DTU Fall 2016, d. 25 October 02635 - Mathematical Software Programming

Student: **Mattia Paterna, 163143**

# ASSIGNMENT 1 - REPORT

The code is organised as follows:

1. a main script (**randGen.c**) deals with the I/O part. It prompts the user to enter all the information, successively compute the sampling based on it and finally prints the matrix back;
2. one script, **generators.c**, provides all the functions involved in the sampling process;
3. one separate script, **array2d.c**, has been made up to deal with all the memory management operations, both allocating and free new memory.

Most of the variables are of type **double**, whereas the two-dimensional matrix with random numbers has a changing data type depending whether the chosen distribution is the exponential or the Poisson one. That is because the support in a Poisson distribution for k is the set of natural numbers. In order to save memory, a data type **int** could be chosen instead of a data type **double**.

Snippets of code have been written to test if the distributions work properly. They are within an **#ifdef DEBUG…#endif** (cfr. Appendix). I tried to run the sampling process for large numbers of generated random variables (e.g. 100.000) and all the sample means obtained were close to the mean of the distribution.
Moreover, a comparison with **poissrnd** Matlab function shows, for instance, how the probability density function (*pdf)* of the implemented function mostly matches the pdf of the Matlab function. The figure in the Appendix shows in the histograms for both sampling. I can also conclude that a large sampling from a Poisson distribution tends to look like a Gaussian distribution.

Both method 1 and 2 have been implemented. Particularly, the code itself switches from 1 to 2 depending on lambda value. Regarding the two methods:

1. the expected run time for the first approach is proportional to the parameter lambda, so that for large arguments leads to slow performances;
2. the second approach has expected run time independent of the variable lambda, but it could be not completely efficient to generate a large number of samples. That is partly due to the type of operations inside the routine (e.g. **exp** and **log**).

# APPENDIX

## a. **the #ifdef DEBUG code chunk.**

```c
/* check for correctness of exponential distribution computing the sample mean */
#ifdef DEBUG
double z = 0.0;
for (size_t i = 0; i < 100000; i++) {
  z += (c == 'E') ? expGen(lambda) : poissonGen(lambda, method);
}
z /= 100000.0;
printf("mean = %e\n", z);
if (c == 'E') printf("lambda^-1 = %e\n", pow(lambda, -1));
else printf("lambda = %e\n", lambda);
#endif
```

## b. **comparison of histograms.**

(The upper shows a sampling from the MATLAB® function **poissrnd**, whilst the lower shows a sampling from the implementation in C. Both sampling contain 1 million random variables.)