Imprimir
Note: You can choose to save the module's content as a PDF document, instead of printing.
To do so, click on the 'Print' link above. In the window that appears, select the PDF
printer as the printer of choice, then click 'Print' and enter any additional information needed.

Resolución de sistemas de ecuaciones lineales

En esta Unidad se estudiará la resolución de sistemas de ecuaciones lineales utilizando los comandos y funciones de MATLAB para tal fin.

Introducción

Instructions: Seguir los enlaces al material. También hay parte del material colgado en "Recursos".

Introducción

El problema de los sistemas lineales de ecuaciones es uno de los más antiguos de la matemática y tiene una infinidad de aplicaciones, como en procesamiento digital de señales, análisis estructural, estimación, predicción y más generalmente en programación lineal así como en la aproximación de problemas no lineales de análisis numérico. Al resolver gran cantidad de problemas que surgen en el ámbito científico y técnico aparecen sistemas de ecuaciones lineales, en la mayor parte de los casos con un número muy grande tanto de ecuaciones como de incógnitas. Es en estos casos donde MATLAB muestra su potencia como entorno diseñado para operar con matrices de gran tamaño.

Tal y como veremos más adelante en el curso, otros métodos numéricos como la resolución numérica de ecuaciones diferenciales, conducen sistemas lineales de ecuaciones.

Al ser el tema de sistemas de ecuaciones lineales un tema básico de cálculo numérico, puede consultarse en cualquier buen libro de métodos numéricos.

Para esta unidad vamos a seguir básicamente el capítulo dedicado a sistemas lineales del libro de Cleve Moller.

También hemos colgado en recursos dos documentos en PDF en los que podéis leer una introducción básica a los sistemas de ecuaciones lineales y la factorización LU.

También son útiles los libros citados en la bibliografía.

Sistemas de ecuaciones lineales

Instructions: Seguir los enlaces al material.

Sistemas de ecuaciones lineales

La forma más habitual de escribir un sistema de ecuaciones es en forma matricial:

A X = B

En el caso más habitual, cuando el número de ecuaciones es el mismo que de incógnitas, A es una matriz cuadrada de tamaño nxn, X es un vector columna cuyas componentes son las n incógnitas y B es un vector columna del mismo tamaño. Para MATLAB, X y B pueden ser matrices siempre y cuando el número de filas sea el adecuado.

La forma inmediata de resolver el sistema, sería poner

X=A^{-1}B

donde A^{-1} es la matriz inversa. Teniendo en cuenta que en problemas científicos el orden del sistema de ecuaciones puede ser muy grande y que el cálculo de la inversa de una matriz requiere bastantes operaciones, hay que buscar una forma alternativa de resolver el sistema más eficiente que calcular inversas.

Los métodos de resolución pueden clasificarse en:

Métodos directos: Mediante manipulación algebraica de las ecuaciones se obtiene una solución exacta del sistema siempre y cuando se utilice aritmética exacta en las operaciones. El método directo más conocido y utilizado para la resolución de sistemas lineales es el método de eliminación de Gauss.

Métodos indirectos o iterativos: La solución del sistema se construye de forma iterativa mediante una seria de aproximaciones sucesivas a la solución. Existen varias variantes de estos métodos. La ventaja de estos métodos frente a los primeros está en el coste computacional, ya que en cálculos científicos la matriz del sistema suele ser muy grande y generalmente con un gran número de ceros (matrices dispersas).

Generalmente la forma de la matriz A es la que indica qué método es el más apropiado para la resolución del sistema de ecuaciones. Existen métodos específicamente desarrollados según la estructura de dicha matriz (por ejemplo, para matrices tridiagonales, matrices banda, matrices simétricas...).

El método de Gauss

El algoritmo de Gauss consiste en transformar el sistema de ecuaciones AX=B, mediante operaciones elementales entre las filas de la matriz A, en un sistema equivalente con una nueva matriz de coeficientes que sea triangular o escalonada, cuya resolución se realiza de manera directa mediante "sustitución regresiva o inversa".

Para obtener el sistema triangular, mediante operaciones elementales con las filas de la matriz, se van anulando los elementos situados debajo de la diagonal principal. La fase de triangularización se realiza en distintas etapas en cada una de las cuales se van transformando en cero los elementos por debajo de la diagonal en cada una de las columnas de la matriz. En el algoritmo juega un papel importante el elemento no nulo de la diagonal principal de la matriz elegido como PIVOTE en cada una de las etapas. En ocasiones hay que hacer una permutación entre las filas o columnas de la matriz para llevar el elemento no nulo a la diagonal para poder usarlo como pivote. Siguiendo el proceso tal y como se muestra en el ejemplo, se obtiene finalmente una matriz triangular superior y el sistema correspondiente se resuelve por sustitución regresiva.

La representación matricial de las operaciones elementales realizadas para transformar el sistema en triangular proporciona la factorización LU de la matriz A, donde L es una matriz triangular inferior con unos en la diagonal y U es una matriz triangular superior o escalonada, siendo U la matriz de coeficientes del "nuevo" sistema de ecuaciones triangular que se resuelve por sustitución. El primer elemento no nulo de cada fila de la matiz U es el pivote de la eliminación de Gauss. Los elementos de L situados por debajo de la diagonal principal, llamados multiplicadores contienen información sobre las operaciones que se han realizado.

En el caso de haber tenido que permutar filas o columnas, el proceso es equivalente a encontrar dos matrices de permutación P y Q, tales que PAQ = LU.

El conocimiento de la descomposición de la matriz A en factores triangulares LU facilita la resolución del sistema de ecuaciones AX=B, planteando dos sistemas triangulares:

LY=B, UX=Y

El primero se resuelve por sustitución progresiva y el segundo por sustitución regresiva o inversa. El coste computacional al resolver el sistema de esta manera es menor, ya que el número de operaciones para resolver un sistema triangular es del orden de n^2 mientras que el algoritmo de Gauss completo sobre la matriz A el número de operaciones es del orden de n^3.

En MATLAB la descomposición LU se realiza con la función **lu** (véase también el LU SOLVER). Dada una matriz A:

>> [L,U]=lu(A) produce las matrices L y U.

>> [L,U,P]= lu(A) en caso de que haya que permutar alguna fila.

Para resolver el sistema de ecuaciones escribiremos en MATLAB:

```
X=U\(L\(P*B))
```

En el caso en el que no hubiera que permutar ninguna fila, la matriz de permutación es la identidad, con lo que es indiferente poner la P o no. Por este motivo y para evitar errores, conviene ponerla siempre.

Errores de redondeo

El algoritmo de eliminación de Gauss proporcionaría la solución exacta si se efectuara con aritmética exacta, sin embargo, cuando un problema se resuelve con ayuda de un ordenador las operaciones aritméticas se realizan con un número determinado de decimales. En cada operación se producen errores de redondeo que pueden llegar a acumularse y dar lugar a grandes errores en la solución final. Dicho error de redondeo están relacionados con la elección del pivote en el proceso de eliminación.

Con el objetivo de disminuir los errores de redondeo, se han propuesto diferentes variantes del método de Gauss. Una de ellas, implementada por MATLAB en su función básica de resolución de sistemas y que veremos en la siguiente sección, es el método con "pivotación parcial" consistente en elegir como pivote el elemento con mayor valor absoluto en la columna a eliminar. Dicha elección reduce el error de redondeo.

Para esta parte leed el capítulo 2 del libro de Cleve Moller, allí se explica los efectos de los errores de redondeo que aparecen asociados a la elección de los pivotes.

Resolución de sistemas de ecuaciones lineales con MATLAB

Instructions: Seguir los enlaces al material.

Resolución de sistemas de ecuaciones lineales con MATLAB

El comando "\"

Veamos ya como resolver sistemas de ecuaciones con MATLAB. Para resolver el sistema Ax = b se utiliza la función de MATLAB **mldivide,** representada por el símbolo " \", de la forma:

```
>> x=A \ b
```

Aunque menos común, también puede ejecutarse como

```
>> x = mldivide(A,b)
```

Las matrices A y b deben tener el mismo número de filas. MATLAB muestra un mensaje de error si A está mal escalada o es casi singular (deteminante cercano a cero) aunque hace el cálculo de todas formas.

La matriz de coeficientes A no tiene porqué ser cuadrada, puede ser una matriz rectangular. Si A es una matriz de tamaño mxn (m es el número de ecuaciones y n el de incógnitas), se pueden dar tres situaciones:

- m=n (A cuadrada): entonces x=A \ b busca una solución exacta de la ecuación A*x = b, en caso de existir dicha solución. Si la matriz A es invertible, x es la única solución del sistema. Si la matriz A es no invertible, MATLAB da un mensaje de aviso, en este caso el sistema podría tener infinitas soluciones o ser incompatible.
- m>n (sistema sobredeterminado): entonces A \ b devuelve la solución óptima por mínimos cuadrados del sistema A*x= b. Esta solución por mínimos cuadrados es la que produce cuando el sistema de ecuaciones es incompatible.
- m<n (sistema subdeterminado): entonces A \ b halla una solución con más m componentes nulas. Nótese que en este caso el sistema tiene infinitas soluciones.

Aunque Matlab resuelve cualquier tipo se sistema de ecuaciones lineales, no indica si el sistema es compatible (determinado o indeterminado) o incompatible.

La función "\" (o mldivide) emplea diferentes algoritmos según sean los coeficientes de la matriz A. El algoritmo implementado por MATLAB para la función "\" está basado en encontrar la factorización de la matriz A más adecuada según su estructura, por ejemplo:

- Si A cuadrada aplicará la factorización LU basada en el método de Gauss (con pivotación parcial).
- Si A cuadrada definida positiva aplicará la factorización de Cholesky (véase la función de MATLAB **chol).**
- Si A rectangular se aplicará la descomposición QR.
- Si el sistema es incompatible o sobredeterminado (tiene más ecuaciones que incógnitas), la función "\" busca la solución que minimiza Ax − b = 0 en el sentido de los mínimos cuadrados, realizando previamente una factorización QR de A.

Cuando la matriz A es invertible, podríamos resolver el sistema como x = A^{−1}b, aunque este caso no es recomendable, salvo para sistemas con pocas ecuaciones, ya que el número de operaciones para el cálculo de la inversa es mucho mayor que el que realiza con el comando "\".

Id a los siguientes enlaces en la documentación del MATLAB para ver más detalles sobre cómo MATLAB resuelve los sistemas lineales de ecuaciones, sobre las funciones específicas de MATLAB para la resolución y, por último, también pueden verse los algoritmos utilizados por MATLAB.

En el enlace pueden encontrarse ejemplos de sistemas de ecuaciones con matriz de coeficientes cuadrada (no singulares y singulares), de sistemas de ecuaciones sobredeterminados, indeterminados... También se muestran las funciones de MATLAB para aplicar métodos iterativos. Aunque en esta unidad nos centraremos en ejemplos que se resuelvan por métodos directos.

Aunque menos habitual, un sistema lineal de ecuaciones puede presentarse de la forma X*A=b, en este caso MATLAB puede resolver el sistema con el comando "/" (función **mrdivide**)

```
>> X=B/A
```

Condicionamiento de la matriz

Otro problema relacionado con la resolución de los sistemas de ecuaciones lineales es el llamado condicionamiento del sistema o de la matriz. Existen sistemas en los que un pequeño cambio en los coeficientes produce una variación grande en la solución, dichos sistemas se dice que están mal condicionados. Estos sistemas son, por tanto, muy sensibles a los errores de redondeo y su solución es poco fiable.

Existen métodos para medir de forma numérica el condicionamiento de una matriz basados en diferentes tipos de normas de vectores. MATLAB dispone de dos funciones para medir dicho condicionamiento **cond** y **rcond** (seguir los enlaces para ver la sintaxis de estas funciones).

Matrices dispersas

Los sistemas lineales que suelen aparecer en aplicaciones científicas, suelen tener una matriz de coeficientes muy grande pero con un gran número de elementos nulos, son las llamadas matrices dispersas. Dichas matrices se almacenan de forma diferente ya que al poseer un número relativamente pequeño de elementos significativos no necesitan guardarse en la memoria del ordenador como una matriz normal.

MATLAB trata de forma específica dicho tipo de matrices. La función **sparse** declara una matriz como dispersa y almacena y opera solamente con los elementos no nulos. Véase el enlace.

Ejemplos

En el libro de Cleve Moller o en otros libros de métodos númericos y MATLAB (véase la bibliografía) podéis encontrar multitud de ejemplos y problemas de sistemas de ecuaciones lineales resueltos con MATLAB. Incluso en el de Moller vienen las soluciones y los scripts .m de MATLAB para la resolución.

Ejemplo (del Moller): Alicia compra tres manzanas, una docena de plátanos y un melón por 2,36 euros. Bob compra una docena de manzanas y dos melones por 5,26 euros y Carol compra dos plátanos y tres melones por 2,77 euros. ¿Cuánto vale cada pieza de fruta?

La respuesta corresponde a resolver el sistema de 3 ecuaciones lineales con 3 incógnitas, A *x=b. En MATLAB se escribiría:

```
>> A = [3 12 1; 12 0 2; 0 2 3]

>> b = [2.36; 5.26; 2.77]
>> x = A\b
>> x =
        0.29
```

0.05
0.89

Ejemplo: Resolver el sistema lineal de ecuaciones A*x = b, donde:

```
>> A = magic(3);

>> b = [15; 15; 15];

>> x = A\b

>> x =

      1.0000

      1.0000

      1.0000
```

Ejemplo: Resolver el sistema lineal de ecuaciones A*x = b. Hallar la descomposición LU de la matriz A y resolver el sistema de ecuaciones usando dicha descomposición:

```
>> A = [4    2    0   -1    3;  -1    7    2    1    4; 1    3    5   -1    2; 2    0   -1    2    3; 1    3   -1    2    4]
>> b=[2 3 4 -2 -1]'
```

Para hallar la descomposición LU de la matriz A y para resolver el sistema usando A y usando la descomposición LU, pondríamos en la línea de comendos de MATLAB:

```
>> [L,U,P]=lu(A);
>> x1=A\b;
>> x2=U\(L\b);
```

Resultando:

```
L =

   1.0000         0         0         0         0
  -0.2500    1.0000         0         0         0
   0.2500    0.3333    1.0000         0         0
   0.5000   -0.1333   -0.1692    1.0000         0
```

```
    0.2500    0.3333   -0.3846    0.6646    1.0000


U =

    4.0000    2.0000         0   -1.0000    3.0000
         0    7.5000    2.0000    0.7500    4.7500
         0         0    4.3333   -1.0000   -0.3333
         0         0         0    2.4308    2.0769
         0         0         0         0    0.1582


P =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1

x1 =

    1.6000
    1.6000
    0.6000
    1.0000
   -2.2000

x2 =

    1.6000
    1.6000
    0.6000
    1.0000
   -2.2000
```

Observar que las dos soluciones son idénticas, aunque el número de operaciones en la segunda es menor al haber resuelto dos sistemas triangulares. En este ejemplo la matriz L es triangular con unos en la diagonal y no hemos tenido que hacer ninguna permutación de filas, eso se refleja también en que la matriz P es la identidad.

Actividades propuestas

Instructions: Realizar las actividades y no pasar a las siguientes unidades hasta no haber sido capaz de completarlas con éxito.

Actividades

¿Qué tengo que hacer?

IMPORTANTE: No pases a las siguientes unidades hasta no haber sido capaz de completar con éxito estas actividades. Estos ejercicios NO hay que entregarlos.

Las actividades propuestas para esta unidad consisten en el planteamiento en MATLAB de algunos ejercicios, comparar la resolución usando "\" con la matriz de coeficientes A y la resolución mediante la descomposición LU.

Para los siguientes problemas: plantear el sistema de ecuaciones, escribir en MATLAB las matrices A y b, hallar la descomposición LU de la matriz, ver también si es necesario realizar alguna permutación entre las filas, resolver el sistema a partir de la matriz A, resolver el sistema usando la descomposición LU, hallar los valores del condicionamiento de la matriz A. IMPORTANTE: Tened en cuenta si se ha efectuado permutación de filas.

Actividad 1:

Resolver el problema 2.3 del libro de C. Moller (página 31). Donde el sistema de ecuaciones que hay que resolver es:

$$
\begin{aligned}
f_2 &= f_6, \\
f_3 &= 10; \\
\alpha f_1 &= f_4 + \alpha f_5, \\
\alpha f_1 + f_3 + \alpha f_5 &= 0; \\
f_4 &= f_8, \\
f_7 &= 0; \\
\alpha f_5 + f_6 &= \alpha f_9 + f_{10}, \\
\alpha f_5 + f_7 + \alpha f_9 &= 15; \\
f_{10} &= f_{13}, \\
f_{11} &= 20; \\
f_8 + \alpha f_9 &= \alpha f_{12}, \\
\alpha f_9 + f_{11} + \alpha f_{12} &= 0; \\
f_{13} + \alpha f_{12} &= 0.
\end{aligned}
$$

siendo las f_i las incógnitas y alpha = 1/ \sqrt{2}.

Actividad 2: Resolver el siguiente sistema de ecuaciones lineales. Comparar la resolución usando "\" con la matriz de coeficientes A y la resolución mediante la descomposición LU. Calcular los números de condicionamiento de la matriz. (Fuente: J.R. Torregrosa, J.L. Hueso, A. Cordero, E. Martínez, Métodos numéricos con Matlab).

3.0210 x_1 + 2.7140 x_2 + 6.9130 x_3 =12.6480

1.0310 x_1 - 4.2730 x_2 + 1.1210 x_3 =-2.1210

5.0840 x_1 - 5.8302 x_2 + 9.1550 x_3 = 8.4070

Bibliografía

- Cleve B. Moler, Numerical Computing with MATLAB.
- J.R. Torregrosa, J.L. Hueso, A. Cordero, E. Martínez, Métodos numéricos con Matlab.
- Documentación en el "Centro de Documentación" sobre sistemas de ecuaciones lineales y descomposición matricial.