Computación de Altas Prestaciones

-Resolución de sistemas de Ecuaciones Lineales.

Contenidos

-Introducción

-Resolución de sistemas triangulares

-Eliminación gaussiana

-Descomposición LU

INTRODUCCIÓN

Pretendemos resolver un sistema de n ecuaciones lineales con incógnitas:

$$
\begin{aligned}
a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1\\
a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2\\
&\vdots\\
a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n &= b_n
\end{aligned}
$$

En forma matricial lo expresamos como:

$$Ax=b$$

donde $A \in \Re^{nxn}$, $x \in \Re^{n}$, $b \in \Re^{n}$

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

INTRODUCCIÓN

1) Sólo consideramos sistemas con el mismo número de incógnitas y de ecuaciones.

2) El sistema tendrá solución única si y sólo sí la matriz A tiene inversa ($x=A^{-1}b$) (Es un método posible pero ineficiente)

3) A tiene inversa ↔ determinante(A) <> 0 ↔

Rango(A)=n

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Sistemas triangulares

Las técnicas directas de resolución de sistemas de ecuaciones lineales de la forma

$$Ax = b \quad , \text{ donde } A \in \Re^{nxn} \qquad ; \qquad b \in \Re^{n}$$

se basan en reducir el problema a la resolución de dos sistemas triangulares:

- Uno triangular superior.(Alg. eliminación progresiva).
- Otro triangular inferior. (Alg. eliminación regresiva).

Método de Eliminación Progresiva

$$\begin{pmatrix} 3 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 2 & 5 & 0 \\ -1 & 2 & 1 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9 \\ 10 \\ 0 \\ 14 \end{pmatrix}$$

La ecuación 1 es $3x_1 = 9 \Rightarrow x_1 = 3$

La ecuación 2 es $2x_1 + 4x_2 = 10$
x_1 ya está calculado,

$$2 \cdot 3 + 4x_2 = 10 \;\Rightarrow x_2 = 1$$

En la ecuación 3, ya tenemos calculados x_1 y x_2 …. etc.

Resolución de un sistema triangular inferior

Supongamos que tenemos que resolver el sistema $Ly = b$ con $L \in \Re^{nxn}$ triangular inferior:

$$L = (l_{ij}) \quad ; \quad l_{ij} = 0 \quad \text{si } i<j \quad y \quad b$$

Para $n=3$ tenemos:

$$\begin{pmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Se resuelve mediante el método de:

Eliminación Progresiva

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Método de Eliminación Progresiva

- De la ecuación 1ª $l_{11}y_1 = b_1$ $y_1 = \frac{b_1}{l_{11}}$ si $l_{11} \neq 0$
- De la ecuación 2ª $l_{21}y_1 + l_{22}y_2 = b_2$ $y_2 = \frac{b_2 - l_{21}y_1}{l_{22}}$ si $l_{22} \neq 0$
- De la ecuación 3ª $l_{31}y_1 + l_{32}y_2 + l_{33}y_3 = b_3$ $y_3 = \frac{b_3 - l_{31}y_1 - l_{32}y_2}{l_{33}}$ siendo $l_{33} \neq 0$

Se debe cumplir $l_{ii} \neq 0$ para todo i → el determinante de $L = l_{11} \cdot l_{22} \cdot l_{33}$ debe de ser distinto de cero, → L debe ser invertible.

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Algoritmo 1. Eliminación Progresiva (por filas)

Dada una matriz $L \in \Re^{nxn}$ triangular inferior e invertible y un vector $b \in \Re^{n}$ este algoritmo calcula un vector $y \in \Re^{n}$ tal que *Ly = b.*

```
For  i = 1: n
        y_i = b_i
        For j = 1: i-1
                y_i = y_i - l_ij y_j
        End for

        y_i = y_i / l_ii
End For
```

Coste : $\sum_{i=1}^{n} 2(i-1)+1 = \sum_{i=1}^{n} 2i-1 = 1+3+\ldots+2n-1 = \frac{1+2n-1}{2} n = n^2$

Algoritmo 1.1. Eliminación Progresiva (con overwriting); versión por filas

```
b(1)=b(1)/L(1,1)
For  i = 2: n
        b(i)=(b(i)-L(i,1:i-1)b(1:i-1))/L(i,i)
End For
```

Producto escalar de L(i,1:i-1) y b(1:i-1)

Algoritmo 2. Eliminación Regresiva (por filas)

Dada una matriz $U \in \Re^{nxn}$ triangular superior e invertible y un vector $b \in \Re^n$ este algoritmo calcula un vector $y \in \Re^n$ tal que ***Uy = b***.

For i = n: 1:-1

$y_i = b_i$

For j = i+1:n

$y_i = y_i - u_{ij}\, y_j$

End for

$y_i = \dfrac{y_i}{u_{ii}}$

End For

Algoritmo 2.1. Eliminación Regresiva (con overwriting); versión por filas

b(n)=b(n)/U(n,n)
For i = n-1: 1:-1
b(i)=(b(i)-U(i,i+1:n)b(i+1:n))/U(i,i)
End For

Producto escalar de U(i,i+1:n) y b(i+1:n)

Algoritmo 1.3. Eliminación Progresiva (con overwriting); versión por columnas

Sistema triangular superior

$$\begin{pmatrix} 3 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 2 & 5 & 0 \\ -1 & 2 & 1 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9 \\ 10 \\ 0 \\ 14 \end{pmatrix} \longrightarrow$$

Obtenemos $x_1=3$ de la primera ecuación, nos quedan la 2,3 y 4:

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Algoritmo 1.3. Eliminación Progresiva (con overwriting); versión por columnas

Sistema triangular superior

$$\begin{pmatrix} 3 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 2 & 5 & 0 \\ -1 & 2 & 1 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9 \\ 10 \\ 0 \\ 14 \end{pmatrix} \longrightarrow$$

Obtenemos x_1=3 de la primera ecuación, nos quedan la 2,3 y 4:

$$\begin{pmatrix} 4 & 0 & 0 \\ 2 & 5 & 0 \\ 2 & 1 & 7 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \\ 14 \end{pmatrix} + ? \longrightarrow$$

La primera columna (menos el primer elemento) L(2:4,1) pasa al lado derecho multiplicada por $-x_1$

Algoritmo 1.3. Eliminación Progresiva (con overwriting); versión por columnas

Sistema triangular superior

$$\begin{pmatrix} 3 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 2 & 5 & 0 \\ -1 & 2 & 1 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 9 \\ 10 \\ 0 \\ 14 \end{pmatrix} \longrightarrow$$

Obtenemos x_1=3 de la primera ecuación, nos quedan la 2,3 y 4:

$$\begin{pmatrix} 4 & 0 & 0 \\ 2 & 5 & 0 \\ 2 & 1 & 7 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 0 \\ 14 \end{pmatrix} - \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} x_1 \longrightarrow$$

La primera columna (menos el primer elemento) L(2:4,1) pasa al lado derecho multiplicada por -x_1

Algoritmo 1.3. Eliminación Progresiva (con overwriting); versión por columnas

Simplificando:

$$\begin{pmatrix} 4 & 0 & 0 \\ 2 & 5 & 0 \\ 2 & 1 & 7 \end{pmatrix}\begin{pmatrix} x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ -3 \\ 11 \end{pmatrix} \longrightarrow$$

Repetimos el proceso; obtenemos $x_2=1$ y la columna L(2:3,2) pasa al lado derecho multiplicada por $-x_2$

$$\begin{pmatrix} 5 & 0 \\ 1 & 7 \end{pmatrix}\begin{pmatrix} x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -3 \\ 11 \end{pmatrix} - x_2\begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

Simplificamos, y repetimos hasta el final…

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Algoritmo 2.3. Eliminación Progresiva (con overwriting); versión por columnas

El algoritmo procede eliminando las columnas del sistema, desde la 1 hacia delante, y obteniendo al mismo tiempo las incógnitas.

```
For j=1:n-1
  b(j)=b(j)/L(j,j)
  for i=j+1:n
    b(i)=b(i)-L(i,j)*b(j)
  end
end
b(n)=b(n)/L(n,n)
```

Acceso a L por columnas

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Algoritmo 2.3. Eliminación Progresiva (con overwriting); versión por columnas

El bucle interno es un saxpy

```
For j=1:n-1
  b(j)=b(j)/L(j,j)
  b(j+1:n)=b(j+1:n)-L(j+1:n,j)*b(j)
end
b(n)=b(n)/L(n,n)
```

saxpy

Acceso a L por columnas

Algoritmo 1.3. Eliminación Regresiva (con overwriting); versión por columnas

Del mismo modo se puede modificar el algoritmo de eliminación regresiva:

```
For j=n:-1:2
   b(j)=b(j)/U(j,j)
   b(1:j-1)=b(1:j-1)-U(1:j-1,j)*b(j)
End
b(1)=b(1)/U(1,1)
```

saxpy

Sistema triangular inferior, con múltiples lados derechos

Resolver LX=B, donde $L \in \Re^{n*n}$ triangular inferior y $X,B \in \Re^{n*q}$

Los algoritmos anteriores se pueden adaptar fácilmente:

b(1,1:q)=b(1,1:q)/L(1,1)
For i = 2: n
b(i,1:q)=(b(i,1:q)-L(i,1:i-1)b(1:i-1,1:q))/L(i,i)
End For

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

-Eliminación Gaussiana

Eliminación Gaussiana

$$\begin{aligned} x + 2y - z &= 3 \\ 2x - y - z &= 4 \\ -x + 3y + 2z &= 5 \end{aligned} \Rightarrow \begin{aligned} x + 2y - z &= 3 \\ 0x - 5y + z &= -2 \\ 0x + 5y + z &= -2 \end{aligned} \Rightarrow \begin{aligned} x + 2y - z &= 3 \\ 0x - 5y + z &= -2 \\ 0x + 0y + 2z &= -4 \end{aligned}$$

Eliminación Gaussiana

Dado un sistema Ax=b, lo transforma en otro Ux=b’, que tienen la misma solución y donde U es triangular superior.

-Podemos multiplicar una ecuación por un escalar, y la solución no cambia :

```
For j=1:N
  A(i,j)=A(i,j)*alpha
End
b(i)=b(i)*alpha
```

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Eliminación Gaussiana

"Operación principal con filas": Podemos sumarle a una ecuación otra diferente multiplicada por otro escalar, y la solución no cambia:

For j=1:N
  A(i,j)=A(i,j)+A(k,j)*alpha
End
b(i)=b(i)+b(k)*alpha

Saxpy por filas

"A la fila i, le sumo la fila k multiplicada por alpha"

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Eliminación Gaussiana

Descripción del algoritmo

D1:Utilizando la operación principal, hacer ceros por debajo de la diagonal principal, hasta que la matriz sea triangular superior.

La matriz tiene N columnas, y hay que hacer ceros en todas ellas salvo en la última:

D2:
For k=1,N-1
Hacer ceros en la columna k-ésima, por debajo de la diagonal.
End

Eliminación Gaussiana

$$
\begin{array}{c}
\begin{array}{cccc} k=1 & k=2 & k=3 & \end{array} \\
\left[\begin{array}{cccc}
x & x & x & \ldots\ldots\ldots\ldots\ldots\ldots x \\
x & x & x & \ldots\ldots\ldots\ldots\ldots\ldots x \\
x & x & x & \ldots\ldots\ldots\ldots\ldots\ \ x \\
. & & & \\
. & & & \\
. & & & \\
. & & & \\
x & x & x & \ldots\ldots\ldots\ldots\ldots\ldots x
\end{array}\right]
\end{array}
$$

Eliminación Gaussiana

Al hacer ceros en la columna k-ésima por debajo de la diagonal, hacemos ceros cada elemento de la columna k-ésima empezando por el de la fila k+1 y acabando por el de la N:

D3:

```
For k=1:N-1
  For i=k+1:N
      Mediante operaciones de filas, hacer cero el
      elemento A(i,k)
  End
End
```

Eliminación Gaussiana

-Se utiliza la fila k-ésima para hacer los ceros;
-Para hacer cero el elemento A(i,k), tenemos que sumarle la fila k-ésima multiplicada por el valor apropiado: (-A(i,k)/A(k,k))

D4:

```
For k=1:N-1
  For i=k+1:N
      Mediante operaciones de filas, hacer cero el
      elemento A(i,k)
  End
End
```

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Eliminación Gaussiana

-Se utiliza la fila k-ésima para hacer los ceros;
-Para hacer cero el elemento A(i,k), tenemos que sumarle la fila k-ésima multiplicada por el valor apropiado: (-A(i,k)/A(k,k))

D4:

```
For k=1:N-1
  For i=k+1:N
    For  j=k:N
      A(i,j)=A(i,j)+A(k,j)*(-A(i,k)/A(k,k))
    End
    b(i)=b(i)+b(k)*(-A(i,k)/A(k,k))
  End
End
```

Eliminación Gaussiana; Pivotación

-La pivotación es necesaria para que el método LU sea estable;

-Utilizar como pivote el mayor elemento, en valor absoluto, de la columna: **pivotación parcial.**

Eliminación Gaussiana

Algoritmo básico sin pivotación:

```
For k=1:n-1
 For i=k+1:n
        If A(k,k)==0 then STOP
       For j=k+1:n
            A(i,j)=A(i,j)-A(k,j)*(A(i,k)/A(k,k))
        End
        b(i)=b(i)-b(k)*(A(i,k)/A(k,k))
        A(i,k)=0
    End
End
```

Eliminación Gaussiana → LU

Entrada: Matriz A cuadrada
Salida matrices L y U, L triangular inferior unidad, U triangular superior, tales que L*U=A

```
U=A
For k=1:n-1
   If U(k,k)==0 then STOP
      For i=k+1:n
         L(i,k)=U(i,k)/U(k,k)
         For j=k+1:n
            U(i,j)=U(i,j)-U(k,j)*L(i,k)
         End
    End
    L(k,k)=1
End
```

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Eliminación Gaussiana ➔ LU

Observaciones:

1)La parte triangular superior tras LU o tras El.Gaus. son idénticas

2) La L se construye a partir de los “multiplicadores” A(i,k)/A(k,k)

3) El lado derecho **b** no se utiliza para la LU.

4) Se puede considerar que el bucle **k** se mueve por la diagonal, el bucle **i** se mueve por las columnas, y el bucle **j** se mueve por las filas

5) A los elementos de la diagonal se les llama **pivotes**.

6) No es necesario almacenar la diagonal de L (es todo unos)➔ es posible almacenar las dos matrices, L y U , en una sóla matriz densa ➔ Es posible usar overwriting ➔

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Eliminación Gaussiana → LU

Entrada: Matriz A cuadrada
Salida matriz sobreescrita con L y U, L triangular inferior unidad, U triangular superior, tales que L*U=A

```
For k=1:n-1
  If A(k,k)==0 then STOP
    For i=k+1:n
        A(i,k)=A(i,k)/A(k,k)
        For j=k+1:n
            A(i,j)=A(i,j)-A(k,j)*A(i,k)
        End
    End
End
```

Eliminación Gaussiana → LU

Para una matriz 4*4, tras dos etapas (tras k=2), la matriz quedaría:

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & u_{22} & u_{23} & u_{24} \\ l_{31} & l_{32} & a_{33}^{(2)} & a_{34}^{(2)} \\ l_{41} & l_{42} & a_{43}^{(2)} & a_{44}^{(2)} \end{pmatrix}$$

Ya procesado

Por procesar

Versiones de la desc. LU

Es posible sacar las divisiones por el elemento de la diagonal del bucle:

```
For k=1:n-1
  If A(k,k)==0 then STOP
   For i=k+1:n
      A(i,k)=A(i,k)/A(k,k)
   End
   For i=k+1:n
         For j=k+1:n
             A(i,j)=A(i,j)-A(k,j)*A(i,k)
         End
    End
End
```

Update de rango 1 por filas

Versiones de la desc. LU

Podemos cambiar el update de filas a columnas:

```
For k=1:n-1
  If A(k,k)==0 then STOP
   For i=k+1:n
      A(i,k)=A(i,k)/A(k,k)
   End
   For j=k+1:n
         For i=k+1:n
             A(i,j)=A(i,j)-A(k,j)*A(i,k)
         End
    End
End
```

Version kji

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Descomposición LU

De modo similar al producto de matrices, existen 6 versiones: kij, kji, ijk, ikj, jki, jik (ver documento en poliformat “versiones ijk, ikj, etc. De lu”)

-También hay versiones a bloques (las mas eficientes)

-Cada versión tiene, al final, el mismo coste en flops, pero diferentes propiedades de acceso a memoria.

-Hay que tener en cuenta que existe sólo una descomposición LU con L triangular unidad, pero, sin esta restricción, existen infinitas descomposiciones.

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN