Universidad de Concepción
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Matemática

Cálculo Numérico (521230)

Laboratorio 4: Estabilidad de sistemas de ecuaciones lineales. Métodos iterativos para la solución de sistemas de ecuaciones lineales.

Generalidades

Consideramos el problema de encontrar $x \in \mathbb{R}^n$ tal que $Ax = b$, dónde A es una matriz invertible con coeficientes reales de orden n y $b \in \mathbb{R}^n$.
En clases se vió que, si x denota la solución exacta al problema $Ax = b$ y $\tilde{x} = x + \delta_x$, la solución exacta a $(A + \delta_A)\tilde{x} = b + \delta_b$, entonces

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \frac{\|A\|\|A^{-1}\|}{1 - \text{cond}(A)\frac{\|\delta_A\|}{\|A\|}} \left(\frac{\|\delta_A\|}{\|A\|} + \frac{\|\delta_b\|}{\|b\|} \right)$$

En caso que $\|A\|\|A^{-1}\|$ (se denomina condición de A y se denota $\text{cond}(A)$) sea muy grande, no es posible garantizar que errores relativos pequeños en A y b (pequeños $\frac{\|\delta_A\|}{\|A\|}$ y $\frac{\|\delta_b\|}{\|b\|}$) produzcan errores relativos pequeños en la solución al sistema de ecuaciones. Si $\text{cond}(A)$ es mucho mayor que 1, se dice que A es mal condicionada. Problemas $Ax = b$ con matriz A mal condicionada son problemas inestables. En Matlab es posible estimar el número de condición en norma 2 de una matriz con el comando `cond`.

Ejercicios

Ejercicio 1 (estabilidad de sistemas de ecuaciones lineales): Las matrices de *Hilbert*

$$H_n = \left(h_{ij}^n\right) \in \mathbb{R}^{n \times n}, \qquad \text{con } h_{ij}^n = \frac{1}{i + j - 1}, \quad i, j = 1, \ldots, n,$$

son matrices muy mal condicionadas. Con el comando `hilb` de Matlab podemos crear la matriz de Hilbert de tamaño n (parámetro de entrada a `hilb`) y con `invhilb` podemos crear su inversa.

1.1 Escriba la función `testingcond` que dado un número $n \in N$, haga lo siguiente:

- *a*) Cree la matriz de Hilbert de tamaño n, H_n, y muestre su número de condición en norma 2.
- *b*) Llame a la función `create_trid` que escribimos en el laboratorio 3 para crear una matriz tridiagonal de tamaño n, T_n, con todos los elementos de su diagonal principal iguales a 4 y los elementos sobre las diagonales secundarias iguales a 1. Muestre también el número de condición de esta matriz.
- *c*) Cree los vectores de tamaño n

$$b = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}, \qquad \tilde{b} = b + \delta_b, \text{ con } |\delta_b(i)| \leq 10^{-6} \ \forall i.$$

  Note que la sentencia Matlab `(2*rand(n,1)-1)*a` genera un vector columna aleatorio de dimensión n, uniformemente distribuido en el intervalo $[-a, a]$. Muestre el número $\frac{\|b - \tilde{b}\|_2}{\|b\|_2}$.
- *d*) Resuelva los sistemas $H_n x = b$, $H_n \tilde{x} = \tilde{b}$ de manera exacta, es decir, mediante $x = H_n^{-1} b$ y $\tilde{x} = H_n^{-1} \tilde{b}$. Muestre el error relativo en la solución, es decir, el número $\frac{\|x - \tilde{x}\|_2}{\|x\|_2}$. Recuerde que con `invhilb` puede calcular H_n^{-1} de manera exacta.
- *e*) Resuelva los sistemas $T_n x = b$ y $T_n \tilde{x} = \tilde{b}$ también de manera exacta. Muestre el error relativo en la solución. La inversa de T_n puede obtenerla llamando a la función `invtrid` (bájela de la página web que le indique el ayudante de su sección).

1.2 Llame a la función `testingcond` y use los resultados que ella muestra para completar la tabla 1.

1.3 Explique los resultados en la tabla.

n	cond(H_n)	cond(T_n)	error rel. en b	error rel. en x (Hilbert)	error rel. en x (tridiagonal)
12					
13					
14					
15					

CUADRO 1. Efectos de matriz mal condicionada

Ejercicio 2 (matrices dispersas en MATLAB): Haga un programa *function* que genere una matriz de la forma

$$B = \begin{pmatrix} 2\,I & -I \\ -I & 2\,I \end{pmatrix} \in \mathbb{R}^{2n\times 2n}.$$

El valor de n debe ser entrada a la función. La matriz identidad de tamaño n puede crearse con ayuda del comando `eye`. Llame a la función para crear $B \in \mathbb{R}^{2000\times 2000}$. Compruebe que B es una matriz dispersa mediante el comando `nnz`(*Number of Non Zeros*) que da la cantidad total de entradas no nulas de la matriz.

2.1 Utilice la siguiente sentencia para almacenar en forma *sparse* otra matriz A igual a B:

```
>> A=sparse(B);
```

Escriba `whos` en la ventana de comandos y observe la cantidad de memoria que requiere el almacenamiento de cada una de esas matrices. Indique cuál forma de almacenamiento resulta más conveniente.

2.2 Genere un vector aleatorio $b \in \mathbb{R}^{2000}$ y compare los tiempos necesarios para resolver los sistemas $Bx = b$ y $Ax = b$. Indique cuál resulta más conveniente. Recuerde que los comandos `tic, toc` pueden usarse para medir el tiempo que demora la ejecución de un conjunto de instrucciones en MATLAB .

Observación: *Hay varios comandos en* MATLAB *que tienen su contraparte para matrices dispersas. Por ejemplo,* `speye` *es semejante a* `eye`*, pero la matriz identidad que genera se almacena como* sparse*.*
Al realizar operaciones con matrices sparse *(por ejemplo, suma, producto, transpuesta,* *, *concatenación, etc.),* MATLAB *almacena los resultados en nuevas matrices* sparse*. Así mismo muchos comandos, cuando se aplican a matrices* sparse*, generan matrices* sparse*. Por ejemplo,* `diag`*,* `tril` *y* `triu`*.*

2.3 Utilice estos comandos para obtener las matrices D, E y F de la descomposición $A = D-E-F$ introducida en clases para derivar los métodos de Jacobi y Gauss-Seidel. Calcule la matriz de iteración del método de Jacobi $J = D^{-1}(E+F)$. Compruebe que las matrices D, E, F y J también se almacenan como matrices *sparse*.

2.4 El comando `spy` permite "espiar" la estructura de una matriz mediante un gráfico con la distribución de sus entradas no nulas. "Espíe" la estructura de las matrices A, D, E, F y J.

2.5 Si bien el almacenamiento de la matriz A requiere mucha menos memoria que el de la matriz B, para generar A como se ha descrito antes fue necesario haber almacenado previamente B. A veces la memoria del computador no alcanza para almacenar B en forma llena. En tal caso, es necesario generar directamente la matriz dispersa A sin pasar nunca por la matriz llena B. Indique alguna forma de hacer esto y compruébelo (modificando la función que escribió para crear B).

2.6 Para algunas aplicaciones es necesario dar la forma llena de una matriz dispersa. El comando inverso de `sparse` que almacena una matriz dispersa en forma de matriz llena es `full`. Compruébelo.

Ejercicio 3 (métodos de Jacobi y Gauss-Seidel): La función `jacobisol`(su ayudante le indicará de dónde bajarla) permite encontrar una aproximación a la solución exacta del sistema $Ax = b$ mediante el método de Jacobi. Esta función tiene cinco parámetros de entrada: la matriz $A \in \mathbb{R}^{n\times n}$, la parte derecha del sistema de ecuaciones a resolver $b \in \mathbb{R}^n$, una aproximación inicial a la solución exacta del sistema de ecuaciones, la tolerancia con que se quiere calcular una aproximación a solución exacta del sistema de ecuaciones y el máximo número de iteraciones a realizar por el método.

3.1 Utilice esta función para resolver el sistema $Ax = b$ con A la matriz tridiagonal de tamaño 50 con cuatros en diagonal principal y unos en las diagonales secundarias (recuerde que esta matriz puede crearse con `create_trid`), b debe ser el vector cuyas 50 componentes son iguales a uno. ¿Logró el método obtener una aproximación a la solución exacta del sistema de ecuaciones con la precisión requerida? ¿Cuántas iteraciones fueron necesarias para ello? ¿Cuál es la norma infinito de la matriz de iteración? ¿Cuál es el radio espectral de la matriz de iteración?

3.2 Almacene la función `jacobisol.m` en `gaussseidelsol.m` y modifíquela de forma que permita obtener una aproximación a la solución exacta de un sistema de ecuaciones lineales mediante el método de Gauss-Seidel. Utilice esta función para resolver el sistema del inciso anterior. ¿Logró el método obtener una aproximación a la solución exacta del sistema de ecuaciones con la precisión requerida? ¿Cuántas iteraciones fueron necesarias para ello? ¿Cuál es la norma infinito de la matriz de iteración? ¿Cuál es el radio espectral de la matriz de iteración?

Observación: *Dado un sistema de ecuaciones lineales $Ax = b$, si descomponemos $A = D - E - F$ como se vió en clases, la matriz de iteración del método de Jacobi es $M_{JAC} = D^{-1}(E + F)$ y la del método de Gauss-Seidel es $M_{GS} = (D - E)^{-1}F$. Las aproximaciones calculadas con los métodos de Jacobi y Gauss-Seidel convergerán a la solución eaxcta del sistema si y sólo si el radio espectral de la matriz de iteración asociada a ellos es estrictamente menor que 1.*

Note que la matriz en el sistema que recién hemos resuelto es estrictamente diagonal dominante. Para matrices como ésta los radios espectrales de las matrices de iteración de los métodos de Jacobi y Gauss-Seidel son estrictamente menores que 1. Además

$$\|M_{GS}\|_\infty \leq \|M_{JAC}\|_\infty < 1, \tag{1}$$

Esta propiedad asegura que el número de iteraciones necesarias por el método de Gauss-Seidel para obtener una determinada precisión en la aproximación calculada será menor que el número de iteraciones necesarias para obtener la misma precisión con el método de Jacobi. Sin embargo, en un sistema de ecuaciones general, la convergencia de ninguno de los dos métodos asegura la convergencia del otro.

3.3 Utilice ambas funciones para resolver los siguientes sistemas de ecuaciones y utilice los resultados para completar la tabla 2. En ella, x_{lu} representa la solución a cada uno de los sistemas usando $\backslash$, x_{jac} a la solución con el método de Jacobi y x_{gs} a la solución con el método de Gauss-Seidel.

Sistema 1: $Bx = c$ con

$$B = \begin{pmatrix} 1 & -2 & 2 \\ -1 & 1 & -1 \\ -2 & -2 & 1 \end{pmatrix}, \qquad c = \begin{pmatrix} 1 \\ -1 \\ 3 \end{pmatrix}.$$

Sistema 2: $Gx = h$ con

$$G = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ -1 & 1 & -1 \\ -\frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix}, \qquad h = \begin{pmatrix} 2 \\ -1 \\ 1 \end{pmatrix}.$$

	Jacobi			Gauss-Seidel		
	$\rho(M_{JAC})$	causa parada	$\frac{\|x_{jac}-x_{lu}\|}{\|x_{lu}\|}$	$\rho(M_{GS})$	causa parada	$\frac{\|x_{gs}-x_{lu}\|}{\|x_{lu}\|}$
$Bx = c$						
$Gx = h$						

CUADRO 2. Métodos de Jacobi y Gauss-Seidel