Análisis Numérico- María Covelli, Juan Pulecio y Mateo Henao

Code

##Ejercicios
Los siguientes ejercicios estan relacionados con el tema de sistemas de ecuaciones lineales, los cuales se solucionan utilizando métodos númericos
Para la realización de los siguientes ejercicios instalar las librerias pracma, matrix y Rlinsolve

Hide

```
library(pracma)
library(Matrix)
library(Rlinsolve)
```

1. a. Revise las siguientes funciones con la matriz A, que puede decir acerca de su funcionamiento y explique como se utilizan para descomponer la matriz A

b. Evalue la matriz de transición para el método **SOR** y de *Jacobi*

Hide

```
n<-4
D1<-eye(n)
D2<-ones(n, m = n)
D3<-zeros(n, m = n)
A = matrix(c(-8.1, -7, 6.123, -2, -1, 4,
-3, -1, 0, -1, -5, 0.6,
-1, 0.33, 6, 1/2), nrow=4, byrow=TRUE)
A
```

Solución 1: Según la información brindada por la profesora para el desarrollo del punto 1 podemos apreciar que se está tratando de una matriz que no es diagonalmente dominante; Esto se sabe gracias a la evaluación de los componentes de la parte diagonal de la matriz (-8.1,4,-5 y 0,5) al compararlos con los demás elementos de las respectivas filas de la siguiente manera: 1) |-8.1|<|-7|+|6.123|+|-2| 2) |4|<|-1|+|-3|+|-1| 3) |-5|>|0|+|-1|+|0.6| 4) |0.5|<|-1|+|0.33|+|6| Se puede ver que los componentes de la diagonal en su mayoría son menores que la suma que el resto de los componentes de sus filas exceptuando la fila 3. Sin embargo, se sabe que para que una matrizz sea diagonalmente dominante se debe presentar el caso de la fila 3 en todas las filas de la matriz. Es decir, que los valores de la matriz diagonal sean mayores a la suma de los demás componentes de la fila. Dado este caso puede que nos tengamos que enfrentar a problemas de convergencia con respecto al método de Jacobi llegando a tener que usar una mayor cantidad de iteraciones o tener un error mucho más grande que por el método SOR. Con ayuda de la librería Rlinsolve vamos a poder desarrollar estos métodos con gran facilidad utilizando la función “lsolve.” en donde después del punto se pondrá sor o jacobi dependiendo del caso.A continuación se aplicarán ambos métodos para encontrar la matriz de transición; Para esto se tendrán dos matrices iniciales A y B en donde se irá resolviendo la matriz A y los valores afectarán la matriz B quedándonos de la siguiente forma:

Hide

```
cat("\n La matriz de transición por el método SOR es la siguiente:")
lsolve.sor(A,D1)

cat("\nLa matriz de transición por el método Jacobi es la siguiente:")
lsolve.jacobi(A,D1)
```

Después de desarrollar la matriz por medio de los dos métodos usando como matriz B la matriz identidad podemos apreciar que el método SOR converge mucho más rápido que el de Jacobi; A pesar de que ambas matrices finales dan resultados y errores muy parecidos nos podemos dar cuenta que si hay gran diferencia en la velocidad de convergencia ya que el método de Jacobi necesita 3 o 4 veces la cantidad de iteraciones que se utilizaron en método SOR para operar cada una de sus columnas.

2. Dada la siguiente matriz, utilice las funciones anteriores para descomponer la matriz $A = L + D + U$, recuerde que esta descomposición es la del metodo de (Jacobi). Verifique su respuesta

Adicionalmente, verifique si A es simétrica, si A es diagonalmente dominante, justifique su respuesta

Hide

```
A = matrix(c(-8.1, -7/4, 6.1, -2, -1, 4,
-3, -1, 0, -1, -5, 0.6,
-1, 1/3, 6, 1/2), nrow=4, byrow=TRUE)
A
```

b. Utilice la función itersolve(A, b, tol , method = “Gauss-Seidel”) y solucionar el sistema asociado a la matriz A con: $b = [1.45, 3, 5.12, -4]^t$ con una tolerancia de error de $1e^-8$

c. Genere las iteraciones del método de Jacobi, calcular error relativo para cada iteracion y comparar la solución con el método de Gauss-Seidel

d. Encuentre la matriz de transición y el radio espectral

Solución 2: a) Para la primera parte del problema lo que se busca es poder descomponer la mátriz A dada por la profesora en sus componentes o partes diagonal inferior (L), su parte diagonal (D) y finalmente su parte diagonal superior (U). Adicionalmente nos pide que verifiquemos si esa matriz es simétrica; Es decir, debemos verificar si la transpuesta de la matriz A es igual a la matriz original (A). Finalmente para la primera parte del punto 2 nos piden determinar si esa matriz es diagonalmente dominante. Esto ocurre cuando el valor absoluto de cada uno de los componentes de la diagonal de la matriz es mayor a la suma de el resto de los componentes de sus filas de la siguiente manera por ejemplo para la fila 1 y 2; Se desarrollara de la misma forma para las demás filas de una mariz de m renglones: |a11|>|a12|+|a13|+|a14| |a22|>|a21|+|a23|+|a24|

Hide

```
#Descomposición de la matriz en L,D y U
n=4
D3<-zeros(n, m = n)
cat("\nmatriz L \n")
aux=tril(A,-1)
L=aux+D3
L
cat("\nmatriz D \n")
D=diag(diag(A))
D
cat("\nmatriz U \n")
aux3=triu(A,1)
U=aux3+D3
U
cat("\nComprobación por medio de la suma de las tres matrices \n")
L+D+U

#Mirar si la matriz es simétrica
A.t<-t(A);
cat("\n matriz de booleanos que nos permiten saber si la matriz transpuesta es igual a la original para dete
rminar si es simétrica o no. Si todos los valores de esta igualación dan TRUE es porque si es simétrica. Por
otro lado, si alguno de esos valores da FALSE entonces la matriz no es simétrica \n")
A.t==A

#mirar si es diagonalmente dominante
diagdom<-function (A)
{
  absA = abs(A)
  diagA = 2 * (diag(absA))
  offdA = colSums(absA)
  if (all(diagA > offdA)) {
    res = "sdd"
  }
  else if ((all(diagA >= offdA)) && (any(diagA == offdA))) {
    res = "wdd"
  }
  else {
    res = FALSE
  }
  return(res)
}
aux=diagdom(A)

if(aux==TRUE){
  cat("\n La matriz A es diagonalmente dominante")
}
if(aux==FALSE) {
  cat("\n La matriz A no es diagonalmente dominante")
}
```

Según lo visto en el código establecido anteriormente pordemos definir que la matriz no es ni simétrica ni diagonalmente dominante; Esto puede complicar algunas aplicaciones de métodos como la de Jacobi y se puede evidenciar ewn su convergencia

b)Para la solución de la parte b del punto 2 se aplicará por medio del método Gauss-Seidel para solucionar el sistema de ecuaciones A igualado a b brindado por la profesora.

Hide

```
A = matrix(c(-8.1, -7/4, 6.1, -2, -1, 4,
-3, -1, 0, -1, -5, 0.6,
-1, 1/3, 6, 1/2), nrow=4, byrow=TRUE)
A
b<-c(1.45,3,5.12,-4)
cat("\n Vector b: ",b)
b

#Aplicación de la función itersolve para aplicar el método de Gauss-Seidel
cat("\n Aplicación del método itersolve: ")
itersolve(A, b, tol=1e-8 , method = "Gauss-Seidel")
D1<-eye(n)
lsolve.gs(A,D1)
#Solución del sistema de ecuaciones de matrices (función solve)
cat("\n solución del sistema de ecuaciones lineales")
solve(A,b,tol=1e-8)
```

c. Genere las iteraciones del método de Jacobi, calcular error relativo para cada iteracion y comparar la solución con el método de Gauss-Seidel

Hide

```
#Aplicación de la función itersolve para aplicar el método de Gauss-Seidel
itersolve(A, b, tol=1e-8 , method = "Jacobi")
D1<-eye(n)
lsolve.jacobi(A,D1)
```

d. Encuentre la matriz de transición y el radio espectral

Hide

```
r = max(abs(eig(A)))
cat("\n El radio espectral es el siguiente: ")
r
```

3. Sea el sistema $AX = b$ dados en ejercicio,y con tol= e^-8

a. Implemente una función en R para que evalue las raíces del polinomio característico asociado a la matriz A
b. Use el teorema de convergencia para determinar cuál método iterativo es más favorable.
c. Evalue la matriz de transición para cada caso (método) y en el caso del método de relajación determine el valor óptimo de ω
d. Teniendo en cuenta lo anterior resolver el sistema
e Comparar con la solución por defecto
e. Evaluar el número de condición de la matriz A
f. Evaluar el efecto en la solución si la entradad $a_{11} = 4.01$ aplicar cambio y solucionar. Después, debe comparar con el valor condicón

Hide

```
A = matrix(c(4, -1, -1, -1, -1, 4,
-1, -1, -1, -1, 4, -1,
-1, -1, -1, 4), nrow=4, byrow=TRUE)
A
b = c(1.11111, 5, 1.5,-2.33)
b
```

4. a. Pruebe el siguiente algoritmo con una matriz A_6, modifiquelo para que $a_{ii} = 0$ para todo i
Para realizar este problema, dado que nos pedian que para una matriz 6x6 de numeros aleatorios hacer que para la posición a_{i,i} esta cambiara de valor a 0 si ya no lo era. Para este caso lo que entendimos era que se volviera la diagonal principal de la matriz con valores de 0, esto lo realizamos en primer lugar de la matriz A inicial a traves de un doble ciclo.Este ejemplo lo hicimos como un ejemplo base. En segundo lugar, utilizando la funcion recomendada la modificamos para que fuera posible hacer el mismo procedimiento anterior es decir que de la diagonal principal de matriz fuera toda del numero 0.

Hide

```
Punto4= matrix(c(1, 2, 3, 4, 5, 6,
7,8,9,10,11,12,
13,14,15,16,17,18,
19,20,21,22,23,24,
25,26,27,28,29,30,
31,32,33,34,35,36), nrow=6, byrow=TRUE)
cat("Matriz Punto4 Original "  )
```

```
Matriz Punto4 Original
```

Hide

```
cat("\n")
```

Hide

```
for(i in 1:6){
  for(j in 1:6){
      if(i==j){
        if(Punto4[i,j]!=0){
          Punto4[i,j]<-0
        }
      }
    }
}
cat("\n")
```

Hide

```
cat("Matriz Punto4 modificada  "  )
```

```
Matriz Punto4 modificada
```

Hide

```
cat("\n")
```

Hide

```
M = matrix(c(12, 2, 23, 44, 5, 66,
7,8,99,10,11,12,
13,143,15,16,37,18,
19,20,241,22,23,24,
25,26,57,28,39,70,
31,72,33,34,35,36), nrow=6, byrow=TRUE)
cat("\n")
```

Hide

```
cat("Matriz M Original "  )
```

```
Matriz M Original
```

Hide

```
cat("\n")
```

Hide

```
M
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   12    2   23   44    5   66
[2,]    7    8   99   10   11   12
[3,]   13  143   15   16   37   18
[4,]   19   20  241   22   23   24
[5,]   25   26   57   28   39   70
[6,]   31   72   33   34   35   36
```

Hide

```
tril1 <- function(M, k)
{
  if (k != 0) {
    M[!lower.tri(M, diag = TRUE)] <- 0
    M[!upper.tri(M, diag = TRUE)] <- 0
  } else {
    M[col(M) == row(M) + k ] <- 0
  }
  cat("\n")
  cat("Matriz M Modificada "  )
  cat("\n")
  return(M)
}
M = matrix(c(12, 2, 23, 44, 5, 66,
7,8,99,10,11,12,
13,143,15,16,37,18,
19,20,241,22,23,24,
25,26,57,28,39,70,
31,72,33,34,35,36), nrow=6, byrow=TRUE)
matriz<- tril1(M, 0)
```

```
Matriz M Modificada
```

Hide

```
print(matriz)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    2   23   44    5   66
[2,]    7    0   99   10   11   12
[3,]   13  143    0   16   37   18
[4,]   19   20  241    0   23   24
[5,]   25   26   57   28    0   70
[6,]   31   72   33   34   35    0
```

5. Cree una función que cuente el número de multiplicaciones en el método directo de Gauss-Jordan, para resolver un sistema de n ecuaciones y pruebelo para $n = 5$

En este punto Lo que hizimos fue inicialmente crear una funcion que nos creara una matriz cuadrada de n filas o columnas, por lo que en cualquier comoento si se desea probar la funcion para otro n es posible solo cambiando el numero de filas y el de columnas en la funcion de crear matriz, y en el vector de b agregar un numero o restarle para que quede con el mismo numero de numeros a n. Despues de esto realizamos la funcion de eliminacion de Gauss Jordan, y dentro de esta cada vez que se realizaba una multiplicacion se contaba. en el ejemplo de n=5 obtuvimos un total de 25 multiplicaciones, mientras que para un n=6 un total de 36 multiplicaciones. Lo que nos permite analizar que para cualquier n obtendremos un numero de multiplicaciones igual a n^2.

Hide

```
GaussJordan <- function(A, b){
  if(is.matrix(A)) {
    n = nrow(A); m = ncol(A)
    if (m != n) stop("'A' La matriz debe ser cuadrada.")
  }
  cont <- 0
  Ab = cbind(A,b)
  for (k in 1:(n)){

    if(k != n){
      fila = which.max( abs(A[k:n,k]) ) + k-1

      Ab[c(k, fila), ] = Ab[c(fila, k), ]

      if(A[fila,k]==0) stop("La matriz es singular")

      for (i in (k+1):n){

        Ab[i, ] = Ab[i, ] - Ab[i, k]/Ab[k,k]*Ab[k, ]
        cont <- cont + 1
      }
    }
    for(i in (k):1){
      if(i == k){
        Ab[i, ] = Ab[i, ]/Ab[i,k]
        cont <- cont + 1
      }
      else{
        Ab[i, ] = Ab[i, ] - Ab[i,k]*Ab[k,]
        cont <- cont + 1
      }
    }
  }
  cat('\n')
  cat("El numero de multiplicaciones es de: ", cont)
  cat('\n')
  #return(Ab)
}
crearMatriz <- function(){
  return (matrix(data = floor(runif(36, min=0, max=20)), nrow = 5, ncol = 5))

}
A=crearMatriz()
A
b = c(1,2,3,4,5)

GaussJordan(A,b)
```

7. Dado el siguiente sistema:

$2x - z = 1$
$\beta x + 2y - z = 2$
$-x + y + \alpha z = 1$ a. Encuentre el valor de α y β para asegura la convergencia por el método de Jacobi y para Gauss Seidel. Sugerencia: utilice el teorema convergencia
b. Genere una tabla que tenga 10 iteraciones, del método de Jacobi con vector inicial $x_0 = [1, 2, 3]^t$

La forma en la que podemos asegurar que el sistema tenga convergencia para el método de Jacobi y de Gauss-Seidel es que la matriz sea diagonalmente dominante; es decir que el valor de la diagonal sea mayor a la suma del valor absoluto del resto de los valores de la fila por lo tanto a continuación lo que se propone es encontrar dos valores alpha y beta tales que nos sirvan para generar una matriz con la condición de que sea diagonalmente dominante

Hide

```
library(pracma)
library(Matrix)

beta = 0
alpha = 0

A = matrix(c(2, 0, -1,
             0,2 , -1,
             -1, 1, 0), nrow=3, byrow=TRUE)
b = matrix (c(1,2,1),nrow=3, byrow=TRUE)

A

#para hallar el valor de beta
i<-2
aux=0
for(j in 1:3){
  if(j!=i){
    aux<-aux+abs(A[i,j])
  }
}
#cat("aux",aux)
for(k in 0:9){
  if(A[2,2]>aux+k){
    beta=k
  }
    else{
      break;
    }
}

#Para hallar el valor de alpha
i<-3
aux2<-0
for(x in 1:3){
  if(x!=i){
    aux2<-aux2+abs(A[i,x])
  }
}
#cat("\naux",aux2)
alpha=aux2+1
cat("\n Los valores encontrados para alpha y beta que aseguran la convergencia son:")
cat("\nalpha",alpha)
cat("\nbeta",beta,"\n")

A = matrix(c(2, 0, -1,
             beta,2 , -1,
             -1, 1, alpha), nrow=3, byrow=TRUE)
b = matrix (c(1,2,1),nrow=3, byrow=TRUE)
cat("\nFinalmente la matriz es:\n")
A
b
```

Con lo visto anteriormente se encontraron dos valores (alpha=3 y beta=0) tales que nos aseguren la convergencia del sistema de ecuaciones para el método de jacobi y gauss seidel. ### 8 ————————————————————————————

8. Instalar el paquete Matrix y descomponga la matriz A (del punto dos) de la forma LU y la factorizarla como $A = QR$. Verifique su respuesta.
   #SOLUCION: Despues de instalar el paquete Matrix, lo que se realiza en el codigo es descomponer la matriz de segundo punto, la cual una vez obtenida de la manera LU,se factoriza en A=QR

Hide

```
library(Matrix)

crearMatriz <- function(){
  return (matrix(data = floor(runif(36, min=0, max=20)), nrow = 6, ncol = 6))

}


con=0
A = crearMatriz()
con=norm(A,"I")*norm(inv(A),"I")

while(con<1000 ){
  A = crearMatriz()
  con=norm(A,"I")*norm(inv(A),"I")
}
A

lu <-lu(A)
lu
```

9

9. Realice varias pruebas que la matriz de transición por el método de Gauss-Seidel esta dada por $T = (-D^{-1}U)(I + LD^{-1})^{-1}$ #SOLUCION:

Hide

```
crearMatriz <- function(){
  return (matrix(data = floor(runif(36, min=0, max=20)), nrow = 6, ncol = 6))

}

con=0
A = crearMatriz()
con=norm(A,"F")*norm(inv(A),"F")

while(con<1000 ){
  A = crearMatriz()
  con=norm(A,"F")*norm(inv(A),"F")
}
A
b = matrix(c(1,2,3,4,5,6), nrow=6, byrow=TRUE)

D = diag(diag(A))
L = tril(A,k=-1,diag = FALSE)
U = triu(A,k=1,diag = FALSE)

N <- 3
A <- Diag(rep(3,N)) + Diag(rep(-2, N-1), k=-1) + Diag(rep(-1, N-1), k=1)
x0 <- rep(0, N)
b = c(4,5,6)
print(itersolve(A, b, tol=1e-9 , methodo = "Gauss-Seidel"))
```

10

10.

a. Determinar numéricamente la intersección entre la circunferencia $x^2 + y^2 = 1$ y la recta $y = x$. Usamos una aproximación inicial $(1, 1)$. #SOLUCION: En R hay varios paquetes para resolver sistema no lineales del tipo F(x) = 0 donde F : Rp → Rp es una función nolineal con derivadas parciales continuas.

Hide

```
require(BB)
sistema = function(p){
f = rep(NA, length(v))
f[1] = p[1]^2+p[2]^2-1 # = 0
f[2] = p[1]-p[2] # = 0
f
}
p0 = c(1,1)
BBsolve(par=v0, fn=sistema)
# Successful convergence.
# sol$par
#[1] 0.7071068 0.7071068
```

b Analizar y comentar el siguiente código #SOLUCION :

Hide

```
trigexp = function(x) {#usa funcion para asignar a disparador
n = length(x) # asigna a n la longitud de x
F = rep(NA, n)# repetir parametros
F[1] = 3*x[1]^2 + 2*x[2] - 5 + sin(x[1] - x[2]) * sin(x[1] + x[2])# asigna a F la funcion
tn1 = 2:(n-1)
F[tn1] = -x[tn1-1] * exp(x[tn1-1] - x[tn1]) + x[tn1] *
( 4 + 3*x[tn1]^2) + 2 * x[tn1 + 1] + sin(x[tn1] -
x[tn1 + 1]) * sin(x[tn1] + x[tn1 + 1]) - 8
F[n] = -x[n-1] * exp(x[n-1] - x[n]) + 4*x[n] - 3
F#asigna a la casilla tn1 el resultado de la funcion que muestra
}
n = 10000 #asigna 10 mil a n
p0 = runif(n) # n realiza conjeturas iniciales al azar
sol = BBsolve(par=p0, fn=trigexp)#resuelve la funcion para saber cuantas soluciones tiene
sol$par
```# 10 mil soluciones tiene la funcion