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

MATE 4061 Análisis Numérico Prof. Pablo Negrón

Laboratorio 5: Eliminación Gaussiana con Pivoteo

En la discusión anterior del proceso de eliminación Gaussiana se asumió que los pivotes `A(k,k)` que se usan en el computo de los multiplicadores al dividir, son distintos de cero. Aun cuando estos multiplicadores no sean cero, si son muy pequeños, los pivotes que se calcules con estos, pueden actuar como factores de magnificación en la propagación de errores cuando los cálculos se hacen con precision finita. Para evitar este problema se usan las técnicas de *pivoteo* las cuales producen siempre multiplicadores menor de uno en valor absoluto. El método de pivoteo más común y fácil de implementar es el de *pivoteo parcial*. En este proceso, previo a la eliminación de la columna k en el método de eliminación Gaussiana, se selecciona el índice del mayor en valor absoluto de los elementos `A(k:n,k)` de la matriz. Si este índice es diferente a k, se intercambia la fila correspondiente a dicho índice con la fila k. La función `eg` modificada queda ahora como sigue:

```
function [L,U,piv]=egpiv(A)
[n n]=size(A);
piv=1:n;
for k=1:n-1
   [vmax,r]=max(abs(A(k:n,k)));
   q=r+k-1;
   if q~=k
      piv([k,q])=piv([q,k]);
      A([k,q],:)=A([q,k],:);
   end
   if A(k,k)~=0
      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
end
L=eye(n,n)+tril(A,-1);
U=triu(A);
```

Las matrices L, U que resultan de este algoritmo satisfacen que $PA = LU$ donde P se obtiene a partir de la matriz identidad pero intercambiando sus filas de acuerdo al vector `piv`. El resultado es que ahora el sistema $Ax = b$ es equivalente a $LUx = Pb$ y procedemos como antes pero con Pb como lado derecho en lugar de b. Tenemos pues que la subrutinas `modld` y `sustpa` no tienen que modificarse. La rutina `linsol` queda ahora como sigue:

```
function x=linsol_piv(A,b)
[L,U,piv]=egpiv(A);
g=modld(L,b(piv));
x=sustpa(U,g);
```

Ejemplo 1. Considere el sistema

$$\begin{cases} 3x - 2y + z &= 1 \\ x - \frac{2}{3} + 2z &= 2 \\ -x + 2y - z &= 0 \end{cases}.$$

La solución exacta del sistema es $x = 1/2,\ y = 3/4,\ z = 1$. Entramos la matriz de coeficientes y el lado derecho en MATLAB como sigue:

```
A=[3 -2 1;1 -0.6666666666666667 2;-1 2 -1];
b=[1,2,0]';
```

Calculamos la solución con `linsol` y `linsol_piv`:

```
linsol(A,b)

ans =

     0
     0
     1

linsol_piv(A,b)

ans =

    0.5000
    0.7500
    1.0000
```

Tenemos que `linsol_piv` produce los resultados correctos al numero de cifras mostradas mientras que `linsol` produce una contestación con ninguna cifra correcta en los primeros dos componentes. Note que el error inicial al entrar la matriz de coeficientes al sistema es de aproximadamente 10^{-16}. El problema con `linsol` es que durante el computo de la factorización LU, surge un pivote bien pequeño el cual se utiliza para calcular los multiplicadores correspondientes teniendo como consecuencia la perdida de cifras significativas.

La subrutina `deteg` debe ser modificada igualmente para hacer uso ahora de `egpiv`. Como $PA = LU$, entonces $\det(A) = \det(P)\det(U)$ ya que P es su propia inversa. Por la estructura particular de P, su determinante se puede calcular de forma relativamente simple. Tenemos pues la nueva rutina para los determinantes:

```
function y=detegp(A)
n=size(A,1);
[L,U,piv]=egpiv(A);
index=1;
for k=1:n
   if piv(k)~=k
      index=-index;
      q=piv(k);
      piv([k,q])=piv([q,k]);
   end
end
y=index*prod(diag(U));
```

El ciclo `for` del programa es el que calcula $\det(P) = (-1)^{\nu}$ donde ν es el numero de cambios de filas inducidos por P.

Eliminación Gaussiana en el Sistema de Punto Flotante F(10,3,9,9)

Para poder estudiar mejor los efectos de la aritmética de punto flotante en el proceso de eliminación Gaussiana discutiremos ahora unas variantes de las subrutinas `egpiv`, `modld`, y `sustpa` en el sistema de punto flotante $F(10, 3, 9, 9)$. Las nuevas rutinas tienen los nombres: `egpf`, `modldpf`, y `sustpapf`. Estas rutinas utilizan las funciones `convertm` y `representm` las cuales son versiones matriciales de `convert` y `represent`. Además de estas ultimas dos rutinas debe tener en su directorio de trabajo las rutinas `float` y `pretty`. Es bien importante e instructivo examinar los programas `egpf`, `modldpf`, y `sustpapf`, para

ver como se usan las rutinas que implementan el sistema $F(10,3,9,9)$. Presentamos aquí la variante de `linsol_piv` en este sistema:

```
function x=linsolpf(A,b)
%
% Se representan a las matrices A, b en el sistema F(10,3,9,9).
%
A1=representm(A);
b1=representm(b);
%
% Eliminacion Gaussiana en el sistema de punto flotante.
%
[L1,U1,piv]=egpf(A1);
g1=modldpf(L1,b1(piv,:,:));
x1=sustpapf(U1,g1);
%
% Convierte el resultado al sistema regular del computador.
%
x=convertm(x1);
```

Veamos un ejemplo que ilustra el uso de estas rutinas.

Ejemplo 2. Trabajamos nuevamente el sistem del Ejemplo (1) pero en el sistema $F(10,3,9,9)$. Esto lo hacemos como sigue:

```
A=[3 -2 1;1 -0.6666666666666667 2;-1 2 -1];
b=[1,2,0]';
x=linsolpf(A,b)
```

lo cual produce la contestación:

```
x =

    0.5000
    0.7520
    1.0000
```

Este calculo aunque no es exacto es mucho mejor que el producido por `linsol` en el Ejemplo (1). ¡Esto aun cuando `linsol` usa toda la presición de la computadora mientras que `linsolpf` usa solo tres cifras decimales! La diferencia es que `linsolpf` usa pivoteo parcial mientras que `linsol` no usa pivoteo.

Método de Residuos o Refinamiento Iterativo

Suponga que $\hat{x}$ es una solución aproximada del sistema $Ax = b$, i.e, $A\hat{x} \approx b$. Definimos el *residual* por $r = b - A\hat{x}$. Si $\hat{x}$ fuese la solución exacta, entonces $r = 0$. Pero $\hat{x}$ no es la solución exacta. Defina ahora el *error* como la solución e del sistema $Ae = r$. Es fácil verificar ahora que $\hat{x} + e$ es la solución exacta del sistema $Ax = b$. Esto es asumiendo que e se puede calcular en forma exacta. Pero en general esto no ocurre y lo que calculamos es un $\hat{e}$ que es una aproximación de e. Pero aun así, $\hat{x} + \hat{e}$ debe ser una mejor aproximación (comparada a $\hat{x}$) a la solución exacta del sistema $Ax = b$. Este proceso se puede repetir ahora con $\hat{x} + \hat{e}$, etc., dando lugar al método de *refinamiento iterativo*. El algoritmo que describe este proceso es:

Algoritmo 1. Proceso de refinamiento iterativo:

1. Sean L y U los factores (aproximados) de A.
2. Sea $x^{(0)}$ la solución aproximada de $Ax = b$ obtenida mediante Eliminación Gaussiana.
3. Para $k = 0, 1, 2, \ldots,$

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

(b) Resuelva $Ae^{(k)} = r^{(k)}$.

(c) Ponga $x^{(k+1)} = x^{(k)} + e^{(k)}$.

En la practica, el ciclo que aparece en el algoritmo no se repite más de dos veces. Note que en el computo del paso (3a) puede haber perdida de cifras significativas ya que se restan cantidades similares. Este computo debe hacerse con mas precisión. Veamos una implementación en MATLAB de este algoritmo:

```
function x=refiterpf(A,b)
n=size(A,1);
%
% Calcula la solucion aproximada en el sistema
% de punto flotante F(10,3,9,9).
%
A1=representm(A);
b1=representm(b);
[L1,U1,piv]=egpf(A1);
g1=modldpf(L1,b1(piv,:,:));
x1=sustpapf(U1,g1);
for k=1:2
   %
   % Calcula el residual a la precision de la computadora
   % y luego lo redondea al sistema.
   %
   x=convertm(x1);
   r=b-A*x;
   r1=representm(r);
   %
   % Resuelve la ecuacion residual y corrige la solucion
   % aproximada.
   %
   g1=modldpf(L1,r1(piv,:,:));
   e1=sustpapf(U1,g1);
   for i=1:n
      x1(i,:)=float(x1(i,:),e1(i,:),'+');
   end
end
x=convertm(x1);
```