Económicas, UBA. Actuario. Análisis Numérico.

Cuatrimestre 1, 2021. RECUPERATORIO del Primer Examen Parcial.
Para aprobar, debe sumar 50 puntos.

Remplace este texto por su Apellido y Nombre, y su Numero de Registro

02/julio/2021

Contents


```
## Warning: package 'flextable' was built under R version 4.1.1

## Warning: package 'tidyverse' was built under R version 4.1.1

## Warning: package 'tibble' was built under R version 4.1.1

## Warning: package 'tidyr' was built under R version 4.1.1

## Warning: package 'readr' was built under R version 4.1.1

## Warning: package 'purrr' was built under R version 4.1.1

## Warning: package 'dplyr' was built under R version 4.1.1

## Warning: package 'stringr' was built under R version 4.1.1

## Warning: package 'forcats' was built under R version 4.1.1
```

1 Resolución de Ecuaciones: Secante. (24 puntos)

Considere la siguiente ecuación: $2cos(x) = e^{-10/x}$.

1.1 Corregir algoritmo

CORREGIR el algoritmo "Secante" en el siguiente bloque de código.

COMENTAR los cambios que realizó (use "#" al final de cada línea modificada).

[***Notar que, para el resto del ejercicio, no debe utilizar su propio algoritmo, sino que debe usar el algoritmo dado y corregido, sin agregar ninguna línea de código adicional.***]

Respuesta:

```
# Edite las líneas que considere erróneas
# Comente al final de cada línea editada
Secante <- function(f,p0,p1,TOL,N){
  i <- 2
  q0 <- f(p0)
  q1 <- f(p1)
  while (i <= N){ #Cambio el == por <=
    p = p1 - q1*(p1-p0)/(q1-q0) #Modifico (q1-q0)/(p1-p0)
    if (abs(p-p1) < TOL){ #q-q1
      return(p)
    }
    i = i + 1
    p0 = p1 #p
    q0 = q1
    p1 = p #p0
    q1 = f(p) #f(q0)
  }
  return(paste('El metodo fallo luego de ', N, ' iteraciones')) #n por N
}
```

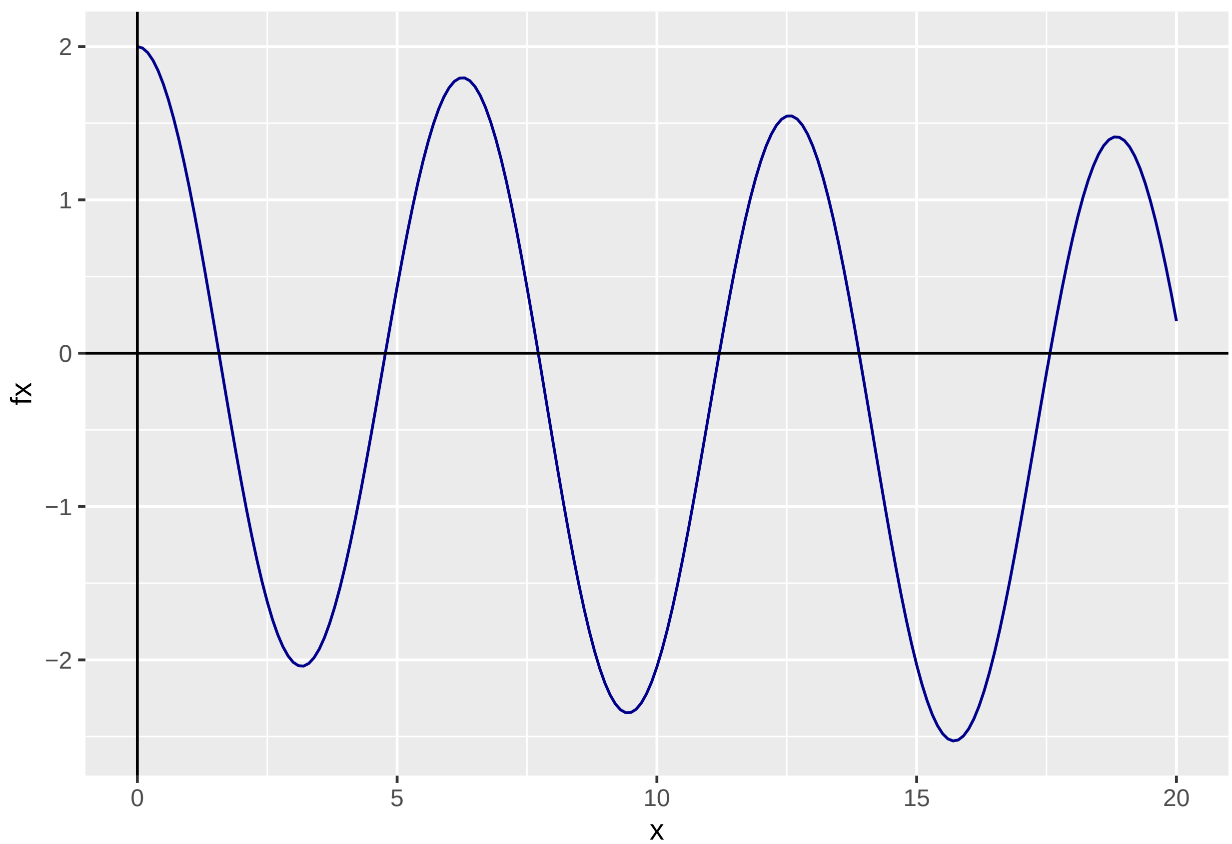
1.2 Graficar función

Plantee la ecuación de la forma $f(x) = 0$ y grafique la función en el intervalo $[0; 20]$ de manera tal que pueda identificar todas las soluciones de la ecuación en el intervalo.

Respuesta:

```
 # Ingrese en este bloque todo el código necesario para resolver el ejercicio 1.2
f <- function(x){
  return(2*cos(x)-exp(-10/x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(0, 20, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
```

```r
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Grafico
gg_fx
```



1.3 Hallar raíces

Utilizando el algoritmo del punto 1.1, halle todas las raíces identificadas en el punto 1.2.

[***OBSERVACIÓN***: Debe usar el algoritmo dado en 1.1 (corregido por usted). No puede usar su propio algoritmo.]

Respuesta:

```r
# Ingrese en este bloque todo el código necesario para resolver el ejercicio 1.3
Secante(f = f(0), p0 = 0, p1 = 2.5, TOL = 0.00001, N = 100)
```

```
## [1] 1.56994
```

```r
Secante(f = f(4), p0 = 4, p1 = 5, TOL = 0.00001, N = 100)
```

```
## [1] 4.773982
```

```r
Secante(f = f(7.5), p0 = 7.5, p1 = 8, TOL = 0.00001, N = 100)
```

```
## [1] 7.716724
```

```r
Secante(f = f(10), p0 = 10, p1 = 11, TOL = 0.00001, N = 100)
```

```
## [1] 11.2018
```

```r
Secante(f = f(13), p0 = 13, p1 = 15, TOL = 0.00001, N = 100)
```

```
## [1] 13.89129
```

```r
Secante(f = f(17), p0 = 17, p1 = 18, TOL = 0.00001, N = 100)
```

```
## [1] 17.56564
```

1.4 Iteraciones

Tome el algoritmo del punto 1.1 (copie y pegue) y agregue las líneas de código que considere necesarias para poder visualizar (imprimir) cada iteración del algoritmo. Una vez editado el algoritmo, imprima 9 iteraciones del algoritmo iniciando en $x_0 = 15.78$ y $x_1 = 16.78$. ¿A cuál de las raíces convergeria el algortimo en este caso?

Respuesta:

```r
# Ingrese en este bloque todo el código necesario para resolver el ejercicio 1.4
f <- function(x){
  return(2*cos(x)-exp(-10/x))
}

Secante <- function(f,p0,p1,TOL,N){
    i <- 2
    q0 <- f(p0)
    q1 <- f(p1)
    while (i <= N){ #Cambio el == por <=
      p = p1 - q1*(p1-p0)/(q1-q0) #Modifico (q1-q0)/(p1-p0)
      if (abs(p-p1) < TOL){ #q-q1
        return(p)
      }
      i = i + 1
      p0 = p1 #p
      q0 = q1
      p1 = p #p0
      q1 = f(p) #f(q0)
      if(i <= 7){
        print(p)
      }
    }
    return(paste('El metodo fallo luego de ', N, ' iteraciones')) #n por N
```

```r
}
Secante(f = f(16.25), p0 = 16.25, p1 = 17.25, TOL = 0.00001, N = 100)
```

```
## [1] 17.62745
## [1] 17.56764
## [1] 17.56562
## [1] 17.56564
```

```
## [1] 17.56564
```

El método converge antes de las 7 iteraciones

2 Resolución de Ecuaciones: Punto Fijo. (24 puntos)

Para este ejercicio, considere la función $h(x) = x^2 \times cos(x)$.

2.1 Teoría

Describa el método de punto fijo que se utiliza para hallar raíces de funciones.

Respuesta (escriba a continuación): El método de punto fijo realiza una primera aproximación en (p0, f(p0)). Luego se toma p1 como el punto donde f(p0) cruza a la recta y = x. Ahora se toma el punto (p1, f(p1)), y asi sucesivamente. El método converge en forma de "telaraña" en sentido de la raiz.

2.2 Hallar raíces

Halle todas las raíces de $h(x)$ en el intervalo $[-5, 5]$ utilizando el método de ***Punto Fijo***.

[***OBSERVACIÓN***: Recuerde hallar en primer lugar la(s) función(es) $g(x)$ y chequee el cumplimiento de las condiciones de existencia.]

Respuesta:

```r
 # Ingrese en este bloque todo el código necesario para resolver el ejercicio 2.2

g <- function(x){
return(x-x^2*cos(x))
}

#Método de punto fijo
PuntoFijo <- function(p0, n = 100, tol){
  #Donde p0 es la aproximación inicial
  #El número máximo de iteraciones n viene por default en 100
  #Y tol es la toleranacia al error

  #Instancio las listas vacias
  lista_p <- c(NULL)
  lista_gp <- c(NULL)

  for (i in 1:n) {
    #Calculo p
    p <- g(p0)

    lista_p[i] <- p0
    lista_gp[i] <- p

    if(abs(p-p0) <= tol){
      #Creo un data frame con las listas
      datos <- data.frame(lista_p, lista_gp)
      colnames(datos) <- c("P", "G(P)")
      print(datos)
      return(p)
    }

    p0 <- p
  }

  #En el caso de que falle el método
```

```r
  return(paste('El método falla luego de: ', n, ' iteraciones'))
}

#Pruebo el cumplimiento del teorema de existencia.
#Se observa que la función es continua

gprima <- D(expression(-x+x^2*cos(x)), "x")

gprima
```

```
## 2 * x * cos(x) - x^2 * sin(x) - 1
```

```r
gprima <- function(x){
  return(-1 + 2 * x * cos(x) - x^2 * sin(x))
}

# Intervalo [-5;-4.5]
x <- seq(-5, -4.5, by =0.1)

#Genero los puntos
gprimax <- gprima(x)

#Creo un data frame con los x e y
df <- data.frame(x, gprimax)

#Lo imprimo
df
```

```
##      x   gprimax
## 1 -5.0 -27.80973
## 2 -4.9 -26.41651
## 3 -4.8 -24.79162
## 4 -4.7 -22.97185
## 5 -4.6 -20.99470
## 6 -4.5 -18.89782
```

```r
# Intervalo [-2.5;-1.5]
x <- seq(-2, -1.5, by =0.1)

#Genero los puntos
gprimax <- gprima(x)

#Creo un data frame con los x e y
df <- data.frame(x, gprimax)

#Lo imprimo
df
```

```
##      x  gprimax
## 1 -2.0 4.301777
## 2 -1.9 3.644644
## 3 -1.8 2.973194
## 4 -1.7 2.303983
## 5 -1.6 1.652347
## 6 -1.5 1.032152
```

```r
# Intervalo [-0.1;0.1]
x <- seq(-0.1, 0.1, by =0.01)

#Genero los puntos
gprimax <- gprima(x)

#Creo un data frame con los x e y
df <- data.frame(x, gprimax)

#Lo imprimo
df
```

```
##        x    gprimax
## 1  -0.10 -1.1980025
## 2  -0.09 -1.1785435
## 3  -0.08 -1.1589768
## 4  -0.07 -1.1393144
## 5  -0.06 -1.1195682
## 6  -0.05 -1.0997501
## 7  -0.04 -1.0798720
## 8  -0.03 -1.0599460
## 9  -0.02 -1.0399840
## 10 -0.01 -1.0199980
## 11  0.00 -1.0000000
## 12  0.01 -0.9800020
## 13  0.02 -0.9600160
## 14  0.03 -0.9400540
## 15  0.04 -0.9201280
## 16  0.05 -0.9002499
## 17  0.06 -0.8804318
## 18  0.07 -0.8606856
## 19  0.08 -0.8410232
## 20  0.09 -0.8214565
## 21  0.10 -0.8019975
```

```r
# Intervalo [1.5;2]
x <- seq(1.5, 2, by =0.1)

#Genero los puntos
gprimax <- gprima(x)

#Creo un data frame con los x e y
df <- data.frame(x, gprimax)

#Lo imprimo
df
```

```
##     x   gprimax
## 1 1.5 -3.032152
## 2 1.6 -3.652347
## 3 1.7 -4.303983
## 4 1.8 -4.973194
## 5 1.9 -5.644644
## 6 2.0 -6.301777
```

```r
# Intervalo [4.5;5]
x <- seq(4.5, 5, by =0.1)

#Genero los puntos
gprimax <- gprima(x)

#Creo un data frame con los x e y
df <- data.frame(x, gprimax)

#Lo imprimo
df
```

```
##     x  gprimax
## 1 4.5 16.89782
## 2 4.6 18.99470
## 3 4.7 20.97185
## 4 4.8 22.79162
## 5 4.9 24.41651
## 6 5.0 25.80973
```

```r
raiz <- PuntoFijo(p0 = 0, tol = 0.00001)
```

```
##   P G(P)
## 1 0    0
```

```r
print(paste("La raiz es: ", raiz))
```

```
## [1] "La raiz es:  0"
```

Se observa que no cumple con:

$$|g'(x)| \leq 1$$

En la raiz ubicada en los intervalos $[-5;-4.5]$, $[-2;-1.5]$, $[1.5;2]$, $[4.5;5]$

2.3 Iteraciones

Considere el punto anterior. Realice 8 iteraciones del algoritmo de Punto Fijo, utilizando en $x_0 = -0.5$. ¿A cuál de las raíces convergeria el algortimo en este caso?

Respuesta:

```r
 # Ingrese en este bloque todo el código necesario para resolver el ejercicio 2.3
```

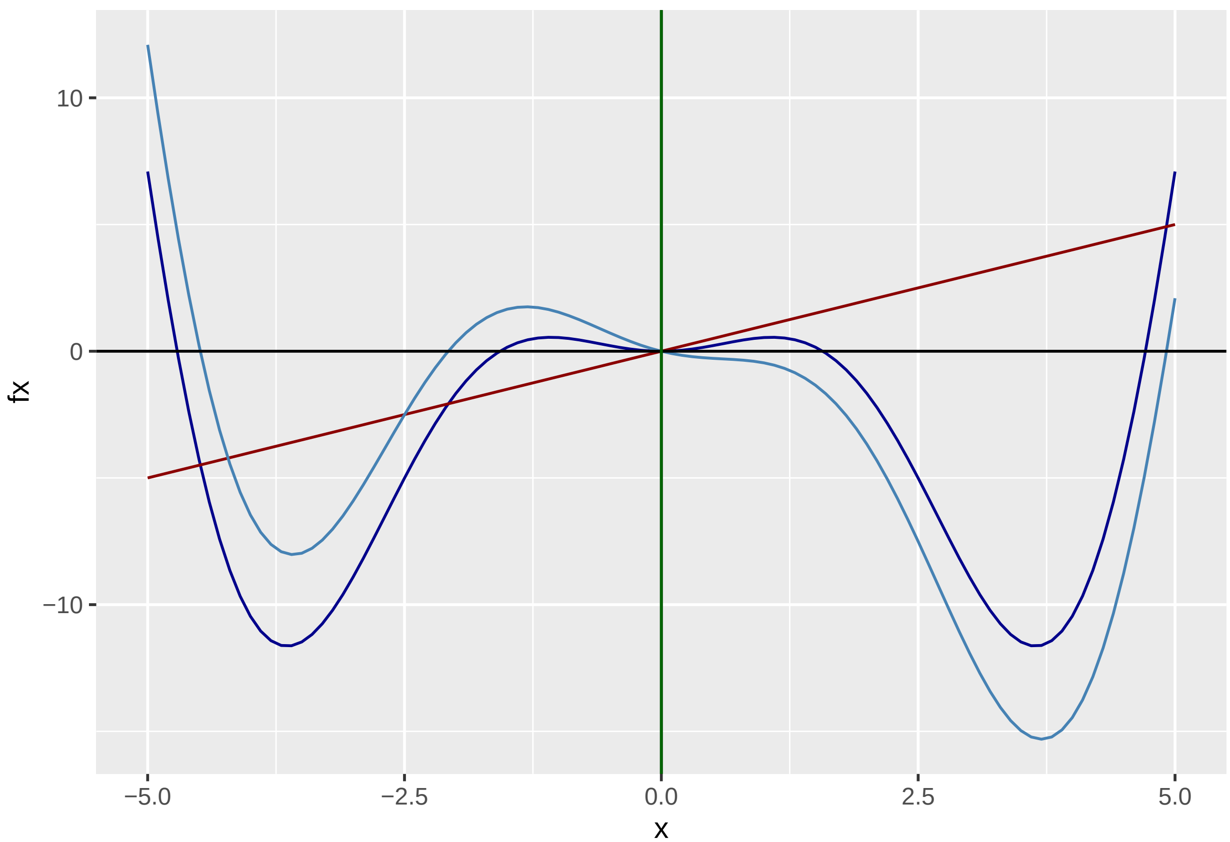
2.4 Graficar $g(x)$ y marcar sus puntos fijos

Grafique la función o las funciones $g(x)$ (que utilizó para hallar las raíces de $h(x)$) e identifique los ***puntos fijos*** hallados en el punto 2.2. En el gráfico (o en cada gráfico), marque las coordenadas de cada punto fijo $g(x) = x$ con un punto de color rojo.

Respuesta:

```r
 # Ingrese en este bloque todo el código necesario para resolver el ejercicio 2.4
f <- function(x){
return(x^2*cos(x))
}

#La función para graficar la raiz
```

```
g <- function(x){
return(-x+x^2*cos(x))
}

#Instancio un vector que me va a indicar los puntos en la función
x <- seq(-5, 5, by = 0.1)

#Genero los puntos
fx <- f(x)

#Creo un data frame con los x e y
df <- data.frame(x, fx)

#Instancio los datos
gg_fx <- ggplot(data = df)

#Agrego la capa con los datos
gg_fx <- gg_fx + aes(x = x, y = fx)

#Est grafica una linea
gg_fx <- gg_fx + geom_line(linetype = 1, colour = "darkblue")

#Gráfico x = y
gg_fx <- gg_fx + geom_line(aes(y = x), colour = "darkred")

#Gráfico la función del ejercicio donde esta la raiz
gg_fx <- gg_fx + geom_line(aes(x = x, y = g(x)), colour = "steelblue")

#Agrego el eje X
gg_fx <- gg_fx + geom_vline(xintercept = 0, linetype = 1)

#Agrego el eje Y
gg_fx <- gg_fx + geom_hline(yintercept = 0, linetype = 1)

#Agrego la linea recta que une los puntos entre la función del ejercicio, el punto fijo y la raiz
gg_fx <- gg_fx + geom_vline(xintercept = raiz, linetype = 1, colour="darkgreen")

#Grafico
gg_fx
```


10
0
−10
fx
−5.0
−2.5
0.0
2.5
5.0
x

3 Factorización de Matrices. (12 puntos)

3.1 Factorización de Cholesky

Realice la factorización de Cholesky de la siguiente matriz:

$$A = \begin{bmatrix} 1 & 0.0481 & 0.0105 \\ 0.0481 & 1 & 0.0116 \\ 0.0105 & 0.0116 & 1 \end{bmatrix}$$

Respuesta:

```
# Ingrese en este bloque todo el código necesario para resolver el ejercicio 3.1
Cholesky <- function(A){
 n <- nrow(A)

 L <- matrix(rep(0, times = n^2), nrow = n, ncol = n, byrow = TRUE)

 # Paso 1 -----------------------------------------------------------------
 L[1,1] <- sqrt(A[1,1])


 # Paso 2 -----------------------------------------------------------------
 for (j in 2:n){
   L[j,1] <- A[j,1]/L[1,1]
 }


 # Paso 3 -----------------------------------------------------------------
 for (i in 2:(n-1)) {

   # Paso 4 ---------------------------------------------------------------
   suma <- 0
   for (k in 1:(i-1)) {
     suma <- suma + L[i,k]^2
   }

   L[i,i] <- sqrt(A[i,i] - suma)


   # Paso 5 ---------------------------------------------------------------
   for (j in (i+1):n) {

     suma <- 0
     for (k in 1:(i-1)) {
       suma <- suma + L[j,k]*L[i,k]
     }

     L[j,i] <- (A[j,i] - suma)/L[i,i]

   }
 }

 # Paso 6 -----------------------------------------------------------------
 suma <- 0
```

```r
  for (k in 1:(n-1)) {
    suma <- suma + L[n,k]^2
  }

  L[n,n] <- sqrt(A[n,n]-suma)

  return(L)

}

A <- matrix(c(1, -0.0126, 0.012,
              -0.0126, 1, 0.0206,
              0.012, 0.206, 1), nrow = 3, ncol = 3, byrow = TRUE)

test <- Cholesky(A)

print(test)
```

```
##         [,1]      [,2]      [,3]
## [1,]  1.0000 0.0000000 0.0000000
## [2,] -0.0126 0.9999206 0.0000000
## [3,]  0.0120 0.2061676 0.9784431
```

```r
print(test%*%t(test))
```

```
##         [,1]    [,2]  [,3]
## [1,]  1.0000 -0.0126 0.012
## [2,] -0.0126  1.0000 0.206
## [3,]  0.0120  0.2060 1.000
```

3.2 Factorización LU

Realice la factorización de LU de la siguiente matriz:

$$B = \begin{bmatrix} 20 & 18 & 16 & 18 \\ 16 & 15 & 29 & 18 \\ 19 & 21 & 15 & 26 \\ 19 & 3 & 16 & 19 \end{bmatrix}$$

Respuesta:

```r
 # Ingrese en este bloque todo el código necesario para resolver el ejercicio 3.2
  LU <- function(matriz_coeficientes){
  n_incognitas = nrow(matriz_coeficientes)

  L <- matrix(rep(0, times = n_incognitas^2), nrow = n_incognitas, ncol = n_incognitas, byrow = TRUE)
  U <- matrix(rep(0, times = n_incognitas^2), nrow = n_incognitas, ncol = n_incognitas, byrow = TRUE)

  # -------- PASO 1
  for (i in 1:(n_incognitas-1)) {
    #Completo con 1 la diagonal principal de L.
    L[i,i] <- 1

    for (j in (i+1):n_incognitas) {
      L[i,j] <- 0
```

```r
  }
}

#El último lo completo a mano
L[n_incognitas, n_incognitas] <- 1

U[1,1] <- matriz_coeficientes[1,1]


if(U[1,1] == 0){
  return("factorizacion imposible")
}

# -------- PASO 2

for (j in 2:n_incognitas) {
  U[1,j] <- matriz_coeficientes[1,j]
  L[j,1] <- matriz_coeficientes[j,1]/U[1,1]
}


# -------- PASO 3

for(i in 2:(n_incognitas-1)){

  # -------- PASO 4
  suma <- 0
  for (k in 1:(i-1)) {
    suma <- suma + L[i,k]*U[k,i]
  }

  U[i,i] <- matriz_coeficientes[i,i] - suma


  if(U[i,i] == 0){
    return("factorizacion imposible")
  }

  # -------- PASO 5

  for (j in (i+1):n_incognitas) {
    sumaU <- 0
    sumaL <- 0
    for (k in 1:(i-1)) {
      sumaU <- sumaU + L[i,k]*U[k,j]
      sumaL <- sumaL + L[j,k]*U[k,i]
    }

    U[i,j] <- (1/L[i,i])* (matriz_coeficientes[i,j] - sumaU)
    L[j,i] <- (1/U[i,i])* (matriz_coeficientes[j,i] - sumaL)

  }
```

```r
  }


  # -------- PASO 6


  suma <- 0
  for (k in 1:(n_incognitas-1)) {
    suma <- suma + L[n_incognitas,k]*U[k,n_incognitas]
  }
  U[n_incognitas, n_incognitas] <- matriz_coeficientes[n_incognitas,n_incognitas] - suma


  # -------- PASO 7
  return(list("L" = L, "U" = U))
}

A <- matrix(c(17, 15, 30, 20,
              22, 23, 27, 8,
              13, 19, 22, 20,
              14, 18, 14, 27), nrow = 4, ncol = 4, byrow = TRUE)

LU(matriz_coeficientes = A)
```

```
## $L
##           [,1]     [,2]     [,3] [,4]
## [1,] 1.0000000 0.000000 0.000000    0
## [2,] 1.2941176 1.000000 0.000000    0
## [3,] 0.7647059 2.098361 1.000000    0
## [4,] 0.8235294 1.573770 0.331044    1
##
## $U
##      [,1]      [,2]      [,3]      [,4]
## [1,]   17 15.000000  30.00000  20.00000
## [2,]    0  3.588235 -11.82353 -17.88235
## [3,]    0  0.000000  23.86885  42.22951
## [4,]    0  0.000000   0.00000  24.69231
```

```r
L <- LU(A)$L
U <- LU(A)$U

L%*%U
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   17   15   30   20
## [2,]   22   23   27    8
## [3,]   13   19   22   20
## [4,]   14   18   14   27
```

4 Interpolación (40 Puntos)

Considere la siguente tabla de datos:

x	y
0.0	0.0000
0.1	0.3205
0.2	0.4175
0.3	0.4916
0.4	0.5555
0.5	0.6143
0.6	0.6708
0.7	0.7276
0.8	0.7877
0.9	0.8574
1.0	1.0000

4.1 Polinomio de Newton: $P_N(x)$

Escriba el Polinomio interpolante de Newton, $P_n(x)$, que pasa por todos los puntos dados.

Respuesta:

```
# Ingrese en este bloque todo el código necesario para resolver el ejercicio 4.1
```

4.2 Interpolar con $P_N(x)$

Calcule $P_N(0.5213)$.

Respuesta:

```
# Ingrese en este bloque todo el código necesario para resolver el ejercicio 4.2
```

4.3 Cubic Splines: $S_i(x)$

Escriba los trazadores cúbicos, $S_i(x); i = 1, ..., n$ que pasan por todos los puntos dados.Indique claramente qué polinomio S_i se debe usar en cada intervalo de x.

Respuesta:

```
# Ingrese en este bloque todo el código necesario para resolver el ejercicio 4.3
```

4.4 Interpolar con $S_i(x)$

Usando los trazadores cúbicos, interpole los datos para el valor $x = 0.5213$.

Respuesta:

```
# Ingrese en este bloque todo el código necesario para resolver el ejercicio 4.4
```

4.5 Graficar

Grafique lo siguiente:

a. Datos dados en la tabla mediante puntos (círculos, rellenos o no).
b. Línea continua de color verde con la función $P_N(x)$ para x en $[0; 1]$.
c. Línea continua de color azul con los trazadores cúbicos para x en $[0; 1]$.

Respuesta:

```
# Ingrese en este bloque todo el código necesario para resolver el ejercicio 4.5
```

4.6 Comentar Resultados

A partir de lo hallado en el punto anterior, comente sobre las diferencias entre los métodos para realizar aproximaciones de la función entre los puntos dados.

Respuesta (escriba sus comentarios a continuación):