Stochastic Simulation
Spring 2025 January 16, 2025

Homework #1: Generating Random Variables
Due: February 13, midnight

Part I: Theoretic Questions

#1 **(Quality of Random Number Generator)** We all know Fibonacci sequence, which is defined by the following recurrence relation:

$$Z_0 = 0, Z_1 = 1, Z_n = Z_{n-1} + Z_{n-2} \ \ \forall\, n > 1.$$

This sequence was used to generate pseudo-random numbers in history, named the *Fibonacci generator*:

$$Z_i = (Z_{i-1} + Z_{i-2}) \mod m,$$

but it has serious deficiencies, as you can prove in the following two parts.

(a) Show that this generator can never produce the following arrangement of three consecutive output values: $U_{i-2} < U_i < U_{i-1}$.

(b) Show that the arrangement in part (a) should occur with probability $\frac{1}{6}$ for a "perfect" random-number generator.

#2 **(Inverse Method to Generate Random Variables)** Give inverse method algorithms for generating random variables with the following density functions:

(a) $f(x) = e^x/(e-1), \quad 0 \le x \le 1;$

(b) $f(x) = \{\pi\,[1 + (x-1)^2]\}^{-1};$

(c) $f(x) = \begin{cases} \frac{x-2}{2} & \text{if } 2 \le x \le 3 \\ \frac{2-x/3}{2} & \text{if } 3 \le x \le 6 \end{cases}.$

Part II: Numerical Experiments: For the following problems, report your numerical results with properly designed tables or graphs and clearly explain the results with a few sentences. Put your .py file (or Jupyter notebook files) in a single zip file and submit it via Blackboard. In case you use any data input, you should also include your .csv files or other type of files in your submission. Please name your file `Lastname_StudentID.zip`. Also, give meaningful names to your decision variables and constraints, and add comments to your code liberally.

#3 **(Generating Poisson Random Variables)** In the lecture, we introduced 2 different algorithms to generating Poisson random variables: (i) Inverse Method; and (ii) ad-hoc method using exponential inter-arrival times.

(a) Implement the two methods in `Python`.

(b) Verify your implementation as follows. Using your codes to generate 10,000 samples from Poisson distribution with mean 1. Compare the histogram of the simulated data with the theoretic probability mass function.

(c) Generate 100,000 samples using the two methods you have implemented and the function `np.random.poisson()`, respectively. Compare their computation times. Can your implementation beat the numpy function?

#4 **(Sampling from Posterior Distribution)** Consider a Bayesian statistic model in which the data are i.i.d. Poisson random variables with mean θ. Suppose the prior of the parameter $\theta > 0$ is a Gamma(α, β) with $\alpha, \beta > 0$ and its PDF is

$$f_0(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \cdot \theta^{\alpha-1} e^{-\beta\theta}, \quad \theta > 0.$$

(a) Derive the posterior distribution $f_1(\theta)$ given i.i.d. data $X_1, ..., X_n$.

(b) Design an A-R algorithm to sample from the posterior distribution $f_1(\theta)$ and write down the pseudo codes.

(c) Implement the A-R algorithm. It should take the parameters (α, β) of the prior distribution and the i.i.d. data samples as input by the users. Illustrate the efficacy of your implementation using any examples you like and report the results.

**********END**********