*Question 2*

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

 (I)

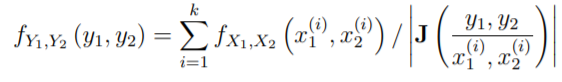
*Part a Proof*

We need to show that;

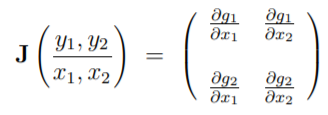
If

Then has density where

That is we need the pdf of a transformed random vector Y in given that we know the pdf, fX(x), of the original random vector X.1 This can be achieved using the following relation;

 (II)

And the  **J** is as follows**;**



In our case we have;

and

Therefore expression II becomes;

/ (III)

Since we are showing this for (U,V) ~ Unif(A) then we know that for a uniform distribution on a two dimenstional set2;

Therefore expression 3 becomes;

/ =

=

Since is a constant and then;

*So*

or where

Q.E.D

*Part b Boundary Conditions*

To provide conditions on f such that A can be enclosed in a rectangle of finite area B requires us to find the limits of this rectangle B3. That is we need to find bounds for the function and where we will let;

and

Since we only need the upper bounds and the lower bound for given by

To find these boundaries requires that we maximise these functions, that is the boundaries of the region B can be written as follows;

To find the maximum of these functions is not trivial so the boundaries were defined as above.

*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

}

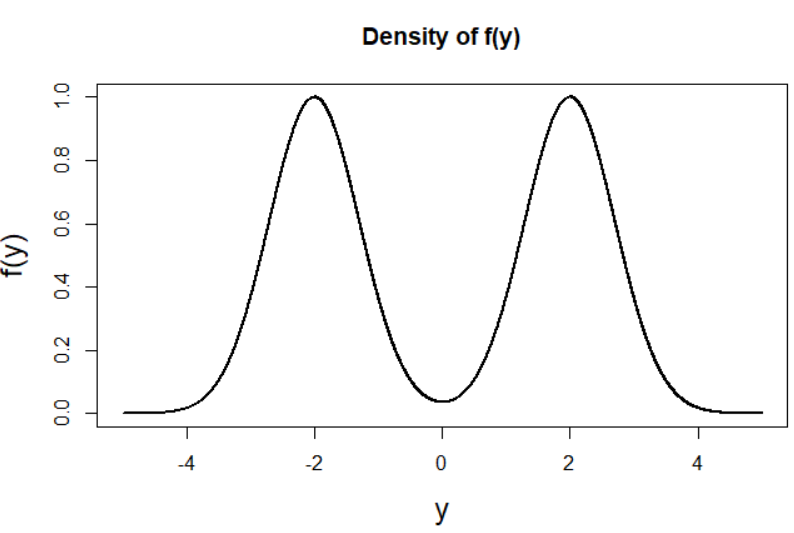
To evaluate and plot up to it’s normalizing constant the following was executed

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)

}

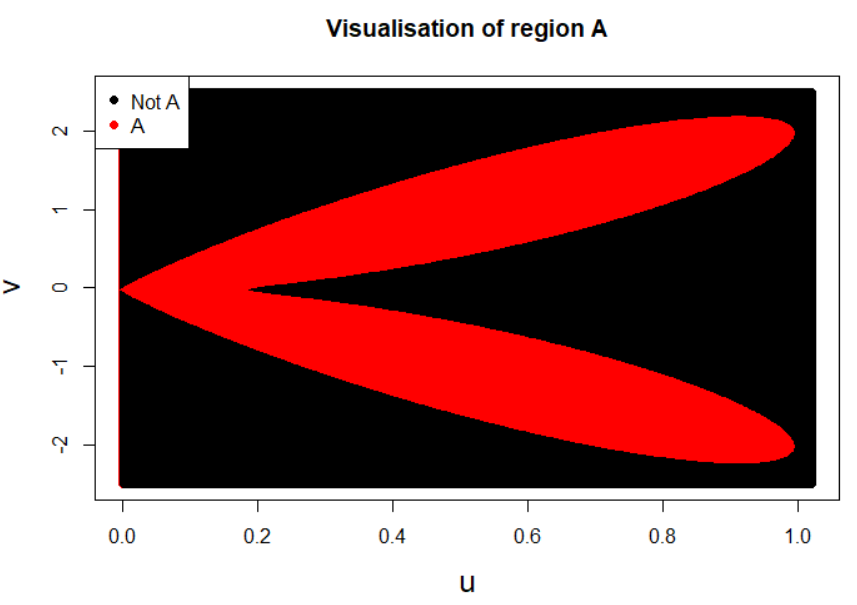
To utiltise the function and produce the plot as shown in Figure 2c(ii) the code below was executed. As explained in the pseudocode, different values for the ranges of u and v were firstly trialled, however upon visualisation of the region, the bounds as set below were deemed sufficient.

#Evaluate function

ubnds = c(-2.5, 2.5)

vbnds = c(0, 1.02)

A\_eval(ubnds, vbnds, 500)



*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 the following code was executed;

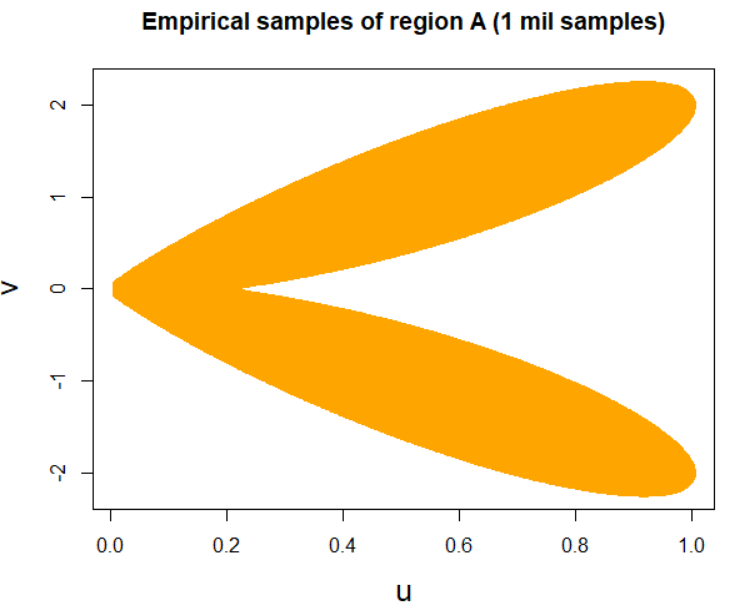
ubnds = c(0, 1.02) #Bounds for u

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

num\_proposals = 1000000

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

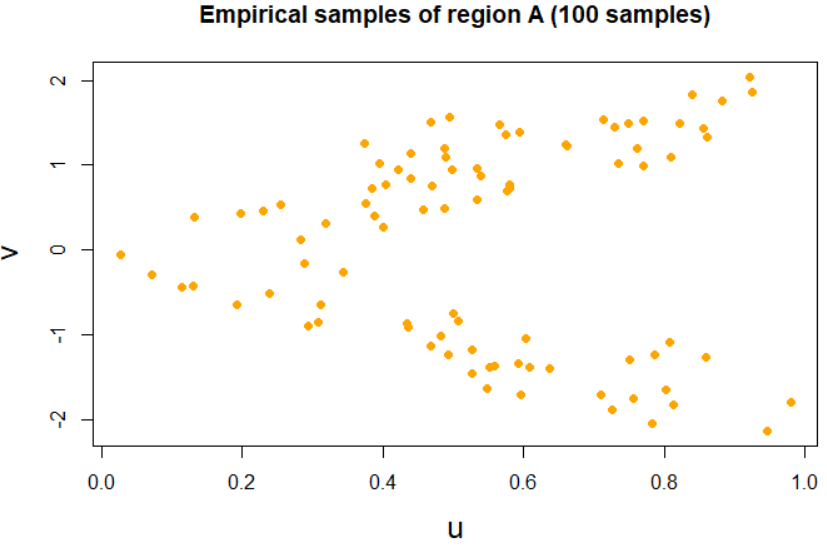
The resultant samples are visualised in Figure 2ei.



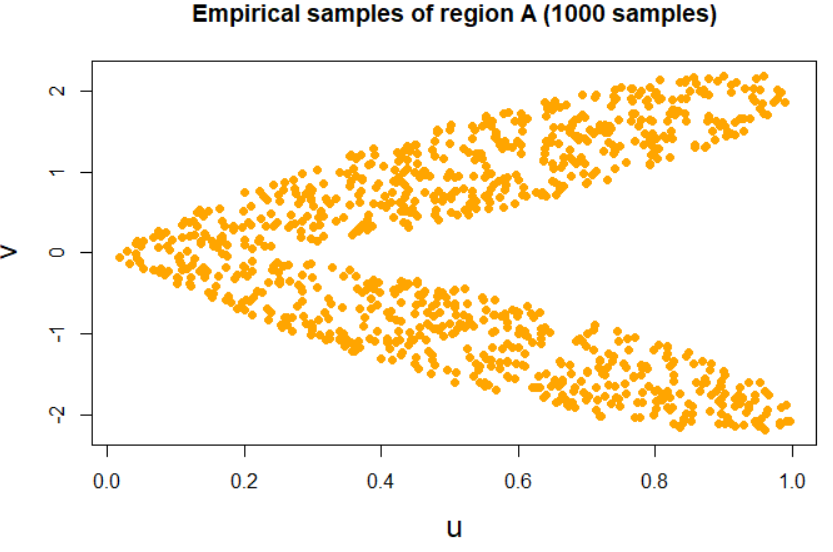
*Figure 2ei. 1,000,000 samples from the density generated using the function A\_sampler*

*Sampling function illustration*

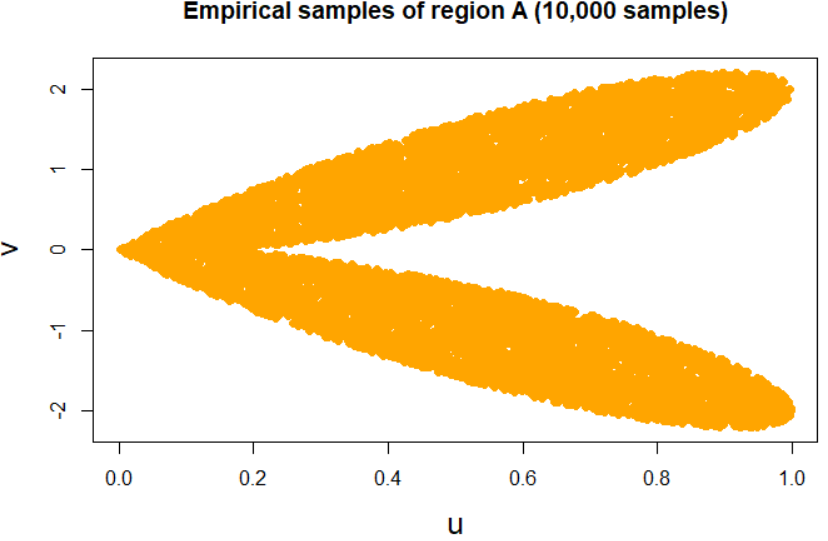
To illustrate the workings of the sampling function A\_sampler and to emphasise the difference between the empirical samples and analytic solution, the function was implemented for sample sizes of increasing orders of magnitude starting at an order of 2 (Figure 2eii), an order of 3 (Figure 2eii), an order or 4, (Figure 2eiii), an order of 5 (Figure 2eiv), while an order of 6 was previously implemented as shown in Figure 2ei.



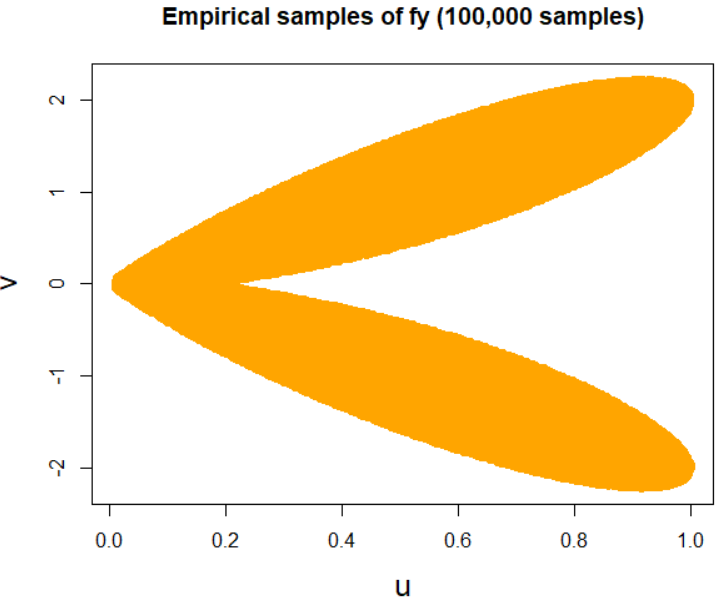
*Figure 2eii*



*Figure 2eiii*



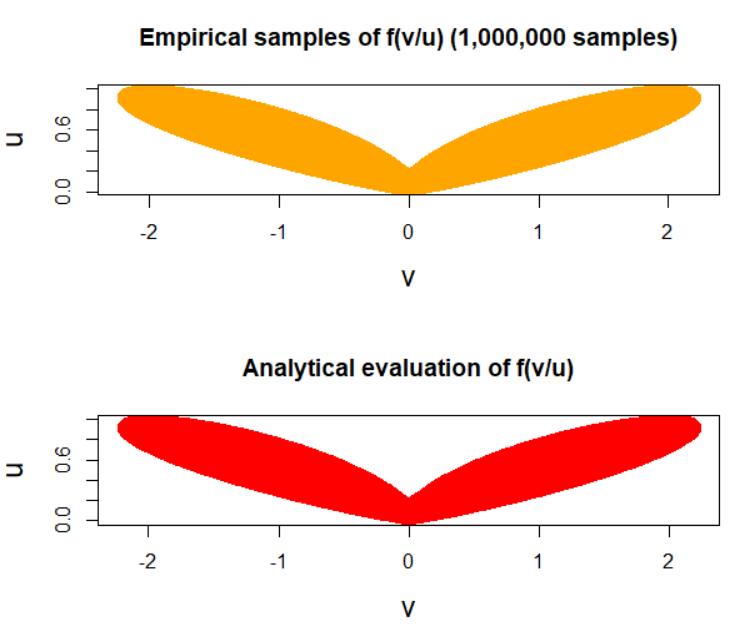
*Figure 2eiv*



*Figure 2ev*

*Comparison*

When the empirical samples generated using the A\_sampler function are compared with the analytic solution found in part c by plotting the results on matching scales, the result is identical, is shown in Figure 2eii.



*Figure 2evi. Comparison of Empirical samples and Analytic evalutation*

Efficiency & Practicality

If the true analytic form of the boundaries given in part b had been found, this would have enabled us to draw uniformly from the exact rectangle B and therefore from the target density A. This could improve the efficiency of the algorithm as it would reduce the amount of required proposal samples. As for a given sample size requirement, the above algorithm requires 4 times the amount of proposals which is a significant factor.

However it could be argued that the function A\_sampler is relatively efficient given it generates a sample size to the order of 6 (1 million samples) in under 3 seconds as displayed in Table 2e. The function was created without the use of any explicit loop in an effort to reduce the overall run time. The elapsed times were found in R using the the sys.time() function. The time corresponds to the average elapsed time of three runs of the function.

Table 2e

|  |  |  |
| --- | --- | --- |
| Function | Number of Samples | Sampling function timing (s) |
|  | 1,000,000 | 1.09 sec |

*References*

1. ‘*Transformation of Random Vectors’* (2019), at <http://ece-research.unm.edu/bsanthan/ece340/note1.pdf>
2. ‘*Probability Theory with Simulations - Two-dimensional continuous distributions’* (2019) at <http://math.bme.hu/~vetier/df/Part-IV.pdf>
3. Martino. L, Miguez. J (2010) ‘*A rejection sampling scheme for posterior probability distributions via the ratio-of-uniforms method’* Universidad Carlos III de Madrid.