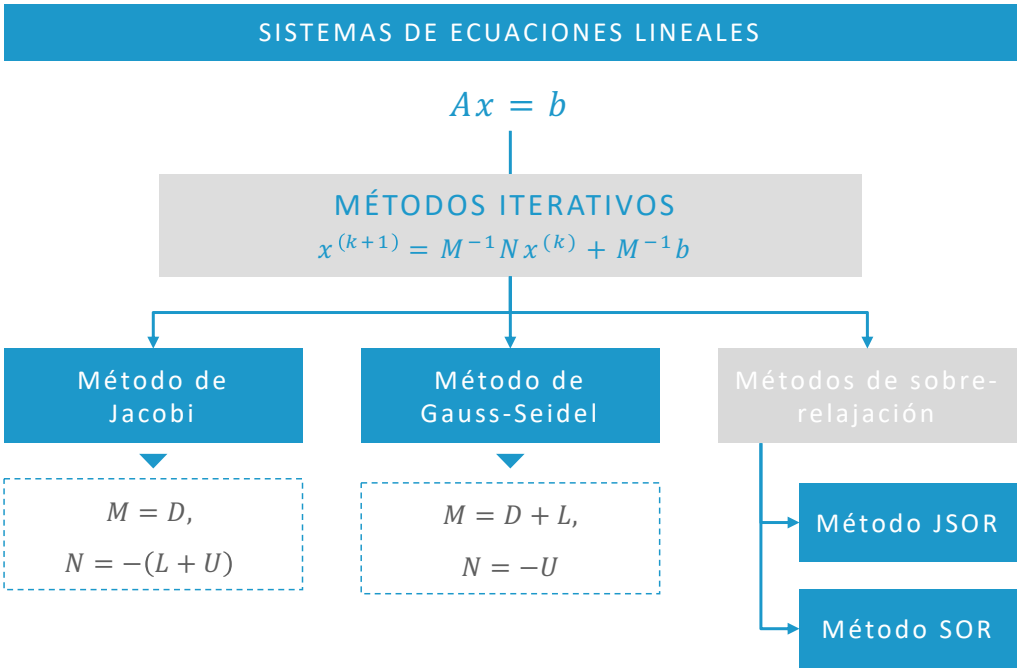
Tema 8

Métodos Numéricos Aplicados I

Sistemas de ecuaciones lineales

Índice

Esquema


SISTEMAS DE ECUACIONES LINEALES
$Ax = b$
MÉTODOS ITERATIVOS
$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b$
Método de Jacobi
Método de Gauss-Seidel
Métodos de sobre-relajación
$M = D,$
$N = -(L + U)$
$M = D + L,$
$N = -U$
Método JSOR
Método SOR

Ideas clave

8.1 Introducción y objetivos

Uno de los problemas fundamentales de los que trata el álgebra lineal es la resolución de sistemas de ecuaciones lineales. Este tipo de problemas surgen, por ejemplo, en estudios de redes eléctricas, modelos económicos o procesos físicos descritos por medio de ecuaciones en derivadas parciales.

En notación matricial, describimos el sistema lineal a resolver como

$$Ax = b,$$

siendo A una matriz real de dimensión $m \times n$ (consideraremos $m = n$) compuesta por los coeficientes del sistema y b un vector de $\mathbb{R}^n$ compuesto por los términos independientes. Existen fundamentalmente dos formas de abordar la resolución de este tipo de sistemas, es decir, el cálculo del vector de incógnitas x: los métodos directos y los métodos iterativos.

Por un lado, los métodos directos utilizan, de una forma o de otra, métodos para factorizar la matriz de coeficientes del sistema y los algoritmos ejecutados se llevan a cabo en un número finito de operaciones. Estos algoritmos de resolución directa consisten en transformar el sistema en otro equivalente cuya resolución sea prácticamente inmediata.

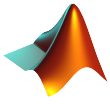
Entre los métodos directos, destacamos el método de Cramer, donde la solución del sistema se obtiene a partir de los determinantes de la matriz A y la matriz modificada, obtenida sustituyendo cada columna de la matriz por el vector de términos indepen-

dientes. Otro método de resolución directa es el método de Gauss-Jordan, que consiste en obtener a partir de la matriz del sistema una matriz que sea triangular superior (o inferior), mediante las llamadas operaciones elementales, y resolver el sistema mediante sustitución regresiva (o progresiva). En ambos casos, el principal inconveniente es el número de operaciones necesarias para alcanzar la solución aunque el proceso sea finito. Además, en el método de Gauss-Jordan los errores de redondeo pueden dispararse si no se elige el pivote adecuado.

Los métodos directos pueden no ser prácticos, sobretodo desde el punto de vista computacional, y en especial cuando la matriz del sistema es de grandes dimensiones, es dispersa o está mal condicionada. Por este motivo, hacemos uso de los métodos iterativos, los cuales producen sucesivas aproximaciones cada vez más próximas a la solución del sistema, de modo que existe la posibilidad de detener el proceso iterativo cuando se haya alzanzado la precisión requerida para la solución.

Dedicaremos este tema a la descripción de los métodos iterativos más sencillos, los cuales nos permitirán obtener la solución del sistema lineal de forma más eficiente que, en genereal, los métodos directos. Por tanto, los objetivos que trataremos de alcanzar son los siguientes:

- Conocer los conceptos básicos de convergencia de los métodos iterativos y el error cometido con la aproximación a la solución del sistema lineal.
- Comprender la expresión iterativa y programación en Matlab de los métodos de Jacobi y Gauss-Seidel.
- Determinar la convergencia de un método iterativo en función de la matriz del sistema y de la matriz de iteración.
- Estudiar los métodos de sobre-relajación como mejora de los métodos clásicos.



Algunas funciones Matlab utilizadas en este tema

- `rref`: resuelve un sistema lineal con el método de Gauss-Jordan
- `triu`, `tril`: extrae la forma triangular superior e inferior, respectivamente, de una matriz
- `cond`: calcula el número de condición de una matriz

8.2 Conceptos básicos

Tal y como se ha introducido anteriormente, en numerosos problemas modelizados mediante sistemas lineales $Ax = b$, la matriz A tiene un tamaño muy grande o es una matriz dispersa, es decir, tiene numerosas entradas nulas distribuidas en general sin un patrón concreto. Estas características desaconsejan el uso de métodos directos y hacen especialemente útil el uso de métodos iterativos para resolver el sistema.

En general, en cualquier método iterativo se obtiene una aproximación al sistema $Ax = b$ construyendo una sucesión de vectores en $\mathbb{R}^n$

$$\{x^{(1)}, x^{(2)}, \ldots, x^{(k)}, \ldots\}$$

partiendo de una vector arbitrario $x^{(0)}$ que se denomina aproximación incial.

Por tanto, si x^* denota el vector solución del sistema, decimos que el método iterativo es convergente si

$$\lim_{k\to\infty} x^{(k)} = x^*.$$

Un método iterativo nunca proporciona la solución exacta incluso en precisión infinita, de modo que definimos el error cometido en la solución como

$$e^{(k)} = x^* - x^{(k)},$$

y el residuo o valor residual en cada iteración como

$$r^{(k)} = b - Ax^{(k)}.$$

Estos conceptos están relacionados de la siguiente forma:

$$\lim_{k\to\infty} x^{(k)} = x^* \quad \Leftrightarrow \quad \lim_{k\to\infty} ||e^{(k)}|| = 0 \quad \Leftrightarrow \quad \lim_{k\to\infty} ||r^{(k)}|| = 0. \qquad (1)$$

Debido a que las soluciones que obtendremos con los métodos iterativos serán aproximadas, los límites (1) nunca alcanzarán el valor 0. Entonces, debemos determinar un valor muy pequeño para el cual consideraremos que hemos alcanzado la solución. Este valor es el que denominamos tolerancia (tol), de modo que habremos alcanzado la precisión deseada para la solución cuando el error absoluto o el error relativo, respectivamente, satisfacen:

$$||e^{(k)}|| < tol \qquad \text{o} \qquad \frac{||e^{(k)}||}{||x^*||} < tol.$$

Sin embargo, como no conocemos el valor de $e^{(k)}$, utilizaremos como condición de parada el criterio del residuo:

$$||r^{(k)}|| < tol \qquad \text{o} \qquad \frac{||r^{(k)}||}{||b||} < tol.$$

Con frecuencia, el residuo será la única información de la cual dispondremos para tratar de acotar el error. Analicemos si el criterio de parada basado en el residuo es conveniente o no. En primer lugar, como

$$r^{(k)} = b - Ax^{(k)} = Ax^* - Ax^{(k)} = A(x^* - x^{(k)}) = Ae^{(k)},$$

entonces

$$r^{(k)} = Ae^{(k)} \quad \Leftrightarrow \quad e^{(k)} = A^{-1}r^{(k)}.$$

Por tanto, tomando normas matriciales

$$||r^{(k)}|| = ||Ae^{(k)}|| \leq ||A|| \cdot ||e^{(k)}|| \quad \Rightarrow \quad ||e^{(k)}|| \geq \frac{||r^{(k)}||}{||A||}$$

y

$$||e^{(k)}|| = ||A^{-1}r^{(k)}|| \leq ||A^{-1}|| \cdot ||r^{(k)}||.$$

Combinando las desigualdades anteriores se tiene

$$\frac{1}{||x^*||}\frac{||r^{(k)}||}{||A||} \leq \frac{||e^{(k)}||}{||x^*||} \leq \frac{||A^{-1}|| \cdot ||r^{(k)}||}{||x^*||}. \tag{2}$$

Como $||b|| = ||Ax^*|| \leq ||A|| \cdot ||x^*||$ y $||x^*|| = ||A^{-1}b|| \leq ||A^{-1}|| \cdot ||b||$, entonces

$$\frac{1}{||A^{-1}|| \cdot ||b||} \leq \frac{1}{||x^*||} \leq \frac{||A||}{||b||}$$

y podemos reescribir (2) como

$$\frac{1}{||A|| \cdot ||A^{-1}||}\frac{||r^{(k)}||}{||b||} \leq \frac{||e^{(k)}||}{||x^*||} \leq ||A|| \cdot ||A^{-1}||\frac{||r^{(k)}||}{||b||}. \tag{3}$$

Definición 1: Número de condición

Definimos el número de condición de una matriz A en una norma matricial $\|\cdot\|$ como el valor

$$\mathcal{K}(A) = \|A\| \cdot \left\|A^{-1}\right\|.$$

Diremos que la matriz A está bien condicionada si el número de condición es un valor pequeño.

De (3), podemos ver que la relación que hay entre el error relativo y el residual relativo depende directamente del número de condición de la matriz del sistema:

$$\frac{1}{\mathcal{K}(A)}\frac{||r^{(k)}||}{||b||} \leq \frac{||e^{(k)}||}{||x^*||} \leq \mathcal{K}(A)\frac{||r^{(k)}||}{||b||}.$$

Por tanto, en las matrices mal condicionadas es difícil controlar el error relativo. El test del residuo es fiable si $\mathcal{K}(A)$ no es muy grande, es decir, si A es una matriz estable.

Otra particularidad de las matrices mal condicionadas es que no soportan un gran número de operaciones, es decir, es difícil controlar el error de redondeo. Veamos a

continuación un ejemplo en el que analizamos el efecto que produce sobre la solución una pequeña perturbación en los datos del sistema a resolver.

Ejemplo 1.

Consideremos el sistema lineal $Ax = b$ donde

$$A = \begin{bmatrix} 2 & 6 \\ 2 & 6.0001 \end{bmatrix}, \qquad b = \begin{bmatrix} 8 \\ 8.0001 \end{bmatrix}.$$

Resolviendo el sistema en Matlab:

```
>> A = [2 6; 2 6.0001];
>> b = [8; 8.0001];
>> x = A\b;
```

obtenemos como solución $x = [1 \quad 1]^t$. Sin embargo, realizando una pequeña perturbación $\alpha = 0.0001$:

$$\tilde{A} = \begin{bmatrix} 2 & 6 \\ 2 & 6.0001 - \alpha \end{bmatrix}, \qquad \tilde{b} = \begin{bmatrix} 8 \\ 8.0001 - \alpha \end{bmatrix},$$

el sistema $\tilde{A}x = \tilde{b}$ tiene como solución $x = [4 \quad 0]^t$. Se trata de una solución muy distinta a la del sistema inicial. Esto se debe a que la matriz A está mal condicionada, ya que si calculamos su número de condición $\mathcal{K}(A)$:

```
>> cond(A)
ans =
   4.0001e+05
```

se obtiene un valor bastante grande. Esta cantidad justifica la inestabilidad de la solución, ya que ante pequeñas variaciones de algunos de los parámetros, la solución ha sufrido un cambio considerable.

Accede al vídeo: Matrices de Hilbert, número de condición y error relativo

Métodos iterativos para sistemas lineales

Dado el sistema lineal

$$Ax = b,$$

si la estructura de la matriz original A es demasiado complicada para obtener una solución directa, podemos sustituirla por una aproximación M que denominamos **precondicionador**, siendo M una matriz invertible, que permita una solución más simple. Partiremos entonces de su diferencia $N = M - A$ y transformaremos el sistema lineal en un sistema de punto fijo equivalente cuya solución se aproxima paso a paso. Reemplazando en el sistema original, se tiene

$$Ax = b \quad \Leftrightarrow \quad (M - N)x = b \quad \Leftrightarrow \quad Mx = Nx + b. \tag{4}$$

El sistema (4) es equivalente al original, de modo que tiene la misma solución. Podemos resolver el sistema de forma iterativa como

$$Mx^{(k+1)} = Nx^{(k)} + b \quad \Leftrightarrow \quad x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b.$$

De esta forma, si la matriz M es fácilmente invertible (es una matriz diagonal, triangular,...), el problema original se reduce a problema mucho más sencillo.

Denotando $H = M^{-1}N$ y $q = M^{-1}b$, tendríamos el proceso iterativo:

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b = Hx^{(k)} + q, \qquad k = 0, 1, \ldots,$$

donde H es la matriz de iteración y $x^{(0)}$ es la aproximación inicial. Se dice que un **método iterativo es estacionario** si la matriz de iteración H es constante en todo el proceso.

Por último, debemos tener en cuenta que no toda matriz M es un precondicionador adecuado. La característica principal que debe tener la matriz precondicionadora es que de lugar a un sistema sencillo de resolver. En los siguientes apartados tomaremos distintos precondicionares obtenidos a partir de la matriz original dando lugar a diferentes métodos iterativos.

8.3 Método de Jacobi

Sea $A = (a_{ij})$ la matriz del sistema lineal $Ax = b$ con $a_{ii} \neq 0$, $i = 1, 2, \ldots, n$. Consideremos la partición de A como suma de las matrices

$$A = L + D + U,$$

donde L es la parte estrictamente trianguar inferior de A, D es la diagonal principal de A y U es la parte estrictamente triangular superior de A. El método de Jacobi es un método iterativo estacionario en el que tomamos $M = D$ y $N = -(L + U)$, de modo que su expresión iterativa es

$$x^{(k+1)} = -D^{-1}(L + U)x^{(k)} + D^{-1}b, \qquad k = 0, 1, 2, \ldots$$

Obtención del esquema iterativo

Supongamos que en el sistema $Ax = b$

$$\left.\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &=& b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &=& b_2 \\ & \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &=& b_n \end{array}\right\}$$

todos los coeficientes de la diagonal satisfacen $a_{ii} \neq 0$. Si despejamos la incógnita x_i de la i-ésima ecuación, obtenemos el sistema equivalente

$$\left.\begin{array}{rcl} x_1 &=& -\frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 - \cdots - \frac{a_{1n}}{a_{11}}x_n + \frac{b_1}{a_{11}} \\ x_2 &=& -\frac{a_{21}}{a_{22}}x_1 - \frac{a_{23}}{a_{22}}x_3 - \cdots - \frac{a_{2n}}{a_{22}}x_n + \frac{b_2}{a_{22}} \\ &\vdots& \\ x_n &=& -\frac{a_{n1}}{a_{nn}}x_1 - \frac{a_{n2}}{a_{nn}}x_2 - \cdots - \frac{a_{nn-1}}{a_{nn}}x_{n-1} + \frac{b_n}{a_{nn}} \end{array}\right\}$$

En forma matricial:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{bmatrix} \left(- \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \right),$$

de modo que el esquema iterativo queda como

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^{(k+1)} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{a_{nn}} \end{bmatrix} \left(- \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^{(k)} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \right),$$

coincidiendo con la expresión iterativa

$$x^{(k+1)} = -D^{-1}(L+U)x^{(k)} + D^{-1}b.$$

Algoritmo del método de Jacobi

- Entrada: matriz A, términos independientes b, estimación inicial x_0, tolerancia tol, máximo número de iteraciones $maxiter$
- Proceso:
  - Inicializar el contador de iteraciones ($iter = 1$) y las variables de control

- Definir las matrices L, U, D y D^{-1} a partir de A
- Cálculo de $M^{-1} = D^{-1}$ y $N = -(L + U)$
- Mientras no se cumpla el criterio de parada ni se alcance el valor de $maxiter$:
  - Calcular $x = -M^{-1}Nx_0 + M^{-1}b$
  - Actualizar criterios de parada y número de iteraciones
  - Actualizar variables: $x_0 = x$

▶ Salida: vector solución x, número de iteraciones $iter$

Ejemplo 2.

Vamos a obtener la solución del sistema lineal

$$\left.\begin{array}{rcl} 10x_1 - x_2 + 2x_3 & = & 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 & = & 25 \\ 2x_1 - x_2 + 10x_3 - x_4 & = & -11 \\ 3x_2 - x_3 + 8x_4 & = & 15 \end{array}\right\}$$

utilizando el método directo de Gauss-Jordan y el método iterativo de Jacobi con estimación inicial $x^{(0)} = [0, 0, 0, 0]^t$ y criterio de parada

$$\frac{||x^{(k+1)} - x^{(k)}||_\infty}{||x^{(k)}||_\infty} < 10^{-3}.$$

Para resolver el sistema lineal con el algoritmo de Gauss-Jordan utilizamos la funicón de Matlab `rref`:

```
>> A=[10 -1 2 0; -1 11 -1 3; 2 -1 10 -1; 0 3 -1 8];
>> b=[6; 25; -11; 15];
>> rref([A b]);
```

Se obtiene como dato de salida de la función la forma escalonada reducida de la

matriz ampliada del sistema:

$$\left[\begin{array}{cccc|c} 10 & -1 & 2 & 0 & 6 \\ -1 & 11 & -1 & 3 & 25 \\ 2 & -1 & 10 & -1 & -11 \\ 0 & 3 & -1 & 8 & 15 \end{array}\right] \sim \left[\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right].$$

Por tanto, la solución exacta del sistema, obtenida con el método de Gauss-Jordan, es $x^* = [1, 2, -1, 1]^t$.

Por otro lado, llamamos `Jacobi.m` a la función de Matlab que implementa este método y definimos los valores de entrada necesarios para ejecutar la función:

```
>> x0=zeros(4,1);
>> tol=1e-3; maxiter=50;
>> [x,iter] = Jacobi(A,b,x0,tol,maxiter);
```

Algunos de los resultados obtenidos se muestran en la Tabla 1.

Solución	$k=0$	$k=1$	$k=2$	$k=3$	$\cdots$	$k=9$	$k=10$
$x_1^{(k)}$	0	0.6000	1.0473	0.9326		1.0006	0.9997
$x_2^{(k)}$	0	2.2727	1.7159	2.0533		1.9987	2.0004
$x_3^{(k)}$	0	-1.1000	-0.8052	-1.0493		-0.9990	-1.0004
$x_4^{(k)}$	0	1.8750	0.8852	1.1309		0.9989	1.0006

Tabla 1: Resultados del método de Jacobi

Coociendo la solución exacta, podemos estimar el error cometido con la solución del método iterativo de Jacobi:

$$||x^* - x^{(10)}||_\infty = 6.1919 \cdot 10^{-4}.$$

Para comparar los tiempos de ejecución de los métodos directo e iterativo, generamos un programa que ejecute cada uno de los métodos 1.000 veces:

```
function [tJ,tGJ]=tiempos(A,b,x0,tol,maxiter)
tGJ=0; tJ=0;
for i=1:1000
    t0=tic;
    rref([A b]);
    tGJ=tGJ+toc(t0);
end
for i=1:1000
    t0=tic;
    Jacobi(A,b,x0,tol,maxiter);
    tJ=tJ+toc(t0);
end
end
```

El tiempo que tarda en ejecutarse Jacobi es 0.0191 segundos, mientras que Gauss-Jordan tarda 0.1997 segundos. Destacar que en cada ejecución del programa los tiempos pueden variar, pero la relación entre ambos tiempos debe ser del mismo orden. Por tanto, el método emplea más tiempo en obtener la solución es el algoritmo de Gauss-Jordan.

8.4 Método de Gauss-Seidel

Considerando de nuevo la misma partición de la matriz del sistema

$$A = L + D + U,$$

si tomamos $M = D + L$ y $N = -U$, obtenemos el método estacionario de Gauss-Seidel cuya expresión iterativa es

$$x^{(k+1)} = -(D+L)^{-1}Ux^{(k)} + (D+L)^{-1}b, \qquad k = 0, 1, 2, \ldots$$

Obtención del esquema iterativo

Dado el sistema lineal $Ax = b$, suponiendo que $a_{ii} \neq 0, \forall i$, si despejamos la incógnita x_i de la i-ésima fila,

$$\left.\begin{array}{rcl} x_1 & = & -\frac{a_{12}}{a_{11}}x_2 - \frac{a_{13}}{a_{11}}x_3 - \cdots - \frac{a_{1n}}{a_{11}}x_n + \frac{b_1}{a_{11}} \\ x_2 & = & -\frac{a_{21}}{a_{22}}x_1 - \frac{a_{23}}{a_{22}}x_3 - \cdots - \frac{a_{2n}}{a_{22}}x_n + \frac{b_2}{a_{22}} \\ & \vdots & \\ x_n & = & -\frac{a_{n1}}{a_{nn}}x_1 - \frac{a_{n2}}{a_{nn}}x_2 - \cdots - \frac{a_{nn-1}}{a_{nn}}x_{n-1} + \frac{b_n}{a_{nn}} \end{array}\right\}$$

el método de Gauss-Seidel, a diferencia del método de Jacobi, propone utilizar los nuevos valores de las incógnitas a medida que se van calculando en cada iteración. Así, una vez calculado el valor de $x_1^{(k+1)}$ en el paso k, este valor se utiliza para actualizar el valor de $x_2^{(k+1)}$, y así de forma sucesiva. De este modo, la expresión iterativa del sistema es

$$\left.\begin{array}{rcl} x_1^{(k+1)} & = & -\frac{a_{12}}{a_{11}}x_2^{(k)} - \frac{a_{13}}{a_{11}}x_3^{(k)} - \cdots - \frac{a_{1n}}{a_{11}}x_n^{(k)} + \frac{b_1}{a_{11}} \\ x_2^{(k+1)} & = & -\frac{a_{21}}{a_{22}}x_1^{(k+1)} - \frac{a_{23}}{a_{22}}x_3^{(k)} - \cdots - \frac{a_{2n}}{a_{22}}x_n^{(k)} + \frac{b_2}{a_{22}} \\ & \vdots & \\ x_n^{(k+1)} & = & -\frac{a_{n1}}{a_{nn}}x_1^{(k+1)} - \frac{a_{n2}}{a_{nn}}x_2^{(k+1)} - \cdots - \frac{a_{nn-1}}{a_{nn}}x_{n-1}^{(k+1)} + \frac{b_n}{a_{nn}} \end{array}\right\}$$

Por tanto, en cada iteración debemos resolver el sistema

$$(D+L)x^{(k+1)} = b - Ux^{(k)}.$$

En el método de Gauss-Seidel las componentes de $x^{(k+1)}$ que ya conocemos se utilizan en la propia iteración $k+1$.

Algoritmo del método de Gauss-Seidel

- Entrada: matriz A, términos independientes b, estimación inicial x_0, tolerancia tol, máximo número de iteraciones $maxiter$
- Proceso:
  - Inicializar el contador de iteraciones ($iter = 1$) y las variables de control
  - Definir las matrices L, U, D y D^{-1} a partir de A
  - Cálculo de $M^{-1} = (D + L)^{-1}$ y $N = -U$
  - Mientras no se cumpla el criterio de parada ni se alcance el valor de $maxiter$:
    - Resolver $(D + L)x = b - Ux_0$
    - Actualizar criterios de parada y número de iteraciones
    - Actualizar variables: $x_0 = x$
- Salida: vector solución x, número de iteraciones $iter$

Ejemplo 3.

Obtén una aproximación a la solución del sistema lineal

$$\left.\begin{aligned} 10x_1 - x_2 + 2x_3 &= 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 &= 25 \\ 2x_1 - x_2 + 10x_3 - x_4 &= -11 \\ 3x_2 - x_3 + 8x_4 &= 15 \end{aligned}\right\}$$

utilizando el método de Gauss-Seidel con estimación inicial $x^{(0)} = [0, 0, 0, 0]^t$ y criterio de parada

$$\frac{||x^{(k+1)} - x^{(k)}||_\infty}{||x^{(k)}||_\infty} < 10^{-3}.$$

Para resolver el sistema lineal con el algoritmo de Gauss-Jordan utilizamos la función de Matlab `rref`:

```
>> A=[10 -1 2 0; -1 11 -1 3; 2 -1 10 -1; 0 3 -1 8];
```

```
>> b=[6; 25; -11; 15];
>> rref([A b]);
```

Si `Gauss_Seidel.m` es la función de Matlab que hemos programado para ejecutar este método, calculamos la solución del sistema ejecutando en Matlab:

```
>> A=[10 -1 2 0; -1 11 -1 3; 2 -1 10 -1; 0 3 -1 8];
>> b=[6; 25; -11; 15];
>> x0=zeros(4,1);
>> tol=1e-3; maxiter=50;
>> [x,iter] = Gauss_Seidel(A,b,x0,tol,maxiter);
```

Los resultados obtenidos se muestran en la Tabla 2.

Solución	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
$x_1^{(k)}$	0	0.6000	1.0302	1.0066	1.0009	1.0001
$x_2^{(k)}$	0	2.3273	2.0369	2.0036	2.0003	2.0000
$x_3^{(k)}$	0	-0.9873	-1.0145	-1.0025	-1.0003	-1.0000
$x_4^{(k)}$	0	0.8789	0.9843	0.9984	0.9998	1.0000

Tabla 2: Resultados del método de Gauss-Seidel

Del Ejemplo 2 sabemos que la solución exacta es $x^* = [1, 2, -1, 1]^t$, de modo que el error cometido con la solución aproximada obtenida con el método de Gauss-Seidel es:

$$||x^* - x^{(5)}||_\infty = 9.1280 \cdot 10^{-5}.$$

Análogamente al Ejemplo 2, ejecutamos el método iterativo 1000 veces para estimar su tiempo de ejecución. Obtenemos que el método de Gauss-Seidel tarda únicamente 0.0497 segundos, siendo de nuevo más rápido que el algoritmo de Gauss-Jordan.

8.5 Convergencia de los métodos iterativos

En este apartado vamos a estudiar algunos aspectos acerca de la convergencia de los métodos iterativos que hemos utilizado para resolver los sistemas de ecuaciones lineales. Concretamente, analizaremos la influencia de los valores propios de la matriz de iteración con la convergencia de los métodos.

Definición 2: Radio espectral

Sean $\lambda_1, \lambda_2, \ldots, \lambda_n$ los valores propios de una matriz A. Definimos el radio espectral de A, denotado $\rho(A)$, como

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i|.$$

Teorema 1

Sea $A \in \mathbb{R}^{n \times n}$ una matriz invertible y $b \in \mathbb{R}^n$. Un método iterativo estacionario con expresión general

$$x^{(k+1)} = Hx^{(k)} + q, \qquad k = 0, 1, 2, \ldots$$

converge a la solución exacta del sistema lineal $Ax = b$ para cualquier aproximación inicial $x^{(0)} \in \mathbb{R}^n$ si, y solo si, la matriz de iteración satisface

$$\rho(H) < 1.$$

Para que el Teorema 1 sea una condición necesaria y suficiente, es necesario que la matriz A sea invertible, ya que sino solo tendríamos una condición suficiente de convergencia.

Teorema 2

Consideremos el sistema lineal $Ax = b$. Si la matriz $I - A$ es invertible, entonces las siguientes condiciones son equivalentes:

- El método iterativo $x^{(k+1)} = Hx^{(k)} + q$, $k \geq 0$, es convergente.
- $\rho(H) < 1$.
- $||H|| < 1$, para alguna norma matricial subordinada.

También podemos proporcionar resultados de convergencia de los métodos iterativos de Jacobi y Gauss-Seidel sin imponer condiciones de invertibilidad.

Definición 3: Matriz estrictamente diagonal dominante

Sea $A = (a_{ij})$ una matriz de dimensión $n \times n$. Decimos que A es estrictamente diagonal dominante si

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

Teorema 3

Si la matriz A es estrictamente diagonal dominante, entonces los métodos de Jacobi y de Gauss-Seidel son convergentes.

El Teorema 3 es una condición suficiente pero no necesaria de convergencia, de modo que los métodos iterativos pueden ser convergentes sin que la matriz del sistema sea estrictamente diagonal dominante. Además, si el sistema inicial $Ax = b$ no tiene una matriz que sea estrictamente diagonal dominante, cabe la posibilidad de efectuar intercambios en las filas de forma que el sistema resultante sí lo es, garantizando la converencia de los métodos.

Ejemplo 4.

Consideremos el sistema lineal $Ax = b$, donde

$$A = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 3 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

La matriz A es invertible pero no es estrictamente diagonal dominante, de modo que el Teorema 3 no permite afirmar nada sobre la convergencia de los métodos iterativos.

Por un lado, en el método de Jacobi se tendría

$$M_J = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \qquad N_J = -\begin{bmatrix} 0 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 0 \end{bmatrix}.$$

De aquí, la matriz de iteración

$$H_J = \begin{bmatrix} 0 & -2/3 & -1/3 \\ -2/3 & 0 & -2/3 \\ -1/3 & -2/3 & 0 \end{bmatrix}.$$

Por otro lado, en el método de Gauss-Seidel:

$$M_{GS} = \begin{bmatrix} 3 & 0 & 0 \\ 2 & 3 & 0 \\ 1 & 2 & 3 \end{bmatrix}, \qquad N_{GS} = -\begin{bmatrix} 0 & 2 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix},$$

siendo la matriz de iteración

$$H_{GS} = \begin{bmatrix} 0 & -2/3 & -1/3 \\ 0 & 4/9 & -4/9 \\ 0 & -2/27 & 11/27 \end{bmatrix}.$$

Calculamos el radio espectral de las dos matrices:

$$\rho(H_J) = 1.1241 > 1, \qquad \rho(H_{GS}) = 0.6083 < 1.$$

El Teorema 1 nos permite afirmar que el método de Jacobi no es convergente para este sistema, mientras que el método de Gauss-Seidel sí. Además, aplicando

el método de Gauss-Seidel con el criterio de parada

$$\frac{\left\|x^{(k+1)} - x^{(k)}\right\|_\infty}{\left\|x^{(k+1)}\right\|_\infty} < 10^{-3}$$

y estimación inicial $x^{(0)} = [1, 1, 1]^t$, se obtiene la solución al sistema con la tolerancia indicada tras 13 iteraciones del método de Gauss-Seidel:

$$x^{(13)} = \begin{bmatrix} 0.6256 \\ -0.5006 \\ 0.1252 \end{bmatrix}.$$

Con las mismas condiciones de parada, el método de Jacobi no converge transcurridas 300 iteraciones.

Sabemos que la convergencia del método de Jacobi no implica la convergenia del método de Gauss-Seidel, y viceversa. En general, tampoco podemos afirmar que un método sea más rápido que otro ya que depedenderá de la matriz del problema. Sin embargo, cuando el método es convergente para un sistema, podemos relacionar su velocidad de convergencia con el radio espectral de la matriz de iteración.

Definición 4: Radio de convergencia

Consideremos un método iterativo con matriz de iteración H. Definimos el radio de convergencia como

$$R = -\log_{10}(\rho(H)).$$

Cuanto más pequeño sea $\rho(H)$, mayor será la convergencia.

En el Ejemplo 5 utilizamos el radio de convergencia para comparar la velocidad de convergencia de los métodos de Jacobi y Gauss-Seidel.

Ejemplo 5.

Consideremos la matriz A,

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}.$$

En el método de Jacobi, la matriz de iteración es

$$H_J = \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{bmatrix}$$

y sus valores propios son $\lambda_1 = \frac{1}{2}$ y $\lambda_2 = -\frac{1}{2}$.

La matriz de iteración del método de Gauss-Seidel es

$$H_{GS} = \begin{bmatrix} 0 & \frac{1}{2} \\ 0 & \frac{1}{4} \end{bmatrix}$$

con valores propios $\lambda_1 = 0$ y $\lambda_2 = -\frac{1}{4}$. Entonces:

$$\rho(H_J) = \frac{1}{2}, \qquad \rho(H_{GS}) = \frac{1}{4}.$$

Como $\rho(H_J) < 1$ y $\rho(H_{GS}) < 1$, y A es una matriz invertible, los dos métodos son convergentes en base al Teorema 1. Además, a partir de los radios de convergencia

$$R_J = -\log_{10}\left(\frac{1}{2}\right) = 0.3010, \qquad R_{GS} = -\log_{10}\left(\frac{1}{4}\right) = 0.6021,$$

podemos decir que para este problema el método de Gauss-Seidel es aproximadamente el doble de rápido que el método de Jacobi.

8.6 Métodos de sobre-relajación

En los métodos de sobre-relajación interviene un parámetro ω del que depende, en gran medida, la convergencia del método. El objetivo es que los errores sean cada vez más pequeños en cada iteración.

Consideremos en primer lugar, como generalización del método de Jacobi, el método de Jacobi relajado:

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_i^{(k)} \right) + (1-\omega) x_i^{(k)}, \qquad i = 1, 2, \ldots, n,$$

donde se ha introducido un parámetro de relajación ω. Denotamos a este método como JSOR, y podemos escribir de forma equivalente su expresión iterativa como

$$x^{(k+1)} = x^{(k)} + \omega D^{-1} r^{(k)}, \qquad k = 0, 1, 2, \ldots,$$

donde $r^{(k)} = b - Ax^{(k)}$ es el vector residual de Jacobi. Además, si el método de Jacobi converge, el método JSOR también converge siempre que $0 \leq \omega \leq 1$.

A continuación, vamos a presentar otro método de sobre-relajación. Recordemos que para definir los métodos de Jacobi y Gauss-Seidel hemos utilizado una partición de la matriz A como $A = L + D + U$, siendo L la parte estrictamente triangular inferior de A, U la parte estrictamente triangular superior y D la diagonal principal de A. Con estas matrices, podemos definir otra descomposición de la matriz A de la forma

$$\omega A = (D + \omega L) - (-\omega U + (1-\omega) D),$$

que da lugar a un método iterativo conocido como el método SOR (Successive Over Relaxation) con expresión:

$$(D + \omega L) x^{(k+1)} = (-\omega U + (1-\omega) D) x^{(k)} + \omega b, \qquad k = 0, 1, 2, \ldots \qquad (5)$$

Notemos que para $\omega = 1$ recuperamos el método de Gauss-Seidel.

Desarrollando escalarmente (5), podemos llegar a

$$\left.\begin{aligned} x_1^{(k+1)} &= (1-\omega)x_1^{(k)} + \frac{\omega}{a_{11}}\left(-a_{12}x_2^{(k)} - \cdots - a_{1n}x_n^{(k)} + b_1\right) \\ x_2^{(k+1)} &= (1-\omega)x_2^{(k)} + \frac{\omega}{a_{22}}\left(-a_{21}x_1^{(k+1)} - \cdots - a_{2n}x_n^{(k)} + b_2\right) \\ &\vdots \\ x_n^{(k+1)} &= (1-\omega)x_n^{(k)} + \frac{\omega}{a_{nn}}\left(-a_{n1}x_1^{(k+1)} - \cdots - a_{nn-1}x_{n-1}^{(k+1)} + b_n\right) \end{aligned}\right\}$$

Si $\bar{x}^{(k+1)}$ denota el iterado $k+1$ del método de Gauss-Seidel, la expresión iterativa vectorial del método SOR se puede escribir como

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega\bar{x}^{(k+1)}, \qquad k = 0, 1, 2, \ldots$$

En el Ejemplo 6 comparamos los resultados obtenidos con los métodos de Gauss-Seidel y sobre-relajación.

Ejemplo 6.

Consideremos el sistema lineal

$$\left.\begin{aligned} x_1 + x_4 &= 2 \\ 2x_2 + x_4 &= 3 \\ 3x_3 + x_4 &= 4 \\ x_1 + x_2 + x_3 + 4x_4 &= 7 \end{aligned}\right\}$$

Vamos a obtener la solución utilizando el método de Gauss-Seidel y el método SOR con $\omega = 1.3$ utilizando para los dos casos como estimación inicial $x^{(0)} = [0, 0, 0, 0]^t$ y el criterio de parada

$$||x^{(k+1)} - x^{(k)}||_\infty < 10^{-7}.$$

Aplicando el método de Gauss-Seidel, para que se cumpla el criterio de parada se necesitan 23 iteraciones, siendo algunos de los resultados los que se muestran en la Tabla 3.

Solución	$k=0$	$k=1$	$k=2$	$k=3$	$\cdots$	$k=23$
$x_1^{(k)}$	0	2.0000	1.4583	1.2101		1.0000
$x_2^{(k)}$	0	1.5000	1.2292	1.1050		1.0000
$x_3^{(k)}$	0	1.3333	1.1528	1.0700		1.0000
$x_4^{(k)}$	0	0.5417	0.7899	0.9037		1.0000

Tabla 3: Resultados del método de Gauss-Seidel

Sin embargo, el método SOR solo requiere de 17 iteraciones para converger con el mismo criterio de parada. Podemos observar en la Tabla 4 resultados parciales obtenidos con este método.

Solución	$k=0$	$k=1$	$k=2$	$k=3$	$\cdots$	$k=17$
$x_1^{(k)}$	0	2.6000	0.9046	1.1424		1.0000
$x_2^{(k)}$	0	1.9500	0.9073	1.0847		1.0000
$x_3^{(k)}$	0	1.7333	0.9082	1.0655		1.0000
$x_4^{(k)}$	0	0.7042	0.9125	0.9741		1.0000

Tabla 4: Resultados del método SOR

En el apartado anterior hemos analizado propiedades de la matriz de coeficientes del sistema y la matriz de iteración para estudiar la convergencia de los métodos iterativos. Por último, mostramos dos resultados relacionados con la convergencia del método SOR dependiendo del valor de ω utilizado.

Teorema 4: Ostrowski-Reich

Si A es una matriz definida positiva y $0 < \omega < 2$, entonces el método SOR converge para cualquier elección de la estimación inicial $x^{(0)}$.

Teorema 5: Young

Si A es una matriz tridiagonal, simétrica y definida positiva, entonces

$$\rho(H_{GS}) = (\rho(H_J))^2 < 1$$

y la elección óptima de ω para el método SOR es:

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(H_{GS})}}.$$

También podemos seleccionar el valor óptimo de ω para resolver un problema en concreto, en el siguiente vídeo mostramos cómo realizar este estudio de forma experimental.

Accede al vídeo: Método de sobre-relajación y selección del parámetro de relajación óptimo.