm{nn}}} \end{pmatrix}$$

y la complejidad del problema se puede medir contando las n operaciones (divisiones) necesarias.

4.1.2. Sistemas triangulares superiores. Método ascendente

Si la matriz de coeficientes es triangular superior (**U «upper»**) y no singular, podemos resolver el sistema por el **método ascendente** :

$$\begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ 0 & a_{22} & ... & a_{2n} \\ . & . & ... & . \\ 0 & 0 & ... & a_{\mathrm{nn}} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ . \\ b_n \end{pmatrix}$$

comenzando por encontrar la última coordenada en la ecuación de abajo y **ascendiendo** por el sistema, calculando una nueva coordenada con cada ecuación.

- $x_n = \frac{b_n}{A_{n,n}}$
- $x_k = (b_k - \sum_{j=k+1}^{n}(A_{k,j} * x_j))/A_{k,k}$ para $k = n-1, n-2, ..., 1$

Ahora la complejidad del algoritmo se mide por el máximo de

$$1 + 3 + ... + (2k-1) + .. + (2n-1) = n^2$$

operaciones necesarias para encontrar la solución.

Algoritmo 4.1 **Método Ascendente.**

Datos de entrada: $A[n][n]$ (Matriz triangular sup. de coeficientes del sistema, sin ceros en la diagonal); $b[n]$ (vector término independiente.);
n (dimensión de A y b)
Variables: $x[n]$;// un vector donde escribir la solución.
Fujo del programa:

```
// Resolvemos el sistema por el método ascendente.
for(k=n;k>=1;k-){
  //x_k = (b_k - sum_{j=k+1}^{n}(A_{k,j} * x_j))/A_{k,k}
  x_k = b_k ;
  j = k + 1;
    for(j=k+1;j<=n;j++){
      x_k = x_k - A_{k,j} * x_j ;
      }
  x_k = x_k/A_{k,k} ;
 }
```

Datos de salida: Solución x del sistema.

4.1.3. Sistemas triangulares inferiores. Método descendente

Si la matriz de coeficientes es triangular inferior (**L «lower»**) y no singular, podemos resolver el sistema por el **método descendente** :

$$\begin{pmatrix} a_{11} & 0 & ... & 0 \\ a_{21} & a_{22} & ... & 0 \\ . & . & ... & . \\ a_{\mathrm{n1}} & a_{\mathrm{n2}} & ... & a_{\mathrm{nn}} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ . \\ b_n \end{pmatrix}$$

comenzando por encontrar primera coordenada en la ecuación de arriba y **descendiendo** por el sistema, calculando una nueva coordenada con cada ecuación.

- $x_1 = \frac{b_1}{A_{n,n}}$
- $x_k = (b_k - \sum_{j=1}^{k-1}(A_{k,j} * x_j))/A_{k,k}$ para $k = 2, 3, ..., n$

En este caso el máximo número de operaciones necesarias para encontrar la solución también es:

$$1 + 3 + ... + (2k - 1) + .. + (2n - 1) = n^2$$

Algoritmo 4.2 **Método Descendente.**

Datos de entrada: $A[n][n]$ (Matriz triangular inf. de coeficientes del sistema, sin ceros en la diagonal); $b[n]$ (vector término independiente.);
n (dimensión de A y b)
Variables: $x[n]$;// un vector donde escribir la solución.
Fujo del programa:

```
// Resolvemos el sistema por el método descendente.
for(k=1;k<=n;k++){
  //x_k = (b_k - sum_{j=k+1}^{n}(A_{k,j} * x_j))/A_{k,k}
  x_k = b_k ;
  j = k + 1;
    for(j=1;j<k;j++){
      x_k = x_k - A_{k,j} * x_j ;
      }
  x_k = x_k/A_{k,k} ;
  }
```

Datos de salida: Solución x del sistema.

4.2. Método de Gauss

Tal y como recordaréis del curso de "Álgebra lineal", el método de Gauss para resolver sistemas de ecuaciones

$$Ax = b$$

consiste en pasar a un sistema equivalente donde la matriz de coeficientes es triangular superior $Ux = v$ y resolver este nuevo sistema por el método ascendente.

La triangulación de la matriz de coeficientes se consigue mediante un proceso de eliminación, que partiendo de la matriz $\mathrm{A}=A^{(1)}=\left(a_{\mathrm{ij}}^{(1)}\right)_{i,j=1,...,n}$ va obteniendo matrices $A^{(k)}=\left(a_{\mathrm{ij}}^{(k)}\right)_{i,j=1,...,n}$

$$A^{(k)} = \begin{pmatrix} a_{11}^{(k)} & a_{12}^{(k)} & ... & a_{1k}^{(k)} & ... & a_{nk}^{(k)} \\ 0 & a_{22}^{(k)} & ... & a_{2k}^{(k)} & ... & a_{2n}^{(k)} \\ 0 & 0 & . & . & ... & . \\ & & & a_{kk}^{(k)} & ... & a_{kn}^{(k)} \\ \vdots & \vdots & . & . & ... & . \\ 0 & 0 & ... & a_{nk}^{(k)} & ... & a_{nn}^{(k)} \end{pmatrix}$$

Con el objetivo de obtener la matriz triangular superior U en el úúltimo paso $U = A^{(n)}$. Evidentemente, para que el sistema resultante sea equivalente al inicial debemos hacer en la columna de los términos independientes $b^{(k)}$ (con $b = b^1$) las mismas operaciones que en las matrices $A^{(k)}$ y quedarnos con $v = b^{(n)}$. Los pasos a seguir para obtener $A^{(k+1)}$ y $b^{(k+1)}$ partiendo de $A^{(k)}$ y $b^{(k)}$ son los siguientes:

(I) Búsqueda del **pivote**: $|a_{kp}^{(k)}| = \text{máx}\{|a_{kj}^{(k)}| : j = k, ..., n\}$, (el mayor elemento de la columna), con localización de la fila y p donde se alcanza. Observad que si el pivote es 0, la matriz de coeficientes es singular y el sistema no es compatible determinado.

(II) Cambio de fila: se permutan la fila k y p de $A^{(k)}$ y de $b^{(k)}$.

En la construcción del algoritmo se sustituye esta permutación por el uso de un puntero (permutación)

$$\sigma : \{1, 2, .., n\} \to \{1, 2, .., n\}$$

de manera que $\sigma(k) = p$ apunta a la fila p donde está el puntero sin necesidad de permutar las dos filas.

(III) Eliminación: Se utiliza que el pivote es no nulo para ir anulando los elementos de la columna k bajo la diagonal, restando a cada fila $i = k + 1, ..., n$ de $A^{(k)}$ y de b^k, la correspondiente fila k multiplicada por el cociente $\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$, dejando invariantes las k primeras filas.

$$A_i^{(k+1)} = A_i^{(k)} - \left(\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\right) * A_k^{(k)} \quad \text{y} \quad b_i^{(k+1)} = b_i^{(k)} - \left(\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\right) * b_k^{(k)}.$$

Algoritmo 4.3 **Método de Gauss con elección de pivote parcial**

Datos de entrada: $A[n][n]$ (Matriz de coeficientes del sistema.); $b[n]$ (vector término independiente.);
n (dimensión de A y b)
Variables: $B[n][n]$;// una matriz donde ir haciendo todas las modificaciones de A.
$v[n]$;// un vector donde ir haciendo las modificaciones de b.
$\sigma[n]$;// un vector "puntero" donde anotar las permutaciones de las filas.
// $\sigma[k]$ "apunta" hacia la fila que ocuparía la posición k
$x[n]$;// un vector donde escribir la solución.
Fujo del programa:

B=A ; v=b ;// Condiciones iniciales
`for(j=1;j<=n;j++){`
$\sigma(j) = j$;
`}`
// Vamos a hacer las etapas $k = 1, 2, ..., n-1$.
`for(k=1;k<n;k++){`
// 1. Elección del pivote (parcial).
p=k; //iniciamos la busqueda en la fila k
`for(i=k+1;i<=n;i++){`
`if(`$|B_{\sigma(i),k}| > |B_{\sigma(p),k}|$`){`
p=i;//apuntamos a la fila donde está el puntero
`}`
`}`
`if(`$B_{\sigma(p),k} == 0$`){`
Parada, Error: A es singular
`}`
// 2. Intercambio de filas (virtual). Cambios en el puntero.
$m = \sigma(k)$; $\sigma(k) = \sigma(p)$; $\sigma(p) = m$;
// 3. Eliminación.
`for(i=k+1;i<=n;i++){`
$mul = B_{\sigma(i),k}/B_{\sigma(k),k}$;
$B_{\sigma(i),k} = 0$;//Asignamos el valor 0 en lugar de hacer las operaciones
`for(j=k+1;j<=n;j++){`
$B_{\sigma(i),j} = B_{\sigma(i),j} - mul * B_{\sigma(k),j}$;
`}`
$v_{\sigma(i)} = v_{\sigma(i)} - mul * v_{\sigma(k)}$;
`}`
`}`
// Resolvemos el sistema por el método ascendente.
`for(k=n;k>=1;k-){`
//$x_k = (v_{\sigma(k)} - \sum_{j=k+1}^{n}(B_{\sigma(k),j} * x_j))/B_{\sigma(k),k}$
$x_k = v_{\sigma(k)}$;
$j = k + 1$;
`while(j<=n){`
$x_k = x_k - B_{\sigma(k),j} * x_j$;
j++ ;
`}`
$x_k = x_k/B_{\sigma(k),k}$;
`}`

Datos de salida: Solución x del sistema o mensaje de error si A es singular.

El número máximo de operaciones (sin contar las permutaciones de filas o el uso del puntero) para reducir el sistema a uno triangular superior dependiendo de la dimensión n del sistema en el método anterior $g(n)$ se puede calcular observando que

(I) $g(1) = 0$.

(II) $g(n) = (n-1) * (2 * n + 1) + g(n-1) = 2 * n^2 - n - 1 + g(n-1)$, teniendo en cuenta que para pasar de $A^{(1)}$ a $A^{(2)}$ y de $b^{(1)}$ a $b^{(2)}$, para cada fila $i = 2, ..., n$ de $A^{(1)}$ tenemos que construir el multiplicador de **(1 división)**, en cada uno de los $n-1$ elementos $j = 2, ...n$ de cada columna hay que hacer una multiplicación y una resta **($2 * (n-1)$ operaciones)** y en la misma fila de $b^{(1)}$ hay que hacer una multiplicación y una resta **(2 operaciones)**. El problema de reducir $A^{(2)}$ equivale a reducir una matriz de dimensión $n-1$.

(III) $g(n) = 2 * \sum_{i=1}^{n} n^2 - \sum_{i=1}^{n} n - n = \frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n$.

Tal y como hacíais en la asignatura de álgebra lineal, se puede modificar el algoritmo para calcular la matriz inversa de A. Aunque el cálculo de esta es equivalente a resolver los sistemas $Ax = e_i$ $(i = 1, ..., n)$ donde e_i es la base canónica de $\mathbb{K}^n$.

Otra variante que se puede realizar es el método de Gauss-Jordan, que consiste en realizar también el proceso de eliminación de los elementos que están sobre la diagonal, transformando el sistema de ecuaciones en uno equivalente donde la matriz de coeficientes va a ser diagonal.

Gauss con pivote parcial

El método de Gauss se puede optimizar al cambiar la estrategia seguida en la elección del pivote, buscando el elemento de mayor módulo

$$|a_{mp}^{(k)}| = \text{máx}\{|a_{ij}^{(k)}| : i = k, ..., n; j = k, ..., n\},$$

entre las filas y columnas $i \geq k$ y $j \geq k$. La finalidad es evitar hacer divisiones por números muy pequeños que pueden producir números muy grandes y cálculos inestables al operar con números de distinto tamaño. De esta manera construimos el algoritmo de Gauss de pivote total.

Como en el caso del pivote parcial, en lugar de permutar columnas en la matriz, lo que significaría permutar filas en el vector solución x, lo que se hace es utilizar un puntero donde señalar la posición de cada columna.

En el algoritmo 4.4 de la siguiente página se puede seguir la construcción.

Algoritmo 4.4 Método de Gauss con elección de pivote total

Datos de entrada: $A[n][n]$ (Matriz de coeficientes del sistema.); $b[n]$ (vector término independiente.);
n (dimensión de A y b)
Variables: $B[n][n]$;
$v[n]$;
fila$[n]$;// un vector "puntero" donde anotar las permutaciones de las filas.
colu$[n]$ // // un vector "puntero" donde anotar las permutaciones de las columnas.
$x[n]$;// un vector donde escribir la solución. **Fujo del programa:**

B=A ; v=b ;// Condiciones iniciales
for(j=1;j<=n;j++){
fila$(j) = j$; colu$(j) = j$
}
// Vamos a hacer las etapas $k = 1, 2, ..., n-1$.
for(k=1;k<n;k++){
// 1. Elección del pivote (total).
p=k; q=k //buscamos desde la fila k y la columna k
for(i=k;i<=n;i++){
for(j=k;j<=n;j++){
if($|B_{\text{fila}(i),\text{colu}(j)}| > |B_{\text{fila}(p),\text{colu}(q)}|$){
p=i; q=j;
}
}
}
if($B_{\text{fila}(p),\text{colu}(q)} == 0$){
Parada, Error: A es singular
}
// 2. Intercambio de filas y columnas. Cambios en los punteros.
$m = \text{fila}(k)$; fila$(k) = \text{fila}(p)$;fila$(k) = \text{fila}(p)$; fila$(p) = m$;
$m = \text{colu}(k)$; colu$(k) = \text{colu}(q)$;colu$(k) = \text{colu}(q)$; colu$(q) = m$;
// 3. Eliminación.
for(i=k+1;i<=n;i++){
$mul = B_{\text{fila}(i),\text{colu}(k)}/B_{\text{fila}(k),\text{colu}(k)}$;
$B_{\text{fila}(i),\text{colu}(k)} = 0$;//Asignamos el valor 0 en lugar de hacer las operaciones
for(j=k+1;j<=n;j++){
$B_{\text{fila}(i),\text{colu}(j)} = B_{\text{fila}(i),\text{colu}(j)} - mul * B_{\text{fila}(k),\text{colu}(j)}$;
}
$v_{\text{fila}(i)} = v_{\text{fila}(i)} - mul * v_{\text{fila}(k)}$;
}
}
// Resolvemos el sistema por el método ascendente.
for(k=n;k>=1;k-){//$x_{\text{colu}(k)} = (v_{\text{fila}(k)} - \sum_{j=k+1}^{n}(B_{\text{fila}(k),\text{colu}(j)} * x_{\text{colu}(j)}))/B_{\text{fila}(k),\text{colu}(k)}$
$x_{\text{colu}(k)} = v_{\text{fila}(k)}$;
$j = k+1$;
while(j<=n){
$x_{\text{colu}(k)} = x_{\text{colu}(k)} - B_{\text{fila}(k),\text{colu}(j)} * x_{\text{colu}(j)}$;
j++ ;
}
$x_{\text{colu}(k)} = x_{\text{colu}(k)}/B_{\text{fila}(k),\text{colu}(k)}$;
}

Datos de salida: Solución x del sistema o mensaje de error si A es singular.

4.3. Factorización LU

Supongamos que la matriz $A = (a_{\text{ij}})_{i,j=1,...,n}$ admite una factorización como el producto de una matriz triangular inferior L por una matriz triangular superior U, $A = LU$

$$\begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ . & . & ... & . \\ a_{11} & a_{12} & ... & a_{\text{nn}} \end{pmatrix} = \begin{pmatrix} l_{11} & 0 & ... & 0 \\ l_{21} & l_{22} & ... & 0 \\ . & . & ... & . \\ l_{11} & l_{12} & ... & l_{\text{nn}} \end{pmatrix} . \begin{pmatrix} u_{11} & u_{12} & ... & u_{1n} \\ 0 & u_{22} & ... & u_{2n} \\ . & . & ... & . \\ 0 & 0 & ... & u_{\text{nn}} \end{pmatrix}$$

Para resolver el sistema lineal de ecuaciones:

$$A.x = b \quad LUx = b$$

resolveriamos consecutivamente los sistemas :

$$Ly = b$$

y

$$Ux{=}y$$

con los métodos descendente y ascendente descritos en los apartados de arriba.

Cuando se puede hacer este tipo de factorizaciones, se dice que A tiene una factorización LU . Además, la factorización no es única, como veremos se puede asignar un valor distinto para cada l_{kk} y cada u_{kk}, aunque los productos l_{kk} u_{kk} si que permanecen constantes.

Para deducir la factorización LU comenzamos múltiplicando las matrices, obteniendo las ecuaciones

$$a_{i,j} = \sum_{s=1}^{n} l_{i,s} u_{s,j} = \sum_{s=1}^{\text{mín}(i,j)} l_{i,s} u_{s,j}$$

Con estas ecuaciones, por etapas, podemos ir determinando las filas de U y las columnas de L. Supongamos que tenemos determinados las (k-1) primeras filas de U y las (k-1) primeras columnas de L.
La ecuación correspondiente al término $a_{k,k}$:

$$a_{k,k} = \sum_{s=1}^{k-1} l_{k,s} u_{s,k} + l_{k,k} u_{k,k}$$

$$l_{k,k} u_{k,k} = a_{k,k} - \sum_{s=1}^{k-1} l_{k,s} u_{s,k}$$

El producto $p_k{=}\ l_{k,k} u_{k,k}$ está definido de forma única. Ahora podemos seguir distintos criterios para determinar los valores de $l_{k,k}$ y de $u_{k,k}$:

- Criterio de Dootlittle .- $l_{k,k}$ =1 $u_{k,k}$ =p_k
- Criterio de Crout .- $u_{k,k}$ =1 $l_{k,k}$ =p_k
- Método de Choleski .- L=U^t , es decir $l_{S,k}{=}u_{k,S}$. En particular $l_{k,k}{=}u_{k,k}{=}\sqrt{p_k}$

Una vez determinados los coeficientes $l_{k,k}$ y $u_{k,k}$, volvemos a las ecuaciones iniciales para escribir:
$a_{k,j}=\sum_{s=1}^{k-1} l_{k,s}u_{s,j}+l_{k,k}u_{k,j}$
$l_{k,k}u_{k,j}= a_{k,j}-\sum_{s=1}^{k-1} l_{k,s}u_{s,j}$
$a_{i,k}=\sum_{s=1}^{k-1} l_{i,s}u_{s,k}+l_{i,k}u_{k,k}$
$l_{i,k}u_{k,k}= a_{i,k}-\sum_{s=1}^{k-1} l_{i,s}u_{s,k}$
Si $p_k= l_{k,k}u_{k,k}\neq 0$, la fila k ($u_{k,j}$) de U y la columna k ($l_{i,k}$) de L están definidas de forma única.

Algoritmo 4.5 **Método de factorización LU (Dootlittle)**

Datos de entrada: $A[n][n]$ (Matriz de coeficientes del sistema.);
n (dimensión de A)
Variables: $L[n][n]$; $U[n][n]$// matrices para escribir las matrices triangulares superiores e inferiores.
aux;// una variable auxiliar para sumatorios y productos escalares.
Fujo del programa:

```
aux = 0. ;
for(k=0;k<n;k++){
  aux = A[k][k];
  for(s=0;s<k;s++){ // de Fila k de L por Columna k de U.
    aux = aux - L[k][s]U[s][k];
    }
  if(aux==0){
    Parada: no hay factorización LU;
    }
  L[k][k] = 1.;
  U[k][k] = aux;
  for(j=k+1;j<n;j++){// de Fila k de L por Columna j de U.
    aux = A[k][j];
    for(s=0;s<k;s++){ aux = aux - L[k][s] * U[s][j];
      }
    U[k][j] = aux;
    }
  for(i=k+1;i<n;i++){// de Fila i de L por Columna k de U.
    aux = A[i][k];
    for(s=0;s<k;s++){ aux = aux - L[i][s] * U[s][k];
      }
    L[i][k] = aux/U[k][k];
    }
  }
```

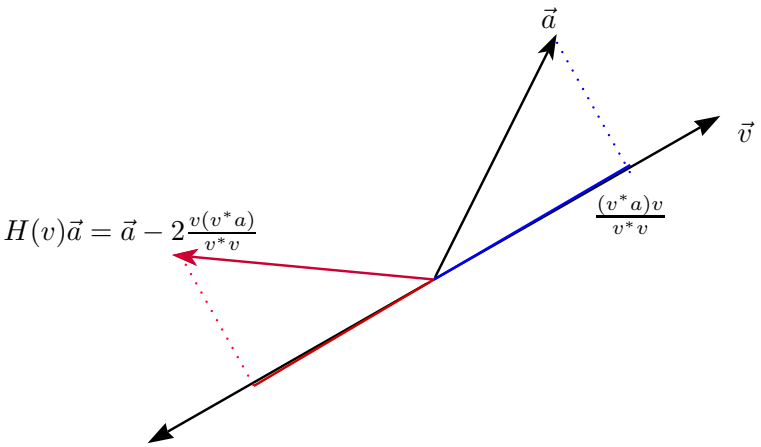
Datos de salida: L y U(Factorización LU)o mensaje de error

4.4. Factorización QR. Método de Householder

Se llaman matrices de Householder a las matrices de la forma

$$H(v) = Id - \frac{2}{v^*v}vv^*, \quad v \neq 0 \text{ un vector de } \mathbb{C}^n,$$

$$H(0) = Id.$$



Geométricamente, el producto $H(v)a$ representa a la "reflexión especular" de a con respecto a la dirección de v, es decir, el vector simétrico del vector a con respecto al vector normal a v en el plano determinado por los vectores a y v.

Ejercicio 4.4.1 Demuestra que las matrices $H(v)$ son unitarias y simétricas.

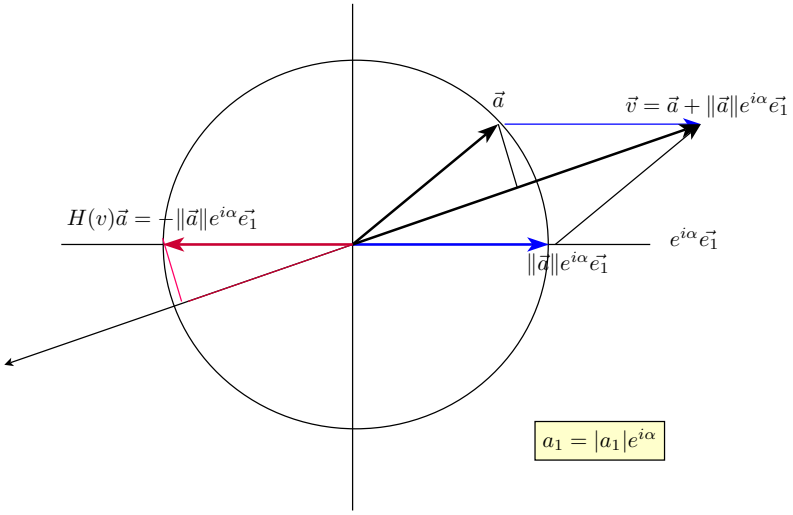
Además de por las propiedades expuestas en el ejercicio anterior, estas matrices son interesantes en análisis numérico por el siguiente teorema

Teorema 4.4.1 *Sea a un vector de $\mathbb{C}^n$. Entonces existen dos matrices de Householder H tales que Ha tiene todas sus coordenadas nulas salvo quizás la primera.*

De forma más precisa, si $a = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$, e_1 es el primer vector de la base canónica de $\mathbb{C}^n$, y $\alpha \in \mathbb{R}$ cumple $a_1 = |a_1|e^{i\alpha}$, entonces para $v = a \pm \|a\|_2 e^{i\alpha}e_1$ se cumple

$$H(a \pm \|a\|_2 e^{i\alpha}) = \mp\|a\|_2 e^{i\alpha}e_1.$$

El siguiente gráfico contiene una prueba geométrica del teorema:



DEMOSTRACIÓN:

Analíticamente, si tomamos $v = a + \|a\|e^{i\alpha}e_1$ y hacemos los correspondientes cálculos, tenemos:

- $v^*a = a^*a + \|a\|e^{-i\alpha}a_1 = \|a\|^2 + \|a\||a_1|$.
- $2(v^*a)v = (2\|a\|^2 + 2\|a\||a_1|)a + (2\|a\|^2 + 2|a_1|)\|a\|e^{i\alpha}e_1$.
- $v^*v = a^*a + \|a\|e^{i\alpha}\overline{a_1} + \|a\|e^{-i\alpha}a_1 + \|a\|^2 = 2\|a\|^2 + 2\|a\||a_1|$.
- $(v^*v)a - 2(v^*a)v = -(2\|a\|^2 + 2\|a\||a_1|)\|a\|e^{i\alpha}e_1$.
- $H(v)a = \frac{(v^*v)a - 2(v^*a)v}{v^*v} = -\|a\|e^{i\alpha}e_1$

□

Observación 4.4.2 Si $a = 0$, entonces $v = 0$ y $H = Id$.

En la practica, para calcular $H(v)b$ se sigue el siguiente proceso:

(I) se calcula la `norma` $\|v\|^2 = v^*v$,

(II) se calcula el `producto escalar` $(v, b) = v^*b$

(III) por último se calcula $H(v)b = b - (2(v, b)/\|v\|^2)v$.

La determinación de α puede hacerse tomando $signo = e^{i\alpha} = \frac{a_1}{\|a\|}$ si $a \neq 0$ $signo = 1$ si a=0. En el caso real $e^{i\alpha} = \pm 1$ es el signo de a_1. Con esta elección se evita el número v^*v del denominador sea demasiado pequeño y pueda producir inestabilidad en los cálculos.

El método de Householder para la resolución de un sistema $Ax = b$ consiste en encontrar $(n-1)$ matrices de Householder $H_1, ..., H_{n-1}$ de manera que $H_{n-1}...H_2H_1A$ sea una matriz triangular, y la solución del sistema es la solución de

$$H_{n-1}...H_2H_1Ax = H_{n-1}...H_2H_1b$$

que se obtiene por el método ascendente.

- Pongamos $A_0 = A$ y $H_0 = Id$.
- Pongamos $A_k = H_k H_{k-1} ... H_1 A$, y supongamos que presenta la forma:

$$A_k = \begin{pmatrix} x & x & x & x & x & x & x & x \\ & x & x & x & x & x & x & x \\ & & x & x & x & x & x & x \\ & & & x & x & x & x & x \\ & & & x & x & x & x & x \\ & & & x & x & x & x & x \\ & & & x & x & x & x & x \\ & & & x & x & x & x & x \end{pmatrix}$$

fila k (flecha a la derecha de la fila k)

$\vec{a_k}$ (flecha a la columna k)

columna k

- Sea $\widetilde{A}_k$ la matriz formada por los elementos a_{ij} de A que están en las filas $i \geq k$ y en las columna $j \geq k$. Sea $\vec{a_k} = \begin{pmatrix} a_{kk} \\ \vdots \\ a_{nk} \end{pmatrix} \in \mathbb{C}^{n-k+1}$, y $\widetilde{H}_{k+1} = H(\widetilde{v}_k)$ la matriz de Householder que da el teorema 4.4.1 de manera que $\widetilde{H}_{k+1}\vec{a_k} = \begin{pmatrix} 1 \\ 0 \\ \vdots \end{pmatrix}$.
- Sea

$$H_{k+1} = \left(\begin{array}{c|c} Id_{k-1} & 0 \\ \hline 0 & \widetilde{H}_{k+1} \end{array}\right),$$

H_k es la matriz de Householder $H(v_k)$ con $v_k = \begin{pmatrix} 0 \\ \vdots \\ \widetilde{v}_k \end{pmatrix}$.

- Construimos $A_{k+1} = H_{k+1} A_k$.

Con esta construcción $R = A_{n-1}$ es una matriz triangular.

En la construcción de A_{k+1} no se hacen modificaciones ni en las $k-1$ primeras filas, ni en las $k-1$ primeras columnas, sólo se modifican los elementos de $\widetilde{A}_k$.

Como las matrices de Householder son unitarias (conservan distancias)

$$cond_2(A) = cond_2(A_1) = ... = cond_n(A_{n-1})$$

En otras palabras. Al triangular la matriz con el método de Householder no varia el condicionamiento del problema.

Algoritmo 4.6 Método de Householder. Factorización QR

Datos de entrada: $A[n][n]$ (Matriz de coeficientes del sistema.); $b[n]$ (vector término independiente.);
n (dimensión de A y b)
Variables: $B[n][n]$;// una matriz donde ir haciendo las modificaciones de A.
$w[n]$;// un vector donde ir haciendo las modificaciones de b.
aux;// una variable auxiliar para sumatorios y productos escalares.
$sign$;// una variable para el signo $\overline{B_{k,k}}/|B_{k,k}|$.
$norma$;// una variable real para la norma del vector $\vec{a}$.
$norma2V$;// una variable real para $\|\vec{v}\|^2$.
$x[n]$;// un vector donde escribir la solución.
Fujo del programa:

B=A ; w=b ;// Condiciones iniciales
// Vamos a hacer las etapas $k = 1, 2, ..., n-1$.
`for(k=1;k<n;k++){`
// Vamos a hacer $\sum_{k+1}^{n} |B_{i,k}|$.
$aux = |B_{k+1,k}|$;
`for(i=k+2;1<=n;i++){` $aux = aux + |B_{i,k}|$; `}`
`if(`$aux == 0$`){ if(`$|B_{k,k}| == 0$`){ Error «Matriz Singular» Fin;}`
Continue; // Pasar a la siguiente etapa $k+1$ del bucle. `}`
`if(`$|B_{k,k}| > 0$`){` $signo = \overline{B_{k,k}}/|B_{k,k}|$`;} else{` $signo = 1$`;}`
$norma = |B_{k,k}|^2$; // Vamos a hacer $\sum_{k}^{n} |B_{i,k}|^2$.
`for(i=k+1;i<=n;i++){` $norma = norma + |B_{i,k}|^2$`;}`
$norma = \sqrt{norma}$; $v[k] = B_{k,k} + norma * signo$; // 1. vector de Householder.
`for(i=k+1;i<=n;i++){`$v[i] = B_{i,k}$`;}`
$norma2V = 2(norma)^2 + 2 * norma * signo * B_{k,k}$;
$B_{k,k} = -norma * signo$; // 2. Acción de la simetría en columna k.
`for(i=k+1;i<=n;i++){` $B_{i,k} = 0$; `}`
`for(j=k+1;j<=n;j++){` // Acción en las demás columnas.
$aux = \overline{v[k]} * B_{k,j}$; //$\vec{v}.\vec{B^j}$.
`for(i=k+1;i<=n;i++){` $aux = aux + \overline{v[i]} * B_{i,j}$; `}`
$aux = 2 * aux/norma2V$
`for(i=k;i<=n;i++){` $B_{i,j} = B_{i,j} - aux * v[i]$; `}`
`}`
$aux = \overline{v[k]} * w[k]$; // Acción en el vector independiente.
`for(i=k+1;i<=n;i++){` $aux = aux + \overline{v[i]} * w[i]$;`}` //$\vec{v}.\vec{w}$
$aux = 2 * aux/norma2V$
`for(i=k;i<=n;i++){` $w[i] = w[i] - aux * v[i]$;`}`
`}`
// 3 Resolvemos el sistema por el método ascendente.
`for(k=n;k>=1;k-){`//$x_k = (w_k - \sum_{j=k+1}^{n}(B_{k,j} * x_j))/B_{k,k}$
$x_k = v_k$; $j = k+1$;
`while(j<=n){` $x_k = x_k - B_{k,j} * x_j$; j++ ;
`}`
$x_k = x_k/B_{k,k}$;
`}`

Datos de salida: Solución x del sistema o mensaje de error si A es singular.

4.5. Tipos especiales de matrices

4.5.1. Matrices estrictamente diagonal dominante

Definición 4.5.1 *Se dice que una matriz cuadrada de dimensión n $A = (a_{ij})$ es* ***estrictamente diagonal dominante*** *cuando*

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

para toda fila $i = 1, ...n$.

Ejemplo 4.5.2 Si consideramos las matrices

$$A = \begin{pmatrix} 7 & 2 & 0 \\ 3 & 5 & 1 \\ 0 & 5 & 6 \end{pmatrix} \quad \text{y} \quad B = \begin{pmatrix} 5 & 3 & -3 \\ 3 & -4 & 0 \\ 3 & 0 & 4 \end{pmatrix}$$

La matriz A es estrictamente diagonal dominante, no simétrica y A^t no es estrictamente diagonal dominante. La matriz B es simétrica pero no es estrictamente diagonal dominante (tampoco lo es $B^t = B$.

No son raros los sistemas lineales estrictamente diagonal dominante que aparecen en muchos modelos de ecuaciones en diferencias (elementos finitos) al discretizar ecuaciones en derivadas parciales y en métodos numéricos como en el caso de los problemas de interpolación con "splines" que estudiaremos más adelante. Las matrices estrictamente diagonal dominante son no singulares y tienen buenas propiedades de estabilidad con relación al método de Gauss.

Teorema 4.5.3 *Toda matriz A estrictamente diagonal dominante es no singular. Además al realizar el proceso de eliminación del método de Gauss sin hacer intercambios de filas ni columnas tomando como pivote en cada etapa el elemento de la diagonal.*

DEMOSTRACIÓN:
Ideas que intervienen

- *Se puede razonar por contradicción: Si A fuese singular existiría $x = (x_1, .., x_n)^t$ tal que $Ax = 0$.*

  Tomamos k tal que $|x_k| = \|x\|_\infty = \max\{|x_1|, ...|x_n|\}$

  Como $\sum_j a_{ij}x_j = 0$ para todo i, en particular para $i = k$ se tiene

  $$a_{kk}x_k = -\sum_{j=1;j\neq k}^{n} a_{kj}x_j.$$

  La desigualdad triangular nos dice entonces que

  $$|a_{kk}||x_k| \leq \sum_{j=1;j\neq k}^{n} |a_{kj}||x_j|,$$

  Lo que contradice la hipótesis de que A es estrictamente diagonal dominante porque

  $$|a_{kk}| \leq \sum_{j=1;j\neq k}^{n} |a_{kj}|\frac{|x_j|}{|x_k|} \leq \sum_{j=1;j\neq k}^{n} |a_{kj}|.$$

- *Si A es estrictamente diagonal dominante, todos los elementos de su diagonal son no nulos. En particular $a_{11} \neq 0$ se puede elegir como pivote en la primera etapa del método de Gauss.*

  En este caso la matriz B que se obtiene tiene la misma primera fila que A y el resto de filas se definen haciendo la eliminación por

  $$b_{ij} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}$$

  La desigualdad triangular nos permite probar que

  $$\sum_{j=2; j\neq i}^{n} |b_{ij}| < |b_{ii}|.$$

  En otras palabras, la matriz $A_1 = B$ que proporciona la primera etapa del método de Gauss es diagonal estrictamente dominante.

  Repitiendo el proceso se tiene que se puede hacer el método de Gauss tomando como pivotes los elementos de la diagonal y la matriz triangular que se obtiene es estrictamente diagonal dominante. Observad que esta última matriz solo puede ser singular si alguno de los elementos de la diagonal es cero, pero en ese caso no sería estrictamente diagonal dominante.

□

Si la matriz A es estrictamente diagonal dominante (no singular) y no tiene ninguna fila "casi" nula, los cálculos del método de Gauss sin hacer cambios de filas ni columnas serán estables ya que los pivotes no resultan demasiado pequeños.

Los métodos iterativos de la siguiente lección para sistemas de ecuaciones con este tipo de matrices son convergentes.

4.5.2. Matrices simétricas definidas positivas.

Definición 4.5.4 *Se dice que una matriz simétrica ($A^t = A$) de dimensión n $A = (a_{ij})$ es* ***definida positiva*** *cuando*

$$x^t A x = \sum_{i,j=1}^{n} a_{ij} x_i x_j > 0$$

para todo vector columna $x \in \mathbb{R}^n$.

Proposición 4.5.5 *Si A es una matriz definida positiva entonces:*

(I) *A es no singular*

(II) *$a_{ii} > 0$ para cada $i = 1, ..., n$.*

(III) $\max\{|a_{ij}| : 1 \leq i, j \leq n\} \leq \max\{|a_{ii}| : 1 \leq i \leq n\}$.

(IV) $a_{ij}^2 < a_{ii} a_{jj}$.

DEMOSTRACIÓN: Ver el Teorema 6.21 del libro de Burden-Faires [2]. □

Una matriz simétrica A es definida positiva si y sólo si todos sus valores propios son estrictamente positivos. Equivalentemente, si y sólo si, todas las submatrices principales

$$A_k = \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \dots & a_{kk} \end{pmatrix}$$

tienen determinante estrictamente positivo.

Se puede probar que para matrices definidas positivas el método de Gauss se puede realizar sin hacer cambios de filas con cálculos estables (ver Teorema 6.21 del libro de Burden-Faires [2]).

Recordando la factorización LU se obtiene:

Teorema 4.5.6 *Sea A una matriz simétrica. Son equivalentes:*

(I) A es definida positiva.

(II) Existe una matriz triangular inferior B con diagonal estrictamente positiva, tal que $A = B^tB$. (Método de Choleski).

(III) Existe una matriz triangular inferior L con unos en la diagonal, y una matriz diagonal D con elementos estrictamente positivos a lo largo de la diagonal, tal que $A = L^tDL$.

DEMOSTRACIÓN: (1) ⇒ (2):

$$A = LU = \begin{pmatrix} 1 & & & & & \\ x & 1 & & & & \\ . & . & . & & & \\ . & . & & . & & \\ . & . & & & . & \\ x & x & . & . & . & 1 \end{pmatrix} \begin{pmatrix} u_{11} & x & . & . & . & x \\ & u_{22} & . & . & . & x \\ & & . & & & . \\ & & & . & & . \\ & & & & . & . \\ & & & & & u_{nn} \end{pmatrix}.$$

Si A es definida positiva, como $A_k = L_kU_k$, se tiene que todos los $u_{kk} > 0$.

Tomando la matriz diagonal $\Delta = diag(u_{ii})$, $B = L\Delta$ y $C = \Delta^{-1}U$ se tiene que $A = BC$. B es triangular inferior y C es triangular superior las diagonales de B y C coinciden con la de Δ.

Como A es simétrica, $A = BC = C^tB^t$ y $C(B^t)^{-1} = B^{-1}C^t$.

Por una parte $C(B^t)^{-1}$ es triangular superior y tiene sólo unos en su diagonal. Y también, $B^{-1}C^t$ es triangular inferior con sólo unos en la diagonal. Así la igualdad de las dos matrices nos dice que ambas coinciden con la identidad y por lo tanto

$$C = B^t$$

(2) ⇒ (3): Si $A = BB^t$, tomamos $\Delta = diag(B_{ii})$, entonces

$$A = B\Delta^{-1}\Delta\Delta\Delta^{-1}B^t = LDL^t.$$

Observad que L es triangular inferior con unos en la diagonal y $D = \Delta\Delta$ es diagonal con elementos D_{ii} estrictamente positivos. (3) ⇒ (1):

$$x^tAx = x^tLDL^tx = (L^tx)^tD(L^tx) = \sum_i D_{ii}y_i^2 > 0$$

donde $y = L^t x \neq 0$ si $x \neq 0$.

(Ver el teorema 4.4.1 de la sección 4.4 de [3].)

□

Algoritmo 4.7 **Método de factorización de (Choleski)**

(sólo para matrices simétricas definidas positivas)
Datos de entrada: $A[n][n]$ (Matriz de coeficientes del sistema.);
n (dimensión de A)
Variables: $L[n][n]$; // matriz para escribir la matriz triangular superior.
aux;// una variable auxiliar para sumatorios y productos escalares.
Fujo del programa:
$aux = 0.$;
`for(k=0;k<n;k++){`
$aux = A[k][k]$;
`for(s=0;s<k;s++){` // de Fila k de L por Columna k de L^t.
$aux = aux - L[k][s] * L[k][s]$;
}
`if(aux<=0){`
Parada: no hay factorización de Choleski;
}
$L[k][k] = \sqrt{aux}$;
`for(i=k+1;i<n;i++){`// de Fila i de L por Columna k de L^t.
$aux = A[i][k]$;
`for(s=0;s<k;s++){` $aux = aux - L[i][s] * L[k][s]$;
}
$L[i][k] = aux/L[k][k]$;
}
}
Datos de salida:L (Factorización de Choleski $A = LL^t$)o mensaje de error

4.5.3. Matrices tridiagonales

Definición 4.5.7 *Una matriz cuadrada A de dimensión n se dice que es una* ***matriz banda*** *cuando existen enteros p y q tales que $a_{ij} = 0$ si $i + p \leq j$ o $j + q \leq i$. El ancho de banda de este tipo se define como $w = p + q - 1$.*

¿ Cuáles son las matrices banda con $p = 1$ y $q = 1$ ($w = 1$)?

Las matrices banda que más suelen aparecer en la práctica tienen la forma $p = q = 2$ y $p = q = 4$. Las matrices de ancho de banda 3 con $p = q = 2$ se llaman **matrices tridiagonales** porque su forma es

Teorema 4.5.8 *Si A es una matriz tridiagonal*

$$A = \begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix}$$

se define la sucesión $\delta_0 = 1$, $\delta_1 = 1$, $\delta_k = b_k\delta_{k-1} - a_k c_{k-1}\delta_{k-2}$ $(2 \leq k \leq n)$.

Entonces, $\delta_k = det(A_k)$ (A_k el menor principal de orden k) y si todos los $\delta_k \neq 0$, la factorización LU de la matriz A es

$$A = LU = \begin{pmatrix} 1 & & & & \\ a_2\frac{\delta_0}{\delta_1} & 1 & & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1}\frac{\delta_{n-3}}{\delta_{n-2}} & 1 & \\ & & & a_n\frac{\delta_{n-2}}{\delta_{n-1}} & 1 \end{pmatrix} \begin{pmatrix} \frac{\delta_1}{\delta_0} & c_1 & & & \\ & \frac{\delta_2}{\delta_1} & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & \frac{\delta_{n-1}}{\delta_{n-2}} & c_{n-1} \\ & & & & \frac{\delta_n}{\delta_{n-1}} \end{pmatrix}$$

DEMOSTRACIÓN: Ver Teorema 4.3.2 de [3] □

Ejemplo 4.5.9 Si A es una matriz tridiagonal simétrica

$$A = \begin{pmatrix} b_1 & a_2 & & & \\ a_2 & b_2 & a_3 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1} & b_{n-1} & a_n \\ & & & a_n & b_n \end{pmatrix}$$

y todos los determinantes de los menores principales son positivos, Escribe una algoritmo que proporcione la factorización de Choleski $A = SS^t$

En la sección 4.3 del libro de Ciarlet [3] y en la sección 6.6 del libro de Burden-Faires [2] podéis estudiar la factorización de LU para sistemas tridiagonales. En esa misma sección tenéis información sobre resolución de sistemas con matrices de coeficientes de los distintos tipos que acabamos de presentar.

4.6. Actividades complementarias del capítulo

Los documentos se pueden descargar de la Zona Compartida de SUMA.

- Hoja de problemas nº4

Bibliografía

[1] A. Delshams A. Aubanell, A. Benseny, *Útiles básicos de cálculo numérico*, Ed. Labor - Univ. Aut. de Barcelona, Barcelona, 1993.

[2] R.L. Burden and J.D. Faires, *Análisis numérico, 7^a edición*, Thomson-Learning, Mexico, 2002.

[3] P.G. Ciarlet, *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson, Paris, 1990.

[4] D. Kincaid and W. Cheney, *Análisis numérico*, Ed. Addison-Wesley Iberoamericana, Reading, USA, 1994.

Capítulo 5

Métodos iterativos de resolución de sistemas de ecuaciones

Interrogantes centrales del capítulo

- Análizar técnicas iterativas de resolución de sistemas de ecuaciones lineales.
- Aprender los métodos de resolución :
  - Método de Jacobi.
  - Método de Gauss-Seidel.
  - Método de relajación.

Destrezas a adquirir en el capítulo

- Resolver sistemas de ecuaciones lineales utilizando los métodos iterativos.
- Implementar los métodos en el ordenador.
- Compararlos entre si y con los métodos directos del capítulo anterior.

En esta unidad se estudian distintos métodos iterativos de resolución de sistemas de ecuaciones lineales. Los métodos iterativos no suelen utilizarse para resolver problemas lineales de dimensión pequeña ya que, para obtener una precisión razonable, requieren más operaciones que los métodos directos. Sin embargo, en el caso de sistemas grandes con muchos ceros en sus coeficientes (matrices banda o estrictamente diagonal dominantes que aparecen en problemas de ecuaciones diferenciales con condiciones frontera) hay métodos iterativos muy eficientes.

Proponemos tres métodos iterativos concretos a partir de una misma idea general.

5.1. Métodos iterativos. Criterios de Convergencia

Idea general:

Un método iterativo para resolver un sistema lineal

$$Ax = b$$

consiste en transformar el sistema de ecuaciones en una ecuación de punto fijo

$$x = Tx + c;$$

donde T es una aplicación lineal. Si el radio espectral $\rho(T) < 1$, la aplicación $Tx+c$ es contractiva y la solución del sistema $\vec{x}$ (el punto fijo) se obtiene como el límite de una sucesión de iteradas funcionales $\vec{x}_k = T\vec{x}_{k-1} + c$, comenzando en una aproximación inicial $\vec{x}_0$ a la solución.

Ejemplo 5.1.1 Consideremos el sistema $Ax = b$ dado por las ecuaciones:

$$\begin{cases} 2x_1 - 2x_2 & = 1 \\ 2x_1 + 3x_2 + x_3 & = 5 \\ -x_1 - 2x_3 & = 7 \end{cases}$$

que tiene como única solución $x = (\frac{20}{9}, \frac{31}{18}, -\frac{83}{18}) \approx (2.22222, 1.72222, -4.61111)$.

Despejando x_i en la ecuación i se tiene la ecuación equivalente

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

Ésta es una ecuación de punto fijo $x = Tx + c$, con

$$T = \begin{pmatrix} 0 & 1 & 0 \\ -\frac{2}{3} & 0 & -\frac{1}{3} \\ -\frac{1}{2} & 0 & 0 \end{pmatrix}$$

$\rho(T) = 0.84657...$, El límite de la sucesión de iteradas $\vec{x}_k = T\vec{x}_{k-1}+c$ obtenido en 194 iteraciones comenzando en $\vec{x}_0 = 1$, con un error relativo en la imagen menor que 10^{-14}, es la solución del sistema lineal $(2.22222, 1.72222, -4.61111)$

Definición 5.1.2 *Dado un sistema lineal $Ax = b$, se llama "método iterativo de resolución del sistema" a cualquier par (T, c) formado por una matriz T y un vector c tales que la solución de $Ax = b$ es el único punto fijo de la función afín $\Phi(x) = Tx + c$, es decir*

$$Ax = b \quad \Rightarrow \quad x = Tx + c.$$

Se dice que el método iterativo es convergente cuando la sucesión de iteradas $x_k = \Phi(x_{k-1}) = Tx_{k-1} + c$ converge hacia el punto fijo para cualquier elección del vector x_0.

A la hora de implementar los métodos iterativos interesa tener presente que podemos utilizar como condición de parada el tamaño de los "vectores residuales" en cada etapa:

$$r_k = Ax_k - b.$$

5.1.1. Criterios de Convergencia

En el siguiente teorema recogemos los resultados estudiados en el capítulo 3 que dan condiciones necesarias y suficientes para que un método iterativo sea convergente:

Teorema 5.1.3 *Sea T una matriz cuadrada de dimensión n. Entonces son equivalentes:*

(I) Existe una norma matricial (subordinada) tal que $\|T\| < 1$.

(II) El radio espectral $\rho(T) < 1$.

(III) $\lim_{k\to\infty} T^k v = 0$, para todo vector v.

(IV) Las sucesiones de iteradas $\vec{x}_k = T\vec{x}_{k-1} + c$ converge comenzando en cualquier vector $\vec{x}_0$.

DEMOSTRACIÓN:

La equivalencia entre (I) y (II) es el Teorema 3.3.2.

La equivalencia entre (II) y (III) es el Teorema 3.3.4.

La implicación (I) $\Rightarrow$ (IV), la da el teorema del punto fijo porque si $\Phi(x) = Tx + c$,

$$\|\Phi(x) - \Phi(y)\| = \|Tx - Ty\| = \|T(x-y)\| \leq \|T\|\|x-y\|,$$

y si $\|T\| < 1$, Φ será contractiva.

Para la implicación (IV) $\Rightarrow$ (III), consideremos v un vector arbitrario, y x el vector solución de $x = Tx + c$. Si tomamos $\vec{x}_0 = x - v$, la sucesión $\vec{x}_k = T\vec{x}_{k-1} + c$ converge hacia x y por lo tanto:

$$x - \vec{x}_k = Tx + c - (T\vec{x}_{k-1} + c) = T(x - \vec{x}_{k-1}) = T^2(x - \vec{x}_{k-2}) \ldots = T^k(x - \vec{x}_0) = T^k(v) \to 0.$$

□

Ejemplo 5.1.4 Volviendo al ejemplo 5.1.1, el polinomio característico de la matriz T es

$$p_T(\lambda) = 6\lambda^3 + 4\lambda + 1.$$

El radio espectral es $\rho(T) = 0.84657\ldots$[1] por lo tanto la iteración $\vec{x}_k = T\vec{x}_{k-1} + c$, comenzando en cualquier vector $\vec{x_0}$, converge hacia la solución del sistema de punto fijo $x = Tx + c$; que también es la solución de la ecuación $Ax = b$.

5.1.2. Construcción de Métodos iterativos

Idea general:

Supongamos que tenemos un sistema lineal

$$Ax = b$$

[1] Aunque no es facil calcular raíces de polinomios de tercer grado, si es fácil comprobar que los ceros del polinomio característico (valores propios de T) tienen módulo menor que $\sqrt[3]{\frac{5}{6}} < 1$

y que la matriz A se puede expresar como diferencia de dos matrices

$$A = M - N,$$

donde M es una matriz " fácil " de invertir (por ejemplo si es diagonal o triangular). Entonces:

$$Ax = b \longleftrightarrow Mx - Nx = b \longleftrightarrow Mx = Nx + b,$$

$$Ax = b \longleftrightarrow x = M^{-1}Nx + M^{-1}b$$

Dada una matriz cuadrada de dimensión n, $A = (a_{ij})$, tal que $a_{ii} \neq 0$ para todo $1 \leq i \leq n$. Entonces la matriz diagonal $D = (a_{ii})$ es muy fácil de invertir. Escribiendo $M = D$ y $N = D - A$ se obtiene el método iterativo de Jacobi.

Los métodos de Gauss-Seidel y de relajación son variaciones del método de Jacobi. Para descibirlos vamos a utilizar la siguiente notación para describir $N = D - A = -(L + U)$:

$$L = \begin{pmatrix} 0 & 0 & & & & 0 \\ a_{21} & 0 & & & & 0 \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & 0 & . \\ a_{n1} & . & . & . & a_{n(n-1)} & 0 \end{pmatrix} \quad U = \begin{pmatrix} 0 & a_{12} & . & . & . & a_{1n} \\ & 0 & a_{23} & . & . & a_{2n} \\ & & . & & & . \\ & & & . & & . \\ & & & & 0 & a_{(n-1)n} \\ & & & & & 0 \end{pmatrix}.$$

5.2. Método de Jacobi

Tal y como hemos mencionado, el método de Jacobi para buscar la solución de un sistema lineal $Ax = b$ si no hay ceros en la diagonal D de A consiste en construir la sucesión de iteradas

$$x_{k+1} = D^{-1}(-(L + U))x_k + D^{-1}b = A_J x_k + D^{-1}b.$$

Para realizar los cálculos de forma eficiente y para utilizarlos en las condiciones de parada comenzamos analizando los vectores residuales y observando cómo pueden utilizarse para construir cada iteración del método:

- $r_k = Ax_k - b = Dx_k + (L + U)x_k - b$
- $D^{-1}r_k = x_k - (D^{-1}(-(L + U))x_k + D^{-1}b) = x_k - x_{k+1}$
- $x_{k+1} = x_k - D^{-1}r_k$

Cada etapa del cálculo de los vectores $r_k = (r_i^k)_i$ y $x_k = (x_i^k)_i$ se realiza coordenada a coordenada, comezando en $i = 1$ y consecutivamente $i = 2, 3, ..., n$

`Para` $i = 1$, `mientras que` $i \leq n$, `haciendo en cada paso` $i = i + 1$:

$$r_i^k = a_{ii}x_i^k + \sum_{j=1;j\neq i} a_{ij}x_j^k - b_i \tag{5.1}$$

$$x_i^{k+1} = x_i^k - \frac{1}{a_{ii}}r_i^k$$

El siguiente algoritmo implementa el método de Jacobi. Prestad atención a la construcción del vector residual y como se utiliza el cuadrado de su norma euclídea como condición de parada:

Algoritmo 5.1 **Método de Jacobi para resolución de sistemas lineales**

Datos de entrada: $A[n][n]$ (Matriz de coeficientes del sistema.);
$b[n]$ (vector término independiente.);
n (dimensión de A y b);
ε (precisión para la condición de parada);
$nmax$ (número máximo de iteraciones);
Variables:$xa[n]$;// (x_k) vector para aproximar la solución del sistema.
$e[n]$;// un vector auxiliar para almacenar el vector residual y el vector de corrección $x_k - 1 - x_k$.
$eadmisible = 0$;// precisión admisible (se usan errores relativos.
$norma = 0$; // registro para el cuadrado de la norma del vector residual.
Fujo del programa:
// Condiciones iniciales y evaluacion de la diagonal
`for(j=1;j<=n;j++){`
$xa(j) = 1$; $eadmisible = eadmisible + b(j)^2$;
`if(`$A_{j,j} == 0$`){`
ERROR; Jacobi no es aplicable;
`}`
`}`
$eadmisible = \varepsilon^2 * eadmisible$ // $(\varepsilon * \|b\|)^2$.
// Vamos a hacer las etapas $k = 1, 2, ..., nmax$.
`for(k=1;k<=nmax;k++){`
$norma = 0$; // 1. cálculo de la corrección.
`for(i=1;i<=n;i++){`
$e(i) = -b(i)$;
`for(j=1;j<=n;j++){`
$e(i) = e(i) + A_{i,j} * xa(j)$;
`}`
$norma = norma + e(i)^2$;
$e(i) = e(i)/A_{i,i}$;
$xa(i) = xa(i) - e(i)$;
`}`
`if(`$norma < eadmisible$`){`
Parada, la solucion es xa
`}`
`}`
Parada, no hay convergencia en nmax iteraciones;
Datos de salida: Solución x del sistema o mensajes de error si la diagonal de A tiene algún cero o la iteración no converge.

Ejemplo 5.2.1 En el ejemplo 5.1.1 se ha considerado la iteración de Jacobi que, como hemos señalado, converge hacia la solución del sistema lineal $Ax = b$.

5.3. Método de Gauss-Seidel

Los vectores del algoritmo de Jacobi se van construyendo coordenada a coordenada de manera que cuando se va a calcular x_i^{k+1} ya se conocen los valores de x_j^{k+1} para $j < i$. La idea en la modificación propuesta en el método de Gauss-Seidel consiste en utilizar para el cálculo de x_i^{k+1} en 5.1 las coordenada conocidas x_j^{k+1} para $j < i$ junto con x_j^k para $i \leq j$, en lugar de utilizar sólo las coordenadas de x_k, en la hipótesis de que si el método va a converger, las coordenadas de x_{k+1} son una mejor aproximación a las de la solución que las de x_k.

`Para` $i = 1$, `mientras que` $i \leq n$, `haciendo en cada paso` $i = i + 1$:

$$\tilde{r}_i^k = a_{ii}x_i^k + \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} + \sum_{j=i+1}^{n} a_{ij}x_j^k - b_i \tag{5.2}$$

$$x_i^{k+1} = x_i^k - \frac{1}{a_{ii}}\tilde{r}_i^k$$

En términos matriciales, el método de Gauss-Seidel consiste en considerar la descomposición $A = (L + D) + U = M - N$, $M = L + D$ y $N = -U$. Para comprobarlo basta con observar las coordenadas de la expresión: $(L + D)x_{k+1} = -Ux_k + b$. La matriz del método iterativo de Gauss-Seidel es $A_G = (L + D)^{-1}(-U)$.

En la página siguiente está el algoritmo 5.2 correspondiente a este método. Observad que para la condición de parada hemos utilizado como vector residual el vector $\tilde{r}_k$, donde cada una de las coordenadas $\tilde{r}_i^k$ se corresponde con la coordenada i de los vectores $A\tilde{x}_{ik} - b$ con $\tilde{x}_{ik} = (x_1^{k+1}, ..., x_{i-1}^{k+1}, x_i^k, ...x_n^k)^t$ $(i = 1, ..., n)$. Si la sucesión x_k converge hacia la solución del sistema lineal, también lo hace $\tilde{x}_{ik}$ y $\tilde{r_k} \to 0$.

5.3.1. Convergencia de los Métodos de Jacobi y Gauss-Seidel

Para las matrices especiales del capítulo anterior se tienen buenos criterios de convergencia:

Teorema 5.3.1 *Sea A una matriz diagonal estrictamente dominante:*

$$\sum_{j=1, j\neq i} |a_{ij}| < |a_{ii}|$$

para todo $i = 1, ..., n$. Entonces el método de Jacobi y el método de Gauss-Seidel para resolver el sistema $Ax = b$ son convergentes, y el método de Gauss-Seidel converge al menos a la misma velocidad que el de Jacobi. De forma más concreta

$$\|A_G\|_\infty \leq \|A_J\|_\infty < 1.$$

La demostración de este teorema la podéis seguir en la sección 3 del capítulo 8 libro de Hämmerlin y Hoffman [4].

Algoritmo 5.2 **Método de Gauss-Seidel para resolución de sistemas lineales**

Datos de entrada: $A[n][n]$ (Matriz de coeficientes del sistema.);
$b[n]$ (vector término independiente.);
n (dimensión de A y b);
ε (precisión para la condición de parada);
$nmax$ (número máximo de iteraciones);
Variables:$xa[n]$;// vector para aproximar la solución del sistema.
$xb[n]$;// vector para las nuevas aproximaciones de la solución del sistema.
$\tilde{e}[n]$;// un vector residual modificado.
$eadmisible = 0$;// precisión admisible.
$norma = 0$; // registro para el cuadrado de la norma del vector residual modificado.
Fujo del programa:
// Condiciones iniciales y evaluacion de la diagonal
`for(j=1;j<=n;j++){`
$xa(j) = 1$; $eadmisible = eadmisible + b(j)^2$;
`if(`$A_{j,j} == 0$`){` ERROR; Gauss-Seidel no es aplicable;`}`
`}`
$eadmisible = \varepsilon^2 * eadmisible$ // $(\varepsilon * \|b\|)^2$.
// Vamos a hacer las etapas $k = 1, 2, ..., nmax$.
`for(k=1;k<=nmax;k++){`
$norma = 0$; // 1. cálculo del residuo.
`for(i=1;i<=n;i++){`
$\tilde{e}(i) = -b(i)$;
`for(j=i;j<=n;j++){`
$\tilde{e}(i) = \tilde{e}(i) + A_{i,j} * xa(j)$; // se usan coordenadas de x_k.
`}`

`for(j=1;j<i;j++){`
$\tilde{e}(i) = \tilde{e}(i) + A_{i,j} * xb(j)$// se usan coordenadas de x_{k+1}.
`}`
$norma = norma + \tilde{e}(i)^2$;
$\tilde{e}(i) = \tilde{e}(i)/A_{i,i}$;
$xb(i) = xa(i) - \tilde{e}(i)$;
`}`
`if(`$norma < eadmisible$`){` Parada, la solucion es xa`}`
$xa = xb$
`}`
Parada, no hay convergencia en nmax iteraciones;
Datos de salida: Solución x del sistema o mensajes de error si la diagonal de A tiene algún cero o la iteración no converge.

Teorema 5.3.2 *Sea A una matriz tridiagonal. Entonces los radios espectrales de las matrices de los métodos de Jacobi y Gauss-Seidel cumplen:*

$$\rho(A_G) = \rho(A_J)^2$$

.

Así los métodos de Jacobi y de Gauss-Seidel para resolver el sistema $Ax = b$ convergen simultáneamente y cuando lo hacen, el método de Gauss-Seidel converge más rápidamente que el de Jacobi.

La demostración de este teorema la podéis seguir en el libro de Ciarlet [3, The 5.3-4].

5.4. Método de Relajación

En la construcción de la sucesión de Gauss-Seidel hemos ido definiendo x_{k+1} coordenada a coordenada conjuntamente con las coordenadas del vector residual $\tilde{r}_k$:

`Para` $i = 1$, `mientras que` $i \leq n$, `haciendo en cada paso` $i = i + 1$:

$$\tilde{r}_i^k = a_{ii}x_i^k + \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} + \sum_{j=i+1}^{n} a_{ij}x_j^k - b_i \qquad (5.2)$$

$$x_i^{k+1} = x_i^k - \frac{1}{a_{ii}}\tilde{r}_i^k$$

Los métodos de relajación consisten en considerar un peso $\omega > 0$ para corregir las coordenadas de x_k poniendo en la ecuación 5.2

$$x_i^{k+1} = x_i^k - \frac{\omega}{a_{ii}}\tilde{r}_i^k$$

Observad que ahora

$$a_{ii}x_i^{k+1} = a_{ii}x_i^k - \omega(a_{ii}x_i^k + \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} + \sum_{j=i+1}^{n} a_{ij}x_j^k - b_i)$$

$$a_{ii}x_i^{k+1} + \omega\sum_{j=1}^{i-1} a_{ij}x_j^{k+1} = (1-\omega)a_{ii}x_i^k - \omega(\sum_{j=i+1}^{n} a_{ij}x_j^k - b_i)$$

Con esta modificación, en términos matriciales, el método de relajación consiste en considerar

$$(D + \omega L)x_{k+1} = ((1-\omega)D - \omega U)x_k + \omega b.$$

Así, la matriz de la iteración del método de relajación es

$$A_{R(\omega)} = (D + \omega L)^{-1}((1-\omega)D - \omega U).$$

En la página siguiente aparece el algoritmo de este método iterativo.

Algoritmo 5.3 **Método de Relajación para resolución de sistemas lineales**

Datos de entrada: $A[n][n]$ (Matriz de coeficientes del sistema.);
$b[n]$ (vector término independiente.);
n (dimensión de A y b);
ω (parámetro de construcción);
ε (precisión para la condición de parada);
$nmax$ (número máximo de iteraciones);
Variables:$xa[n]$;// vector para aproximar la solución del sistema.
$xb[n]$;// vector para las nuevas aproximaciones de la solución del sistema.
$\tilde{e}[n]$;// un vector de corrección para xa.
$eadmisible = 0$;// precisión admisible.
$norma = 0$; // registro para el cuadrado de la norma de la corrección.
Fujo del programa:
// Condiciones iniciales y evaluacion de la diagonal
`for(j=1;j<=n;j++){`
$xa(j) = 1$; $eadmisible = eadmisible + b(j)^2$;
`if(`$A_{i,i} == 0$`){` ERROR; Relajación no es aplicable;`}`
`}`
$eadmisible = \varepsilon^2 * eadmisible$ // $(\varepsilon * \|b\|)^2$.
// Vamos a hacer las etapas $k = 1, 2, ..., nmax$.
`for(k=1;k<=nmax;k++){`
$norma = 0$; // 1. cálculo de la corrección.
`for(i=1;i<=n;i++){`
$\tilde{e}(i) = -b(i)$;
`for(j=i;j<=n;j++){`
$\tilde{e}(i) = \tilde{e}(i) + A_{i,j} * xa(j)$; `}`
`for(j=1;j<i;j++){`
$\tilde{e}(i) = \tilde{e}(i) + A_{i,j} * xb(j)$; `}`
$norma = norma + \tilde{e}(i)^2$;
$\tilde{e}(i) = \tilde{e}(i) * \omega / A_{i,i}$;
$xb(i) = xa(i) - \tilde{e}(i)$;
`}`
`if(`$norma < eadmisible$`){` Parada, la solucion es xa`}`
$xa = xb$
`}`
Parada, no hay convergencia en nmax iteraciones;
Datos de salida: Solución x del sistema o mensajes de error si la diagonal de A tiene algún cero o la iteración no converge.

Ejemplo 5.4.1 En el caso concreto del ejemplo 5.1.1 el método de Gauss-Seidel no rebaja el número de iteraciones utilizadas por el de Jacobi para aproximar la solución de $Ax = b$ con la precisión de 10^{-14}, sin embargo el método de relajación con $w = 0.85$ si que las rebaja significativamente pues alcanza la solución en sólo 34 iteraciones.

En relación con la convergencia de los métodos de relajación tenemos el siguiente resultado:

Teorema 5.4.2 *Si A es una matriz simétrica definida positiva, el método de relajación converge para $0 < \omega < 2$.*

En general el radio espectral de la matriz del método de relajación $A_{R(\omega)}$ cumple siempre que $\rho(R(\omega)) > |\omega - 1|$. Por lo tanto el método de relajación sólo puede ser convergente si $0 < \omega < 2$.

Si A es una matriz simétrica definida positiva, tridiagonal, los métodos de Jacobi, Gauss-Seidel y Relajación para $0 < \omega < 2$ son convergentes y el mínimo de los radios espectrales de las matrices de los métodos de Relajación se alcanza en

$$\omega_0 = \frac{2}{1 + \sqrt{1 - \rho(A_J)^2}},$$

de manera que

$$\rho(A_{R(\omega_0)}) = \min_{0<\omega<2}\{\rho(A_{R(\omega)})\} < \rho(A_G) = \rho(A_J)^2 < \rho(A_J).$$

El parámetro ω_0 nos da el método iterativo óptimo, con convergencia más rápida, de entre los estudiados.

La demostración de este teorema la podéis seguir en la sección 5.3 del libro de Ciarlet [3].

Bibliografía

[1] A. Delshams A. Aubanell, A. Benseny, *Útiles básicos de cálculo numérico*, Ed. Labor - Univ. Aut. de Barcelona, Barcelona, 1993.

[2] R.L. Burden and J.D. Faires, *Análisis numérico, 7ª edición*, Thomson-Learning, Mexico, 2002.

[3] P.G. Ciarlet, *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson, Paris, 1990.

[4] G. Hammerlin and K.H. Hoffmann, *Numerical mathematics*, Springer-Verlag, New York, 1991.

[5] D. Kincaid and W. Cheney, *Análisis numérico*, Ed. Addison-Wesley Iberoamericana, Reading, USA, 1994.

Capítulo 6

Valores y vectores propios

Interrogantes centrales del capítulo

- Generalidades sobre valores y vectores propios.
- El método de la potencia. El método de deflación de Wielandt.
- El método de Jacobi.

Destrezas a adquirir en el capítulo

- Localizar aproximadamente los valores propios de una matriz a partir de sus coeficientes.
- Describir los algoritmos correspondientes a distintos métodos de aproximación de valores y vectores propios.
- Comparar la convergencia de las aproximaciones proporcionadas por cada método.
- Implementar en el ordenador los programas de los métodos de este capítulo.
- Aplicar los programas construidos de forma efectiva en la búsqueda de todos los valores y vectores propios de matrices concretas.

Desarrollo de los contenidos fundamentales

Es este capítulo abordamos el problema de calcular, o más bien aproximar, los valores y vectores propios de una matriz, con especial atención al caso de las matrices simétricas.

En la primera sección indicamos brevemente los problemas de condicionamiento que pueden presentarse y formulamos el teorema de los círculos de Gerschgorin, que proporciona una primera aproximación (bastante burda) al problema de localizar los valores propios de una matriz.

A continuación presentamos el método de la potencia, que es una manera muy sencilla de localizar el valor propio de mayor valor absoluto de una matriz (cuando exista) y el correspondiente vector propio, y vemos como el método puede explotarse (con el método de deflación de Wielandt) para calcular *todos* los valores y vectores propios de una matriz siempre que los valores propios sean todos reales y distintos dos a dos.

En la última sección presentamos el elegante método de Jacobi, que permite calcular *siempre* los valores propios y *prácticamente siempre* los vectores propios de una matriz simétrica.

Los contenidos del capítulo se han extraído principalmente de la Sección 6.1 (pp. 111–117) del libro de Ciarlet [2], las Secciones 5.1 y 5.2 (pp. 231–250) del texto de Kincaid y Cheney [3], y la Sección 8.4 (pp. 488–511) del libro de Burden y Faires [1].

6.1. Generalidades sobre valores y vectores propios. El teorema de Gerschgorin

Recordemos que si $A \in \mathcal{M}_{n\times n}(\mathbb{C})$ es una matriz con coeficientes complejos, $\lambda \in \mathbb{C}$ es un *valor propio* de A si existe un vector $v \in \mathbb{C}^n$ no nulo (un *vector propio* asociado al valor propio λ) tal que $Av = \lambda v$. Ello es equivalente a que λ sea raíz del *polinomio característico* $p(\lambda) = |\lambda\,\mathrm{Id} - A|$ de la matriz A. Es importante resaltar que si la matriz es de coeficientes reales (lo que será habitualmente el caso), aún es posible que parte, o incluso todos sus valores propios sean complejos, pero sólo a los valores propios reales corresponderán vectores propios con coeficientes reales.

El problema del cálculo de los valores propios de una matriz es equivalente al del cálculo de las raíces de un polinomio. De una parte, los valores propios de una matriz son las raíces de su polinomio característico. E inversamente, las raíces del polinomio

$$p(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n$$

son los valores propios de la matriz

$$\begin{pmatrix} -a_{n-1} & -a_{n-2} & \dots & -a_1 & -a_0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \vdots & \vdots & & \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix},$$

que se conoce como la *matriz compañera* del polinomio p. Dado que no se dispone de algoritmos ("fórmulas") que tras un número finito de operaciones proporcionen las raíces de un polinomio de grado mayor que cuatro, tampoco podemos esperar disponer de algoritmos directos para el cálculo de valores y vectores propios: todos habrán de ser iterativos y proporcionaran, por

tanto, sólo aproximaciones (eso sí, tan precisas como se requiera) de los valores y vectores propios buscados.

Por otra parte, los problemas de valores propios suelen presentar problemas de condicionamiento. Un buen ejemplo es la matriz

$$A(\epsilon) = \begin{pmatrix} 0 & 0 & \dots & 0 & \epsilon \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 \\ & \vdots & \vdots & & \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix},$$

que en el caso $\epsilon = 0$ tiene todas sus valores propios iguales a cero. Si embargo si $\epsilon = 10^{-40}$ y la dimensión de la matriz es 40 entonces, dado que el polinomio característico es $\lambda^{40} - \epsilon = \lambda^{40} - 10^{-40}$, todos los valores propios tienen módulo 1/10: ¡un error de 10^{-40} se amplifica a un error 10^{39} veces mayor! Debe señalarse, no obstante, que las matrices simétricas no presentan este tipo de problemas (la razón se explica en [2, pp. 34–35]), lo que en particular garantiza la robustez del método de Jacobi que explicaremos en la última sección de este capítulo.

Los métodos numéricos de búsqueda de valores y vectores propios se dividen en dos clases en función de que lo que se pretenda sea conseguir todos los valores propios y una base de vectores propios, o solamente un valor propio (generalmente el de mayor tamaño), como en el método de la potencia que vamos a estudiar en primer lugar.

Para calcular aproximaciones del conjunto de valores propios de una matriz A una idea explotada asiduamente es construir una sucesión de matrices P_k tal que la sucesión de matrices $P_k^{-1}AP_k$ converge hacia una matriz de valores propios conocidos, es decir, diagonal o triangular. Esta idea está en la base de algunos métodos como el de Jacobi para las matrices simétricas que también estudiaremos, o los métodos de Givens-Householder y QR para matrices arbitrarias (véase el libro de Ciarlet [2]).

Antes de pasar a analizar estos métodos enunciamos y demostramos un resultado conocido como *teorema de los círculos de Gershgorin* que permite localizar, de manera poco precisa, la parte del plano complejo donde estarán situados los valores propios de una cierta matriz:

Teorema 6.1.1 *El conjunto de los valores propios de una matriz $A = (a_{ij}) \in \mathcal{M}_{n\times n}(\mathbb{C})$ está contenido en la unión de los discos del plano complejo*

$$D_i = \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{1\leq j\leq n, j\neq i} |a_{ij}| \right\}, \quad i = 1, 2, \dots n.$$

Demostración:

Ideas que intervienen:

- *Si λ es un valor propio de A y v es un vector propio asociado, dividiendo v por su norma obtenemos otro vector propio de norma 1.*

Sea λ un valor propio de A. Sea x un vector tal que $\|x\|_\infty = 1$ y $Ax = \lambda x$ y sea k un índice tal que $|x_k| = 1$. Como la coordenada k-ésima de Ax es λx_k, tenemos que

$$\lambda x_k = \sum_{j=1}^{n} a_{kj} x_j.$$

En consecuencia,

$$(\lambda - a_{kk})x_k = \sum_{j \neq k} a_{kj} x_j.$$

Tomando valores absolutos, aplicando la desigualdad triangular y teniendo en cuenta que $|x_j| \leq |x_k| = 1$, se obtiene

$$|\lambda - a_{kk}| \leq \sum_{j \neq k} |a_{kj}||x_j| \leq \sum_{j \neq k} |a_{kj}|,$$

de modo que $\lambda \in D_k$. □

Ejemplo 6.1.2 Los valores propios de la matriz

$$A = \begin{pmatrix} 1+i & 0 & \frac{1}{4} \\ \frac{1}{4} & 1 & \frac{1}{4} \\ 1 & 1 & 3 \end{pmatrix}$$

estarán en la unión de los discos de centro $1+i$ y radio $1/4$, de centro 1 y radio $1/2$, y de centro 3 y radio 2. En particular, si λ es valor propio de A entonces se tendrá $1/2 \leq |\lambda| \leq 5$.

6.2. El método de la potencia

Este método permite, bajo ciertas hipótesis, aproximar el valor propio de estrictamente mayor tamaño de una matriz $A \in \mathcal{M}_{n\times n}(\mathbb{R})$, si es que tal valor propio existe, con lo que en particular obtendremos el radio espectral. Nótese que en este caso el valor propio será un número real, porque el polinomio característico de la matriz tiene coeficientes reales y si un número complejo es raíz del polinomio, es decir, un valor propio de la matriz, entonces su conjugado, que tiene el mismo módulo, también será valor propio (recordaremos el motivo en el próximo capítulo).

Supondremos en todo lo que sigue que los valores propios están ordenados según su módulo:

$$\rho = |\lambda_1| = \text{máx}\{|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|\}.$$

El *método de la potencia* también se conoce como el método del cociente de Rayleigh y consiste en lo siguiente:

- se parte de una matriz A y dos vectores x_0 e y;
- se construye un sucesión de vectores x_k y una sucesión de cocientes r_k como

$$x_{k+1} = Ax_k \qquad \text{y} \quad r_k = \frac{\langle x_{k+1}, y\rangle}{\langle x_k, y\rangle};$$

- entonces, bajo ciertas hipótesis, la sucesión r_k converge hacia λ_1 y los vectores normalizados $x_k/\|x_k\|$ convergen a un vector propio de valor propio λ_1.

Veamos cuáles son esas hipótesis. En primer lugar es necesario elegir x_0 e y de forma que, si $v_1, v_2 \ldots, v_n$ es una base (en el espacio vectorial $\mathbb{C}^n$ sobre el cuerpo de los complejos $\mathbb{C}$) de vectores propios asociados a los valores propios $\lambda_1, \lambda_2, \ldots, \lambda_n$ (suponiendo que tal base existe), entonces x_0 no sea combinación lineal de los vectores $v_2 \ldots, v_n$. También suponemos $\langle v_1, y\rangle \neq 0$

y $\langle x_k, y\rangle \neq 0$ para cada k. Finalmente hay que suponer que $|\lambda_1| > |\lambda_2|$ lo que excluye la posibilidad de que este valor propio dominante λ_1 sea una raíz múltiple o compleja del polinomio característico.

Si se cumplen las hipótesis, lo que normalmente, en la práctica, seremos incapaces de verificar (ése es el principal problema del método de la potencia), entonces se demuestra que:

Teorema 6.2.1 *Con las hipótesis anteriores, la sucesión r_k converge hacia λ_1.*

Demostración:

Ideas que intervienen:

- *Si $\phi(x) = \langle x, y\rangle$, basta escribir r_k en la forma $\lambda_1 \frac{\phi(\alpha_1 v_1 + \epsilon_{k+1})}{\phi(\alpha_1 v_1 + \epsilon_k)}$ para una sucesión vectorial (ϵ_k) convergente a cero.*

Reescribamos $\phi(x) = \phi_y(x) = \langle x, y\rangle$ y pongamos x_0 como combinación lineal de los elementos de la base,

$$x_0 = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n.$$

Observemos que $\alpha_0 \neq 0$ por una de las hipótesis. Notemos que, por la forma en que está definida la sucesión x_k, se tiene $x_k = A^k x_0$, donde A^k es el resultado de multiplicar la matriz A por sí misma k veces. Como los vectores v_j son vectores propios de A, también lo serán para cada A^k con valores propios λ_j^k, y así

$$x_k = \alpha_1 \lambda_1^k v_1 + \alpha_2 \lambda_2^k v_2 + \cdots + \alpha_n \lambda_n^k v_n = \lambda_1^k \left(\alpha_1 v_1 + \sum_{j=2}^{n} \left(\frac{\lambda_j}{\lambda_1} \right)^k \alpha_j v_j \right) =: \lambda_1^k (\alpha_1 v_1 + \epsilon_k),$$

con $\epsilon_k \to 0$ porque λ_1 es el valor propio dominante y los cocientes $(\lambda_j/\lambda_1)^k$ tienden a cero.

Por ser ϕ lineal (y por tanto continua), y ocurrir que $\alpha_1 \neq 0$ y $\phi(v_1) \neq 0$ por las hipótesis establecidas al principio, podemos tomar límites en la expresión

$$r_k = \frac{\phi(x_{k+1})}{\phi(x_k)} = \lambda_1 \frac{\phi(\alpha_1 v_1 + \epsilon_{k+1})}{\phi(\alpha_1 v_1 + \epsilon_k)} = \lambda_1 \frac{\alpha_1 \phi(v_1) + \phi(\epsilon_{k+1})}{\alpha_1 \phi(v_1) + \phi(\epsilon_k)}$$

y concluir $\lim_{k\to\infty} r_k = \lambda_1$. □

Ejemplo 6.2.2 Considérese la matriz

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix}.$$

En este caso es muy sencillo calcular directamente las tres raíces $\lambda_1 = 3$, $\lambda_2 = 1$ y $\lambda_3 = -1$ del polinomio característico. En particular, 3 será el valor propio dominante. Si partimos de $x_0 = (1, 0, 0)$, las primeras iteradas son los vectores

$$\begin{aligned} x_1 &= (1, 2, 4), \\ x_2 &= (5, 4, 8), \\ x_3 &= (13, 14, 28), \\ x_4 &= (41, 40, 80), \\ x_5 &= (121, 122, 244). \end{aligned}$$

Si tomamos $y = (1, 0, 0)$, los correspondientes productos escalares serán

$$\begin{aligned}
\langle x_0, y\rangle &= 1,\\
\langle x_1, y\rangle &= 1,\\
\langle x_2, y\rangle &= 5,\\
\langle x_3, y\rangle &= 13,\\
\langle x_4, y\rangle &= 41,\\
\langle x_5, y\rangle &= 121,
\end{aligned}$$

que proporcionan los cocientes

$$\begin{aligned}
r_1 &= 1,\\
r_2 &= 5\\
r_3 &= 2.6,\\
r_4 &= 3.15384615,\\
r_5 &= 2.95121951.
\end{aligned}$$

Como vemos, bastan cinco iteraciones para aproximar el valor propio dominante con un error inferior a 0.05.

En el ejemplo anterior la lista de vectores x_k sugiere que $(1, 1, 2)$ es un vector propio de valor propio 3, y en efecto así ocurre. Sin embargo no podemos decir que los vectores x_k convergen a $(1, 1, 2)$: sus módulos crecen hacia infinito. En general los vectores $x_k = \lambda_1^k(\alpha_1 v_1 + \epsilon_k)$ no pueden converger a ningún vector propio porque o bien convergen a cero si $|\lambda_1| < 1$, o bien divergen a ∞ si $|\lambda_1| > 1$. Sin embargo la sucesión

$$y_k := \frac{x_k}{\|x_k\|} = \frac{\lambda_1^k(\alpha_1 v_1 + \epsilon_k)}{|\lambda_1|^k \|\alpha_1 v_1 + \epsilon_k\|}$$

converge al vector propio $\alpha_1 v_1/\|\alpha_1 v_1\|$ si $\lambda_1 > 0$ y alternativamente (en términos pares e impares) a los vectores propios $\pm\alpha_1 v_1/\|\alpha_1 v_1\|$ si $\lambda_1 < 0$.

A la hora de escribir el algoritmo es importante evitar calcular y_k obteniendo primero el correspondiente vector x_k y dividiendo luego por su norma, pues como ha quedado dicho los vectores x_k pueden diverger a infinito y los errores se dispararían. En lugar de eso se calcula la sucesión (y_k) por recurrencia. Partiendo de $y_0 = x_0/\|x_0\|$, y supuesto y_k conocido, se tendrá que

$$y_{k+1} = \frac{x_{k+1}}{\|x_{k+1}\|} = \frac{A(x_k)}{\|x_{k+1}\|} = \frac{\|x_k\|}{\|x_{k+1}\|} A(y_k).$$

Tomando normas,

$$1 = \|y_{k+1}\| = \frac{\|x_k\|}{\|x_{k+1}\|} \|A(y_k)\|,$$

es decir,

$$y_{k+1} = \frac{A(y_k)}{\|A(y_k)\|}.$$

Por tanto la sucesión (y_{k+1}) puede calcularse manejando siempre vectores uniformemente acotados por la norma de la matriz A, con lo que los errores estarán bajo control.

Puede probarse además que la velocidad de la convergencia depende del tamaño del cociente. En concreto y para una constante C se cumple

$$|r_k - \lambda_1| \leq C \left(\frac{|\lambda_2|}{|\lambda_1|} \right)^k$$

(ver [3, pp. 236–237]). Cuando $|\lambda_2|$ está próximo a $|\lambda_1|$ la convergencia es muy lenta. En este caso se puede acelerar la convergencia haciendo la sucesión t_k del método de la Δ^2 de Aitken. La velocidad de convergencia de esta sucesión t_k depende el cociente $\frac{|\lambda_3|}{|\lambda_1|}$. Si $|\lambda_3|$ está muy próximo a $|\lambda_2|$ no se gana prácticamente nada en velocidad; en este caso se puede volver a acelerar aplicando el método de Aitken a t_k. La velocidad de convergencia de esta nueva sucesión dependerá del cociente $\frac{|\lambda_4|}{|\lambda_1|}$. Se podría proseguir con estas aceleraciones, aunque desde el punto de vista de la programación del algoritmo no es aconsejable por el riesgo de hacer demasiadas iteraciones.

En el algoritmo 6.1 describimos el método sin incluir la aceleración de Aitken: el lector puede añadirla al algoritmo si así lo desea.

Es sencillo diseñar diversas variantes del método de la potencia que permitan obtener información sobre otros valores propios de A. Por ejemplo, si A es invertible, entonces λ es valor propio de A si y sólo si λ^{-1} lo es de A^{-1} con el mismo vector propio:

$$Au = \lambda u \Leftrightarrow \lambda^{-1} u = A^{-1} u.$$

En consecuencia, para hallar el valor propio de módulo *mínimo* bastaría aplicar el método de la potencia a A^{-1} y calcular el inverso de su valor propio dominante. Nótese que en este caso, en lugar de calcular la inversa de A previamente para luego ir generando la sucesión $x_{k+1} = A^{-1} x_k$, es más eficiente resolver el sistema $Ax_{k+1} = x_k$ (guardando las operaciones realizadas sobre la matriz A para no repetirlas en cada iteración). A este método se le denomina el *método de la potencia inversa*. Análogamente, los valores propios de la matriz $A - \mu\,\mathrm{Id}$ son los números de la forma $\lambda - \mu$, con λ los valores propios de A, con lo que los métodos de la potencia y de la potencia inversa aplicados a esta matriz (a estos métodos se les llama métodos de la potencia y la potencia inversa *con desplazamiento*) nos darían, respectivamente, el valor propio más alejado y más cercano a μ. En particular, si tenemos una idea aproximada de donde puede estar situado un cierto valor propio, el método de la potencia inversa con desplazamiento permitirá calcularlo con bastante rapidez.

Algoritmo 6.1 **Valor propio dominante (método de la potencia)**

Datos de entrada:
$A[n][n]$ (matriz cuyo valor propio dominante queremos obtener);
tol (precisión para la condición de parada);
nmax (número máximo de iteraciones);
v (vector inicial);
y (vector para los productos);
Variables:
uini; // vector para almacenar y_k
ufin; // vector para almacenar y_{k+1}
rini; // real para almacenar r_k
rfin; // real para almacenar r_{k+1}
Flujo del programa:
// Inicialización de las variables.
rini $= 0$;
uini $= v/\|v\|$;
// Vamos a hacer las etapas $m = 1, 2, ...,$ nmax.
`for(m=1;m<=nmax;m++){`
  `if(`$|\langle$`uini`$, y\rangle| = 0$`){`
    Parada: el método de la potencia no es aplicable
  }
  ufin $= A \cdot$ uini;
  `if(ufin` $= 0$`){`
    Parada: el método de la potencia no es aplicable
  }
  rfin $= \langle$ufin$, y\rangle / \langle$uini$, y\rangle$;
  ufin $=$ ufin$/\|$ufin$\|$;
  `if(|rfin` $-$ `rini|` $<$ `tol){`
    Parada: rfin es el valor propio dominante y ufin el correspondiente vector propio
  }
  rini $=$ rfin;
  uini $=$ ufin;
}
Parada: no hay convergencia en nmax iteraciones
Datos de salida: Valor propio dominante de la matriz A y correspondiente vector propio, o mensaje de error si la iteración no converge.

El *método de deflación de Wielandt* enfoca la cuestión de una manera más ambiciosa. La idea es partir del método de la potencia para obtener el valor propio dominante λ_2 y un vector propio asociado, y a partir de ellos generar una matriz $(n-1)\times(n-1)$ que tenga como valores propios $\lambda_2, \dots \lambda_n$ y dar una fórmula que permita calcular los vectores propios de la matriz original a partir de los de la nueva. Aplicando el método de la potencia a la nueva matriz, podremos obtener λ_2 y su correspondiente vector propio. Repitiendo el proceso, podremos obtener *todos* los valores y vectores propios de la matriz original. En cierto sentido, como vemos, el proceso recuerda al de la resolución de una ecuación polinómica: una vez que encontramos una raíz del polinomio, lo factorizamos y el problema queda reducido a encontrar las raíces de un polinomio un grado menor. Naturalmente el proceso sólo puede llegar a buen puerto si todos los valores propios son reales y distintos (lo que no podemos saber a priori). Nótese que en este caso cada subespacio propio tiene dimensión uno así que, salvo multiplicación por constantes, existen n vectores propios.

Dicho sea de paso, estos vectores propios formarán una base. En efecto, si $v_1, \dots, v_n$ son vectores propios de valores propios $\lambda_1, \dots, \lambda_n$, con $\lambda_i \neq \lambda_j$ si $i \neq j$, entonces los vectores $v_1, \dots, v_n$ son linealmente independientes. Esto es fácil de demostrar por inducción sobre n. La afirmación es obvia si $n = 1$. Supongámosla cierta para $n-1$ y supongamos que $\alpha_1 v_1 + \alpha_2 v_2 + \dots \alpha_n v_n = 0$. Entonces, por un lado, multiplicando por λ_1,

$$\alpha_1\lambda_1 v_1 + \alpha_2\lambda_1 v_2 + \dots + \alpha_n\lambda_1 v_n = 0,$$

y por otro, multiplicando por A,

$$\alpha_1\lambda_1 v_1 + \alpha_2\lambda_2 v_2 + \dots + \alpha_n\lambda_n v_n = 0.$$

Restando ambas igualdades

$$\alpha_2(\lambda_2 - \lambda_1)v_2 + \dots \alpha_n(\lambda_n - \lambda_1)v_n = 0,$$

y como los números $\lambda_j - \lambda_1$ son distintos de cero, la hipótesis de inducción implica $\alpha_2 = \dots = \alpha_n = 0$, con lo que $\alpha_1 v_1 = 0$ y también $\alpha_1 = 0$. Hemos probado que los vectores $v_1, \dots, v_n$ son linealmente independientes.

Nótese que el mismo argumento demuestra que si un valor propio λ_1 es distinto del resto y $v_2, \dots, v_n$ es una familia linealmente independiente de vectores propios de valores propios $\lambda_2, \dots, \lambda_n$, entonces cada vector propio v_1 de λ_1 forma, junto con $v_2, \dots, v_n$, una base.

El método de deflación de Wielandt se basa en el siguiente resultado:

Teorema 6.2.3 *Supongamos que λ_1 es un valor propio de A con vector propio v_1 y x es un vector tal que $x^T v_1 = 1$. Entonces 0 es valor propio de*

$$B = A - \lambda_1 v_1 x^T \tag{6.1}$$

con vector propio v_1. Si además $w_2 \dots, w_n$ son vectores propios de B de valores propios $\lambda_2, \dots, \lambda_n$ y $0 \neq \lambda_j \neq \lambda_i$ para cada $j \neq 1$, entonces los vectores

$$v_j = (\lambda_j - \lambda_1)w_j + \lambda_1(x^T w_j)v_1 \tag{6.2}$$

son vectores propios de A con valores propios λ_j, $j = 2, \dots, n$.

DEMOSTRACIÓN:

Ideas que intervienen:

- *Dados los vectores $v = (v_1, \ldots, v_n)$ y $x = (x_1, \ldots, x_n)$, vx^T es la matriz $n \times n$ que tiene como coeficiente ij el producto $v_i x_j$.*
- *Por la asociatividad del producto de matrices, si w es otro vector, entonces el resultado de multiplicar la matriz vx^T por w es el de multiplicar el vector v por el producto escalar $x^T w$.*

Por ser λ_1 valor propio de vector propio v_1, por la propiedad asociativa del producto de matrices y usando la hipótesis $x^T v_1 = 1$,

$$Bv_1 = Av_1 - \lambda_1(v_1 x^T)v_1 = \lambda_1 v_1 - \lambda_1 v_1(x^T v_1) = \lambda_1 v_1 - \lambda_1 v_1 = 0.$$

Demostramos a continuación que los vectores v_j son vectores propios de A de valor propio λ_j, $j = 2, \ldots, n$.

Notemos para empezar que los vectores v_j son no nulos porque $\lambda_j \neq 0$ para cada $j \neq 1$, cada par de vectores w_j, v_1 es linealmente independiente y $\lambda_j - \lambda_1 \neq 0$. Además,

$$\begin{aligned}
Av_j &= (A - \lambda_1 v_1 x^T + \lambda_1 v_1 x^T)v_j \\
&= Bv_j + \lambda_1(v_1 x^T)v_j \\
&= B[(\lambda_j - \lambda_1)w_j + \lambda_1(x^T w_j)v_1] \\
&\qquad +\lambda_1(v_1 x^T)[(\lambda_j - \lambda_1)w_j + \lambda_1(x^T w_j)v_1] \\
&= \lambda_j(\lambda_j - \lambda_1)w_j \\
&\qquad +\lambda_1(\lambda_j - \lambda_1)(v_1 x^T)w_j + \lambda_1(x^T w_j)\lambda_1(v_1 x^T)v_1 \\
&= \lambda_j(\lambda_j - \lambda_1)w_j \\
&\qquad +\lambda_1(\lambda_j - \lambda_1)(x^T w_j)v_1 + \lambda_1\lambda_1(x^T w_j)v_1 \\
&= \lambda_j(\lambda_j - \lambda_1)w_j \\
&\qquad +\lambda_j\lambda_1(x^T w_j)v_1 \\
&= \lambda_j v_j;
\end{aligned}$$

en la tercera igualdad hemos usado que w_j es vector propio de B de valor propio λ_j y que $Bv_1 = 0$. □

En el método de deflación de Wielandt el vector x se elige de acuerdo con la fórmula

$$x = \frac{1}{\lambda_1 v_{1,k}} \begin{pmatrix} a_{k1} \\ a_{k2} \\ \vdots \\ a_{kn} \end{pmatrix}, \tag{6.3}$$

donde $v_{1,k}$ es una componente no nula del vector v_1 (suele elegirse la de mayor valor absoluto para minimizar los errores de cálculo) y $(a_{k1}, a_{k2}, \ldots, a_{kn})$ es la fila k-ésima de la matriz A. En efecto, obsérvese que $(a_{k1}, a_{k2}, \ldots, a_{kn})^T v_1$ es la componente k-ésima del vector $Av_1 = \lambda_1 v_1$, es decir,

$$(a_{k1}, a_{k2}, \ldots, a_{kn})^T v_1 = \lambda_1 v_{1,k}$$

y por tanto $x^T v_1 = 1$.

La ventaja de escoger x de esta manera radica en que la matriz $B = A - \lambda_1 v_1 x^T$ tiene ceros en la fila k-ésima, dado que la componente c_{kj} de la matriz $\lambda_1 v_1 x^T$ es el número

$$\lambda_1 v_{1,k} x_j = \lambda_1 v_{1,k} \frac{1}{\lambda 1 v_{1,k}} a_{kj} = a_{kj}.$$

Ello significa que si w es vector propio de B de valor propio $\lambda \neq 0$, entonces $w_k = 0$. Más aún, si A' es la matriz $(n-1) \times (n-1)$ que resulta de eliminar la fila y columna k-ésimas de A y v' es un vector propio de A' de valor propio λ, entonces el vector w que resulta de añadir a v' un cero en el lugar k-ésimo es un vector propio de B de valor propio λ.

En resumen, el algoritmo tiene las siguientes fases:

(I) Se obtiene el valor propio dominante λ_1 de A por el método de la potencia y su vector propio asociado.

(II) Se identifica la coordenada k-ésima de v_1 de mayor valor absoluto y se construye el vector x de acuerdo con la fórmula (6.3).

(III) Se construye la matriz B según la formula (6.1) y, tras quitarle la fila y columna k-ésimas, la matriz $A' \in \mathcal{M}_{(n-1)\times(n-1)}(\mathbb{R})$.

(IV) A partir de los valores propios $\lambda_2, \ldots, \lambda_n$ de A' y la correspondiente base de vectores propios $v'_2, \ldots, v'_n$, construimos vectores propios $w_2, \ldots, w_n$ para B añadiendo ceros en el lugar k-ésimo.

(V) Finalmente, a partir de la fórmula (6.2) generamos los vectores propios $v_2, \ldots, v_n$ de A que junto a v_1 completarán la base de vectores propios buscada.

Notemos que se trata de un algoritmo recursivo: la parte clave del es el punto 4, donde el algoritmo se llama a sí mismo, de modo que la dimensión de la matriz se va reduciendo hasta que se llega a una matriz de dimensión uno, que tiene trivialmente como valor propio su única componente y como vector propio la unidad. Tal y como lo hemos escrito a continuación, se ha determinado que cada vez que llame al método de la potencia fije los vectores iniciales x_0 e y al azar.

Algoritmo 6.2 **Valores y vectores propios (método de deflación de Wielandt)**

Datos de entrada:
$A[n][n]$ (matriz cuyos valores propios queremos obtener);
tol (precisión para la condición de parada);
nmax (número máximo de iteraciones);
Variables:
Sol$[n][n+1]$; // matriz para devolver los resultados
$B[n-1][n-1]$; // matriz deflacionada
Soldefl$[n-1][n]$; // matriz con los valores y vectores propios de B
vectvaldom; // vector para guardar el vector y valor propio dominantes de A
u, lambda // vector y real para guardar el contenido de vectvaldom
v, y, x, w; // vectores auxiliares
max, lambdadefl; // reales auxiliares
k; // entero auxiliar
Flujo del programa:
// El caso $n = 1$ es trivial.
`if(`$n = 1$`){`
  Sol $= (1, A[0][0])$;
  Parada: caso trivial
}
// Si $n = 1$ calculamos los valores y vectores propios recursivamente.
`else{`
  // Elegimos al azar v e y para el método de la potencia.
  `for(i=0;i<n;i++){`
    $v[i] =$ `Math.random();`
    $y[i] =$ `Math.random();`
  }
  // Aplicamos el método de la potencia para obtener el valor propio dominante.
  vectvaldom = `potencia(A, tol, nmax, v, y);`
  `for(i=0;i<n;i++){`
    $u[i] =$ vectvaldom$[i]$;
  }
  lambda = vectvaldom$[n]$;
  // Elegimos la componente más grande $u[k]$ del vector propio u.
  aux $= 0$;
  $k = 0$;
  `for(i=0;i<n;i++){`
    `if(|u[i]|>|aux|){`
      aux $= u[i]$;
      $k = i$;
    }
  }

Algoritmo 6.2 **Valores y vectores propios (cont. mét. de deflación de Wielandt)**

```
// Calculamos la matriz deflacionada B a partir de k.
x = A[k][]/(lambda * aux);
B ← A − lambda u x^T;
// Calculamos los valores y vectores propios de B.
Soldefl = deflacionWielandt(B, tol, nmax);
if(|Soldefl[n-2][n-1]-lambda|<tol){
  Parada: valor propio múltiple
}
// Obtenemos los valores y vectores propios de A.
for(i=0;i<n-1;i++){
  for(j=0;j<k;j++){
    w[j] = Soldefl[i][j];
  }
  w[k] = 0;
  for(j=k+1;j<n;j++){
    w[j] = Soldefl[i][j − 1];
  }
  lambdadefl = Soldefl[i][n − 1];
  w = (lambdadefl − lambda) * w + lambda (x^T w) u;
  for(j=0;j<n;j++){
    Sol[i][j] = w[j]/‖w‖;
  }
  Sol[i][n] = lambdadefl;
}
Sol[n − 1][] = vectvaldom;

}
```

Datos de salida: Valores y vectores propios de la matriz A y correspondiente vector propio, o mensaje de error si el método no funciona.

Ejemplo 6.2.4 A continuación ilustramos el método de deflación de Wielandt aplicándolo al cálculo de los valores y vectores propios de la matriz

$$A = \begin{pmatrix} 4 & -1 & 0 & 2 \\ -2 & 5 & 0 & 1 \\ 3 & -1 & 1 & -3/2 \\ 0 & 0 & 0 & 8 \end{pmatrix}.$$

Necesitamos para empezar el valor propio dominante λ_1 y el correspondiente vector propio asociado. Para ello usaríamos el método de la potencia, que nos proporcionaría $\lambda_1 = 8$ y $v_1 = (1, 0, 0, 2)$. (Por supuesto, en la práctica el método de la potencia no proporcionará el valor exacto 8, sino una —muy buena– aproximación, y lo mismo ocurrirá con v_1, que además aparecerá dividido por su norma.) Por tanto $k = 4$, de donde

$$x = 1/(\lambda_1 v_{1,4})(a_{41}, a_{42}, a_{43}, a_{44}) = (1/(8 \cdot 2))(0, 0, 0, 8) = (0, 0, 0, 1/2).$$

A continuación calculamos

$$\begin{aligned} B &= A - \lambda_1 (v_1 x^T) \\ &= \begin{pmatrix} 4 & -1 & 0 & 2 \\ -2 & 5 & 0 & 1 \\ 3 & -1 & 1 & -3/2 \\ 0 & 0 & 0 & 8 \end{pmatrix} - 8 \begin{pmatrix} 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 4 & -1 & 0 & 2 \\ -2 & 5 & 0 & 1 \\ 3 & -1 & 1 & -3/2 \\ 0 & 0 & 0 & 8 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 \end{pmatrix} \\ &= \begin{pmatrix} 4 & -1 & 0 & -2 \\ -2 & 5 & 0 & 1 \\ 3 & -1 & 1 & -3/2 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \end{aligned}$$

Suprimiendo la cuarta fila y columna de B llegamos a

$$A' = \begin{pmatrix} 4 & -1 & 0 \\ -2 & 5 & 0 \\ 3 & -1 & 1 \end{pmatrix}.$$

De nuevo, supongamos que tras aplicar el método de la potencia obtenemos el valor propio dominante $\lambda_2 = 6$ de A' y un vector propio asociado, digamos $v_2' = (1, -2, 1)$. En este caso $k = 2$, con lo que

$$x' = 1/(\lambda_2 v'_{2,2})(a'_{21}, a'_{22}, a'_{23}) = (1/(6(-2)))(-2, 5, 0) = (1/6, -5/12, 0).$$

Ahora

$$\begin{aligned} B' &= A' - \lambda_2 (v_2' (x')^T) \\ &= \begin{pmatrix} 4 & -1 & 0 \\ -2 & 5 & 0 \\ 3 & -1 & 1 \end{pmatrix} - 6 \begin{pmatrix} 1/6 & -5/12 & 0 \\ -1/3 & 5/6 & 0 \\ 1/6 & -5/12 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 4 & -1 & 0 \\ -2 & 5 & 0 \\ 3 & -1 & 1 \end{pmatrix} - \begin{pmatrix} 1 & -5/2 & 0 \\ -2 & 5 & 0 \\ 1 & -5/2 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 3 & 3/2 & 0 \\ 0 & 0 & 0 \\ 2 & 3/2 & 1 \end{pmatrix}. \end{aligned}$$

A partir la matriz B', suprimiendo la segunda fila y columna, obtenemos

$$A'' = \begin{pmatrix} 3 & 0 \\ 2 & 1 \end{pmatrix}.$$

A esta matriz aplicaríamos de nuevo el método de la potencia (por supuesto para una matriz 2×2 siempre podemos hacer los cálculos a mano, pero adoptamos el "punto de vista"de la máquina,

que no distingue entre matrices grandes y pequeñas) y obtendríamos su valor propio dominante, $\lambda_3 = 3$, y el correspondiente vector propio asociado, $v_3'' = (1, 1)$. En este caso tomamos $k = 1$, de donde

$$x'' = 1/(\lambda_3 v''_{3,1})(a''_{11}, a''_{12}) = (1/(3 \cdot 1))(3, 0) = (1, 0).$$

Ahora

$$\begin{aligned} B'' &= A'' - \lambda_3(v_3''(x'')^T) \\ &= \begin{pmatrix} 3 & 0 \\ 2 & 1 \end{pmatrix} - 3 \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ -1 & 1 \end{pmatrix}. \end{aligned}$$

Tras suprimir la primera fila y columna llegamos a $A''' = (1)$, que tiene trivialmente como vector propio $\lambda_4 = 1$ y como vector propio asociado $v_4''' = (1)$.

Ya sabemos que los cuatro valores propios de A son $\lambda_1 = 8$, $\lambda_2 = 6$, $\lambda_3 = 3$, $\lambda_4 = 1$ y que el vector propio asociado a λ_1 es $v_1 = (1, 0, 0, 2)$. Para calcular los otros tres tenemos que deshacer el camino andado. Para empezar generamos

$$w_4'' = (0, 1)$$

añadiendo un cero en la primera componente a v_4''' y, a partir de él, el vector

$$\begin{aligned} v_4'' &= (\lambda_4 - \lambda_3)w_4'' + \lambda_3((x'')^T w_4'')v_3'' \\ &= -2(0, 1) + 3(0)(1, 1) \\ &= (0, -2). \end{aligned}$$

Así pues, a los valores propios de A'', $\lambda_3 = 3$ y $\lambda_4 = 1$, corresponden los vectores propios $v_3'' = (1, 1)$, que obtuvimos con el método de la potencia, y $v_4'' = (0, -2)$.

A continuación construimos

$$w_3' = (1, 0, 1)$$

y

$$w_4' = (0, 0, -2)$$

a partir de los vectores v_3'' y v_4'' añadiendo un cero en la segunda componente. En este punto calculamos

$$\begin{aligned} v_3' &= (\lambda_3 - \lambda_2)w_3' + \lambda_2((x')^T w_3')v_2' \\ &= (-3)(1, 0, 1) + 6(1/6)(1, -2, 1) \\ &= (-3, 0, -3) + (1, -2, 1) \\ &= (-2, -2, -2), \end{aligned}$$

y

$$\begin{aligned} v_4' &= (\lambda_4 - \lambda_2)w_4' + \lambda_2((x')^T w_4')v_2' \\ &= (-5)(0, 0, -2) + 6(0)(1, -2, 1) \\ &= (0, 0, 10), \end{aligned}$$

que son vectores propios de valores propios $\lambda_3 = 3$ y $\lambda_4 = 1$ para A', a los que tenemos que añadir $v_2' = (1, -2, 1)$, que era el vector inicial de valor propio $\lambda_2 = 6$ que proporcionaba el método de la potencia.

Ya casi hemos terminado. Generamos

$$w_2 = (1, -2, 1, 0)$$

$$w_3 = (-2, -2, -2, 0)$$

y

$$w_4 = (0, 0, 10, 0)$$

a partir de los vectores v_2', v_3' y v_4' añadiendo un cero en la cuarta componente. (Nótese que podemos tomar, si así lo deseamos, los vectores más sencillos $(1, 1, 1, 0)$ y $(0, 0, 1, 0)$ en lugar de w_3 y w_4, pues son igualmente vectores propios de B). Finalmente

$$\begin{aligned} v_2 &= (\lambda_2 - \lambda_1)w_2 + \lambda_1(x^T w_2)v_1 \\ &= (-2)(1, -2, 1, 0) + 8(0)(1, 0, 0, 2) \\ &= (-2, 4, -2, 0), \end{aligned}$$

$$\begin{aligned} v_3 &= (\lambda_3 - \lambda_1)w_3 + \lambda_1(x^T w_3)v_1 \\ &= (-5)(-2, -2, -2, 0) + 8(0)(1, 0, 0, 2) \\ &= (10, 10, 10, 0), \end{aligned}$$

y

$$\begin{aligned} v_4 &= (\lambda_3 - \lambda_1)w_4 + \lambda_1(x^T w_4)v_1 \\ &= (-7)(0, 0, 10, 0) + 8(0)(1, 0, 0, 2) \\ &= (0, 0, -70, 0), \end{aligned}$$

junto a $v_1 = (1, 0, 0, 2)$, completan la base de vectores propios de A que buscábamos. Naturalmente, en lugar de v_2, v_3 y v_4 podemos usar respectivamente los vectores propios más sencillos $(1, -2, 1, 0)$, $(1, 1, 1, 0)$ y $(0, 0, 1, 0)$.

6.3. El método de Jacobi

Este método se emplea cuando buscamos todos los valores propios y una base de vectores propios de una matriz simétrica real.

Recordemos que las matrices simétricas son diagonalizables: existe una matriz ortogonal Ω (es decir, una matriz cuya traspuesta coincide con su inversa) tal que

$$\Omega^T A \Omega = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \lambda_n \end{pmatrix}$$

es diagonal con los valores propios de A en la diagonal. En particular todos los valores propios de A son reales (aunque pueden repetirse) y los vectores columna de la matriz Ω forman una base ortonormal de vectores propios, siendo el vector de la columna i el vector propio asociado al valor propio λ_i.

El método de Jacobi consiste en ir construyendo una sucesión de matrices ortogonales "elementales"(por su forma simple) $(O_k)_{k=1}^{\infty}$ de manera que la sucesión de matrices:

$$A_0 = A, \qquad A_k = O_k^T A_{k-1} O_k = (O_1 \cdots O_k)^T A (O_1 \cdots O_k) \ \ (k \geq 1)$$

converja hacia una matriz diagonal formada por los valores propios. Así, la esperanza será que la sucesión de matrices ortogonales $\Omega_0 = \mathrm{Id}$ y $\Omega_k = \Omega_{k-1} O_k = O_1 \cdots O_k$ converja hacia una matriz ortogonal cuyas columnas formen una base ortogonal de vectores propios.

La idea de la construcción es la de ir anulando en cada paso k dos elementos de la matriz A_k que estén fuera de la diagonal y en posiciones simétricas (los correspondientes a ciertos coeficientes pq y qp). Para ello se utilizan rotaciones en el plano determinado por los vectores p-ésimo y q-ésimo de la base canónica de $\mathbb{R}^n$ descritas por las matrices ortogonales:

$$O = \begin{array}{c} \begin{matrix} & & & p & & & q & & & \end{matrix} \\ \begin{matrix} \downarrow & & & \downarrow \end{matrix} \\ \begin{pmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & \cos\theta & & & \operatorname{sen}\theta & & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & -\operatorname{sen}\theta & & & \cos\theta & & & \\ & & & & & & & 1 & & \\ & & & & & & & & \ddots & \\ & & & & & & & & & 1 \end{pmatrix} \end{array} \begin{matrix} \\ \\ \\ \\ \leftarrow \ p \\ \\ \\ \\ \leftarrow \ q \\ \\ \\ \\ \end{matrix}$$

Lema 6.3.1 *Sean p y q enteros $1 \leq p < q \leq n$ y θ un número real a los cuales asociamos la matriz ortogonal O descrita más arriba.*

(i) Si $A = (a_{ij})$ es una matriz $n \times n$ simétrica, entonces la matriz

$$B = O^T A O = (b_{ij})$$

también es simétrica y cumple

$$\sum_{i,j=1}^{n} b_{ij}^2 = \sum_{i,j=1}^{n} a_{ij}^2.$$

(ii) Si $a_{pq} \neq 0$, existe un único valor de $\theta \in (-\frac{\pi}{4}, \frac{\pi}{4}] \setminus \{0\}$ tal que $b_{pq} = 0$. El número θ está determinado por la ecuación

$$\cot 2\theta = \frac{a_{qq} - a_{pp}}{2a_{pq}}.$$

Para este valor de θ se cumple

$$\sum_{i=1}^{n} b_{ii}^2 = \sum_{i=1}^{n} a_{ii}^2 + 2a_{pq}^2.$$

DEMOSTRACIÓN:

Ideas que intervienen:

- *La norma*
$$\|A\|_{DOS} = \left(\sum_{i,j=1}^{n} a_{ij}^2 \right)^{1/2}$$
puede expresarse en términos de la traza matricial y ésta es invariante por cambios de base.
- *Si se multiplica una matriz A por la derecha por otra que tiene como vectores fila, salvo los de las filas p y q, los vectores de la base canónica, entonces la matriz producto tiene los mismos vectores columna que A excepto los correspondientes a las columnas p y q.*

(i) Como A es simétrica,
$$B^T = (O^T AO)^T = O^T A^T (O^T)^T = O^T AO = B,$$
así que B es simétrica también.

Por otra parte, recordemos que la traza $\operatorname{tr} C = \sum_{i=1}^{n} c_{ii}$ de una matriz $C = (c_{ij})$ se conserva por cambios de base, es decir, si P es invertible entonces $\operatorname{tr}(P^{-1}CP) = \operatorname{tr} C$. En particular, si recordamos que $O^T = O^{-1}$, tendremos
$$\begin{aligned} \sum_{i,j=1}^{n} b_{ij}^2 &= \operatorname{tr}(B^T B) = \operatorname{tr}(BB) = \operatorname{tr}(O^T AOO^T AO) \\ &= \operatorname{tr}(O^{-1} AAO) = \operatorname{tr}(AA) = \operatorname{tr}(A^T A) \\ &= \sum_{i,j=1}^{n} a_{ij}^2. \end{aligned}$$

(ii) Por la estructura de la matriz O, si C es una matriz cualquiera el resultado del producto CO es una matriz con los mismos vectores columna que C excepto que los vectores columna p-ésimo y q-ésimo de C, v_p y v_q, son reemplazados respectivamente por $\cos\theta v_p - \operatorname{sen}\theta v_q$ y $\operatorname{sen}\theta v_p + \cos\theta v_q$. Análogamente, el producto $O^T C$ tiene los mismos vectores fila que C excepto que los vectores fila p-ésimo y q-ésimo de C, w_p y w_q, son reemplazados respectivamente por $\cos\theta w_p - \operatorname{sen}\theta w_q$ y $\operatorname{sen}\theta w_p + \cos\theta w_q$.

Así pues, B y A tienen los mismos coeficientes salvo los de las filas y las columnas p y q. Más aún, los coeficientes de los lugares pp, pq, qp y qq están conectados por la igualdad
$$\begin{pmatrix} b_{pp} & b_{pq} \\ b_{qp} & b_{qq} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\operatorname{sen}\theta \\ \operatorname{sen}\theta & \cos\theta \end{pmatrix} \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} \cos\theta & \operatorname{sen}\theta \\ -\operatorname{sen}\theta & \cos\theta \end{pmatrix},$$
y razonando como en (i) obtenemos $a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2 = b_{pp}^2 + b_{qq}^2 + 2b_{pq}^2$ independientemente de lo que valga θ.

Haciendo operaciones y usando las conocidas fórmulas trigonométricas $\cos 2\theta = \cos^2\theta - \operatorname{sen}^2\theta$ y $\operatorname{sen} 2\theta = 2\operatorname{sen}\theta\cos\theta$, obtenemos
$$b_{pq} = b_{qp} = a_{pq}\cos 2\theta + \frac{a_{pp} - a_{qq}}{2}\operatorname{sen} 2\theta.$$

Por tanto, si elegimos θ como en el enunciado de (ii) se tendrá que $b_{pq} = b_{qp} = 0$ y en consecuencia, dado que los coeficientes en las diagonales principales de A y B son los mismos excepto los correspondientes a los lugares pp y qq, obtenemos

$$\sum_{i=1}^{n} b_{ii}^2 = \sum_{i=1}^{n} a_{ii}^2 + 2a_{pq}^2.$$

Enfatizamos que la aplicación $\theta \mapsto \cot 2\theta$ lleva biyectivamente $(-\frac{\pi}{4}, \frac{\pi}{4}] \setminus \{0\}$ a $\mathbb{R}$ así que θ está bien definido y es único. □

Antes de pasar a describir el algoritmo y estudiar los resultados de convergencia es conveniente hacer las siguientes observaciones:

(I) Si vemos $\mathcal{M}_{n\times n}(\mathbb{R})$ como el espacio euclídeo $\mathbb{R}^{n^2}$ y en él usamos la norma euclídea $\|\cdot\|_{\text{DOS}}$ habitual, es decir,

$$\|A\|_{\text{DOS}} = \left(\sum_{i,j=1}^{n} a_{ij}^2\right)^{1/2}$$

(en el libro de Ciarlet se usa la notación $\|\cdot\|_E$ para referirse a esta norma), el Lema 6.3.1(i) nos dice que la transformación $B = O^T AO$ deja invariante a la norma. El Lema 6.3.1(ii), por su parte, muestra que la suma de los cuadrados de la diagonal de B aumenta al tiempo que los coeficientes b_{pq} y b_{qp} se anulan.

(II) Esta transformación sólo afecta a las filas p y q, y las columnas p y q. De forma más precisa:

- $b_{pi} = b_{ip} = a_{ip}\cos\theta - a_{iq}\operatorname{sen}\theta$ si $i \neq p$ e $i \neq q$;
- $b_{qi} = b_{iq} = a_{ip}\operatorname{sen}\theta + a_{iq}\cos\theta$ si $i \neq p$ e $i \neq q$;
- $b_{pp} = a_{pp}\cos^2\theta + a_{qq}\operatorname{sen}^2\theta - a_{pq}\operatorname{sen} 2\theta$;
- $b_{qq} = a_{pp}\operatorname{sen}^2\theta + a_{qq}\cos^2\theta + a_{pq}\operatorname{sen} 2\theta$;
- $b_{pq} = b_{qp} = a_{pq}\cos 2\theta + \frac{a_{pp}-a_{qq}}{2}\operatorname{sin} 2\theta{=}0$;
- $b_{ij} = a_{ij}$ en el resto de los casos.

Resulta que las relaciones entre las fórmulas trigonométricas permiten describir los coeficientes de B a partir de los A sin necesidad de calcular explícitamente θ. En efecto, sea

$$x = \cot 2\theta = \frac{a_{qq} - a_{pp}}{2a_{pq}} \tag{6.4}$$

y escribamos $t = \operatorname{tg}\theta$. Entonces

$$x = \cot 2\theta = \frac{\cos^2\theta - \operatorname{sen}^2\theta}{2\operatorname{sen}\theta\cos\theta} = \frac{1 - \operatorname{tg}^2\theta}{2\operatorname{tg}\theta} = \frac{1-t^2}{2t},$$

y de aquí $t^2 + 2xt - 1 = 0$. Despejando t en función de x obtenemos $t = -x \pm \sqrt{x^2+1}$, y con el dato adicional de que $|t| \leq 1$ (porque $|\theta| \leq \pi/4$) podemos precisar más:

$$t = \begin{cases} -x + \sqrt{x^2+1} & \text{si } x \geq 0, \\ -x - \sqrt{x^2+1} & \text{si } x < 0. \end{cases} \tag{6.5}$$

De nuevo recordando que $|\theta| \leq \pi/4$ (con lo que $\cos\theta > 0$) y usando que $\mathrm{tg}^2\,\theta + 1 = 1/\cos^2\theta$, obtenemos

$$c = \frac{1}{\sqrt{t^2+1}} \tag{6.6}$$

y

$$s = \frac{t}{\sqrt{t^2+1}} \tag{6.7}$$

para $c = \cos\theta$ y $s = \mathrm{sen}\,\theta$. Ahora podemos encontrar expresiones muy convenientes para b_{pp} y b_{qq}:

$$\begin{aligned} b_{pp} &= c^2 a_{pp} + s^2 a_{qq} - 2sca_{pq} \\ &= a_{pp} + s^2(a_{qq} - a_{pp}) - 2sca_{pq} \\ &= a_{pp} + \frac{t^2}{t^2+1} x 2a_{pq} - \frac{2t}{t^2+1} a_{pq} \\ &= a_{pp} + \frac{t(1-t^2)}{t^2+1} a_{pq} - \frac{2t}{t^2+1} a_{pq} \\ &= a_{pp} - t a_{pq}; \end{aligned}$$

análogamente, $b_{qq} = a_{qq} + t a_{pq}$.

Ya estamos en condiciones de describir el método de Jacobi:

(I) Partiendo de $A_0 = A$ y $\Omega_0 = \mathrm{Id}$ se van construyendo una sucesión de matrices $A_k = O_k^T A_{k-1} O_k$ mediante cambios de base dados por rotaciones y las correspondientes matrices $\Omega_k = \Omega_{k-1} O_k$ de cambio de base, con lo que $A_k = \Omega_k^T A \Omega_k$.

(II) Supuesta construida $A_k = (a_{ij})$, se elige un término a_{pq} con $p < q$ para anular con una rotación; para ello seguiremos el llamado *criterio de Jacobi clásico*, que consiste tomar el término de mayor tamaño,

$$|a_{pq}| = \max\{|a_{ij}| : i < j\};$$

obsérvese que si $a_{pq} = 0$ entonces la matriz es diagonal y tras un número finito de pasos hemos obtenido la matriz Ω_k ortogonal cuyos vectores columna son la base de vectores propios buscada (con los correspondientes valores propios los términos en la diagonal de A_k).

(III) Se definen x, t, c y s con arreglo a las fórmulas (6.4)-(6.7) y se calculan los coeficientes la matriz $A_{k+1} = (b_{ij})$ de acuerdo con las fórmulas

- $b_{pi} = b_{ip} = ca_{ip} - sa_{iq}$ si $i \neq p$ e $i \neq q$,
- $b_{qi} = b_{iq} = sa_{ip} + ca_{iq}$ si $i \neq p$ e $i \neq q$,
- $b_{pp} = a_{pp} - ta_{pq}$,
- $b_{qq} = a_{qq} + ta_{pq}$,
- $b_{pq} = b_{qp} = 0$,
- $b_{ij} = a_{ij}$ en el resto de los casos.

(IV) Se calculan los coeficientes de la matriz $\Omega_{k+1} = (\omega_{ij})$ a partir de los coeficientes de $\Omega_k = (o_{ij})$ de acuerdo con las fórmulas

- $\omega_{ip} = co_{ip} - so_{iq}$,

- $\omega_{iq} = so_{ip} + co_{iq}$,
- $\omega_{ij} = o_{ij}$ si $j \neq p, q$.

(v) Como las sumas de los cuadrados de los coeficientes de las matrices A_k son siempre las mismas pero las sumas de los cuadrados de sus coeficientes diagonales son estrictamente crecientes, esperamos que la sucesión (A_k) converja a una matriz diagonal en la que encontraremos todos los valores propios de A, y que la sucesión de los productos de rotaciones Ω_k converja hacia una matriz ortogonal Ω cuyas columnas determinan una base de vectores propios de A.

Ejemplo 6.3.2 Aplicamos el primer paso del algoritmo de Jacobi a la matriz

$$A = \begin{pmatrix} 1 & -1 & 3 & 4 \\ -1 & 4 & 0 & -1 \\ 3 & 0 & 0 & -3 \\ 4 & -1 & -3 & 1 \end{pmatrix}.$$

En este caso $p = 1$ y $q = 4$, con $a_{14} = 4$. Entonces

$$x = \frac{a_{44} - a_{11}}{2a_{14}} = \frac{1-1}{8} = 0,$$

$$t = -x + \sqrt{x^2+1} = 1,$$

$$c = \frac{1}{\sqrt{t^2+1}} = \frac{\sqrt{2}}{2},$$

$$s = \frac{t}{\sqrt{t^2+1}} = \frac{\sqrt{2}}{2}.$$

Por tanto, si $A_1 = (b_{ij})$, tendremos que

$$\begin{aligned} b_{11} &= a_{11} - ta_{14} = 1 - 4 = -3, \\ b_{12} = b_{21} &= ca_{21} - sa_{24} = \frac{\sqrt{2}}{2}(-1) - \frac{\sqrt{2}}{2}(-1) = 0, \\ b_{13} = b_{31} &= ca_{31} - sa_{34} = \frac{\sqrt{2}}{2}(3) - \frac{\sqrt{2}}{2}(-3) = 3\sqrt{2}, \\ b_{14} = b_{41} &= 0, \end{aligned}$$

$$\begin{aligned} b_{22} &= a_{22} = 4, \\ b_{23} = b_{32} &= a_{32} = 0, \\ b_{24} = b_{42} &= sa_{21} + ca_{24} = \frac{\sqrt{2}}{2}(-1) + \frac{\sqrt{2}}{2}(-1) = -\sqrt{2}, \end{aligned}$$

$$\begin{aligned} b_{33} &= a_{33} = 0, \\ b_{34} = b_{43} &= sa_{31} + ca_{34} = \frac{\sqrt{2}}{2}(3) + \frac{\sqrt{2}}{2}(-3) = 0, \end{aligned}$$

$$b_{44} = a_{44} + ta_{14} = 1 + 4 = 5.$$

Así pues,

$$A_1 = \begin{pmatrix} -3 & 0 & 3\sqrt{2} & 0 \\ 0 & 4 & 0 & -\sqrt{2} \\ 3\sqrt{2} & 0 & 0 & 0 \\ 0 & -\sqrt{2} & 0 & 5 \end{pmatrix}.$$

Finalmente, en esta primera etapa la matriz Ω_1 es la propia matriz O_1, es decir,

$$\Omega_1 = \begin{pmatrix} c & 0 & 0 & s \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s & 0 & 0 & c \end{pmatrix} = \begin{pmatrix} \sqrt{2}/2 & 0 & 0 & \sqrt{2}/2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sqrt{2}/2 & 0 & 0 & \sqrt{2}/2 \end{pmatrix}.$$

A la hora de escribir un algoritmo con este método usaremos como condición de parada que los coeficientes de A_k fuera de la diagonal son suficientemente pequeños o bien que se realice un número excesivo de iteraciones. En el primer caso tendremos aproximados los valores propios y una base de vectores propios, mientras que en el segundo habremos parado para evitar entrar en un bucle infinito. De hecho probaremos enseguida que la sucesión (A_k) siempre converge a una matriz diagonal con los valores propios de A pero puede ocurrir (si existen valores propios repetidos) que las matrices (Ω_k) no converjan. Por ello, en el caso en que al detener el proceso veamos que aparecen valores propios repetidos (esto es, si sus aproximaciones son suficientemente parecidas) lanzaremos un mensaje de advertencia (que no de error) pues los vectores de Ω_k en el momento de la parada no tienen por qué aproximar a los vectores propios buscados (aunque en todo caso proporcionarán una base ortonomal).

Vale la pena subrayar que para la matriz

$$A = \begin{pmatrix} 1 & -1 & 3 & 4 \\ -1 & 4 & 0 & -1 \\ 3 & 0 & 0 & -3 \\ 4 & -1 & -3 & 1 \end{pmatrix}$$

anterior, el algoritmo proporciona en tan sólo 4 iteraciones los cuatro valores propios $\lambda_1 = -6$, $\lambda_2 = \lambda_3 = 3$ y $\lambda_4 = 6$ y la base ortonormal de vectores propios

$$\begin{aligned} v_1 &= (1, 0, -1, -1)/\sqrt{3}, \\ v_2 &= (1, 2, 0, 1)/\sqrt{6}, \\ v_3 &= (1, 0, 2, -1)/\sqrt{6}, \\ v_4 &= (1, -1, 0, 1)/\sqrt{3} \end{aligned}$$

con la máxima precisión de la máquina.

Algoritmo 6.3 **Valores y vectores propios (método de Jacobi)**

Datos de entrada:
$A[n][n]$ (matriz simétrica cuyos valores propios queremos obtener);
tol (precisión para la condición de parada);
nmax (número máximo de iteraciones);
Variables:
$O[n][n]$; // matriz auxiliar donde se guardan los vectores propios aproximados
$B[n][n]$; // matriz auxiliar donde se guardan los valores propios aproximados
Sol$[n][n+1]$; // matriz para devolver los resultados
p; q; x; t; c; s; aux; // variables auxiliares
Flujo del programa:
$B = A$;
$O =$ Id; // se inicializa O a la identidad
// Vamos a hacer las etapas $k = 1, 2, ...,$ nmax.

```
for(k=1;k<=nmax;k++){
  aux = 0;
  // Elección del coeficiente más grande.
  for(i=0;i<n;i++){
    for(j=i+1;j<n;j++){
      if(|B_ij| > aux){
        p = i;
        q = j;
        aux = |B_ij|;
      }
    }
  }
```

Algoritmo 6.3 **Valores y vectores propios (método de Jacobi), continuación**

```
if(aux < tol){
  for(i=0;i<n;i++){
    for(j=i+1;j<n;j++){
      if(|B_ii − B_jj| < tol){
        Advertencia: posible error en los vectores propios
      }
    }
  }
  for(i=0;i<n;i++){
    for(j=0;j<n;j++){
      Sol_ij = O_ji; // se escriben los vectores propios solución
    }
    Sol_in = B_ii; // se escriben los valores propios solución
  }
  Parada: los vectores y valores propios buscados están en Sol
}
// Cálculo de los coeficientes modificados de B.
x = (B_qq − B_pp)/(2 * B_pq);
if(x ≥ 0){
  t = −x + sqrt(1 + x * x);
}
else{
  t = −x − sqrt(1 + x * x);
}
c = 1/sqrt(1 + t * t);
s = t/sqrt(1 + t * t);
for(i=0;i<n;i++){
  aux = O_ip;
  O_ip = c * aux − s * O_iq;
  O_iq = s * aux + c * O_iq;
}
B_pp = B_pp − t * B_pq;
B_qq = B_qq + t * B_pq;
B_pq = B_qp = 0;
for(i=0;i<n;i++){
  if(i ≠ p, q){
    aux = B_ip;
    B_pi = B_ip = c * aux − s * B_iq;
    B_qi = B_iq = s * aux + c * B_iq;
  }
}
}
Parada: no hay convergencia en nmax iteraciones
```

Datos de salida: Valores y vectores propios de la matriz A, mensaje de error si la iteración no converge, mensaje de advertencia si se repiten valores propios.

El siguiente teorema garantiza la convergencia del método de Jacobi clásico. La clave de su prueba esta en este lema de topología:

Lema 6.3.3 *Si (x_k) es una sucesión acotada en un espacio vectorial normado de dimensión finita X sobre $\mathbb{R}$, (x_k) posee una cantidad finita de puntos de acumulación y $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$, entonces (x_k) es una sucesión convergente.*

Demostración:

Ideas que intervienen:

- *No es restrictivo suponer $X = \mathbb{R}^n$.*
- *Para los puntos de acumulación a_i de la sucesión (x_k) encontramos pequeñas bolas centradas en a_i y separadas a una distancia positiva unas de otras.*
- *Usando que toda sucesión acotada en $\mathbb{R}^n$ tiene un punto de acumulación vemos que a partir de un cierto índice todos los términos de la sucesión están en la unión de las bolas.*
- *Usamos la hipótesis $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$ para concluir que, de hecho, los términos de sucesión están todos en la misma bola.*

Es sabido que dos espacios vectoriales normados sobre $\mathbb{R}$ de la misma dimensión (finita) son homeomorfos. Por ejemplo, es sencillo demostrar que cualquier isomorfismo lineal es un homeomorfismo. Por tanto podemos suponer, sin pérdida de generalidad, que $X = \mathbb{R}^n$.

Sean a_i, $1 \leq i \leq M$, los puntos de acumulación de la sucesión (x_k). Entonces, para cada $\epsilon > 0$, existe un natural $l(\epsilon)$ tal que si $k \geq l(\epsilon)$ entonces

$$x_k \in \bigcup_{i=1}^{M} B(a_i, \epsilon)$$

(aquí, $B(a, \rho) = \{x \in \mathbb{R}^n : \|x - a\| < \rho\}$ es la bola abierta de centro a y radio ρ para la distancia inducida por la norma $\|\cdot\|$). La razón es que, en caso contrario, podríamos extraer una subsucesión (x_{k_m}) de (x_k) con la propiedad de que

$$x_{k_m} \notin \bigcup_{i=1}^{M} B(a_i, \epsilon)$$

para cada m. Pero sabemos que en $\mathbb{R}^n$ cada sucesión acotada, en particular (x_{k_m}), posee algún punto de acumulación. Este punto de acumulación, que también lo es de (x_k), no puede estar en ninguna de las bolas $B(a_i, \epsilon)$, en contradicción con el hecho de que los puntos a_i son los únicos puntos de acumulación de (x_k).

Elijamos en particular

$$\epsilon_0 = \frac{1}{3} \min_{i \neq i'} \|a_i - a_{i'}\| > 0$$

y encontremos l_0 tal que

$$x_k \in \bigcup_{i=1}^{M} B(a_i, \epsilon_0), \qquad \|x_{k+1} - x_k\| < \epsilon_0$$

para cada $k \geq l_0$ (también hemos usado la hipótesis $\lim_{k\to\infty} \|x_k - x_{k+1}\| = 0$).

Sea i_0 tal que $x_{l_0} \in B(a_{i_0}, \epsilon_0)$. Como $\|x_{l_0} - a_{i_0}\| < \epsilon_0$ y $\|x_{l_0+1} - x_{l_0}\| < \epsilon_0$, obtenemos $\|x_{l_0+1} - a_{i_0}\| < 2\epsilon_0$ por la desigualdad triangular y por tanto, gracias de nuevo a la desigualdad triangular y a la definición de ϵ_0, $\|x_{l_0+1} - a_i\| > \epsilon_0$ para cada $i \neq i_0$. Esto significa que x_{l_0+1} no puede pertenece a ninguna de las bolas $B(a_i, \epsilon_0)$ excepto tal vez a la bola $B(a_{i_0}, \epsilon_0)$. Como de hecho x_{l_0+1} ha de estar en alguna de las bolas, la conclusión es que $x_{l_0+1} \in B(a_{i_0}, \epsilon_0)$. Reiterando el razonamiento, concluimos que $x_k \in B(a_{i_0}, \epsilon_0)$ y $x_k \notin B(a_i, \epsilon_0)$ si $i \neq i_0$ para cada $k \geq l_0$.

Vemos, pues, que ninguno de los puntos a_i, $i \neq i_0$, puede ser punto de acumulación de (x_k), es decir, a_{i_0} es el único punto de acumulación de (x_k). Esto equivale a decir que $\lim_{k\to\infty} x_k = a_{i_0}$. □

Teorema 6.3.4 (convergencia del método de Jacobi clásico) *La sucesión (A_k) de matrices obtenidas por el método de Jacobi clásico para una matriz simétrica A es convergente hacia una matriz diagonal,*

$$diag(\lambda_{\sigma(1)}, \ldots, \lambda_{\sigma(n)}),$$

en la que aparecen los valores propios $\lambda_1, \ldots, \lambda_n$ de A convenientemente permutados. Si suponemos además que los valores propios de la matriz A son distintos entre sí, la sucesión de matrices (Ω_k) definida por

$$\Omega_k = O_1 O_2 \cdots O_k$$

converge hacia una matriz ortogonal cuyas columnas determinan una base de vectores propios de A con valores propios los correspondientes en la anterior matriz diagonal.

Demostración:

Demostramos la primera parte del teorema. Para ello seguiremos el siguiente esquema:

Ideas que intervienen:

- *Si $A_k = (a_{ij}^k) = D_k + B_k$, con $D_k = diag(a_{11}^k, \ldots, a_{nn}^k)$, probamos que $\lim_{k\to\infty} B_k = 0$.*
- *Demostramos que (D_k) tiene a lo sumo un número finito de puntos de acumulación, cada uno de ellos de la forma $diag(\lambda_{\sigma(1)}, \ldots, \lambda_{\sigma(n)})$ para una cierta permutación σ de los índices $\{1, 2, \ldots, n\}$.*
- *Probamos que $\lim_{k\to\infty} D_{k+1} - D_k = 0$.*
- *Comprobamos que la sucesión (D_k) está acotada.*
- *Aplicamos el Lema 6.3.3.*

(i) Consideremos los números

$$\epsilon_k = \sum_{i\neq j} |a_{ij}^k|^2 = \|B_k\|_{\text{DOS}}^2, \quad k \geq 0.$$

Como ϵ_k es la suma de los cuadrados de los coeficientes de la matriz A_k fuera de la diagonal principal, sabemos (Lema 6.3.1) que si que $p < q$ son los índices elegidos para generar A_{k+1} a partir de A_k con el método de Jacobi, entonces

$$\epsilon_{k+1} = \epsilon_k - 2|a_{pq}^k|^2.$$

Por otro lado, y dado que $|a_{pq}| = \max_{i\neq j} |a_{ij}|$, resulta

$$\epsilon_k \leq n(n-1)|a_{pq}^k|^2$$

(pues el número de coeficientes fuera de la diagonal principal es $n(n-1)$). Combinando ambas relaciones obtenemos

$$\epsilon_{k+1} \leq \left(1 - \frac{2}{n(n-1)}\right) \epsilon_k,$$

de donde se deduce $\lim_{k\to\infty} \epsilon_k = 0$ y por tanto $\lim_{k\to\infty} B_k = 0$.

(ii) Supongamos que una cierta subsucesión (D_{k_m}) de (D_k) converge a una cierta matriz D. Entonces la matriz D será diagonal. Por otro lado, como $B_k \to 0$ por (i) también será cierto $B_{k_m} \to 0$, así que (A_{k_m}) converge también a D. Por continuidad tenemos que, para cada número $\lambda \in \mathbb{R}$ fijado,

$$\lim_{m\to\infty} \det(\lambda \operatorname{Id} - A_{k_m}) = \det(\lambda \operatorname{Id} - D).$$

Como las matrices A_k y A son semejantes, también lo serán las matrices $\lambda \operatorname{Id} - A_k$ y $\lambda \operatorname{Id} - A$ para cada $\lambda \in \mathbb{R}$, con lo que deducimos que sus determinantes son los mismos y por tanto

$$\det(\lambda \operatorname{Id} - A) = \det(\lambda \operatorname{Id} - D)$$

para cada $\lambda \in \mathbb{R}$.

Estamos diciendo que polinomios característicos de A y D toman los mismos valores para cada valor de la incógnita. Como es sabido, si dos polinomios de grado n toman los mismos valores en $n+1$ puntos distintos entonces ambos polinomios son iguales. La conclusión que es los polinomios característicos de A y D (y por tanto sus valores propios, incluidas multiplicidades) son los mismos. Como la matriz D es diagonal, los elementos de su diagonal serán los valores propios de A, con las mismas multiplicidades, adecuadamente permutados. Hemos probado que (D_k) tiene a lo sumo un número finito de puntos de acumulación, cada uno de ellos de la forma $\operatorname{diag}(\lambda_{\sigma(1)}, \ldots, \lambda_{\sigma(n)})$ para una cierta permutación σ de los índices $\{1, 2, \ldots, n\}$.

(iii) Recordemos que

$$a_{ii}^{k+1} - a_{ii}^k = \begin{cases} 0 & \text{si } i \neq p, q, \\ -\operatorname{tg}\theta_k a_{pq}^k & \text{si } i = p, \\ \operatorname{tg}\theta_k a_{pq}^k & \text{si } i = q, \end{cases}$$

donde $\theta_k \in (-\pi/4, 0) \cup (0, \pi/4]$. Como $|\operatorname{tg}\theta_k| \leq 1$ y $|a_{pq}^k| \leq \|B_k\|_{\text{DOS}}$ para cada k, y además se tiene $\lim_{k\to\infty} B_k = 0$ por (i), concluimos que $\lim_{k\to\infty}(D_{k+1} - D_k) = 0$.

(iv) Observemos que

$$\|D_k\|_{\text{DOS}} \leq \|A_k\|_{\text{DOS}} = \|A\|_{\text{DOS}}$$

por el Lema 6.3.1(i). Por tanto la sucesión (D_k) está acotada.

Ya estamos listos para probar la primera parte del Teorema 6.3.4. De acuerdo con el Lema 6.3.3 y (ii), (iii) y (iv), la sucesión (D_k) converge a una matriz diagonal $\operatorname{diag}(\lambda_{\sigma(1)}, \ldots, \lambda_{\sigma(n)})$. Por (i), la sucesión (A_k) converge a la misma matriz.

Probamos ahora la segunda parte del teorema conforme al siguiente esquema:

Ideas que intervienen:

- *Demostramos que la sucesión (Ω_k) sólo tiene un número finito de puntos de acumulación, que son necesariamente de la forma $(\pm p_{\sigma(1)}, \pm p_{\sigma(2)}, \ldots, \pm p_{\sigma(n)})$, siendo $p_1, p_2, \ldots, p_n$ una base ortonormal de vectores propios de valores propios correspondientes $\lambda_1, \lambda_2, \ldots, \lambda_n$.*
- *Probamos que $\lim_{k\to\infty}(\Omega_{k+1} - \Omega_k) = 0$.*
- *Se demuestra que las matrices ortogonales están acotadas para la norma $\|\cdot\|_{DOS}$.*
- *Se concluye la prueba usando el Lemma 6.3.3.*

(i) Sea Ω un punto de acumulación de la sucesión (Ω_k) y sea (Ω_{k_m}) una subsucesión de (Ω_k) convergente a Ω. Entonces la sucesión de las matrices traspuestas $(\Omega_{k_m}^T)$ convergerá a Ω^T. Como $\Omega_k^T\Omega_k = \mathrm{Id}$ para cada k y el producto de las sucesiones $(\Omega_{k_m}^T)$ y (Ω_{k_m}) converge por continuidad a $\Omega^T\Omega$, concluimos que $\Omega^T\Omega = \mathrm{Id}$, es decir, la matriz Ω es ortogonal.

Recordemos que por la primera parte del teorema las matrices $A_k = \Omega_k^T A \Omega_k$ convergen a una matriz diagonal $\mathrm{diag}(\lambda_{\sigma(1)}, \ldots, \lambda_{\sigma(n)})$. Tomando límites en la subsucesión $(\Omega_{k_m}^T A \Omega_{k_m})$ llegamos a que

$$\Omega^T A \Omega = \mathrm{diag}(\lambda_{\sigma(1)}, \ldots, \lambda_{\sigma_(n)}).$$

La igualdad implica que los vectores columna de Ω forman una base ortonormal de vectores propios con valores propios respectivos los números $\lambda_{\sigma(i)}$. Como por hipótesis todos los valores propios son distintos, cada subespacio propio tiene dimensión 1 y para cada valor propio λ existen exactamente dos vectores propios (uno opuesto del otro) de norma 1. Hemos probado que Ω es de la forma $(\pm p_{\sigma(1)}, \pm p_{\sigma(2)}, \ldots, \pm p_{\sigma(n)})$.

(ii) Recordemos que los ángulos θ_k son tales que $|\theta_k| \leq \pi/4$ y satisfacen la igualdad

$$\mathrm{tg}\, 2\theta_k = \frac{2a_{pq}^k}{a_{qq}^k - a_{pp}^k}$$

para cada k. Enfatizamos que los índices p y q dependen de k, pero para simplificar la notación no hacemos explícita esta dependencia. Como los coeficientes a_{ii}^k de la diagonal principal de las matrices A_k convergen a los valores propios $\lambda_{\sigma(i)}$ y dichos valores propios son distintos dos a dos, existirá un número l suficientemente alto tal que si $k \geq l$ entonces

$$\min_{i\neq j} |a_{ii}^k - a_{jj}^k| \geq \frac{1}{2} \min_{i\neq j} |\lambda_i - \lambda_j| =: M > 0;$$

en particular, $|a_{qq}^k - a_{pp}^k| \geq M$ para cada $k \geq l$. Como todos los coeficientes fuera de la diagonal principal de A_k tienden a cero, la sucesión (a_{pq}^k) tiende a cero (aunque los índices p y q vayan variando con k). Así pues, $(\mathrm{tg}\, 2\theta_k)$ tiende a cero e igualmente (θ_k) tiende a cero. En resumen, hemos probado que

$$\lim_{k\to\infty} O_k = \mathrm{Id}.$$

Dado que $\Omega_{k+1} - \Omega_k = \Omega_k(O_{k+1} - \mathrm{Id})$, se deduce

$$\lim_{k\to\infty} (\Omega_{k+1} - \Omega_k) = 0.$$

(iii) En el Capítulo 3 (Teorema 3.3.1) se vio que la norma $\|\cdot\|_2$ de todas las matrices ortogonales es 1. Como en $\mathcal{M}_{n\times n}(\mathbb{R})$ (que puede verse como el espacio euclídeo $\mathbb{R}^{n^2}$) todas las normas son

equivalentes, existe en particular una cierta constante $\beta > 0$ tal que $\|B\|_{\mathrm{DOS}} \leq \beta \|B\|_2$ para cada $B \in \mathcal{M}_{n\times n}(\mathbb{R})$. De esto se deduce que $\|O\|_{\mathrm{DOS}} \leq \beta$ para cada matriz ortogonal O.

Finalmente, la segunda parte del Teorema 6.3.4 se deduce del Lema 6.3.3 y (i), (ii) y (iii). $\square$

Bibliografía

[1] R. L. Burden y J. D. Faires, *Análisis Numérico*, Grupo Editorial Iberoamérica, México, 1985.

[2] P. G. Ciarlet, *Introduction à l'Analyse Numérique Matricielle et à l'Optimisation*, Dunod, París, 1990.

[3] D. Kincaid y W. Cheney, *Análisis Numérico. Las Matemáticas del Cálculo Científico*, Addison-Wesley Sudamericana, Wilmington, 1994.

Capítulo 7

Ceros de polinomios

Interrogantes centrales del capítulo

- Evaluación de polinomios. Esquema de Horner.
- Ceros de polinomios. Teorema fundamental del Álgebra.
- Localización de ceros de polinomios.
- Métodos numéricos para aproximación de ceros:
  - Método de Newton para polinomios.
  - Iteración de Laguerre.
  - El método de separación de Sturm.

Destrezas a adquirir en el capítulo

- Describir los algoritmos correspondientes a los distintos métodos de aproximación de ceros y factorización de polinomios.
- Comparar la convergencia de las aproximaciones proporcionadas por cada método.
- Implementar en el ordenador los programas de los métodos de este capítulo utilizando variables complejas en los algoritmos.
- Aplicar los programas construidos de forma efectiva en la búsqueda de todas las raíces reales y complejas de un polinomio concreto.

Desarrollo de los contenidos fundamentales

Un caso particular de ecuaciones para las que es muy importante y útil la determinación de sus raíces es el de las ecuaciones polinómicas. En este capítulo vamos a ampliar la lista de métodos de resolución de ecuaciones no lineales aprendidos en un capítulo anterior incluyendo otros que resultan de aplicar propiedades específicas de los polinomios.

En la primera sección repasamos el algoritmo de Horner de evaluación y deflación de polinomios, que resulta más económico desde el punto de vista numérico que la evaluación directa de combinaciones lineales de potencias, y ahondamos en sus propiedades.

En la segunda sección recordamos las propiedades de los ceros de un polinomio, haciendo hincapié en el teorema fundamental del Álgebra y en la factorización de polinomios.

En la tercera sección exponemos resultados de localización de ceros de polinomios útiles en la determinación de condiciones iniciales de los métodos numéricos de cálculo de raíces.

Concluimos estudiando y analizando distintos algoritmos numéricos de aproximación de raíces de polinomios.

Hemos seguido la Sección 3.5 (pp. 91–111) del libro de Kincaid y Cheney [3] para redactar la mayor parte de este capítulo.

7.1. Polinomios complejos. El algoritmo de Horner

Recordemos que un *polinomio complejo* es una combinación lineal de potencias naturales de la variable z,

$$p(z) = a_0 + a_1 z + \cdots + a_n z^n,$$

donde los coeficientes $a_k \in \mathbb{C}$ son números complejos y $a_n \neq 0$. En estas condiciones se dice que el polinomio $p(z)$ tiene *grado* $n \in \mathbb{N}$. Se adopta la convención de que el grado del polinomio constante cero es $-\infty$.

Como ya explicamos, para calcular el valor del polinomio p en un punto z_0 lo más rentable desde el punto de vista computacional es usar el *esquema de Horner* (popularmente conocido como *regla de Rufini*)

$$p(z_0) = (\ldots((a_n z_0 + a_{n-1})\, z_0 + a_{n-2})\, z_0 + \cdots + a_1)\, z_0 + a_0.$$

Los multiplicadores de z_0 en esta fórmula se van obteniendo de forma recursiva siguiendo el esquema

	a_n	a_{n-1}	a_{n-2}	$\ldots\ldots$	a_1	a_0
z_0		$z_0 b_n$	$z_0 b_{n-1}$	$\ldots\ldots$	$z_0 b_2$	$z_0 b_1$
	b_n	b_{n-1}	b_{n-2}	$\ldots\ldots$	b_1	b_0

con $b_n = a_n$ y $b_k = b_{k+1} z_0 + a_k$ para $k = (n-1), \ldots, 1, 0$. Los coeficientes b_k también determinan el cociente $c(z)$ de la división de $p(z)$ entre $z - z_0$.

En efecto, recuérdese que si $p(z)$ y $q(z)$ son polinomios dados entonces existen polinomios $c(z)$ y $r(z)$ (con el grado de $r(z)$ estrictamente menor que el de $q(z)$) de manera que

$$p(z) = c(z)q(z) + r(z).$$

Los polinomios $c(z)$ y $r(z)$ (el *cociente* y el *resto* de la división de $p(z)$ entre $q(z)$) están unívocamente determinados. Si $z_0 \in z$ y tomamos $q(z) = z - z_0$ entonces $r(z)$ debe ser una constante $r_0 \in \mathbb{C}$, y sustituyendo z por z_0 en la igualdad $p(z) = c(z)(z - z_0) + r_0$ obtenemos $z_0 = p(z_0)$, es decir,

$$c(z) = q(z)(z - z_0) + p(z_0). \tag{7.1}$$

Definamos

$$c(z) = b_n z^{n-1} + b_{n-1} z^{n-2} + \cdots + b_2 z + b_1,$$

con los coeficientes b_k como en el algoritmo de Horner. Entonces

$$c(z)(z - z_0) = b_n z^n + b_{n-1} z^{n-1} + \cdots + b_2 z^2 + b_1 z - \left(b_n z^{n-1} z_0 + \cdots + b_2 z_0 + b_1 z_0\right),$$

$$c(z)(z - z_0) + b_0 = b_n z^n + (b_{n-1} - b_n z_0) z^{n-1} + \cdots + (b_1 - b_2) z + (b_0 - b_1 z_0).$$

De acuerdo con la definición de los coeficientes b_k resulta $b_n = a_n$ y $b_k - b_{k+1} z_0 = a_k$, de donde se obtiene la igualdad (7.1).

Recordamos a continuación el pseudocódigo del algoritmo de Horner, enfatizando que si sólo necesitamos obtener el valor de $p(z)$ podemos acelerar el algoritmo utilizando una única variable b en lugar de una lista b_k (se inicia b con el valor de a_n y en el bucle `for` se hace la asignación $b = bz + a_k$).

Algoritmo 7.1 **Esquema de Horner**

Datos de entrada:
$a_0, a_1, \ldots, a_n$ (coeficientes del polinomio p);
z_0 (número donde se evalúa el polinomio);
Flujo del programa:
$b_n = a_n$;
`for(k=n-1;k>=0;k--){`
  $b_k = b_{k+1} z_0 + a_k$;
`}`
Datos de salida: b_0 (valor de $p(z_0)$) y $b_1, \ldots, b_n$ (coeficientes de $q(z)$).

Ejemplo 7.1.1 Evaluemos el polinomio $p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$ en $z = 2$. Entonces

	1	−4	7	−5	−2
2		2	−4	6	2
	$\mathbf{b_4 = 1}$	$\mathbf{b_3 = -2}$	$\mathbf{b_2 = 3}$	$\mathbf{b_1 = 1}$	$\mathbf{b_0 = 0}$

con lo que $p(2) = 0$ y $p(z) = (z - 2)(\mathbf{1}z^3 - \mathbf{2}z^2 + \mathbf{3}z + \mathbf{1})$.

Al derivar en la expresión (7.1) se tiene que $p'(z) = c'(z)(z - z_0) + c(z)$ y por lo tanto $p'(z_0) = c(z_0)$. Así, para evaluar la derivada $p'(z_0)$ bastará con aplicar el esquema de Horner a $c(z)$ en el punto z_0. Nótese que estamos usando aquí derivación compleja: la definición es la misma que la derivada usual, sólo que trabajando en $\mathbb{C}$ en lugar de $\mathbb{R}$. La demostración de las reglas de derivación usuales no queda afectada por este hecho.

El esquema de Horner aplicado a $c(z)$ nos proporciona $c(z_0)$ y el polinomio cociente $m(z)$ de $c(z)$ entre $(z - z_0)$, que sustituido en la factorización de p nos da la expresión

$$p(z) = (m(z)(z - z_0) + c(z_0))(z - z_0) + p(z_0) = m(z)(z - z_0)^2 + c(z_0)(z - z_0) + p(z_0).$$

Al hacer la segunda derivada obtenemos que $p''(z_0) = 2!\, m(z_0)$.

Si repetimos la aplicación de Horner vamos obteniendo las distintas derivadas $p^{(k)}(z_0)$.

También podemos expresar el desarrollo de Taylor de un polinomio en el punto z_0 utilizando el método de Horner para evaluar los coeficientes $c_k = \frac{p^{(k)}(z_0)}{k!}$ en la expresión

$$p(z) = c_n(z - z_0)^n + c_{n-1}z^{n-1} + \cdots + c_1(z - z_0) + c_0.$$

Veámoslo en el siguiente ejemplo.

Ejemplo 7.1.2 Desarrollamos a continuación el polinomio $p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$ en potencias de $(z - 3)$:

	1	−4	7	−5	−2
3		3	−3	12	21
	1	−1	4	7	$\mathbf{c_0 = 19}$
3		3	6	30	
	1	2	10	$\mathbf{c_1 = 37}$	
3		3	15		
	1	5	$\mathbf{c_2 = 25}$		
3		3			
	$\mathbf{c_4 = 1}$	$\mathbf{c_3 = 8}$			

Así pues, $p(z) = \mathbf{1}(z - 3)^4 + \mathbf{8}(z - 3)^3 + \mathbf{25}(z - 3)^2 + \mathbf{37}(z - 3) + \mathbf{19}$.

En las siguientes secciones, utilizaremos el esquema de Horner para evaluar simultáneamente los polinomios y sus derivadas en las implementaciones de los distintos métodos numéricos de localización de raíces.

7.2. Ceros de polinomios. El teorema fundamental del Álgebra

Si $p(z_0) = 0$ se dice que z_0 es un *cero del polinomio* o una *raíz* de $p(z)$. La fórmula (7.1) nos da la factorización $p(z) = (z - z_0)c(z)$. A la obtención del polinomio $c(z)$ se le llama *deflación* de $p(z)$ en z_0 (ver el Ejemplo 7.1.1).

Si $c(z_0) = 0$ podemos factorizar c obteniendo $c(z) = (z - z_0)c_1(z)$ y $p(z) = (z - z_0)^2 c_1(z)$. Así seguiremos hasta llegar a una factorización de p de la forma

$$p(z) = (z - z_0)^{k_0} c_0(z),$$

donde $c_0(z)$ es un polinomio con $c_0(z_0) \neq 0$. El número k_0 es la *multiplicidad* de la raíz z_0 de p.

Antes de buscar ceros de polinomios debemos preguntarnos sobre su existencia. El *teorema fundamental del Álgebra* nos responde a esta cuestión:

Teorema 7.2.1 (teorema fundamental del Álgebra; Gauss, 1799) *Todo polinomio no constante con coeficientes complejos tiene al menos una raíz compleja.*

DEMOSTRACIÓN: Aunque el enunciado es puramente algebraico todas las demostraciones asequibles que se conocen hacen uso de recursos del Análisis Matemático. Vamos a exponer esquemáticamente una de estas pruebas. Puede encontrarse el desarrollo detallado de la misma en el Teorema 2.2.7 del libro *Análisis Matemático II* de J. A. Fernández Viña [2].

Ideas que intervienen:

- *Sea $p(z)$ un polinomio no constante de grado n, digamos*

$$p(z) = a_0 + a_1 z + \cdots + a_n z^n$$

  con $a_n \neq 0$.

- *Comparando $p(z)$ con el sumando $a_n z^n$ podemos afirmar que*

$$\lim_{|z|\to\infty} |p(z)| = \lim_{|z|\to\infty} |a_n||z|^n = \infty.$$

- *Por la afirmación anterior, como $|p(z)|$ es una función continua en $\mathbb{C}$ podemos concluir que la función continua $|p(z)|$ alcanza su valor mínimo en un punto $z_0 \in \mathbb{C}$.*

- *Para ver que z_0 es una raíz de p, es decir $p(z_0) = 0$, razonamos por reducción al absurdo:* supondremos que $|p(z_0)| > 0$ y llegaremos a una contradicción.

- *Si $|p(z_0)| > 0$ vamos a elegir una recta que pasa por z_0 de manera que $|p(z)|$ restringida a esa recta tome valores estrictamente menores que $|p(z_0)|$. En consecuencia, $|p(z)|$ no tiene un mínimo en z_0, en contradicción con la elección de z_0.*

Si $k = k_0$ es la multiplicidad de z_0 como raíz de $p(z) - p(z_0)$ podemos escribir

$$p(z) = p(z_0) + (z - z_0)^k c(z) = p(z_0)(1 + (z - z_0)^k \tilde{c}(z)),$$

donde $\tilde{c}$ es un polinomio de grado $n-k$ y $b_0 = \tilde{c}(z_0) \neq 0$. Escribimos $z - z_0 = re^{it}$ en coordenadas polares y $\tilde{c}(z)$ en la forma $\tilde{c}(z) = b_0 + (z - z_0)h(z)$.

Si $b_0 = |b_0|e^{is}$, fijamos la dirección correspondiente a $t = \frac{1}{k}(\pi - s)$ y tenemos

$$p(z) = p(z_0)(1 - r^k|b_0| + r^{k+1}e^{i(k+1)t}h(z)).$$

Tomando r pequeño, $r^k|b_0| < 1$ y la desigualdad triangular nos da

$$|p(z)| \leq |p(z_0)|((1 - r^k|b_0|) + r^{k+1}|h(z)|) = |p(z_0)|(1 - r^k(|b_0| - r|h(z)|)).$$

Como $r|h(z)| \to 0$ cuando $r \to 0$, si r es suficientemente pequeño se tendrá

$$|b_0| - r|h(z)| > 0{,}5|b_0|$$

y por lo tanto

$$|p(z)| \leq |p(z_0)|(1 - r^k(|b_0| - r|h(z)|)) < |p(z_0)|(1 - 0{,}5r^k|b_0|) < |p(z_0)|.$$

Esta última cadena de desigualdades nos da la contradicción esperada. □

Factorizando el polinomio $p(z)$ en z_0 tenemos $p(z) = (z - z_0)^{k_0} q(z)$. Si $q(z)$ no es constante podemos aplicar otra vez el teorema fundamental del Álgebra para garantizar la existencia de un cero z_1 de q que también será un cero de $p(z)$. Repitiendo el proceso obtenemos el siguiente teorema de factorización.

Teorema 7.2.2 (factorización de polinomios) *Si $p(z) = a_0 + a_1 z + \cdots + a_n z^n$ es un polinomio de grado n con coeficientes en $\mathbb{C}$, existen $s+1$ números complejos distintos $z_0, z_1, \ldots, z_s$, con $s + 1 \leq n$, y enteros positivos $k_0, \ldots, k_s$, verificando $k_0 + \cdots + k_s = n$ y*

$$p(z) = a_n (z - z_0)^{k_0} (z - z_1)^{k_1} \cdots (z - z_s)^{k_s}.$$

De la factorización anterior se deduce que un polinomio de grado n tiene, a lo más, n raíces distintas. Repitiendo las raíces según su multiplicidad, un polinomio de grado n tiene exactamente n raíces. También se sigue que dos polinomios $p(z)$ y $q(z)$de grado menor o igual a n que coinciden en $n+1$ números complejos distintos ($p(z_k) = q(z_k)$ con $z_1, \ldots, z_{n+1}$ distintos) deben ser iguales ($p(z) = q(z)$ en todo z).

Una observación a tener en cuenta es que un polinomio puede tener todos sus coeficientes reales y sus raíces no tienen porque ser reales. Por ejemplo, en el polinomio

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

los coeficientes son números reales y las raíces i y $-i$ no lo son. En estos casos, si $z_0 \in \mathbb{C}$ es una raíz del polinomio de coeficientes reales $p(z) = a_0 + a_1 z + \cdots + a_n z^n$ tomando conjugados en la expresión $p(z_0) = 0$ se deduce que $\overline{z_0}$ también es raíz de $p(z)$ ya que

$$p(\overline{z_0}) = a_0 + a_1 \overline{z_0} + \cdots + a_n \overline{z_0}^n = \overline{a_0 + a_1 z_0 + \cdots + a_n z_0^n} = \overline{p(z_0)} = 0.$$

Así, cada vez que obtenemos una raíz compleja también obtenemos su conjugada, es decir, si $z - (a + bi)$ es un factor de $p(z)$ también lo será $z - (a - bi)$ y por tanto igualmente lo será su producto

$$(z - (a + bi))(z - (a - bi)) = (z - a)^2 + b^2 = z^2 - 2az + a^2 - b^2.$$

En otras palabras, todo polinomio con coeficientes reales puede descomponerse como producto de monomios y polinomios de grado 2 con coeficientes reales.

A la hora de construir algoritmos de búsqueda de raíces de polinomios deberemos tener presente la conveniencia de programarlos utilizando variables complejas (en $\mathbb{C}$) aunque estemos pensando en utilizarlos con polinomios reales.

7.3. Localización de ceros

Para aplicar alguno de los métodos que enseguida describiremos para aproximar raíces es conveniente disponer de buenas aproximaciones iniciales de las mismas. En esta sección vamos a *localizar* regiones del plano complejo en las que tenemos la certeza de que podemos encontrar raíces de un polinomio, lo que será útil para elegir nuestras aproximaciones iniciales.

Teorema 7.3.1 *Todas las raíces del polinomio $p(z) = a_0 + a_1 z + \cdots + a_n z^n$ de grado n están en el disco de centro 0 y radio*

$$\rho = 1 + \frac{1}{|a_n|} \max\{|a_0|, \ldots, |a_{n-1}|\}.$$

Demostración:

Ideas que intervienen:

- *Vamos a ver que para $|z| > \rho$ se tiene $|p(z)| > 0$. Así, todas las raíces de $p(z)$ están dentro del disco de centro 0 y radio ρ. Sólo usamos la desigualdad triangular.*

Sean $M = \text{máx}\{|a_0|, |a_1|, \ldots, |a_{n-1}|\}$ y ρ como en el enunciado. Entonces $\rho - 1 = \frac{M}{|a_n|}$. Por tanto, si $|z| > \rho$ se tiene que

$$|a_n|(|z| - 1) > |a_n|(\rho - 1) = |a_n|\frac{M}{|a_n|} = M.$$

Ahora es suficiente hacer uno de la desigualdad triangular en la siguiente cadena de desigualdades:

$$\begin{aligned}
|p(z)| &= |\left(a_0 + a_1 z + \cdots + a_{n-1}z^{n-1}\right) + a_n z^n| \\
&\geq |a_n||z|^n - \left(|a_0| + |a_1||z| + \cdots + |a_{n-1}||z|^{n-1}\right) \\
&\geq |a_n||z|^n - M\left(1 + |z| + \cdots + |z|^{n-1}\right) \\
&= |a_n||z|^n - M\frac{|z|^n - 1}{|z| - 1} \\
&\geq |a_n||z|^n - M\frac{|z|^n}{|z| - 1} \\
&= |z|^n\left(|a_n| - M\frac{1}{|z| - 1}\right) \\
&= \frac{|z|^n}{|z| - 1}\left(\ |a_n|(|z| - 1) - M\ \right) \\
&> \frac{|z|^n}{|z| - 1}(M - M) = 0
\end{aligned}$$

□

Ejemplo 7.3.2 Los ceros del polinomio $p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$ están en el circulo de centro 0 y radio $\rho = 1 + \text{máx}\{2, 5, 7, 4\} = 1 + 7 = 8$. En la Figura 7.1 representamos gráficamente

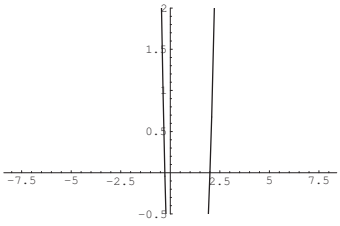


Figura 7.1: Localización de las raíces reales de $p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$

los valores del polinomio real $|p(x)|$ comprendidos entre $-0{,}5$ y 2 cuando x varia en el intervalo real $[-8, 8]$ y observamos que el polinomio sólo tiene dos raíces reales en los cortes de la gráfica con el eje x.

Por otra parte en la Figura 7.2 representamos las curvas de nivel de la función $|p(z)|$ en el rectángulo $[-1, 4] \times [-2{,}5, 2{,}5]$ del plano complejo, contenido en el disco con centro en el origen y radio 8, y visualizamos la posición de las cuatro raíces (las dos reales y las dos complejas conjugadas).

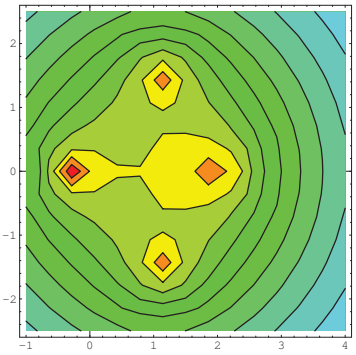


Figura 7.2: Localización de las raíces reales y complejas de $p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$

El teorema de localización 7.3.1 nos da una acotación superior del módulo de los ceros de un polinomio. ¿Es posible tener también una acotación inferior?

La respuesta a esta cuestión es afirmativa y para conseguir encontrar una acotación inferior vamos a recurrir al habitual truco de invertir la variable, es decir, hacer el cambio $z = \frac{1}{w}$ que transforma números z pequeños en números w grandes y viceversa.

Pero antes de hacer el cambio de variable vamos a detenernos un instante en el caso en que la acotación inferior es 0, es decir, cuando $z_0 = 0$ es una raíz del polinomio. En este caso no necesitamos recurrir a ningún algoritmo particular para descubrir este cero pues el polinomio tiene una expresión de la forma

$$p(z) = a_k z^k + a_{k+1} z^{k+1} + \cdots + a_n z^n = z^k \left(a_k + \cdots + a_n z^{n-k} \right)$$

con $a_k \neq 0$. De la expresión deducimos sin más que 0 es una raíz de p de multiplicidad k y el problema de búsqueda de raíces se reduce al caso de polinomios que no se anulan en 0.

Supongamos que $p(z) = a_0 + a_1 z + \cdots + a_n z^n$ es un polinomio de grado n que no se anula en $z = 0$, y por lo tanto,con $a_0 \neq 0$ y $a_n \neq 0$.

Haciendo $z = \frac{1}{w}$ tenemos $p(z) = p(\frac{1}{w}) = \frac{1}{w^n} \left(a_0 w^n + a_1 w^{n-1} + \cdots + a_{n-1} w + a_n \right)$. Como $\frac{1}{w} \neq 0$, resulta que $p(z) = 0$ si, y sólo si, $q(w) = a_0 w^n + a_1 w^{n-1} + \cdots + a_{n-1} w + a_n = 0$. Ahora hacemos uso del teorema de localización y podemos asegurar que $p(z) = 0$ sólo cuando $|w| = \frac{1}{|z|} \leq \rho'$ ($|z| \geq \frac{1}{\rho'}$), donde

$$\rho' = 1 + \frac{1}{|a_0|} \operatorname{máx}\{|a_1|, \ldots, |a_n|\}.$$

Volviendo al Ejemplo 7.3.2, para $p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$, $\rho' = 1 + \frac{7}{2} = \frac{9}{2}$ y podemos asegurar que las raíces de p están en el anillo $A = \{z : \frac{2}{9} \leq |z| \leq 8\}$.

Reuniendo la observación con el Teorema 7.3.1, tenemos el siguiente resultado:

Corolario 7.3.3 *Sea $p(z) = a_0 + a_1 z + \cdots + a_n z^n$ un polinomio de grado n tal que $a_0 \neq 0$ y $a_n \neq 0$, y sean*

$$\rho = 1 + \frac{1}{|a_n|} \operatorname{máx}\{|a_0|, \ldots, |a_{n-1}|\}, \quad \rho' = 1 + \frac{1}{|a_0|} \operatorname{máx}\{|a_1|, \ldots, |a_n|\}.$$

Entonces, los ceros de $p(z)$ están en el anillo $A = \{z : \frac{1}{\rho'} \leq |z| \leq \rho\}$.

7.4. El método de Newton para polinomios

En un capítulo anterior estudiamos el método de Newton para aproximar ceros de funciones $p(z)$, y ya sabemos que si partimos de una aproximación inicial x_0 suficientemente próxima a un cero simple de p, la iteración de Newton dada por

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

converge cuadráticamente hacia ese cero de $p(z)$.

El esquema de Horner se puede utilizar para evaluar simultáneamente los valores de p y p', tal y como reflejamos en la versión del algoritmo de Newton para polinomios en la página siguiente. Obsérvese que el algoritmo tiene también sentido para polinomios con coeficientes complejos, pero no que no puede utilizarse para aproximar raíces complejas de polinomios con coeficientes reales a menos que partamos de una condición inicial compleja.

Otra propiedad del método de Newton para polinomios interesante la proporciona el siguiente teorema que permite localizar un cero del polinomio dadas dos iteraciones consecutivas, y ofrece una estimación del error cometido al aproximar una raíz del polinomio por una iterada de Newton.

Teorema 7.4.1 *Sean x_k y x_{k+1} dos iteraciones sucesivas del método de Newton para un polinomio $p(z)$ de grado $n \geq 1$. Entonces existe un cero de $p(z)$ en el disco complejo de centro x_k y radio $n|x_{k+1} - x_k|$.*

Demostración:

Ideas que intervienen:

- *El teorema de factorización 7.2.2 proporciona una representación del cociente $\frac{p'(z)}{p(z)}$ derivada logarítmica de $p(z)$ (llamado así porque coincide con la derivada de $\log(p(z))$). En efecto, si $z_1, z_2, \ldots, z_n$ son las raíces de $p(z)$ repetidas según su multiplicidad, entonces*

$$p(z) = a_n(z - z_1)(z - z_2)\cdots(z - z_n) = a_n \prod_{i=1}^{n}(z - z_i)$$

$$p'(z) = a_n \sum_{i=1}^{n} \prod_{j=1, j\neq i}^{n} (z - z_j) = \sum_{i=1}^{n} \frac{p(z)}{z - z_i} = p(z) \sum_{i=1}^{n} \frac{1}{z - z_i}.$$

- *El corrector de la aproximación inicial x_k en el método de Newton viene dado por $\frac{-p(x_k)}{p'(x_k)}$ que en términos de las raíces del polinomio se puede escribir como*

$$x_{k+1} - x_k = -\frac{p(x_k)}{p'(x_k)} = \frac{-1}{\sum_{i=1}^{n} \frac{1}{x_k - z_i}}.$$

- *Alguna de las raíces cumple $|x_k - z_i| \leq n|x_{k+1} - x_k|$, pues en otro caso la desigualdad triangular nos llevará a una contradicción.*

Algoritmo 7.2 **Newton para polinomios**

Datos de entrada:
$a_0, a_1, \ldots, a_n$ (coeficientes del polinomio p);
z (condición inicial);
tol (precisión para la parada);
nmax (número máximo de iteraciones);
Variables:
xini; // variable para almacenar x_k
xfin; // variable para almacenar x_{k+1}
b; // variable para almacenar $p(x_k)$
c; // variable para almacenar $p'(x_k)$
Flujo del programa:
xini $= z$;
xfin $= 0$;
`for(j=1;j<=nmax;j++){`
  $b = a_n$;
  $c = b$;
  // Evaluación de p(xini) y p'(xini) con Horner.
  `for(k=n-1;k>=1;k--){`
    $b = b$ xini $+ a_k$;
    $c = c$ xini $+ b$; // p'(xini) $= c$
  }
  $b = b$ xini $+ a_0$; // p(xini) $= b$
  `if(`$c \neq 0$`){`
    xfin $=$ xini $-$ (b/c);
  }
  `else{`
    Parada: derivada nula
  }
  `if((`$|b| <$ `tol) or (`$|$`xfin` $-$ `xini`$| <$ `tol)){`
    Parada: raíz en xini
  }
  `else{`
    xini $=$ xfin
  }
}
Parada: no hay convergencia en nmax iteraciones
Datos de salida: Raíz en xini o mensaje de error.

Si suponemos que $|x_k - z_i| > n|x_{k+1} - x_k|$ para $i = 1, \ldots, n$, tendremos la siguiente cadena contradictoria de desigualdades:

$$\begin{aligned} |x_{k+1} - x_k| &= \frac{1}{|\sum_{i=1}^{n} \frac{1}{x_k - z_i}|} \\ &\geq \frac{1}{\sum_{i=1}^{n} \frac{1}{|x_k - z_i|}} \\ &> \frac{1}{\sum_{i=1}^{n} \frac{1}{n|x_{k+1} - x_k|}} = \frac{1}{\frac{1}{|x_{k+1} - x_k|}} = |x_{k+1} - x_k|. \end{aligned}$$

□

Ejemplo 7.4.2 Calculando con ocho cifras decimales, las cinco primeras iteraciones del método de Newton aplicado al polinomio de los ejemplos anteriores $p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$ con inicio en $x_0 = 0$ y estimaciones del error con respecto a la raíz próxima a $x_0 = 0$ (véase la Figura 7.1) obtenemos los siguientes datos:

k	$p(x_k)$	$p'(x_k)$	x_k	$4\lvert x_{k-1} - x_k\rvert$
0	-2	-5	0	
1	1.4016	-12.776	-0.4	1.6
2	1.46321856	-10.1732291	-0.2902943	0.4388228
3	0.00225803	-9.86029921	-0.27591126	0.05753214
4	$5{,}6\,10^{-7}$	-9.85536835	-0.27568226	0.00028398
5	$3{,}6\,10^{-14}$	-9.85536711	-0.27568220	0.00000024

El teorema anterior nos asegura que $-0{,}27568226$ es una aproximación de la raíz buscada con un error menor que $2{,}4\,10^{-7}$.

En general no tenemos ningún procedimiento que nos proporcione una aproximación inicial para la cual el algoritmo de Newton sea convergente. Sin embargo esta regla general sí existe en algunos casos importantes, como en el siguiente teorema.

Teorema 7.4.3 *Sea $p(z)$ un polinomio de grado $n \geq 2$ con todas sus raíces y coeficientes reales. Si*

$$r_n \leq r_{n-1} \leq \cdots \leq r_2 \leq r_1$$

son los ceros de $p(z)$ ordenados en forma decreciente y $x_0 > r_1$, entonces el método de Newton proporciona una sucesión de iteradas x_k estrictamente decreciente convergente hacia r_1.

Demostración:

Ideas que intervienen:

- *El teorema de Rolle para funciones reales de variable real nos asegura que entre cada dos raíces reales consecutivas, r_k y r_{k+1}, hay una raíz s_k de la derivada de $p(x)$,*
$$r_n \leq s_{n-1} \leq r_{n-1} \leq \cdots \leq r_2 \leq s_1 \leq r_1.$$
En consecuencia $p'(x)$ tiene $n-1$ ceros reales $s_1, \ldots, s_{n-1}$ y todos son menores que r_1, así que $p'(x)$ no se anula en $[r_1, \infty)$.
- *Haciendo el mismo razonamiento con los $n-1$ ceros de $p'(x)$ podemos concluir que $p''(x)$ tampoco se anula en $[r_1, \infty)$.*
- *Estas condiciones garantizan la monotonía y convergencia de las iteradas de Newton con inicio en $x_0 > r_1$.*

Enfatizamos para empezar que todas las raíces de $p'(x)$ (y del mismo modo las de $p''(x)$) están entre r_n y r_1 incluso que el caso de que $p(x)$ tenga raíces repetidas (porque si r es una raíz the $p(x)$ de multiplicidad m entonces también lo es de $p'(x)$ con multiplicidad al menos $m-1$).

Podemos suponer que $p(x) > 0$ para cada $x > r_1$ (en caso contrario, cambiaríamos el signo de p; nótese que el método de Newton proporciona la misma sucesión de números).

Como $p(x)$ converge a ∞ o $-\infty$ cuando $x \to \infty$, se tendrá en este caso $\lim_{x\to\infty} p(x) = \infty$. Si a_n es el coeficiente principal de $p(x)$, sabemos que el comportamiento asintótico de $p(x)$ y $a_n x^n$ es el mismo, con lo que se concluye que $a_n > 0$. Como los coeficientes principales de $p'(x)$ y $p''(x)$ son, respectivamente, na_n y $n(n-1)a_n$, concluimos de igual manera $\lim_{x\to\infty} p'(x), p''(x) = \infty$ y por tanto (ya que ni $p'(x)$ ni $p''(x)$ se anulan a la derecha de r_1) que $p'(x), p''(x) > 0$ para cada $x > r_1$.

Comprobemos que en estas condiciones el método de Newton converge a r_1 siempre que la condición inicial x_0 está a la derecha de r_1.

Sea (x_k) la sucesión de iteradas por el método de Newton, es decir, $x_{k+1} = x_k - p(x_k)/p'(x_k)$. Basta demostrar que la sucesión de errores $e_k = x_k - r_1$ es positiva y decreciente, pues entonces (x_k) converge a un número $r \geq r_1$, y de hecho $r = r_1$ ya que de otro modo se llegaría, tomando límites en la expresión $x_{k+1} = x_k - p(x_k)/p'(x_k)$, a la contradicción $r = r - p(r)/p'(r)$.

Probamos a continuación que (e_k) es una sucesión positiva y decreciente. Haciendo desarrollos de Taylor de orden dos obtenemos $0 = p(r_1) = p(x_k - e_k) = p(x_k) - p'(x_k)e_k + (1/2)p''(\xi_k)e_k^2$ para un cierto número ξ_k comprendido entre r_1 y x_k. Por tanto

$$e_{k+1} = x_{k+1} - r_1 = x_k - r - \frac{p(x_k)}{p'(x_k)} = e_k - \frac{p(x_k)}{p'(x_k)} = \frac{p'(x_k)e_k - p(x_k)}{p'(x_k)} = \frac{(1/2)p''(\xi_k)e_k^2}{p'(x_k)}.$$

A partir de aquí la afirmación se deduce inmediatamente por inducción. □

Reuniendo este teorema y el teorema de localización 7.3.1 obtenemos el siguiente corolario.

Corolario 7.4.4 *Si $p(z) = a_0 + a_1 z + \cdots + a_n z^n$ tiene todos sus coeficientes y raíces reales y elegimos $x_0 > 1 + \frac{1}{|a_n|} \max\{|a_k| : k = 1, \ldots, n-1\}$, entonces el algoritmo de Newton $x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)}$, genera una sucesión monótona decreciente hacia la mayor de las raíces de $p(z)$.*

Para el caso de polinomios con todas sus raíces reales, el corolario anterior nos marca un camino a seguir para localizar todas las raíces. Una vez localizada la mayor raíz r_1 si hacemos la deflación de p en r_1 obtenemos un polinomio de grado $n-1$ con todas sus raíces reales al cual podemos aplicar el proceso. Naturalmente para un polinomio dado no podemos saber de antemano si todas las raíces son reales, pero incluso entonces ésta puede ser una buena estrategia para fijar el valor inicial.

7.5. El método de Laguerre

Este método se emplea en varios paquetes de cálculo debido a que sus propiedades de convergencia son muy favorables: puede probarse que tiene convergencia de orden tres para condiciones iniciales próximas a una raíz simple. Además sus cálculos numéricos son bastante robustos.

Antes de describir este algoritmo necesitamos algunos resultados previos que nos ayudarán a localizar las raíces de un polinomio. En un primer lema vamos a posicionar n números reales en un intervalo centrado en su media aritmética y con longitud medida en términos de su desviación típica. Sin llegar a describir y usar estos estadísticos lo enunciamos como sigue:

Lema 7.5.1 *Dados n números reales $v_1, v_2, \ldots, v_n$, sean $\alpha = \sum_{j=1}^{n} v_j$ y $\beta = \sum_{j=1}^{n} v_j^2$. Entonces todos los números v_j pertenecen al intervalo de extremos*

$$\frac{\alpha \pm \sqrt{(n-1)(n\beta - \alpha^2)}}{n}.$$

Demostración:

Ideas que intervienen:

- *Vamos a ver que para un polinomio $q(x)$ de grado 2 que toma valores positivos para $|x|$ grande, $q(v_j) \leq 0$ en cada número v_j. Por lo tanto estos números están en el intervalo cuyos extremos son las dos raíces del polinomio q, que van a coincidir con los números*

$$\frac{\alpha \pm \sqrt{(n-1)(n\beta - \alpha^2)}}{n}.$$

No es restrictivo trabajar sólo con el número v_1. Para los otros números basta con permutar las posiciones, y observar que estas permutaciones no afectan a los valores de α y β.

Tenemos

$$\begin{aligned}
\alpha^2 - 2\alpha v_1 + v_1^2 &= (\alpha - v_1)^2 \\
&= (v_2 + \cdots + v_n)^2 \\
&= (1.v_2 + 1.v_3 + \cdots + 1.v_n)^2 \\
&\leq (n-1)(v_2^2 + v_3^2 + \cdots + v_n^2) \\
&= (n-1)(\beta - v_1^2) \\
&= (n-1)\beta - nv_1^2 + v_1^2;
\end{aligned}$$

hemos usado la desigualdad de Cauchy-Schwartz

$$\sum_{k=1}^{m} a_k b_k \leq (\sum_{k=1}^{m} a_i^2)^{\frac{1}{2}} (\sum_{k=1}^{m} b_i^2)^{\frac{1}{2}}.$$

Por tanto

$$0 \leq (n-1)\beta - n v_1^2 + 2\alpha v_1 - \alpha^2$$

y

$$n v_1^2 - 2\alpha v_1 + \alpha^2 - (n-1)\beta \leq 0.$$

Considérese el polinomio $q(x) = nx^2 - 2\alpha x + (\alpha^2 - (n-1)\beta)$ y notemos que $q(v_1) \leq 0$. Como $\lim_{x\to\infty} q(x) = +\infty$, $q(x)$ cambia de signo, q tiene raíces reales, y v_1 está comprendido entre las dos raíces reales

$$x = \frac{\alpha \pm \sqrt{\alpha^2 - n(\alpha^2 - (n-1)\beta)}}{n} = \frac{\alpha \pm \sqrt{(n-1)(n\beta - \alpha^2)}}{n}.$$

□

En la sección anterior analizábamos las distancias de un punto a las raíces de un polinomio escribiendo

$$\frac{p'(x)}{p(x)} = \sum_{j=1}^{n} \frac{1}{x - z_j},$$

donde $z_1, \ldots, z_n$ son las n raíces de $p(x)$. Derivando en esa expresión con respecto a x y cambiando los signos, tenemos

$$\frac{-p(x)p''(x) + p'(x)^2}{p(x)^2} = \sum_{j=1}^{n} \frac{1}{(x - z_j)^2}.$$

Reescribiendo el lema anterior con $v_j = \frac{1}{x - z_j}$, $\alpha = \frac{p'(x)}{p(x)}$, $\beta = \frac{p'(x)^2 - p(x)p''(x)}{p(x)^2}$, se tiene el siguiente resultado.

Lema 7.5.2 *Sea $p(x)$ un polinomio de grado n con coeficientes reales cuyos ceros, $z_1, \ldots, z_n$, son todos reales. Entonces para cualquier número real x que no sea cero de p, los números $\frac{1}{x - z_j}$ están en el intervalo de extremos*

$$\frac{p'(x) \pm \sqrt{(n-1)^2 p'(x)^2 - n(n-1)p(x)p''(x)}}{np(x)}.$$

Demostración: Basta con utilizar el lema anterior y escribir:

$$\frac{\alpha \pm \sqrt{(n-1)(n\beta - \alpha^2)}}{n} = \frac{p'(x) \pm \sqrt{(n-1)^2 p'(x)^2 - n(n-1)p(x)p''(x)}}{np(x)}.$$

□

Supongamos que x_i es una aproximación de una raíz de p y que z_j es la raíz de p más próxima a x_i. Entonces la fracción $\frac{1}{x_i - z_j}$ será la de mayor tamaño (en valor absoluto). La idea de Laguerre consiste en aproximar el valor de esta fracción por el del extremo, C, del intervalo dado por el lema, que tiene mayor tamaño. Invirtiendo la fracción $\frac{1}{x_i - z_j}$ y su aproximación C obtenemos $x_f = x_i - \frac{1}{C}$ como una nueva aproximación de z_j.

Así el algoritmo de Laguerre consiste en el siguiente proceso iterativo:

(I) Se parte de una aproximación inicial x_i de una raíz de p.

(II) Se calcula $A = \frac{p'(xi)}{p(xi)}$ ($A = \alpha$ en la discusión anterior).

(III) Se calcula $B = \frac{p'(xi)^2 - p(x)p''(xi)}{p(xi)^2}$ ($B = \beta$ en la discusión anterior).

(IV) Se elige el signo $\pm$ para que $C = \frac{A \pm \sqrt{(n-1)(nB-A^2)}}{n}$ tenga el mayor valor absoluto.

(V) Se toma $x_f = x_i - \frac{1}{C}$ como la nueva aproximación de la raíz de p.

En el Algoritmo 7.3 se presenta el método de Laguerre en términos de las dos primeras derivadas del polinomio (evaluadas simultáneamente con el método de Horner) e incluyendo las correspondientes condiciones de parada.

Es importante resaltar que aunque (al igual que para el método de Newton) sólo podremos garantizar la convergencia cuando todas las raíces del polinomio sean reales, el algoritmo tiene sentido (y a menudo converge) cuando el radicando $(n-1)(nB-A^2)$ se hace negativo en algún momento. A partir de entonces los términos x_k de la sucesión de iterados se hacen complejos. De hecho, el algoritmo también sentido para condiciones iniciales complejas e incluso para polinomios con coeficientes complejos.

El siguiente teorema asegura la *convergencia global* del método de Laguerre aplicado a polinomios con todas sus raíces reales. El método converge a una raíz para cualquier condición inicial que se tome.

Teorema 7.5.3 *Sea $p(x)$ un polinomio real cuyas raíces son todas reales. Entonces la sucesión de iteradas de Laguerre con un punto inicial arbitrario x_0 converge monótonamente hacia uno de los dos ceros de p que tiene más próximos.*

Algoritmo 7.3 Método de Laguerre

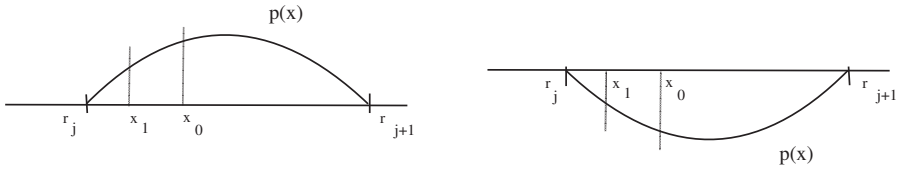
Datos de entrada:
$a_0, a_1, \ldots, a_n$ (coeficientes del polinomio p); z (condición inicial);
tol (tolerancia para la parada); nmax (número máximo de iteraciones);
Variables:
xini, xfin; // variables para almacenar x_k y x_{k+1}
b, c, d; // variables para almacenar $p(x_k)$, $p'(x_k)$ y $p''(x_k)$
aux1, aux2, Cmax; // variables auxiliares para almacenar C
Flujo del programa:
xini $= z$;
`for(j=1;j<=nmax;j++){`
  $b, c, d = a_n$;
  // Evaluación de p(xini), p'(xini) y p''(xini) con Horner.
  `for(k=n-1;k>=2;k--){`
    $b = b\,$xini $+ a_k$;
    $c = c\,$xini $+ b$;
    $d = d\,$xini $+ c$; // d $= p''$(xini)$/2$
  }
  $d = 2d$; // $d = p''$(xini)
  $b = b\,$xini $+ a_1$;
  $c = c\,$xini $+ b$; // $c = p'$(xini)
  $b = b\,$xini $+ a_0$; // $b = p$(xini)
  `if(b=0){`
    Parada: raíz en xini
  }
  `else{`
    // $A = c/b$, $B = (c^2 - b * d)/b^2$, $C = (1/n) * (A \pm \sqrt{(n-1)(nB - A^2)}$.
    aux1 $= \left(\frac{1}{nb}\right)\left(c + \sqrt{(n-1)^2c^2 - n(n-1)bd}\right)$;
    aux2 $= \left(\frac{1}{nb}\right)\left(c - \sqrt{(n-1)^2c^2 - n(n-1)bd}\right)$;
    `if(`|aux1| $\geq$ |aux2|`){`
      Cmax $=$ aux1;
    }
    `else{`
      Cmax $=$ aux2;
    }

  }
  `if(Cmax=0){`
    Parada: primera y segunda derivadas nulas
  }
  `else{`
    xfin $=$ xini $- \frac{1}{\text{Cmax}}$;
  }
  `if(|xfin-xini|<tol){`
    Parada: raíz en xini;
  }
  `else{`
    xini $=$ xfin
  }
}
Parada: no hay convergencia en nmax iteraciones
Datos de salida: Raíz en xini o mensaje de error.

DEMOSTRACIÓN:

Ideas que intervienen:

- *Si $r_1 \leq r_2 \leq \cdots \leq r_n$ son las n raíces de p, entonces entre cada par de raíces consecutivas el signo de p es constante. Además, como entre cada par de raíces hay algún cero de $p'(x)$ y $p'(x)$ tiene a lo sumo $n-1$ ceros reales, concluimos que $p'(x)$ tiene exactamente un cero entre dos raíces consecutivas distintas. Así, si p es positiva, p pasa de ser creciente a decreciente y si p es negativa entonces p pasa de ser decreciente a ser creciente.*



- *Si por ejemplo $r_j < x_0 < r_{j+1}$ con $p(x) > 0$ entre las dos raíces y además $p'(x_0) > 0$, entonces $p(x)$ es creciente entre r_j y x_0 y vamos a ver que $r_j \leq x_1 < x_0 < r_{j+1}$, que $p(x_1) > 0$ y que $p'(x_1) > 0$. Repitiendo el argumento, $r_j \leq x_2 < x_1 < x_0 < r_{j+1}$. Es decir, x_k es monótona decreciente y está acotada inferiormente por r_j y, en consecuencia, es convergente,*

- *Si x_k converge hacia y, de las fórmulas de iteración resulta que $x_{k+1} - x_k = \frac{1}{C(k)} \to 0$, y por tanto $C(k) \to \infty$. Ahora, por la construcción de los valores de $C(k)$,*

$$C(k) = \frac{p'(x_k) \pm \sqrt{(n-1)^2 p'(x_k)^2 - n(n-1)p(x_k)p''(x_k)}}{np(x_k)}.$$

se deduce que el denominador $p(x_k)$ tiende a $p(y) = 0$. Así, y es un cero de p, y por la disposición de los x_k, $y = r_j$.

Vamos a restringirnos al caso $r_j < x_0 < r_{j+1}$ con $p(x) > 0$ entre las dos raíces y además $p'(x_0) > 0$. Los otros casos se pueden analizar en la misma forma.

Como x_0 no es una raíz, el Lema 7.5.2 posiciona todas las fracciones $\frac{1}{x_0 - r_j}$ en el intervalo de extremos

$$u(x_0) = \frac{p'(x_0) + \sqrt{(n-1)^2 p'(x_0)^2 - n(n-1)p(x_0)p''(x_0)}}{np(x_0)}$$

y

$$v(x_0) = \frac{p'(x_0) - \sqrt{(n-1)^2 p'(x_0)^2 - n(n-1)p(x_0)p''(x_0)}}{np(x_0)}.$$

En particular, para las dos raíces r_j y r_{j+1} se tiene que

$$v(x_0) \leq \frac{1}{x_0 - r_{j+1}} < 0 < \frac{1}{x_0 - r_j} \leq u(x_0).$$

De estas desigualdades se sigue que

$$x_0 - r_{j+1} \leq \frac{1}{v(x_0)} < 0 < \frac{1}{u(x_0)} \leq x_0 - r_j$$

y

$$r_j \leq x_0 - \frac{1}{u(x_0)} < x_0 < x_0 - \frac{1}{v(x_0)} \leq r_{j+1}.$$

Como $p'(x_0) > 0$, se tiene que $|v(x_0)| < |u(x_0)|$ y $x_1 = x_0 - \frac{1}{u(x_0)}$, lo que prueba la desigualdad buscada $r_j \leq x_1 < x_0$. Además, p es creciente entre r_j y x_0, por lo que también se tiene $p'(x_1) > 0$.

Un buen ejercicio para comprobar como funciona el teorema es realizar la demostración en los casos restantes. □

Al igual que ocurre con el Teorema 7.4.1 para el método de Newton, la construcción de Laguerre también proporciona un disco donde localizar raíces expresado en términos de los cálculos realizados en cada iteración. Una prueba del siguiente teorema puede verse en el libro de Kincaid y Cheney [3] (pp. 103–105).

Teorema 7.5.4 (Kahan, 1967) *Sean $p(z)$ un polinomio de grado n y $z_0 \in \mathbb{C}$ un número complejo que tomamos como aproximación inicial en el primer paso de la iteración de Laguerre. Si C se calcula como en el algoritmo ($1/|C| = |x_f - x_i|$), entonces $p(z)$ tiene un cero en el disco*

$$D\left(z_0, \frac{\sqrt{n}}{|C|}\right) = \left\{z : |z - z_0| < \frac{\sqrt{n}}{|C|}\right\}.$$

Para finalizar revisitamos el polinomio de los ejemplos anteriores:

Ejemplo 7.5.5 Las cinco primeras iteraciones del método de Laguerre aplicado a

$$p(z) = z^4 - 4z^3 + 7z^2 - 5z - 2$$

con inicio en $x_0 = 0$ muestran la rápida convergencia del método de Laguerre a la raíz más próxima a x_0. En la última columna escribimos la cota del error que proporciona el teorema de Kahan:

k	x_k	$p(x_k)$	$\sqrt{4}/\|C\|$
0	0	-2	0.5577742855120604
1	-0.2788871427560302	0.031696585387203324	0.006409886096878962
2	-0.2756821997075907	0.011457452669712431	7.886788553624726E-9
3	-0.275682203650985	0	0

Obsérvese cómo en la primera aproximación del ejemplo anterior se tienen fijas 2 cifras decimales, en la segunda pasan a ser 6 las cifras de precisión y en la tercera se alcanza la precisión de la máquina, en consonancia con que el orden de convergencia de este método es 3.

7.6. Separación de raíces reales. Sucesiones de Sturm.

En esta sección vamos a trabajar con polinomios con coeficientes reales buscando localizar intervalos en los que podamos asegurar la existencia de raíces reales y determinar el número de raíces reales distintas que tiene el polinomio. Seguiremos el libro de Stoer y Bulirsh [4, Section 5.6, pp. 297–301] y el de Aubanell, Benseny y Delshams [1, Sección 5.2.4, pp. 404–406].

Vamos a utilizar las sucesiones de Sturm para construir un método que cuenta el número de raíces reales que un polinomio tiene en un intervalo. Después usamos ese método como en el algoritmo de la bisección: partiremos de un intervalo $[-A, A]$, donde A es el radio del disco que contiene a todas las raíces (véase el teorema 7.3.1); este intervalo contiene a todas las

raíces reales, luego iremos dividiendo los intervalos en mitades, contaremos el número de raíces en cada mitad, y nos iremos quedando con los trozos que contengan alguna raíz. Conforme vayamos avanzando con la bisección de los intervalos iremos localizando las raíces reales de forma simultánea. Como el proceso de bisección no es muy rápido, una vez separadas las raíces en intervalos disjuntos se puede ensayar con Newton o Laguerre para localizarlas más rápidamente.

Definición 7.6.1 *Una sucesión de funciones reales continuas*

$$\{f_0, f_1, \ldots, f_m\}$$

definidas en un intervalo $[a, b]$*, es una* sucesión de Sturm *si se cumplen las cuatro condiciones siguientes:*

(I) f_0 *es derivable en* $[a, b]$*;*

(II) f_m *no tiene raíces reales en* $[a, b]$ *(no se anula);*

(III) *si* $f_0(x) = 0$ *entonces* $f_1(x)f_0'(x) > 0$*;*

(IV) *si* $f_i(x) = 0$ *(*$i \in \{1, 2, \ldots, m-1\}$*) entonces* $f_{i+1}(x)f_{i-1}(x) < 0$.

La tercera condición implica que f_0 sólo puede tener raíces simples, y tanto esta condición como la siguiente permiten detectar las raíces de f_0 observando el número de cambios $n(x)$ de signo en la sucesión $\{f_0(x), f_1(x), \ldots, f_m(x)\}$ tal y como establece el siguiente teorema.

Teorema 7.6.2 (Sturm) *Sea* $\{f_0, f_1, \ldots, f_m\}$ *una sucesión de Sturm en el intervalo* $[a, b]$ *y sea* $n(x)$ *el número de cambios de signo (los ceros se ignoran) en la sucesión*

$$\{f_0(x), f_1(x), \ldots, f_m(x)\}.$$

Entonces el número de raíces reales de f_0 *en* $(a, b]$ *es igual a* $n(a) - n(b)$.

DEMOSTRACIÓN:

Ideas que intervienen:

- *Por la continuidad de las funciones* f_i *el número de cambios de signo* $n(x)$ *va a ser constante en entornos de los puntos* x_0 *en los que no se anula ninguna de las funciones. Por tanto este número* $n(x)$ *sólo puede verse modificado al pasar por alguno de los ceros de las mismas.*
- *Bastará con observar que* $n(x)$ *se incrementa en una unidad al pasar por cada una de las raíces de* f_0*, y permanece constante en otro caso.*

Supongamos que $f_i(x_0) = 0$ con $i \in \{1, 2, \ldots, m-1\}$. Entonces por la condición 4ª de la definición de Sturm tendremos que $f_{i+1}(x_0)f_{i-1}(x_0) < 0$. En particular, estas dos funciones no se anulan en un entorno de x_0 y tienen distintos signos. Por lo tanto cerca de x_0 sucederá alguno de los casos siguientes (representamos por $-$ los valores negativos, por $+$ los positivos, por $\otimes$ indistintamente 0, $+$ o $-$, y $h > 0$ es suficientemente pequeño):

	$x_0 - h$	x_0	$x_0 + h$
f_{i-1}	$-$	$-$	$-$
f_i	$\otimes$	0	$\otimes$
f_{i+1}	$+$	$+$	$+$

	$x_0 - h$	x_0	$x_0 + h$
f_{i-1}	$+$	$+$	$+$
f_i	$\otimes$	0	$\otimes$
f_{i+1}	$-$	$-$	$-$

En cualquiera de los casos el número de cambios de signo de la sucesión $\{f_{i-1}, f_i, f_{i+1}\}$ es constante e igual a 1 en las proximidades de x_0.

Si $f_0(x_0) = 0$, por la tercera propiedad de las sucesiones de Sturm $f_0'(x_0) \neq 0$ y además el signo de $f_1(x_0)$ es el mismo que el de $f_0'(x_0)$. Así, si esta derivada es positiva (f_0 es creciente en x_0) entonces $f_1(x_0) > 0$ y por la continuidad f_1 es positiva en un entorno de x_0. De la misma forma, cuando la derivada es negativa, la función f_1 es negativa en un entorno de x_0. Por lo tanto sólo se presentan los dos casos siguientes:

	$x_0 - h$	x_0	$x_0 + h$	
f_0	$-$	0	$+$	$(f_0'(x_0) > 0)$
f_1	$+$	$+$	$+$	$(f_1(x_0) > 0)$

	$x_0 - h$	x_0	$x_0 + h$	
f_0	$+$	0	$-$	$(f_0'(x_0) < 0)$
f_1	$-$	$-$	$-$	$(f_1(x_0) < 0)$

En los dos casos deja de haber un cambio de signo en la sucesión $\{f_0, f_1\}$ al pasar por x_0.

En resumen, el número de cambios de signo $n(x)$ sólo disminuye en una unidad al pasar de izquierda a derecha sobre cada una de las raíces de f_0. De esta forma, el total $n(a) - n(b)$ mide el número de raíces reales de f_0 en $(a, b]$. □

La construcción de sucesiones de Sturm asociadas a polinomios proviene del método de Euclides de búsqueda del máximo común divisor.

- Sea $p_0 = p$ un polinomio real de grado $n \geq 1$.
- Se define $p_1 = p'(x)$.
- Para $i = 1, 2, 3, \ldots$ se define $p_{i+1} = -c_{i+1}\text{resto}\,(p_{i-1} : p_i)$, donde c_{i+1} es una constante positiva y por resto entendemos el resto que queda al hacer la división entera del polinomio p_{i-1} entre p_i. Cuando la sucesión de Sturm se construye con un ordenador se elige simplemente $c_{i+1} = 1$. Cuando lo hacemos a mano, conviene elegirlo de manera que los coeficientes del polinomio p_{i+1} sean tan simples como sea posible. En otros términos:

  $$p_{i-1}(x) = q_i(x)p_i(x) - c_{i+1}^{-1}p_{i+1}(x). \tag{7.2}$$

  Esta construcción se realiza hasta que $p_{m+1} = 0$.
- En está construcción el polinomio p_m es el máximo común divisor de p_0 y $p_1 = p'$.
- Por último consideramos la sucesión

  $$\left\{ f_0 = \frac{p_0}{p_m}, f_1 = \frac{p_1}{p_m}, \ldots, f_m = \frac{p_m}{p_m} = 1 \right\}.$$

  Obsérvese que f_0 es un polinomio que tiene las mismas raíces que p_0 pero, para f_0, todas las raíces son simples. En efecto, las raíces múltiples de p son las raíces comunes de p y p', es decir, son las raíces de p_m. Si una raíz x_0 tiene multiplicidad k en p, entonces tiene multiplicidad $k - 1$ en p' y en p_m. Por consiguiente x_0 es una raíz simple de $f_0 = \frac{p}{p_m}$.

Lema 7.6.3 *Si p es un polinomio de grado $n \geq 1$, entonces la sucesión $\{f_0, f_1, \ldots, f_m\}$ que acabamos de construir es una sucesión de Sturm.*

DEMOSTRACIÓN: Todas las funciones $f_0, \ldots, f_m$ son polinomios, así que en particular son funciones continuas (primera condición en la definición de sucesiones de Sturm).

Por la definición, $f_m = \frac{p_m}{p_m} = 1$ es constante y no nula. Por tanto se cumple la segunda condición de Sturm.

Supongamos que $f_0(x_0) = 0$ para un cierto x_0. Ello equivale a x_0 es una raíz de p_0, digamos de multiplicidad m. Entonces

$$p_0(x) = (x - x_0)^m q(x)$$

con $q(x_0) \neq 0$. Asimismo,

$$\begin{aligned} p_1(x) &= p_0'(x) = m(x - x_0)^{m-1} q(x) + (x - x_0)^m q'(x) = (x - x_0)^{m-1}(mq(x) + (x - x_0)q'(x)) \\ &=: (x - x_0)^{m-1} t(x) \end{aligned}$$

con $t(x_0) \neq 0$. Dado que p_m es el máximo común divisor de p_0 y p_1, tendremos

$$p_m(x) = (x - x_0)^{m-1} c(x),$$

donde $c(x)$ es divisor de ambos polinomios $q(x)$ y $t(x)$. Pongamos $q(x) = a(x)c(x)$. Entonces $f_0(x) = (x - x_0)a(x)$ y

$$f_0'(x) = (x - x_0)a'(x) + a(x)$$

con lo que en particular $f_0'(x_0) = a(x_0) \neq 0$. Por otro lado

$$f_1(x) = \frac{t(x)}{c(x)} = \frac{mq(x) + (x - x_0)q'(x)}{c(x)}$$

y así $f_1(x_0) = mq(x_0)/c(x_0) = ma(x_0)$. En consecuencia $f_1(x_0)f_0'(x) = ma(x_0)^2 > 0$. Ésta es la tercera condición en las sucesiones de Sturm.

A partir de (7.2), y dividiendo por p_m, obtenemos

$$f_{i-1}(x) = q_i(x) f_i(x) - c_{i+1}^{-1} f_{i+1}(x).$$

Si $f_i(x_0) = 0$ $(1 \leq i \leq m - 1)$ para un cierto punto x_0, entonces

$$f_{i-1}(x_0) f_{i+1}(x_0) = -c_{i+1}^{-1} f_{i+1}(x_0)^2 \leq 0,$$

y se anula sólo si $f_{i+1}(x_0) = 0$. Si esto último sucediese, podemos repetir el argumento con f_{i+1}, para obtener que

$$f_i(x_0) = f_{i+1}(x_0) = f_{i+2}(x_0) \cdots = f_m(x_0) = 0.$$

Como $f_m(x_0) = 1$ esto no puede suceder. Por lo tanto también se cumplirá la última condición de la definición, lo que completa la prueba de que $\{f_0, \ldots, f_m\}$ es una sucesión de Sturm. □

Notemos que si $p(a) \neq 0 \neq p(b)$ entonces también ocurrirá $p_m(a) \neq 0 \neq p_m(b)$, con lo que a la hora de computar los cambios de signo podremos usar la sucesión $\{p_0, p_1, \ldots, p_m\}$ en lugar de $\{f_0, f_1, \ldots, f_m\}$. Hemos probado:

Corolario 7.6.4 *Sea $p(x)$ un polinomio con coeficientes reales y constrúyase la correspondiente sucesión $\{p_0, p_1, \ldots, p_m\}$. Sea $V(x)$ el número de cambios de signo (los ceros se ignoran) en la sucesión*

$$\{p_0(x), p_1(x), \ldots, p_m(x)\}.$$

Entonces, si $p(a) \neq 0 \neq p(q)$, el número de raíces reales de p en $[a, b]$ es igual a $V(a) - V(b)$.

Estamos ahora en condiciones de explicar cómo separar todas las raíces reales de un polinomio con coeficientes reales.

El teorema 7.3.1 localiza todas las raíces reales de un polinomio $p(z) = a_0+a_1z+\cdots+a_nz^n$ en el intervalo $[-\rho, \rho]$, (de hecho en el abierto $(-\rho, \rho)$, como se desprende de una lectura cuidadosa de la prueba) con $\rho = 1 + \text{máx}\{|a_0|, \ldots, |a_{n-1}|\}/|a_n|$. Por otra parte, el teorema de Sturm nos permite conocer el número de raíces reales en cada intervalo.

Así, si comenzando con la lista de intervalos $\mathcal{L}_0 = \{I_0\}$, con $I_0 = (\rho, \rho]$, podemos razonar como en el método de la bisección, y en cada etapa k:

- dividir en dos intervalos semiabiertos de la misma longitud cada intervalo de la lista $\mathcal{L}_k$,
- mirar, con la sucesión de Sturm, si hay ceros reales en cada una de las mitades, y
- construir una nueva lista $\mathcal{L}_{k+1}$ con las mitades que contienen alguna raíz.

Este proceso iterativo puede detenerse cuando tengamos una sola raíz en cada intervalo y la longitud de estos intervalos sea suficientemente pequeña. Como el proceso de biseccionar intervalos no permite aproximarnos a las raíces de forma rápida podemos optar por

- continuar con el proceso hasta que la proximidad de los extremos de los intervalos sea tan pequeña que nos permita admitir que aproximan a una raíz, o
- detener la bisección de los intervalos cuando cualquiera de sus extremos sirva como aproximación inicial para aplicar métodos más rápidos como el de Newton o Laguerre.

Para poder aplicar el teorema de Sturm 7.6.2 necesitamos un método que cuente los cambios de signo en una sucesión de números reales sin tener en cuenta los ceros:

Algoritmo 7.4 **Método para contar los cambios de signo de una lista**

Datos de entrada:
lista[m] (lista de números reales);
Variables: ncambios; // para contar los cambios de signo
ini, sig; // índices para movernos por la sucesión
pini, psig; // variables auxiliares
Flujo del programa:
// Usamos un método signo(x) =1, -1 o 0, según que $x > 0$, $x < 0$ o $x = 0$
ini $= 0$;
pini $=$ lista[0];
ncambios $= 0$;
`while(ini` $< m-1$ `&& signo(pini)` $= 0$`){`
  ini $=$ ini $+ 1$;
  pini $=$ lista[ini];
}
`if(ini` $= m-1$`){`
  Parada: devolver ncambios;
}
`while(ini` $< m-1$`){`
  sig $=$ ini $+ 1$;
  psig $=$ lista[sig];
  `while(sig` $< m-1$ `&& signo(pini)` $*$ `signo(psig)` $\neq -1$`){`
    sig $=$ sig $+ 1$;
    psig=lista[sig];
  }
  `if(sig` $= m-1$ `&& signo(pini)` $*$ `signo(psig)` $\neq -1$`){`
    Parada: devolver ncambios;
  }
  `else{`
    ncambios $=$ ncambios $+ 1$;
    ini $=$ sig;
  }
}
Parada: devolver ncambios;
Datos de salida: ncambios (número de cambios de signo de una lista).

Los algoritmos siguientes describen el método de construcción de la sucesión de Sturm asociada a un polinomio, y el método de la bisección para localizar raíces en un intervalo.

Algoritmo 7.5 **Método para calcular la sucesión de Sturm de un polinomio**

Datos de entrada:
$a[n]$ (lista con los coeficientes de un polinomio p de grado $n-1$);
Variables:
sturm$[n][]$; // matriz para la sucesión de Sturm
listaauxiliar[]; // vector auxiliar
Flujo del programa:
sturm$[0][n-1] = a[n-1]$;
`if(`$n=1$`){`
  Parada: devolver sturm
}
`for(i=0;i<n-1;i++){`
  sturm$[0][i] = a[i]$;
  sturm$[1][i] = (i+1) * a[i+1]$; // $p_0 = p$ y $p_1 = p'$
}
`if(`$n=2$`){`
  Parada: devolver sturm
}
`for(k=2;k<n;k++){`
  listaauxiliar= resto(sturm[k-2]:sturm[k-1]);
  `if(listaauxiliar.length`$=1$ `&& listaauxiliar`$[0]=0$`){`
    `for(j=0;j<=k-1;j++){`
      sturm$[j]$ = cociente(sturm$[j]$, sturm$[k-1]$);
    }
    Parada: devolver sturm
  }
  sturm$[k] = -$listaauxiliar;
}
Parada: devolver sturm
Datos de salida: sturm$[][]$ (matriz con los coeficientes sturm$[i][]$ de los polinomios de la sucesión de Sturm; nótese que a partir de $i = k$ los coeficientes quedan vacíos).

Algoritmo 7.6 **Método para separar los ceros de un polinomio**

Datos de entrada:
prec (precisión para longitud de intervalos);
sturm$[k][]$ (matriz con los coeficientes de la sucesión de Sturm de un polinomio p);
a, b (extremos del intervalo inicial);
Variables: sturma[], sturmb[]; // vectores para guardar las evaluaciones de
// la sucesión de Sturm
intervalos; // lista de longitud variable para guardar los extremos de los intervalos
// que contienen exactamente una raíz real
s, c; // variables auxiliares
Flujo del programa:
// Evaluamos la sucesión de Sturm en a y b.
`for(i=0;i<k;i++){`
  sturma$[i]=$ (evaluación de sturm$[i]$ en $x=a$);
  sturmb$[i]=$ (evaluación de sturm$[i]$ en $x=b$);
}
$s=$ cuentaCambiosSigno(sturma) $-$ cuentaCambiosSigno(sturmb);
`if(`$s \leq 0$`){`
  intervalos $=\emptyset$; // lista de intervalos vacía
  Parada: devolver intervalos
}
`if(`$s=1$ `&&` $b-a<$ `prec){`
  intervalos $=\{a,b\}$; // lista con $(a,b]$ como único intervalo
  Parada: devolver intervalos
}
`else{`
  $c=a+0.5*(b-a)$;
  intervalos $=$ biseccionSturm(prec, sturm, $a, c) \cup$ biseccionSturm(prec, sturm, c, b);
// lista unión de las dos listas correspondientes a a, c y b, c
  Parada: devolver intervalos
}
Datos de salida: intervalos (lista de intervalos conteniendo exactamente una raíz real del polinomio).

El último método propuesto utiliza recursividad, es decir, se va llamando a si mismo, y listas de longitud variable. Si programamos en JAVA la clase `Vector` es una herramienta bastante adecuada, aunque una vez construida requiere una transformación en una lista (array) estándar de números reales.

Como en otros apartados vamos a finalizar volviendo a visitar el polinomio utilizado en los ejemplos anteriores:

Ejemplo 7.6.5 La sucesión de Sturm asociada al polinomio $p(x) = x^4 - 4x^3 + 7x^2 - 5x - 2$ y generada por el algoritmo (con todos los coeficientes $c_i = 1$) es la siguiente

$$\begin{aligned}
p_0(x) &= x^4 - 4x^3 + 7x^2 - 5x - 2 \\
p_1(x) &= p'(x) = 4x^3 - 12x^2 + 14x - 5 \\
p_2(x) &= -\text{resto}\,(p_0 : p_1) = -0.5x^2 + 0.25x + 3.25 \\
p_3(x) &= -\text{resto}\,(p_1 : p_2) = -35x + 70 \\
p_4(x) &= -\text{resto}\,(p_2 : p_3) = -1.75
\end{aligned}$$

Si realizamos las operaciones a mano los polinomios propuestos como alternativa son múltiplos positivos de los otros, pero con coeficientes más sencillos para las evaluaciones. Por ejemplo, en lugar de $p_2(x)$ podríamos haber multiplicado todos los coeficientes por 4 y usar $-2x^2 + x + 13$.

Comenzando en el intervalo $(-8, 8]$, que contiene a todas las raíces reales, observamos que en -8 la sucesión de Sturm toma valores {+,-,-,+,-} con lo que aparecen tres cambios de signo, mientras que en 8 tenemos {+,+,-,-,-}, así que hay un único cambio de signo. La diferencia del número de cambios de signo es 2, el número de raíces reales. Tomando el punto medio en el intervalo, que es 0, y viendo el signo de los valores de la sucesión de Sturm en esos puntos, {-,-,+,+,-}, observamos que hay dos cambios de signo y por lo tanto podemos concluir que una de las raíces es negativa y la otra positiva.

Aplicando el método descrito en el algoritmo con una precisión de 10^{-14} obtenemos los intervalos

$$(-0.2756822036512858, -0.27568220365083107] \quad \text{y} \quad (1.9999999999995453, 2.0]$$

que proporcionan excelentes aproximaciones de las dos raíces reales $-0.27568220365\ldots$ y 2.

Bibliografía

[1] A. Aubanell, A. Benseny y A. Delshams, *Útiles Básicos de Cálculo Numérico*, Labor, Barcelona, 1993.

[2] J. A. Férnandez Viña, *Análisis Matemático II : Topología y Cálculo Diferencial. 2ª ed. corr.*, Tecnos, Madrid, 1992.

[3] D. Kincaid y W. Cheney, *Análisis Numérico. Las Matemáticas del Cálculo Científico*, Addison-Wesley Sudamericana, Wilmington, 1994.

[4] J. Stoer y R. Bulirsh, *Introduction to Numerical Analysis. 2nd. edition*, Springer-Verlag, Nueva York, 1993.

Capítulo 8

Interpolación y aproximación

Interrogantes centrales del capítulo

- Interpolación polinomial. El algoritmo de las diferencias divididas.
- Interpolación de Hermite.
- Aproximación por mínimos cuadrados. Sistemas sobredeterminados.
- Aproximación uniforme por polinomios. El teorema de Marcinkiewicz y el algoritmo de Rémès.

Destrezas a adquirir en el capítulo

- Describir los algoritmos correspondientes a los distintos métodos de aproximación (interpolación, mínimos cuadrados y uniforme) por polinomios.
- Estimar, cuando sea posible, el error de las aproximaciones obtenidas.
- Implementar en el ordenador los programas de los métodos de este capítulo representando gráficamente las polinomios obtenidos para visualizar la calidad de la aproximación.

Desarrollo de los contenidos fundamentales

En este capítulo abordamos la cuestión de cómo aproximar funciones mediante funciones sencillas (polinomios). La cuestión es importante no sólo porque los polinomios sean más fáciles de manejar que las funciones arbitrarias. En muchas ocasiones no sabemos cuál es la función sino sólo los valores que toma en unos ciertos puntos (por ejemplo porque se hayan obtenido de forma experimental), por lo que es crucial disponer de una función de aproximación plausible que nos permita inferir el valor de la función, salvo un pequeño error, en el resto de los puntos.

Existen dos formas de enfocar el problema. La primera es fijar una familia de puntos de la función y encontrar el polinomio de grado más pequeño posible que pasa por dichos puntos. Aparece así el problema de la interpolación, que trataremos en las dos primeras secciones del capítulo (en la primera en su versión más simple, en la segunda en una versión más general que permite usar, si es que disponemos de ella, información sobre los valores que toman las derivadas sucesivas de la función a aproximar en dichos puntos).

En principio, cuanto mayor sea el número de puntos a interpolar la calidad de la aproximación será mejor. El precio a pagar es que el grado del polinomio aumenta y con ello las dificultades para manejarlo. Un enfoque alternativo es fijar, con independencia del número de puntos disponible, el grado del polinomio para a continuación escoger, de entre todos los polinomio de grado a lo sumo el fijado, el que mejor aproxima (en términos de mínimos cuadrados) a la familia de puntos. Este problema será tratado en la tercera sección.

Acabamos de decir que la calidad de la aproximación de los polinomios interpoladores aumenta con el tamaño de la familia de puntos a interpolar, pero el problema es mucho más sutil de lo que cabría pensar, pues resulta que esto no tiene por qué ocurrir necesariamente, salvo que elijamos las sucesivas familias de puntos de manera adecuada. Estudiaremos este problema en la sección cuarta del capítulo.

8.1. Interpolación polinomial

Los contenidos de esta sección están bien explicados en el libro de Kincaid y Cheney [3] (Secciones 6.1 y 6.2, pp. 285–313).

Consideremos una familia de puntos $\{(x_i, y_i)\}_{i=0}^n$ con las abscisas distintas dos a dos. Llamamos *polinomio interpolador en los puntos* $\{(x_i, y_i)\}$ a un polinomio de grado a lo sumo n (abreviadamente $p \in \Pi_n$) tal que $p(x_i) = y_i$ para cada $i = 0, 1, \ldots, n$.

Teorema 8.1.1 *El polinomio interpolador existe y es único.*

Demostración:

Ideas que intervienen:

- *La unicidad es consecuencia de que un polinomio de grado n no nulo no puede tener $n+1$ raíces distintas.*
- *La existencia puede probarse directamente (forma del polinomio interpolador de Lagrange) o mediante un razonamiento inductivo, construyendo el polinomio interpolador para una familia de $n+1$ puntos sabido el polinomio interpolador para los n primeros (forma del polinomio interpolador de Newton).*

La demostración de la unicidad es inmediata: si $p(x)$ y $q(x)$ son ambos polinomios interpoladores en los puntos $\{(x_i, y_i)\}$, entonces $p(x) - q(x)$ sería un polinomio de grado a lo sumo n que tiene todos los puntos x_i, $i = 0, 1, \ldots, n$, como raíces. Por tanto $p(x) - q(x)$ debe ser el polinomio nulo, es decir, $p = q$.

Damos dos demostraciones de la existencia del polinomio interpolador. Una es la construcción directa: es simple comprobar que

$$p(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

es polinomio interpolador (ésta es la llamada *forma del polinomio interpolador de Lagrange*).

La segunda demostración es por inducción sobre n y tiene la ventaja de proporcionar una manera de construir el polinomio interpolador bastante práctica desde el punto de vista computacional (la llamada *forma del polinomio interpolador de Newton*). Para $n = 0$ es claro; basta tomar el polinomio constante $p(x) = y_0$. Supongamos ahora que p_{n-1} es polinomio interpolador en los puntos $\{(x_i, y_i)\}_{i=0}^{n-1}$ y busquemos un polinomio p_n interpolador en los puntos $\{(x_i, y_i)\}_{i=0}^{n}$ de la forma

$$p_n(x) = p_{n-1}(x) + c_n \prod_{j=0}^{n-1}(x - x_j).$$

De hecho, con independencia del valor del número c_n, se satisface de inmediato $p_n(x_i) = y_i$ para todo $i < n$. Si queremos que también se cumpla $p_n(x_n) = y_n$ bastará elegir

$$c_n = \frac{y_n - p_{n-1}(x_n)}{\prod_{j=0}^{n-1}(x_n - x_j)}.$$

□

De acuerdo con la construcción por inducción del polinomio interpolador de Newton, éste viene dado por

$$p(x) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1}(x - x_j)$$

para ciertos coeficientes c_i, $i = 0, 1, \ldots, n$. Obsérvese que c_n es el coeficiente principal del polinomio interpolador. Más exactamente, si $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, entonces $c_n = a_n$, aunque c_n puede ser cero: recuérdese que $p \in \Pi_n$, es decir, p tiene grado a lo sumo n pero puede ser inferior a n. Por otro lado, el resto de coeficientes c_i ya no coinciden con los a_i. De hecho, el coeficiente c_i será el coeficiente principal del polinomio interpolador en los puntos $\{(x_j, y_j)\}_{j=0}^{i}$. En general, si para un subconjunto dado $\{x_{i_0}, x_{i_1}, \ldots, x_{i_{r-1}}, x_{i_r}\}$ de $\{x_i\}_{i=0}^{n}$ denotamos por $p[x_{i_0}, x_{i_1}, \ldots, x_{i_{r-1}}, x_{i_r}]$ al coeficiente principal del polinomio interpolador en $\{(x_{i_k}, y_{i_k})\}_{k=0}^{r}$, entonces podrá escribirse

$$p(x) = \sum_{i=0}^{n} p[x_0, \ldots, x_i] \prod_{j=0}^{i-1}(x - x_j). \tag{8.1}$$

Resulta que hay una manera muy sencilla de calcular los números $p[x_0, \ldots, x_i]$: es el llamado *algoritmo de las diferencias divididas*. Obviamente, $p[x_i] = y_i$ para cada i. En el resto de los casos:

Teorema 8.1.2 *Con la notación anterior, se cumple*

$$p[x_{i_0}, x_{i_1}, \ldots, x_{i_{r-1}}, x_{i_r}] = \frac{p[x_{i_1}, \ldots, x_{i_{r-1}}, x_{i_r}] - p[x_{i_0}, x_{i_1}, \ldots, x_{i_{r-1}}]}{x_{i_r} - x_{i_0}}.$$

Demostración:

Ideas que intervienen:

- *Se usa que los números $p[x_0, x_1, \ldots, x_{n-1}]$, $p[x_1, \ldots, x_{n-1}, x_n]$ y $p[x_0, x_1, \ldots, x_{n-1}, x_n]$ son los coeficientes principales de ciertos polinomios interpoladores.*
- *Se combinan los dos primeros polinomios interpoladores para generar el tercero aprovechando la unicidad del polinomio interpolador.*
- *Se igualan el coeficiente principal del polinomio generado combinando los dos primeros y el coeficiente principal del tercero.*

Basta demostrar (en el caso general se razona análogamente) que

$$p[x_0, x_1, \ldots, x_{n-1}, x_n] = \frac{p[x_1, \ldots, x_{n-1}, x_n] - p[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0}.$$

Para ello es suficiente demostrar que si p_n, p_{n-1} y q son los polinomios interpoladores en $\{(x_i, y_i)\}_{i=0}^{n}$, $\{(x_i, y_i)\}_{i=0}^{n-1}$ y $\{(x_i, y_i)\}_{i=1}^{n}$, respectivamente, entonces

$$p_n(x) = \frac{q(x)(x - x_0)}{x_n - x_0} - \frac{p_{n-1}(x)(x - x_n)}{x_n - x_0}.$$

Por la unicidad de polinomio interpolador, es suficiente comprobar que si definimos

$$h(x) = \frac{q(x)(x - x_0)}{x_n - x_0} - \frac{p_{n-1}(x)(x - x_n)}{x_n - x_0},$$

entonces $h(x_i) = y_i$ para cada i. Ahora bien, si $1 \leq i \leq n$, entonces

$$h(x_i) = \frac{q(x_i)(x_i - x_0)}{x_n - x_0} - \frac{p_{n-1}(x_i)(x_i - x_n)}{x_n - x_0} = \frac{y_i(x_i - x_0)}{x_n - x_0} - \frac{y_i(x_i - x_n)}{x_n - x_0} = y_i.$$

Además,

$$h(x_0) = -\frac{p_{n-1}(x_0)(x_0 - x_n)}{x_n - x_0} = -\frac{y_0(x_0 - x_n)}{x_n - x_0} = y_0$$

y

$$h(x_n) = \frac{q(x_n)(x_n - x_0)}{x_n - x_0} = \frac{y_n(x_n - x_0)}{x_n - x_0} = y_n.$$

□

Cuando se usa en la práctica el algoritmo de las diferencias divididas, lo que se hace es escribir una secuencia de tablas cuyos términos superiores son los números $p[x_0, x_1, \ldots, x_i]$ de la fórmula (8.1).

Ejemplo 8.1.3 Calculamos el polinomio interpolador en la familia de puntos

$$\{(5, 1), (-7, -23), (-6, -54), (0, -954)\}$$

(nótese que no hay obligación de escribir los puntos en ningún orden concreto). De acuerdo con (8.1) se tendrá

$$p(x) = p[5] + p[5,-7](x-5) + p[5,-7,-6](x-5)(x+7) + p[5,-7,-6,0](x-5)(x+7)(x+6).$$

Ahora escribimos los números a calcular como sigue:

$$\begin{array}{c|cccc} 5 & p[5] & & & \\ & & p[5,-7] & & \\ -7 & p[-7] & & p[5,-7,-6] & \\ & & p[-7,-6] & & p[5,-7,-6,0] \\ -6 & p[-6] & & p[-7,-6,0] & \\ & & p[-6,0] & & \\ 0 & p[0] & & & \end{array}$$

En la primera columna escribimos $p[5] = 1$, $p[-7] = -23$, $p[-6] = -54$ y $p[0] = -954$. En la segunda obtenemos

$$p[5,-7] = \frac{p[-7]-p[5]}{-7-5} = 2, \qquad p[-7,-6] = \frac{p[-6]-p[-7]}{-6-(-7)} = -31,$$

$$p[-6,0] = \frac{p[0]-p[-6]}{0-(-6)} = -150.$$

En la tercera se obtiene

$$p[5,-7,-6] = \frac{p[-7,-6]-p[5,-7]}{-6-5} = 3, \quad p[-7,-6,0] = \frac{p[-6,0]-p[-7,-6]}{0-(-7)} = -17.$$

Por último

$$p[5,-7,-6,0] = \frac{p[-7,-6,0]-p[5,-7,-6]}{0-5} = -4.$$

Así pues, el polinomio interpolador buscado será

$$\begin{aligned} p(x) &= 1 + 2(x-5) + 3(x-5)(x+7) + 4(x-5)(x+7)(x+6)) \\ &= 4x^3 + 35x^2 - 84x - 954. \end{aligned}$$

Describimos a continuación la implementación numérica del algoritmo:

Algoritmo 8.1 **Coeficientes de la forma del polinomio interpolador de Newton**

Datos de entrada:
$n+1$ (número de puntos a interpolar);
$x[n+1]$ (abscisas de los puntos a interpolar);
$y[n+1]$ (ordenadas de los puntos a interpolar);

Variables:
n; // grado del polinomio interpolador
$p[n+1]$; // coeficientes a calcular

Flujo del programa:
```
for(i=0;i<=n;i++){
  p_i = y_i;
}
for(i=1;i<=n;i++){
  for(j=n;j>=i;j--){
    p_j = (p_j - p_{j-1})/(x_j - x_{j-i});
  }
}
```

Datos de salida: Coeficientes $p_0, p_1, \ldots, p_n$ para la forma del polinomio interpolador de Newton

Una vez calculados los coeficientes c_i de la forma del polinomio interpolador de Newton

$$p(x) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1} (x - x_j),$$

la manera más eficiente de calcular el polinomio es usando el llamado algoritmo de *Horner* (o *anidación*). Para simplificar, imaginemos de momento que queremos calcular una suma del tipo

$$s = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1} d_j$$

para ciertos números c_i, $i = 0, 1, \ldots, n$, y d_j, $j = 0, \ldots, n-1$. Entonces, sacando factor común repetidas veces, obtenemos:

$$\begin{aligned} s &= c_0 + c_1 d_0 + c_2 d_0 d_1 + \cdots + c_{n-2} d_0 d_1 \cdots d_{n-3} + c_{n-1} d_0 d_1 \cdots d_{n-3} d_{n-2} \\ &\quad + c_n d_0 d_1 \cdots d_{n-3} d_{n-2} d_{n-1} \\ &= c_0 + d_0(c_1 + d_1(c_2 + \cdots + d_{n-3}(c_{n-2} + d_{n-2}(c_{n-1} + d_{n-1} c_n)) \cdots)). \end{aligned}$$

En pseudocódigo la suma se calcularía como sigue:

Algoritmo 8.2 **Algoritmo de anidación**

Datos de entrada:
$n+1$ (número de sumandos en el sumatorio);
$c[n+1]$ (valores de los números c_k);
$d[n]$ (valores de los números d_j);

Variables:
sum; // valor del sumatorio $\sum_{i=0}^{n} c_i \prod_{j=0}^{n-1} d_j$

Flujo del programa:
sum $= c_n$;
`for(i=n-1;i>=0;i--){`
  sum $= c_i + d_i *$ sum;
}

Datos de salida: Sumatorio sum

Usamos a continuación el algoritmo de anidación para escribir el polinomio interpolador en la forma usual $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} + a_nx^n$. Para ello cubriremos $n+1$ etapas (indexadas de forma decreciente desde $i = n$ a $i = 0$), de manera que en la etapa i calculamos un polinomio $p_{n-i} \in \Pi_{n-i}$ a partir de otro $p_{n-i-1} \in \Pi_{n-i-1}$ usan la fórmula

$$p_{n-i}(x) = c_i + (x - x_i)p_{n-i-1}(x).$$

El polinomio final p_n es el polinomio interpolador p buscado. Hacemos notar que en en la etapa inicial $i = n$ entendemos que el polinomio p_{-1} es el polinomio nulo, de manera que $p_0(x) \equiv c_n$.

Si denotamos

$$p_{n-i-1}(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-i-1}x^{n-i-1},$$

$$p_{n-i}(x) = b_0 + b_1x + b_2x^2 + \cdots + b_{n-i}x^{n-i},$$

se tendrá

$$\begin{aligned} p_{n-i}(x) &= c_i + (x - x_i)(a_0 + a_1x + a_2x^2 + \cdots + a_{n-i-1}x^{n-i-1}) \\ &= (c_i - a_0x_i) + (a_0 - a_1x_i)x + (a_1 - a_2x_ik)x^2 + \cdots + (a_{n-i-3} - a_{n-i-2}x_i)x^{n-i-2} \\ &\quad +(a_{n-i-2} - a_{n-i-1}x_k)x^{n-i-1} + a_{n-i-1}x^{n-i}, \end{aligned}$$

así que la relación entre los coeficientes a_k y b_k vendrá dada por $b_{n-i} = a_{n-i-1}$,

$$b_k = a_{k-1} - a_kx_i, \quad k = 1, \ldots, n-i-1,$$

y $b_0 = c_i - a_0x_i$. En la práctica los coeficientes a_k calculan sin ayuda de los coeficientes auxiliares b_k, dándoles de inicio el valor cero y usando la definición provisional de los coeficientes $a_0, a_1, \ldots, a_{n-i-1}$ para recalcular en la etapa i, también de forma provisional, los coeficientes $a_0, a_1, \ldots, a_{n-i}$.

Algoritmo 8.3 Polinomio interpolador de Newton

Datos de entrada:
$n+1$ (número de puntos a interpolar);
$x[n+1]$ (abscisas de los puntos a interpolar);
$c[n+1]$ (coeficientes de la forma del polinomio interpolador de Newton);

Variables:
n; // grado del polinomio interpolador
$a[n+1]$; // coeficientes del polinomio interpolador

Flujo del programa:

```
for(i=0;i<=n;i++){
  a_i = 0;
}
for(i=n;i>=0;i--){
  for(k=n-i;k>=1;k--){
    a_k = a_{k-1} - a_k * x_i;
  }
  a_0 = c_i - a_0 * x_i;
}
```

Datos de salida: Coeficientes $a_0, a_1, \ldots, a_n$ del polinomio interpolador de Newton

En interesante resaltar que, aunque en apariencia no sea así, el número de operaciones que se necesita para calcular el valor $p(x)$ del polinomio interpolador en un cierto punto x combinando el algoritmo de las diferencias divididas y el de anidación es inferior al que se necesita usando directamente la forma del polinomio interpolador de Lagrange.

En efecto, el Algoritmo 8.1 consta de n etapas, numeradas de $i = 1$ a $i = n$, en cada una de las cuales hay que hacer $2(n - i + 1)$ sumas o restas y $n - i + 1$ productos o divisiones. Por tanto el total de sumas o restas es

$$SR_1 = 2n + 2(n-1) + \cdots + 2 \cdot 2 + 2 \cdot 1 = n(n+1)$$

y el de productos o divisiones $PD_1 = n(n+1)/2$. Por su parte el Algoritmo 8.2 (reemplazando d_i por $x - x_i$) consta de n etapas, cada una de las cuales acarrea dos sumas o restas y un producto. El total será por tanto de $SR_2 = 2n$ sumas o restas y $PD_2 = n$ productos. El total requerido de operaciones será, por tanto, $SR = SR_1 + SR_2 = n(n+1) + 2n = n^2 + 3n$ sumas o restas y $PD = PD_1 + PD_2 = (n^2 + 3n)/2$ productos o divisiones.

Si usamos la forma del polinomio interpolador de Lagrange

$$p(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

para calcular $p(x)$ encontramos que en cada una de las etapas $i = 0, 1, \ldots, n$ hay que hacer $2n$ restas, n divisiones y n productos ($n - 1$ correspondientes al producto de las n fracciones $(x - x_j)/(x_i - x_j)$ y otra más de multiplicar por y_i). Por tanto el total de sumas o restas es en este caso $2n(n+1)$ y el de productos o divisiones también $2n(n+1)$.

Hasta ahora hemos estudiado el polinomio interpolador de una familia de puntos sin considerar la procedencia de los puntos de partida. En el caso que estos puntos sean del tipo $\{(x_i, f(x_i))\}_{i=0}^n$ para una cierta función $f : [a, b] \to \mathbb{R}$ definida en un intervalo $[a, b]$ que contenga a los puntos $\{x_0, x_1, \ldots, x_n\}$ es natural preguntarse hasta qué punto el polinomio interpolador proporcionará una buena aproximación de la función. El siguiente teorema da respuesta a esta pregunta.

Teorema 8.1.4 *Sea $f : [a, b] \to \mathbb{R}$ una función de clase C^{n+1} y sea $p \in \Pi_n$ el polinomio que interpola a la función f en $n + 1$ puntos distintos $\{x_i\}_{i=0}^n$ (es decir, el polinomio interpolador en los puntos $\{(x_i, f(x_i))\}_{i=0}^n$). Entonces para cada $x \in [a, b]$ existe $\xi_x \in [a, b]$ tal que*

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

Demostración:

Ideas que intervienen:

- *Fijo x, se considera la función $\phi(t) = f(t) - p(t) - \lambda \prod_{i=0}^n (t - x_i)$, con $\lambda = \frac{f(x)-p(x)}{\prod_{i=0}^n (x-x_i)}$.*
- *Se usa el teorema de Rolle para probar que su derivada $(n + 1)$-ésima tiene un cero ξ_x.*
- *Sustituyendo en dicha derivada, se obtiene la fórmula pedida.*

Fijemos un punto $x \in [a, b]$. Si x es alguno de los puntos x_i no hay nada que demostrar. Por tanto supondremos que $x \neq x_i$ para todo i.

Consideremos la función

$$\phi(t) = f(t) - p(t) - \lambda \prod_{i=0}^{n} (t - x_i),$$

donde elegimos λ de manera que $\phi(x) = 0$. En concreto,

$$\lambda = \frac{f(x) - p(x)}{\prod_{i=0}^n (x - x_i)}.$$

Como también $\phi(x_i) = 0$ para cada i, vemos que ϕ tiene al menos $n + 2$ raíces. Aplicando repetidas veces el teorema de Rolle, la derivada k-ésima de ϕ tendrá al menos $n + 2 - k$ raíces. En particular, $\phi^{(n+1)}$ tiene alguna raíz, que denotaremos por ξ_x.

Ahora bien, dado que $\phi(t)$ el es resultado de restarle a $f(t)$ un polinomio de grado a lo sumo n, $p(t)$, y otro de grado $n + 1$, $\lambda \prod_{i=0}^n (t - x_i)$, su derivada $(n + 1)$-ésima será el resultado de restarle a $f^{(n+1)}$ el coeficiente principal del polinomio $\lambda \prod_{i=0}^n (t - x_i)$ multiplicado por $(n + 1)!$, es decir,

$$\phi^{(n+1)}(t) = f^{(n+1)}(t) - (n + 1)!\lambda.$$

Sustituyendo t por ξ_x y despejando se obtiene el resultado buscado. □

Por ejemplo, si $[a, b] = [-1, 1]$, $f(x) = \operatorname{sen} x$, y tomamos como familia de puntos interpoladores, fijado un cierto entero positivo n, los puntos $\{(i/n, \operatorname{sen}(i/n))\}_{i=0}^n$, entonces se cumple para

el correspondiente polinomio interpolador p_n que

$$|p_n(x) - f(x)| = \frac{|f^{(n+1)}(\xi_x)|}{(n+1)!} \prod_{i=0}^{n} |x - x_i| \leq \frac{2^{n+1}}{(n+1)!}$$

para cada $x \in [-1, 1]$. Consecuentemente la sucesión de polinomios interpoladores p_n converge uniformemente a f cuando $n \to \infty$, lo que se corresponde con la intuición. Veremos más adelante que, sorprendentemente, esto no tiene por qué ocurrir siempre.

8.2. Interpolación de Hermite

En esta sección extendemos los resultados de la sección anterior permitiendo que podamos fijar no sólo los valores del polinomio interpolador en ciertos puntos sino también los de algunas de sus derivadas sucesivas. Obtenemos así el llamado *polinomio interpolador de Hermite*. Lo que sigue es una reelaboración de algunos de los contenidos de la Sección 6.3 (pp. 313–323) de [3].

Proposición 8.2.1 *Considérese la familia* $\{(x_i, y_{i,l})\}$, *con* $i = 0, 1, \ldots, m$, $0 \leq l < k_i$, *y con las abscisas* x_i *distintas dos a dos. Sea* $n = k_0 + k_1 + \cdots k_m - 1$. *Entonces existe un único polinomio* $p \in \Pi_n$ *tal que*

$$p^{(j)}(x_i) = y_{i,l}$$

para cada i, l.

Demostración:

Ideas que intervienen:

- *La existencia y unicidad del polinomio interpolador de Hermite es equivalente a la de un sistema lineal de* $n + 1$ *ecuaciones que tiene como incógnitas los* $n + 1$ *coeficientes del polinomio interpolador de Hermite.*
- *Para que dicho sistema tenga solución única es necesario y suficiente que la matriz que lo define sea no singular.*
- *Dado que los coeficientes de dicha matriz no dependen de los datos* $y_{i,l}$, *basta demostrar la existencia y unicidad de soluciones en el caso en que todos los datos* $y_{i,l}$ *son nulos.*
- *Dado que en dicho caso el polinomio proporcional trivialmente una solución, debe demostrarse que es la única solución posible.*
- *Ello se consigue usando que si un polinomio de grado a lo sumo* n *tiene* $n + 1$ *raíces (contando multiplicidades) entonces es el polinomio nulo.*

Sea $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ el polinomio candidato a satisfacer las condiciones prefijadas. Entonces éstas determinar un sistema de $n + 1$ ecuaciones lineales donde las incógnitas son los coeficientes $a_0, a_1, \ldots, a_n$ del polinomio $p(x)$. Por ejemplo $p(x_0) = y_{0,0}$ equivale a

$$a_0 + x_0 a_1 + x_0^2 a_2 + \cdots x_0^n a_n = y_{0,0},$$

$p'(x_0) = y_{0,1}$ equivale a

$$a_1 + 2x_0a_2 + \cdots nx_0^{n-1}a_n = y_{0,1},$$

y así sucesivamente.

Como es bien sabido, el que el sistema tenga solución única depende de que la matriz de coeficientes del sistema sea no singular. Como la matriz no depende de los números $y_{i,l}$, si demostramos que el problema con las condiciones $y_{i,l} = 0$ para cada i y l tiene solución única, habremos terminado.

Evidentemente el polinomio nulo es solución en este caso, así que debemos demostrar que ésa es, de hecho, la única solución posible. Ahora bien, si un polinomio $p(x)$ satisface que las derivadas en un punto x_i hasta un cierto orden $k_i - 1$ se anulan, entonces admite un factor $(x-x_i)^{k_i}$ en su descomposición factorial (esto es fácil de ver escribiendo el desarrollo de Taylor del polinomio en x_i). Así pues, el polinomio $p(x)$ es divisible por $(x-x_0)^{k_0}(x-x_1)^{k_1}\cdots,(x-x_m)^{k_m}$, un polinomio de grado $n+1$, cosa imposible salvo que $p(x)$ sea el polinomio idénticamente nulo. □

Hay otra manera de probar la Proposición 8.2.1, muy semejante a la que se utiliza para demostrar la existencia y unicidad del polinomio interpolador (Teorema 8.1.1). Primero redenotamos la familia $\{(x_i, y_{i,l})\}$, donde a partir de ahora supondremos que $x_0 < x_1 < \cdots < x_m$ para simplificar la notation, como $\{(\overline{x_i}, \overline{y_i})\}_{i=0}^{n}$, teniéndose entonces $\overline{x_0} \leq \overline{x_1} \leq \cdots \overline{x_n}$. Así, $(\overline{x_0}, \overline{y_0})$ sería $(x_0, y_{0,0})$, y $(\overline{x_n}, \overline{y_n})$ sería (x_m, y_{0,k_m-1})). Ahora demostramos por inducción que el polinomio interpolador de Hermite puede escribirse como

$$p(x) = \sum_{k=0}^{n} c_k \prod_{j=0}^{k-1}(x - \overline{x_j}), \tag{8.2}$$

afirmación que es obvia para $n = 0$ y que probamos a continuación para el caso n, suponiéndola verdadera para el caso $n - 1$.

Sea p_{n-1} el polinomio interpolador de Hermite en $\{(\overline{x_i}, \overline{y_i})\}_{i=0}^{n-1}$. Basta demostrar que el polinomio interpolador de Hermite en $\{(\overline{x_i}, \overline{y_i})\}_{i=0}^{n}$, $p_n(x)$, puede escribirse como

$$p_n(x) = p_{n-1}(x) + c_n \prod_{j=0}^{n-1}(x - \overline{x_j}) = p_{n-1}(x) + c_n(x - x_0)^{k_0}\cdots(x - x_{m-1})^{k_{m-1}}(x - x_m)^{k_m-1}.$$

Para empezar, obsérvese que para cada x_i, $i = 0, \ldots, m-1$, aparece en el producto de la derecha un factor $(x - x_i)^{k_i}$, lo que significa que las derivadas de dicho producto hasta el orden $k_i - 1$ se anulan cuando se evalúan en x_i. Otro tanto puede decirse para x_m pero esta vez sólo hasta el orden $k_m - 2$. A la luz de la hipótesis de inducción, y con independencia del valor de c_n, se tendrá entonces

$$p_n^{(l)}(x_i) = y_{i,l}$$

para cada $0 \leq i < m$ y $0 \leq l < k_i$, y también para $i = m$ y $0 \leq l < k_m - 1$.

Ahora bien, cuando se hace la derivada de orden $k_m - 1$ de la expresión de la derecha entonces aparecen distintos sumandos, todos los cuales se anulan en x_m, excepto

$$q(x) = (k_m - 1)!(x - x_0)^{k_0}\cdots(x - x_{m-1})^{k_{m-1}}$$

que sólo contiene factores de la forma $x - x_0, \ldots, x - x_{m-1}$. Dado que

$$p_n^{(k_m-1)}(x_m) = p_n^{(k_m-1)}(x_m) + c_n q(x_m),$$

como $q(x)$ no se anula en x_m bastará tomar

$$c_n = \frac{y_{m,k_m-1} - p_{n-1}^{(k_m-1)}(x_m)}{q(x_m)}.$$

Llegados a este punto cabría preguntarse si existe una variante del algoritmo de las diferencias divididas en este caso. La respuesta es afirmativa y, de hecho, las diferencias son muy pequeña. Para cada subfamilia $\overline{x_{i_0}} \leq \cdots \leq \overline{x_{i_r}}$ de $\overline{x_0} \leq \cdots \leq \overline{x_n}$ definimos, al estilo del polinomio interpolador convencional, $p[\overline{x_{i_0}}, \ldots, \overline{x_{i_r}}]$ como el coeficiente que multiplica a x^r del polinomio interpolador de Hermite en los puntos correspondientes. Por ejemplo, si la familia de partida es

$$(x_0, y_{0,0}),\ (x_0, y_{0,1}),\ (x_1, y_{1,0}),\ (x_1, y_{1,1}),\ (x_1, y_{1,2}),\ (x_2, y_{2,0}),\ (x_2, y_{2,1})$$

entonces el polinomio interpolador de Hermite en los puntos $(x_0, y_{0,0})$, $(x_0, y_{0,1})$ y $(x_1, y_{1,0})$ vendría dado por

$$p(x) = p[x_0, x_0, x_1]x^2 + \cdots,$$

el polinomio interpolador de Hermite en los puntos $(x_0, y_{0,0})$, $(x_1, y_{1,0})$, $(x_1, y_{1,1})$ y $(x_2, y_{2,0})$ sería

$$p(x) = p[x_0, x_1, x_1, x_2]x^3 + \cdots,$$

etc. Nótese que (8.2) puede reescribirse entonces como

$$p(x) = \sum_{k=0}^{n} p[\overline{x_0}, \overline{x_1}, \ldots, \overline{x_k}] \prod_{j=0}^{k-1} (x - \overline{x_j}). \tag{8.3}$$

El siguiente teorema es la clave del algoritmo:

Teorema 8.2.2 *Se tiene que*

$$p[\overline{x_0}, \overline{x_1}, \ldots, \overline{x_n}] = \begin{cases} \dfrac{p[\overline{x_1}, \ldots, \overline{x_n}] - p[\overline{x_0}, \ldots, \overline{x_{n-1}}]}{\overline{x_n} - \overline{x_0}} & \text{si } \overline{x_0} \neq \overline{x_n}, \\ \dfrac{y_{0,n}}{n!} & \text{si } \overline{x_0} = \overline{x_n}. \end{cases}$$

Demostración:

Ideas que intervienen:

- *La idea fundamental de la prueba es esencialmente la misma que la del Teorema 8.1.2.*
- *El caso $\overline{x_0} = \overline{x_1} = \cdots = \overline{x_n}$ se considera aparte y es consecuencia del teorema de Taylor.*

El caso más simple es cuando $\overline{x_0} = \overline{x_n}$. En este caso $(x_0, y_{0,0}), \ldots, (x_0, y_{0,n})$ es la familia a interpolar y el polinomio interpolador de Hermite es simplemente el polinomio de Taylor

$$p(x) = y_{0,0} + y_{0,1}(x - x_0) + \frac{y_{0,2}}{2!}(x - x_0)^2 + \cdots + \frac{y_{0,n}}{n!}(x - x_0)^n.$$

Por tanto

$$p[\overline{x_0}, \overline{x_1}, \ldots, \overline{x_n}] = p[x_0, x_0, \ldots, x_0] = \frac{y_{0,n}}{n!}.$$

Supongamos ahora que $x_0 = \overline{x_0} \neq \overline{x_n} = x_m$. Al igual que en la versión del teorema para la interpolación convencional, si

$$p_{n-1}(x) = p[\overline{x_0}, \ldots, \overline{x_{n-1}}]x^{n-1} + \cdots ,$$

$$q(x) = p[\overline{x_1}, \ldots, \overline{x_n}]x^{n-1} + \cdots ,$$

y

$$p_n(x) = p[\overline{x_0}, \overline{x_1}, \ldots, \overline{x_n}]x^n + \cdots$$

son los correspondientes polinomios interpoladores de Hermite, es suficiente demostrar que

$$p_n(x) = \frac{q(x)(x - \overline{x_0})}{\overline{x_n} - \overline{x_0}} - \frac{p_{n-1}(x)(x - \overline{x_n})}{\overline{x_n} - \overline{x_0}} = \frac{q(x)(x - x_0)}{x_m - x_0} - \frac{p_{n-1}(x)(x - x_m)}{x_m - x_0}.$$

Si llamamos $h(x)$ al polinomio de la derecha basta demostrar, por la unicidad del polinomio interpolador de Hermite, que $h^{(l)}(x_i) = y_{i,l}$ para cada $0 \leq i \leq m$ y $0 \leq l < k_m$.

Se prueba de inmediato por inducción que la derivada l-ésima de $q(x)(x - x_0)$ viene dada por

$$[q(x)(x - x_0)]^{(l)} = q^{(l)}(x)(x - x_0) + lq^{(l-1)}(x),$$

y del mismo modo

$$[p_{n-1}(x)(x - x_m)]^{(l)} = p_{n-1}^{(l)}(x)(x - x_m) + lp_{n-1}^{(l-1)}(x).$$

Si $0 < i < m$, entonces

$$q^{(l)}(x_i) = p_{n-1}^{(l)}(x_i) = y_{i,l},$$

para cada $0 \leq l < k_i$, de modo que también

$$\begin{aligned} h^{(l)}(x_i) &= \frac{q^{(l)}(x_i)(x_i - x_0) + lq^{(l-1)}(x_i)}{x_m - x_0} - \frac{p_{n-1}^{(l)}(x_i)(x_i - x_m) + lp_{n-1}^{(l-1)}(x_i)}{x_m - x_0} \\ &= \frac{y_{i,l}(x_i - x_0) + ly_{i,l-1}}{x_m - x_0} - \frac{y_{i,l}(x_i - x_m) + ly_{i,l-1}}{x_m - x_0} = y_{i,l}. \end{aligned}$$

Asimismo tenemos

$$q^{(l-1)}(x_0) = p_{n-1}^{(l-1)}(x_0) = y_{0,l-1}$$

si $l < k_0$, con lo que

$$h^{(l)}(x_0) = \frac{ly_{0,l-1}}{x_m - x_0} - \frac{y_{0,l}(x_0 - x_m) + ly_{0,l-1}}{x_m - x_0} = y_{0,l}$$

para cada $0 \leq l < k_0$. De igual manera se comprueba que $h^{(l)}(x_m) = y_{m,l}$, $0 \leq l < k_m$. □

Como en el caso de las diferencias divididas de Newton, el Teorema 8.2.2 permite calcular recursivamente, y de una manera sencilla, los coeficientes $p[\overline{x_0}, \overline{x_1}, \ldots, \overline{x_k}]$.

Ejemplo 8.2.3 Calculamos a continuación el polinomio $p(x)$ de grado a lo sumo 4 que cumple las condiciones $p(1) = 2$, $p'(1) = 3$, $p(2) = 6$, $p'(2) = 7$, $p''(2) = 8$. De acuerdo con (8.3) se tendrá

$$\begin{aligned} p(x) &= p[1] + p[1,1](x-1) + p[1,1,2](x-1)(x-1) + p[1,1,2,2](x-1)(x-1)(x-2) \\ &\quad + p[1,1,2,2,2](x-1)(x-1)(x-2)(x-2). \end{aligned}$$

Ahora se tendrá el siguiente esquema:

$$\begin{array}{c|ccccc}
1 & p[1] & & & & \\
 & & p[1,1] & & & \\
1 & p[1] & & p[1,1,2] & & \\
 & & p[1,2] & & p[1,1,2,2] & \\
2 & p[2] & & p[1,2,2] & & p[1,1,2,2,2] \\
 & & p[2,2] & & p[1,2,2,2] & \\
2 & p[2] & & p[2,2,2] & & \\
 & & p[2,2] & & & \\
2 & p[2] & & & &
\end{array}$$

En la primera columna obtenemos $p[1] = 2$ y $p[2] = 6$. En la segunda tendremos

$$p[1,1] = \frac{p'(1)}{1!} = 3, \quad p[1,2] = \frac{p[2]-p[1]}{2-1} = 4, \quad p[2,2] = \frac{p'(2)}{1!} = 7.$$

En la tercera tendremos

$$p[1,1,2] = \frac{p[1,2]-p[1,1]}{2-1} = 1, \quad p[1,2,2] = \frac{p[2,2]-p[1,2]}{2-1} = 3, \quad p[1,2,2] = \frac{p''(2)}{2!} = 4.$$

En la cuarta

$$p[1,1,2,2] = \frac{p[1,2,2]-p[1,1,2]}{2-1} = 2, \quad p[1,2,2,2] = \frac{p[2,2,2]-p[1,2,2]}{2-1} = 1,$$

Finalmente

$$p[1,1,2,2,2] = \frac{p[1,2,2,2]-p[1,1,2,2]}{2-1} = -1.$$

Por tanto el polinomio buscado será

$$\begin{aligned}
p(x) &= 2 + 3(x-1) + (x-1)(x-1) + 2(x-1)(x-1)(x-2) \\
&\quad -(x-1)(x-1)(x-2)(x-2) \\
&= -8 + 23x - 20x^2 + 8x^3 - x^4.
\end{aligned}$$

Finalizamos la sección escribiendo el pseudocódigo correspondiente al método de interpolación de Hermite. Hemos añadido al final el algoritmo de anidación de manera que el algoritmo arroje como resultado los coeficientes del polinomio interpolador.

Algoritmo 8.4 **Polinomio interpolador de Hermite**

Datos de entrada:
$m+1$ (número de puntos a interpolar);
$x[m+1]$ (abscisas de los puntos a interpolar;
$y[m+1][-]$ (ordenadas de los puntos a interpolar;
$y[i][k]$ es la derivada k-ésima del polinomio interpolador en $x[i]$);
$l[m+1]$ (longitudes de las filas de y);
Variables:
aux$[m+2]$; // vector auxiliar para contar las multiplicidades
n; // grado del polinomio interpolador
$a[n+1]$; // coeficientes del polinomio interpolador
$u[n+1]$; $p[n+1]$; // variables auxiliares
Flujo del programa:

```
for(i=0;i<=m;i++){
  for(j=i+1;j<=m;l++){
    if(x_i = x_j){
      Parada: abscisas repetidas
    }
  }
}
```

// Calculamos el grado del polinomio interpolador.
$\text{aux}_0 = 0$;

```
for(i=0;i<=m;i++){
```

$\text{aux}_{i+1} = \text{aux}_i + l_i$;

```
}
```

$n = \text{aux}_{m+1} - 1$; // grado del polinomio interpolador

Algoritmo 8.4 **Polinomio interpolador de Hermite (continuación)**

```
// Cálculo de abscisas generalizadas.
for(i=0;i<=m;i++){
  for(j=aux[i];j<aux[i+1];j++){
    u_j = x_i;
  }
}
// Algoritmo de las diferencias divididas.
for(k=0;k<=n;k++){
  for(i=m;i>=0;i--){
    for(j=aux[i+1]-1;j>=max{aux[i],k};j--){
      if(u_{j-k} = u_j){
        p_j = y_{i,k};
        for(r=1;r<=k;l++){
          p_j = p_j/r;
        }
      }
      else{
        p_j = (p_j - p_{j-1})/(u_j - u_{j-k})
      }
    }
  }
}
// Cálculo de los coeficientes del polinomio interpolador.
for(k=0;k<=n;k++){
  a_k = 0;
}
for(k=n;k>=0;k--){
  for(i=n-k;i>=1;i--){
    a_i = a_{i-1} - a_i * u_k;
  }
  a_0 = p_k - a_0 * u_k;
}
```

Datos de salida: Polinomio interpolador con coeficientes $a_0, a_1, \ldots, a_n$

8.3. Aproximación por mínimos cuadrados

Considérese la familia de puntos $\{(x_i, y_i)\}_{i=0}^m$ con abscisas distintas dos a dos. La teoría de la interpolación nos dice que existe un único polinomio $q(x)$ de grado a lo sumo m tal que $q(x_i) = y_i$ para cada i. Por tanto, en general, si $n < m$ no existirá para estos puntos un polinomio interpolador de grado menor o igual que n. En estas condiciones tiene sentido, no obstante, preguntarse si existirá algún polinomio $p(x) \in \Pi_n$ que minimice la norma euclídea $\|(y_0 - p(x_0), \ldots, y_m - p(x_m))\|$ o, equivalentemente, la suma $\sum_{i=0}^m (y_i - p(x_i))^2$. La respuesta a esta pregunta es afirmativa pero para formularla adecuadamente necesitamos introducir los lla-

mados *sistemas lineales sobredeterminados*. Los contenidos de esta sección se inspiran (bastante libremente) en parte de la Sección 7.1 (pp. 394–418) de [1].

Sean $n < m$, $A \in \mathcal{M}_{m\times n}(\mathbb{R})$ y $b \in \mathbb{R}^m$. En lo que sigue supondremos que A tiene rango n. En general el sistema

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ &\cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m, \end{aligned}$$

que en forma matricial puede escribirse como $Ax = b$ para el vector columna $x \in \mathbb{R}^n$ cuyas componentes son las incógnitas $x_1, \ldots, x_n$, carecerá de solución. Mostraremos a continuación que existe $u \in \mathbb{R}^n$ tal que $\|Au - b\| < \|Ax - b\|$ para cada $x \neq u$ y daremos un método sencillo para calcular u. La base del procedimiento es el bien conocido teorema de la proyección, cuya prueba puede encontrarse en muchos textos introductorios al Álgebra Lineal:

Teorema 8.3.1 *Sean V un espacio vectorial euclídeo sobre $\mathbb{R}$ y W un subespacio finitodimensional de V. Sea $v \in V$. Entonces existe $w \in W$ tal que $\|w - v\| < \|w - y\|$ para cada $y \in W$, $y \neq w$. Este vector w está caracterizado por la propiedad de que $w - v$ es ortogonal a W, es decir, $\langle w - v, y\rangle = 0$ para cada $y \in W$.*

Ahora procedemos como sigue. Véase $A : \mathbb{R}^n \to \mathbb{R}^m$ como una aplicación lineal y pongamos $V = \mathbb{R}^m$, $W = A(\mathbb{R}^n)$. El Teorema 8.3.1 nos dice que existe $w \in W$ tal que $\|w - b\| < \|y - b\|$ para cada $y \in W$ distinto de w o, equivalentemente, por ser A inyectiva (recordemos que su rango es n), que existe $u \in \mathbb{R}^n$ (el único con la propiedad $Au = w$) tal que $\|Au - b\| < \|Ax - b\|$ para cada $x \in \mathbb{R}^n$, $x \neq u$.

Más aún, sabemos que w está caracterizado por la propiedad de que $w - b$ es ortogonal a W, es decir, $\langle Au - b, Ax\rangle = 0$ para cada $x \in \mathbb{R}^n$. Dado que $\langle A^T c, d\rangle = \langle c, Ad\rangle$ para cada $c \in \mathbb{R}^m$, $d \in \mathbb{R}^n$, ello equivale a decir que $\langle A^T Au - A^T b, x\rangle = 0$ para cada $x \in \mathbb{R}^n$ o, lo que es lo mismo, $A^T Au - A^T b = 0$. En otras palabras, u será la solución del sistema (de n ecuaciones y n incógnitas)

$$A^T Ax = A^T b,$$

que es la llamada *ecuación normal* asociada al sistema sobredeterminado. Conviene enfatizar que la ecuación tiene solución única pues la matriz $A^T A$ es regular. En efecto, $A^T Ax = 0$ implica $\langle A^T Ax, x\rangle = 0$ y por tanto $\|Ax\|^2 = \langle Ax, Ax\rangle = 0$. Esto significa que $Ax = 0$ y así, por la linealidad de A, $x = 0$.

Resumimos nuestras conclusiones en el siguiente teorema:

Teorema 8.3.2 *Sean $n < m$, $A \in \mathcal{M}_{m\times n}(\mathbb{R})$ y $b \in \mathbb{R}^m$. Supongamos que A tiene rango n. Entonces existe $u \in \mathbb{R}^n$ tal que $\|Au - b\| < \|Ax - b\|$ para cada $x \in \mathbb{R}^n$, $x \neq u$. El vector u es la única solución de la ecuación normal $A^T Ax = A^T b$.*

Estamos ahora en disposición de responder a la cuestión que abría el epígrafe. Recordemos que buscamos el polinomio $p(x) = a_0 + a_1 x + \cdots a_n x^n$ que mejor aproxima en norma euclídea a los puntos $\{(x_i, y_i)\}_{i=0}^m$, es decir, que minimiza la cantidad $\sum_{i=0}^m (y_i - p(x_i))^2$. Si consideramos

la matriz

$$A = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ & & \vdots & & \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{pmatrix}$$

y los vectores $x = (a_0, a_1, \ldots, a_n)$ y $b = (y_0, y_1, \ldots, y_m)$, el problema se reduce a buscar $u = (c_0, c_1, \ldots, c_n)$ tal que $\|Au - b\| < \|Ax - b\|$ para cada $x \neq u$ (nótese que estamos utilizamos la el término x con un doble sentido, como variable del polinomio y como vector de $\mathbb{R}^n$, pero esto no debería inducir a confusión). En otras palabras, el polinomio $p(x) = c_0 + c_1 x + \cdots c_n x^n$ buscado es aquel cuyo vector de coeficientes u es la única solución de la ecuación normal $A^T Ax = A^T b$. Naturalmente la discusión anterior carece de sentido a menos que la matriz A tenga rango $n+1$, pero esto es fácil de probar. Por ejemplo

$$A' = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ & & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$$

es regular porque el sistema

$$\begin{aligned} a_0 + x_0 a_1 + \cdots + x_0^n a_n &= y_0, \\ a_0 + x_1 a_1 + \cdots + x_1^n a_n &= y_1, \\ &\cdots \\ a_0 + x_n a_1 + \cdots + x_n^n a_n &= y_n \end{aligned}$$

tiene solución única (los números $a_0, a_1, \ldots a_n$ son los coeficientes del polinomio interpolador en $\{(x_i, y_i)\}_{i=0}^n$, que como sabemos existe y es único).

Veamos a continuación un ejemplo ilustrativo.

Ejemplo 8.3.3 Consideremos la tabla de puntos

x	1	2	3	4	5	6	7	8	9
y	2.1	3.3	3.9	4.4	4.6	4.8	4.6	4.2	3.4

y busquemos el polinomio $p(x) = a + bx + cx^2$ que mejor la aproxime. El sistema sobredimensionado a considerar es $Ax \approx d$, donde

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \\ 1 & 6 & 36 \\ 1 & 7 & 49 \\ 1 & 8 & 64 \\ 1 & 9 & 81 \end{pmatrix},$$

$x = (a, b, c)$ y $d = (2.1, 3.3, 3.9, 4.4, 4.6, 4.8, 4.6, 4.2, 3.4)$. Los coeficientes (a, b, c) buscados serán la solución de la ecuación normal $A^T Ax = A^T d$, donde

$$A^T A = \begin{pmatrix} 9 & 45 & 285 \\ 45 & 285 & 2025 \\ 285 & 2025 & 15333 \end{pmatrix},$$

y $A^T d = (35.3, 186.2, 1178.2)$. Resolviendo la ecuación normal resulta $a = 0.9333$, $b = 1.3511$, $c = -0.1189$.

En la práctica usaremos alguno de los métodos de resolución de sistemas lineales ya conocidos para resolver la ecuación normal. Notemos que la matriz $A^T A$ es siempre simétrica y definida positiva (ya que si $x \neq 0$ entonces $\langle A^T Ax, x\rangle = \langle Ax, Ax\rangle = \|Ax\|^2 > 0$) lo que hace que estemos en un contexto óptimo para usar el método de Cholesky.

Con adecuadas modificaciones del método de aproximación por mínimos cuadrados descrito podemos aproximar familias de puntos por funciones no necesariamente polinómicas.

Ejemplo 8.3.4 Consideremos la tabla de puntos

x	1	2	3	4
y	7	11	17	27

y buscamos la función $y = ae^{bx}$ que mejor la aproxime. Tomando logaritmos y escribiendo $c = \log a$ tendríamos

$$\log y = \log a + bx = c + bx,$$

con lo que podemos encontrar c y b aplicando nuestro método a la tabla de datos

x	1	2	3	4
$\log y$	$\log 7$	$\log 11$	$\log 17$	$\log 27$

y una vez obtenidos c y b, calculamos $a = e^c$. Concretamente tendremos

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{pmatrix},$$

$x = (c, b)$ y $d = (\log 7, \log 11, \log 17, \log 27)$. Entonces

$$A^T A = \begin{pmatrix} 4 & 10 \\ 10 & 30 \end{pmatrix}$$

y

$$A^T d = (\log 35343, \log 2211491279151) = (10.4729\ldots, 28.4227\ldots),$$

obteniéndose $c = 1.497$ y $b = 0.485$ como solución de la ecuación normal $A^T Ax = A^T d$. Finalmente $a = e^c = 4.468$.

Es importante resaltar que en el anterior ejemplo *no* estamos afirmando que la función $f(x) = 4.468e^{0.485x}$ sea aquella entre las de la forma $f(x) = ae^{bx}$ que minimiza $(f(1) - 7)^2 + (f(2) - 11)^2 + (f(3) - 17)^2 + (f(4) - 27)^2$. El último ejemplo de la sección ilustra con más énfasis esta cuestión.

Ejemplo 8.3.5 Se supone que el cometa Tentax, descubierto el año 1968, es un objeto del Sistema Solar. En cierto sistema de coordenadas polares (r, φ), centrado en el Sol, se han medido experimentalmente las siguientes posiciones del cometa:

r	2.20	2.00	1.61	1.20	1.02
φ	48^o	67^o	83^o	108^o	126^o

Si se desprecian las perturbaciones de los planetas, las leyes de Kepler garantizan que el cometa se moverá en una órbita elíptica, parabólica o hiperbólica, que en dichas coordenadas polares tendrá en cualquier caso la ecuación

$$r = \frac{p}{1 + e\cos\varphi},$$

donde p es un parámetro y e la excentricidad. Ajustemos por mínimos cuadrados los valores p y e a partir de las medidas hechas.

A partir de los datos dados hay varias maneras de formular un problema de mínimos cuadrados, todas válidas pero no todas equivalentes entre sí. Mostramos a continuación dos posibles formas de hacerlo.

Posibilidad 1. Despejando en la ecuación

$$r = \frac{p}{1 + e\cos\varphi}$$

llegamos a

$$r + er\cos\varphi = p$$

y de aquí

$$r = p + e(-r\cos\varphi).$$

Por tanto se trata de minimizar $\|Ax - b\|$, donde

$$A = \begin{pmatrix} 1 & -2.20\ \cos 48^o \\ 1 & -2.00\ \cos 67^o \\ 1 & -1.61\ \cos 83^o \\ 1 & -1.20\ \cos 108^o \\ 1 & -1.02\ \cos 126^o \end{pmatrix} = \begin{pmatrix} 1 & -1.47209 \\ 1 & -0.78146 \\ 1 & -0.19621 \\ 1 & 0.37082 \\ 1 & 0.59954 \end{pmatrix},$$

$$b = \begin{pmatrix} 2.20 \\ 2.00 \\ 1.61 \\ 1.20 \\ 1.02 \end{pmatrix}$$

y

$$x = \begin{pmatrix} p \\ e \end{pmatrix}.$$

Dado que A tiene rango 2 tiene sentido plantear la ecuación normal $A^TAx = A^Tb$, es decir,

$$\begin{pmatrix} 5 & -1.47940 \\ -1.47940 & 3.31318 \end{pmatrix} \begin{pmatrix} p \\ e \end{pmatrix} = \begin{pmatrix} 8.03 \\ -4.06090 \end{pmatrix}$$

cuya solución $p = 1.43262$ y $e = -0.58599$ nos da los valores de p y e buscados (obteniéndose $\|Ax - b\| = 0.22686$ como medida de la aproximación).

Posibilidad 2. Dividiendo en

$$r + er\cos\varphi = p$$

por er y despejando se obtiene

$$\cos\varphi = (-1/e) + (p/e)(1/r).$$

Se trata ahora de minimizar $\|Ax - b\|$, donde

$$A = \begin{pmatrix} 1 & 1/2.20 \\ 1 & 1/2 \\ 1 & 1/1.61 \\ 1 & 1/1.20 \\ 1 & 1/1.02 \end{pmatrix} = \begin{pmatrix} 1 & 0.45454 \\ 1 & 0.5 \\ 1 & 0.62112 \\ 1 & 0.83333 \\ 1 & 0.98039 \end{pmatrix},$$

$$b = \begin{pmatrix} \cos 48^{\circ} \\ \cos 67^{\circ} \\ \cos 83^{\circ} \\ \cos 108^{\circ} \\ \cos 126^{\circ} \end{pmatrix} = \begin{pmatrix} 0.66913 \\ 0.39073 \\ 0.12187 \\ -0.30902 \\ -0.58779 \end{pmatrix}$$

y

$$x = \begin{pmatrix} c \\ d \end{pmatrix}$$

con $c = -1/e$ y $d = p/e$. A partir de la ecuación normal $A^T Ax = A^T b$, es decir,

$$\begin{pmatrix} 5 & 3.38939 \\ 3.38939 & 2.49801 \end{pmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} 0.28493 \\ -0.25856 \end{pmatrix},$$

obtenemos $c = 1.58479$ y $d = -2.25381$, de donde $p = 1.42215$ y $e = -0.63100$ (obteniéndose ahora $\|Ax - b\| = 0.29359$ como medida de la aproximación).

8.4. Aproximación uniforme

A la vista del Teorema 8.1.4 parece bastante natural pensar que f es una función con buenas propiedades de derivabilidad y elegimos los puntos de la interpolación de la manera natural

$$x_i = a + \frac{(b-a)i}{n}$$

(*nodos equidistribuidos*) entonces los correspondientes polinomios interpoladores de f convergerán uniformemente a la función. Sin embargo, como demostró Runge en 1901, éste no es ni mucho menos el caso: un sorprendente ejemplo es la función $f(x) = 1/(x^2 + 1)$ vista en el intervalo $[-5, 5]$. Sin embargo puede probarse que eligiendo en el intervalo $[a, b] = [-5, 5]$ los nodos según la regla

$$x_i = a + \frac{b-a}{2}\left(1 + \cos\left(\frac{2i+1}{2n+2}\,\pi\right)\right)$$

(*nodos de Chebyshev*) entonces los polinomios interpoladores correspondientes convergen uniformemente a f. Como veremos a continuación, los nodos de Chebyshev tienen la interesante

propiedad que hacen la cantidad $\max_{x\in[a,b]} |\prod_{i=0}^{n}(x-x_i)|$ lo más pequeña posible (compárese con el Teorema 8.1.4) pero ni siquiera para estos nodos hay garantía de convergencia: Faber probó en 1914 que para cualquier sistema de nodos $a \le x_0^{(n)} < x_1^{(n)} < \cdots < x_n^{(n)} \le b$ existe una función continua f de manera que si $p_n \in \Pi_n$ es el polinomio interpolador en los puntos $(x_i^{(n)}, f(x_i^{(n)}))_{i=0}^{n}$, entonces (p_n) *no* converge uniformemente a f.

Probamos a continuación la mencionada propiedad de los nodos de Chebyshev, para lo que seguiremos [3, pp. 292–293].

Para empezar consideramos el caso $[a,b] = [-1,1]$ introduciendo inductivamente los llamados *polinomios de Chebyshev*:

$$T_0(x) = 1,$$

$$T_1(x) = x,$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \ge 1.$$

Los polinomios de Chebyshev tienen la siguiente útil propiedad:

Proposición 8.4.1 *Para $x \in [-1,1]$ los polinomios de Chebyshev admiten la siguiente expresión:*

$$T_n(x) = \cos(n \cos^{-1} x), \quad n \ge 0.$$

Demostración: Partiendo de la conocida igualdad trigonométrica

$$\cos(A+B) = \cos A \cos B - \operatorname{sen} A \operatorname{sen} B$$

deducimos

$$\cos(n+1)\theta = \cos\theta \cos n\theta - \operatorname{sen}\theta \operatorname{sen} n\theta,$$

$$\cos(n-1)\theta = \cos\theta \cos n\theta + \operatorname{sen}\theta \operatorname{sen} n\theta,$$

de donde

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

Usemos la biyección que la función cos establece entre los intervalos $[0,\pi]$ y $[-1,1]$, escribamos $\cos\theta = x$, $\theta = \cos^{-1} x$, y apliquemos la última igualdad para deducir que las funciones $f_n(x) = \cos(n\cos^{-1} x)$ (con dominio de definición el intervalo $[-1,1]$) satisfacen, además de las obvias propiedades $f_0(x) = 0$ y $f_1(x) = x$, la igualdad

$$f_{n+1}(x) = 2xf_n(x) - f_{n-1}(x), \quad n \ge 1.$$

De aquí concluimos que $T_n = f_n$ para todo n. □

A partir de la formula de la proposición anterior es fácil obtener las siguientes propiedades adicionales de los polinomios de Chebyshev para todo $n \ge 1$:

$$|T_n(x)| \le 1, \quad x \in [-1,1],$$

$$T_n(\cos(j\pi/n)) = (-1)^j, \quad 0 \le j \le n,$$

$$T_n(\cos((2j-1)\pi/(2n))) = 0, \quad 1 \le j \le n.$$

En particular, dado que (fijo n), se cumple $|T_n(c_j)| = 1$ para los puntos

$$c_j = \cos(j\pi/n), \quad 0 \le j \le n,$$

se tiene que

$$\|T_n\|_\infty = \max_{x\in[-1,1]} |T_n(x)| = 1.$$

Asimismo, observemos que los puntos

$$r_j = \cos((2j-1)\pi/(2n)), \quad 1 \le j \le n,$$

son exactamente los ceros del polinomio T_n.

Decimos que un polinomio $p \in \Pi_n$ es *mónico* si su coeficiente principal es 1. De la definición inicial de los polinomios de Chebyshev se deduce que el coeficiente principal de T_n es 2^{n-1}, es decir, $P_n(x) = 2^{1-n}T_n(x)$ es mónico. Claramente $\|P_n\|_\infty = 2^{1-n}$. Resulta que no hay polinomio mónico que tenga una norma menor:

Proposición 8.4.2 *Si $p \in \Pi_n$ es mónico, entonces $\|p\|_\infty \ge 2^{1-n}$.*

Demostración: Supongamos que $|p(x)| < 2^{1-n}$ para todo $x \in [-1,1]$. Entonces

$$(-1)^j p(c_j) \le |p(c_j)| < 2^{1-n} = (-1)^j P_n(c_j), \quad 0 \le j \le n,$$

con lo que

$$(-1)^j (p(c_j) - P_n(c_j)) < 0, \quad 0 \le j \le n.$$

Así pues, el polinomio $p - P_n$ toma alternativamente valores negativos y positivos en los $n+1$ puntos c_j, lo que significa que tiene al menos n ceros distintos, cosa imposible dado que p y P_n son mónicos y por tanto $p - P_n$ tiene grado a lo sumo $n-1$. □

En este punto retomemos el Teorema 8.1.4. Si $g : [-1,1] \to \mathbb{R}$ es de clase C^{n+1}, $\{s_i\}_{i=0}^n$ es una familia de $n+1$ puntos distintos en $[-1,1]$ y $t \in \Pi_n$ es el polinomio interpolador en $\{s_i, g(s_i)\}_{i=0}^n$, la fórmula del teorema puede reescribirse como

$$\|g - t\|_\infty \le \frac{\|g^{(n+1)}\|_\infty}{(n+1)!}\|q\|_\infty,$$

donde $q(x) = \prod_{i=0}^n (x - s_i)$. Como q es mónico de grado $n+1$, la Proposición 8.4.2 nos dice que $\|q\|_\infty \ge 2^{-n}$. Si elegimos los puntos s_i como los nodos de Chebyshev, es decir, como los ceros r_j del polinomio de Chebyshev de grado $n+1$,

$$s_i = \cos((2i+1)\pi/(2n+2)), \quad 0 \le i \le n,$$

se tiene exactamente $\|q\|_\infty = 2^{-n}$. En consecuencia hemos probado:

Teorema 8.4.3 *Sean $g : [-1,1] \to \mathbb{R}$ una función de clase C^{n+1}, $\{s_i\}_{i=0}^n$ los nodos de Chebyshev y $t \in \Pi_n$ el correspondiente polinomio interpolador. Entonces*

$$\|g - t\|_\infty \le \frac{\|g^{(n+1)}\|_\infty}{2^n (n+1)!}.$$

Es fácil probar un resultado análogo para una función $f : [a,b] \to \mathbb{R}$ de clase C^{n+1} definida en un intervalo arbitrario $[a,b]$ y el polinomio interpolador $p \in \Pi_n$ en los nodos de Chebyshev generalizados

$$x_i = a + \frac{b-a}{2}\left(1 + \cos\left(\frac{2i+1}{2n+2}\,\pi\right)\right), \quad 0 \le i \le n.$$

En efecto, si $y = \phi(x) = a + \frac{b-a}{2}(1+x)$ es la biyección lineal entre los intervalos $[-1,1]$ y $[a,b]$, definimos $g(x) = f(\phi(x))$ y denotamos por $t(x)$ al polinomio interpolador en los puntos $\{s_i, g(s_i)\}$, es fácil comprobar que $t(x) = p(\phi(x))$. Dado que

$$\|f-p\|_\infty = \max_{y\in[a,b]} |f(y)-p(y)| = \max_{x\in[-1,1]} |g(x)-t(x)| = \|g-t\|_\infty$$

y

$$\|g^{(n+1)}\|_\infty = \frac{(b-a)^{n+1}}{2^{n+1}}\|f^{(n+1)}\|_\infty,$$

deducimos del Teorema 8.4.3 que

$$\|f-p\|_\infty \leq \frac{(b-a)^{n+1}\|f^{(n+1)}\|_\infty}{2^{2n+1}(n+1)!}.$$

Demostramos a continuación que si partimos de una función continua dada y elegimos adecuadamente el sistema de nodos, entonces los polinomios interpoladores resultantes convergen uniformemente a la función (teorema de Marcinkiewicz). Seguimos la Sección 1.7 (pp. 35–42) del libro de Crouzeix y Mignot [2].

El punto de partida es el famoso teorema de aproximación de Weierstrass, que garantiza que las funciones continuas pueden aproximarse uniformemente por polinomios (aunque no da información acerca de la naturaleza y propiedades de dichos polinomios). Traducimos a continuación una prueba bastante asequible que hemos encontrado en el sitio de internet `http://planetmath.org`. Puede encontrarse una demostración alternativa en [3, pp. 296–299].

Teorema 8.4.4 (Teorema de aproximación de Weierstrass) *Sean $f : [a,b] \to \mathbb{R}$ y $\epsilon > 0$. Entonces existe un polinomio p tal que $\|f-p\|_\infty < \epsilon$.*

Demostración: Para simplificar la notación suponemos que estamos trabajando en el intervalo $[0,1]$. Esto no implica pérdida de generalidad porque si estuviésemos trabajando en otro intervalo, podríamos hacer un cambio de coordenadas lineal (o hablando con más propiedad, afín) que lleve el dominio de f a $[0,1]$.

El caso $f(x) = 1-\sqrt{1-x}$. Comenzamos demostrando unos cuantos casos particulares del teorema, empezando con la función $f(x) = 1-\sqrt{1-x}$. En este caso podemos usar el antiguo método babilónico de calculo de raíces cuadradas para construir polinomios aproximadores. Definimos los polinomios $P_0, P_1, P_2, \ldots$ recursivamente mediante

$$P_0(x) = 0,$$

$$P_{n+1}(x) = \frac{1}{2}\left(P_n(x)^2 + x\right).$$

Es consecuencia obvia de la definición que si $0 \leq x \leq 1$ entonces $0 \leq P_n(x) \leq 1$ para todo n. Es igualmente obvio que P_n es una función creciente en el intervalo $[0,1]$. Restando las fórmulas recursivas para $n+1$ y $n+2$, cancelando y factorizando obtenemos la relación

$$P_{n+2}(x) - P_{n+1}(x) = \frac{1}{2}(P_{n+1}(x)+P_n(x))(P_{n+1}(x)-P_n(x)).$$

De ésta concluimos que $P_{n+1}(x) \geq P_n(x)$ para todo n y todo x en $[0,1]$. Esto implica que $\lim_{n\to\infty} P_n(x)$ existe para todo x en $[0,1]$. Tomando límites en ambos lados de la recursión que define P_n y simplificando, vemos que $\lim_{n\to\infty} P_n(x) = 1-\sqrt{1-x}$. La relación implica también que $P_{n+1}(x) - P_n(x)$ es una función creciente of x en el intervalo $[0,1]$ para todo n. Por tanto

$$P_{n+1}(x) - P_n(x) \leq P_{n+1}(1) - P_n(1).$$

Sumando en n y cancelando, vemos que

$$P_m(x) - P_n(x) \leq P_m(1) - P_n(1)$$

siempre que $m > n$. Tomando el límite cuando m tiende a infinito, concluimos que

$$1 - \sqrt{1-x} - P_n(x) \leq 1 - P_n(1).$$

Dado que los polinomios P_n convergen, para cada $\epsilon > 0$, existe n tal que $1 - P_n(1) < \epsilon$. Para este n, $|f(x) - P_n(x)| < \epsilon$, así que el teorema de aproximación de Weierstrass se cumple en este caso. Obsérvese que se tiene igualmente que $|\sqrt{y} - q(y)| < \epsilon$ para todo $y \in [0,1]$, donde $q(y)$ es el polinomio $1 - P_n(1-y)$.

El caso $f(x) = |x - c|$. A continuación consideramos el caso especial $f(x) = |x - c|$, donde $0 < c < 1$. Usando que $\sqrt{a^2 + b^2} \leq a + b$ obtenemos que

$$\sqrt{(x-c)^2 + \epsilon^2/4} - |x - c| \leq \epsilon/2.$$

Si ϵ es suficientemente pequeño para que $(x-c)^2 + \epsilon^2/4$ esté en el intervalo $[0,1]$ para cada $x \in [0,1]$, podemos usar el caso del teorema de aproximación ya probado y encontrar un polinomio P tal que

$$|\sqrt{(x-c)^2 + \epsilon^2/4} - P(x)| < \epsilon/2$$

cuando $x \in [0,1]$. Combinando las dos desigualdades y aplicando la desigualdad triangular se llega a $|f(x) - P(x)| < \epsilon$, y por tanto hemos probado el teorema de aproximación de Weierstrass para el caso $f(x) = |x - c|$. Nótese que si esta función puede aproximarse uniformemente por polinomios lo mismo ocurrirá para $g(x) = a\frac{|x-c|+(x-c)}{2}$. Observemos que esta función se anula en el intervalo $[0,c]$ y es lineal con pendiente a en el intervalo $[c,1]$.

El caso de las funciones lineales a trozos. Como corolario del resultado que acabamos de probar se obtiene que el teorema también se cumple para las funciones lineales a trozos. Para comprenderlo basta razonar por inducción, suponiendo que las funciones que constan de $n-1$ trozos lineales pueden aproximarse uniformemente por polinomio, y probándolo para las que constan de n trozos (las que constan de un único trozo son directamente polinomios de grado cero o uno).

Sea pues $f(x)$ una función que consta de $n-1$ trozos lineales y sea $0 < c < 1$ el último punto en el que la función f no es derivable. Supongamos que en los trozos lineales adyacentes a c la función f tiene pendientes a_i y a_d respectivamente y modifiquemos f en el intervalo $[c,1]$ de manera que la pendiente en dicho intervalo sea también a_i. De esta manera construimos una función g con $n-1$ trozos lineales, para que existirá un polinomio P_1 tal que $|g(x) - P_1(x)| < \epsilon/2$. Según el caso anterior, si h es la función que se anula en $[0,c]$ y es lineal con pendiente $a_d - a_i$ en $[c,1]$, entonces existe otro polinomio $P_2(x)$ tal que $|h(x) - P_2(x)| < \epsilon/2$. Claramente, $f(x) = g(x) + h(x)$. Entonces $|f(x) - P(x)| < \epsilon$ para $P = P_1 + P_2$.

El caso general. A la vista del caso anterior basta demostrar que si f es continua en [0,1] entonces para todo $\epsilon > 0$ existe una función lineal a trozos ϕ tal que $|f(x) - \phi(x)| < \epsilon/2$ para cada $x \in [0,1]$. En efecto, si tal función existe entonces también existe un polinomio P tal que $|\phi(x) - P(x)| < \epsilon/2$, y entonces $|f(x) - P(x)| < \epsilon$.

Dado que $[0,1]$ es compacto, f es uniformemente continua. Por tanto, para cada $\epsilon > 0$ existe un entero N tal que $|f(x) - f(y)| < \epsilon/2$ siempre que $|x - y| \leq 1/N$.

Definamos ϕ conforme a las siguientes condiciones: si m es un entero entre 0 y N, $\phi(m/N) = f(m/N)$; en cada uno de los intervalos $[m/N, (m+1)/N]$, ϕ es lineal.

Para cada punto x del intervalo $[0,1]$, existe un entero m tal que x pertenece al intervalo $[m/N,(m+1)/M]$. Dado que una función lineal está acotada por los valores en los extremos, $\phi(x)$ está entre $\phi(m/N)=f(m/N)$ y $\phi((m+1)/N)=f((m+1)/N)$. Dado que $|f(m/N)-f((m+1)/N)|\leq\epsilon/2$, se tiene que $|f(m/N)-\phi(x)|<\epsilon/2$. Como $|x-m/N|\leq 1/N$, también se cumple $|f(m/N)-f(x)|<\epsilon/2$. Por la desigualdad triangular, $|f(x)-\phi(x)|<\epsilon$. □

Lema 8.4.5 *Sea $f:[a,b]\to\mathbb{R}$ una función continua y sean $n+2$ puntos distintos $x_0<x_1<\cdots<x_n<x_{n+1}$ en $[a,b]$. Entonces existe un único polinomio $p\in\Pi_n$ tal que*

$$f(x_i)-p(x_i)=(-1)^i(f(x_0)-p(x_0)) \quad \textit{para cada } i=1,2,\ldots,n+1. \tag{8.4}$$

Más aún, este polinomio se caracteriza por la propiedad

$$\max_{0\leq i\leq n+1}|f(x_i)-p(x_i)|<\max_{0\leq i\leq n+1}|f(x_i)-q(x_i)|, \quad q\in\Pi_n,\ q\neq n. \tag{8.5}$$

Demostración: Nótese que las ecuaciones de (8.4) constituyen un sistema de $n+1$ ecuaciones lineales en las $n+1$ incógnitas $a_0,a_1,\ldots,a_n$ de los coeficientes del polinomio $p(x)=a_0+a_1x+\cdots+a_nx^n$ buscado. Demostramos a continuación que dicho polinomio, de existir, verifica (8.5). Esto es suficiente para concluir la afirmación del lema porque garantiza que la solución del sistema, de existir, es única. En efecto, si f es la función nula, el polinomio nulo (que obviamente es solución) será la única solución del sistema, lo que equivale a decir que la matriz de coeficientes del sistema lineal dado por (8.4) (que no depende de f) es regular.

Probemos pues que de (8.4) se concluye (8.5). Sea $q\in\Pi_n$ con $q\neq p$ y supongamos, por reducción al absurdo, que

$$\max_{0\leq i\leq n+1}|f(x_i)-q(x_i)|\leq\max_{0\leq i\leq n+1}|f(x_i)-p(x_i)|=|f(x_0)-p(x_0)|.$$

No es restrictivo suponer que $f(x_0)-p(x_0)\geq 0$. Entonces

$$\begin{aligned}(-1)^i(q(x_i)-p(x_i)) &= (-1)^i(f(x_i)-p(x_i))-(-1)^i(p(x_i)-q(x_i))\\ &\geq f(x_0)-p(x_0)+|p(x_i)-q(x_i)|\\ &\geq 0,\end{aligned}$$

para cada i, lo que implica que

$$\begin{aligned}(-1)^i\int_{x_i}^{x_{i+1}}q'(x)-p'(x)\,dx &= (-1)^i\left[q(x)-p(x)\right]_{x_i}^{x_{i+1}}\\ &= (-1)^i(q(x_{i+1})-p(x_{i+1}))-(-1)^i(q(x_i)-p(x_i))\\ &\leq 0.\end{aligned}$$

Ahora existen dos posibilidades. Si $q'-p'\equiv 0$ en alguno de los intervalos $[x_i,x_{i+1}]$ entonces, por ser $q'-p'$ un polinomio, tendremos que $q'-p'\equiv 0$ en todo $\mathbb{R}$, con lo que $q-p$ es constante. Como el signo de $q-p$ va cambiando en los puntos x_i, no hay más alternativa que $q=p$, contradicción.

Concluimos entonces que existe $\xi_i\in(x_i,x_{i+1})$ tal que $(-1)^i(q'(\xi_i)-p'(\xi_i))<0$ para cada $i=0,1,\ldots,n$, con lo que por la propiedad de los valores intermedios $q'-p'$ admitirá una raíz en cada uno de los intervalos (ξ_i,ξ_{i+1}), $i=0,1,\ldots,n-1$. Esto hace un total de al menos n raíces para el polinomio (no nulo) $q'-p'$, que tiene a lo sumo grado $n-1$. Hemos llegado de nuevo a una contradicción. □

Algoritmo 8.5 **Algoritmo de primera equioscilación**

Datos de entrada:
$n+2$ (número de puntos para la primera equioscilación);
$x[n+2]$ (abscisas de los puntos para la primera equioscilación);
$y[n+2]$ (ordenadas de los puntos para la primera equioscilación);

Variables:
$e[n+1]$; // coeficientes a calcular
$b[n+1]$; // vector de términos independientes
$S[n+1][n+1]$; // matriz del sistema a resolver
signo; // variable auxiliar para los cambios de signo
aux0, aux1: // variables auxiliares

Flujo del programa:
signo $= 1$;
`for(i=1;i<=n+1;i++){`
  signo $= -$signo;
  $b_{i-1} = y_0 -$ signo $* y_i$;
  `for(j=0;j<=n;j++){`
    aux0 $= 1$;
    aux1 $= 1$;
    $S_{i-1,j} =$ aux0 $-$ signo $*$ aux1;
    aux0 $=$ aux0 $* x_0$;
    aux1 $=$ aux1 $* x_i$;

  }
}
$e = S^{-1}b$;

Datos de salida: Coeficientes $e_0, e_1, \ldots, e_n$ del primer polinomio equioscilador

Se dice que una función continua $h : [a,b] \to \mathbb{R}$ *equioscila* en los puntos $x_0 < x_1 < \cdots < x_n < x_{n+1}$ si $\|h\|_\infty = |h(x_i)|$, $i = 0, 1, \ldots, n+1$, y $h(x_i) = -h(x_{i+1})$, $i = 0, 1, \ldots, n$. Si $p \in \Pi_n$ es un polinomio tal que $f - p$ equioscila en $x_0 < x_1 < \cdots < x_n < x_{n+1}$ y $q \in \Pi_n$ es distinto de p entonces, de acuerdo con (8.5),

$$\|f-p\|_\infty = \max_{0\le i\le n+1} |f(x_i)-p(x_i)| < \max_{0\le i\le n+1} |f(x_i)-q(x_i)| \le \|f-q\|_\infty.$$

Ello significa que p es el polinomio de Π_n que mejor aproxima a f en la norma uniforme (o, como se dice a veces, en el *sentido de Chebyshev*). Probamos a continuación que, en efecto, existe un polinomio $p \in \Pi_n$ tal que $f - p$ equioscila en ciertos puntos $x_0 < x_1 < \cdots < x_n < x_{n+1}$ del intervalo $[a,b]$. Este polinomio se construye como límite de una sucesión de polinomios construidos mediante el llamado *(segundo) algoritmo de Rémès*. Este algoritmo funciona como sigue.

Algoritmo 8.6 **Calculo del máximo absoluto de una función**

Datos de entrada:
malla (tamaño de la malla de puntos);
h (función cuyo máximo se quiere calcular);
a (extremo izquierdo del intervalo);
b (extremo derecho del intervalo);

Variables:
max; // punto donde la función $|h|$ alcanza el máximo
y; // variable auxiliar

Flujo del programa:
max $= a$;
`for(i=1;i<=malla;i++){`
  $y = a + (b-a)i/$malla;
  `if(`$|h(y)| > |h($`max`$)|$`){`
    max $= y$;
  }
}

Datos de salida: Punto max donde la función $|h|$ alcanza el máximo

Inicialización. Partimos de puntos $a \le x_0^0 < x_1^0 < \cdots < x_n^0 < x_{n+1}^0 \le b$ arbitrariamente fijados (por ejemplo, igualmente distribuidos en $[a, b]$).

Etapa k del algoritmo. Supongamos conocidos los puntos

$$a \le x_0^k < x_1^k < \cdots < x_n^k < x_{n+1}^k \le b.$$

A estos puntos les asociamos el polinomio $p_k \in \Pi_n$ tal que

$$f(x_i^k) - p_k(x_i^k) = (-1)^i (f(x_0^k) - p_k(x_0^k)) \quad \text{para cada } i = 1, 2, \ldots, n+1;$$

el Lema 8.4.5 garantiza que el polinomio p_k está bien definido. Ahora aparecen dos casos:

Primer caso: $\|f - p_k\|_\infty = |f(x_i^k) - p_k(x_i^k)|$. En este caso $f - p_k$ equioscila en los puntos x_i^k, con lo que p_k es la mejor aproximación de f en norma uniforme que andábamos buscando; el algoritmo acaba.

Segundo caso: existe $y \in [a, b]$ tal que

$$|f(y) - p_k(y)| = \|f - p_k\|_\infty > |f(x_i^k) - p_k(x_i^k)|, \quad i = 1, 2, \ldots, n+1.$$

Entonces se construye una nueva sucesión de puntos

$$a \le x_0^{k+1} < x_1^{k+1} < \cdots < x_n^{k+1} < x_{n+1}^{k+1} \le b$$

reemplazando uno de los puntos x_j^k por y de manera que los signos de $f(x_i^{k+1}) - p_k(x_i^{k+1})$ se vayan alternando, es decir,

$$(f(x_{i+1}^{k+1}) - p_k(x_{i+1}^{k+1})(f(x_i^{k+1}) - p_k(x_i^{k+1})) \le 0 \quad \text{para cada } i = 1, 2, \ldots, n.$$

En concreto, aparecen seis posibilidades:

(a) Si $y \in [a, x_0^k)$ y $(f(x_0^k) - p_k(x_0^k))(f(y) - p_k(y)) \geq 0$, tomamos

$$x_0^{k+1} = y \qquad \text{y} \qquad x_i^{k+1} = x_i^k, \quad i = 1, 2, \ldots, n+1.$$

(b) Si $y \in [a, x_0^k)$ y $(f(x_0^k) - p_k(x_0^k))(f(y) - p_k(y)) < 0$, tomamos

$$x_0^{k+1} = y \qquad \text{y} \qquad x_i^{k+1} = x_{i-1}^k, \quad i = 1, 2, \ldots, n+1.$$

(c) Si $y \in (x_j^k, x_{j+1}^k)$ y $(f(x_j^k) - p_k(x_j^k))(f(y) - p_k(y)) \geq 0$, tomamos

$$x_j^{k+1} = y \qquad \text{y} \qquad x_i^{k+1} = x_i^k, \quad i \neq j.$$

(d) Si $y \in (x_j^k, x_{j+1}^k)$ y $(f(x_j^k) - p_k(x_j^k))(f(y) - p_k(y)) < 0$, tomamos

$$x_{j+1}^{k+1} = y \qquad \text{y} \qquad x_i^{k+1} = x_i^k, \quad i \neq j+1.$$

(e) Si $y \in (x_{n+1}^k, b]$ y $(f(x_{n+1}^k) - p_k(x_{n+1}^k))(f(y) - p_k(y)) \geq 0$, tomamos

$$x_{n+1}^{k+1} = y \qquad \text{y} \qquad x_i^{k+1} = x_i^k, \quad i = 0, 1, \ldots n.$$

(f) Si $y \in (x_{n+1}^k, b]$ y $(f(x_{n+1}^k) - p_k(x_{n+1}^k))(f(y) - p_k(y)) < 0$, tomamos

$$x_{n+1}^{k+1} = y \qquad \text{y} \qquad x_i^{k+1} = x_{i+1}^k, \quad i = 0, 1, \ldots n.$$

Algoritmo 8.7 **Algoritmo de Rémès**

Datos de entrada:
n; (grado del polinomio equioscilador);
malla (tamaño de la malla de puntos para el cálculo de la norma);
a (extremo izquierdo del intervalo);
b (extremo derecho del intervalo);
f (función a aproximar);
tol (tolerancia en la aproximación);
nmax (número máximo de iteraciones);

Variables:
$e[n+1]$; // coeficientes del polinomio equioscilador
$x[n+2]$; // abscisas de los puntos para la equioscilación
$y[n+2]$; // vector auxiliar
z; // variable auxiliar
j; // variable auxiliar

Flujo del programa:

```
for(i=0;i<=n+1;i++){
  x_i = a + (b - a)i/(n + 1);
}
for(k=1;k<=nmax;k++){
  for(i=0;i<=n+1;i++){
    y_i = f(x_i);
  }
  e = primeraEquioscilacion(x, y);
  // p : polinomio de coeficientes e_0, e_1, ..., e_n
  z = maximo(f - p, a, b, malla);
  if(|f(z) - p(z)| - |y_0 - p(x_0)| < tol){
    Parada: p es el polinomio equioscilador
  }
```

Algoritmo 8.7 **Algoritmo de Rémès (continuación)**

```
    else{
      if(z < x_0){
        j = -1;
      }
      if(z > x_{n+1}){
        j = n + 1;
      }
      for(i=0;i<=n;i++){
        if(z > x_i && z < x_{i+1}){
          j = i;
        }
      }
      if(j = -1){
        if((y_0 - p(x_0))(f(z) - p(z)) < 0){
          for(i=n+1;i>=1;i--){
            x_i = x_{i-1};
          }
        }
        x_0 = z;
      }
      if(j = n + 1){
        if((y_{n+1} - p(x_{n+1}))(f(z) - p(z)) < 0){
          for(i=0;i<=n;i++){
            x_i = x_{i+1};
          }
        }
        x_{n+1} = z;
      }
      if(j ≥ 0 && j < n + 1){
        if((y_j - p(x_j))(f(z) - p(z)) < 0){
          x_{j+1} = z;

        }
        else{
          x_j = z;
        }
      }
    }
  }
```

Parada: no hay convergencia en nmax iteraciones

Datos de salida: Coeficientes $e_0, e_1, \ldots, e_n$ del polinomio equioscilador o mensaje de error.

Demostraremos a continuación que si el algoritmo no se detiene entonces la sucesión de polinomios p_k converge al polinomio de mejor aproximación.

Lema 8.4.6 *Supongamos que el algoritmo de Rémès no se detiene en un número finito de iteraciones. Entonces*

$$\epsilon_k < \epsilon_{k+1} < \inf_{q\in\Pi_n} \|f-q\|_\infty \quad \textit{para todo } k \geq 0,$$

donde

$$\epsilon_k = \max_{0\leq i\leq n+1} |f(x_i^k) - p_k(x_i^k)|.$$

Demostración: Por el Lema 8.4.5 sabemos que

$$\epsilon_k = \max_{0\leq i\leq n+1} |f(x_i^k) - p_k(x_i^k)| \leq \max_{0\leq i\leq n+1} |f(x_i^k) - q(x_i^k)| \leq \|f-q\|_\infty \tag{8.6}$$

para cada $q \in \Pi_n$, así que $\epsilon_k \leq \inf_{q\in\Pi_n} \|f-q\|_\infty$ para cada k. Por tanto basta demostrar que la sucesión (ϵ_k) es estrictamente creciente.

Fijemos $k \geq 0$ y sea $x_{i_0}^{k+1}$ el nuevo término con relación a los x_i^k. No es restrictivo suponer

$$f(x_{i_0}^{k+1}) - p_k(x_{i_0}^{k+1}) = \|f-p_k\|_\infty \tag{8.7}$$

(el caso $-(f(x_{i_0}^{k+1}) - p_k(x_{i_0}^{k+1})) = \|f-p_k\|_\infty$ es análogo). Entonces, por la definición de p_k y la manera de elegir $x_{i_0}^{k+1}$, tenemos que

$$f(x_i^{k+1}) - p_k(x_i^{k+1}) = (-1)^{i-i_0}\epsilon_k, \quad i \neq i_0. \tag{8.8}$$

Asimismo

$$f(x_i^{k+1}) - p_{k+1}(x_i^{k+1}) = (-1)^{i-i_0}\overline{\epsilon_{k+1}}, \quad i = 0,1,\ldots,n+1, \tag{8.9}$$

siendo

$$f(x_{i_0}^{k+1}) - p_{k+1}(x_{i_0}^{k+1}) = \overline{\epsilon_{k+1}} = \pm\epsilon_{k+1}. \tag{8.10}$$

Si a (8.7) le restamos (8.10) se obtiene

$$p_{k+1}(x_{i_0}^{k+1}) - p_k(x_{i_0}^{k+1}) = \|f-p_k\|_\infty - \overline{\epsilon_{k+1}}. \tag{8.11}$$

Por otra parte, si a (8.8) le restamos (8.9) resulta

$$p_{k+1}(x_i^{k+1}) - p_k(x_i^{k+1}) = (-1)^{i-i_0}(\epsilon_k - \overline{\epsilon_{k+1}}), \quad i \neq i_0;$$

en otras palabras, si $q_k \in \Pi_n$ es el único polinomio que cumple $q_k(x_i^{k+1}) = (-1)^{i-i_0+1}$, $i \neq i_0$, entonces

$$p_{k+1} - p_k = (\overline{\epsilon_{k+1}} - \epsilon_k)q_k. \tag{8.12}$$

Finalmente, combinando (8.6) y (8.11) llegamos a

$$(\overline{\epsilon_{k+1}} - \epsilon_k)q_k(x_{i_0}^{k+1}) = \|f-p_k\|_\infty - \overline{\epsilon_{k+1}} \geq \inf_{q\in\Pi_n} \|f-q\|_\infty - \epsilon_{k+1} \geq 0. \tag{8.13}$$

Nótese que $q_k(x_{i_0}^{k+1}) > 0$. En efecto, si $q_k(x_{i_0}^{k+1}) < 0$ entonces habrían (al menos) $n+1$ cambios de signo de x_0 a x_{n+1} y por tanto $n+1$ ceros distintos, cosa imposible porque el grado de q_k es a lo sumo n. El mismo argumento permite descartar $q_k(x_{i_0}^{k+1}) = 0$ en los casos $i_0 = 0$ e $i_0 = n+1$, y concluir que $q_k(x) \geq 0$ para todo $x \in (x_{i_0-1}^{k+1}, x_{i_0+1}^{k+1})$ cuando $i_0 \in \{1,\ldots,n\}$. Por

tanto, si en este último caso, $q_k(x_{i_0}^{k+1}) = 0$, x_{i_0} es un mínimo relativo de q_k y podemos aplicar el teorema de Rolle para encontrar n ceros de la derivada de q_k, una contradicción.

Como $q_k(x_{i_0}^{k+1}) > 0$, (8.13) implica que $\overline{\epsilon_{k+1}} - \epsilon_k \geq 0$, de donde $\overline{\epsilon_{k+1}} = \epsilon_{k+1}$ y $\epsilon_{k+1} \geq \epsilon_k$. Finalmente, si $\epsilon_{k+1} = \epsilon_k$ entonces $p_{k+1} = p_k$ por (8.12), y aplicando (8.11) concluimos

$$\|f - p_k\|_\infty = \epsilon_{k+1} = \epsilon_k = \max_{0 \leq i \leq n+1} |f(x_i^k) - p_k(x_i^k)|,$$

y el algoritmo se habría detenido. □

Observación 8.4.7 Nótese que como consecuencia de (8.13) la demostración del Lema 8.4.6 también proporciona la cadena de desigualdades

$$0 \leq \|f - p_k\|_\infty - \epsilon_{k+1} \leq (\epsilon_{k+1} - \epsilon_k)\|q_k\|_\infty. \tag{8.14}$$

Lema 8.4.8 *Existe un número $\nu > 0$ tal que $|x_{i+1}^k - x_i^k| \geq \nu$ para cada $k \geq 0$, $i = 0, 1, \ldots, n$.*

Demostración: Supongamos lo contrario para encontrar una subsucesión $1 \leq k_1 < k_2 < \cdots < k_l < \cdots$ tal que $\lim_{l\to\infty} x_i^{k_l} = x_i$ para cada $i = 0, 1, \ldots, n+1$ y no todos los puntos x_i son distintos. Entonces podemos encontrar un polinomio interpolador en dichos puntos, es decir, existe $p \in \Pi_n$ tal que $p(x_i) = f(x_i)$ para cada i. Ahora bien, gracias a (8.5) y al Lema 8.4.6 sabemos que

$$0 \leq \epsilon_0 < \epsilon_1 \leq \epsilon_{k_l} = \max_{0 \leq i \leq n+1} |f(x_i^{k_l}) - p_{k_l}(x_i^{k_l})| \leq \max_{0 \leq i \leq n+1} |f(x_i^{k_l}) - p(x_i^{k_l})|$$

lo que es incompatible con

$$\lim_{l\to\infty} \max_{0 \leq i \leq n+1} |f(x_i^{k_l}) - p(x_i^{k_l})| = \max_{0 \leq i \leq n+1} |f(x_i) - p(x_i)| = 0.$$

□

Ya estamos listos para probar:

Teorema 8.4.9 *Sean $f : [a, b] \to \mathbb{R}$ continua y $n \geq 0$. Entonces existe un único polinomio p de mejor aproximación a f en Π_n para la norma uniforme. Este polinomio está caracterizado porque $f - p$ equioscila en $n + 2$ puntos de $[a, b]$. El algoritmo de Rémès proporciona o bien p tras un número finito de pasos o una sucesión de polinomios que converge uniformemente a p.*

Demostración: Debemos demostrar que la sucesión (p_k) de polinomios generada por el algoritmo converge al polinomio p buscado.

Con la notación del Lema 8.4.6 pongamos $\epsilon = \lim_{k\to\infty} \epsilon_k$. Escribamos cada uno de los polinomios q_k que allí introducíamos mediante la forma del polinomio interpolador de Lagrange, es decir,

$$q_k(x) = \sum_{i \neq i_0} (-1)^{i - i_0 + 1} \prod_{i_0 \neq j \neq i} \frac{x - x_j^{k+1}}{x_i^{k+1} - x_j^{k+1}}.$$

Usando el Lema 8.4.8 vemos que

$$\|q_k\| \leq (n+1)\frac{(b-a)^n}{\nu^n} =: C,$$

es decir, las normas de los polinomios q_k están acotadas, con lo que podemos aplicar (8.14) para deducir

$$\lim_{k\to\infty} \|f - p_k\|_\infty = \epsilon. \tag{8.15}$$

En particular, la sucesión de polinomios (p_k) también está acotada en el espacio normado de dimensión finita $(\Pi_n, \|\cdot\|_\infty)$. Por tanto podremos encontrar una subsucesión (k_l), puntos $x_0 < x_1 < \cdots < x_n < x_{n+1}$ y un polinomio $p \in \Pi_n$ tales que

$$\lim_{l\to\infty} p_{k_l} = p$$

y

$$\lim_{l\to\infty} x_i^{k_l} = x_i, \quad i = 0, 1, \ldots, n+1.$$

Dado que

$$\epsilon_k = \|f(x_i^k) - p_k(x_i^k)\| \qquad \text{y} \qquad f(x_{i+1}^k) - p_k(x_{i+1}^k) = -(f(x_i^k) - p_k(x_i^k))$$

para cada k, tomando límites se obtiene

$$\epsilon = \|f(x_i) - p(x_i)\| \qquad \text{y} \qquad f(x_{i+1}) - p(x_{i+1}) = -(f(x_i) - p(x_i)).$$

Por (8.15), $f - p$ equioscila en los puntos $\{x_i\}$, con lo que p es el polinomio de mejor aproximación en la norma uniforme. Más aún, obsérvese que cualquier otro punto de acumulación de la sucesión (p_k) equioscilaría en $n+2$ puntos, con lo que también sería el polinomio de mejor aproximación en la norma uniforme, es decir, p. Es bien sabido que si una sucesión acotada tiene un único punto de acumulación, entonces ése es su límite. Hemos demostrado que (p_k) converge uniformemente a p. □

Ponemos fin a la sección obteniendo como corolario el teorema de Marcinkiewicz anunciado al comienzo de la misma:

Corolario 8.4.10 (Teorema de Marcinkiewicz) *Sea $f : [a, b] \to \mathbb{R}$ continua. Entonces existe un sistema de puntos de interpolación $x_0^{(n)} < x_1^{(n)} < \cdots < x_n^{(n)}$ de manera que si $p_n \in \Pi_n$ es el polinomio interpolador en los puntos $\{(x_i^{(n)}, f(x_i^{(n)})\}_{i=0}^n$, entonces la sucesión (p_n) converge uniformemente a f.*

Demostración: Basta tomar como p_n el polinomio que mejor aproxima a f en la norma uniforme. En efecto, como $f - p$ equioscila en $n+2$ puntos tendrá $n+1$ ceros distintos, que serán los puntos $\{x_i^{(n)}\}$ buscados. Nótese que la convergencia de (p_n) a f está garantizada por el teorema de aproximación de Weierstrass. □

Bibliografía

[1] R. Burden y J. D. Faires, *Análisis Numérico. 7ª ed.*, Thomson Learning, Madrid, 2002.

[2] M. Crouzeix y A. L. Mignot, *Analyse Numérique des Équations Différentielles. 2e édition révisee et augmentée*, Masson, París, 1992.

[3] D. Kincaid y W. Cheney, *Análisis Numérico. Las Matemáticas del Cálculo Científico*, Addison-Wesley Sudamericana, Wilmington, 1994.