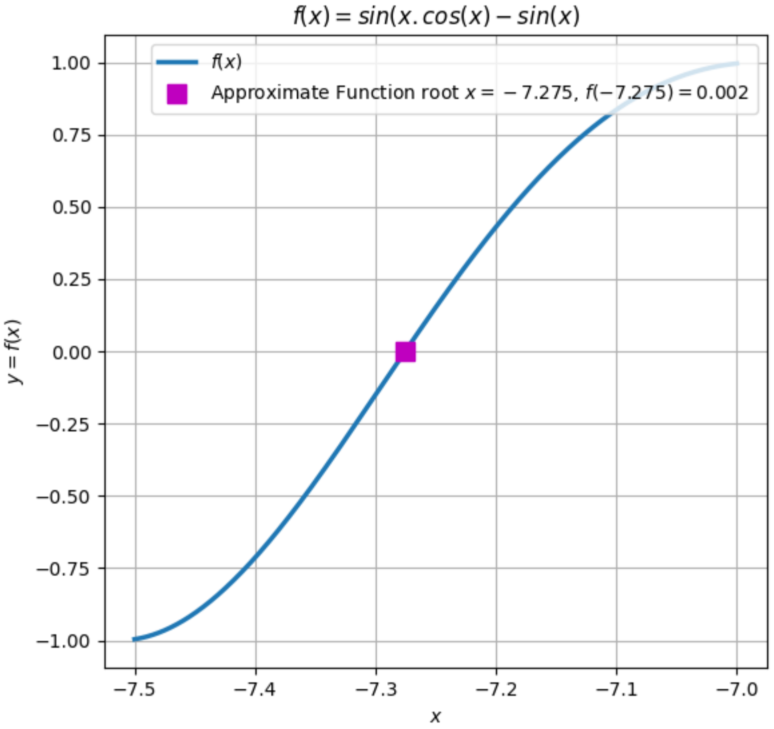
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO (UFERSA)
Programa de Pós Graduação em Engenharia Elétrica

Fundamentos de Modelagem Computacional
Atividade de Pesquisa de Raízes

Gleidson Leite da Silva

Questão 1.

A função apresenta multiplos zeros, assim, escolheu-se o zero que se apresenta entre o range de -7,5 até -7. Através da biblioteca matplotlib em conjunto com a linguagem de programção Python 3, foi possível obter a seguinte figura:



```fortran
program bissection_method
  implicit none
  real :: a, b, c, f, tol, f_a, f_b, f_c
  integer :: getNIt, i, n
  logical :: verifyInterval, isValidInterval

  print *, "Type a precision"
  read (*,*) tol
  if ( tol<=0 ) then
    print *, "You should provide a positive tolerance value"
    stop
  end if

  call getInterval(a, b)
  isValidInterval = verifyInterval(f(a), f(b))
  do while(.not.isValidInterval)
    print *, "You should provide an interval where both f(a) and f(b) have different sign"
    call getInterval(a,b)
    isValidInterval = verifyInterval(f(a), f(b))
  end do

  n = getNIt()
  do i = 1, n
    c = (a+b)/2
    f_c = f(c)
    print *, "c: ", c
    print *, "f(c) = ", f_c
    print *, "Iteraction: ", i
    if ( (abs(f_c).eq.0).or.(abs(f_c)<=tol) ) then
      print *, "--------------\\--------------"
      print *, "The found root is ", c
      print *, "f(x = root) = ", f_c
      print *, "Number of iteraction used:", i
      stop
    end if
    f_a = f(a)
    f_b = f(b)
    if ( ((f_a>0).and.(f_c>0)).or.((f_a<0).and.(f_c<0)) ) then
      a = c
    else
      b = c
    end if
  end do

  if ( f_c>tol ) then
    print *, "The simulation require more number of interactions."
    print *, " Please, restart the program and type a greater number of iteractions"
  end if

end program bissection_method

subroutine getInterval(a, b)
implicit none
real :: a, b
print *, "Type a value for 'a'"
read (*,*) a

print *, "type a value for 'b'"
```

```fortran
read (*,*) b

end subroutine getInterval

function getNIt() result(n)
  implicit none
  integer :: n
  print *, "Type a number of interactions"
  read (*,*) n
  if ( n<=0 ) then
    print *, "You should provide a number of interaction more than zero"
    stop
  end if
end function getNIt

function f(x) result(y)
  implicit none
  real, intent(in) :: x
  real :: y
  y = sin(x*cos(x)-sin(x))
end function f

function verifyInterval(f_a, f_b) result(isValid)
  implicit none
  real, intent(in) :: f_a, f_b
  logical :: isValid
  isValid = .true.
  print *, f_a, f_b
  if ( f_a.eq.0 ) then
    print *, "The interval 'a' is a root of the equation"
    stop
  end if
  if ( f_b.eq.0 ) then
    print *, "The interval 'b' is a root of the equation"
    stop
  end if
  if(((f_a>0).and.(f_b>0)).or.((f_a<0).and.(f_b<0))) then
    isValid = .false.
  end if
end function verifyInterval
```

```fortran
program secant_method
  implicit none

  real :: x0, x1, x2, tol
  real, external :: f
  integer :: nIter, i

  call getTol(tol)
  call getIter(nIter)
  call getPoints(x0, x1)

  do i = 1, nIter
    call nextX(f, x0, x1, x2)
    call testX2(f, x2, tol, i)
  end do

  if ( abs(f(x2))>tol ) then
    call error(f, x2)
  end if

end program secant_method

subroutine error(f, x2)
  implicit none
  real :: f, x2
  print *, "The program cannot reach to a root value less than tolerance specified"
  print *, "Please, try to increase the number of iteractions"
  print *, "The last value calculated was", x2
  print *, "f(x=last calculated) = ", f(x2)
end subroutine error

subroutine testX2(f, x2, tol, i)
  implicit none
  real :: f, x2, tol
  integer :: i
  if ( abs(f(x2))<=tol ) then
    print *, "The root of f(x) is ", x2
    print *, "f(x=root) = ", f(x2)
    print *, "The number of iteractions used was", i
    call exit()
  end if
end subroutine testX2

subroutine getTol(tol)
  implicit none
  real :: tol
  do while(.true.)
    print *, "Type a tolerance"
    read (*,*) tol
    if ( tol<=0 ) then
      print *, "You should provide a tolerance greather than 0"
    else
      exit
    end if
  end do
end subroutine getTol

subroutine getIter(nIter)
  implicit none
  integer :: nIter
  do while(.true.)
    print *, "Type a number of iteractions"
    read (*,*) nIter
    if ( nIter<=0 ) then
      print *, "You should provide a number of iteractions greather than 0"
    else
      exit
    end if
  end do
end subroutine getIter

subroutine getPoints(x0, x1)
  implicit none
  real :: x0, x1
  print *, 'Type a value for x0'
  read (*,*) x0
  print *, 'Type a value for x1'
  read (*,*) x1
end subroutine getPoints

subroutine nextX(f, x0, x1, x2)
  implicit none
  real :: f, x0, x1, x2
  x2 = x1-f(x1)*(x1-x0)/(f(x1)-f(x0))
  x0 = x1
  x1 = x2
end subroutine nextX

function f(x) result(y)
  implicit none
  real, intent(in) :: x
  real :: y
  y = sin(x*cos(x) - sin(x))
end function f


```