Distribution Sampling

- Discrete event simulation depends on the ability to sample distributions, e.g. Exponential, Weibull, (truncated) Normal, Cauchy, Pareto, Geometric...
- We'll look at three commonly-used methods:
  1. Inverse transform method
  2. Acceptance-Rejection (AR) method
  3. Convolution method

Short Diversion (UNASSESSED): Sampling $U(0, 1)$

- All distribution samplers assume the ability to generate uniform-random numbers in the interval $(0, 1)$
- The robust RNGs exploit number theory and are based on the *Linear Congruential Generator* which generates a sequence x_0, x_1, x_2... via the rule

$$x_{n+1} = (ax_n + c) \ mod \ m$$

  where a is a *multiplier*, c is an *increment* and m is the *modulus*
- Samples from $U(0, 1)$, viz. $u_0, u_1, u_2, \ldots$ are obtained by setting $u_i = x_i/m, i = 0, 1, 2, \ldots$.
- Note that $0 \leq x_i < m$ so the maximum *period* is m
- An important objective is to maximise the period

Multiplicative Congruential Generators

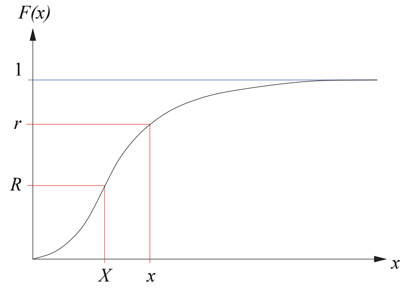
- A *Multiplicative* Congruential Generator (MCG) is obtained by setting $c = 0$ in an LCG (but note now that x_i must be non-zero)
- It can be shown that a multiplicative generator has (maximum) period $m - 1$ if m is prime and if the smallest integer k for which $(a^k - 1) \ mod \ m \ = \ 0$ is $k = m - 1$
- Example: $a = 3, m = 7$ whereupon, (only) when $k = m - 1 = 6$, $a^k - 1 = 728 = 104 \times 7$; starting with $x_0 = 1$, we obtain:

$$\underbrace{x_0 = 1, x_1 = 3, x_2 = 2, x_3 = 6, x_4 = 4, x_5 = 5}_{Period=6}, x_6 = 1, x_7 = 3...$$

- Java's `Math.random()` uses an MCG with a 48-bit "seed"
- Modern generators combine several MCGs or generalise the MCG approach e.g. the "Mersenne Twister" (period $2^{19937} - 1$)!

1. The Inverse Transform method

- Suppose X is a continuous r.v. with cdf $F(x) = P(X \leq x)$ and that we are trying to sample X
- Let $U \sim U(0, 1)$; what can we say about $F^{-1}(U)$?



- Because $F(x)$ increases monotonically, we have: $P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$,

- Algorithm: Sample $U(0,1)$ giving some value $0 \leq U \leq 1$, then compute $F^{-1}(U)$
- Of course, this only works if $F(x)$ has an inverse
- Example: If $X \sim U(a,b)$ then

$$F(x) = \frac{x-a}{b-a}, \quad a \leq x \leq b$$

- Setting $U = F(x)$ and inverting F gives

$$X = F^{-1}(U) = U(b-a) + a$$

- This confirms what we (should!) already know: if $U \sim U(0,1)$, then $(U(b-a)+a) \sim U(a,b)$

- Example: If $X \sim \exp(\lambda)$ then

$$F(x) = 1 - \exp^{-\lambda x}, \quad x \geq 0$$

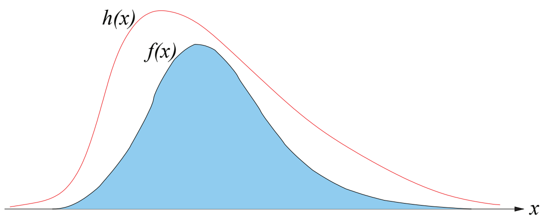
- Setting $U = F(X)$ and inverting, we get:

$$\begin{aligned} U &= 1 - \exp^{-\lambda X} \\ 1 - U &= \exp^{-\lambda X} \\ \log_e(1-U) &= -\lambda X \\ \frac{-\log_e(1-U)}{\lambda} &= X \end{aligned}$$

- So, if $U \sim U(0,1)$, then $-\log_e(1-U)/\lambda \sim exp(\lambda)$
- Note that we can replace $1-U$ with U since $(1-U) \sim U(0,1)$, viz. $-\log_e(U)/\lambda$

2. The Acceptance-Rejection (AR) Method

- If $F(x)$ cannot be inverted we can sometimes work with the density function $f(x)$
- Idea: Find a function $h(x)$ that dominates $f(x)$, i.e. for which $h(x) \geq f(x)$ for all x



- Now generate a density function, $g(x)$ say, from $h(x)$ by "normalising" it so that the area under $g(x)$ is 1:

$$\begin{aligned} \int_x h(x)dx = c \;\Rightarrow\; g(x) &= \frac{h(x)}{c} \\ \text{cdf } G(x) &= \int_{-\infty}^{x} g(t)dt = \int_{-\infty}^{x} \frac{h(t)}{c} dt \end{aligned}$$

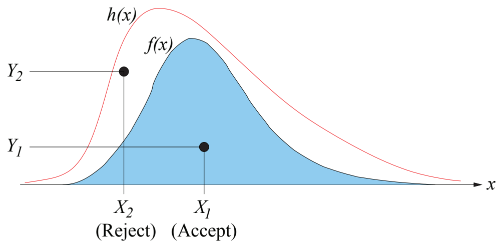
and assume that it is easy to sample $g(x)$ and/or $G(x)$

- Aternatively... start with a density function, $g(x)$, and scale it by factor c to give a function $h(x)$ that dominates $f(x)$
- Once again the scaling (normalising) factor, c, is the area under $h(x)$:

$$c = c\int_x g(x)dx = \int_x h(x)dx$$

since $g(x)$ is a density function

- AR Algorithm:
  1. Let X be a sample generated using $g(x)$ or $G(x)$
  2. Generate a $U(0,1)$ sample, U, and let $Y = Uh(X)$
  3. If $Y \leq f(X)$, i.e. if $U \leq \frac{f(X)}{h(X)} = \frac{f(X)}{cg(X)}$, then accept X; otherwise reject it and start again
- It's a "dart throwing" exercise!
- By construction, the samples X and Y define a point that lies under $h(X)$; if (X, Y) lies under $f(X)$ as well we accept X



Example: Half-normal

- Suppose we wish to sample a standard "half-normal" distribution:

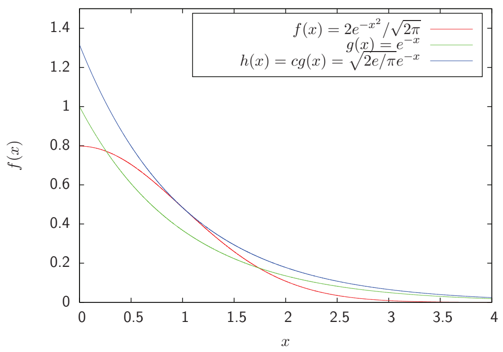
$$f(x) = \frac{2}{\sqrt{2\pi}} e^{-x^2/2}, \quad x \geq 0$$

- We'll arbitrarily choose $g(x) = e^{-x}$ (exponential, parameter 1), as it's easy to sample
- We need an $h(x) = cg(x)$ that dominates $f(x)$ for $x \geq 0$
- c can be found by computing $\max_x f(x)/g(x)$, i.e.

$$c = \max_{x \geq 0} \frac{\frac{2}{\sqrt{2\pi}} e^{-x^2/2}}{e^{-x}} = \max_{x \geq 0} \sqrt{\frac{2}{\pi}} e^{x - x^2/2}$$

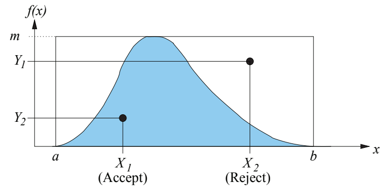
- By differentiation, this is maximal when $x = 1$, whence $c = \sqrt{2/\pi} e^{1/2} = \sqrt{2e/\pi}$, so

$$h(x) = \sqrt{2e/\pi} e^{-x} \quad \text{and} \quad \frac{f(x)}{h(x)} = e^{-(x-1)^2/2}$$



- Notice how $h(x)$ dominates $f(x)$ (just...! by design...!)
- Thus, sample X from $-\log(1 - U_1)$ (inverse transform method applied to exponential distribution, parameter 1) and accept X iff $U_2 \leq \frac{f(X)}{h(X)} = e^{-(X-1)^2/2}$, where $U_1, U_2 \sim U(0,1)$

- Special case: if $a \leq x \leq b$ then we can enclose $f(x)$ within a $U(a,b)$ density (a rectangle) by choosing $g(x) = 1/(b-a)$ and $h(x) = \max_x f(x) = m$, say:



- Thus, sample X from $U_1(b-a) + a$ and accept X iff $U_2 \leq \frac{f(X)}{h(X)} = f(X)/m$, where $U_1, U_2 \sim U(0,1)$
- Intuitively, the method works because the smaller $f(X)$ is the less likely you are to accept X
- More rigorously, we need to show that the cdf for those values of X that we accept is precisely F, i.e. we need to show that:

$$P(X \leq x \mid U \leq \frac{f(X)}{h(X)}) = F(x)$$

- Here, we'll use the Bayes' Theorem variant:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- The proof goes as follows (recall that $\int_x f(x)dx = 1$ and $\int_x h(x)dx = c$):

$$\begin{aligned}
P(X \le x \mid U \le \frac{f(X)}{h(X)}) &= P(X \le x \mid U \le \frac{f(X)}{cg(X)}) \\
&= P(U \le \frac{f(X)}{cg(X)} \mid X \le x)\frac{P(X \le x)}{1/c} \\
&= \frac{F(x)}{c\,G(x)}\,\frac{G(x)}{1/c} \\
&= F(x)
\end{aligned}$$

- The efficiency depends on the number of rejections R before accepting a value of X
- The probability of accepting X in any one experiment, p say, is simply the ratio of the areas of the two functions:

$$p = \frac{1}{c}$$

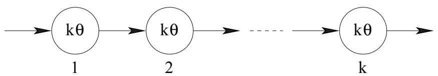
- Since each "experiment" is independent, R is geometrically distributed:

$$\begin{aligned}
P(R = r) &= p(1-p)^r \\
\text{i.e.} \quad E(R) &= \frac{1-p}{p}
\end{aligned}$$

- Example: For the half-normal (above) $c = \sqrt{2e/\pi} = 1.315$ so $1/c = 0.760$. Thus $E(R) = 0.24/0.760 = 0.315$, i.e. $2 \times (1 + 0.315) = 2.63$ random $U(0,1)$ samples on average per half-normal sample (very efficient!).

3. The Convolution Method

- A number of distributions can be expressed in terms of the (possibly weighted) sum of two or more random variables from other distributions
- "The distribution of the sum is the convolution of the distributions of the individual rvs"
- Example: An Erlang(k, θ) rv is defined as the sum of k rvs each with distribution exp$(k\theta)$
- We can think of X being the time taken to pass through a chain of k delays, each with an exp$(k\theta)$ delay time distribution:



- Notice that

$$E[X] = \frac{1}{k\theta} + \frac{1}{k\theta} + \ldots \frac{1}{k\theta} \;=\; \frac{1}{\theta}$$

- We can generate Erlang(k, θ) samples using the sampler for the exponential distribution: if $X_i \sim$ exp$(k\theta)$ then

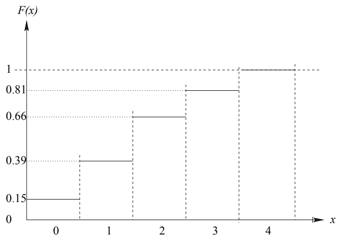
$$X = \sum_{i=1}^{k} X_i \;\sim\; \text{Erlang}(k, \theta)$$

- If $U_i \sim U(0,1)$ then X_i is sampled using $-\log U_i/(k\theta)$
- We can save the expensive $\log$ calculations in the summation by turning the sum into a product:

$$X \;=\; \sum_{i=1}^{k} -\frac{\log U_i}{k\theta} \;=\; -\frac{1}{k\theta}\log\prod_{i=1}^{k} U_i$$

Sampling Discrete Distributions

- We can apply the inverse transform method, again by inverting the cumulative distribution function $F(x)$
- For a discrete rv the cdf is a "step function", e.g.



x	$p(x)$	$F(x)$
0	0.15	0.15
1	0.24	0.39
2	0.22	0.61
3	0.20	0.81
4	0.19	1.00

- To generate a sample, we drive the table "backwards" by mapping a $U(0, 1)$ sample to the corresponding value of x

Quiz: A naive implementation involves a lookup ($O(n)$ or $O(log(n))$. Can you solve it in $O(1)$ time by pre-processing the distribution?