Yanes Garcia

Math 4670

Bisection and Newton's Methods

1. This Fortran program is used to solve x^3-30=0 using the method of bisection. The Bisection method is used to determine, to any specific accuracy that your computer will permit, a solution to f(x)=0 on an interval [a, b], provided that f is continuous on the interval and f(a) and f(b) are opposite sign. In this program, to begin the Bisection method, we have to set a=3.0 and b=4.0. We enter in a do-while loop that will end when f(x) will be equal 0.0. The program lets m be the midpoint of the interval [a, b]. We get the midpoint by adding a and b and dividing this result by 2.0. Depending if f(m) is bigger or smaller than 0.0, we set a=m or b=m. We care about the sing of the function on the midpoint and on both edges. Like, I declared all my variables as real values, my variables must have numbers with decimal points. Code for this is:

```
implicit none
real :: m, a, b, fx

a=3.0
b=4.0
fx=1.0

do while (fx /= 0.0)
m=(a+b)/2.0
fx=(m**3.0-30.0)

if (fx<0.0) then
a=m
end if

if (fx>0.0) then
```

```
b=m
end if

print*, 'x=', m
end do
print*, 'x=', m
stop
end
```

I'll show the output.

```
x= 3.10723
x= 3.10723
…
Infinite
```

The equation x^3-30 is solved when it is equal 0.

a. The equation will get closer to the number, which makes the function 0.0; however, it will never achieve this number. Although, we get many times the 3.10723 number, it is not always the same. This value has different decimal, which we cannot see.
b. The equation will get closer to the number, which makes the function gets 0.0; this number is around 3.10723. The Bisection method always takes so long because this method is so slow.

2. This Fortran program is used to solve problem 17 of section 2.3 in the text book using the bisection's method. The Bisection method is used to determine, to any specific accuracy that your computer will permit, a solution to f(x)=0 on an interval [a, b], provided that f is continuous on the interval and f(a) and f(b) are opposite sign. In this program, the particle starts at rest on a smooth inclined plane whose angle is changing at a constant rate. At the end of t seconds, the position of the object is given by an

equation. If we suppose the particle has moved 1.7ft in 1s. We can find, to within 10^-5, the rate w at which the angle changes. Assume that g=-32.17ft/s^2.

To begin the Bisection method, we have to set a=-1.0 and b=-0.1. We enter in a do loop that can end; for three different reasons. The first reason, it can finish after entering 1,500 times. The second motive why the do loop can finish is because the function gets equal 0.0. The third reason makes stop the do loop, if (b-a)/2 has a smaller value than the TOL value, which is 10^-5 in this case. The program lets m be the midpoint of the interval [a, b]. We get the midpoint by adding a and b and dividing this result by 2.0. Depending if FA*FM is bigger or smaller than 0.0, we set a=m or b=m. We care about the sing of the function on the midpoint and on both edges. Below of the program, we write the function, f=-32.17/(2.0*p**2.0)*((exp(p)-exp(-p))/2.0-sin(p))-1.7, and we return it. Like, I declared all my variables as real values, my variables must have numbers with decimal points.

```
implicit none
real :: a, b, TOL, m, FM, FA,f
integer::i

a=-1.0
b=-0.1
TOL=10**(-5.0)

do i=1,500
FA= f(a)
m= (a+b)/2.0
FM=f(m)

if ( FM == 0.0) then
print*, m
stop
```

```
end if

if ((b-a)/2 < TOL) then
print*, m
stop
end if

if( FA*FM > 0.0) then
a = m
FA=FM
end if

if (FA*FM < 0.0) then
b = m
end if
end do

print*, m
stop
end

function f(p)
implicit none
real:: p,f
f=-32.17/(2.0*p**2.0)*((exp(p)-exp(-p))/2.0-sin(p))-1.7
return
end
```

I'll show the output.

-0.317056

Using the Bisection method, setting a to -1.0 and b to -0.1, and having 10^-5 as tolerance value, we get m= -0.317056.

3. This Fortran program is used to solve problem 15 of section 2.4 in the text book using the Newton's method. The Newton's method use the line that best approximates the graph of the function at a point on its graph; this line is the tangent line to the graph at that point. We have to suppose that p0 is an initial approximation to the root p of the equation f(x)=0 and f' exits in an interval containing all the approximations to p. The slope of the tangent line to graph of f at the point (p0, f(p0)) is f'(p0). In this program, the player A will shut out player B in a game of racquetball with a probability equation where p denotes the probability that A will win any specific rally. We can determine, to within 10^-3, the minimal value of p that will ensure that A will shut out B in at least half the matches they play.

To begin the Newton's method, we have to set p0=0.75. We enter in a do loop that can end; for two different reasons. The first reason, it can finish after entering 1,500 times. The second motive why the do loop can finish is because the absolute value of the subtraction of p-p0 is smaller than TOL(tolerance), which is 10^-3 in this case. The program lets p be p0-f(p0)/fp(p0). The p value is written in the p0 variable until the do loop stops for one of the two reasons that I said before. The last value, which the p variable holds, is the one that the program prints. Below of the program, we write a function, its prime function and we return both of them. The function is f=((1.0+p)/2.0)*(p/(1.0-p+p**2.0))**21.0, and its prime function is fp=((p**20.0)*(1.0+p)*(441.0*((1.0/21.0)-p**2.0))+(p**21.0)*(1.0-p+p**2.0))/(2.0*(p**2.0-p+1)**22.0). Like, I declared all my variables as real values, my variables must have numbers with decimal points.

```
implicit none
real :: p0, TOL,p, fp, f
integer ::i
```

```
p0=0.75
TOL= 10**(-3.0)

do i= 1,500
p = p0-f(p0)/fp(p0)

if( abs(p-p0) &lt; TOL ) then
print*, p
stop
end if

p0=p
end do
print*, p
stop
end

function f(p)
implicit none
real:: p,f
f=((1.0+p)/2.0)*(p/(1.0-p+p**2.0))**21.0
return
end

function fp(p)
implicit none
real:: p,fp
fp=((p**20.0)*(1.0+p)*(441.0*((1.0/21.0)-p**2.0))+(p**21.0)*(1.0-
p+p**2.0))/(2.0*(p**2.0-p+1)**22.0)
return
end
```

I'll show the output.

2.11865