

**Linköpings universitet**

**Computer Lab Report**

**Lab 1**

**Computational statistics 732A38**

**Department of Computer and Information Science**

***Amirhossein Sadoghi amisa242@student.liu.se***

***Preliminaries***

The solutions to the assignments involving R coding should be reported in details, all R code that you write should be **included** into the report. Also, all appropriate pictures or diagrams should be included.

***Assignment 1: Be careful with ‘==’***

A pupil of a school is bad in arithmetic but good in programming. He writes a program to check if 1/3-1/4==1/12:

x1<-1/3;

x2<-1/4;

if (x1-x2==1/12){

print("Teacher said true")

} else{

print("Teacher lied")}

1. Check the result of this program. Comment why this happened.
2. Specify how the program can be modified to give a correct result

**from definition of Machine epsilon** which gives us an upper bound on the relative error due to rounding in floating point arithmetic.

and definition relative error :



we can procedure Rounding for choosing the representation of a real number in a floating point number system. Base on the different system, machine epsilon is the maximum relative error of the chosen rounding procedure which can be applied for number system and a rounding procedure.

General speaking, a floating point number system is consisted by a base, b, and by the number of fractional digits, p.

For all the numbers we can assign be − p space (with the same exponent, e,)This changes at the numbers that are perfect powers of b , this process assign space for large part of power b times larger the part of smaller one.

Based on the IEEE arithmetic standard , all floating point operations are done as if it were possible to perform the infinite-precision operation, and then, and after that the result should be rounded to a floating point number. Very complex explanation. It would be enough to say that ¼ was rounded off and 1/3 was rounded off, that’s why it’s not identical to 1/12

for instance for given operation between two known floating point number we have some approximation process which is mentioned earlier : round

x \bullet y = \mbox {round} (x \circ y)

For modify our code we apply this process : There is a problem with your code. Consider that: round(1/3)==round(1/2). This is true. Does it mean that 1/3==1/2????

x1<-1/3;

x2<-1/4;

if (round(x1-x2)==round(1/12)){

print("Teacher said true")

} else{

print("Teacher lied")}

***Assignment 2: Derivative***

A widely known way to compute the derivative of function *f(x)* in point *x* is to use

1. Write your own function computing the derivative of function *f(x)=*|*x*| in this way. Take ε=10-15
2. Compute your derivative function at point x=100000.
3. What is the value you obtained? What is the real value of the derivative? Explain the reason behind the discovered difference

myDerivative<- function(x, eps) {

n1 <- abs(x+eps)-abs(x)

tst <- n1/eps;

tst;

}

result <- myDerivative(100000,10e-16);

result

0

x<-as.numeric(1:18);

y<-as.numeric(1:18);

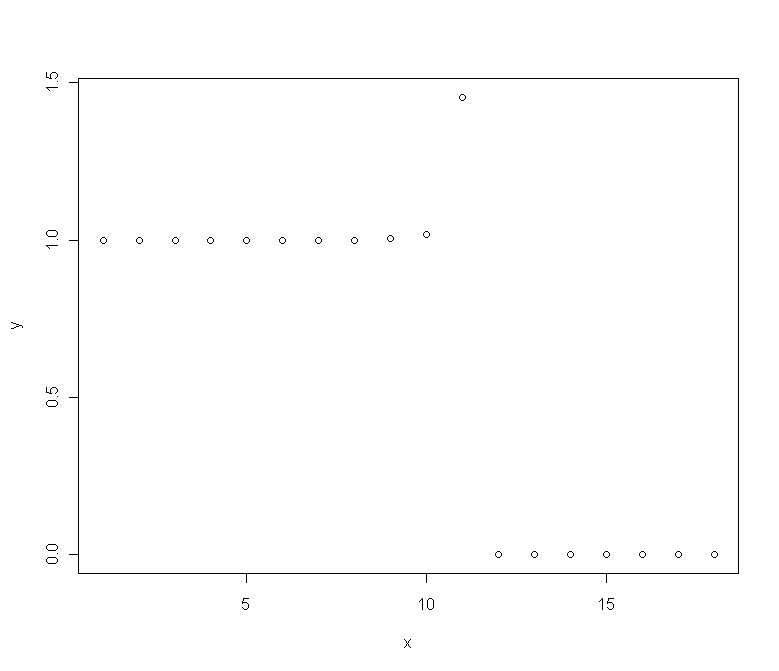
for(i in 1:18 ) {

y[i] <- myDerivative(100000,10^(-i));

x[i] <- i;

}

plot(x, y)



From definition of derivative if ε close to zero, the equation can give the derivative of function

here result with reducing until ε=10-10 can be obtained correct answer but after that it start to converge to zero (we can see from above graph ) this phenomena again can be explained from some theories which are mentioned in pervious assignment . In fact, based on different machine and its operation system , we some restrictions which are came from limitation of them ; here we can see behavior of function until such approximation (machine epsilon) limit can be follow the theory . But after that we can see some diversification from that. I can not see direct answer to the question. Need to be reformulated.

***Assignment 3: Variance***

A known formula for estimating variance is

1. Write your own function *myvar* estimating variance in this way
2. Generate vector x with 10000 random numbers, normally distributed with mean 108 and variance 1
3. For each subset Xi= {x1…xi}, i=1…10000 compute difference Yi= myvar(x)-var(x), where var(x) is a standard variance estimation function in R. Plot the dependence Yi on Xi. Draw necessary conclusions. How well your function works? What is the reason behind such behavior?

myvar <- function(y) {

n1 <- length(y);

sumsqure <- sum(y^2);

sum1 <- sum(y);

myvarVal <- (sumsqure -((sum1)^2)/n1)/(n1 -1);

myvarVal

}

x <- rnorm(10000, mean = 10, sd = 1)

y<-as.numeric(1:100)

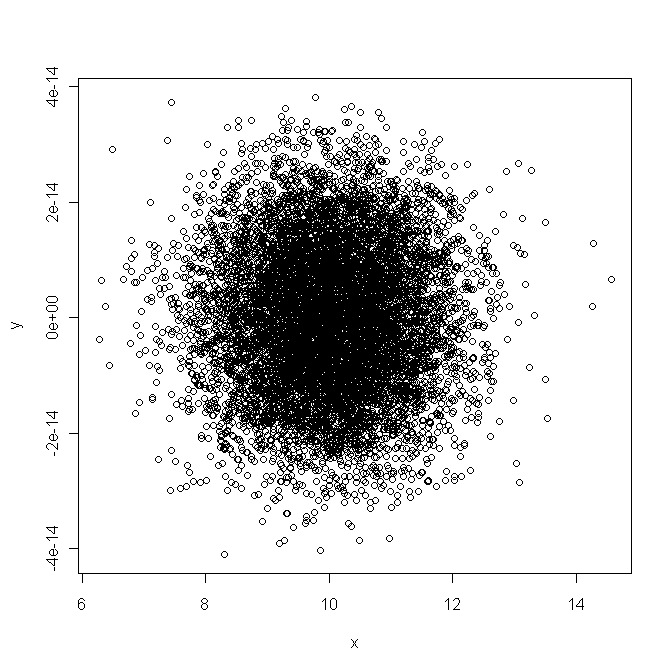
for(i in 1:10000 ) {

y[i] <- (myvar(x[1:i])-var(x[1:i]))

}

plot(x, y)

The plot you attached is OK but it will not work with the code you presented. Correct it.



As we mentioned earlier, a computer program to implement serially the algorithm implied by will converge to some number much smaller than the largest floating-point number.

Here we have sum of numbers and also sum of square of them and n (number of sample) is large and can be assume n is infinite for machine therefore we cannot expect correct result

Residual plot shows us in some case difference between two function is positive and for other is negative

If the number of samples is increased , we get more error in the result .

As solution we can improve the accuracy of result by first sorting the numbers and summing them in order of increasing magnitude. If the numbers are all of the same sign and have roughly the same magnitude, a pair wise “fan-in” method may yield good accuracy (which is mention in detail in course literature )

first numbers to be summed partially and then this partial sum are added until all sums are completed .