Practical 9

Harshal Marathe

2023-11-25

#Q1  
#function to define the equation  
f <- function(x){  
 return(x^3-2\*x-5)  
}  
#Bisection Method  
bisection <- function(f,a,b,tol=1e-6){  
 iter <- 1  
 while((b-a)>tol){  
 c <- (a+b)/2  
 if(f(c)==0){  
 break  
 }else if (f(c)\*f(a)<0){  
 b <- c  
 }else{  
 a <- c  
 }  
 iter <- iter+1  
 }  
 root <- (a+b)/2  
 return(root)  
}  
#initial interval[a,b]  
a <- 1;b <- 3;  
  
#initial solution   
x0=5  
root <- bisection(f,1,3)  
#print the result  
cat("the root of the equation x^2-4x-7=0 with initial interval a=",a," and b=",b," is ",root)

## the root of the equation x^2-4x-7=0 with initial interval a= 1 and b= 3 is 2.094552

#Q2  
f <- function(x){  
 return(x^2- 4\*x -7)  
}  
#derivative of the function  
f\_1 <- function(x){  
 return(2\*x-4)  
}  
#Newton Raphson method function  
newton\_raphson <- function(f,f\_1,x0,tol=1e-6){  
 iter <- 1  
 x <- x0  
 while(abs(f(x))>tol){  
 x=x-f(x)/f\_1(x)  
 iter <- iter+1  
 }  
 return(x)  
}  
#Initial solution  
x0=5  
  
#call the Newton-Raphson method function  
root <- newton\_raphson(f,f\_1,x0)  
  
cat("the root of the equation x^2 - 4x -7 = 0 with initial solution x=",x0,"is :",root,"\n")

## the root of the equation x^2 - 4x -7 = 0 with initial solution x= 5 is : 5.316625

#Q3  
lagrange\_interpolation <- function(x,x\_val, y\_val) {  
 n <- length(x\_val)  
 result <- 0  
   
 for (i in 1:n) {  
 term <- y\_val[i]  
 for (j in 1:n) {  
 if (j != i) {  
 term <- term \* (x - x\_val[j]) / (x\_val[i] - x\_val[j])  
 }  
 }  
 result <- result + term  
 }  
   
 return(result)  
}  
  
# Example usage:  
# Given points (x, y)  
x <- c(5,7,11,13)  
y <- c(150,392,1452,2366)  
  
# Point at which to interpolate  
interpolation\_point <- 9  
  
# Calculate the interpolated value at the specified point  
result <- lagrange\_interpolation(interpolation\_point,x, y)  
# Print the result  
cat("Interpolated value at x =", interpolation\_point, "is", result, "\n")

## Interpolated value at x = 9 is 810

#Q4  
A=matrix(c(4,1,1,1,5,2,1,2,3),nrow=3,byrow=T)  
b=matrix(c(2,-6,-4,nrow=3))  
x0=c(0,0,0)  
x1=c(0,0,0)  
err=1  
while(err>0.00001)  
{  
 x1[1]=(b[1]-A[1,2]\*x0[2]-A[1,3]\*x0[3])/A[1,1]  
 x1[2]=(b[2]-A[2,1]\*x0[1]-A[2,3]\*x0[3])/A[2,2]  
 x1[3]=(b[3]-A[3,1]\*x0[1]-A[3,2]\*x0[2])/A[3,3]  
 err=max(abs(x0-x1))  
 x0=x1  
 print(x0)  
}

## [1] 0.500000 -1.200000 -1.333333  
## [1] 1.1333333 -0.7666667 -0.7000000  
## [1] 0.8666667 -1.1466667 -1.2000000  
## [1] 1.0866667 -0.8933333 -0.8577778  
## [1] 0.9377778 -1.0742222 -1.1000000  
## [1] 1.0435556 -0.9475556 -0.9297778  
## [1] 0.9693333 -1.0368000 -1.0494815  
## [1] 1.0215704 -0.9740741 -0.9652444  
## [1] 0.9848296 -1.0182163 -1.0244741  
## [1] 1.0106726 -0.9871763 -0.9827990  
## [1] 0.9924938 -1.0090149 -1.0121067  
## [1] 1.0052804 -0.9936561 -0.9914880  
## [1] 0.996286 -1.004461 -1.005989  
## [1] 1.0026126 -0.9968614 -0.9957881  
## [1] 0.9981624 -1.0022073 -1.0029632  
## [1] 1.0012926 -0.9984472 -0.9979159  
## [1] 0.9990908 -1.0010921 -1.0014661  
## [1] 1.0006396 -0.9992317 -0.9989688  
## [1] 0.9995501 -1.0005404 -1.0007254  
## [1] 1.0003164 -0.9996199 -0.9994898  
## [1] 0.9997774 -1.0002674 -1.0003589  
## [1] 1.0001566 -0.9998119 -0.9997476  
## [1] 0.9998899 -1.0001323 -1.0001776  
## [1] 1.0000775 -0.9999069 -0.9998751  
## [1] 0.9999455 -1.0000655 -1.0000879  
## [1] 1.0000383 -0.9999540 -0.9999382  
## [1] 0.999973 -1.000032 -1.000043  
## [1] 1.0000190 -0.9999772 -0.9999694  
## [1] 0.9999867 -1.0000160 -1.0000215  
## [1] 1.0000094 -0.9999887 -0.9999849  
## [1] 0.9999934 -1.0000079 -1.0000106  
## [1] 1.0000046 -0.9999944 -0.9999925  
## [1] 0.9999967 -1.0000039 -1.0000053  
## [1] 1.0000023 -0.9999972 -0.9999963

x0

## [1] 1.0000023 -0.9999972 -0.9999963

#Q5  
# Function to integrate  
f <- function(x) {1/(1 + x)}  
# Trapezoidal rule for definite integral  
trapezoidal\_rule <- function(f, a, b, n) {  
 h <- (b - a)/n  
 x <- seq(a, b, length.out = n + 1)  
 result <- (h/2) \* (f(a) + 2 \* sum(f(x[2:n])) + f(b))  
 return(result)  
}  
# Define the limits of integration  
a <- 0 ;b <- 1  
# Number of subintervals  
n\_values <- c(2, 4, 8)  
# Evaluate the integral for different numbers of subintervals  
for (n in n\_values) {  
 result <- trapezoidal\_rule(f, a, b, n)  
 cat("With", n, "subintervals, the result is:", result, "\n")  
}

## With 2 subintervals, the result is: 0.7083333   
## With 4 subintervals, the result is: 0.6970238   
## With 8 subintervals, the result is: 0.6941219

#Q6  
f=function(x){return(1/(1+x))}  
a=0;b=1;n=8;h=(b-a)/n  
x=seq(a,b,h)  
y=f(x)  
#print(y)  
s1=y[1]+y[n+1]  
s1=s1+4\*sum(y[seq(2,n,2)])+2\*sum(y[seq(3,n,2)])  
s1\*h/3

## [1] 0.6931545

#Q7  
# Simpson’s 3/8th Rule  
f=function(x){return(1/(1+x))}  
a=0;b=1;n=6;h=(b-a)/n  
x=seq(a,b,h)  
y=f(x)  
s1=y[1]+y[n+1]  
s1=s1+3\*sum(y[seq(2,n)])-sum(y[seq(4,n,3)])  
s1\*3\*h/8

## [1] 0.6931953