Taller Sistemas

Jimenez Nelson, Velandia Joan

17 de agosto de 2019

Punto 1

A)

Para una matriz A de tamaño $n * n$ la función $eyes()$ genera la matriz identidad, o mejor dicho genera una matriz en su diagonal 1 y demás 0. La función $ones()$ genera una matriz con unicamente valores 1. La función $zeros()$ genera una matriz con unicamente valores nulos. ##B)Matriz de transición por el método SOR

EL méto de SOR es una mejora a la iteración de Gauss-Seidel, donde se escoge un parametro W de ralajación tal que:

si $w = 1$, La solición dada es equivalente a la solición que proporciona Gauss-Seidel y se dice que no hay relajación.

si $0 < w < 1$, El valor de la aproximación estará más cercano a la iteración $k + 1$ o a la actual, y se dice que hay subrelajación. Generalmente se usa para la convergencia de sistemas que no convergen por Gauss-Seidel.

si $1 < w < 2$, El valor de la aproximación será cercano a la iteracion $k + 1$ y se dice que hay sobre-relajación y acelera la convergencia de sistemas convergentes y lentos como Gauss-Seidel.

Usaremos un factor de relajacion $\omega = 1.9$. Obteniendo el siguiente resultado

```
## 4 x 4 Matrix of class "dgeMatrix"
##           [,1]        [,2]      [,3]       [,4]
## [1,] -0.900000  -1.6419753  1.436259 -0.4691358
## [2,] -0.427500  -1.6799383  2.107223  0.2521605
## [3,]  0.162450   0.6383765 -1.700745  0.1321790
## [4,] -6.587775 -18.6878488 41.592309 -6.0126068
```

Punto 2

A)

Diagonal:

```
##      [,1] [,2] [,3] [,4]
## [1,] -8.1    0    0  0.0
## [2,]  0.0    4    0  0.0
## [3,]  0.0    0   -5  0.0
## [4,]  0.0    0    0  0.5
```

Triangular inferior

```
## 4 x 4 Matrix of class "dtrMatrix"
##      [,1]  [,2]  [,3]  [,4]
## [1,]  0.00     .     .     .
## [2,] -1.00  0.00     .     .
## [3,]  0.00 -1.00  0.00     .
## [4,] -1.00  0.33  6.00  0.00
```

Triangular superior

```
## 4 x 4 Matrix of class "dtrMatrix"
##      [,1]   [,2]   [,3]   [,4]
## [1,]  0.000 -7.000  6.123 -2.000
## [2,]      .  0.000 -3.000 -1.000
## [3,]      .      .  0.000  0.600
## [4,]      .      .      .  0.000
```

B)

Con una tolerancia $= e^{-9}$, la solución de Gauss-Seidel fue:

```
## $x
## [1] -2.842399e+196  2.194379e+196  1.238980e+196 -2.200084e+197
##
## $iter
## [1] 1000
##
## $method
## [1] "Gauss-Seidel"
```

C)

```
## Error  0   4.118073
## Error  1   1.014324
## Error  2   2.830197
## Error  3   1.268594
## Error  4   2.063461
## Solucion a 5 iteraciones:            [,1]
## [1,]  -4.627352
## [2,]  -1.394185
## [3,]  -4.904280
## [4,] -89.515747
```

sin embargo la función $solve()$ de R nos da la siguiente solución:

```
##              [,1]
## [1,] -1.091148268
## [2,]  0.002898634
## [3,] -0.920772493
## [4,]  0.865060281
```

Punto 3

A) En orden de mayor grado el resultado es:

```
##      [,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

## [1]    1.0000    8.6000  -31.7200 -224.3022  320.0681
```

B) Mejor método iterativo

```
## [1] "Convergencia Gauss-Siedel"
```

```
## [1] 1.6026
## [1] "Convergencia Jacobi"
## [1] 14.66
```

C) Matriz de transición

```
## [1] "Matriz transicion Gauss"
## 4 x 4 Matrix of class "dgeMatrix"
##             [,1]       [,2]      [,3]       [,4]
## [1,]  0.12047325 -0.7289198 0.4596296 -0.2469136
## [2,] -0.06072531  0.2418750 1.0500000  0.2500000
## [3,] -0.01803704  0.0261000 0.1440000  0.1200000
## [4,]  0.00000000  0.0000000 0.0000000  0.0000000
## [1] "Matriz transicion Jacobi"
## 4 x 4 Matrix of class "dgeMatrix"
##      [,1]       [,2]        [,3]       [,4]
## [1,] 0.00 -0.8641975   0.7559259 -0.2469136
## [2,] 0.25  0.0000000   0.7500000  0.2500000
## [3,] 0.00 -0.2000000   0.0000000  0.1200000
## [4,] 2.00 -0.6600000 -12.0000000  0.0000000
```

D)

```
##      [,1] [,2] [,3] [,4]
## [1,]    4   -1   -1   -1
## [2,]   -1    4   -1   -1
## [3,]   -1   -1    4   -1
## [4,]   -1   -1   -1    4
## [1]  1.00  5.00  1.50 -2.33
## $x
## [1] 1.234 2.034 1.334 0.568
##
## $iter
## [1] 60
##
## $method
## [1] "Jacobi"
## $x
## [1] 1.234 2.034 1.334 0.568
##
## $iter
## [1] 36
##
## $method
## [1] "Gauss-Seidel"
## [1] 1.234 2.034 1.334 0.568
```

F)

Modificación función tril1:

```r
tril1 <- function(M, k = 0) {
  if (k == 0)
  {
    M[upper.tri(M, diag = TRUE)] <- 0
  }
  else
  {

    M[col(M)==row(M)] <- 0
  }
  return(M)
}

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

```
##      [,1] [,2] [,3]
## [1,]    2    1    5
## [2,]    3    2    6
## [3,]    4    3    7
```

```r
print(tril1(M, k=1))
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    5
## [2,]    3    0    6
## [3,]    4    3    0
```

```r
tril2 <- function(M, k = 0) {
  if (k == 0)
  {
    M[upper.tri(M)] <- 0
  }
  else
  {
    M[col(M) >= row(M) + k + 1] <- 0
  }
  return(M)
}

print(tril2(M,k=1))
```

```
##      [,1] [,2] [,3]
## [1,]    2    1    0
## [2,]    3    2    6
## [3,]    4    3    7
```

G)

```r
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)
```

```
##      [,1] [,2] [,3]
## [1,]    2    1    5
## [2,]    3    2    6
## [3,]    4    3    7
```

```
print(diag1(M))
```

```
##      [,1] [,2] [,3]
## [1,]    2    0    0
## [2,]    0    2    0
## [3,]    0    0    7
```

Punto 4

```
gauss = function(A, b)
{
  mult = 0
  n = nrow(A) # = ncol(A) para que sea cuadrada

  # matriz ampliada
  Ab = cbind(A,b)
  print(Ab)
  # Eliminación
  for (k in 1:(n-1)){ # desde columna k=1 hasta k=n-1
    if(Ab[k,k]==0){ # intercambio de fila
      fila = which(Ab[k, ]!=0)[1]
      Ab[c(k, fila), ] = Ab[c(fila, k), ]
    }

    # Eliminación columna k
    for (i in (k+1):n){# debajo de la diagonal
      # Fi = Fi - a_ik/a_kk * Fk, i=k+1,...,n
      Ab[i, ] = Ab[i, ] - Ab[i, k]/Ab[k,k]*Ab[k, ]
      mult = mult + 2*(ncol(Ab))
    }
  }

  # Sustitución hacia atrás-------------------------
  # b(i) = A[i, n+1]
  x = rep(NA, times=n)
  x[n] = Ab[n, n+1]/Ab[n,n] # xn = bn/a_nn
  mult = mult + n+1

  for(i in (n-1):1 ){
    x[i]= (Ab[i, n+1] - sum(Ab[i, (i+1):n]*x[(i+1):n]) ) /Ab[i,i]
    mult = mult + 2*(n-2)
  }
```

```
  #cat(x, "\n")
  cat("Numero de multiplicaciones:", mult, "\n")
  return(x)
}

A = matrix(c( 0,  2,  3, 3, 3,
             -5, -4,  1, 4, 5,
              0,  0,  0, 3, 7,
             -4, -7, -8, 9, 7,
              3,  4,  5, 5, 6), nrow=5, byrow=TRUE)
b = matrix(c(1,0,0,0,1), nrow=5, byrow=TRUE)
print("solucion")
```

```
## [1] "solucion"
```

```
gauss(A,b)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    0    2    3    3    3    1
## [2,]   -5   -4    1    4    5    0
## [3,]    0    0    0    3    7    0
## [4,]   -4   -7   -8    9    7    0
## [5,]    3    4    5    5    6    1
## Numero de multiplicaciones: 150
```

```
## [1] -0.28978300  0.44032550 -0.08137432  0.21202532 -0.09086799
```

Punto 5

A)

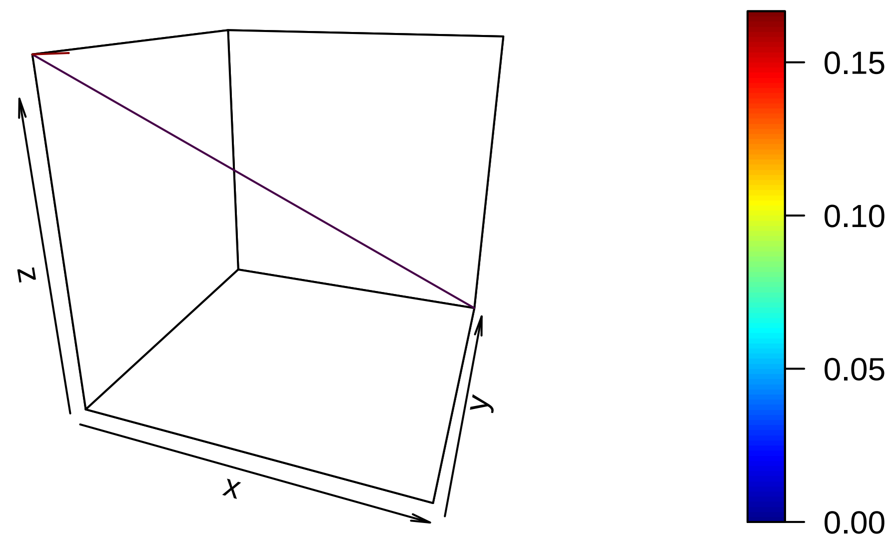
Se llega a los valores de alpha y beta por las operaciones de $\alpha > 1 + 1$, $\beta + 1 < 2$ de acuerdo a su posisción en la matriz

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    0   -1    1
## [2,]    0    2   -1    2
## [3,]   -1    1    3    1
```

B)

```
## Error  1   0.5335937
## Solucion iteracion  1 :  2   2.5   0
## Error  2   0.4728572
## Solucion iteracion  2 :  0.5   1   0.1666667
## Error  3   0.07071068
## Solucion iteracion  3 :  0.5833333   1.083333   0.1666667
## Error  4   1.51394e-17
## Solucion iteracion  4 :  0.5833333   1.083333   0.1666667
## Error  5   0
## Solucion iteracion  5 :  0.5833333   1.083333   0.1666667
## Error  6   0
## Solucion iteracion  6 :  0.5833333   1.083333   0.1666667
## Error  7   0
## Solucion iteracion  7 :  0.5833333   1.083333   0.1666667
```

```
## Error  8   0
## Solucion iteracion  8 :  0.5833333   1.083333   0.1666667
## Error  9   0
## Solucion iteracion  9 :  0.5833333   1.083333   0.1666667
```




```
## Solucion a  10  iteraciones:  0.5833333 1.083333 0.1666667
```

Punto 6

A) Descomposición LU

Descomposición LU:

L:

```
Mlu = lu(A)
Mlu = expand(Mlu)

Mlu$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
```

```r
print("Matriz A")
```

```
## [1] "Matriz A"
```

```r
A
```

```
##      [,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
```

```r
print("A = PLU")
```

```
## [1] "A = PLU"
```

```r
print(Mlu$P%*%Mlu$L%*%Mlu$U)
```

```
## 4 x 4 Matrix of class "dgeMatrix"
##      [,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
```

B) Factorización A = QR

Q:

```
##            [,1]       [,2]        [,3]        [,4]
## [1,] -0.9850983 -0.1436118 -0.04743408 -0.08189671
## [2,] -0.1216171  0.9447015 -0.30403534 -0.01763100
## [3,]  0.0000000 -0.1978603 -0.65697987  0.72748111
## [4,] -0.1216171  0.2185544  0.68825138  0.68099435
```

```
##         [,1]     [,2]      [,3]       [,4]
## [1,] 8.22253 6.369086 -6.396608  2.0310050
## [2,] 0.00000 5.054072 -1.412812 -0.6669168
## [3,] 0.00000 0.000000  8.036075  0.3488413
## [4,] 0.00000 0.000000  0.000000  0.9584103
```

```
##            [,1]       [,2]        [,3]        [,4]
## [1,] -0.9850983 -0.1436118 -0.04743408 -0.08189671
## [2,] -0.1216171  0.9447015 -0.30403534 -0.01763100
## [3,]  0.0000000 -0.1978603 -0.65697987  0.72748111
## [4,] -0.1216171  0.2185544  0.68825138  0.68099435
```

R:

```
##         [,1]     [,2]      [,3]       [,4]
## [1,] 8.22253 6.369086 -6.396608  2.0310050
## [2,] 0.00000 5.054072 -1.412812 -0.6669168
## [3,] 0.00000 0.000000  8.036075  0.3488413
```

```
## [4,] 0.00000 0.000000  0.000000  0.9584103
```

A = Q*R

```r
print(Q %*% R)
```

```
##       [,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
```

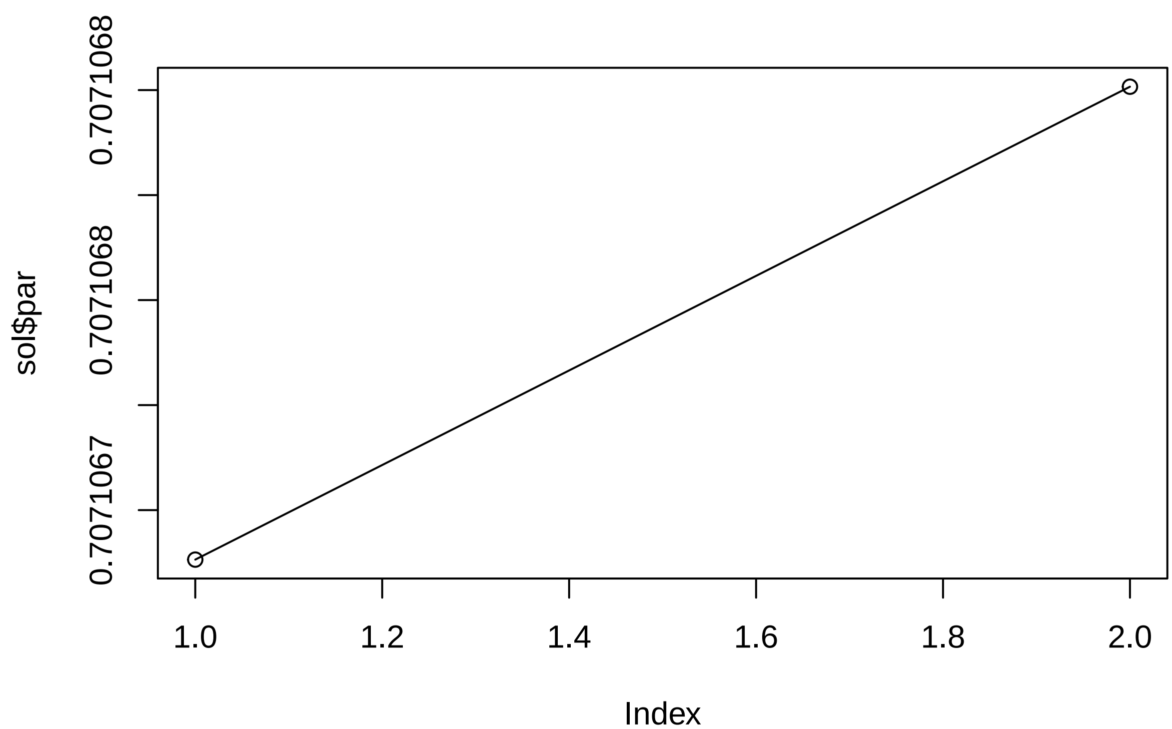
Punto 7

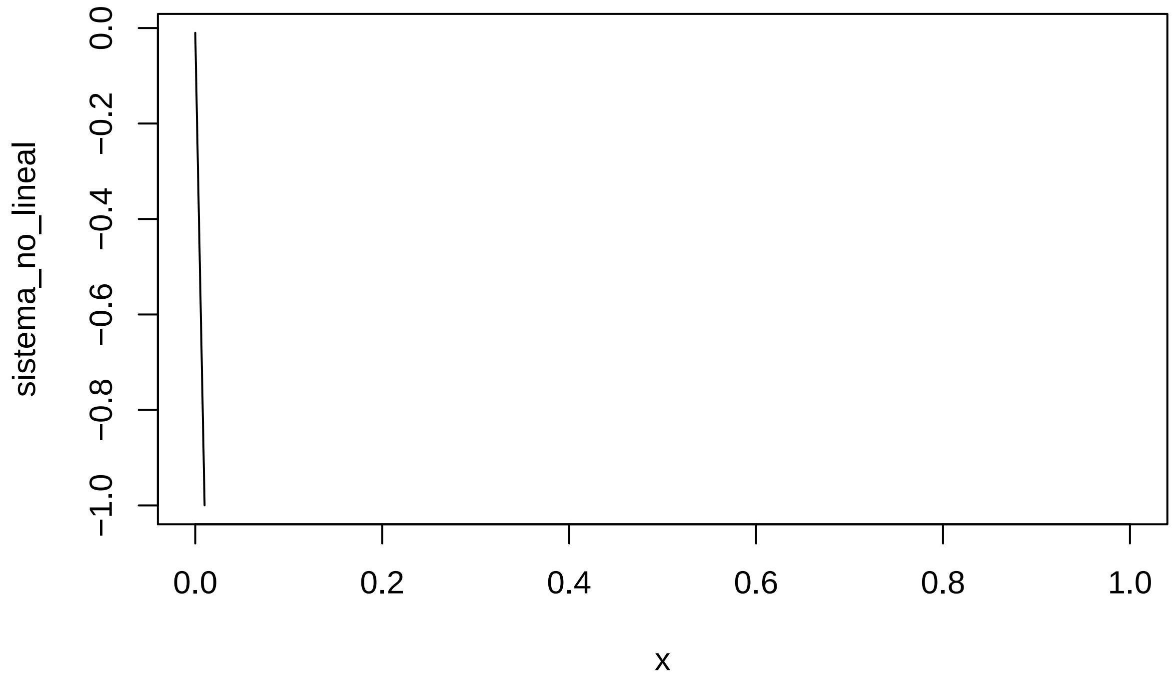
A)

```
##   Successful convergence.
```

```
## [1] 0.7071067 0.7071068
```

```r
plot(sol$par, type = 'o')
```





Punto 8

```
trigexp = function(x) {

  #Tamaño del vector que llega por parámetro
  n = length(x)
  #se crea un vector F vacío
  F = rep(NA, n)

  #Se enuncian las ecuaciones del sistema
  F[1] = 3*x[1]^2 + 2*x[2] - 5 + sin(x[1] - x[2]) * sin(x[1] + x[2])
  #Se crea una secuencia de 2 hasta n-1
  tn1 = 2:(n-1)
  #Se evalúan tn_1 ecuaciones
  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
  #Se evalúa la última ecuación n
  F[n] = -x[n-1] * exp(x[n-1] - x[n]) + 4*x[n] - 3
  #Se retorna F
  F
}
n = 10000
p0 = runif(n) #Genera n numeros aleatorios entre 0 y 1.
```

```r
#se halla la solución del sistema trigexp usando BBsolve de la librería BB, utilizando n valores inicia
sol = BBsolve(par=p0, fn=trigexp)
```

```
##   Successful convergence.
```

```r
  #sol$par #Muestra el vector solución del sistema para cada n valores iniciales
```

Punto 9

A) Demostración:

Sistemas de ecuaciones lineales $AX = B$

Ecuación recurrente equivalente sustituyendo $A = LDU$

$X = D^{-1}B - D^{-1}LX - D^{-1}UX$, siempre que D^{-1} exista

Ecuación recurrente iterativa de Gauss-Seidel $X^{k+1} = D^{-1}B - D^{-1}LX^{k+1} - D^{-1}UX^k$

Restar las ecucaciones para aplicar la definición de convergencia: $E^{k+1} = TE^k$ $X - X^{k+1} = D^{-1}B - D^{-1}L(X - X^{k+1}) - D^{-1}U(xX - X^k)$ $E^{k+1} = -D^{-1}LE^{k+1} - D^{-1}UE^k$ $E^{k+1}(I + D^{-1}L) = -D^{-1}UE^k$ $E^{k+1} = (I + D^{-1}L)^{-1}(-D^{-1}U)E^k$

Matriz de transición: $T = (I + D^{-1}L^{-1}(-D^{-1}U))$

B)

```
##      [,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

## $x
## [1] -1.369011e+197  1.056899e+197  5.967412e+196 -1.059647e+198
##
## $iter
## [1] 1000
##
## $method
## [1] "Gauss-Seidel"

## [1] "Convergencia Gauss"

## [1] 1.6026

## [1] "Matriz transicion Gauss"

## 4 x 4 Matrix of class "dgeMatrix"
##             [,1]       [,2]      [,3]       [,4]
## [1,]  0.12047325 -0.7289198 0.4596296 -0.2469136
## [2,] -0.06072531  0.2418750 1.0500000  0.2500000
## [3,] -0.01803704  0.0261000 0.1440000  0.1200000
## [4,]  0.00000000  0.0000000 0.0000000  0.0000000
```