Taller de ecuaciones lineales

Pontificia Universidad Javeriana
Miércoles 6 de Marzo de 2019

Integrantes:

Geraldine Gómez
Brayan Loaiza
Santiago Roa

1. Para el siguiente ejercicio, instale el paquete “pracma”

a) Revise las siguientes funciones con la matriz del ejercicio 2

Código:

```
library(pracma)
library(Matrix)

n = 4

D1 <- eye(n, m = n)
D2 <- ones(n, m = n)
D3 <- zeros(n, m = n)

print("D1")
D1
print("D2")
D2
print("D3")
D3

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=n, byrow=TRUE)
print("A")
A
b = matrix(c(1.45,3,5.12,-4), nrow=n, byrow=TRUE)
print("b")
```

Los resultados obtenidos fueron:

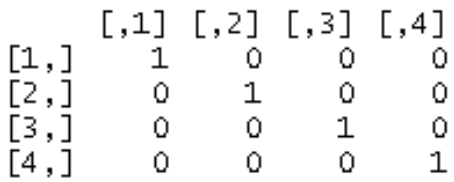
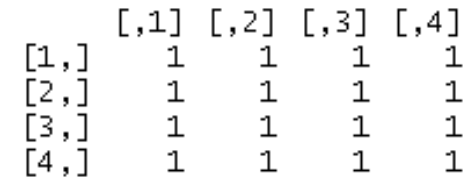
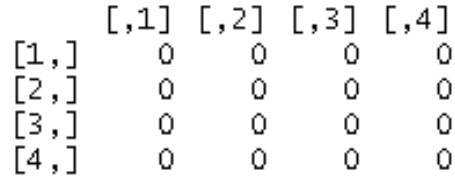
D1	
D2	
D3	

Tabla 1: se muestran los resultados obtenidos en las matrices del 1 a)

b) Evalué la matriz de transición para el método SOR

Método SOR:

- $T = -D^{-1}(L + U)$

Código:

```
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=n, byrow=TRUE)
print("A")
A

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

T = (-solve(D)) %*%(L+U)
print("T")
print(T)
print("Norma")
print(norm(T,"F"))
```

El resultado que se obtuvo fue:

```
      [,1]
[1,]  1.45
[2,]  3.00
[3,]  5.12
[4,] -4.00
```

Imagen 1: se muestran el resultado del 1 b)

2. Dada la siguiente matriz, utilice las funciones del paquete para descomponer la matriz A=L+D+U (Jacobi)

```
     [,1]  [,2]   [,3] [,4]
[1,] -8.1 -7.00  6.123 -2.0
[2,] -1.0  4.00 -3.000 -1.0
[3,]  0.0 -1.00 -5.000  0.6
[4,] -1.0  0.33  6.000  0.5
```

Imagen 2: se muestra la matriz que se va a usar en el punto 2

a) Utilice la función:
itersolve(A, b, tol , method = “Gauss-Seidel”) y solucionar el sistema asociado a la matriz AA con b=[1.45, 3, 5.12, -4] tb = [1.45, 3, 5.12, -4]t con una tolerancia de 1e-9

Código:

```
print("Gauss-Seidel:")
tol = 1e-9
sol = itersolve(A, b, x0=c(1,2,1,1), tol=1e-9 , method = "Gauss-Seidel")
print(sol)
```

b) Genere 5 iteraciones del metodo de Jacobi calcular error relativo para cada iteración
Código:

```
jacobi <- function(A,b, x0, tol){
xk = matrix(x0)

 i = 0
 repeat
  {
     inn = matrix(b-((L+U)%*%x_k))
     D1 = (solve(D))
      xk1 = D1%*%inn
      cat("Error ",i," ",norm(xk1-xk,"F")/norm(xk),"\n")
      xk = xk1
       i = i + 1
     if(i == tol)
        break
   }
    cat("5 iteraciones: ",xk,"\n")
}
```

```
x0 = c(1,2,1,1)
jacobi(A, b, x0, 5)
```

3. Sea el sistema AX = b

a) Pruebe el siguiente algoritmo con una matriz A3A3 modifiquelo para que aii = 0 para todo i

Código:

```
tril1 <- function(M, k = 0) {
if (k == 0) {
M[upper.tri(M, diag = TRUE)] <- 0
} else {
M[col(M) >= row(M) + k + 1] <- 0
M[col(M)==row(M)] <- 0

}
return(M)
}

M = matrix(c(2,3,4,1,2,3,5,6,7),nrow=3)
print(M)
print(tril1(M, k=1))
```

Estos son los resultados obtenidos, primero se muestra la matriz M, con los valores que tiene, la segunda matriz es la matriz modificada para poder cumplir la condición presentada.

print(M)	print(tril1(M, k=1))
[,1] [,2] [,3] [1,] 2 1 5 [2,] 3 2 6 [3,] 4 3 7	[,1] [,2] [,3] [1,] 0 1 0 [2,] 3 0 6 [3,] 4 3 0

Tabla 2. Se muestran los valores de la matriz M y la matriz M modificada

b) Implemente una función en R que obtenga la diagonal

Codigo:

```
diag1 <- function(M) {

  M[col(M)!=row(M)] <- 0

  return(M)

}

M = matrix(c(2,3,4,1,2,3,5,6,7),nrow=3)

print(M)

print(diag1(M))
```

Se va a volver a imprimir la matriz m y después se va a mostrar la diagonal de la matriz presentada

print(M)	print(diag1(M))
[,1] [,2] [,3] 1,] 2 1 5 2,] 3 2 6 3,] 4 3 7	[,1] [,2] [,3] 1,] 2 0 0 2,] 0 2 0 3,] 0 0 7

Tabla 3: se muestran los valores de la matriz M y de su diagonal

Para lograr un buen desarrollo de los siguiguentes puntos se van a usar algunas librerias que nos permitan realizar operaciones para simplificar el codigo, se van a usar 3, que son pracma, Matrix y BB.

4. Cree una función que cuent el número de multiplicaciones en el método directo de Gauss Jordan, para resolver un sistema de ecuaciones de n ecuaciones y compruebelo para n = 5

```
#Eliminacion gauss-jordan while
A <- matrix(c(2,-5,4,1,-2.5,1,1,-4,6),byrow=T,nrow=5,ncol=5)

## Warning in matrix(c(2, -5, 4, 1, -2.5, 1, 1, -4, 6), byrow = T, nrow = 5, :
## la longitud de los datos [9] no es un submúltiplo o múltiplo del número de
## filas [5] en la matriz

b <- matrix(c(-3,5,10),nrow=5,ncol=1)

## Warning in matrix(c(-3, 5, 10), nrow = 5, ncol = 1): la longitud de los
## datos [3] no es un submúltiplo o múltiplo del número de filas [5] en la
## matriz

p <- nrow(A)
(U.pls <- cbind(A,b))
```

	[1]	[2]	[3]	[4]	[5]	[6]
[1]	2	-5	4.0	1.0	-2.5	-3
[2]	1	1	-4.0	6.0	2.0	5
[3]	-5	4	1.0	-2.5	1.0	10
[4]	1	-4	6.0	2.0	-5.0	-3
[5]	4	1	-2.5	1.0	1.0	5

Tabla 4: Se muestran los valores de la matriz A

```
cont=1
U.pls[1,] <- U.pls[1,]/U.pls[1,1]
U.pls
```

	[1]	[2]	[3]	[4]	[5]	[6]
[1]	1	-2.5	2.0	0.5	-1.25	-1.5
[2]	1	1.0	-4.0	6.0	2.00	5.0
[3]	-5	4.0	1.0	-2.5	1.00	10.0
[4]	1	-4.0	6.0	2.0	-5.00	-3.0
[5]	4	1.0	-2.5	1.0	1.00	5.0

Tabla 5: Se muestran los cambios de los valores de la matriz A

```
i <- 2
while (i < p+1) {
```

```
 j <- i
 U.pls
 while (j < p+1) {
  U.pls
  U.pls[j, ] <- U.pls[j, ] - U.pls[i-1, ] * U.pls[j, i-1]
  j <- j+1
  cont<-cont+1
 }
 while (U.pls[i,i] == 0) {
   U.pls
  U.pls <- rbind(U.pls[-i,],U.pls[i,])
 }
 U.pls
 U.pls[i,] <- U.pls[i,]/U.pls[i,i]
 i <- i+1
 cont<-cont+1
}
for (i in p:2){
 for (j in i:2-1) {
   U.pls
  U.pls[j, ] <- U.pls[j, ] - U.pls[i, ] * U.pls[j, i]
  cont<-cont+1
 }
}
U.pls

cat("Numero de multiplicaciones: ",cont,"\n")
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]
1,]	1	0	0	0	0	2.004716
2,]	0	1	0	0	0	3.322281
3,]	0	0	1	0	0	5.459768
4,]	0	0	0	1	0	1.723882
5,]	0	0	0	0	1	5.584393

Tabla 6: Se muestra el resultado del sistema de ecuaciones

Numero de multiplicaciones: 25

5. Dado el sistema:

$$2x - z = 1$$
$$\beta x + 2y - z = 2$$
$$-x + y + \alpha z = 1$$

a) Encuentre el valor de α y β para asegurar la convergencia por el metodo de Jacobi.
Si la matriz del sistema original es diagonalmente dominante, seguro converge por método de Jacobi.
La matriz diagonalmente dominante se denomina como aquella matriz en la cual cada valor de su diagonal debe ser mayor que la suma de los otros coeficientes de la fila en la que se encuentra el valor.
En este caso beta debe ser:

$$\beta - 1 < 2$$
$$\beta < 3$$

para alpha se toma el valor de:

$\alpha > -1 + 1$
$\alpha > 0$

En mi caso tomaremos a alpha y beta con un valor igual a 2 cada uno

b) Genere una tabla que tenga 10 iteraciones del método de Jacobi con el vector inicial:

$$x_0 = [1,2,3]^T$$

```
#Punto 5b
eye <- function(n, m = n) {
  stopifnot(is.numeric(n), length(n) == 1,is.numeric(m), length(m) == 1)
  n <- floor(n)
  m <- floor(m)
  if (n <= 0 || m <= 0){
    return(matrix(NA, 0, 0))
  }
  else{
    return(base::diag(1, n, m))
  }
}

jacobi <- function(A, b, x0 = NULL,nmax = 10, tol = .Machine$double.eps^(0.5)){
  stopifnot(is.numeric(A), is.numeric(b))
  n <- nrow(A)
  if (ncol(A) != n){
    stop("Argumento 'A' tiene que ser cuadrado.")
  }
  b <- c(b)
  if (length(b) != n){
    stop("Argumento 'b' tiene que tener valor n = ncol(A) = nrow(A).")
  }
  if (is.null(x0)) {
    x0 <- rep(0, n)
  } else {
    stopifnot(is.numeric(x0))
    x0 <- c(x0)
    if (length(x0) != n){
      stop("Argumento 'x0' tiene que tener valor n=ncol(A)=nrow(A).")
    }
  }

  L <- diag(diag(A))
  U <- eye(n)
  beta <- 2; alpha <- 2


  b <- as.matrix(b)
  x <- x0 <- as.matrix(x0)
  r <- b - A %*% x0
  r1 <- error <- norm(r, "f")
```

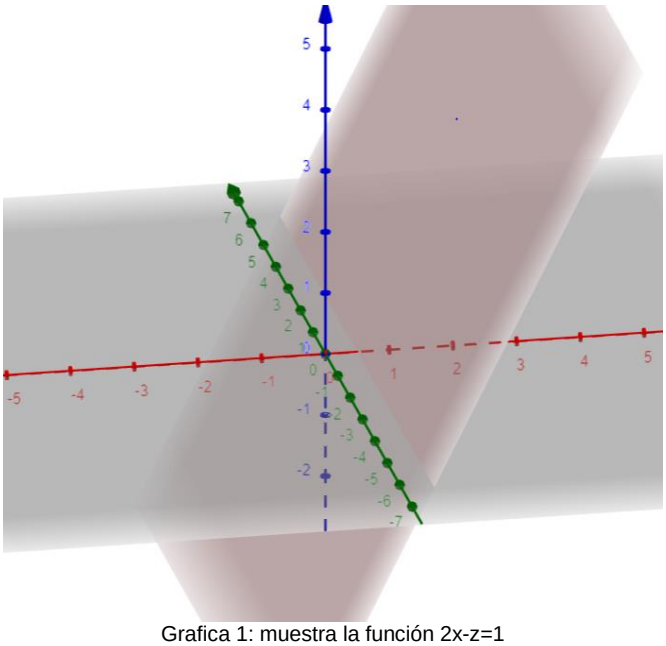
```
  i <- 0
  while (error > tol && i < nmax) {
    cat("i= ",i,"\tE error= ",error,"\n")
    i <- i + 1
    z <- qr.solve(L, r)
    z <- qr.solve(U, z)
    if (beta == 2){
      alpha <- drop(t(z) %*% r/(t(z) %*% A %*% z))
    }
    x <- x + alpha * z
    r <- b - A %*% x
    error <- norm(r, "f") / r1
  }
  return(list(x = c(x), i = i))
}
A = matrix(c(2,0,-1,
             1,2,-1,
             -1, 1,1), nrow=3, byrow=TRUE)
b = c(1,2,1)
x0 = c(1,2,3)
jacobi(A, b, tol = 1e-8)
```

Tabla de resultados

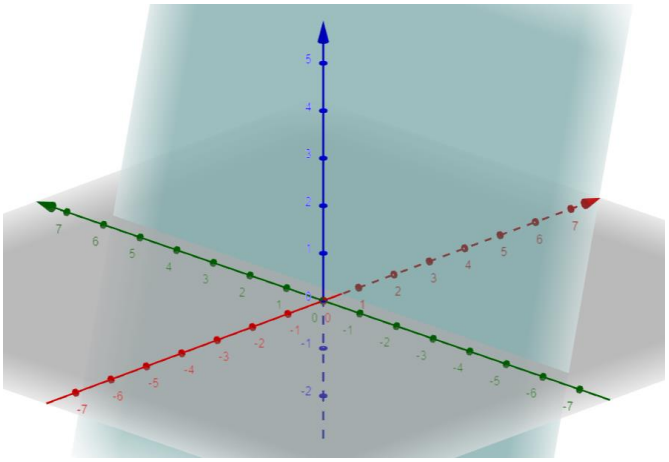
Cantidad	Error
0	2.44949
1	0.5204165
2	0.2529292
3	0.1373767
4	0. 1314463
5	0.03397383
6	0.02734092
7	0.02165982
8	0.01040671
9	0.006381741

Tabla 7: el valor del error que tuvo cada una de las iteraciones

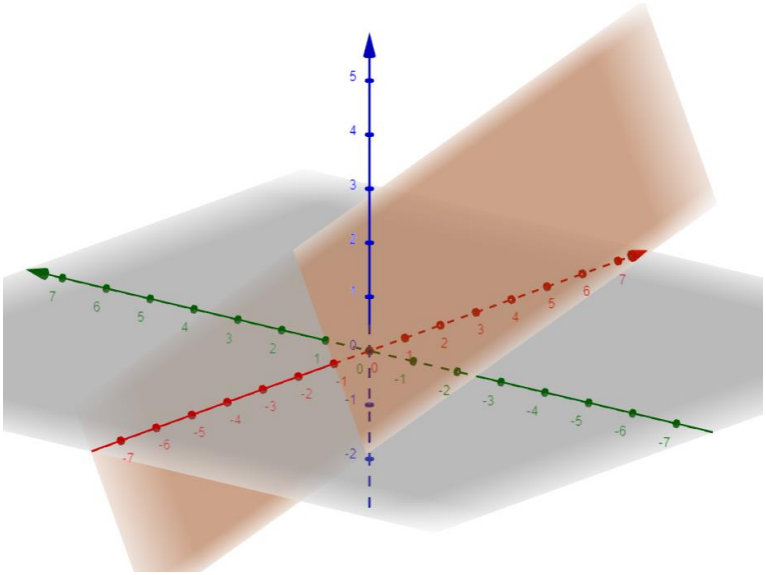
c) Grafique cada ecuación y la solución



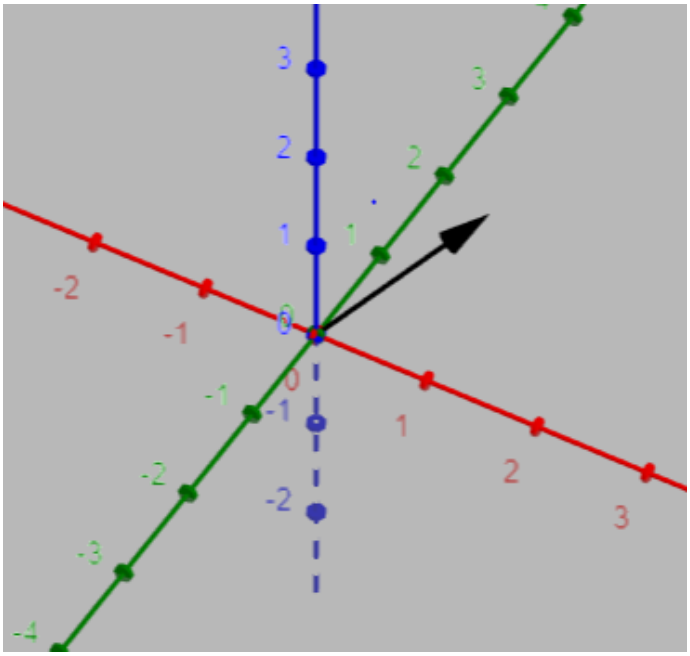
Grafica 1: muestra la función 2x-z=1



Grafica 2: muestra la función 2x+2y-z=2



Grafica 3: muestra la función -x+2y+2z=1



Grafica 4: muestra el Vector Solución

6. Instalar el paquete Matrix y descomponga la matriz A (del punto dos) de la forma LU y factoricela de la forma $A = QR$

```
#Punto 6
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)

luA = expand(lu(A))
qrA <- qr(A)
Q = qr.Q(qrA)
R = qr.R(qrA)
```

La descomposición LU es:

```
$L
4 x 4 Matrix of class "dtrMatrix" (unitriangular)
     [,1]       [,2]       [,3]       [,4]
[1,]  1.0000000          .          .          .
[2,]  0.1234568  1.0000000          .          .
[3,]  0.1234568  0.2455076  1.0000000          .
[4,]  0.0000000 -0.2055838 -0.9360990  1.0000000
```

```
$U
4 x 4 Matrix of class "dtrMatrix"
     [,1]       [,2]       [,3]       [,4]
[1,] -8.1000000 -7.0000000  6.1230000 -2.0000000
[2,]          .  4.8641975 -3.7559259 -0.7530864
[3,]          .          .  6.1661825  0.9318020
[4,]          .          .          .  1.3174366
```

La factorización de la forma A=QR es:

```
$qr
          [,1]       [,2]       [,3]      [,4]
[1,] 8.2225300  6.3690859 -6.3966078 2.0310050
[2,] 0.1216171 -5.0540721  1.4128123 0.6669168
[3,] 0.0000000 -0.1978603  8.0360748 0.3488413
[4,] 0.1216171  0.2273528 -0.7262035 0.9584103
```

7.

a) Determinar numérica la intersección entre la circunferencia $x^2 + y^2 = 1$y la recta y = x. Usamos una aproximación inicial (1,1). Utilice el paquete BB y la función BBsolve() del paquete, grafique la solución.

Para poder hallar los puntos de corte que tiene la circunferencia $x^2 + y^2 = 1$, ecuación 1, con x = y, ecuación 2. Se tiene reemplazar el valor de y que nos da la ecuación 2 en la ecuación 1, para que así nos quede:

$x^2 + x^2 = 1$

$2x^2 = 1$

$x = \sqrt{\frac{1}{2}}$

$x = \pm 0{,}70710$

Este valor de x lo que nos indica es que va a tener dos puntos de corte en el x, que es cuando x = -0,70710 y x = 0,70710.

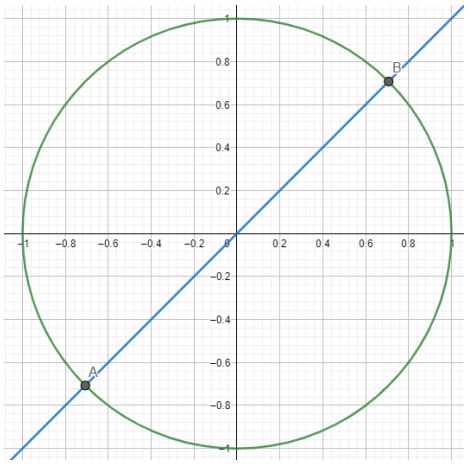
Para poder hallar los valores de y, vamos a reemplazar los valores de x en la ecuación 2, de esta manera nos queda:

$y = \pm 0{,}70710$, es decir, va a tener los mismos valores que va a tener x.

Finalmente se tienen 4 posibles puntos de corte, pero la ecuación 2 es la que nos va a definir los dos puntos de corte. Ya que x, y tienen que tener los mismos valores, para poder tener lógica con respecto a esta ecuación, los puntos de corte son:

A = (-0,70710, -0,70710) y

B = (0,70710, 0,70710)



Grafica 5: muestra la grafica de las dos funciones

b) Analizar y comentar el siguiente código

```
trigexp <- function(x) {#nombre de la función
  n <- length(x)
  F <- rep (NA, n)
  F [1] <- 3*x [1]^2 + 2*x [2] - 5 + sin (x [1] - x [2]) * sin (x [1] + x [2])
  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
}
n <- 10000
p0 <- runif(n) # n initial random starting guesses
sol <- BBsolve (par=p0, fn=trigexp)
sol$par
```

8. Demuestre y realice varias pruebas que la matriz de transición por el método de Gauss-Seidel está dada por $T = (-D^{-1}U)(I + LD^{-1})^{-1}$

Código:

```
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)
```

itersolve(A, b, tol=1e-9 , method = "Gauss-Seidel")

Demostración:

Para demostrar la matriz de transición para el método SQR $T = (-D^{-1}U)(I + LD^{-1})^{-1}$, toca aplicar la definición de convergencia del método del error de truncamiento, que establece que:

$$E^{k+1} = TE^{k}$$

Desarrollando así el lado izquierdo de la igualdad se obtiene:

$$X - X^{k+1} = -D^{-1}L(X - X^{k+1}) - UD^{-1}(X - X^{k}))$$

$$E^{k+1} = -D^{-1}LE^{k+1} - UD^{-1}E^{k}$$

$$E^{k+1} + D^{-1}LE^{k+1} = -UD^{-1}E^{k}$$

$$E^{k+1}(I + D^{-1}L) = -D^{-1}E^{k}$$

Finalmente se llega a la expresión:

$$E^{k+1} = (-D^{-1}U)(I + D^{-1}L)^{-1}E^{k}$$

Comparando con la ecuación original $E^{k+1} = TE^{k}$, queda demostrado que la matriz transición T es: $T = (-D^{-1}U)(I + LD^{-1})^{-1}$