4. La Factorización QR

Dada una matriz cuadrada y nosingular A de orden n x n, entonces existe una matriz ortogonal Q y una matriz triangular superior R tal que

$$A=QR$$

esta es llamada la factorización QR de A.

Si la matriz A no es cuadrada y de orden m x n con m mayor que n entonces:

$$A = QR = \begin{pmatrix} R_1 \\ O \end{pmatrix}$$

donde R_1 es una matriz triangular superior de orden n x n y 0 es una matriz de ceros de orden (m-n) x n.
Si la matriz A es de orden m x n con m menor que n entonces

$$A = QR = \begin{pmatrix} R_1 & S \end{pmatrix}$$

donde S es un matriz de orden (n-m) por m.

Existen tres métodos de obtener la factorización QR

a) Transformaciones Householder
b) Rotaciones Givens
c) Proceso de ortogonalización de Gram-Schmidt

4.1 Transformaciones Householder

Una matriz de la forma

$$H = I - 2\frac{uu'}{u'u}$$

es llamada una matriz Householder, donde I es la matriz identidad y u es un vector no nulo.

Propiedades de la matriz H:

a) H es una matriz simétrica y ortogonal.
b) $||H\mathbf{x}||_2=||\mathbf{x}||_2$ para todo vector **x**. Es decir, la matriz Householder no cambia la longitud del vector.
c) $H^2= I$
d) Det(H)=-1.

La importancia de las matrices Householder es que ellas pueden ser usadas para crear ceros en un vector y por lo tanto pueden dar lugar a matrices triangulares. Consideremos el vector elemental $\mathbf{e}_1=(1,0,\ldots,0)$ '. Entonces para todo vector no nulo $\mathbf{x}\neq\mathbf{e}_1$ existe siempre una matriz Householder H tal que Hx es un múltiplo de e_1.

Basta considerar el vector $\mathbf{u}=\mathbf{x}+\text{sign}(x_1)||\mathbf{x}||\mathbf{e}_1$ y se puede ver que $H\mathbf{x}=-\text{sign}(x_1)||\mathbf{x}||\mathbf{e}_1$. Si x_1 es cero entonces se puede escoger los signos + o -. Para evitar overflow o underflow en el cálculo de $||\mathbf{x}||$ se recomienda re-escalar el vector y usar en su lugar $\mathbf{x}/\max\{|x_i|\}$.

Algoritmo para crear ceros un vector usando una matriz Householder

Dado un vector no nulo **x**, el siguiente algoritmo calcula un vector **u** y una constante σ tal que $Hx=(1-2\frac{uu'}{u'u})x=(\sigma,0,\ldots.,0)'$, u es guardado encima de x.

1) $m=\max\{|x_i|)$, $i=1,2\ldots.n$
2) $u_i=x_i/m$, $i=1,2\ldots.n$
3) $\sigma=\text{sign}(u_1)\sqrt{u_1^2+u_2^2+\ldots..+u_n^2}$
4) $u_1=u_1+\sigma$
5) $\sigma=-m*\sigma$

la siguiente función **housecero** en MATLAB ejecuta el algoritmo

```
function [u,sigma] = housecero(x)
%HOUSECERO Crea ceros en un  vector usando una matriz Householder.
%[u,sigma] = housecero(x)  produce un vector u
%que define una matriz Householder H, y una constante sigma
%tal que Hx = [sigma, 0, ...., 0]'.
%input  :  vector x
%output :  vector u, y constante sigma

   [m,n] = size(x);
         mm = max(abs(x));
         x = x/mm;
         s = sign(x(1));
         if s == 0
          s = 1;
         end;
         sigma = s * norm(x,2);
         u = x + sigma * eye(m,1);
         sigma = -mm * sigma;
```

Ejemplo: Obtener ceros en el vector x=(3,4,9)' usando la función **housecero** . Hallar el vector transformado y la matriz Householder

```
» x=[3;4;9]

x =

     3
     4
     9
```

```
» addpath c:\matlab\acuna
» [u,sigma]=housecero(x)

u =

    1.4773
    0.4444
    1.0000


sigma =

  -10.2956

» u1=2*u*u'/(u'*u)

u1 =

    1.2914    0.3885    0.8742
    0.3885    0.1169    0.2630
    0.8742    0.2630    0.5917

» % matriz Householder
» H=eye(3)-u1

H =

   -0.2914   -0.3885   -0.8742
   -0.3885    0.8831   -0.2630
   -0.8742   -0.2630    0.4083

» H*H

ans =

    1.0000         0    0.0000
         0    1.0000         0
    0.0000         0    1.0000

»
```

El vector transformado sera Hx=(-10.2956,0,0)’.

Ahora se mostrará el efecto de multiplicar una matriz Householder por un vector y por una matriz.
Sea **x** un vector de dimension n y H una matriz Householder entonces

H**x**=$(I-2\frac{uu'}{u'u})$x=**x**-βu(u'x) donde β=2/(u'u).

Algoritmo para obtener el producto de una matriz Householder por un vector cualquiera.

Dado el vector n dimensional u que define la matriz Householder H=1-2βuu', y un vector cualquiera x=(x_1,x_2,…x_n)'. Entonces el siguiente algoritmo calcula el producto H**x** superponiendo **x** con H**x.**

Paso 1: Calcular β=2/(u'u).

Paso 2. Calcular la suma s=$\sum_{i=1}^{n} u_i x_i$

Paso 3. Modificar β=βs

Paso 4. For i=1,…, n do

x_i=x_i-βu_i

Consideremos ahora una matriz A, entonces HA=A-βuu'A. Luego, la entrada (i,j) de HA es igual a a_{ij}-β$(\sum_{i=1}^{m} u_i a_{ij})$ u_i , cada columna puede ser calculada usando el algoritmo anterior. Similarmente, AH=A-βAuu', cada fila de AH puede ser calculada usando el algoritmo anterior.

Notar que no hay que calcular explicitamente la matriz H.

La siguiente función calcula el producto de una matriz Householder por una matriz A

```
function A = housemult(A,u)

%HOUSEMULT Postmultiplica una matriz por una matriz
%Householder H
%A = housemult(A,u) calcula AH, donde H es una matriz
%Householder generada por un vector u.
%La matrix resultante A contiene el producto AH.
%input  : Matriz A y vector u
%output : Matriz A

  [m1,n] = size(A);
       beta = 2/(u'*u);
       for i = 1 : m1
         s = 0;
           s =  s + u(1:n) * A(i,1:n);
           s =  beta * s;
           A(i,1:n) = A(i,1:n) - (s*u(1:n))';
        end;
       end;
```

4.1.1 La factorización QR usando matrices Householder.
Si A es una matriz cuadrada entonces existe una matriz ortogonal Q y una matriz triangular superior R tal que A=QR, con la matriz $Q=H_1H_2\ldots H_{n-1}$ donde cada H_i es una matriz de Hoseholder.
La factorización puede ser obtenida en n-1 pasos.

Paso 1: Construir una matriz Householder H_1 tal que H_1A tenga zeros debajo de la entrada (1,1) en la primera columna. Es decir,

$$H_1A=\begin{bmatrix} * & * & .... & * \\ 0 & * & .... & * \\ .... & ..... & ..... & .... \\ 0 & * & * & * \end{bmatrix}$$

Es suficiente construir $H_1= I_n - 2u_nu_n^{'}/(u_n^{'}u_n)$ tal que

$$H_1\begin{pmatrix} a_{11} \\ a_{21} \\ .... \\ a_{n1} \end{pmatrix}=\begin{pmatrix} * \\ 0 \\ .... \\ 0 \end{pmatrix}$$

Superponer la matriz A con la matriz $A^{(1)}=H_1A$

Paso 2: Construir una matriz Householder H_2 tal que $H_2A^{(1)}$ tenga zeros debajo de la entrada (2,2) en la segunda columna y que los ceros que ya se crearon en la primera columna de matriz $A^{(1)}$ no cambien. Es decir,

$$A^{(2)}=H_2A^{(1)}=\begin{bmatrix} * & * & * & ... & * \\ 0 & * & * & ... & * \\ 0 & 0 & * & ... & * \\ ... & ... & ... & ... & ... \\ 0 & 0 & * & ... & * \end{bmatrix}$$

H_2 puede ser construido como sigue: primero construir una matriz Householder $\tilde{H}_2 = I_{n-1} - 2u_{n-1}u_{n-1}^{'}/(u_{n-1}^{'}u_{n-1})$ de orden n-1 tal que

$$\tilde{H}_2\begin{pmatrix} a_{22} \\ a_{32} \\ \dots \\ \dots \\ a_{n2} \end{pmatrix} = \begin{pmatrix} * \\ 0 \\ \dots \\ \dots \\ 0 \end{pmatrix}$$

y luego definir,

$$H_2 = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \dots & & \tilde{H}_2 & \\ 0 & & & \end{pmatrix}$$

Superponer A por $A^{(2)}$.

Paso k: Construir una matriz Householder H_k tal que $H_kA^{(k-1)}$ tenga zeros debajo de la entrada (k,k) en la k-ésima columna y que los ceros que ya se crearon en los pasos anteriores no cambien. H_k puede ser construido como sigue: primero construir una matriz Householder $\tilde{H}_2 = I_{n-k+1} - 2u_{n-k+1}u'_{n-k+1}/(u'_{n+k+1}u_{n-k+1})$ de orden n-k+1 tal que

$$\tilde{H}_k\begin{pmatrix} a_{kk} \\ a_{kk+1} \\ \dots \\ \dots \\ a_{kn} \end{pmatrix} = \begin{pmatrix} * \\ 0 \\ 0 \\ \dots \\ 0 \end{pmatrix}$$

y luego definir,

$$H_2 = \begin{pmatrix} I_{k-1} & 0 \\ 0 & \tilde{H}_k \end{pmatrix}$$

Calcular $A^{(k)}=H_kA^{(k-1)}$. Superponer A por $A^{(k)}$.

Al final en el paso (n-1) la matriz resultante $A^{(n-1)}$ será la matriz triangular R.
Como, $A^{(k)}=H_kA^{(k-1)}$, para k=n-1,……2.
Tenemos

$$R=A^{(n-1)}=H_{n-1}A^{(n-2)}=H_{n-1}H_{n-2}A^{(n-3)}=\dots=H_{n-1}H_{n-2}\dots H_2H_1A$$

Hacer,

$Q'=H_{n-1}H_{n-2}\ldots\ldots H_2H_1$

Como cada matriz H_k es orthogonal tambien lo es Q'. Así que R=Q'A o A=QR.

Algoritmo para obtener la factorización QR usando Matrices Householder

Dada una matriz cuadrada A con el siguiente algoritmo se crea el vector $u_{n-k+1}=(u_{kk},\ldots u_{nk})'$, para k=1,2….n-1 que define las matrices H_1 hasta H_{n-1} y la matriz triangular superior R tal que A=QR con $Q=H_1H_2\ldots H_{n-1}$. Las componenetes $u_{k+1,k}$ hasta u_{nk} son almacenadas en la posiciones (k+1,k) hasta (n,k) de A. Las primeras componenntes u_{kk} son almacenadas en un vector unidemensional v.
For k=1,2……n-1 do

Paso 1. Hallar el vector $u_{n-k+1}=(u_{kk},\ldots u_{nk})'$ que define la matriz Householder $\widetilde{H}_k$ y la constante σ tal que

$$\widetilde{H}_k\begin{pmatrix} a_{kk} \\ a_{k+1k} \\ \ldots \\ \ldots \\ a_{nk} \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \\ \ldots \\ \ldots \\ 0 \end{pmatrix}$$

(Usar **Housecero**)

Paso 2. Superponer a_{kk} por σ

Paso 3. Almacenar el vector un-k+1 como sigue:

$a_{ik}\equiv u_{ik}$, I=k+1,…..,n

$v_k\equiv u_{kk}$

Paso 4. Calcular $\beta=2/(u'_{n-k+1}u_{n-k+1})$

Paso 5. Modificar las entradas de la submatriz A que contiene las filas k hasta n y las columnas k+1 hasta n.

For j=k+1,….n do

1. $s=\beta\sum_{i=k}^{n} u_{ik} a_{ij}$

2. $a_{ij}=a_{ij}-su_{ik}$ (i=k,k+1,….,n)

La siguiente función en MATLAB calcula la factorización QR de una matriz cudradada o no. usando matrices Householder

```
function [Q,R] = houseqr(A)
%HOUSEQR Factorizacion QR de una matriz A usando matrices
Householder
%[Q,R] = houseqr(A) produce an ortogonal matriz Q
%y una matriz triangular superior R del mismo tamaño que A
%con ceros debajo de la diagonal A tal que  A = QR.
%Este program llama a los programas HOUSECERO y HOUSEMULT.
%input   : Matriz A
%output  : Matrices Q y R

  [m,n] = size(A);
       S= min(n,m-1);
       Q = eye(m,m);
       for k = 1 : S
         [x,sigma] = housecero(A(k:m,k));
         Q(1:m,k:m) = housemult(Q(1:m,k:m),x);
         A(k,k) = sigma ;
         s1 = size(x);
         A(k+1:m,k) = x(2:s1);
         v(k) = x(1);
         beta = 2/(x'*x);
           for j = k+1:n
             s = 0;
             s = s + x(1:m-k+1)' * A(k:m,j);
             s = beta * s;
             A(k:m,j) = A(k:m,j) - s * x(1:m-k+1);
             end;
          end;
   R = triu(A);
       end;
```

Ejemplo: Calcular la factorizacón QR de las matrices

$$A=\begin{pmatrix} 4 & 2 & 5 \\ 8 & 6 & 7 \\ 1 & 9 & 5 \end{pmatrix}$$

$$B=\begin{pmatrix} 4 & 5 & 7 \\ 3 & 2 & 2 \\ 1 & 7 & 0 \\ 5 & -1 & 4 \end{pmatrix}$$

Usando Matlab y R.

Solución:
En R,

```
> A=rbind(c(4,2,5),c(8,6,7),c(1,9,5))
> A
     [,1] [,2] [,3]
[1,]    4    2    5
[2,]    8    6    7
[3,]    1    9    5
> rqa=qr(A)
> qr.Q(rqa)
           [,1]        [,2]       [,3]
[1,] -0.4444444  0.14582171  0.8838581
[2,] -0.8888889  0.05059121 -0.4553208
[3,] -0.1111111 -0.98801648  0.1071343
> qr.R(rqa)
     [,1]      [,2]      [,3]
[1,]   -9 -7.222222 -9.000000
[2,]    0 -8.296958 -3.856835
[3,]    0  0.000000  1.767716
>
> B=rbind(c(4, 5, 7),c(3, 2, 2),c(1, 7, 0),c( 5, -1, 4))
> B
     [,1] [,2] [,3]
[1,]    4    5    7
[2,]    3    2    2
[3,]    1    7    0
[4,]    5   -1    4
> qrb=qr(B)
> qr.Q(qrb)
          [,1]        [,2]       [,3]
[1,] -0.560112  0.35151479  0.7498522
[2,] -0.420084  0.04424662 -0.3576556
[3,] -0.140028  0.80872982 -0.4748942
[4,] -0.700140 -0.46950576 -0.2903096
> qr.R(qrb)
          [,1]      [,2]       [,3]
[1,] -7.141428 -3.920784 -7.5615125
[2,]  0.000000  7.976682  0.6710737
[3,]  0.000000  0.000000  3.3724160
>
```

En Matlab

```
» addpath c:\matlab\acuna
» A=[4 2 5;8 6 7;1 9 5]

A =
```

```
     4     2     5
     8     6     7
     1     9     5

» [q,r]=houseqr(A)


q =

   -0.4444    0.1458    0.8839
   -0.8889    0.0506   -0.4553
   -0.1111   -0.9880    0.1071


r =

   -9.0000   -7.2222   -9.0000
         0   -8.2970   -3.8568
         0         0    1.7677

» B=[4 5 7;3 2 2;1 7 0; 5 -1 4]

B =

     4     5     7
     3     2     2
     1     7     0
     5    -1     4

» [q,r]=houseqr(B)

q =

   -0.5601    0.3515    0.7499   -0.0208
   -0.4201    0.0442   -0.3577   -0.8329
   -0.1400    0.8087   -0.4749    0.3175
   -0.7001   -0.4695   -0.2903    0.4529


r =

   -7.1414   -3.9208   -7.5615
         0    7.9767    0.6711
         0         0    3.3724
         0         0         0

»
```