CS 590M
Simulation **February 27, 2020**
Peter J. Haas **Spring Semester 2020**

Assignment #4 (Due March 12)

1. (Truncated distributions) Law, problems 8.3 and 8.4.

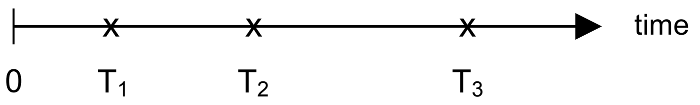
2. (Geometric random variates) Law, problem 8.14. For the second part of (a), you can simply use the fact, given on page 305, that the cdf of a geometric random variable is $F(x) = 1-(1-p)^{\lfloor x \rfloor + 1}$ for $x \geq 0$ or, for extra credit, you can derive this formula

3. Law, problem 8.11(a). For the composition method, express the original distribution as a weighted combination of a distribution on [-1,0] and a distribution on [0,1]. For acceptance-rejection, use an appropriate uniform random variable as the majorizing random variable.

4. A *non-homogeneous Poisson process* (NHPP) with points (e.g., arrival times) $T_1, T_2, \ldots$ and *hazard function h* has the defining property that, taking $T_0 = 0$,

$$P(T_n > t \mid T_{n-1} = y_{n-1}, \ldots, T_1 = y_1) = P(T_n > t \mid T_{n-1} = y_{n-1}) = e^{-\left(H(t) - H(y_{n-1})\right)} \qquad (*)$$

for $n \geq 1$ and $t > y_{n-1}$, where $H(t) = \int_0^t h(u)\,du$ is the *cumulative hazard function*. (The name of the process arises from the fact that, when $h(t) \equiv \lambda$ for all $t \geq 0$, the NHPP reduces to an ordinary Poisson process. NHPPs can be used to model time-of-day effects in an arrival process.)



Suppose that $h(t) = at^{a-1}$ for some $a > 0$. Give an algorithm for generating points $T_1, T_2, \ldots$, assuming the availability of a sequence $U_1, U_2, \ldots$ of uniform[0,1] random numbers. Your algorithm should not use acceptance/rejection. [Hint: use (*) and the inversion method.]

5. Read Section 7.4.1 in the Law textbook, and apply the runs-up test to your favorite (or least favorite) random number generator with *n* = 5000 samples. (Work on this problem individually. You may use the algorithm in problem 7.8 of the text to compute the number of runs up.)

6. (**Computing problem**) This problem introduces you to the ratio-of-uniforms method, which typically incorporates acceptance-rejection and can lead to very fast random number generators. (It also gives you some experience with Q-Q plots, which were briefly mentioned in class.)

   a) Read Section 8.2.5 in the textbook, and then compute the constants u^*, v_*, and v^* in the case of a standard (mean 0, variance 1) normal distribution, using a value of $p = \sqrt{2\pi}$ to make the algebra less messy. [Hint: u^* is obvious, and both v_* and v^* can be found by setting derivatives equal to 0.]

b) Sketch the set S and the enclosing rectangle T. [Hint: Substitute the formulas for p and f into the boundary condition $u = \sqrt{pf(v/u)}$, then solve for v as a function of u to get the formulas for the upper and lower boundaries. You get these boundaries by taking a positive and negative square root.]

c) Give the final algorithm for generating standard normal random numbers.

d) What is the acceptance probability for the generator? What is the expected number of uniform random numbers required to produce a single normal sample Z?

e) Implement the random number generator in Python. Generate 500 samples from $N(0,1)$ and create a Q-Q plot to see if the output actually does look normal, i.e., lies mostly along the diagonal line. The textbook discusses Q-Q plots on pp. 339-344. You may use the function scipy.stats.probplot to create the Q-Q plot, e.g., if the array samples contains the generated normal samples, then the code is

```
from scipy.stats import probplot
import matplotlib.pyplot as plt
probplot(samples, dist=”norm”, plot=plt)
plt.show()
```