Universidad de Puerto Rico
Departamento de Matemáticas
Humacao, Puerto Rico 00791

MATE 4061 Análisis Numérico Prof. Pablo Negrón

Laboratorio 4: Eliminación Gaussiana sin Pivoteo

En este laboratorio vamos a discutir los aspectos prácticos, i.e., implementativos del método de eliminación Gaussiana y sus variantes. Aunque MATLAB cuenta con unas rutinas altamente eficientes para hacer esto, el propósito de estudiar las implementaciones es ver detalles implementativos, hacer modificaciones a los programas para estudiar los efectos de pivoteo y la aritmética de punto flotante, y para hacer conteos operacionales.

En clase vimos que la factorización LU de la matriz A, suponiendo que esta es nosingular, se puede calcular mediante eliminación Gaussiana como sigue: para $k = 1, \ldots, n-1$ calcule,

$$\begin{cases} m_{ik} = \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \quad , \quad i = k+1, \ldots, n \\ \\ a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)} \quad , \quad i.j = k+1, \ldots, n. \end{cases} \tag{1}$$

El primer grupo de ecuaciones produce los multiplicadores para eliminar la columna k y el segundo grupo modifica las filas $k+1, \ldots, n$ pero solo las columnas $k+1, \ldots, n$. Un primer programa en MATLAB que implementa estas ecuaciones es:

```
for k=1:n-1
   for i=k+1:n
      m(i,k)=A(i,k)/A(k,k);
      for j=k+1:n
         A(i,j)=A(i,j)-m(i,k)*A(k,j);
      end
   end
end
```

Note que la matriz A se re-escribe con la nueva matriz. Este programa se puede mejorar de dos formas: primero los multiplicadores m_{ik} no necesitan un arreglo aparte para guardarse. Se pueden almacenar en la parte inferior de A la cual es cero de todas formas; segundo, podemos usar las propiedades *vectoriales* de MATLAB para escribir varios ciclos en forma implícita resultando así en un código mas eficiente. Veamos como queda el programa con estas ideas:

```
for k=1:n-1
   A(k+1:n,k)=A(k+1:n,k)/A(k,k);
   A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-A(k+1:n,k)*A(k,k+1:n);
end
```

Note que `A(k+1:n,k)` es un vector de tamaño $(n-k)\times 1$ y `A(k,k+1:n)` es de temaño $1\times(n-k)$. De modo que el termino `A(k+1:n,k)*A(k,k+1:n)` en el ciclo produce una matriz $(n-k)\times(n-k)$ que al restarse a `A(k+1:n,k+1:n)` produce el resultado deseado (¿Porqué?). Finalmente usando las instrucciones `tril` y `triu` de MATLAB para extraer matrices inferiores y superiores respectivamente de una matriz dada, podemos montar el ciclo de arriba en una función que realiza el proceso de descomposicón de A:

```
function [L,U]=eg(A)
[n n]=size(A);
for k=1:n-1
   A(k+1:n,k)=A(k+1:n,k)/A(k,k);
```

```
   A(k+1:n,k+1:n)=A(k+1:n,k+1:n)-A(k+1:n,k)*A(k,k+1:n);
end
L=eye(n,n)+tril(A,-1);
U=triu(A);
```

Ejemplo 1. Un ejemplo de como usar esta función es en el calculo de la factorización LU de

$$A = \begin{pmatrix} 1 & 2 & 1 \\ -1 & 4 & 2 \\ 3 & 1 & 1 \end{pmatrix}. \tag{2}$$

Escribimos en MATLAB:

```
A=[1 2 1;-1 4 2;3 1 1];
[L,U]=eg(A)
```

lo cual produce el resultado:

```
L =

    1.0000         0         0
   -1.0000    1.0000         0
    3.0000   -0.8333    1.0000


U =

    1.0000    2.0000    1.0000
         0    6.0000    3.0000
         0         0    0.5000
```

Eso se puede verificar calculando `L*U` el cual debe dar a `A`.

Ejercicio 1. *Usando la instrucción* `flops` *de MATLAB, calcule el numero de operaciones en el calculo de la factorización LU para matrices de tamaños* $n = 5, \ldots, 30$ *usando la función* `eg`*. Haga una gráfica de los resultados. Use la instrucción* `polyfit` *de MATLAB para hallar el polinomio cúbico que mejor aproxima a los datos. ¿Cúal es el coeficiente de* n^3 *en este polinomio? Explique.* ***Nota:*** *Debido a un error en la función* `flops`*, debe reemplazar la instrucción que calcula* `A(k+1:n,k+1:n)` *en* `eg` *por los dos ciclos* `for` *equivalentes.*

Pasamos ahora a las implementaciones de la modificación del lado derecho y la sustitución para atrás. El lado derecho se modifica de acuerdo a las ecuaciones:

$$b_j^{(k+1)} = b_j^{(k)} - m_{jk} b_k^{(k)} \quad , \quad k = 1, \ldots, n-1 \ , \ j = k+1, \ldots, n. \tag{3}$$

Suponiendo que se ha hecho una llamada a `eg` y se tienen almacenadas las matrices `L` y `U`, podemos implementar estas ecuaciones como sigue recordando que `L` contiene los multiplicadores:

```
function g=modld(L,b)
n=length(b);
g=b;
for k=1:n-1
   g(k+1:n)=g(k+1:n)-g(k)*L(k+1:n,k);
end
```

Las ecuaciones de la sustitución para atrás están dadas por:

$$x_n = \frac{g_n}{u_{nn}} \quad , \quad x_i = \frac{g_i - \sum_{j=i+1}^{n} u_{ij} x_j}{u_{ii}} \quad , \quad i = n-1, \ldots, 1, \tag{4}$$

donde g es el resultado de la modificación del lado derecho y U es la matriz triangular superior en la descomposición de A. Esto lo programamos como sigue:

```
function x=sustpa(U,g)
n=length(g);
x=g;
x(n)=g(n)/U(n,n);
for k=n-1:-1:1
   x(k)=(g(k)-U(k,k+1:n)*x(k+1:n))/U(k,k);
end
```

Las tres funciones: `eg`, `modld`, y `sustpa`, se pueden combinar ahora en el siguiente programa para resolver un sistema lineal $Ax = b$:

```
function x=linsol(A,b)
[L,U]=eg(A);
g=modld(L,b);
x=sustpa(U,g);
```

Ejemplo 2. Suponga queremos resolver el sistema $Ax = b$ donde A es la matriz del Ejemplo (1) y $b = (1, -1, 0)^t$. Usando la función `linsol` esto lo hacemos mediante:

```
A=[1 2 1;-1 4 2;3 1 1];
b=[1,-1,0]';
x=linsol(A,b)
```

lo cual produce el resultado:

```
x =

     1
     3
    -6
```

Calculo de la Inversa de una Matriz

Si A es una matriz nosingular $n \times n$, su inversa se puede calcular resolviendo los n sistemas lineales:

$$Ax_i = e_i \quad , \quad 1 \leq i \leq n,$$

donde e_i es el vector con todos los componentes cero excepto en la posición i que tiene un uno. La matriz inversa es ahora:

$$A^{-1} = (x_1, x_2, \ldots, x_n).$$

Usando las funciones `eg`, `modld`, y `sustpa` podemos escribir un programa para el calculo de la inversa:

```
function a=inveg(A)
[n,n]=size(A);
a=zeros(n,n);
[L,U]=eg(A);
E=eye(n,n);
for k=1:n
   g=modld(L,E(:,k));
   a(:,k)=sustpa(U,g);
end
```

Ejercicio 2. *Usando la instrucción `flops` de MATLAB, calcule los numeros de operaciones necesarias para resolver el $Ax = b$ usando eliminación Gaussiana y el método de la inversa para matrices de tamaños $n = 5, \ldots, 30$. Usando `polyfit` aproxime ambos conjuntos de datos con polinomios cúbicos. Compare los coeficientes de los términos de n^3. Haga una gráfica de los resultados.*

Calculo de Determinantes

La descomposición LU de una matriz A nos da un método eficiente para el calculo del determinante de A. Esto es

$$\det(A) = \det(LU) = \det(L)\det(U) = \det(U),$$

donde usamos que como L tienes unos en la diagonal, entonces $\det(L) = 1$. Ahora, como U es triangular superior, su determinante se calcula multiplicando la diagonal principal. Tenemos ahora el siguiente programa para calcular el determinante:

```
function y=deteg(A)
[L,U]=eg(A);
y=prod(diag(U));
```

El método teórico para calcular el determinante es usando *cofactores*. Este método tiene un conteo operacional de operaciones de punto flotante proporcional a $n!$ de modo que no debe usarse como un procedimiento practico para calcular determinantes. A modo de comparar los conteos operacionales, incluimos un programa para calcular el determinante de A usando el método de cofactores:

```
function y=detcof(A)
n=size(A,1);
if n==1
   y=A(1,1);
elseif n==2
   y=A(1,1)*A(2,2)-A(1,2)*A(2,1);
else
   isign=1;
   y=0;
   for i=1:n
      y=y+isign*A(1,i)*detcof(A(2:n,[1:i-1 i+1:n]));
      isign=-isign;
   end
end
```

Vamos a comparar el numero de operaciones de punto flotante en ambos métodos para matrices aleatorias de tamaños $n = 2, \ldots, 8$:

```
tam=2:8;
teg=zeros(size(tam));
tcof=teg;
i=1;
for n=tam
   A=rand(n,n);
   flops(0);
   y=deteg(A);
   teg(i)=flops;
   flops(0);
   y=detcof(A);
   tcof(i)=flops;
   i=i+1;
end
semilogy(tam,teg,'k',tam,tcof,'k-.')
legend('Eliminacion Gaussiana','Metodo de cofactores')
```

Los resultados aparecen en la Figura (1) los cuales muestran claramente la superioridad del método que usa eliminación Gaussiana.

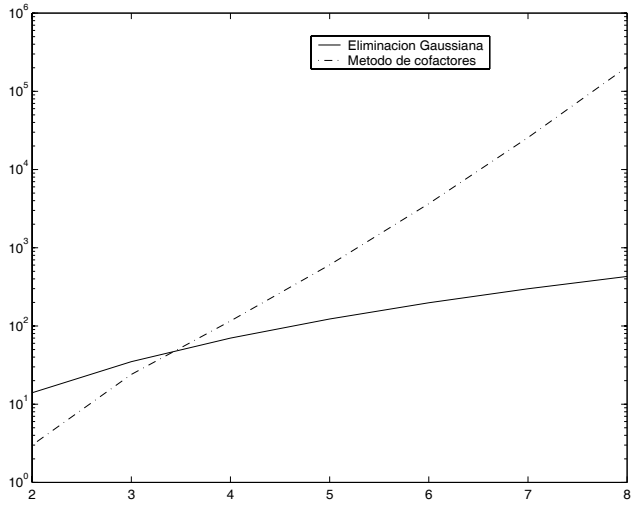

Figure 1: Totales de operaciones de punto flotante de los métodos de Eliminación Gaussiana y Cofactores para el calculo de determinantes.