Tarea5Algebra_VictorGomez

November 6, 2019

1 Tarea5 Algebra Matricial, Victor Manuel Gómez Espinosa

2 1.- Factorización LU

2.1 Funcion para hacer la factorizacion

Datos de entrada: A matriz de nxn
Salidas:
L
U

2.2 Instrucciones:

2.3 1.1 se ejecuta el siguiente bloque de código

[35]:
```
swaprows<-function(p,r1,r2){#devuelve la matriz de permutación
    di<-dim(p)
    n<-di[1] #renglones
    m<-di[2] #columnas
    #I<-diag(x=1,n,m) #Identity matrix
    vt<-matrix(0,1,m) #vector renglon de 0's
    if(r1!=r2){
        vt=I[r1,] #vector temporal
        p[r1,]=I[r2,] #intercambia renglones
        p[r2,]=vt
    }
    return(I)

}

lowerZ<-function(A,j){#devuelve una matriz de ceros excepto los elementos␣
 ↪debajo del pivote
    di<-dim(A)
    n<-di[1] #renglones
    m<-di[2] #columnas
    z<-matrix(0,n,m) #vector renglon de 1's
    i<-1:j
```

```
    z[-i,j]<-A[-i,j] #selecciona elementos debajo del pivote
    return(z)

}

my_lu<-function(A){#resuleve
    di<-dim(A) #dimensiones
    n<-di[1] #renglones
    m<-di[2] #columnas
    err<-0 #se puede hacer la factorizacion

    if(n==m){
        z<-matrix(0,n,m) #vector renglon de 0's
        l<-z #inicializa matrices y vectores
        u<-z
        tk<-z
        li<-z
        I<-diag(x=1,n,m) #Identity matrix
        p<-I
        i<-0
        j<-0
        pivot<-0

        #realiza el esscalonamiento por lu
        for(j in 1:n){
            pivot<-A[j,j] #pivote

            if(pivot!=0){#sin hacer cambios de renglon
                tk<-lowerZ(A,j)/pivot  #obtiene los elementos debajo del pivote
                li<-I-tk #matriz del tipo 3
                A<-li%*%A #se eliminan los elementos debajo del pivote

            }else if(pivot==0){#pivote 0
                cat("Error el pivote en i :",j,"j: ",j,"es cero \n" )
                vres <- list(0, 0)
                err<-1


            }

            z<-z+tk #acumula los elementos debajo del pivote

        }
        l<-I+z #obtiene la matriz l
        u<-A #u

        vres <- list(l, u)
```

```
        if(err==1){
            vres <- list(0, 0)
        }


    }else{
        cat("Error! Las dimensiones son diferentes \n")
        vres <- list(0, 0)
    }

    return(vres)

}
```

2.4 Ejemplo:

matriz de ejemplo

[36]:
```
vec<-c(2,4,-6,4,3,7,-10,6,0,2,0,4,0,0,1,5) #llenar por columnas
A<-matrix(vec,4,4) #matrix
A
```

2	3	0	0
4	7	2	0
-6	-10	0	1
4	6	4	5

Probando la funcion para obtener la matriz L y U

[37]:
```
listt<-my_lu(A) #mi funcion
l<-listt[[1]] #obtiene L
l
u<-listt[[2]] #obtiene U
u
```

1	0	0	0
2	1	0	0
-3	-1	1	0
2	0	2	1

2	3	0	0
0	1	2	0
0	0	2	1
0	0	0	3

[38]:
```
library(matlib)
LU(A) #funcion de r
```

$P

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

$L

1	0	0	0
2	1	0	0
-3	-1	1	0
2	0	2	1

$U

2	3	0	0
0	1	2	0
0	0	2	1
0	0	0	3

3 2.- Factorización LU por pivoteo parcial

3.1 Funcion para hacer la factorizacion

Datos de entrada: A matriz de nxn
Salidas:
P
L
U

[129]:
```
swaprows<-function(p,r1,r2){#devuelve la matriz de permutación
    di<-dim(p)
    n<-di[1] #renglones
    m<-di[2] #columnas
    vt<-matrix(0,1,m) #vector renglon de 0's
    if(r1!=r2){
        vt=p[r1,] #vector temporal
        p[r1,]=p[r2,] #intercambia renglones
        p[r2,]=vt
    }

    return(p)

}

lowerZ<-function(A,j){#devuelve una matriz de ceros excepto los elementos␣
 ↪debajo del pivote
    di<-dim(A)
    n<-di[1] #renglones
    m<-di[2] #columnas
    z<-matrix(0,n,m) #vector renglon de 1's
    i<-1:j
    z[-i,j]<-A[-i,j] #selecciona elementos debajo del pivote
    return(z)

}

my_lu<-function(A){#resuleve
    di<-dim(A) #dimensiones
```

```
   n<-di[1] #renglones
   m<-di[2] #columnas
   if(n==m){
       z<-matrix(0,n,m) #vector renglon de 0's
       l<-z #inicializa matrices y vectores
       u<-z
       tk<-z
       li<-z
       I<-diag(x=1,n,m) #Identity matrix
       p<-I
       i<-0
       j<-0
       pivot<-0
       x<-0

       #realiza el esscalonamiento por lu
       for(j in 1:(n-1)){
           pivot<-A[j,j] #pivote

               if(j==1){## cuando es el primer pivote
                   r1<-j
                   r2<-which(abs(A[,j])==max(abs(A[,j])))[1] #encuentra el␣
↪renglon a intercambiar
               }else{# despues del primero
                   x<-1:(j-1)
                   r1<-j
                   r2<-which(abs(A[,j])==max(abs(A[-x,j])))[1] #encuentra el␣
↪renglon a intercambiar
               }




               p<-swaprows(p,r1,r2) #actualiza la matriz p
               z<-swaprows(z,r1,r2) #actualiza la matriz l

               pp<-swaprows(I,r1,r2)
               A<-pp%*%A#intercambia los renglones


               pivot<-A[j,j]

               if(pivot!=0){
                   tk<-lowerZ(A,j)/pivot
                   li<-I-tk
```

```
                    A<-li%*%A
                }




            z<-z+tk #acumula los elementos debajo del pivote

        }
        l<-I+z #obtiene la matriz l
        u<-A #u
        vres <- list(p,l, u)

        return(vres)

    }else{
        cat("Error! Las dimensiones son diferentes \n")
    }

}
```

3.2 Ejemplo:

matriz de ejemplo

[131]:
```
vec<-c(2,4,-6,4,3,7,-10,6,0,2,0,4,0,0,1,5) #llenar por columnas
A<-matrix(vec,4,4) #matrix
A
```

2	3	0	0
4	7	2	0
-6	-10	0	1
4	6	4	5

Probando la funcion para obtener la matriz L y U

[132]:
```
listt<-my_lu(A) #mi funcion
p<-listt[[1]] #obtiene P
p
l<-listt[[2]] #obtiene L
l
u<-listt[[3]] #obtiene U
u
```

0	0	1	0
0	0	0	1
0	1	0	0
1	0	0	0

1.0000000 0.0 0.0 0
-0.6666667 1.0 0.0 0
-0.6666667 -0.5 1.0 0
-0.3333333 0.5 -0.5 1
-6 -10.0000000 0 1.000000
0 -0.6666667 4 5.666667
0 0.0000000 4 3.500000
0 0.0000000 0 -0.750000