STAT - S611 ASSIGNMENT 6

Due Sunday April 26. PLEASE TURN YOUR WORK ELECTRONICALLY VIA CANVAS

- All these questions require you write computer programs. Please write a report with your code, and an explanation of what you've done. Additionally upload the files containing your computer code to Canvas. Include instructions detailing how to use them in your report.

1. The double exponential function has density

$$f^L(x) = \frac{\theta}{2}\exp(-\theta|x|)$$

for $\theta > 0$.

   (a) Derive an algorithm to generate variates from a double-exponential distribution using the inverse CDF method.

   (b) Derive an accept-reject algorithm for obtaining samples from the $N(0,1)$ distribution using the double exponential with $\theta = 1$ as a proposal. Make sure that you use the optimal enveloping constant.

   (c) Implement a function in R that generates variates from the $N(0,1)$ distribution using your accept-reject algorithm. Use it for generating 1000 variates. Plot some interesting plots (histogram, density, etc.). Test your samples for normality using a Kolmogorov-Smirnov test `ks.test` (see R help). Comment your results.

2. Implement your accept-reject standard normal sampler in C++. You will have to implement a function `rnormal` with prototype,

```
double rnormal();
```

   that generates a single draw from a standard normal distribution using your rejection algorithm.

   For this you will need to generate uniform variates. Don't use the default prng from the C library—`rand()`; it is a low-quality algorithm, unsuitable for statistical applications. Instead you will use the Mersenne twister algorithm. You can found a C++ implementation in the "MersenneTwister.h" file, uploaded to the "files/code" section in Canvas. To use it, download the file and save it in the same directory of your code. Include the following lines after your "include" statements in your source code file.

```
#include"MersenneTwister.h"
#include<math.h>

MTRand rng;
```

   These lines declare and initialize a **global** object called `rng`, of class MTRand, that can be used to generate pseudo-random numbers. To get a pseudo-random number on the interval $(0,1)$ you can use the class method "rand()" within any of your functions. For example:

```
double u = rng.rand();
```

   If your object `rng` is global, a call to `rng.rand()` will generate a different pseudo-random number each time.

   Test your code with the following `main` function (you need to include the header file `stdio.h` for this to work)

```
int main(){
    FILE* f = fopen("numbers.dat", "w");
    for (int i = 0; i < 1000; i++){
        fprintf(f, "%1.15e\n", rnormal());
    }
    fclose(f);
}
```

This program will generate 1000 normal variates using your `rnorm` function (assuming your function is correct!) and will write them into a text file called "numbers.dat".

Load these numbers into an object in R by using the `scan` function (read about this function in the R help files).

```
>x <- scan("numbers.dat")
```

Again generate some plots and perform a test of normality. Comment your results.

3. Now you will write an R interface to your C++ code. Write a function `my_norm`, callable from R using the `.C` mechanism (return type `void`, and pointer arguments). This function should generate an arbitrary number (specified by the user) of random variates using your `rnormal` function, and return them to R. Don't forget to include the "`extern"C"`{}" declaration when compiling with g++.

   Write and R wrapper function to call your C function from R. Use it to generate 1,000,000 samples and compare speed with your R implementation from the previous question using the `system.time` function (read the corresponding help file to learn about this function). Comment your results.