4.2 Matrices Givens y la factorización QR

Una matriz Givens es de la forma

$$G(i,j,\theta)=\begin{pmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & 1 & 0 & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \cos\theta & \sin\theta & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & -\sin\theta & \cos\theta & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & \dots & 0 & \dots & 1 \end{pmatrix}$$

(Col i ↓, Col j ↓; fila i; fila j)

El efecto de una matriz Givens aplicado a un vector es rotarlo un ángulo de θ grados. Por ejemplo,

$$G(1,2,\pi/4)\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}\cos\frac{\pi}{4}\\-\sin\frac{\pi}{4}\end{pmatrix}=\begin{pmatrix}\frac{\sqrt{2}}{2}\\-\frac{\sqrt{2}}{2}\end{pmatrix}$$

Cuando un vector n dimesional

$$x=\begin{pmatrix}x_1\\x_2\\\dots\\x_n\end{pmatrix}$$

es premultiplicado por una matriz de Householder solo las componentes i y j son afectadas, las otras componentes permanecen igual.
Como $\cos^2\theta+\sin^2\theta=1$, se puede concluir que $G(i,j,\theta)G'(i,j,\theta)=I$. En consecuencia una matriz de Givens es orthogonal.

Ejemplo. Dada la matriz de Givens

$$G(1,2,\theta)=\begin{pmatrix}c & s\\-s & c\end{pmatrix} \qquad \text{y el vector } x=\begin{pmatrix}x_1\\x_2\end{pmatrix}$$

Hallar los valores de c y s para transformar x en $\begin{pmatrix} * \\ 0 \end{pmatrix}$.

Solución.

G*x=$\begin{pmatrix} cx_1 + sx_2 \\ -sx_1 + cx_2 \end{pmatrix}$

Igualando la segunda fila a cero resulta $c=sx_1/x_2$ y usando el hecho que $c^2+s^2=1$ se llega a que $s^2(\frac{x_1^2}{x_2^2}+1)=1$, de donde $s=\frac{x_2}{\sqrt{x_1^2+x_2^2}}$ (la solución negativa se descarta por conveniencia). Sustotuyendo, se obtiene $c=\frac{x_1}{\sqrt{x_1^2+x_2^2}}$

Sin embargo, el cálculo anterior puede causar overflow o underflow y se recomienda hacer lo siguiente:

1. Si $|x_2|\geq|x_1|$ entonces

   $t=x_1/x_2$ $\quad s=\frac{1}{\sqrt{1+t^2}}$ $\quad c=st$

2. Si $|x_2|<|x_1|$ entonces

   $t=x_2/x_1$ $\quad c=\frac{1}{\sqrt{1+t^2}}$ $\quad s=ct$

La siguiente función en MATLAB encuentra las constants c y s.

```
function [c,s] = givcero(x)
%GIVCERO Hace ceros en un vector x usando matrices Givens.
%[c,s] = givcero(x) produce las constantes  c y s de una
matriz
%Givens para un vector x de dimension 2 tal que
%G(1,2,theta)x tiene un cero en la segunda entrada
%c = cos(theta), s = sin(theta).
%input   : Vector x
%output  : constantes c y s


        if abs(x(2)) > abs(x(1))
           t = x(1)/x(2);
            s = 1/((1+t*t)^.5);
            c = s*t ;
```

```
else
   t = x(2)/x(1);
    c = 1/((1+t*t)^.5);
    s = c*t ;
end
```

Ejemplo. Hallar las constantes c y s de la matriz Givens que al premultiplicarla por el vector

$$x=\begin{pmatrix}3\\5\end{pmatrix}$$

produce un cero en la segunda componente del vector x.

Solución:

» addpath c:\matlab\acuna
» x=[3;.5]

x =

3.0000
0.5000

» [c,s]=givcero(x)

c =

0.9864

s =

0.1644

Pre-multiplicación de una matriz por una matriz Givens

Cuando una matriz A se premultiplica por una matriz Givens G(i,j,θ)A el único efectos es en las filas i y j de la matriz A. Las nuevas filas son una combinación lineal de las anteriores. Mas especificamente, la nueva fila i es c veces la fila i de A mas s veces la fila j y la nueva fila j es –s veces la fila i de A mas c veces la fila j.

Dada una matriz A de orden m por n , las constants c y s de una matriz de Givens y los indice i, j tal que $(1 \le i \le j \le m)$ entonces el siguiente algoritmo superpone A con G(i,j,θ)A.

For k=1,…..,n do

$a \equiv a_{ik}$

$b=a_{jk}$

$a_{ik}=ac+bs$

$a_{jk}=-as+bc$

La siguiente función **givmult** implementa el algoritmo

```
function A = givmult(A,i,j,c,s)
%PGIVMUL Premultiplicacion  por una matriz Givens.
%A = givmult(A,i,j,c,s)  calcula the premultiplicacion
%de la matrix A por la matriz Givens G(i,j,theta)
%La matriz salida  A contiene el producto GA.
%Input   : Matriz A, parametros Givens  c y s, indices i y
j
%Output  : Matriz A

     [m,n] = size(A);
        a1 = A(i,:);
        a2 = A(j,:);
        A(i,:) = c * a1 + s * a2;
        A(j,:) = -s * a1 + c * a2;
```

Haciendo ceros en una entrada especificada de un vector cualquiera.

Sea el vector

$$x=\begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_i \\ \cdots \\ x_k \\ \cdots \\ x_n \end{pmatrix}$$

supongamos que queremos hacer 0 solamente la entrada x_k del vector. Entonces se debe usar la matriz G(i,j,θ) con i<k, las constants c y s de ella se obtienen de tal manera que

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} x_i \\ x_k \end{pmatrix}=\begin{pmatrix} * \\ 0 \end{pmatrix}$$

Ejemplo: Sea el vector

$$x=\begin{pmatrix}2\\3\\7\end{pmatrix}$$

Hallar la matriz Givens que hace 0 la tercera componente y hallar el nuevo vector.

Solucion:
Escogiendo k=3 y I=2, usando la función givcero se obtiene

```
» x=[2;3;7]

x =

     2
     3
     7

» [c,s]=givcero(x(2:3))

c =

    0.3939


s =

    0.9191

» g1=[1,0,0;0,c,s;0,-s,c]

g1 =

    1.0000         0         0
         0    0.3939    0.9191
         0   -0.9191    0.3939

» g1*x

ans =

    2.0000
    7.6158
```

```
  -0.0000
```

Otra posilidad seria escoger k=3 y i=1.

```
» x1=[2;7]

x1 =

     2
     7

» [c,s]=givcero(x1)

c =

    0.2747


s =

    0.9615

» g2=[c,0,s;0,1,0;-s,0,c]

g2 =

    0.2747         0    0.9615
         0    1.0000         0
   -0.9615         0    0.2747

» g2*x

ans =

    7.2801
    3.0000
   -0.0000

»
```

Haciendo ceros en una posición especificada de una matriz

Si se desea crear un cero en la posición (j,i) (j>i) de una matriz A entonces se construye una matriz Givens G(i,j,θ), la cual al premultiplicarla por A solo afectará las filas i y j de A.

```
function A = givmcero(i,j,A)
%GIVMCERO Hace cero una entrada de una matriz A usando
% matrices Givens
%A = givmcero(i,j,A) crea un cero en la posicion (j,i)de la
%matriz A usando una matriz Givens G. The matriz salida
%contiene el product GA tal que su entrada (j,i) es cero.
%Este programa llama a los programas GIVMULT y GIVCERO.
%input   : Enteros i, j y Matriz A
%output  : Matriz A

     [m,n] = size(A);
        x = zeros(2,1);
        x(1) = A(i,i);
        x(2) = A(j,i);
        [c,s] = givcero(x);
  A = givmult(A,i,j,c,s);
```

Ejemplo: Sea la matriz

$$A=\begin{pmatrix} 3 & -1 & 4 & 2 \\ 8 & 7 & 6 & 2 \\ 5 & 3 & 2 & 6 \\ -2 & 8 & 1 & 9 \end{pmatrix}$$

Hacer cero la entrada (4,2) usando matrices Givens.

Solución: Tenemos que usar I=2 j=4 en la función givmcero.

```
» addpath c:\matlab\acuna
» A=[3 -1 4 2;8 7 6 2;5 3 2 6;-2 8 1 9]

A =

   3   -1    4    2
   8    7    6    2
   5    3    2    6
  -2    8    1    9


» B=givmcero(2,4,A)

B =

   3.0000   -1.0000    4.0000    2.0000
```

```
   3.7629   10.6301    4.7036    8.0902
   5.0000    3.0000    2.0000    6.0000
  -7.3376         0   -3.8570    4.4214

»
```

4.2.1 Obteniendo la factorización QR usando matrices Givens

Similar a lo que se hizo con las matrices de Householder, la factorización QR de una matriz puede ser obtenida usando matrices Givens, la cual es mas conveniente para calcular los valores propios de una matriz y para resolver sistemas de ecuaciones lieneales con matrices que tienen muchos ceros. Aunque el tiempo de computación es casi el doble comparado con matrices Householder se pueden adaptar más facilmente a computación paralela.
Consideremos la matriz A de orden m x n. Lo que se busca es convertir a A en una matriz triangular superior R paso a paso.
En el primer paso se hace cero debajo de la entrada (1,1) de la primera columna obteniendose una matriz $A^{(1)}=Q_1A$ donde

$Q_1=G(1,m,\theta)G(1,m-1,\theta)\ldots G(1,2,\theta)$

es ortogonal, ya que el producto de matrices ortogonales es orthogonal.

En el segundo paso se hace cero debajo de la entrada (2,2) de la primera columna obteniendose una matriz $A^{(2)}=Q_2A$ donde

$Q_2=G(2,m,\theta)G(2,m-1,\theta)\ldots G(2,3,\theta)$

es también ortogonal, y así sucesivamente.
Sea s=min(n,m-1). Entonces

$R=A^{(s)}=Q_sA^{(s-1)}=Q_sQ_{s-1}A^{(s-2)}=\ldots..=Q_sQ_{s-1}\ldots.Q_2Q_1A=Q'A$

O equivalentemente A=QR con $Q'= Q_sQ_{s-1}A^{(s-2)}=\ldots..=Q_sQ_{s-1}\ldots.Q_2Q_1$

En la practica no es necesario formar las matrices G(i,j,θ) y G(i,j,θ)A explicitamente, ya que la primera es determinada por las constants c y s y la segunda es igual a la matriz excepto por las filas i y j.

Algoritno: Factorización QR usando matrices Givens de rotación
El siguiente algoritmo calcula la factorización QR de la matriz A, superponiendo A con R.

For k=1,2,,,,,,min(n,m-1) do
For l=k+1,….m do
Paso 1: Hallar c y s tal que

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}\begin{pmatrix} a_{kk} \\ a_{lk} \end{pmatrix}=\begin{pmatrix} * \\ 0 \end{pmatrix}$$

Paso 2: Guardar los indices k y l y las constants c y s
Paso 3: Superponer A con G(k,l,θ) usando el algoritmo de prenultiplicación de una matriz por una matriz Givens.

La siguiente función givensqr calcula la factorización QR usando matrices Givens

```
function [Q,R] = givensqr(A)
%GIVENSQR Factorizacion QR usando matrices Givens
%[Q,R] = givensqr(A) produce una matriz ortogonal Q
%y una matriz R del mismo tamaño de A con ceros
%debajo de la diagonal, tal que  A = QR.
%
%Este programa usa el programa GIVCERO
%input  : Matriz A
%output : Matrices Q and R

        [m,n] = size(A);
        Q = eye(m,m);
        for k= 1:min(n,m-1)
         for l = k+1:m
           x=[A(k,k) A(l,k)]';
           [c,s] = givcero(x);
           A1 = c * A(k,:) + s*A(l,:);
           A2 =-s * A(k,:) + c *A(l,:);
           A(k,:) = A1;
           A(l,:) = A2;
           A3 = c * Q(:,k) + s*Q(:,l);
           A4 = -s * Q(:,k) + c *Q(:,l);
           Q(:,k) = A3;
           Q(:,l) = A4;
         end
        end
        R = A;
```

Ejemplo: Hallar la factorización QR de las siguientes matrices

$$A=\begin{pmatrix} 3 & -1 & 4 & 2 \\ 8 & 7 & 6 & 2 \\ 5 & 3 & 2 & 6 \\ -2 & 8 & 1 & 9 \end{pmatrix} \quad y \quad A1=\begin{pmatrix} 3 & -1 & 4 \\ 8 & 7 & 6 \\ 5 & 3 & 2 \\ -2 & 8 & 1 \end{pmatrix}$$

```
» addpath  c:\matlab\acuna
» [q,r]=givensqr(A)

q =

    0.2970   -0.2575    0.8392   -0.3757
    0.7921    0.2974    0.0485    0.5308
    0.4951    0.0459   -0.4836   -0.7204
   -0.1980    0.9182    0.2438   -0.2413


r =

   10.0995    5.1488    6.7330    3.3665
         0    9.8229    1.7646    8.6193
         0         0    2.9245    1.0682
         0         0         0   -6.1833

» A1=[3 -1 4;8 7 6;5 3 2;-2 8 1]

A1 =

     3    -1     4
     8     7     6
     5     3     2
    -2     8     1

» [q1,r1]=givensqr(A1)

q1 =

    0.2970   -0.2575    0.8392   -0.3757
    0.7921    0.2974    0.0485    0.5308
    0.4951    0.0459   -0.4836   -0.7204
   -0.1980    0.9182    0.2438   -0.2413
```

```
r1 =

   10.0995    5.1488    6.7330
         0    9.8229    1.7646
         0         0    2.9245
         0         0         0

»
```

4.3 Los algoritmos de Gram-Schmidt y la factorización QR

Asumiendo que existen Q y R tales que la matriz A de orden m x n puede ser escrita como A=QR entonces la k-ésima columna de A se puede escribir como

$$a_k = \sum_{i=1}^{k} r_{ik} q_i$$

donde q_k es la k-ésima columna de Q. Asumiendo que A es de rango completo de la anterior ecuación se obtiene que

$$q_k = (a_k - \sum_{i=1}^{k-1} r_{ik} q_i) / r_{kk}$$

donde para que q_k tenga norma 1 se puede tomar $r_{kk} = \| a_k - \sum_{i=1}^{k-1} r_{ik} q_i) / r_{kk} \|$. Asimismo,

$$r_{ik} = q_i^{'} a_k \text{ para i=1,…k-1.}$$

En el k-ésimo paso las k-ésimas columnas de Q y R son generadas

El procedimiento anterior es llamado el método clásico de Gram-Scmidt, pero puede presentar problemas en el cálculo de q_k porque el numerador se puede cancelar. El siguiente programa en MATLAB crea una función clgs que hace QR usando el metodo clásico de Gram-Schmidt.

```
function [Q,R] = clgs(A);
%CLGS Gram-Schmidt clasico para obtener factorizacion QR
%[Q,R] = clgs(A) calcula la factorizacion QR de una
%matriz A de orden  m x n usando el metodo Gram-Schmidt
clasico
%A = QR, R es una matriz triangular superior n x n y Q es
%una matriz m x n con columnas ortonormales.

%input   : Matriz A
%output  : Matrices Q y R
```

```
  [m,n] = size(A);
 Q=eye(m,m);
  for k = 1 :n
       R(1:k-1,k) = Q(:,1:k-1)'* A(:,k);
          sum = 0;
          for i = 1:k-1
            sum = sum+ R(i,k) * Q(:,i);
          end
          Q(:,k) = A(:,k) - sum;
          R(k,k) = norm(Q(:,k));
          Q(:,k) = Q(:,k) / R(k,k);
  end
```

La siguiente función **mdgs** lleva a cabo el algoritmo modificado de Gram-Schmidt donde en el k-ésimo paso se calcula la k-ésima columna de Q y la k-ésima fila de R.

```
function [Q,R] = mdgs(A);
%MDGS Gram-Schmidt modificado para factorizacion QR
%[Q,R] = mdgs(A) calcula la factorizacion QR de una matriz
A de orden m x n
%usando el metodo modificado de Gram-Schmidt: A = QR,
%R es una matriz triangular superior  n x n
%y Q is m x n y tiene columnas orotnormales.
%input   : Matriz A
%output  : Matrices Q y R

  [m,n] = size(A);
       Q(:,1:n) = A(:,1:n);
  for k = 1 :n
    R(k,k) = norm(Q(:,k));
          Q(:,k) = Q(:,k) /R(k,k);
          for j = k+1:n
              R(k,j) = Q(:,k)'*Q(:,j);
              Q(:,j) = Q(:,j) - R(k,j) * Q(:,k);
          end;
        end;
```

Ejemplo: Hallar la factorización QR de la matriz A del ejemplo anterior usando el algoritmo clasico y modificado de Gram-Scmidt.

Solución:

```
» addpath c:\matlab\acuna
» A=[3 -1 4 2;8 7 6 2;5 3 2 6;-2 8 1 9]

A =

     3    -1     4     2
     8     7     6     2
     5     3     2     6
    -2     8     1     9

» [q,r]=clgs(A)

q =

    0.2970   -0.2575    0.8392    0.3757
    0.7921    0.2974    0.0485   -0.5308
    0.4951    0.0459   -0.4836    0.7204
   -0.1980    0.9182    0.2438    0.2413


r =

   10.0995    5.1488    6.7330    3.3665
         0    9.8229    1.7646    8.6193
         0         0    2.9245    1.0682
         0         0         0    6.1833

» [q1,r1]=mdgs(A)

q1 =

    0.2970   -0.2575    0.8392    0.3757
    0.7921    0.2974    0.0485   -0.5308
    0.4951    0.0459   -0.4836    0.7204
   -0.1980    0.9182    0.2438    0.2413


r1 =

   10.0995    5.1488    6.7330    3.3665
         0    9.8229    1.7646    8.6193
         0         0    2.9245    1.0682
         0         0         0    6.1833

» A
```

A =

```
    3    -1     4     2
    8     7     6     2
    5     3     2     6
   -2     8     1     9
```

»