Pontificia Universidad Javeriana

Taller N.1 análisis numérico

Juan Felipe Vanegas

Diego Mateus Cruz

Solución 1 punto (a)

```
using namespace std;
#include <iostream>
#include <conio.h>
#include <math.h>

main ( )
 {
    int n, i, m, k, sumas,multi;
    double t;
    sumas = 0;
    multi = 0;

   cout << "\n\n\t\t METODO DE HORNER PARA EVALUAR POLINOMIOS ";
   cout << "\n\n\t Ingrese el grado del polinomio ";
   cin >> n;

   int a[n], b[n];
   cout << "\n\n Ingrese los coeficientes con su signo correspondiente \n";
   for(i=0; i<=n; i++)
   {
        m = n-i;
        cout << "\n a("<< m <<") : > ";
        cin >> a[n-i];
```

```
    }

    cout << "\n\n ingrese el polinomio: \n\n P(x) = ";
    for(i=0; i<=n; i++)
    {
        m=n-i;
        if(i!=n)
        {
            cout << " " << a[m] << " x' " << m << " + ";
         }
        else
         {
           cout << " " << a[m] << " ";
          }
     }

cout << "\n\n\n Coloque el valor para evaluar el P(x): ";
cin >> t;

b[n] = a[n];
for(k=(n-1); k>=0; k--)
{
    b[k] = t*b[k+1]+a[k];
    cout<<"b["<<k<<"] = "<<t<<" * "<<b[k+1]<<" + "<<a[k]<<endl;
     cout<<"b["<<k<<"] = "<<b[k]<<endl;
    if(a[k] != 0)
    {
       sumas++;
```

```
        }
        multi++;
    }
    cout<<"procesos: "<<sumas<<" sumas y "<<multi<<" multiplicaciones."<<endl;
    cout << "\n\n\t Resultado:  " << b[0]<<endl;
    cout << endl << endl;
    system("PAUSE");
    return 0;
```

Solución 4 punto

Código de grafico en coordenadas polares

```
polar <- function (theta, r, color=4){
  y <- 0
  x <- 0
  ejex <- 1

  for (i in 1:length(r)){
    if(is.nan(r[i])== T){
      r[i] <- 0
    }
  }

  angulo <- seq(-max(theta),max(theta),by=theta[2]-theta[1])
  y <- r*sin(theta)
  x <- r*cos(theta)
  plot.new()
  plot.window(xlim = c(-max(r), max(r)), ylim = c(-max(r), max(r)), asp = 1)

  aux <- max(r)
```

```
  # Dibuja los ejes.
  while (aux > 0){
    fi <- aux*sin(angulo)
    cir <- aux*cos(angulo)
    points(cir,fi,pch="-",col="gray",cex=0.3)
    text(ejex+0.2,-0.2,ejex,col="gray")
    ejex <- ejex + 1
    aux <- aux - 1
  }

  abline(v=((max(cir)+min(cir))/2),col="gray")
  abline(h=((max(cir)+min(cir))/2),col="gray")
  segments(-max(r)+0.5,-max(r)+0.5,max(r)-0.5,max(r)-0.5,col="gray")
  segments(-max(r)+0.5,max(r)-0.5,max(r)-0.5,-max(r)+0.5,col="gray")

  points(x,y,pch=20,col=color,cex=1)
}
e = 2.71828182845904523536028747135 27
dim <- seq(-pi, pi, by=pi/300)
r=cos(3*dim) +e^dim
polar(dim,r,"blue")
```

Código de método de newton

```
newtonDN = function(f, x0, tol, maxiter){
  # Derivada numérica con diferencia central
  fp = function(x) { h = 1e-15
  (f(x+h) - f(x-h)) / (2*h)
  }
  k = 0
```

```
  #Par imprimir estado
  cat("------------------------------------------------------------------------\n")
  cat(formatC( c("x_k"," f(x_k)","Error est."), width = -20, format = "f", flag = " "), "\
     n")
  cat("------------------------------------------------------------------------\n")
  repeat{
    correccion = f(x0)/fp(x0)
    x1 = x0 - correccion
    dx = abs(correccion)
    # Imprimir iteraciones
    cat(formatC( c(x1 ,f(x1), dx), digits=15, width = -15, format = "f", flag = " "), "\n")
    x0 = x1
    k = k+1
    # until
    if(dx <= tol || k > maxiter ) break;
  }
  cat("------------------------------------------------------------------------\n")
  if(k > maxiter){
    cat("Se alcanzó el máximo número de iteraciones.\n")
    cat("k = ", k, "Estado: x = ", x1, "Error estimado <= ", correccion)
  } else {
    cat("k = ", k, " x = ", x1, " f(x) = ", f(x1), " Error estimado <= ", correccion) }
}
## --- Pruebas
e = 2.71828182845904523536028747135 27
f = function(x) 2+ cos(3*x)  # en esta parte se ingresa la función a resolver iterativamente
options(digits = 15)
newtonDN(f, 0.5, 1e-10, 10)
```

Código método de punto fijo

```
puntofijo =function(g, x0, tol=1e-9, maxIter=100){
  k = 1
  # iteración hasta que abs(x1 - x0) <= tol o se alcance maxIteraciones
  repeat{
    x1 = g(x0)
    dx = abs(x1 - x0)
    x0 = x1
    #Imprimir estado
    cat("x_", k, "= ", x1, "\n")
    k = k+1
    #until
    if(dx< tol|| k > maxIter) break;
  }
  # Mensaje de salida
  if( dx > tol ){
    cat("No hubo convergencia ")
    #return(NULL)
  } else{
    cat("x* es aproximadamente ", x1, " con error menor que ", tol)
  }
}
e = 2.71828182845904523536028747135 27
g = function(x) 2 - e^x # En esta parte se ingresa la función a resolver iterativamente
puntofijo(g, 0.5, 1e-9)
```

Código de graficación en coordenadas polares

```
polar <- function (theta, r, color=4){
  y <- 0
  x <- 0
  ejex <- 1

  for (i in 1:length(r)){
    if(is.nan(r[i])== T){
      r[i] <- 0
    }
  }

  angulo <- seq(-max(theta),max(theta),by=theta[2]-theta[1])
  y <- r*sin(theta)
  x <- r*cos(theta)
  plot.new()
  plot.window(xlim = c(-max(r), max(r)), ylim = c(-max(r), max(r)), asp = 1)

  aux <- max(r)
  # Dibuja los ejes.
  while (aux > 0){
    fi <- aux*sin(angulo)
    cir <- aux*cos(angulo)
    points(cir,fi,pch="-",col="gray",cex=0.3)
    text(ejex+0.2,-0.2,ejex,col="gray")
    ejex <- ejex + 1
    aux <- aux - 1
  }
```

```
  abline(v=((max(cir)+min(cir))/2),col="gray")
  abline(h=((max(cir)+min(cir))/2),col="gray")
  segments(-max(r)+0.5,-max(r)+0.5,max(r)-0.5,max(r)-0.5,col="gray")
  segments(-max(r)+0.5,max(r)-0.5,max(r)-0.5,-max(r)+0.5,col="gray")

  points(x,y,pch=20,col=color,cex=1)
}
e = 2.71828182845904523536028747135 27
dim <- seq(-pi, pi, by=pi/300)
r=cos(3*dim) +e^dim
polar(dim,r,"blue")
```

Punto 5

```
t = function(maxiter = 100, x){
  n = 0;
  while(1.0+(maxiter*0.5) > 1.0){
    n=n+1
    maxiter = maxiter*0.5
  }
  cat("Epsilon de la maquina en forma hexadecimal = ", maxiter, "\n")
  cat("Epsilon de la maquina en forma binaria = 2^-", n, "\n")
  suma = 1
  i = 1
  while(i<=1000){
    suma = suma + x
    i = i + 1
  }
  maxiter = 10000 * x + 1
```

```
  cat("El error acumulado es  = ", suma, "\n")
}
t(1.0,0.4)
```

Punto 6

14.

```
def poli(x):
    funcion = pow(x, 2) - (3 * x) - 4
    return (funcion)

#MAIN PROGRAM
a = float(input('Por favor introducir valor de limite innferior(a): '))
b = float(input('Por favor introducir valor de limite superior(b): '))
E = float(input('Por favpr ingrese la precision (E): '))

x1 = a
#Valor a incrementar
d = (b-a)/10
x0 = x1
x1 = x1 + d
noRaiz = False

while abs(d) >= E:
    if poli(x0)*poli(x1) < 0:
        x1 = x1 - d
        d = d/10
        x1 = x1 + d
    else:
        x0 = x1
```

```
        x1 = x1 + d
    if x1 > b:
        print('X1: ', x1, 'b: ', b)
        print('La funcion ha superado el limite superior establecido en el rango')
        noRaiz = True
        break

print('La raiz es :', x1)

#ANALISIS:
#a) El intervalo escogido debe ser cercano a la raiz
# el limite superior e inferor no deben ser tan cercanas a la raiz
#b) Exponencial
```