MATH 416/516 ASSIGNMENT 2 SOLUTIONS

Note: assume all Us are RVs from $Uni(0, 1)$ and the page references are to the textbook.

- Problem 4.1: Matlab

```
p = [1 2]/3; F = [1/3 1]; js = [1 2];
N=100; for i=1:N, X(i) = min(js(rand<F)); end, disp(sum(X==1)/N)  % Part a)
         0.32
N=1000; for i=1:N, X(i) = min(js(rand<F)); end, disp(sum(X==1)/N)  % Part b)
        0.324
N=10000; for i=1:N, X(i) = min(js(rand<F)); end, disp(sum(X==1)/N)  % Part c)
       0.3262
```

- Problem 4.2: Matlab

```
function X = myrand( p, n )
% Input p is pmf vector of length n
% Output is random integer 1, ... ,n with pmf p.
js = [1:n]; F = cumsum(p); X = min( js( rand < F ) );
% end myrand
```

- Problem 4.4*: Matlab

```
K = 10000; n = 100; I = [1:n]; % initial deck
for i = 1:K, D = I; % now generate random permutation of deck
  for k = n:-1:2, j = ceil(k*rand); D([j k])=D([k j]); end
   X(i) = sum( D == I ); % count total hits
end, disp([mean(X),var(X)])
       0.9915      0.99693
```

Mean and standard deviation should both be 1.

- Problem 4.7: Matlab code uses vector v with v_j as a T/F flag for dice sum $= j$:

```
for K = 10.^[2 3 4]  % use K runs for K = 100,1000,10000
  for k = 1:K, v = zeros(1,12); i = 0; % i counts dice throws
    while sum(v)<11, i = i+1; % check dice sum for ith throw
      v( sum( ceil(6*rand(1,2)) ) ) = 1;
    end, X(k) = i;
  end, disp([K mean(X)]);
end
          100        57.16
         1000       62.207
        10000       61.339
```

Expected number of dice rolls is approximately 61.

- Problem 4.10

  a) A negative binomial RV $X = \sum_{i=1}^{r} X_i$, if each X_i is a geometric RV with parameter p (see p.23), so $X_i = 1 + \lfloor \frac{\ln(U_i)}{\ln(1-p)} \rfloor$ (p.54), and therefore $X = r + \sum_{i=1}^{r} \lfloor \frac{\ln(U_i)}{\ln(1-p)} \rfloor$.

  b) Just check the algebra.

  c) Given r and p, generate $U \sim Uni(0,1)$, initialize $pj = p^r$, $F = pj$, $j = r$, and then (p.50) use while $U > F$, $j = j+1$; $pj = (j-1)(1-p)pj/(j-r)$; $F = F + pj$; end, $X = j$.

  d) Initialize $i = 0$, $j = 0$, then, following the suggestion, use while $i < r$, $j = j+1$, if $Uni(0,1) > p$, $i = i+1$ end, end, $X = i$.

- Problem 4.11; use a modified version of the permutation algorithm to generate random subsets of size r, until one of $1, \ldots, k$ is present.

  Algorithm (assuming each U is a new Uniform(0,1) RV):

  Do: initialize $P_i = i$, $i = 1, 2 \ldots, n$

  for m = n : -1 : n-r+1, set $j = \lceil mU \rceil$ and swap P_j, P_m values; **end**

  Until $\min(P_{n-r+1}, P_{n-r+2} \ldots, P_n) \le k$:

  Note: for small r, k this might not be very efficient. Modified algorithm:

  Initialize $P_i = i$, $i = 1, 2 \ldots, n$; set $j = \lceil kU \rceil$ and swap P_j, P_n values;

  for m = n-1 : -1 : n-r+1, set $j = \lceil mU \rceil$ and swap P_j, P_m values; **end**

  Output $\{P_{n-r+1}, P_{n-r+2} \ldots, P_n\}$.

- Problem 4.12*

  $E(|Z|) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} |x| e^{-x^2/2} dx = \frac{2}{\sqrt{2\pi}} \int_{0}^{\infty} x e^{-x^2/2} dx = -\frac{2}{\sqrt{2\pi}} e^{-x^2/2} \Big|_0^\infty = \frac{2}{\sqrt{2\pi}} \approx 0.798.$

- Problem 4.16: for composition there are 4 cases: $5x.06 = .3$, $2x.15 = .3$, $2x.13 = .26$, and $1x.14$, corresponding to respective X values $(1, 2, 3, 4, 5)$, $(6, 9)$, $(7, 10)$, and (8).

  Algorithm:

  Generate U_1, U_2;

  If $U_1 < .3$ then $X = \lceil 5U_2 \rceil$; Elseif $U_1 < .6$ then if $U_2 < .5$, $X = 6$, else $X = 9$; Elseif $U_1 < .86$ then if $U_2 < .5$, $X = 7$, else $X = 10$; Else $X = 8$.

- Problem 4.17: using a little algebra, rewrite the pmf as $P\{X = j\} = \frac{1}{2}[(\frac{1}{2})(\frac{1}{2})^{j-1}] + \frac{1}{2}[(\frac{1}{3})(\frac{2}{3})^{j-1}]$, a composition of two equally weighted geometric distributions (see p.53 with $p = 1/2, 1/3$).

  Algorithm for X: if $U_1 < 1/2$, set $X = 1 + \lfloor \frac{\ln(U_2)}{\ln(1/2)} \rfloor$, otherwise set $X = 1 + \lfloor \frac{\ln(U_2)}{\ln(2/3)} \rfloor$.

  The first two Us from p.48 are $U_1 = .23$ and $U_2 = .66$, so $X = 1 + \lfloor \frac{\ln(.66)}{\ln(1/2)} \rfloor = 1 + \lfloor .59946 \rfloor = 1$.

- Problem 4.18*:

  a) we have $p_1 = \lambda_1(1 - \sum_{j=1}^{0} p_j) = \lambda_1$, $p_2 = \lambda_2(1 - \sum_{j=1}^{1} p_j) = \lambda_2(1 - p_1) = \lambda_2(1 - \lambda_1)$;

  Using induction, assume $p_i = \lambda_i(1 - \sum_{j=1}^{i-1} p_j) = \lambda_i(1-\lambda_1)\cdots(1-\lambda_{i-1})$, for $i = 1, \ldots, k$; then

  $p_{k+1} = \lambda_{k+1}(1 - \sum_{j=1}^{k} p_j) = \lambda_{k+1}((1-\lambda_1)\cdots(1-\lambda_{k-1}) - p_k)$

  $= \lambda_{k+1}(((1-\lambda_1)\cdots(1-\lambda_{k-1}))(1-\lambda_k))$, so induction hypothesis is satisfied.

  b) The algorithm generates Us, rejecting each j with probability $(1-\lambda_j)$, for $j = 1, \ldots, n-1$, until accepting $j = n$ with probability λ_n, so probability of accepting n is $\prod_{j=1}^{n-1}(1-\lambda_j)\lambda_n = p_n$.

  c) If X is geometric then $p_j = pq^{j-1}$, with $q = 1-p$ so

  $\sum_{j=1}^{n-1} p_j = p(1 + q + \cdots + q^{n-2}) = p(1 - q^{n-1})/(1 - (1-p)) = (1 - q^{n-1})$,

  and therefore $\lambda_n = p_n/(1 - \sum_{j=1}^{n-1} p_j) = pq^{n-1}/q^{n-1} = p$.

  The algorithm rejects with probability q until n is accepted with correct probability $p_n = pq^{n-1}$.

- Problem 5.1: after integration $F(X) = (e^X - 1)/(e-1) = U$; solving for X, $X = \ln(1+(e-1)U)$.

- Problem 5.2: after integration of the first part $F(x) = (x-2)^2/4$ for $2 \le x < 3$,
  with $F(3) = 1/4$; the integral of second part is $3/4 - 3(2-x/3)^2/4$, so
  $F(x) = 1/4 + 3/4 - 3(2-x/3)^2/4 = 1 - 3(2-x/3)^2/4$, for $3 \le x \le 6$.
  If $U < 1/4$, invert first part, solving $U = (X-2)^2/4$, with $X - 2 = \sqrt{4U}$, so $X = 2 + 2\sqrt{U}$;
  otherwise solve $U = 1-3(2-X/3)^2/4$, with $2-X/3 = \sqrt{4(1-U)/3}$, so $X = 6-6\sqrt{(1-U)/3}$.

- Problem 5.3: inverting $U = (X^2+X)/2$, $X = (\sqrt{1+8U} - 1)/2$.

- Problem 5.5*: $F(x) = e^{2x}/2$, if $x < 0$, otherwise $F(x) = 1 - e^{-2x}/2$ with both parts easily inverted: if $U_1 < .5$, $X = \ln(2U_2)/2$, otherwise $X = -\ln(2(1-U_2))/2$.

- Problem 5.7: the method is similar to the discrete composition method (p. 61):
  first generate a discrete RV J using the pmf $P\{J = j\} = p_j$; then generate an $X \sim F_J(x)$.

- Problem 5.8:
  a) This has $p_1 = p_2 = p_3 = 1/3$, with $F_1 = x$, $F_2 = x^3$, $F_3 = x^5$, so
  if $U_1 < 1/3$, set $X = U_2$; if $U_1 \ge 2/3$, set $X = U_2^{1/5}$; otherwise set $X = U_2^{1/3}$.
  c) First use U_1 to generate a discrete RV I using the pmf $P\{I = i\} = \alpha_i$; then generate $X = U_2^{1/I}$.

- Problem 5.9*: $F(x)$ is a continuous composition of x^y cdfs, with exponential weight function e^{-y}.
  So first set $Y = -\ln(U_1)$ to get a random $F_Y(x) = x^Y$, then invert this to get $X = U_2^{1/Y}$.

- Problem 5.10: The simulation should use binomial(1000,.05) RVs to determine the number N of claims each month, then use $Exp(1/800)$ RVs to determine the amount of each claim. For each set of claims, determine if the total exceeds \$50000. Some Matlab for a simulation using $K = 1000$ months (with the binomial RV method from class lecture):

```
K = 1000; n = 1000; p =.05; q = 1-p; r(1) = q^n; js = [1:n+1]; % binomial RV setup
for j = 1 : n, r(j+1) = r(j)*p*(n-j+1)/(j*q); end, F = cumsum(r);  % binomial RV setup
for i = 1 : K % determine claim total for each month
  N = min( js(F>=rand) ) - 1; % binomial(1000,.05) RV
  X(i) = sum( -800*log(rand(1,N)) ); % total claims
end, disp( sum( X > 50000 )/K ) % display proportion > 50000
 0.109
```

  Approximately 11% of the months had total claims exceeding \$50000.

- Problem 5.16: you need the pdf $f(x) = e^{-x}(1+2e^{-x} - 3e^{-2x})$.
  a) the easiest algorithm uses AR with $g(x) = e^{-x}$, so $h(x) = f/g = (1+2e^{-x} - 3e^{-2x})$.
  Then $h' = e^{-x}(-2+6e^{-x}) = 0$ when $x^* = \ln(3)$, so rejection constant is $h(x^*) = 4/3 = c$.
  AR Algorithm:
  1) Generate U_1, U_2, set $X = -\ln(U_2)$;
  2) If $U_1 > \frac{f(X)}{cg(X)}$ goto 1), otherwise accept X.
  b) another algorithm could use AR with $g(x) = e^{-x/2}/2$, so $h(x) = f/g = 2e^{-x/2}(1+2e^{-x}-3e^{-2x})$.
  A graphical solution shows maximum $c \approx 1.77$ at $x^* \approx .64$, not as efficient as a).
  AR Algorithm:
  1) Generate U_1, U_2, set $X = -2\ln(U_2)$;
  2) If $U_1 > \frac{f(X)}{cg(X)}$ goto 1), otherwise accept X.
  Note: $F(r) = 1 - r - r^2 + r^3$, with $r = e^{-x}$, so an inversion algorithm could
  solve the cubic $F(r) = U$ for r and then use $X = -\ln(r)$.

- Problem 5.17:
  a) this is composition of $\frac{1}{4}(1)$, $\frac{1}{2}4x^3$ and $\frac{1}{4}5x^4$ with cdf's x, x^4, x^5.
  Algorithm:
    Generate U_1, U_2
    If $U_1 < .25$ then $X = U_2$;
    Elseif $U_1 < .75$ then $X = U_2^{\frac{1}{4}}$;
    Else $X = U_2^{\frac{1}{5}}$.
  b) simplest algorithm is AR with $g(x) = 1$, but $\max(f/g) = f(1) = 7/2$; not very efficient.

- Problem 5.19: a) use problem 5.4 $X = (\sqrt{1+8U} - 1)/2$;
  b) could use AR $g(x) = 1$, $f(x) = \frac{1}{2} + x$, with $c = \max(f/g) = f(1) = 3/2$;
  c) this is a composition of $\frac{1}{2}(x)$ and $\frac{1}{2}(x^2)$.
    Algorithm: Generate U_1, U_2; if $U_1 < .5$, $X = U_2$, otherwise $X = U_2^{\frac{1}{2}}$.
  b) might be fastest; on average all you need are 3 U's, and no sqrt's.

- Problem 5.20: for AR you need a $g(x)$ with a thicker tail, try $g(x) = e^{-x/2}/2$.
    Then $h(x) = f(x)/g(x) = e^{-x/2}(1+x)$; $h'(x) = e^{-x/2}(1-x)/2$, so maximum at $x = 1$,
      with $c = 2/\sqrt{e} \approx 1.21$, fairly efficient.
  AR Algorithm:
    1) Generate U_1, U_2, set $X = -2\ln(U_2)$;
    2) If $U_1 > \frac{f(X)}{cg(X)}$ goto 1), otherwise accept X.

- Problem 5.21*: a $[c, \infty)$ truncated Gamma has pdf $f(x) = Ax^{\alpha-1}e^{-x}$ for some constant A.
  A $[c, \infty)$ truncated Exp pdf is $g(x) = \lambda e^{-\lambda(x-c)}$, with $0 < \lambda \le 1$ so that g decays less rapidily.
  Now consider $h(x) = \frac{f}{g} = Ax^{\alpha-1}e^{-c\lambda}e^{-x(1-\lambda)}/\lambda$. This is a decreasing function for $x > c$, because
    $0 < \alpha < 1$ and $1 - \lambda \ge 0$, so the maximum is at $x^* = c$.
  Given $x^* = c$ you need to minimize $h(c)$ as function of λ. But $h(c) = B/\lambda$ for constant B,
    with minimum at $\lambda = 1$. So rejection constant $C = h(c) = Ac^{\alpha-1}e^{-c\lambda}$, and
    rejection function is just $h(x)/C = (x/c)^{\alpha-1}$.
  This $g(x)$ does not usually have mean α.

- Problem 5.22: following the example 5d use AR with $g(x) = 1$, $h(x) = f(x)/g(x) = 30x^2(1-x)^2$,
  with $h'(x) = 30x(1-x)(2-4x) = 0$ when $x^* = 1/2$; rejection constant $c = h(x^*) = 15/8 = 1.875$.
  This method, with ≈ 2 steps for each accept, is moderately efficient.

- Problem 5.23*: if you use AR with truncated uniform $g(x) = 5$,
    $h(x) = \frac{f}{g} = \frac{x(1-x)^3}{.00168}$, with max ≈ 19 at $x = .8$, so not efficient.
  So try to better match f with easily invertible $g = K(1-x)^3$; $\int_{.8}^1 g(x)dx = 1$ gives $\frac{1}{K} = .0004$.
  Now $h(x) = \frac{f}{g} = \frac{4x}{3.36}$, with maximum value $c = 4/3.36 \approx 1.19$ at $x = 1$, so AR is very efficient.
  To set up the AR algorithm you need $G(x) = \int_{.8}^x \frac{(1-x)^3}{.0004}dx = 1 - \frac{(1-x)^4}{.0016}dx$,
    with the inversion formula $X = 1 - (.0016(1-U))^{\frac{1}{4}}$; also notice $\frac{f}{cg} = x$.
  AR Algorithm:
    1) Generate U_1, U_2, and set $X = 1 - (.0016(1-U_2))^{\frac{1}{4}}$;
    2) If $U_1 > X$ goto 1), otherwise accept X.

- Problem 5.24: $h(x) = f/g = 2e^{-x^2/2+\lambda x}/(\sqrt{2\pi}\lambda)$. h is maxmized when $h' = (-x+\lambda)h = 0$ at $x = \lambda$, so $c(\lambda) = 2e^{\lambda^2/2}/(\sqrt{2\pi}\lambda)$. $c' = (\lambda - 1/\lambda)c = 0$ when $\lambda = \pm 1$ with minimum at $\lambda = 1$.

- Problem 5.29: interarrival times are $\sim Exp(5)$, and bus sizes are uniform 20-40; Matlab:

```
K = 10000; lam = 5;
for i = 1 : K, t=-log(rand)/lam; X = 0; % X is fan total
  while t <= 1, X = X+20+floor(21*rand); t=t-log(rand)/lam; end, F(i) = X;
end, disp( mean(F) ) % display average # of fan arrivals
 149.66
```

- Problem 6.5*: the covariance matrix Cholesky decomposition is (p.100)

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sigma_2\sqrt{1-\rho^2} \end{bmatrix} \begin{bmatrix} \sigma_1 & \rho\sigma_2 \\ 0 & \sigma_2\sqrt{1-\rho^2} \end{bmatrix} = AA'.$$

  If you let $\mathbf{x} = (x_1, x_2)'$, with $\mu = (\mu_1, \mu_2)'$, the bivariate Normal cdf is

$$B(X_1, X_2) = \int_{-\infty}^{X_1} \int_{-\infty}^{X_2} \frac{e^{-(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)/2}}{2\pi\sqrt{|\Sigma|}} dx_2 dx_1.$$

  Then make the change of variables $\mathbf{x} = A\mathbf{y} + \mu$, so that
  $y_1 = (x_1 - \mu_1)/\sigma_1$, $y_2 = (x_2 - \mu_2 - \rho\sigma_2 y_1)/(\sigma_2\sqrt{1-\rho^2})$, with

$$B(X_1, X_2) = \int_{-\infty}^{(X_1-\mu_1)/\sigma_1} \frac{e^{-y_1^2/2}}{\sqrt{2\pi}} \int_{-\infty}^{(X_2-\mu_2-\rho\sigma_2 y_1)/(\sigma_2\sqrt{1-\rho^2})} \frac{e^{-y_2^2/2}}{\sqrt{2\pi}} dy_2 dy_1.$$

  Scaling and shifting y_2 produces the required result.

- Problem 6.6: first find Cholesky decomposition of

$$C = \begin{bmatrix} 3 & -2 & 1 \\ -2 & 5 & 3 \\ 1 & 3 & 4 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix} \begin{bmatrix} a & b & d \\ 0 & c & e \\ 0 & 0 & f \end{bmatrix} = AA'.$$

  So $a = \sqrt{3}$, $b = \frac{-2}{\sqrt{3}}$, $d = \frac{1}{\sqrt{3}}$, $c = \sqrt{5-b^2} = \frac{\sqrt{11}}{3}$, $e = \frac{3-bd}{c} = \sqrt{\frac{11}{3}}$, $f = \sqrt{4-d^2-e^2} = 0$.
  Algorithm computes $X = AZ+\mu$, with $\mu = (1,2,3)'$ and $Z = (Z_1, Z_2, Z_3)'$ with $Z_i \sim Normal(0,1)$, but you only need Z_1, Z_2 for each X because $f = 0$.

- Problem 6.7: this is not very clear.
  If you assume the bivariate joint distribution is $H(X, Y)$, with marginals $F(X)$, $G(Y)$, then $C_{X,X}$ could mean $C_{X,X}(x, x) = P(F(X) \leq x, F(X) \leq x) = G(F^{-1}(X), F^{-1}(X))$.