**Please, upload your answers in Canvas: assignment PC LAB – April 23**

Name: Vu The Doan

Student nr: 12918687

**PC4b – Roots**

**1.** The function has the following roots on the interval : with



©NvG, UvA (2018)

a. The function can be rewritten as in at least the following two ways:

Investigate the convergence of the fix-point method by plotting the derivative of for the interval Determine the derivative by using the symbolic capacities of R:

>> D(expression(acos(1/2\*sin(x))), "x")

Hence, we can define the derivative of as

dg1x<-function(x){

-(1/2 \* cos(x)/sqrt(1 - (1/2 \* sin(x))^2))

}

For , it is best to raise the number of evaluation points, e.g.  
x<-seq(-2\*pi,2\*pi,length=1e3), and also limit the range of the y-axis, e.g.  
ylim=c(-10,10). Also, plot horizontal lines at –1 and +1.

Which function do you prefer: or ?

b. Plot both functions and in one graph (x-range and y-range from –6 to 6).

In which point do they approximately intersect? To answer this question, redo the plot and limit the range on the horizontal and vertical axis to [1.0,1.2].

Is it possible to find all the roots of for the interval by the fixed-point method? Try different starting values for the fixed-point method, e.g. –5, –4, –3, …, 4 and 5.

Hint: install the package “spuRs” that is created by the authors of the book Jones et al.

Modify the code of the function fixedpoint() (see also p. 185 of Jones et al.) such that

(i) it doesn’t generate output

(ii) returns a vector that contains both the final solution [xnew] as the number of iterations [iter] if the algorithm has converged.

Sample output:

x0=-5: Root=1.10715 found in 15 iterations

x0=-4: Root=1.10715 found in 16 iterations

x0=-3: Root=1.10715 found in 17 iterations

x0=-2: Root=1.10715 found in 14 iterations

x0=-1: Root=1.10715 found in 15 iterations

x0= 0: Root=1.10715 found in 17 iterations

x0= 1: Root=1.10715 found in 15 iterations

x0= 2: Root=1.10715 found in 14 iterations

x0= 3: Root=1.10715 found in 17 iterations

x0= 4: Root=1.10715 found in 16 iterations

x0= 5: Root=1.10715 found in 15 iterations

**2.** Next, use the bisection method to approximate all roots of the function on the interval Use the function bisection from the packages spuRs and the appropriate bounding intervals.

**3.** Consider the function

a. Determine the roots of using Newton-Raphson. For this, first create a graph with x-range . For this, write two R-functions: f(x) and dfx(x) that calculate the function and its derivative .

Next, code a R-function fNR(x) that returns a vector with f(x) as first element and dfx(x) as second element. Assign

x0<-(-3:3)

root<-rep(NA,length(x0))

where x0[i] denotes the i-th starting value and root[i] is assigned the output value of the function newtonraphson from the spuRs package (see p. 187 of Jones et al.) when starting from x0[i].

Example output:

x0=-3: Root=-3.14159 found

x0=-2: Root=-1.00000 found

x0=-1: Root=-1.00000 found

x0= 0: Root= 0.00000 found

x0= 1: Root= 1.00000 found

x0= 2: Root= 1.00000 found

x0= 3: Root= 3.14159 found

b. Create a graph of for the interval

Find all roots of on the interval using

uniroot.all(dfx, c(-4,4))

from the package rootSolve.

Which roots of are found by the Newton-Raphson algorithm if the starting value is close to a root of e.g. 0.7 or 2.5? What is the reason for this behavior?

**4**. Modify the script of newtonraphson.r from the spuRs package to write the function

secant <- function(ftn, x0, x1, tol = 1e-9, max.iter = 100)

that implements the secant root-finding method:

First test your program by finding the root of the function

using and .

Next, see how the secant method performs in finding the root of

using and .

Compare its performance with that of bisection and Newton-Raphson.

**5.** Approximate the roots of the equations:

Use the R function polyroot(∙). Make sure you specify the coefficients in the correct order (see help file).

Roots

Roots