Assignment q2

The region of interest in question 2 is defined as follows;

 (I)

*Part a*

d

*Part b*

df

*Part c(i)*

The density of interest is as follows;

 (II)

The function fy\_eval was written in R to evaluate given as follows;

fy\_eval = function(y){

fy = exp(-(y-2)^2) + exp(-(y+2)^2)

fy

}

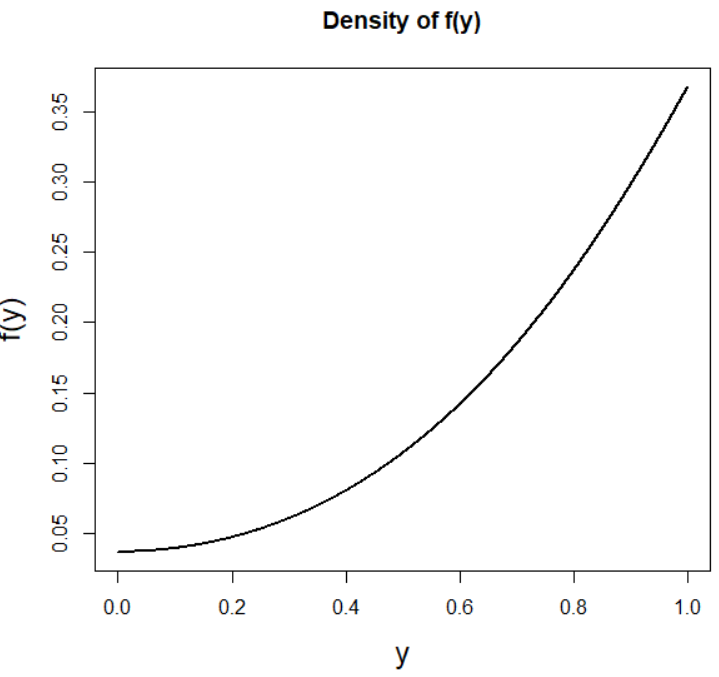
To evaluate and plot up to it’s normalizing constant the following was executed

y = seq(0,1, length = 5000)

fy = fy\_eval(y)

plot(y, fy)

The plot is shown in figure 2c below



*Figure 2.c Density plot of f(y)*

Part c(ii)

The function A\_eval was written to evaluate and visualise the region (given by I). The following pseudocode explains the

*Pseudocode of function A\_eval which is used to evaluate and visualise the region A*

*A\_eval(u.bounds, v.bounds, dimension\_of\_variables)*

1. Initialise the variables u and v
   * u and v were defined using the seq function. The bounds of both u and v are left as user-defined inputs to the function to allow for different bound variations to be easily trialled.
   * Initially no constraints were put on v (for e.g a sequence from -10 to 10 of length 1000 was created) while in the case of u the only constraint on that the initial/minimum value was 0 (for e.g a sequence from 0 to 10 of length 1000 was created)
2. Evaluate *f(v/u)* for every possible combination of u and v
   * A data frame with columns u and v was created using the expand.grid function in R to encompass all possible combinations of u and v in the rows.
   * The function (given by fy\_eval as defined in c(i)) was evaluated for all possible combinations of u and v, specifically and the result was appended as a column to the dataframe.
3. Check which values of u meet the constraint
   * A logical variable called ‘check’ was found and appended to the data frame whereby;
4. Plot region A (u, v)

* Plot u against v and colour the plot according to whether the boundary check is satisfied or not

The function A\_eval below is the realisation of the pseudocode as explained above

**A\_eval** <- function(ubnds, vbnds, dimension1){

#1. Setup variables

v = seq(ubnds[1],ubnds[2], length = dimension1)

u = seq(vbnds[1], vbnds[2], length = dimension1)

#2. Create dataframe

data <- expand.grid(v=v, u=u)

#3. Evaluate function at y = v/u

data <- within(data, f <- sqrt(fy\_eval(v / u)))

#4. Check where boundary constraints met i.e 0 <= u <= fy(v/u)

data$check <- (data$u <= data$f) & (data$u >= 0)

#5. Plot v vs u and colour according to whether the boundary check satisfied or not

plot(v ~ u, data = data, col = data$check + 1,

pch = 16, cex.lab=1.5, main = 'Visualisation of region A')

legend('topleft', legend = c('Not A', 'A'), col = 1:2, pch = 16)

}

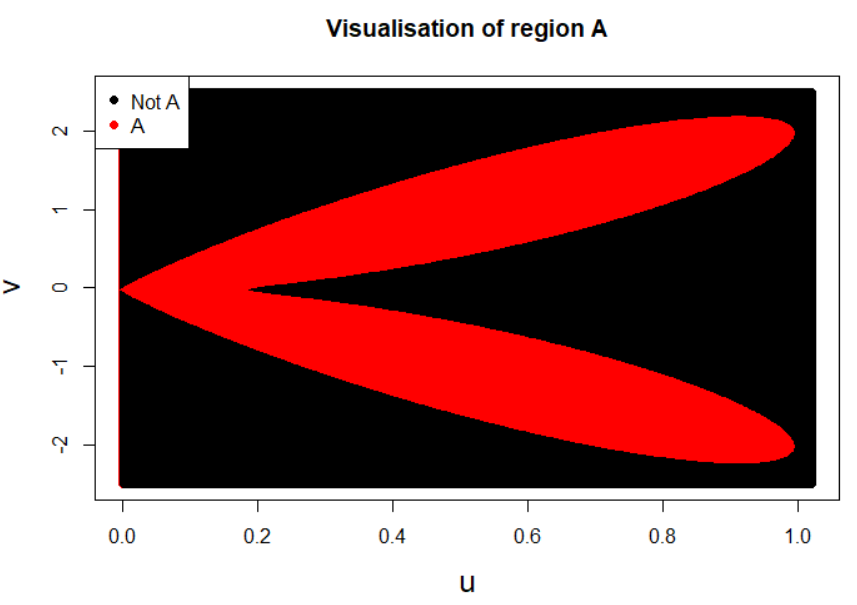
To utiltise the function and produce the plot as shown in Figure 2c(ii) the code below was executed. As explained in the pseudocode, different values for the ranges of u and v were firstly trialled, however upon visualisation of the region, the bounds as set below were deemed sufficient.

#Evaluate function

ubnds = c(-2.5, 2.5)

vbnds = c(0, 1.02)

A\_eval(ubnds, vbnds, 500)



Bounds on A

A = {u, v : u [0,1], v [-2.5, 2.5]

*Figure 2cii. Visualisation of region A*

*Appropriate bounds*

Upon evaluation and visualisation of region A, the following bounds were deemed appropriate;

(III)

*Part d. Samples from*

To generate samples from the density using only randomness from the runif command, the function A\_sampler was created as below. A uniform sample is taken over the area of the rectangle given by bounds as defined in c (given by III). Samples are accepted if they meet the constraint on the region of boundary A, i.e . Note that the lower bound on u is set to be 0 so that constraint is always met. It was found that to generate the required amount of samples the number of proposals had to be increased by a factor of 3.9. The function A\_sampler is as follows;

A\_sampler = function(num\_samps, ubnds, vbnds){

#1. Variables

us = c() #Initialise empty vector

vs = c() #Initialise empty vector

num\_samps = num\_proposals\*3.9 #Increase num of samples by a factor of 3.9

#2. Uniformly sample over Rectangle

u = runif(num\_samps, ubnds[1], ubnds[2]) #uniform [0, 1]

v = runif(num\_samps, vbnds[1], vbnds[2]) #uniform [-2.5, 2.5]

#3. Accept samples based on constraint on the boundary of region A

accept <- u <= sqrt(fy\_eval(v/u))

us = u[accept]

vs = v[accept]

#Return region A i.e (u,v)

return(list(us = us,vs = vs))

}

*Part e Execution & Visualisation*

To obtain 1,000,000 samples from the density involves simply executing the following;

To evaluate this function and obtain samples the following code was executed;

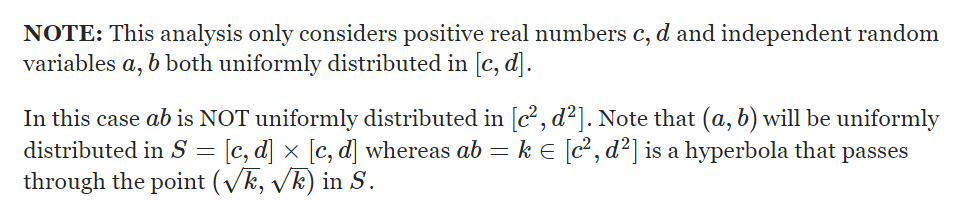
#Apply sampling function

ubnds = c(0, 1.02) #Bounds for u

vbnds = c(-2.5, 2.5) #Bounds for v

num\_proposals = 1000000

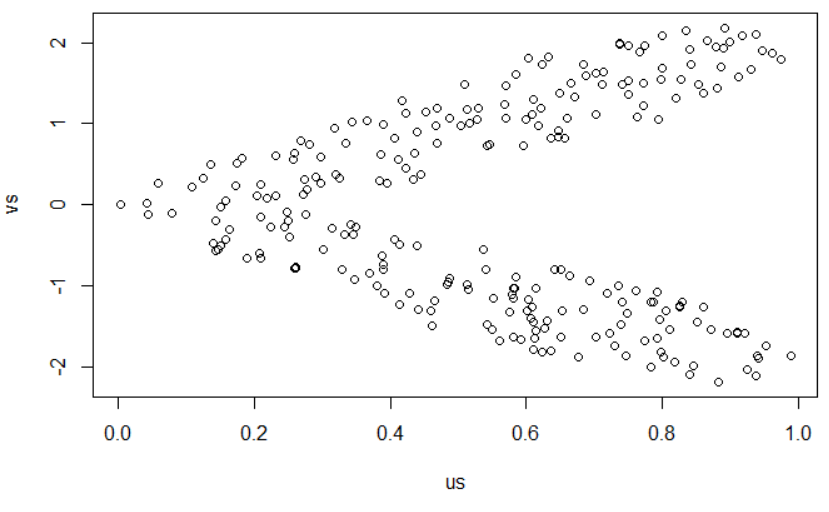
As = A\_sampler(num\_samps, ubnds, vbnds)



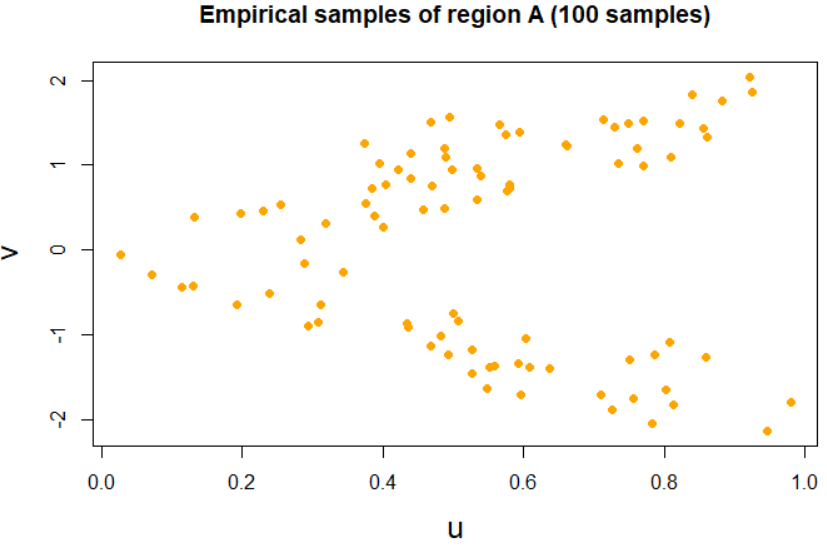
<https://math.stackexchange.com/questions/894453/is-the-product-of-uniformly-distributed-numbers-uniformly-distributed-too>

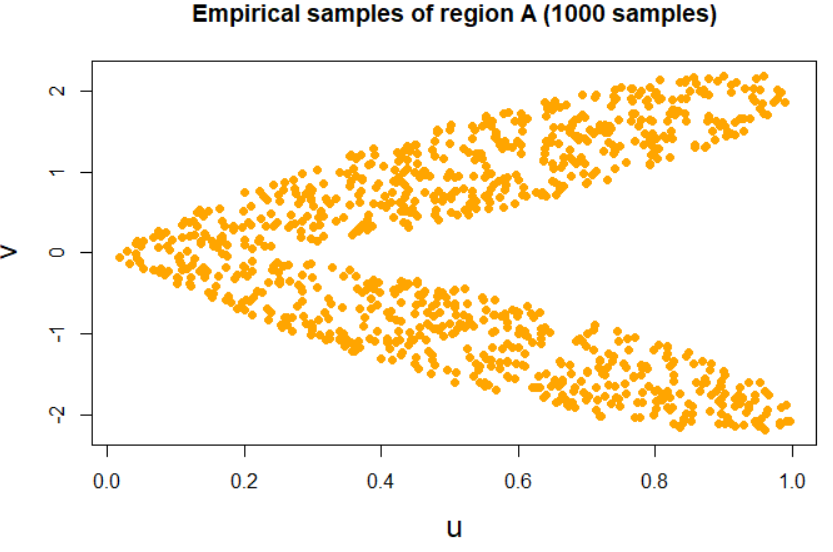
Ac

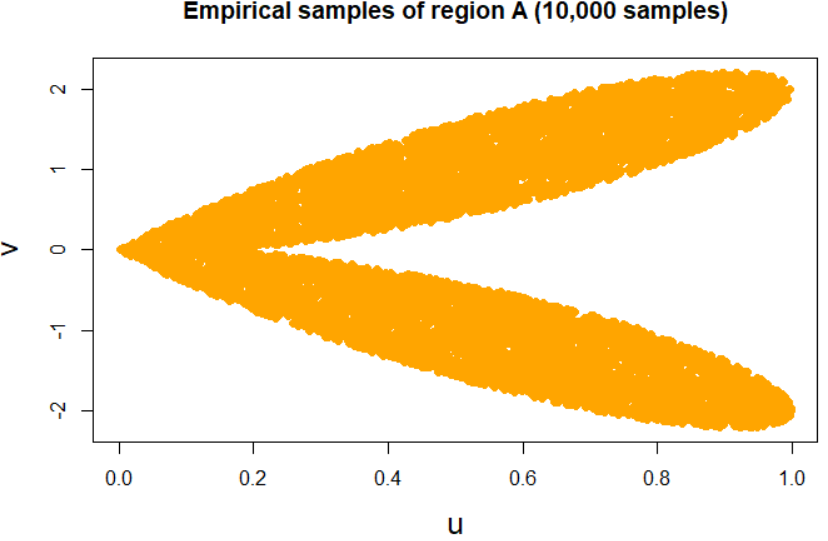
Samples ☺

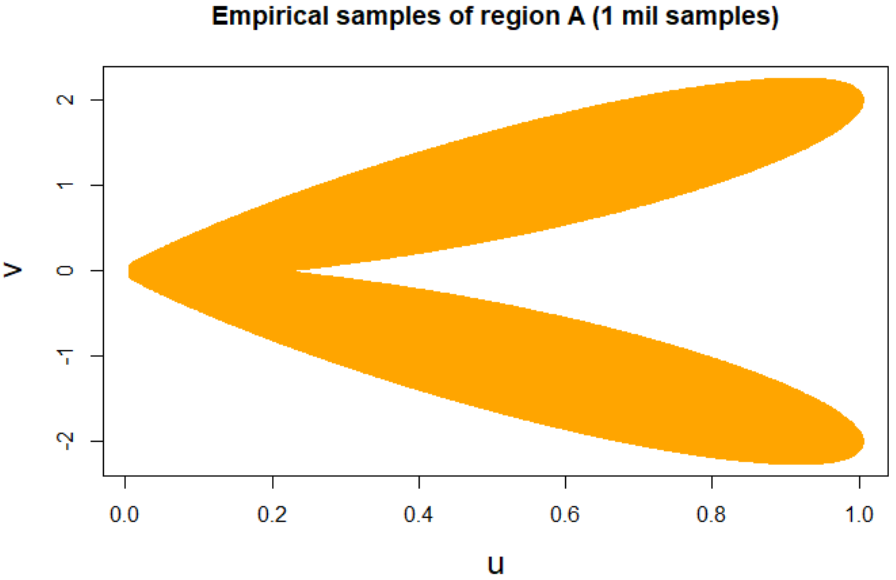


Appendix









Comparison

