MÉTODOS NUMÉRICOS

EDUARDO CASAS RENTERIA

Universidad de Cantabria

2023

Capítulo 1

Resolución Aproximada de Sistemas de Ecuaciones No Lineales

1.1 Introducción

El objetivo de este capítulo es la descripción de los métodos numéricos más utilizados para resolver ecuaciones no lineales. Estas ecuaciones se escriben en la forma $f(x) = 0$, donde $f : \text{Dom}(f) \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$ es una función y $\text{Dom}(f)$ denota el dominio de definición de la misma. Usaremos métodos iterativos para resolver estas ecuaciones, comenzando las iteraciones en un punto $x^0 \in \text{Dom}(f)$, a partir del cual se genera una sucesión $x^0, x^1, x^2, \dots, x^k, \dots$. El algoritmo es el proceso por el que se pasa de un punto x^k de la sucesión al siguiente punto x^{k+1} . Lo que se espera del algoritmo es que la sucesión generada $\{x^k\}_{k=0}^{\infty}$ sea convergente y su límite $\bar{x}$ sea solución de la ecuación, es decir, $f(\bar{x}) = 0$.

Además de la convergencia del algoritmo, es muy importante la velocidad de convergencia. Después daremos una definición precisa de lo que se entiende por velocidad de convergencia. Pero, de momento vamos a ver un ejemplo en el que usaremos dos algoritmos distintos para resolver una ecuación y en el que uno de ellos se muestra claramente más rápido que el otro. Vamos a resolver la ecuación $x - \cos x = 0$. Para ello implementamos primeramente el siguiente algoritmo:

$$\begin{cases} x^0 = 0.5, \\ x^{k+1} = \cos(x^k), \quad k = 0, 1, 2, \dots \end{cases}$$

En la tabla 1 se observa la convergencia del método, pero no es hasta la iteración 89 que se estabiliza la sucesión. Es importante tener en cuenta que no se ha encontrado la solución exacta, que es un número irracional y, por lo tanto, tiene infinitos decimales. Simplemente hemos encontrado los primeros 15 decimales. En un método iterativo convergente, la solución de la ecuación es el límite de

la sucesión, pero el método debe detenerse en algún momento, no se pueden realizar infinitas iteraciones. Nos detendremos cuando se haya alcanzado la precisión deseada. Teniendo en cuenta que la solución exacta puede poseer infinitos decimales, solamente podemos aspirar a obtener una aproximación de la solución. Sin embargo, en la mayoría de los casos, unos pocos decimales de precisión es suficiente.

A continuación implementamos un segundo algoritmo para resolver la ecuación anterior:

$$\begin{cases} x^0 = 0.5, \\ x^{k+1} = x^k - \frac{x^k - \cos(x^k)}{1 + \sin(x^k)}, \quad k = 0, 1, 2, \dots \end{cases}$$

En la tabla 2 tenemos los resultados correspondientes a este algoritmo.

$1 \leq k \leq 23$	$24 \leq k \leq 46$	$47 \leq k \leq 69$	$70 \leq k \leq 92$
0.877582561890373	0.739069001204012	0.739085135039528	0.739085133214954
0.639012494165259	0.739095999835755	0.739085131986245	0.739085133215300
0.802685100682335	0.739077813285175	0.739085134042973	0.739085133215067
0.694778026788006	0.739090063988251	0.739085132657536	0.739085133215224
0.768195831282016	0.739081811778109	0.739085133590783	0.739085133215118
0.719165445942419	0.739087370571036	0.739085132962137	0.739085133215189
0.752355759421527	0.739083626103480	0.739085133385601	0.739085133215141
0.730081063137823	0.739086148422879	0.739085133100350	0.739085133215174
0.745120341351440	0.739084449358649	0.739085133292498	0.739085133215152
0.735006309014843	0.739085593868961	0.739085133163065	0.739085133215167
0.741826522643246	0.739084822913141	0.739085133250253	0.739085133215157
0.737235725442231	0.739085342238298	0.739085133191522	0.739085133215163
0.740329651878263	0.739084992414645	0.739085133231084	0.739085133215159
0.738246238332233	0.739085228060075	0.739085133204435	0.739085133215162
0.739649962769661	0.739085069326482	0.739085133222386	0.739085133215160
0.738704539356983	0.739085176251341	0.739085133210294	0.739085133215161
0.739341452281210	0.739085104225471	0.739085133218439	0.739085133215160
0.738912449332103	0.739085152742964	0.739085133212952	0.739085133215161
0.739201444135799	0.739085120060997	0.739085133216648	0.739085133215160
0.739006779780813	0.739085142075963	0.739085133214159	0.739085133215161
0.739137910762293	0.739085127246417	0.739085133215836	0.739085133215161
0.739049580595209	0.739085137235778	0.739085133214706	0.739085133215161
0.739109081420527	0.739085130506825	0.739085133215467	0.739085133215161

Algoritmo 1

0.755222417105636
0.739141666149879
0.739085133920807
0.739085133215161
0.739085133215161

Algoritmo 2

Vemos cómo en la cuarta iteración ya tenemos la aproximación que con el algoritmo anterior nos costó 89 iteraciones. Por lo tanto, la convergencia es muy importante, pero la velocidad de convergencia no lo es menos. Estas dos cuestiones serán estudiadas en la próxima sección. En la sección 1.3 estudiaremos el método de Newton para resolver estas ecuaciones. En la sección 1.4 veremos los métodos Cuasi-Newton.

En este capítulo todas las normas matriciales que se consideran son compatibles con la norma vectorial y satisfacen que $\|I\| = 1$.

1.2 Sobre la Convergencia de un Algoritmo

En esta sección vamos a considerar una sucesión $\{x^k\}_{k=0}^{\infty} \subset \mathbb{R}^n$ generada por un algoritmo a partir de un punto inicial x^0 . Queremos que esta sucesión converja hacia una solución $\bar{x}$ de la ecuación $f(x) = 0$. Se distinguen dos tipos de convergencia que pasamos a definir.

DEFINICIÓN 1.1 *Se dice que un algoritmo es globalmente convergente si la sucesión generada es convergente hacia una solución de la ecuación cualquiera que sea el punto inicial $x^0 \in \text{Dom}(f)$ que tomemos. El algoritmo se dice que es localmente convergente si existe $\varepsilon > 0$ tal que la sucesión converge hacia la solución $\bar{x}$ siempre que $x^0 \in B_\varepsilon(\bar{x})$.*

Parece razonable buscar algoritmos que sean globalmente convergentes, pero no lo es. En la resolución de la ecuación $x - \cos x = 0$ utilizamos un primer algoritmo que es globalmente convergente, sin embargo su convergencia es muy lenta. El segundo algoritmo que utilizamos es localmente convergente, pero su velocidad de convergencia es muy superior. Seguidamente vamos a definir la velocidad de convergencia de una sucesión.

DEFINICIÓN 1.2 *Supongamos que la sucesión $\{x^k\}_{k=0}^{\infty}$ converge hacia $\bar{x}$. Se dice que la velocidad (o el orden) de convergencia es p con $1 \leq p < \infty$ si existe una constante C_p y un entero k_0 tal que*

$$\|x^{k+1} - \bar{x}\| \leq C_p \|x^k - \bar{x}\|^p \quad \forall k \geq k_0. \quad (1.1)$$

- Si $p = 1$ y $C_p \geq 1$, se dice que la convergencia es sublineal.
- Si $p = 1$ y $C_p < 1$, se dice que la convergencia es lineal.

- Si $p = 2$, la convergencia se dice cuadrática y cúbica si $p = 3$.

Cuando la convergencia es lineal, adquirir un decimal de precisión en la solución requiere típicamente varias iteraciones, su número dependiendo de si C_1 es próximo a 0 o a 1. Sin embargo, cuando la convergencia es cuadrática el número de decimales de precisión típicamente se dobla de una iteración a otra. Esto se puede observar en las tablas de resultados de los dos algoritmos usados en la introducción. Si la convergencia es cúbica, entonces se triplica el número de decimales de precisión de una iteración a otra.

Entre la convergencia lineal y la de orden $p > 1$ con p arbitrariamente próximo a 1 hay un tipo de convergencia intermedio que tiene su interés, y que definimos a continuación.

DEFINICIÓN 1.3 Supongamos que la sucesión $\{x^k\}_{k=0}^{\infty}$ converge hacia $\bar{x}$. Se dice que la velocidad de convergencia es superlineal si existe una sucesión de números reales $\{\varepsilon_k\}_{k=0}^{\infty}$ satisfaciendo las siguientes condiciones

$$\lim_{k \rightarrow \infty} \varepsilon_k = 0 \quad y \quad \|x^{k+1} - \bar{x}\| \leq \varepsilon_k \|x^k - \bar{x}\| \quad \forall k \geq 0. \quad (1.2)$$

Obviamente, la convergencia superlineal es más rápida que la simplemente lineal. Sin embargo, si $p > 1$ y $\{x^k\}_{k=0}^{\infty}$ tiene velocidad de convergencia p , entonces $\{x^k\}_{k=0}^{\infty}$ converge superlinealmente. En efecto, es suficiente tomar $\varepsilon_k = C_p \|x^k - \bar{x}\|^{p-1}$ para cada $k \geq k_0$ y, por ejemplo, $\varepsilon_k = \frac{\|x^{k+1} - \bar{x}\|}{\|x^k - \bar{x}\|}$ si $0 \leq k < k_0$. Entonces usando (1.1) y la definición de ε_k se tiene:

$$\begin{aligned} \lim_{k \rightarrow \infty} x^k = \bar{x} \quad y \quad p > 1 &\Rightarrow \lim_{k \rightarrow \infty} \varepsilon_k = 0, \\ \|x^{k+1} - \bar{x}\| &\leq C_p \|x^k - \bar{x}\|^p = \varepsilon_k \|x^k - \bar{x}\| \quad \forall k \geq k_0. \end{aligned}$$

Veamos la siguiente propiedad importante de las sucesiones que convergen superlinealmente.

TEOREMA 1.4 Supongamos que $\{x^k\}_{k=0}^{\infty}$ converge superlinealmente hacia $\bar{x}$, entonces se cumple

$$\lim_{k \rightarrow \infty} \frac{\|x^{k+1} - x^k\|}{\|x^k - \bar{x}\|} = 1. \quad (1.3)$$

Demostración. Sabemos que para cada par de elementos $u, v \in \mathbb{R}^n$ se tiene que $\| \|u\| - \|v\| \| \leq \|u + v\|$. Aplicando esta desigualdad con $u = x^{k+1} - x^k$ y $v = x^k - \bar{x}$ y utilizando (1.2), se obtiene

$$\begin{aligned} \left| \frac{\|x^{k+1} - x^k\|}{\|x^k - \bar{x}\|} - 1 \right| &= \frac{\left| \|x^{k+1} - x^k\| - \|x^k - \bar{x}\| \right|}{\|x^k - \bar{x}\|} \\ &\leq \frac{\|x^{k+1} - \bar{x}\|}{\|x^k - \bar{x}\|} \leq \varepsilon_k \xrightarrow{k \rightarrow \infty} 0. \end{aligned}$$

□

Analicemos la importancia de la propiedad (1.3). Cualquier algoritmo destinado a calcular una aproximación de la solución de una ecuación debe tener un test de parada que finalice las iteraciones. Este test de parada debe recoger la precisión que se desee alcanzar. La forma natural y habitual para finalizar el algoritmo es comparar dos iteraciones consecutivas y examinar el grado de coincidencia de sus dígitos. Más concretamente, dado un número $\varepsilon > 0$ que indica la precisión deseada se realizará el test $\|x^{k+1} - x^k\| < \varepsilon$ para decidir si la precisión es satisfactoria. Esto tiene sentido cuando la sucesión converge superlinealmente porque, de acuerdo a (1.3), $\|x^k - \bar{x}\| \approx \|x^{k+1} - x^k\|$. Además $\|x^{k+1} - \bar{x}\| \leq \varepsilon_k \|x^k - \bar{x}\| \approx \varepsilon_k \|x^{k+1} - x^k\| < \varepsilon_k \varepsilon$ con $\varepsilon_k \rightarrow 0$ cuando $k \rightarrow \infty$. Sin embargo, cuando la convergencia es lineal, entonces puede ocurrir que $\|x^{k+1} - x^k\|$ sea pequeño y sin embargo x^{k+1} esté muy lejos de la solución.

1.3 Método de Newton

El método Newton para la resolución del sistema no lineal $f(x) = 0$ tiene la siguiente formulación básica:

$$\begin{cases} x^0 \in \mathbb{R}^n \text{ y para cada } k \geq 0 \\ Df(x^k)d^k = -f(x^k) \\ x^{k+1} = x^k + d^k \end{cases} \quad (1.1)$$

TEOREMA 1.5 Sea $\bar{x} \in \text{Dom}(f) \subset \mathbb{R}^n$ y $f : \text{Dom}(f) \rightarrow \mathbb{R}^n$ una función de clase C^1 en la bola $B_r(\bar{x}) \subset \text{Dom}(f)$. Además, suponemos que $f(\bar{x}) = 0$, $Df(\bar{x})$ es una matriz inversible y existe una constante $L > 0$ tal que

$$\|Df(x^2) - Df(x^1)\| \leq L\|x^2 - x^1\| \quad \forall x^2, x^1 \in B_r(\bar{x}). \quad (1.2)$$

Entonces, existe $0 < \varepsilon \leq r$ tal que si $x^0 \in B_\varepsilon(\bar{x})$ la sucesión generada por el método de Newton (1.1) está bien definida, $\{x^k\}_{k=0}^\infty \subset B_\varepsilon(\bar{x})$, converge hacia $\bar{x}$ y satisface

$$\|x^{k+1} - \bar{x}\| \leq L\|Df(\bar{x})^{-1}\|\|x^k - \bar{x}\|^2 \quad \forall k \geq 0. \quad (1.3)$$

LEMA 1.6 Sea A una matriz $n \times n$. Entonces se cumplen las siguientes propiedades

1. Si $\|A\| < 1$, entonces $I - A$ es inversible y $\|(I - A)^{-1}\| \leq \frac{1}{1 - \|A\|}$.
2. Si $\|I - A\| < 1$, entonces A es inversible y $\|A^{-1}\| \leq \frac{1}{1 - \|I - A\|}$.

Demostración. (1) Dado que $\|A\| < 1$, la serie $\sum_{k=0}^{\infty} A^k$ es convergente.

Además es inmediato comprobar que $(I - A) \sum_{k=0}^{\infty} A^k = I$. Por lo tanto $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$ y

$$\|(I - A)^{-1}\| \leq \sum_{k=0}^{\infty} \|A\|^k = \frac{1}{1 - \|A\|}. \quad (1.4)$$

(2) Si $\|I - A\| < 1$, entonces de (1) se deduce que $A = I - (I - A)$ es inversible y $\|A^{-1}\| \leq \frac{1}{1 - \|I - A\|}$. $\square$

LEMA 1.7 Sea $\Omega \subset \mathbb{R}^n$ un conjunto abierto y convexo, y $f : \Omega \rightarrow \mathbb{R}^n$ una función de clase C^1 . Entonces, se cumple

$$f(x^2) - f(x^1) = \int_0^1 Df(x^1 + t(x^2 - x^1))(x^2 - x^1) dt \quad \forall x^1, x^2 \in \Omega. \quad (1.5)$$

Demostración. Dados $x^1, x^2 \in \Omega$, denotemos por $g : [0, 1] \rightarrow \mathbb{R}^n$ la función $g(t) = f(x^1 + t(x^2 - x^1))$. Entonces g es de clase C^1 y se tiene

$$\begin{aligned} f(x^2) - f(x^1) &= g(1) - g(0) = \int_0^1 g'(t) dt \\ &= \int_0^1 Df(x^1 + t(x^2 - x^1))(x^2 - x^1) dt. \end{aligned}$$

$\square$

Demostración del Teorema 1.5. Sea

$$\varepsilon = \min \left\{ r, \frac{1}{2L\|Df(\bar{x})^{-1}\|} \right\}. \quad (1.6)$$

Probemos que $Df(x^k)$ es inversible si $x^k \in B_\varepsilon(\bar{x})$. De (1.2) y (1.6) se obtiene

$$\begin{aligned} \|I - Df(\bar{x})^{-1}Df(x^k)\| &\leq \|Df(\bar{x})^{-1}\| \|Df(\bar{x}) - Df(x^k)\| \\ &\leq \|Df(\bar{x})^{-1}\| L \|\bar{x} - x^k\| \leq \frac{1}{2}. \end{aligned}$$

Del Lema 1.6-(2) se sigue que $Df(\bar{x})^{-1}Df(x^k)$ es inversible y, por lo tanto, $Df(x^k)$ también lo es. Además, del mismo lema y la desigualdad anterior se concluye

$$\begin{aligned} \|Df(x^k)^{-1}\| &= \|[Df(\bar{x})^{-1}Df(x^k)]^{-1}Df(\bar{x})^{-1}\| \\ &\leq \|[Df(\bar{x})^{-1}Df(x^k)]^{-1}\| \|Df(\bar{x})^{-1}\| \\ &\leq \frac{\|Df(\bar{x})^{-1}\|}{1 - \|I - Df(\bar{x})^{-1}Df(x^k)\|} \leq 2\|Df(\bar{x})^{-1}\|. \end{aligned} \quad (1.7)$$

A continuación demostramos que

$$\|x^{k+1} - \bar{x}\| \leq \frac{1}{2}\|x^k - \bar{x}\|, \quad (1.8)$$

donde x^{k+1} es el punto proporcionado por el algoritmo a partir de x^k . Entonces, con la ayuda del Lema 1.7 obtenemos

$$\begin{aligned} x^{k+1} - \bar{x} &= x^k - \bar{x} - Df(x^k)^{-1}f(x^k) \\ &= x^k - \bar{x} + Df(x^k)^{-1}[f(\bar{x}) - f(x^k)] \\ &= Df(x^k)^{-1}[f(\bar{x}) - f(x^k) - Df(x^k)(\bar{x} - x^k)] \\ &= Df(x^k)^{-1} \left[\int_0^1 Df(x^k + t(\bar{x} - x^k))(\bar{x} - x^k) dt - Df(x^k)(\bar{x} - x^k) \right] \\ &= Df(x^k)^{-1} \int_0^1 [Df(x^k + t(\bar{x} - x^k)) - Df(x^k)](\bar{x} - x^k) dt. \end{aligned}$$

De esta identidad, la hipótesis (1.2), (1.6) y (1.7) se sigue

$$\begin{aligned} \|x^{k+1} - \bar{x}\| &\leq L\|Df(x^k)^{-1}\| \int_0^1 t dt \|\bar{x} - x^k\|^2 \\ &= \frac{L}{2}\|Df(x^k)^{-1}\| \|\bar{x} - x^k\|^2 \\ &\leq L\|Df(\bar{x})^{-1}\| \|\bar{x} - x^k\|^2 \leq \frac{1}{2}\|\bar{x} - x^k\|. \end{aligned} \quad (1.9)$$

Esto demuestra (1.8). Ahora (1.8) implica que la sucesión $\{x^k\}_{k=0}^\infty$ está bien definida y contenida en la bola $B_r(\bar{x})$. Finalmente, (1.3) es consecuencia de (1.9).

TEOREMA 1.8 (Kantorovich) *Sea $x^0 \in \mathbb{R}^n$ y $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ una función de clase C^1 en la bola $B_r(x^0)$. Se supone que $Df(x^0)$ es una matriz inversible y se cumplen las siguientes hipótesis*

$$\exists L > 0 : \|Df(x^2) - Df(x^1)\| \leq L\|x^2 - x^1\| \quad \forall x^2, x^1 \in B_r(x^0), \quad (1.10)$$

$$\alpha = L\|Df(x^0)^{-1}\| \|Df(x^0)^{-1}f(x^0)\| \leq \frac{1}{2}, \quad (1.11)$$

$$r_0 = \frac{1 - \sqrt{1 - 2\alpha}}{L\|Df(x^0)^{-1}\|} \leq r. \quad (1.12)$$

Entonces, la sucesión $\{x^k\}_{k=0}^\infty$ generada por el método de Newton (1.1) está bien definida y converge hacia un punto $\bar{x}$ que es la única solución de la ecuación $f(x) = 0$ en la bola $B_{r_0}(x^0)$. Si $\alpha < \frac{1}{2}$, entonces $\bar{x}$ es el único cero de f en $B_{r_1}(x^0)$, donde

$$r_1 = \min \left\{ r, \frac{1 + \sqrt{1 - 2\alpha}}{L\|Df(x^0)^{-1}\|} \right\},$$

y además

$$\|x^k - \bar{x}\| \leq \frac{1}{L\|Df(x^0)^{-1}\|}(2\alpha)^{2^k}. \quad (1.13)$$

Dado el carácter local de la convergencia del método de Newton, se incorporan modificaciones al método para promover la convergencia cuando el punto x^0 elegido está lejos de la solución. En el caso de ecuaciones escalares la estrategia pasa por combinar el método de Newton con el de bisección. Para ello se debe localizar un intervalo $[a, b]$ tal que $f(a)f(b) < 0$ y entonces proceder como se señala en el siguiente algoritmo.

Algoritmo de Newton-Bisección

1. Tomar $x^0 \in [a, b]$; $k = 0$; $\varepsilon > 0$;
 - Si $f(a)f(x^0) < 0 \Rightarrow a^0 = a, b^0 = x^0$.
 - Si no $\Rightarrow a^0 = x^0, b^0 = b$.
2.
 - Si $|f'(x^k)| > \varepsilon_M |f(x^k)| \Rightarrow y^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$.
 - Si $y^{k+1} \in [a^k, b^k] \Rightarrow x^{k+1} = y^{k+1}$.
 - Si no $\Rightarrow x^{k+1} = \frac{a^k + b^k}{2}$.
 - Si no $\Rightarrow x^{k+1} = \frac{a^k + b^k}{2}$.
3.
 - Si $f(a^k)f(x^{k+1}) < 0 \Rightarrow a^{k+1} = a^k, b^{k+1} = x^{k+1}$.
 - Si no $\Rightarrow a^{k+1} = x^{k+1}, b^{k+1} = b^k$.
4.
 - Si $|x^{k+1} - x^k| < \varepsilon \max\{1, |x^{k+1}|\}$ y $|f(x^{k+1})| < \varepsilon \Rightarrow \text{STOP}$.
 - Si $|f(x^{k+1})| < \varepsilon_f \Rightarrow \text{STOP}$.
 - Si no $\Rightarrow k \rightarrow k + 1$ y volver al paso 2.

En el algoritmo anterior se realiza un control de la derivada $f'(x^k)$ para evitar divisiones por cero o incluso un overflow.

Obviamente en el caso de sistemas de ecuaciones el procedimiento anterior no se puede adaptar. Una estrategia frecuentemente empleada pasa por tomar $x^{k+1} = x^k + \rho_k d^k$, donde $0 < \rho_k \leq 1$. El paso ρ_k es elegido de forma que $\|f(x^{k+1})\| < \|f(x^k)\|$, donde la norma elegida es la euclídea. Primero se prueba

el paso $\rho_k = 1$ y si este no satisface la condición anterior entonces se reduce hasta que se satisfaga. Observemos que siempre es posible encontrar un paso ρ_k de forma que $\|f(x^{k+1})\| < \|f(x^k)\|$. En efecto, consideremos la función $h_k : [0, 1] \rightarrow \mathbb{R}$ definida por $h_k(\rho) = \|f(x^k + \rho d^k)\|^2$. Entonces, usando la definición de d^k tenemos

$$h'_k(0) = 2f(x^k)^T Df(x^k)d^k = -2f(x^k)^T f(x^k) = -2\|f(x^k)\|^2 < 0.$$

Esto implica la existencia de $\hat{\rho} \in (0, 1]$ tal que $h_k(\rho) < h_k(0)$ para todo $\rho \in (0, \hat{\rho}]$, lo que es equivalente a $\|f(x^k + \rho d^k)\| < \|f(x^k)\|$.

Así como en el algoritmo anterior se modifica el método si $f'(x^k)$ se anula o se hace muy pequeño, en varias variables debemos controlar si la matriz jacobina es inversible o está muy mal condicionada. El algoritmo siguiente controla este asunto.

Algoritmo de Newton

1. Tomar un punto inicial $x^0 \in \mathbb{R}^n$; $\beta \in (0, 1/2]$; $\varepsilon > 0$; $\varepsilon_f > 0$; $k = 0$. Ir al paso 2.
2. $A_k = Df(x^k)$.
 - Mientras $\text{cond}(A_k) > \frac{100}{\sqrt{\varepsilon_M}} \Rightarrow A_k = A_k + \lambda_k I$, con $\lambda_k = \sqrt{\varepsilon_M} \|Df(x^k)\|$.
 Ir al paso 3.
3. $A_k d^k = -f(x^k)$. Ir al paso 4.
4. Tomar

$$m_k = \min\{j \in \mathbb{Z}^+ : \|f(x^k + \beta^j d^k)\| < \|f(x^k)\|\},$$

$$\rho_k = \beta^{m_k} \text{ y } x^{k+1} = x^k + \rho_k d^k.$$
- Ir al paso 5.
5.
 - Si $\|x^{k+1} - x^k\| < \varepsilon \max\{1, \|x^{k+1}\|\}$ y $\|f(x^{k+1})\| < \varepsilon \Rightarrow \text{STOP}$.
 - Si $\|f(x^{k+1})\| < \varepsilon_f \Rightarrow \text{STOP}$.
 - Si no $\Rightarrow$ Incrementar el contador de iteraciones $k \rightarrow k + 1$. Volver a 2.

NOTAS:

- ε es la precisión que se desea en la solución.
- ε_f es la precisión con la que se evalúa la función f . Si no se conoce, se debe elegir un número mayor que la precisión de la máquina. En algunos casos $\varepsilon_f = \varepsilon_M^{3/4}$ puede ser un número razonable.

- En el paso 2 es aconsejable utilizar la función *rcond* de Matlab para obtener una estimación numérica del inverso del número de condición (según la norma 1) de la matriz A_k . De esta forma, la condición del paso 2 podría escribirse en la forma $rcond(A_k) < \frac{\sqrt{\varepsilon_M}}{100}$. Análogamente es aconsejable utilizar *norm*($A_k, 1$) para obtener la norma 1 de A_k como alternativa a la norma 2, que requiere un mayor coste computacional.

1.4 Métodos Cuasi-Newton

Los métodos Cuasi-Newton son aquellos en los que se reemplaza la matriz jacobiana por una aproximación. En el caso de ecuaciones escalares es bien conocido que el método de la secante es una alternativa eficiente al método de Newton cuando la derivada $f'(x)$ no está disponible o su cálculo es computacionalmente costosa. El método de la secante es también localmente convergente, pero no converge cuadráticamente como lo hace el método de Newton. No obstante, tiene una velocidad de convergencia satisfactoria: $p = \frac{1+\sqrt{5}}{2} \approx 1.6180\dots$. Todas estas propiedades se cumplen bajo las mismas hipótesis que en el método de Newton. A continuación describimos el algoritmo de la secante combinado con bisección.

Algoritmo de la Secante-Bisección

1. Tomar $x^0 = a$; $x^1 = b$; $a^1 = a$; $b^1 = b$; $\varepsilon > 0$;
2. Calcular

$$m_0 = \frac{f(x^1) - f(x^0)}{x^1 - x^0} \quad \text{y} \quad x^2 = x^1 - \frac{f(x^1)}{m_0}.$$
3. $k = 2$.
 - Si $f(x^2)f(a^1) < 0 \Rightarrow a^2 = a^1$, $b^2 = x^2$, $x^1 = x^0$.
 - Si no $\Rightarrow a^2 = x^2$, $b^2 = b^1$.
4. $m_k = \frac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}}$.
 - Si $|m_k| > \varepsilon_M |f(x^k)| \Rightarrow y_{k+1} = x^k - \frac{f(x^k)}{m_k}$.
 - Si $y_{k+1} \in [a^k, b^k] \Rightarrow x^{k+1} = y_{k+1}$.
 - Si no $\Rightarrow x^{k+1} = \frac{a^k + b^k}{2}$.
 - Si no $\Rightarrow x^{k+1} = \frac{a^k + b^k}{2}$.
5.
 - Si $f(a^k)f(x^{k+1}) < 0 \Rightarrow a^{k+1} = a^k$, $b^{k+1} = x^{k+1}$.

- Si no $\Rightarrow a^{k+1} = x^{k+1}$, $b^{k+1} = b^k$.

6.

- Si $|x^{k+1} - x^k| < \varepsilon \max\{1, |x^{k+1}|\}$ y $|f(x^{k+1})| < \varepsilon \Rightarrow \text{STOP}$.
- Si $|f(x^{k+1})| < \varepsilon_f \Rightarrow \text{STOP}$.
- Si no $\Rightarrow k \rightarrow k + 1$ y volver al paso 4.

Es evidente que el método de la secante no tiene una traducción al caso de sistemas de ecuaciones. En efecto, en este caso necesitamos obtener una matriz B_k que sea aproximación de $Df(x^k)$. La fórmula $m_k = \frac{f(x^k) - f(x^{k-1})}{x^k - x^{k-1}}$ no es trasladable al caso de varias dimensiones. Sin embargo, si escribimos la identidad anterior en la forma $m_k(x^k - x^{k-1}) = f(x^k) - f(x^{k-1})$, entonces sí podemos escribir en varias dimensiones una identidad análoga: $B_k(x^k - x^{k-1}) = f(x^k) - f(x^{k-1})$. Esta ecuación se denomina ecuación cuasi-Newton.

En 1965, Broyden propuso un método para obtener las matrices B_k que reemplazan a $Df(x^k)$. La idea es comenzar con una aproximación B_0 de $Df(x^0)$, lo que se puede lograr mediante alguna fórmula de diferenciación numérica. A partir de ahí calculamos x^1 . Seguidamente determinamos una matriz B_1 que satisfaga la ecuación cuasi-Newton y sea lo más parecida a B_0 , que contiene una buena información de la matriz jacobiana. Este procedimiento se extiende a todas las iteraciones. Así, una vez calculado x^{k+1} procedemos a calcular B_{k+1} de forma que esta matriz se parezca lo más posible a B_k y también cumpla la ecuación cuasi-Newton $B_{k+1}s^k = y^k$, donde $s^k = x^{k+1} - x^k$ y $y^k = f(x^{k+1}) - f(x^k)$. A tal efecto Broyden propuso la siguiente fórmula

$$B_{k+1} = B_k + \frac{(y^k - B_k s^k)s^k{}^T}{s^k{}^T s^k}.$$

Es fácil comprobar que $B_{k+1}s^k = y^k$ y $B_{k+1}u = B_k u$ para cada $u \in \mathbb{R}^n$ ortogonal a s^k . Así B_{k+1} es una adaptada de rango uno de B_k .

Al igual que en el método de Newton es necesario controlar la inversibilidad y mal condicionamiento de la matriz B_k . También es necesario introducir el paso ρ_k , pero en este caso observemos que

$$h'_k(0) = 2f(x^k)^\top Df(x^k)d^k = -2f(x^k)^\top [Df(x^k)B_k^{-1}]f(x^k).$$

Ahora no podemos garantizar que $h'_k(0) < 0$ como en el método de Newton. Si tras varias iteraciones B_k no es una aproximación de $Df(x^k)$ o, al menos, si d^k no es una aproximación de la dirección de Newton, entonces el algoritmo puede fallar al buscar ρ_k . En este caso, si después de un determinado número de iteraciones, no se ha encontrado un ρ_k adecuado, se reemplaza la matriz B_k por la aproximación numérica de $Df(x^k)$. Estas estrategias están incorporadas en el algoritmo siguiente.

Algoritmo de Broyden

1. Tomar un punto inicial $x^0 \in \mathbb{R}^n$; $\beta \in (0, 1/2]$; $\varepsilon > 0$; $\varepsilon_f > 0$; $B_0 \simeq Df(x^0)$; $k = 0$; $IND = 1$; $max > 0$.
2. Mientras $\text{cond}(B_k) > \frac{100}{\sqrt{\varepsilon_M}} \Rightarrow B_k = B_k + \lambda_k I$, con $\lambda_k = \sqrt{\varepsilon_M} \|B_k\|$.
3. $B_k d^k = -f(x^k)$.
4. $m_k = \min\{j \in \mathbb{Z}^+ : \|f(x^k + \beta^j d^k)\| < \|f(x^k)\|\}$.
 - Si $m_k > max$ y $IND = 1 \Rightarrow \text{STOP}$. No hay convergencia.
 - Si $m_k > max$ y $IND = 0 \Rightarrow B_k \simeq Df(x^k)$; $IND = 1$. Volver a 2.
 - Si no $\Rightarrow IND = 0$, $\rho_k = \beta^{m_k}$; $x^{k+1} = x^k + \rho_k d^k$. Ir a 5.
5.
 - Si $\|x^{k+1} - x^k\| < \varepsilon \max\{1, \|x^{k+1}\|\}$ y $\|f(x^{k+1})\| < \varepsilon \Rightarrow \text{STOP}$.
 - Si $\|f(x^{k+1})\| < \varepsilon_f \Rightarrow \text{STOP}$.
 - Si no $\Rightarrow s^k = x^{k+1} - x^k$, $y^k = f(x^{k+1}) - f(x^k)$ y hacer

$$B_{k+1} = B_k + \frac{(y^k - B_k s^k) s^{kT}}{s^{kT} s^k}$$

Incrementar el contador de iteraciones $k \rightarrow k + 1$. Volver a 2.

NOTAS:

- Las notas relativas al método de Newton son trasladables al método de Broyden.
- Las aproximaciones $B_k \simeq Df(x^k)$ se consiguen mediante una fórmula de derivación numérica.
- La elección del número max depende de la elección de β . β^{max} no debe ser excesivamente pequeño.

Ejercicios del Capítulo 1

1. Resolver los siguientes sistemas de ecuaciones no lineales con la máxima precisión que te permita tu ordenador.

$$\left\{ \begin{array}{l} 3x_1 - \cos(x_2x_3) = \frac{1}{2} \\ x_1^2 - 81(x_2 + 0.1)^2 + \operatorname{sen} x_3 = -1.06 \\ e^{-x_1x_2} + 20x_3 = 1 - \frac{10\pi}{3} \end{array} \right. \quad \left\{ \begin{array}{l} \cos(x_1x_2x_3) + x_1 = 1 \\ \sqrt[4]{1-x_1} + 0.05x_3^2 - 0.15x_3 = 1 \\ x_1^2 + 0.1x_2^2 - 0.01x_2 - x_3 = -1 \end{array} \right.$$

$$\left\{ \begin{array}{l} 6x_1 - 2\cos(x_2x_3) - 1 = 0 \\ 9x_2 + \sqrt{x_1^2} + \operatorname{sen} x_3 + 1.06 + 0.9 = 0 \\ 60x_3 + 3e^{-x_1x_2} + 10\pi = 3 \end{array} \right. \quad \left\{ \begin{array}{l} \frac{1}{2}\operatorname{sen}(x_1x_2) - \frac{x_2}{4\pi} - \frac{x_1}{2} = 0 \\ \left(1 - \frac{1}{4\pi}\right)(e^{2x_1} - e) + \frac{e}{\pi}x_2 = 2ex_1 \end{array} \right.$$

$$\left\{ \begin{array}{l} x_2x_3 - x_3^2 = 1 \\ x_1^2 - x_2^2 + x_1x_2x_3 = 3.7 \\ e^{x_1} - ex_2 = 1 \end{array} \right. \quad \left\{ \begin{array}{l} x_1x_2 + \operatorname{sen}(x_3x_4) = \frac{1}{2} \\ \cos(x_1x_2) + x_3x_4 = 3 \\ 2x_1 - x_2 + x_3 - 2x_4 = 1 \\ x_1^2 + x_2^2 + x_3^2 + x_4^2 = 25 \end{array} \right.$$

2. Dado el siguiente sistema de ecuaciones

$$\left\{ \begin{array}{l} \exp(u) + v^2 = 0 \\ u^2 + \exp(v) = 1 \end{array} \right.$$

probar que no posee soluciones reales. Se plantea entonces el cálculo de las soluciones complejas del mismo: $u = x_1 + ix_2$ y $v = x_3 + ix_4$. Transformar el sistema complejo en un sistema real de doble dimensión y obtener mediante algún método numérico una solución u y v . ¿Es la solución única?. (Nota: para comenzar las iteraciones no olvidar que no hay soluciones reales u y v al sistema planteado).

3. Dada la matriz

$$A = \begin{pmatrix} x_1 & 2 & -3 & 5 & 0 & x_1x_2 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & x_1 & x_2 & 0 & 5 & -2 \\ 1 & -1 & 1 & -1 & x_1 + x_2 & 0 \\ 1 & 0 & 1 & 0 & x_1x_2 & -2 \\ 0 & 0 & 1 & 0 & -x_1 & -x_2 \end{pmatrix}$$

determinar un valor de $x = (x_1, x_2)$, con la precisión que te permita tu ordenador, tal que

$$\begin{cases} \det(A) = 1 \\ \text{Traza}(A) = 0. \end{cases}$$

4. Hallar una solución del sistema de ecuaciones siguiente con la máxima precisión que te permita tu ordenador

$$\begin{cases} \det A(x) = 0 \\ \|x\| = 2, \end{cases}$$

donde

$$A(x) = \begin{pmatrix} x_1 + 2 & \cos x_2 & 1 \\ \text{sen } x_2 & x_1 + x_2 & 3 \\ 4 & 2 & x_1 x_2 \end{pmatrix}.$$

5. Hallar una solución del sistema de ecuaciones siguiente con la máxima precisión que te permita tu ordenador

$$\begin{cases} \det(A) = -2 \\ \|x\|^2 = 7 \\ x_1 + 2x_2 + 3x_3 + 4x_4 = 5 \\ x_1 - x_2 + 2x_3 - 2x_4 = 2 \end{cases} \quad \text{donde } A = \begin{pmatrix} x_1 & x_1 x_2 & x_3 x_4 & x_4 \\ 0 & 2 & 1 & 0 \\ 0 & 4 & 3 & 1 \\ x_1 & x_2 & x_3 & x_4 \end{pmatrix}.$$

6. Resolver los siguientes problemas de contorno con la máxima precisión que te permita tu ordenador.

$$\begin{cases} y^{(4)} - y'' + \frac{e^t}{1 + y(t)^2} = 1 + \text{sen } t, \quad t \in [0, \pi] \\ y(0) = 0, y'(0) = 0, y(\pi) = 1, y'(\pi) = 0. \end{cases}$$

$$\begin{cases} y''(t) = ty^3(t) - 1 - \text{sen } t \quad t \in [0, \frac{\pi}{2}] \\ y(0) = y(\frac{\pi}{2}) \\ y'(0) + y'(\frac{\pi}{2}) = 0. \end{cases}$$

$$\begin{cases} y^{(4)}(t) = y''(t) - ty(t)^3 + 2t \quad t \in [0, \pi/2] \\ y(0) = y'(0) = 0 \\ y(\pi/2) + y'(\pi/2) = 0, [y''(\pi/2)]^2 + [y'''(\pi/2)]^2 = 1. \end{cases}$$

Capítulo 2

Optimización. Métodos de Gradiente, Newton y Cuasi-Newton

2.1 Introducción

Dada una función $f : \mathbb{R}^n \rightarrow \mathbb{R}$ se considera el siguiente problema de optimización

$$(P) \min_{x \in \mathbb{R}^n} f(x).$$

El objetivo de este capítulo es analizar las condiciones bajo las cuales este problema posee solución y proporcionar algoritmos eficientes para su cálculo. El lector puede preguntarse por qué se estudia el problema de minimizar una función y no el de maximizar la función. La razón es que todo problema de maximización se convierte de forma trivial en un problema de minimización. En efecto, supongamos que queremos resolver el siguiente problema de maximización

$$(Q) \max_{x \in \mathbb{R}^n} g(x),$$

donde $g : \mathbb{R}^n \rightarrow \mathbb{R}$ es una función dada. Entonces, es suficiente definir $f(x) = -g(x)$ y hallar las soluciones $\bar{x}$ de (P). Entonces, $\bar{x}$ es también la solución de (Q). Por consiguiente, el problema (P) posee solución si y solamente si (Q) posee solución. Además las soluciones son las mismas, la única diferencia radica en el valor mínimo de f y el valor máximo de g : $f(\bar{x}) = -g(\bar{x})$.

El plan del capítulo es como sigue. Primeramente analizaremos la existencia y/o unicidad de solución de (P). Distinguiremos entre mínimos locales y globales. En la sección 2.3 estudiaremos las condiciones necesarias y suficientes de optimalidad de mínimo local o global. Finalmente, en la sección 4 formularemos los algoritmos que usaremos para resolver el problema (P).

2.2 Existencia y Unicidad de Solución

DEFINICIÓN 2.1 Un punto $\bar{x} \in \mathbb{R}^n$ se dice que es un *mínimo local* de f si existe $\varepsilon > 0$ tal que $f(\bar{x}) \leq f(x)$ para cada $x \in B_\varepsilon(\bar{x})$. Se dice que $\bar{x}$ es un *mínimo local estricto* si la desigualdad anterior es estricta para cada $x \in B_\varepsilon(\bar{x})$ con $x \neq \bar{x}$. Un punto $\bar{x} \in \mathbb{R}^n$ se dice que es un *mínimo global* o *solución* de (P) si $f(\bar{x}) \leq f(x)$ para cada $x \in \mathbb{R}^n$.

DEFINICIÓN 2.2 Una función $f : \mathbb{R}^n \rightarrow \mathbb{R}$ se dice *coerciva* si

$$\lim_{\|x\| \rightarrow +\infty} f(x) = +\infty.$$

TEOREMA 2.3 (Existencia) Si $f : \mathbb{R}^n \rightarrow \mathbb{R}$ es continua y coerciva, entonces (P) posee al menos una solución.

DEFINICIÓN 2.4 Una función $f : \mathbb{R}^n \rightarrow \mathbb{R}$ se dice *convexa* si

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \quad \forall x, y \in \mathbb{R}^n \quad \text{y} \quad 0 < \lambda < 1.$$

La función f se dice *estrictamente convexa* si la desigualdad anterior es estricta.

TEOREMA 2.5 (Unicidad) Si f es estrictamente convexa, entonces (P) posee a lo sumo una solución.

TEOREMA 2.6 Sea f una función convexa, entonces $\bar{x}$ es un *mínimo local* de f si y sólo si $\bar{x}$ es un *mínimo global*.

2.3 Condiciones de Optimalidad

TEOREMA 2.7 (Condiciones Necesarias) Sea $\bar{x}$ un *mínimo local* de f .

- Si f es derivable en $\bar{x}$, entonces $\nabla f(\bar{x}) = 0$ (*Condición de Primer Orden*).
- Si f es de clase C^2 en un entorno de $\bar{x}$, entonces $\nabla^2 f(\bar{x}) \geq 0$ (*Condición de Segundo Orden*).

TEOREMA 2.8 (Condiciones Suficientes) Sea f una función de clase C^2 en un entorno de un punto $\bar{x} \in \mathbb{R}^n$. Si $\nabla f(\bar{x}) = 0$ y $\nabla^2 f(\bar{x}) > 0$, entonces $\bar{x}$ es un *mínimo local estricto* de f .

TEOREMA 2.9 Sea $f : \mathbb{R}^n \rightarrow \mathbb{R}$ una función convexa y derivable en un punto $\bar{x} \in \mathbb{R}^n$. Entonces $\bar{x}$ es un *mínimo global* de f si y sólo si $\nabla f(\bar{x}) = 0$.

NOTA: Con $\nabla^2 f(\bar{x}) \geq 0$ ($\nabla^2 f(\bar{x}) > 0$) queremos indicar que la matriz $\nabla^2 f(\bar{x})$ es semidefinida positiva (definida positiva).

2.4 Algoritmos de Minimización

Método de Newton

1. Tomar un punto inicial $x^0 \in \mathbb{R}^n$; $\beta \in [1/10, 1/2]$; $\sigma \in [0.001, 0.1]$; $\varepsilon > 0$; $\varepsilon_f > 0$; $k = 0$. Ir al paso 2.
2. Tomar $A_k = \nabla^2 f(x^k)$. Si A_k no es definida positiva o $\text{cond}(A_k) > \varepsilon_M^{-1/2}$, entonces hacer

- Si $\Lambda_k = \mu_k$, entonces $A_k = I_n$,
- En otro caso, $A_k = A_k + \lambda_k I_n$, con $\lambda_k = \frac{\sqrt{\varepsilon_M} \Lambda_k - \mu_k}{1 - \sqrt{\varepsilon_M}}$,

donde Λ_k y μ_k denotan los valores propios más grande y más pequeño de A_k y I_n designa la matriz identidad $n \times n$. Ir al paso 3.

3. $A_k d^k = -\nabla f(x^k)$. Ir al paso 4.

4. Tomar

$$m_k = \min\{j \in \mathbb{Z}^+ : f(x^k + \beta^j d^k) \leq f(x^k) + \beta^j \sigma \nabla f(x^k)^T d^k\},$$

$$\rho_k = \beta^{m_k} \text{ y } x^{k+1} = x^k + \rho_k d^k.$$

Ir al paso 5.

- 5.

- Si $\|x^{k+1} - x^k\| < \sqrt{\varepsilon} \max\{1, \|x^{k+1}\|\}$, $f(x^k) - f(x^{k+1}) < \varepsilon \max\{1, |f(x^{k+1})|\}$ y $\|\nabla f(x^{k+1})\| < \sqrt[3]{\varepsilon} \max\{1, |f(x^{k+1})|\} \Rightarrow \text{STOP}$.
- Si $\|\nabla f(x^{k+1})\| < \varepsilon_f \Rightarrow \text{STOP}$.
- Si no $\Rightarrow$ Incrementar el contador de iteraciones $k \rightarrow k + 1$. Volver a 2.

NOTA: En el algoritmo anterior ε_f denota la precisión de cálculo del gradiente de f .

Método BFGS

1. Tomar un punto inicial $x^0 \in \mathbb{R}^n$; $\sigma \in [0.001, 0.1]$; $\beta \in [0.7, 0.9]$; $\varepsilon > 0$; $\varepsilon_f > 0$; $B_0 = I_n$; $k = 0$. Ir al paso 2.
2. Tomar $B_k d^k = -\nabla f(x^k)$. Ir al paso 3.
3. Tomar $\rho_k > 0$ satisfaciendo la regla de Wolfe

$$\begin{cases} h_k(\rho_k) \leq h_k(0) + \rho_k \sigma h'_k(0) \\ h'_k(\rho_k) \geq \beta h'_k(0) \end{cases}$$

con $h_k(\rho) = f(x^k + \rho d^k)$, probando inicialmente con

$$\rho_k = \begin{cases} \frac{-2 \max\{1, |f(x^0)|\}}{\nabla f(x^0)^T d^0} & \text{si } k = 0 \\ 1 & \text{en otro caso.} \end{cases}$$

Hacer $x^{k+1} = x^k + \rho_k d^k$. Ir al paso 4.

4.
 - Si $\|x^{k+1} - x^k\| < \sqrt{\varepsilon} \max\{1, \|x^{k+1}\|\}$, $f(x^k) - f(x^{k+1}) < \varepsilon \max\{1, |f(x^{k+1})|\}$ y $\|\nabla f(x^{k+1})\| < \sqrt[3]{\varepsilon} \max\{1, |f(x^{k+1})|\} \Rightarrow$ STOP.
 - Si $\|\nabla f(x^{k+1})\| < \varepsilon_f \Rightarrow$ STOP.
 - Si no $\Rightarrow$ Ir al Paso 5.
5. Hacer $s^k = x^{k+1} - x^k$ y $y^k = \nabla f(x^{k+1}) - \nabla f(x^k)$.

$$\begin{cases} \text{Si } s^{kT} y^k < 0.2 s^{kT} B_k s^k \Rightarrow \theta_k = \frac{0.8 s^{kT} B_k s^k}{s^{kT} B_k s^k - s^{kT} y^k} \text{ y } y^k = \theta_k y^k + (1 - \theta_k) B_k s^k. \\ \text{Si } k = 0 \Rightarrow B_0 = \frac{y^{0T} y^0}{s^{0T} y^0} I_n. \\ B_{k+1} = B_k + \frac{y^k y^{kT}}{s^{kT} y^k} - \frac{B_k s^k s^{kT} B_k}{s^{kT} B_k s^k}. \end{cases}$$

Incrementar el contador de iteraciones $k \rightarrow k + 1$. Volver a 2.

Ejercicios del Capítulo 2

1. Estudiar la existencia de solución para los siguientes problemas de optimización. Resuelve con la máxima precisión que te permita tu ordenador aquellos que tengan solución ($n = 10$). Comprueba si el punto calculado es un mínimo local estricto.

$$\min_{x \in \mathbb{R}^3} f(x) = \exp(x_1 + x_2 + x_3)$$

$$\min_{x \in \mathbb{R}^3} f(x) = x_1^2 - x_1x_2 - 2x_3x_2 + 2x_2^2 + x_3^2 - 5x_1 - 10x_3$$

$$\min_{x \in \mathbb{R}^2} f(x) = x_1^6 + 4x_1^2x_2 + x_2^2 - 3x_1 + x_2$$

$$\min_{x \in \mathbb{R}^2} f(x) = x_1^6 + 4x_1^2x_2 + \frac{1}{4}x_2^2 - 3x_1 + x_2$$

$$\min_{x \in \mathbb{R}^2} f(x) = x_1^4 + 4x_1^2x_2 + x_2^2 - 3x_1 + x_2$$

$$\min_{x \in \mathbb{R}^n} f(x) = \left(\sum_{j=1}^n x_j \right) \exp \left(\sum_{j=1}^n x_j \right) + \sum_{j=1}^n x_j^2 + \sum_{j=1}^n jx_j$$

$$\min_{x \in \mathbb{R}^3} f(x) = x_1^2 + 3x_2^2 + \frac{5}{2}x_3^2 - 3x_1x_2 + x_1x_3 + x_2x_3 - x_1 + x_2 - 2x_3 + \text{sen}(x_1x_2x_3)$$

$$\min_{x \in \mathbb{R}^5} f(x) = 4(x_1^2 + x_2^2) - 8x_3^2 - 12x_4^2 + 12x_1x_2 + 5x_1(x_3 + x_4) \\ + 8x_1x_5 + 2x_2x_3 - 4x_3(x_4 + x_5) - x_5(7x_4 + x_2) + x_1 - x_5$$

$$\min_{x \in \mathbb{R}^4} f(x) = 3(x_1^2 + x_3^2) + 2x_2^2 + 29x_4^2 - x_1x_2 + 2x_1x_3 + 4x_1x_4 - x_2x_3 \\ - 3x_2x_4 + 6x_4x_3 + x_2 + x_3 - x_4$$

$$\min_{x \in \mathbb{R}^3} f(x) = x_1^3 + x_2^4 + x_3^2 + 3x_1^4x_2^4$$

$$\min_{x \in \mathbb{R}^5} f(x) = 3x_1^2 + x_2^2 - (x_3^2 + x_4^2) + 5x_5^2 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_2 - x_5$$

$$\min_{x \in \mathbb{R}^3} f(x) = \log(1 + x_1^2 + 2x_2^2 + 3x_3^2) + x_1^4 + x_2^2 + x_3^2 - 2x_1x_2 - x_2x_3 - x_1 - x_2 - x_3$$

$$\min_{x \in \mathbb{R}^n} f(x) = 0.001 \exp \left(\sum_{j=1}^n x_j^2 \right) + \text{sen} \left(\sum_{j=1}^n x_j \right) + \sum_{j=1}^n jx_j^3$$

$$\min_{x \in \mathbb{R}^n} f(x) = \exp \left(\sum_{i=1}^n \frac{x_i}{i} \right) + \log(1 + \|x\|^2) + \text{sen} \left(\prod_{i=1}^n x_i \right)$$

$$\min_{x \in \mathbb{R}^4} f(x) = x_1^4 - x_2^3 + x_3^4x_4^2 + e^{x_2}(1 + x_1^2 + x_3^2) + x_3 + x_4$$

$$\min_{x \in \mathbb{R}^3} f(x) = x_1^4 + x_2^2 + x_3^2 + 3x_1x_2 + 6x_1x_3 + x_2x_3 + x_1 + x_2 + x_3 + \exp(x_1 - x_2 + x_3)$$

$$\min_{x \in \mathbb{R}^3} f(x) = x_1 \cos(x_1x_2x_3) + x_2 \text{sen}(x_1x_2x_3) + x_1^2x_2^2x_3^2$$

$$\min_{x \in \mathbb{R}^n} f(x) = \sum_{j=1}^n x_j \operatorname{sen} x_j + \exp \left(\sum_{j=1}^n j x_j \right) + \frac{1}{1 + \sum_{j=1}^n x_j^2}$$

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{\exp \left(\sum_{j=1}^n x_j \right)}{1 + \|x\|^2} + \operatorname{sen} \left(\sum_{j=1}^n x_j \right)$$

2. Estudiar si alguno de los problemas siguientes posee mínimos locales o globales y resolverlo en su caso

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} x^T H x + p^T x,$$

donde

$$(2.1) \quad H = \begin{pmatrix} 4 & -1 & -3 & -2 \\ -1 & 2 & -6 & -9 \\ -3 & -6 & -9 & -12 \\ -2 & -9 & -12 & -13 \end{pmatrix} \quad p = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

$$(2.2) \quad H = \begin{pmatrix} 15 & 33 & 51 & 7 & 3 \\ 33 & 78 & 123 & 16 & 9 \\ 51 & 123 & 195 & 25 & 15 \\ 7 & 16 & 25 & 5 & 3 \\ 3 & 9 & 15 & 3 & 9 \end{pmatrix} \quad p = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 2 \\ -2 \end{pmatrix}$$

$$(2.3) \quad H = \begin{pmatrix} 3 & 3 & 6 & 5 & 13 & 16 \\ 3 & 9 & 20 & 5 & 21 & 33 \\ 6 & 20 & 45 & 10 & 44 & 70 \\ 5 & 5 & 10 & 11 & 27 & 26 \\ 13 & 21 & 44 & 27 & 79 & 94 \\ 16 & 33 & 70 & 26 & 94 & 142 \end{pmatrix} \quad p = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ -5.25 \end{pmatrix}$$

$$(2.4) \quad H = \begin{pmatrix} 3 & 3 & 6 & 5 & 13 & 16 \\ 3 & 9 & 20 & 5 & 21 & 33 \\ 6 & 20 & 45 & 10 & 44 & 70 \\ 5 & 5 & 10 & 11 & 27 & 26 \\ 13 & 21 & 44 & 27 & 79 & 94 \\ 16 & 33 & 70 & 26 & 94 & 142 \end{pmatrix} \quad p = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -2 \\ 1 \\ -3 \end{pmatrix}$$

$$(2.5) \quad H = \begin{pmatrix} 1 & 2 & 3 & -1 & 0 & -1 \\ 2 & 6 & 14 & -6 & 0 & -6 \\ 3 & 14 & 44 & -28 & 3 & -16 \\ -1 & -6 & -28 & 40 & -5 & 0 \\ 0 & 0 & 3 & -5 & 12 & 3 \\ -1 & -6 & -16 & 0 & 3 & 18 \end{pmatrix} \quad p = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}$$

3. Dado el problema de optimización

$$(P) \quad \min_{x \in \mathbb{R}^4} f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

resuélvelo con la máxima precisión que puedas mediante el método de Newton, con cálculo analítico del hessiano, comenzando las iteraciones en el punto

$$x^0 = \begin{pmatrix} 3 \\ -1 \\ 0 \\ 1 \end{pmatrix}.$$

¿Ha sido la convergencia cuadrática? ¿Cuál es la solución exacta del problema? Compara la solución obtenida con la exacta y explica los resultados.

Capítulo 3

Cálculo de Valores y Vectores Propios. Métodos de Potencias, QR y Bisección

3.1 Introducción

Dada una matriz A real o compleja, $n \times n$, nos planteamos el problema de hallar números $\lambda \in \mathbb{C}$ y vectores $u \in \mathbb{C}^n$, con $u \neq 0$, tales que

$$\boxed{Au = \lambda u}$$

Este es el problema de determinación de los valores y vectores propios de una matriz. Se dice que λ es un valor propio o autovalor de A y u es un vector propio o autovector de A asociado a λ . La ecuación anterior se puede escribir de forma equivalente como $(A - \lambda I)u = 0$, lo que prueba que el vector propio u es un elemento del núcleo de $A - \lambda I$ y consecuentemente la matriz $A - \lambda I$ es singular. Por lo tanto, los valores propios de A son soluciones de la ecuación $\det(A - \lambda I) = 0$. Puesto que $p(\lambda) = \det(A - \lambda I)$ es un polinomio de grado n , el Teorema Fundamental del Álgebra implica que $p(\lambda)$ posee exactamente n raíces (reales o complejas) contadas con su multiplicidad. Esto implica que A posee exactamente n valores propios. Además si A es una matriz real, entonces los coeficientes del polinomio p son reales, lo que significa que si λ es una raíz compleja (consecuentemente, valor propio de A), entonces $\bar{\lambda}$ es también raíz de p (valor propio de A).

Se llama espectro de A al conjunto de sus valores propios y escribiremos frecuentemente:

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

El radio espectral de la matriz es el número

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|.$$

3.2 Localización de los Valores Propios de una Matriz

El siguiente teorema permite obtener cotas superiores sobre los valores propios.

TEOREMA 3.1 *Para cualquier norma matricial $\|\cdot\|$ se verifica la desigualdad $\rho(A) \leq \|A\|$.*

Los siguientes teoremas proporcionan información sobre la distribución de los valores propios en el plano complejo.

TEOREMA 3.2 (Primer Teorema de Geršgorin) *Cada valor propio de la matriz A se encuentra en al menos uno de los siguientes círculos:*

$$R_i = \left\{ \lambda \in \mathbb{C} : |\lambda - a_{ii}| \leq r_i = \sum_{j=1, j \neq i}^n |a_{ij}| \right\} \quad 1 \leq i \leq n.$$

El resultado se mantiene si reemplazamos los círculos anteriores por los siguientes

$$D_j = \left\{ \lambda \in \mathbb{C} : |\lambda - a_{jj}| \leq r_j = \sum_{i=1, i \neq j}^n |a_{ij}| \right\} \quad 1 \leq j \leq n.$$

Los círculos R_i y D_j se llaman círculos de Geršgorin.

TEOREMA 3.3 (Segundo Teorema de Geršgorin) *Dada una matriz A , supongamos que m de sus círculos de Geršgorin constituyen un conjunto disjunto del resto de los círculos. Entonces hay exactamente m autovalores de A en la unión de estos círculos.*

3.3 Diagonalización y Triangularización de una Matriz

Es fácil comprobar que vectores propios de A asociados a valores propios diferentes son linealmente independientes. Por lo tanto, si existen n valores propios distintos de la matriz A y $\mathcal{B} = \{u_1, \dots, u_n\}$ es una familia de vectores propios asociados, $Au_i = \lambda_i u_i$ $1 \leq i \leq n$, entonces $\mathcal{B}$ es una base de $\mathbb{C}^n$. Si denotamos por $U = [u_1 \dots u_n]$ la matriz cuyas columnas j son las coordenadas del vector u_j respecto de la base canónica $\{e_j\}_{j=1}^n$ de $\mathbb{C}^n$, entonces se tiene $A = UDU^{-1}$, donde D es la matriz diagonal con $d_{jj} = \lambda_j$, $1 \leq j \leq n$. En este caso se dice

que la matriz A es diagonalizable, D es su forma diagonal y U es la matriz de cambio. También resulta sencillo comprobar que si cada valor propio múltiple λ_j , con multiplicidad m_j tiene asociados m_j vectores propios linealmente independientes, entonces A también es diagonalizable y la matriz de cambio se obtiene como antes. En caso contrario, si para algún λ_j su multiplicidad es mayor que la dimensión del núcleo de $A - \lambda_j I$, entonces A no es diagonalizable.

Recordemos aquí que una matriz A se dice hermitiana o hermítica si $A = A^*$, donde $A^* = \bar{A}^T$ (traspuesta y conjugada). En el caso de matrices reales, la definición anterior es equivalente a la igualdad $A = A^T$, y en este caso se dice que A es simétrica.

TEOREMA 3.4 (Teorema Espectral) *Si A es una matriz hermítica, entonces $\sigma(A) \subset \mathbb{R}$ y existe una base ortonormal $\mathcal{B} = \{u_1, \dots, u_n\}$ de $\mathbb{C}^n$ formada por vectores propios de A .*

Como consecuencia de lo anterior, tenemos que toda matriz hermítica es diagonalizable. Además, la matriz de cambio $U = [u_1, \dots, u_n]$ asociada a la base ortonormal $\mathcal{B}$ es obviamente unitaria ($U^{-1} = U^*$). Así se tiene: $A = UDU^*$. En el caso de una matriz real, los vectores de la base pueden tomarse reales. En consecuencia, la matriz de cambio U es ortogonal, es decir, $U^{-1} = U^T$.

Aunque no todas las matrices son diagonalizables, si podemos conseguir una matriz triangular semejante. Recordemos que A y B se dicen semejantes si existe una matriz inversible P tal que $A = PBP^{-1}$.

TEOREMA 3.5 (Teorema de Triangularización de Schur) *Para toda A matriz compleja, $n \times n$, existen una matriz unitaria U y una matriz triangular superior T , $n \times n$, tal que $A = UTU^*$. Además la diagonal de T coincide con el espectro de A .*

TEOREMA 3.6 (Teorema de Triangularización de Schur Real) *Sea A una matriz real, $n \times n$. Entonces existen una matriz ortogonal Q , $n \times n$, y una matriz T , $n \times n$, tal que $A = QTQ^T$, con*

$$T = \begin{pmatrix} T_{11} & T_{12} & \dots & T_{1m} \\ 0 & T_{22} & \dots & T_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & T_{mm} \end{pmatrix}$$

donde cada T_{ii} ($1 \leq i \leq m$) es un escalar o una matriz 2×2 . Los escalares T_{ii} corresponden a los valores propios reales de A , mientras que los bloques T_{ii} , 2×2 , poseen dos valores propios complejos conjugados de A .

3.4 El Método de Potencias

En esta sección estudiamos el cálculo aproximado de los valores extremos de una matriz, es decir, los valores más grandes en valor absoluto y los más pequeños,

también en valor absoluto. Supondremos que los valores propios de A están ordenados como sigue

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|.$$

3.4.1 Cálculo del Valor Propio Dominante

Sea A una matriz real o compleja $n \times n$. Bajo ciertas condiciones, el algoritmo

1-Tomar un vector inicial x_0 , con $\|x_0\|_\infty = 1$

2-Para $j \geq 0$ hacer

$$\hat{x}_j = Ax_j$$

$$k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\}$$

$$\sigma_{j+1} = \hat{x}_{j,k_j}$$

$$x_{j+1} = \frac{1}{\sigma_{j+1}} \hat{x}_j$$

produce sucesiones $\{\sigma_j\}_{j=1}^\infty$ y $\{x_j\}_{j=1}^\infty$ convergentes: $\lim_{j \rightarrow \infty} \sigma_j = \lambda_1$ y $\lim_{j \rightarrow \infty} x_j = u_1$, donde u_1 es un vector propio de A asociado al valor propio λ_1 , $Au_1 = \lambda_1 u_1$. En particular, esta convergencia se produce siempre que $|\lambda_1| > |\lambda_2|$ y $x_0 = \sum_{j=1}^n \alpha_j u_j$, con $\alpha_1 \neq 0$. Aunque esta última condición es más teórica que práctica puesto que, por acción del redondeo, los vectores x_j poseen una componente no nula en la dirección u_1 .

3.4.2 Cálculo del Valor Propio de Módulo Mínimo

Supongamos ahora que λ_n es el valor propio de módulo mínimo de A , entonces el *Método de Potencias Inversas* siguiente proporciona sucesiones $\{\sigma_j\}_{j=1}^\infty$ y $\{x_j\}_{j=1}^\infty$ convergentes: $\lim_{j \rightarrow \infty} \sigma_j = \lambda_n$ y $\lim_{j \rightarrow \infty} x_j = u_n$, donde u_n es un vector propio de A asociado al valor propio λ_n , $Au_n = \lambda_n u_n$.

1-Tomar un vector inicial x_0 , con $\|x_0\|_\infty = 1$

2-Para $j \geq 0$ hacer

$$A\hat{x}_j = x_j$$

$$k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\}$$

$$\sigma_{j+1} = \frac{1}{\hat{x}_{j,k_j}}$$

$$x_{j+1} = \frac{1}{\hat{x}_{j,k_j}} \hat{x}_j$$

3.4.3 Cálculo del Valor Propio Más Próximo a un Número Dado

Sean $\mu \in \mathbb{C}$ y λ_i un valor propio de A tal que $|\lambda_i - \mu| < |\lambda_k - \mu|$ para todo valor propio λ_k con $k \neq i$. Entonces λ_i y un vector propio asociado u_i pueden

ser aproximados por el siguiente *Método de Potencias Inversas*.

$$\begin{aligned}
 &1\text{--Tomar un vector inicial } x_0, \text{ con } \|x_0\|_\infty = 1 \\
 &2\text{--Para } j \geq 0 \text{ hacer} \\
 &\quad (A - \mu I_n)\hat{x}_j = x_j \\
 &\quad k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\} \\
 &\quad s_j = \hat{x}_{j,k_j} \\
 &\quad \sigma_{j+1} = \mu + \frac{1}{s_j} \\
 &\quad x_{j+1} = \frac{1}{s_j}\hat{x}_j
 \end{aligned}$$

donde I_n denota la matriz identidad $n \times n$. Entonces se tiene $\lim_{j \rightarrow \infty} \sigma_j = \lambda_i$ y $\lim_{j \rightarrow \infty} x_j = u_i$, con $Au_i = \lambda_i u_i$.

3.4.4 Utilización de Shifts

En los métodos anteriores podemos utilizar un shift que es actualizado en cada iteración y conduce a una velocidad de convergencia muy rápida. Para ello se requiere previamente disponer de una aproximación razonable del valor propio y de un vector propio asociado y a continuación proceder como se señala en el algoritmo siguiente.

$$\begin{aligned}
 &1\text{--Tomar un par inicial } (\sigma_0, x_0) \approx (\lambda_i, u_i), \text{ con } \|x_0\|_\infty = 1 \\
 &2\text{--Para } j \geq 0 \text{ hacer} \\
 &\quad (A - \sigma_j I_n)\hat{x}_j = x_j \\
 &\quad k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\} \\
 &\quad s_j = \hat{x}_{j,k_j} \\
 &\quad \sigma_{j+1} = \sigma_j + \frac{1}{s_j} \\
 &\quad x_{j+1} = \frac{1}{s_j}\hat{x}_j
 \end{aligned}$$

3.4.5 El Cociente de Rayleigh para Matrices Hermíticas

En el caso de matrices hermíticas, una alternativa al método de potencias para el cálculo del valor propio dominante visto anteriormente es la variante

de Rayleigh:

1-Tomar un vector inicial x_0 , con $\|x_0\|_\infty = 1$

2-Para $j \geq 0$ hacer

$$\hat{x}_j = Ax_j$$

$$k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\}$$

$$\sigma_{j+1} = \frac{x_j^* \hat{x}_j}{x_j^* x_j}$$

$$x_{j+1} = \frac{1}{\hat{x}_{j,k_j}} \hat{x}_j$$

Este algoritmo produce sucesiones $\{\sigma_j\}_{j=1}^\infty$ y $\{x_j\}_{j=1}^\infty$ convergentes: $\lim_{j \rightarrow \infty} \sigma_j = \lambda_1$ y $\lim_{j \rightarrow \infty} x_j = u_1$.

También podemos calcular el valor propio de módulo mínimo en la forma siguiente:

1-Tomar un vector inicial x_0 , con $\|x_0\|_\infty = 1$

2-Para $j \geq 0$ hacer

$$A\hat{x}_j = x_j$$

$$\sigma_{j+1} = \frac{x_j^* x_j}{x_j^* \hat{x}_j}$$

$$k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\}$$

$$x_{j+1} = \frac{1}{\hat{x}_{j,k_j}} \hat{x}_j$$

Así se obtienen sucesiones $\{\sigma_j\}_{j=1}^\infty$ y $\{x_j\}_{j=1}^\infty$ convergentes: $\lim_{j \rightarrow \infty} \sigma_j = \lambda_n$ y $\lim_{j \rightarrow \infty} x_j = u_n$.

Por último, también es posible utilizar shifts para acelerar la convergencia del método. Si tenemos una aproximación razonable (σ_0, x_0) de un valor propio λ_i y un vector propio asociado u_i de A , se puede proceder como sigue.

1-Tomar un par inicial $(\sigma_0, x_0) \approx (\lambda_i, u_i)$, con $\|x_0\|_\infty = 1$

2-Para $j \geq 0$ hacer

$$(A - \sigma_j I_n) \hat{x}_j = x_j$$

$$\sigma_{j+1} = \sigma_j + \frac{x_j^* x_j}{\hat{x}_j^* x_j}$$

$$k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\}$$

$$x_{j+1} = \frac{1}{\hat{x}_{j,k_j}} \hat{x}_j$$

Este algoritmo produce sucesiones $\{\sigma_j\}_{j=1}^\infty$ y $\{x_j\}_{j=1}^\infty$ convergentes: $\lim_{j \rightarrow \infty} \sigma_j = \lambda_i$ y $\lim_{j \rightarrow \infty} x_j = u_i$.

3.4.6 Iteración Inversa para el Cálculo de Vectores Propios

Supongamos que σ es una buena aproximación de un valor propio λ_i : $|\lambda_i - \sigma| \ll |\lambda_j - \sigma|$ para cualquier otro valor propio $\lambda_j \neq \lambda_i$. Entonces podemos obtener en muy pocas iteraciones una buena aproximación de un vector propio u_i correspondiente a λ_i mediante la iteración inversa siguiente

$$\begin{aligned} &1\text{-Tomar un vector inicial } x_0, \text{ con } \|x_0\|_\infty = 1 \\ &2\text{-Para } j \geq 0 \text{ hacer} \\ &\quad (A - \sigma I_n)\hat{x}_j = x_j \\ &\quad k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\} \\ &\quad x_{j+1} = \frac{1}{\hat{x}_{j,k_j}} \hat{x}_j \end{aligned}$$

Un test de parada razonable para este algoritmo es $\|Ax_j - \sigma x_j\|_\infty < C\varepsilon_M \|A\|_\infty$, donde ε_M es la precisión de la máquina y $C > 1$. La elección $5 \leq C \leq 10$ provoca la parada del algoritmo después de una a tres iteraciones en la mayoría de los casos.

3.4.7 Método de Deflacción

Supongamos conocido el valor propio λ_1 de la matriz A y un vector propio asociado v_1 . Construyamos la siguiente matriz de Householder

$$\alpha = \begin{cases} \|v_1\| & \text{si } v_{1,1} = 0 \\ \frac{v_{1,1}}{|v_{1,1}|} \|v_1\| & \text{si } v_{1,1} \neq 0 \end{cases} \quad \begin{cases} u = v_1 + \alpha e_1 \\ H = I - \frac{2uu^*}{u^*u} \end{cases} \quad \text{donde } e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Observemos que H posee las siguientes propiedades:

1. $H^* = H$.
2. $H^2 = I$.
3. $\sigma(H) = \{1, -1\}$, donde el 1 es un valor propio de multiplicidad $n - 1$.
4. $\det(H) = -1$.
5. $Hv_1 = -\alpha e_1$.

Además tenemos

$$HAH = \begin{pmatrix} \lambda_1 & b_1^T \\ 0 & A_1 \end{pmatrix}$$

donde $b_1 \in \mathbb{C}^{n-1}$ y A_1 es una matriz $(n - 1) \times (n - 1)$ con valores propios $\{\lambda_2, \dots, \lambda_n\}$. Si por algún procedimiento calculamos el valor propio λ_2 de A_1 ,

con $\lambda_2 \neq \lambda_1$, y un vector propio asociado w_2 , entonces el vector

$$v_2 = H\hat{w}_2, \quad \text{con} \quad \hat{w}_2 = \begin{pmatrix} \frac{b_1^T w_2}{\lambda_2 - \lambda_1} \\ w_2 \end{pmatrix}$$

es un vector propio de A asociado a λ_2 .

3.5 El Método QR

3.5.1 El Método QR Básico

Sea A una matriz real o compleja $n \times n$. Sabemos que A puede factorizarse en la forma $A = QR$, donde Q es una matriz unitaria ($QQ^* = Q^*Q = I_n$) y R es una matriz triangular superior, ambas $n \times n$. Si A es real, entonces Q y R son reales y Q es ortogonal. Esta factorización se conoce como factorización QR. Sobre Matlab la factorización QR se puede obtener mediante el comando qr : $[Q, R] = qr(A)$.

Bajo ciertas condiciones, las iteraciones

$$\begin{aligned} 1-A_0 &= A \\ 2-Para \ j \geq 0 \text{ hacer} \\ &A_j = Q_j R_j \\ &A_{j+1} = R_j Q_j \end{aligned}$$

producen una sucesión de matrices $\{A_j\}_{j=1}^{\infty}$ tal que $\lim_{j \rightarrow \infty} A_j = T$, donde T es la forma de Schur real o compleja de A , más precisamente

- Si A es compleja $\Rightarrow T$ es triangular superior con diagonal coincidente con el espectro de A .
- Si A es real $\Rightarrow T$ es triangular superior por bloques, con bloques diagonales 2×2 o 1×1 . Los dos valores propios de los bloques 2×2 se corresponden con dos valores propios complejos y conjugados de A . Los bloques 1×1 son los valores propios reales de A .

En el caso de una matriz real A , la estructura de T es

$$T = \begin{pmatrix} T_{11} & T_{12} & \dots & T_{1m} \\ 0 & T_{22} & \dots & T_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & T_{mm} \end{pmatrix}$$

3.5.2 El Algoritmo QR con Matrices Hessenberg

Para favorecer la convergencia del método QR y para reducir el número de operaciones del algoritmo, se reduce primeramente la matriz A a la forma Hessenberg superior H y luego se aplica el método QR a la matriz H , que posee los mismos valores propios que A . La matriz H posee la estructura

$$H = \begin{pmatrix} \times & \times & \times & \times & \dots & \times & \times \\ \times & \times & \times & \times & \dots & \times & \times \\ 0 & \times & \times & \times & \dots & \times & \times \\ 0 & 0 & \times & \times & \dots & \times & \times \\ 0 & 0 & 0 & \ddots & \ddots & \times & \times \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \times & \times \end{pmatrix}$$

La reducción de A a la forma Hessenberg superior puede realizarse mediante el uso de transformaciones de Householder de manera que las matrices A y H son semejantes. Todavía más, existe una matriz unitaria P tal que $A = PHP^*$. Sobre Matlab, se pueden obtener H y P mediante el comando `hess`: $[P, H] = \text{hess}(A)$. Si solamente se quiere obtener H , entonces basta con hacer $H = \text{hess}(A)$.

Las matrices Hessenberg superior tienen la ventaja de que pueden factorizarse en la forma QR con un número de operaciones muy reducido frente a las que se deben realizar con una matriz plena, lo que supone un ahorro computacional importante cuando se desea aplicar el método QR para calcular los valores propios. Para obtener la factorización QR de H el método más eficiente es el uso de las rotaciones de Givens. Para explicar como funcionan estas matrices, comencemos trabajando en $\mathbb{C}^2$. Sea $x \in \mathbb{C}^2$, con $x \neq 0$. Se considere la matriz

$$U = \begin{pmatrix} \bar{c} & -\bar{s} \\ s & c \end{pmatrix}, \quad \text{donde } c = \frac{x_1}{\|x\|} \text{ y } s = -\frac{x_2}{\|x\|},$$

$\bar{c}$ y $\bar{s}$ siendo los conjugados de los complejos c y s , respectivamente. Entonces, U posee las siguientes propiedades

1. U es unitaria

$$UU^* = \begin{pmatrix} \bar{c} & -\bar{s} \\ s & c \end{pmatrix} \begin{pmatrix} c & \bar{s} \\ -s & \bar{c} \end{pmatrix} = \begin{pmatrix} |c|^2 + |s|^2 & 0 \\ 0 & |c|^2 + |s|^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

2. $\det(U) = |c|^2 + |s|^2 = 1$.

3. $Ux = \begin{pmatrix} \|x\| \\ 0 \end{pmatrix}$

Dada la matriz de Hessenberg H podemos conducirla a la forma triangular R mediante el uso de las matrices de Givens

$$G_1 = \begin{pmatrix} U_{2,1} & 0 \\ 0 & I_{n-2} \end{pmatrix}, \dots, G_i = \begin{pmatrix} I_{n-1} & 0 & 0 \\ 0 & U_{i+1,i} & 0 \\ 0 & 0 & I_{n-i+1} \end{pmatrix}, \dots, G_{n-1} = \begin{pmatrix} I_{n-2} & 0 \\ 0 & U_{n,n-1} \end{pmatrix}$$

donde las matrices $U_{i+1,i}$ son de la forma

$$U_{i+1,i} = \begin{pmatrix} \bar{c}_i & -\bar{s}_i \\ s_i & c_i \end{pmatrix}, \text{ con } c_i = \frac{x_{i,1}}{\|x_i\|^2}, \quad s_i = -\frac{x_{i,2}}{\|x_i\|^2}, \text{ y } x_i = \begin{pmatrix} h_{ii}^i \\ h_{i+1,i}^i \end{pmatrix}.$$

Si ponemos $H_1 = H$ y $H_1 = (h_{ij}^1)_{1 \leq i,j \leq n}$, entonces $G_1 H_1$ provoca la anulación del elemento $h_{2,1}$, obteniéndose una nueva matriz $H_2 = (h_{ij}^2)_{1 \leq i,j \leq n}$ que es Hessenber superior poseyendo ceros en la primera columna por debajo del elemento h_{11}^2 . Si ahora calculamos $H_3 = G_2 H_2$, entonces la primera columna de H_2 coincide con la de H_1 , el elemento $h_{3,2}$ se anula y la matriz H_3 sigue siendo Hessenber superior. Por repetición de este proceso se consigue $R = G_{n-1} \dots G_2 G_1 H$ una matriz triangular superior. Además $Q = G_1^* \dots G_{n-1}^*$ es unitaria y $H = QR$. Finalmente, para realizar una iteración del método QR, debemos calcular $RQ = RG_1^* \dots G_{n-1}^*$ que vuelve a ser una matriz Hessenber superior.

Resumiendo, el método QR aplicado a una matriz Hessenber superior H genera una sucesión de matrices Hessenber superior $\{H_j\}_{j=1}^\infty$ que, bajo ciertas condiciones, converge a una matriz de Schur T que contiene los valores propios de H . El paso de H_j hacia H_{j+1} requiere del orden de $O(n^2)$ operaciones, mientras que para matrices plenas se requieren $O(n^3)$. La función programada en Matlab *qrh.m*, realiza una iteración del método QR a partir de una matriz Hessenber: $H = qrh(H)$.

NOTA 3.7 Si durante la aplicación del algoritmo QR se anula un elemento subdiagonal $h_{k+1,k}$, entonces la matriz H debe subdividirse en dos submatrices: $H_1 = (h_{ij})_{1 \leq i,j \leq k}$ y $H_2 = (h_{ij})_{k+1 \leq i,j \leq n}$. Cada una de las dos matrices resultantes es Hessenber superior y entre las dos contienen todos los valores propios de H . Entonces se aplicará el método QR a cada uno de ellas por separado. La estructura de H es

$$\left(\begin{array}{cccc|cccc} \times & \times & \times & \dots & \times & \times & \times & \times & \times & \dots & \times & \times \\ \times & \times & \times & \dots & \times & \times & \times & \times & \times & \dots & \times & \times \\ 0 & \times & \times & \dots & \times & \times & \times & \times & \times & \dots & \times & \times \\ 0 & 0 & \times & \ddots & \times & \times & \times & \times & \times & \dots & \times & \times \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \times & \times & \times & \times & \times & \dots & \times & \times \\ \hline 0 & 0 & 0 & \dots & 0 & 0 & \times & \times & \times & \dots & \times & \times \\ 0 & 0 & 0 & \dots & 0 & 0 & \times & \times & \times & \dots & \times & \times \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & \times & \times & \dots & \times & \times \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \times & \ddots & \times & \times \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & \times & \times \end{array} \right) = \left(\begin{array}{c|c} H_1 & M \\ \hline 0 & H_2 \end{array} \right).$$

En la práctica no debemos esperar a la anulación de un elemento $h_{k+1,k}$. Es suficiente que $|h_{k+1,k}| \leq C\varepsilon_M(|h_{kk}| + |h_{k+1,k+1}|)$ para una constante C pequeña,

donde ε_M denota la precisión de la máquina. Esto se justifica porque errores de redondeo del orden $\varepsilon_M \|H\|$ ya están presentes en la matriz.

3.5.3 Uso de Shifts para Acelerar la Convergencia

Para acelerar la velocidad de convergencia del método QR cuando se aplica a una matriz Hessenberg superior hay estrategias para elegir shifts convenientes. Para matrices complejas la técnica más empleada es el shift de Wilkinson que detallaremos a continuación. En la iteración k , disponemos de una matriz Hessenberg superior $H_k = (h_{ij}^{(k)})_{1 \leq i, j \leq n}$. La idea de Wilkinson consiste en tomar el valor propio μ_k de la submatriz de H_k

$$\begin{pmatrix} h_{n-1, n-1}^{(k)} & h_{n-1, n}^{(k)} \\ h_{n, n-1}^{(k)} & h_{n, n}^{(k)} \end{pmatrix}$$

que sea más próximo a $h_{n, n}^{(k)}$. Entonces el algoritmo queda en la forma siguiente

- 1-Tomar $H_0 = H$
- 2-Para $k \geq 0$
 - Calcular el shift de Wilkinson μ_k
 - Factorizar $H_k - \mu_k I_n = Q_k R_k$
 - Hacer $H_{k+1} = R_k Q_k + \mu_k I_n$.

La función *shift.m* calcula el shift de Wilkinson de una matriz.

3.5.4 Autovalores de Matrices Reales

Cuando la matriz H es real, es deseable trabajar en el campo real por razones de velocidad de los cálculos. Esto no es posible, en general, si utilizamos el shift de Wilkinson porque éste puede ser complejo en algún momento, lo que provoca que las matrices H_k subsiguientes sean complejas. Sin embargo, es posible evitar la aritmética compleja mediante la realización de una iteración QR de doble shift. Veamos esto. Si la matriz 2×2 , que es real,

$$\begin{pmatrix} h_{n-1, n-1}^{(k)} & h_{n-1, n}^{(k)} \\ h_{n, n-1}^{(k)} & h_{n, n}^{(k)} \end{pmatrix}$$

posee valores propios complejos, éstos deben ser conjugados, sean $\mu_{k,1}$ y $\mu_{k,2}$. Entonces, realizamos dos iteraciones consecutivas del método QR tomando como shifts $\mu_{k,1}$ y $\mu_{k,2}$.

1. $H_k - \mu_{k,1} I_n = \hat{Q}_k \hat{R}_k$
2. $\hat{H}_{k+1} = \hat{R}_k \hat{Q}_k + \mu_{k,1} I_n$
3. $\hat{H}_{k+1} - \mu_{k,2} I_n = \hat{Q}_{k+1} \hat{R}_{k+1}$

$$4. \hat{H}_{k+2} = \hat{R}_{k+1}\hat{Q}_{k+1} + \mu_{k,2}I_n$$

Entonces $\hat{H}_{k+2}$ es esencialmente real, más precisamente, existe una matriz diagonal unitaria D ($d_{ij} = 0$ si $i \neq j$ y $|d_{ii}| = 1$) tal que $H_{k+2} = \bar{D}\hat{H}_{k+2}D$ es real. Nuestro objetivo es determinar H_{k+2} sin realizar los cálculos anteriores que nos obligan a la aritmética compleja. Se puede comprobar que el siguiente algoritmo nos proporciona un modo alternativo de calcular H_{k+2} realizando los cálculos en aritmética real.

1. Calcular $t_k = h_{n-1,n-1}^{(k)} + h_{nn}^{(k)}$ y $s_k = h_{n-1,n-1}^{(k)}h_{nn}^{(k)} - h_{n,n-1}^{(k)}h_{n-1,n}^{(k)}$.
2. Formar $N_k = H_k^2 - t_k H_k + s_k I_n$.
3. Factorizar $N_k = Q_k R_k$.
4. Obtener $H_{k+2} = Q_k^T H_k Q_k$.

Esto se conoce como iteración QR de doble shift explícita.

Desafortunadamente la estrategia propuesta requiere demasiadas operaciones, del orden de $O(n^3)$. Sin embargo, podemos modificarla de manera eficiente en la forma siguiente.

1. Calcular la primera columna n_1 de N_k .
2. Hallar una matriz de Householder P_0 tal que $P_0 n_1 = -\alpha e_1$.
3. Hacer $H'_k = P_0^T H_k P_0$.
4. Transformar H'_k en una matriz Hessenberg superior H'_{k+2} mediante la utilización de matrices de Householder.

Entonces, se puede probar que las matrices H_{k+2} y H'_{k+2} son esencialmente coincidentes, más precisamente, existe una matriz diagonal real con elementos diagonales iguales a ± 1 tal que $H'_{k+2} = D H_{k+2} D$. Esta estrategia se conoce como iteración QR de doble shift implícito. Éste es el método habitual para calcular los valores propios de una matriz real no simétrica.

La función programada en Matlab *doblegqr.m* realiza una iteración implícita del método QR con doble shift de Wilkinson aplicada a una matriz Hessenberg superior H , obteniendo de nuevo una matriz Hessenberg superior semejante a la primera. La sintaxis es $H = \text{doblegqr}(H)$.

La experiencia muestra que solamente de 5 a 9 iteraciones son necesarias para calcular un valor propio λ_n siguiendo las estrategias anteriormente descritas. Sin embargo, hay situaciones excepcionales en que esta estrategia falla. Consideremos el siguiente ejemplo

$$H = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}$$

H es una matriz Hessenberg superior y al mismo tiempo unitaria, así que su factorización QR es $H = QR$, con $Q = H$ y $R = I_n$. Además tenemos que $\mu_1 = \mu_2 = 0$. Por lo tanto, si $H_0 = H$, entonces $H_1 = H_0$ y el algoritmo no produce ningún progreso. Debido a la existencia de estas situaciones especiales, la mayoría de los códigos que implementan el método QR incluyen un shift *excepcional*. La estrategia que siguen es la siguiente:

1. El procedimiento estándar es realizar una iteración de doble shift implícito.
2. Si después de un cierto número de iteraciones (10 es el habitual) no se ha detectado ningún valor propio, entonces se realiza un paso doble con shifts *excepcionales*. A continuación se procede con el doble shift implícito estándar hasta conseguir un valor propio. Si éste no se consigue después de 9 iteraciones, otro paso doble con shifts excepcionales es realizado, seguido de no más de 9 iteraciones con el shift implícito estándar. Si después de estas últimas todavía no se ha identificado un valor propio, el programa abandona.

El valor del shift excepcional no es importante, pero debería ser del mismo orden de magnitud que los elementos de la matriz. La siguiente estrategia es usada frecuentemente: se realiza una iteración de doble shift implícita tomando $t_k = 1.5(|h_{n-1,n-2}^{(k)}| + |h_{n,n-1}^{(k)}|)$ y $s_k = (|h_{n-1,n-2}^{(k)}| + |h_{n,n-1}^{(k)}|)^2$. Notemos que las raíces de $p_k(\lambda) = \lambda^2 - t_k\lambda + s_k$ son complejas y conjugadas.

3.5.5 Cálculo de los Vectores Propios

Una vez aplicado el método QR a la matriz A para calcular sus valores propios, podemos desear determinar también los vectores propios. Una forma eficiente de hacer esto es calcular primero los vectores propios de la matriz Hessenberg superior H semejante a A . Para ello se aplica el método de iteración inversa visto en el capítulo anterior. Una vez obtenidos los vectores propios de H , $\{v_1, \dots, v_n\}$, podemos calcular los vectores propios de A , $\{u_1, \dots, u_n\}$, mediante las transformaciones $u_i = Pv_i$, $i = 1, \dots, n$, donde P es la matriz unitaria que reduce A a la forma Hessenberg superior $H = P^*AP$ o equivalentemente $A = PHP^*$. Recordemos que H y P son propocionados por el comando *hess* de Matlab: $[P, H] = \text{hess}(A)$.

3.6 Matrices Hermíticas

3.6.1 El Método QR para Matrices Hermíticas

Si A es una matriz compleja hermítica (o real y simétrica), entonces podemos reducir A a la forma Hessenberg de tal manera que el resultado es una matriz H real, tridiagonal y simétrica. Esto es proporcionado por la función Matlab *hess.m*. En este caso, todos los valores propios de A (y de H , puesto que son los mismos) son reales. Además, el shift de Wilkinson es un número real y consecuentemente todas las operaciones son en el campo real. Esto justificaría el uso

de la función *qrh.m* combinada con *shift.m* para obtener los valores propios de H . Sin embargo, podemos aprovechar la simetría de H y su carácter tridiagonal para ahorrar un número significativo de operaciones durante la iteración del método QR. Este es el cometido de las funciones *qrt.m* y *shifts.m*. La primera sustituye a *qrh.m* y la segunda a *shift.m*.

3.6.2 Método de Bisección

Dada una matriz compleja hermítica A (o real y simétrica), el objetivo que nos planteamos es calcular los valores propios de A situados en un intervalo real $[a, b]$. El primer paso consiste en reducir A a la forma H real, tridiagonal y simétrica, para calcular después los valores propios de H situados en el intervalo $[a, b]$. El soporte teórico del método que vamos a proponer lo constituye la Ley de Inercia de Sylvester, que explicaremos a continuación.

Sea una matriz real y simétrica M , $n \times n$. Se llama inercia de la matriz M al triplete $(\nu(M), \zeta(M), \pi(M))$, donde $\nu(M)$ es el número de valores propios negativos de M , $\zeta(M)$ es la multiplicidad del valor propio cero ($\zeta(M) = 0$ si M es invertible) y $\pi(M)$ es el número de valores propios positivos de M . Se tiene por lo tanto: $\nu(M) + \zeta(M) + \pi(M) = n$.

Se dice que dos matrices reales y simétricas M y N son congruentes si existe una matriz invertible S tal que $M = SNS^T$.

TEOREMA 3.8 (Ley de Inercia de Sylvester) *Supongamos que M y N son dos matrices reales, simétricas y congruentes. Entonces la inercia de M y N es la misma.*

Observemos que el número de valores propios de la matriz H en el intervalo $[a, b]$ (suponemos que a y b no son valores propios de H) es igual a $\pi(H - aI_n) - \pi(H - bI_n)$. Si dispusiéramos de un algoritmo eficiente para determinar $\pi(H - sI_n)$ para un número $s \in \mathbb{R}$, entonces podríamos usarlo para averiguar cuántos valores propios posee H en $[a, b]$ e incluso para separarlos. En efecto, si denotamos por

$$k = \pi(H - aI_n) - \pi(H - bI_n),$$

entonces podemos subdividir el intervalo $[a, b]$ en dos subintervalos de igual longitud $[a, (a+b)/2]$ y $[(a+b)/2, b]$ y a continuación podemos calcular

$$\pi(H - aI_n) - \pi(H - \frac{a+b}{2}I_n) \quad \text{y} \quad \pi(H - \frac{a+b}{2}I_n) - \pi(H - bI_n)$$

para determinar cuántos de los k valores propios están en cada uno de los subintervalos $[a, (a+b)/2]$ y $[(a+b)/2, b]$. Este proceso puede continuarse hasta aislar todos los valores propios en subintervalos. Una vez que sepamos que en un determinado intervalo $[\alpha_i, \beta_i]$ hay un solo valor propio λ_i , entonces podemos

calcularlo aplicando iteración inversa con un shift apropiado:

- 1-Tomar $\mu = \frac{\alpha_i + \beta_i}{2}$ y $\sigma_0 = \mu$
- 2-Tomar un vector inicial x_0 , con $\|x_0\|_\infty = 1$
- 3-Para $j \geq 0$ hacer
 - Si $\sigma_j \notin [\alpha_i, \beta_i]$, entonces $\sigma_j = \mu$
 - $(H - \sigma_j I_n)\hat{x}_j = x_j$
 - $k_j = \min\{k : |\hat{x}_{j,k}| = \|\hat{x}_j\|_\infty\}$
 - $s_j = \hat{x}_{j,k_j}$
 - $\sigma_{j+1} = \sigma_j + \frac{1}{s_j}$
 - $x_{j+1} = \frac{1}{s_j}\hat{x}_j$

donde I_n denota la matriz identidad $n \times n$. Entonces se tiene $\lim_{j \rightarrow \infty} \sigma_j = \lambda_i$ y $\lim_{j \rightarrow \infty} x_j = v_i$, con $Hv_i = \lambda_i v_i$. De esta forma podemos determinar los k valores propios de H , así como sus vectores propios asociados, en el intervalo $[a, b]$ una vez que han sido aislados en diferentes subintervalos $[\alpha_i, \beta_i]$. Como ya indicamos anteriormente, una vez calculados los vectores propios v_i de H , podemos determinar los correspondientes vectores propios de A mediante la transformación $u_i = Pv_i$, donde P es la matriz unitaria que conduce A a la forma real, tridiagonal y simétrica: $H = P^*AP$.

Nos queda por precisar un algoritmo eficiente para determinar $\pi(H - sI_n)$ para cualquier número real s . Para ello notemos que si

$$H - sI_n = \begin{pmatrix} d_1 & e_2 & 0 & \dots & 0 & 0 \\ e_2 & d_2 & e_3 & \dots & 0 & 0 \\ 0 & e_3 & d_3 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \ddots & \ddots & e_n \\ 0 & 0 & \dots & \dots & e_n & d_n \end{pmatrix}$$

entonces podemos factorizar $H - sI_n$ en la forma $H - sI_n = L_s D_s L_s^T$, con L_s y D_s teniendo la siguiente estructura

$$L_s = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ \gamma_2 & 1 & 0 & \dots & 0 & 0 \\ 0 & \gamma_3 & 1 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & \dots & \gamma_n & 1 \end{pmatrix} \quad \text{y} \quad D_s = \begin{pmatrix} \delta_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & \delta_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & \delta_3 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & \dots & 0 & \delta_n \end{pmatrix}$$

mediante las fórmulas

$$\left\{ \begin{array}{l} \delta_1 = d_1 \\ \gamma_2 = \frac{e_2}{\delta_1} \end{array} \right. \quad \text{y para } j=2, \dots, n \quad \left\{ \begin{array}{l} \delta_j = d_j - \frac{e_j^2}{\delta_{j-1}} \\ \gamma_{j+1} = \frac{e_{j+1}}{\delta_j} \end{array} \right.$$

Ahora, de la Ley de Inercia de Sylvester deducimos que $\pi(H - sI_n) = \pi(D_s) =$ número de δ_j mayores que cero. En realidad, solamente necesitamos calcular los δ_j , pudiendo prescindir de los γ_j . Notemos que la factorización anterior es posible siempre que $\delta_j \neq 0$ para cada $j < n$. Si suponemos que H no posee un valor propio en el intervalo $[s - \varepsilon_M, s + \varepsilon_M]$, donde ε_M es la precisión de la máquina, entonces podemos reemplazar δ_j por un número ε muy pequeño $0 < \varepsilon \ll \varepsilon_M$ en el caso de que δ_j se anule. Esto equivaldría a calcular $\pi(H + D_{\varepsilon, s} - sI_n)$, donde $D_{\varepsilon, s}$ es una matriz diagonal con elementos diagonales nulos o iguales a ε . Ya que H no posee un valor propio en el intervalo $[s - \varepsilon_M, s + \varepsilon_M]$, entonces $\pi(H - sI_n) = \pi(H + D_{\varepsilon, s} - sI_n)$. La función programada en Matlab `ldltm.m` realiza la tarea de determinar $\pi(H - sI_n)$ siguiendo el algoritmo anterior. Su sintaxis es $p = \text{ldltm}(H, s)$, con $p = \pi(H - sI_n)$.

3.7 Análisis de Sensibilidad de Valores y Vectores Propios

La matriz A cuyos valores propios se desean calcular no se conoce exactamente en la mayor parte de los casos. En muchos casos, la matriz A es fruto de computaciones numéricas o de ciertas medidas. En ambos casos hay errores de redondeo (o errores propios del método numérico) o de medición. Así, en la práctica, nosotros tenemos una matriz $A + \delta A$, con $\|\delta A\|$ pequeño. Pero incluso en el caso de conocer exactamente la matriz A , como paso previo al método QR, se calcula la forma Hessenberg H de la matriz. Una vez más los errores de redondeo nos proporcionan una matriz $H + \delta H$ que es la forma Hessenberg de una cierta matriz $A + \delta A$. Por lo tanto, en todos los casos estamos aproximando los valores propios de una matriz $A + \delta A$. Esto nos plantea la cuestión de cómo cercanos son los valores propios de $A + \delta A$ a los de la matriz A . Aun más, nos gustaría estimar su diferencia en términos de $\|\delta A\|$. Este es el objetivo de esta sección.

Como paso previo, vamos a analizar otra cuestión de importancia práctica y que está relacionada con la anterior. Como tests de parada en el cálculo de los vectores propios hemos considerado el residual $r = Ax - \lambda x$. Si $\|r\|$ es pequeño, ¿podemos asegurar que x y λ son aproximaciones aceptables de un vector propio y su valor propio asociado? El siguiente teorema muestra que esta cuestión se reduce también al análisis de la sensibilidad de A .

TEOREMA 3.9 *Sea A una matriz real o compleja $n \times n$, $(v, \lambda) \in \mathbb{C}^n \times \mathbb{C}$, con $\|v\|_2 = 1$, y $r = Av - \lambda v$. Entonces (v, λ) es un par formado por un vector propio y valor propio correspondiente a una matriz $A + \delta A$, donde $\|\delta A\|_2 = \|r\|_2$.*

3.7. ANALISIS DE SENSIBILIDAD DE VALORES Y VECTORES PROPIOS 39

La demostración es inmediata. Es suficiente tomar $\delta A = -rv^*$ y observar que $\|\delta A\|_2 = \|r\|_2$ y $(A + \delta A)v = \lambda v$.

Ahora analizamos la sensibilidad del espectro de la matriz A respecto de pequeñas perturbaciones de sus valores. En primer lugar, observemos que los valores propios de A son funciones continuas de sus elementos a_{ij} . En efecto, es suficiente observar que los coeficientes del polinomio característico de A , $p(\lambda) = \det(A - \lambda I)$, son función continua de los elementos a_{ij} y las raíces de cualquier polinomio son función continua de los coeficientes del mismo. Por lo tanto, si $\|\delta A\|$ es suficientemente pequeño, entonces los valores propios de A y $A + \delta A$ serán próximos. Pero esta información no es suficiente en la práctica y necesitamos alguna estimación de la variación de los valores propios en términos de $\|\delta A\|$.

TEOREMA 3.10 (Bauer y Fike) *Sea A una matriz diagonalizable y supongamos que $A = UDU^{-1}$, donde D es la matriz diagonal formada por los valores propios de A . Sea δA una perturbación de A y $\mu \in \mathbb{C}$ un valor propio de $A + \delta A$. Entonces, existe un valor propio λ de A tal que*

$$|\mu - \lambda| \leq \kappa_p(U) \|\delta A\|_p,$$

donde $\kappa_p(U) = \|U\|_p \|U^{-1}\|_p$ es el condicionamiento de U para la norma matricial $\|\cdot\|_p$ inducida por la p -norma vectorial con $1 \leq p \leq \infty$.

COROLARIO 3.11 *Sea A una matriz hermítica y δA una perturbación de A . Supongamos que μ es un valor propio de $A + \delta A$, entonces existe un valor propio λ de A tal que*

$$|\mu - \lambda| \leq \|\delta A\|_2.$$

Demostración. Si A es hermítica, entonces existe una matriz unitaria U tal que $A = UDU^*$, donde D es la matriz diagonal formada por los valores propios de A . Es entonces suficiente aplicar el Teorema de Bauer y Fike junto con el hecho de que $\|U\|_2 = \|U^*\|_2 = 1$ para toda matriz unitaria U . ■

Este corolario prueba que los valores propios de una matriz hermítica están perfectamente condicionados, en el sentido de que una pequeña perturbación de la matriz original produce una perturbación de los valores propios inferior a la perturbación de la matriz. En particular, del Teorema 3.9 y el Corolario 3.11 se deduce que $|\mu - \lambda| \leq \|Av - \mu v\|_2$, donde v es el vector propio calculado con $\|v\|_2 = 1$ asociado al valor propio μ .

Una debilidad del Teorema de Bauer y Fike es que el número de condición $\kappa(U)$ es el mismo para todos los autovalores. Sin embargo, sucede que para una misma matriz puede haber unos autovalores bien condicionados mientras que otros están mal condicionados, en el sentido de que para pequeños valores de $\|\delta A\|$ unos autovalores sufren cambios pequeños mientras otros experimentan cambios importantes. En el teorema siguiente se introduce el número de condición o condicionamiento de un valor propio.

TEOREMA 3.12 *Sea A una matriz $n \times n$ teniendo n valores propios distintos. Sea λ un valor propio de A y u y v vectores propios de A y A^* asociados a λ y $\bar{\lambda}$*

respectivamente, con $\|u\|_2 = \|v\|_2 = 1$. Entonces, se cumple que $\kappa(\lambda) = |u^*v|^{-1}$ es independiente de la elección de u y v y $1 \leq \kappa(\lambda) \leq \kappa_2(U)$. Además, si μ es el valor propio de $A + \delta A$ que aproxima a λ , entonces

$$|\mu - \lambda| \leq \kappa(\lambda)\|\delta A\|_2 + O(\|\delta A\|_2^2).$$

DEFINICIÓN 3.13 Se dice que $\kappa(\lambda)$ es el condicionamiento del autovalor λ .

EJEMPLO 3.14 Consideremos la matriz 10×10 triangular superior

$$A = \begin{pmatrix} 10 & 10 & & & & & & & & \\ & 9 & 10 & & & & & & & \\ & & 8 & 10 & & & & & & \\ & & & & \ddots & \ddots & & & & \\ & & & & & & 2 & 10 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & 1 \end{pmatrix}.$$

Los elementos a_{ij} no marcados son todos nulos. Es obvio que $\sigma(A) = \{10, 9, 8, \dots, 1\}$. Tomemos la matriz A_ε coincidente con A en todos sus elementos, excepto el $(10, 1)$ que es igual a ε . Entonces se tiene que $\|\delta A\|_2 = \|A_\varepsilon - A\|_2 = \varepsilon$. Los cálculos sobre Matlab para $\varepsilon = 10^{-6}$ se presentan en la siguiente tabla

$\sigma(A)$	$\sigma(A_\varepsilon)$	$\kappa(\lambda)$	$ \lambda - \mu $
10	10.0027	0.0453×10^5	0.0027
9	8.9740	0.3612×10^5	0.0260
8	8.0909	1.3264×10^5	0.0909
7	6.6614	2.9308×10^5	0.3386
6	6.4192	4.2810×10^5	0.4192
5	4.5808	4.2810×10^5	0.4192
4	4.3386	2.9308×10^5	0.3386
3	2.9091	1.3264×10^5	0.0909
2	2.0260	0.3612×10^5	0.0260
1	0.9973	0.0453×10^5	0.0027

Si tomamos $\varepsilon = 10^{-5}$, entonces el resultado es el siguiente

$\sigma(A)$	$\sigma(A_\varepsilon)$	$\kappa(\lambda)$	$ \lambda - \mu $
10	10.0256	0.0453×10^5	0.0256
9	$8.6804 + 0.2886i$	0.3612×10^5	0.4306
8	$8.6804 - 0.2886i$	1.3264×10^5	0.7391
7	$6.6427 + 0.9764i$	2.9308×10^5	1.0397
6	$6.6427 - 0.9764i$	4.2810×10^5	1.1689
5	$4.3573 + 0.9764i$	4.2810×10^5	1.1689
4	$4.3573 - 0.9764i$	2.9308×10^5	1.0397
3	$2.3196 + 0.2886i$	1.3264×10^5	0.7391
2	$2.3196 - 0.2886i$	0.3612×10^5	0.4306
1	0.9744	0.0453×10^5	0.0256

3.7. ANALISIS DE SENSIBILIDAD DE VALORES Y VECTORES PROPIOS 41

Utilizando la función `cond.m` de Matlab, obtenemos $\kappa_2(U) \approx 2.27 \times 10^6$. Así el Teorema de Bauer y Fike da una estimación más grande del error $|\lambda - \mu|$ que la obtenida por los números de condición $\kappa(\lambda)$. Éste no es un hecho casual, la desigualdad $\kappa(\lambda) \leq \kappa_2(U)$ se cumple para todas las matrices diagonalizables A .

En el caso de matrices no diagonalizables A , los valores propios múltiples con un subespacio de autovectores asociados de dimensión inferior a la multiplicidad poseen un condicionamiento que podemos definir como ∞ . En efecto, dado un autovalor múltiple λ con autoespacio asociado defectivo, se puede construir una sucesión de matrices con n valores propios diferentes $\{A_j\}_{j=1}^{\infty}$ tal que $A_j \rightarrow A$, teniendo autovalores $\lambda_j \rightarrow \lambda$, tal que $\kappa(\lambda_j) \rightarrow \infty$.

EJEMPLO 3.15 Consideremos la matriz

$$A = \begin{pmatrix} 0 & -2 & +2 \\ -1 & 0 & -1 \\ -2 & -2 & 0 \end{pmatrix}.$$

Es sencillo comprobar que el polinomio característico de A es $p(\lambda) = -\lambda^3$. Por lo tanto, $\lambda = 0$ es un valor propio triple. El autoespacio asociado posee dimensión igual a 1. En efecto, basta notar que el rango de A es 2. Por lo tanto A no es diagonalizable. Si calculamos los valores propios de A sobre Matlab obtendremos

$$\sigma(A) = \{0.9175 \times 10^{-5}, (-0.4588 + 0.7946i) \times 10^{-5}, (-0.4588 - 0.7946i) \times 10^{-5}\}.$$

Conviene recordar aquí que, de acuerdo al corolario del Teorema de Bauer y Fike, los autovalores de una matriz hermítica siempre están bien condicionados, independientemente de su multiplicidad. Consideremos el siguiente ejemplo.

EJEMPLO 3.16 Ejecutemos las siguientes instrucciones Matlab

$$\begin{aligned} A &= \text{diag}([1 \ 1 \ 2 \ 2 \ 3 \ 3]); \text{randn}('seed', 0); \\ [Q \ R] &= \text{qr}(\text{randn}(6, 6)); A = Q^* A^* Q'; \text{sort}(\text{eig}(A)) \end{aligned}$$

El resultado es

```
1.000000000000000
1.000000000000000
2.000000000000000
2.000000000000000
3.000000000000000
3.000000000000000
```

Sin embargo, este buen comportamiento no necesariamente es así para los autovectores asociados. El condicionamiento de un vector propio asociado a un valor propio λ_i depende del condicionamiento de los demás valores propios de la matriz y de su proximidad al valor propio λ_i . Más precisamente, si u_i es un vector propio de A asociado al valor propio λ_i y $u_i + \delta u_i$ es un vector propio de $A + \delta A$ asociado al valor propio $\lambda_i + \delta \lambda_i$, entonces se tiene

$$\frac{\|\delta u_i\|_2}{\|u_i\|_2} \leq \left(\sum_{k \neq i} \frac{\kappa(\lambda_k)}{|\lambda_i - \lambda_k|} \right) \|\delta A\|_2 + O(\|\delta A\|_2^2).$$

3.8 Cálculo de las raíces de un polinomio

Terminamos este capítulo señalando un procedimiento para calcular las raíces de un polinomio. Dado el polinomio $p(x) = x^n + a_{n-1}x^{n-1} + \dots + a_2x^2 + a_1x + a_0$, se llama matriz compañera de p a la matriz

$$A = \begin{pmatrix} 0 & 0 & \dots & 0 & -a_0 \\ 1 & 0 & \dots & 0 & -a_1 \\ 0 & 1 & \dots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -a_{n-1} \end{pmatrix}.$$

Es fácil comprobar por inducción sobre n que $\det(xI - A) = p(x)$. Por lo tanto, las raíces de p coinciden con los valores propios de A , siendo éste un procedimiento frecuentemente utilizado para calcular las raíces de un polinomio.

Ejercicios del Capítulo 3

1. Ejecuta las siguientes instrucciones en Matlab:

```
n=10; randn('seed',100); A=randn(n,n);
```

A continuación calcula el valor propio dominante de A y un vector propio asociado mediante el método de potencias sin la utilización de shifts. Repite los cálculos pero utilizando el shift apropiado. Repite el ejercicio con las matrices generadas en la forma siguiente

```
n=10; randn('seed',1000); A=randn(n,n);A=A+A';  
n=10; randn('seed',1000); A=randn(n,n);
```

¿Qué sucede?

2. Ejecuta las siguientes instrucciones en Matlab:

```
n=10; randn('seed',100); A=randn(n,n)+i*randn(n,n);
```

A continuación calcula el valor propio dominante de A y un vector propio asociado mediante el método de potencias.

3. Dada la matriz A definida en la forma siguiente

```
n=10; randn('seed',-10); A=randn(n,n);
```

calcular el valor propio de módulo mínimo y un vector propio asociado. Repite el ejercicio para la matriz siguiente

```
n=10;randn('seed',100);A=randn(n,n)+i*randn(n,n);
```

4. Ejecuta las siguientes instrucciones en Matlab:

```
n=10; randn('seed',100); A=randn(n,n);A=A+A';
```

A continuación calcular el valor propio dominante utilizando el cociente de Rayleigh así como un vector propio asociado.

5. Aplicación de los Teoremas de Geršgorin. Consideremos la matriz

$$A = \begin{pmatrix} -8 & 1 & 1 & 2 \\ 0 & 20 & 2 & 1 \\ 1 & 2 & 8 & 3 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Aplicando los Teoremas de Geršgorin, probar que todos sus valores propios son reales y distintos. Calcularlos, así como los vectores propios asociados.

6. Caso de una matriz real con valor propio dominante complejo. Si A es una matriz real con valor propio dominante complejo λ_1 , entonces $\lambda_2 = \bar{\lambda}_1$. Supongamos que $|\lambda_1| = |\lambda_2| > |\lambda_3| \geq \dots |\lambda_n|$. Entonces podemos realizar las iteraciones

$$\left\{ \begin{array}{l} \text{Tomar } k = 1, x_1 \in \mathbb{R}^n \text{ con } \|x_1\|_\infty = 1 \\ \left\{ \begin{array}{l} \text{Para } j = 1, 2, 3, x_{k+j} = Ax_{k+j-1} \\ m = \min\{i : |x_{k+3,i}| = \|x_{k+3}\|_\infty\} \\ \text{Para } j=0,1,2,3, x_{k+j} = \frac{x_{k+j}}{x_{k+3,m}} \end{array} \right. \\ \left\{ \begin{array}{l} M_k = x_{k+1,m}^2 - x_{k+2,m}x_{k,m} \\ M_{k+1} = x_{k+2,m}^2 - x_{k+3,m}x_{k+1,m} \\ N_k = x_{k,m}x_{k+3,m} - x_{k+2,m}x_{k+1,m} \end{array} \right. \\ p_k = \frac{N_k}{M_k}, q_k = \frac{M_{k+1}}{M_k}, \text{ Incrementar } k \rightarrow k + 3 \end{array} \right.$$

Entonces $\lim_{k \rightarrow \infty} p_k = p$, $\lim_{k \rightarrow \infty} q_k = q$ y λ_1 y λ_2 son las raíces del polinomio $\lambda^2 + p\lambda + q$.

Aplicar este algoritmo para calcular los valores propios dominantes λ_1 y λ_2 de la matriz A generada en la forma

$$n=10; \text{randn}('seed',0); A=\text{randn}(n,n);$$

Obtener también los vectores propios asociados.

7. Combinar el método de potencias con el de deflacción para calcular los tres valores propios dominantes de las matrices definidas por

$$\begin{aligned} n=10; \text{randn}('seed',100); A=\text{randn}(n,n); \\ n=100; \text{randn}('seed',100); A=\text{randn}(n,n); A=A+A'; \\ n=100; \text{randn}('seed',100); A=\text{randn}(n,n)+i*\text{randn}(n,n); \end{aligned}$$

Calcular también los vectores propios asociados.

8. Hallar los tres valores propios de módulo mínimo de la matriz A definida como sigue

```
n=25; randn('seed',pi); A=randn(n,n);
n=100; randn('seed',200); A=randn(n,n);A=A+A';
n=100; randn('seed',200); A=randn(n,n)+i*randn(n,n);
```

Determinar también un vector propio asociado a cada uno de los valores propios hallados.

9. Sea la matriz A definida como sigue

```
n=100; randn('seed',0); A=randn(n,n);
```

Hallar los tres valores propios de A más próximos a 1, 8 y 10, respectivamente. Calcula también un vector propio correspondiente a cada uno de los tres valores propios.

10. Hallar todos los valores propios y vectores propios asociados de las siguientes matrices

```
n=10; randn('seed',20); A=randn(n,n)+i*randn(n,n);
n=10; randn('seed',20); A=randn(n,n);
n=10; randn('seed',20); A=randn(n,n);A=A+A';
n=10; randn('seed',20); A=randn(n,n)+i*randn(n,n);A=A+A';
```

11. Hallar todos los valores propios de las matrices siguientes situados en los intervalos indicados. Hallar también un vector propio asociado a cada valor propio.

```
n=50; randn('seed',0); A=randn(n,n); A=A+A'; Intervalo: [-2, +2]
n=50; randn('seed',5); A=randn(n,n); A=A+A'; Intervalo: [-pi/2, +pi/2]
n=50; randn('seed',5); A=randn(n,n)+i*randn(n,n); A=A+A';
Intervalo: [-2, +2]
```

12. Hallar todos los valores propios positivos de la siguiente matriz

```
n=10; randn('seed',2000); A=randn(n,n); A=A+A';
```

13. Hallar todos los valores propios negativos de la siguiente matriz

```
n=10; randn('seed',200); A=randn(n,n)+i*randn(n,n); A=A+A';
```