STATISTICAL COMPUTATIONAL METHODS

Seminar Nr. 3

Computer Simulations of Random Variables and Monte Carlo Studies; Inverse Transform Method, Rejection Method, Special Methods

1.

a) Use the DITM to generate a $Geo(p), p \in (0,1)$, variable.

b) Then use that to generate a $NB(n,p), n \in \mathbb{N}, p \in (0,1)$, variable.

Solution:

We used the DITM in Example 3.4. (Lecture 3) to generate a $SGeo(p)$ and a $Geo(p), p \in (0,1)$, variable. We found

$$X = \left\lceil \frac{\ln(U)}{\ln(1-p)} - 1 \right\rceil.$$

(equation (3.7) in Lecture 3)

So, now let's implement it. We will follow the same procedure as before, first generate *one* variable, and then a sample, from which we make certain estimates. In Matlab the function log stands for ln (and $loga$ stands for log_a).

```
% Simulate Geometric distr. Geo(p) variables, using the Discrete
% Inverse Transform method, i.e. X = ceil(ln(U)/ln(1 - p)-1).
clear all
p = input('p (in (0,1)) = '); % the parameter of the Geo distr.
% Generate one variable
X = ceil(log(1 - rand)/log(1 - p) - 1);
```

Now, to generate a sample, we will use an appropriate number of simulations, in order to get a certain accuracy.

Given a tolerable error $\varepsilon > 0$ and a significance level (probability of error) $\alpha \in (0,1)$, in order to have

$$P\big(|\overline{p} - p| > \varepsilon\big) \leq \alpha,$$

or, equivalently,

$$P\big(|\overline{p} - p| < \varepsilon\big) \geq 1 - \alpha,$$

we determined that the the size of the MC study, N, should be

$$N \geq \frac{1}{4}\left(\frac{z_{\alpha/2}}{\varepsilon}\right)^2,$$

where $z_{\alpha/2}$ is the quantile (inverse of the cdf $\Phi = F_{N(0,1)}$) of order $\alpha/2$ for the $N(0,1)$ distribution.

```
% Simulate Geometric distr. Geo(p) variables, using the Discrete
% Inverse Transform, i.e. X = ceil(ln(U)/ln(1 - p)-1).
clear all
p = input('p (in (0,1)) = '); % the parameter of the Geo distr.
% Generate one variable
% X = ceil(log(1 - rand)/log(1 - p) - 1);

err = input('error = '); % maximum error
alpha = input('alpha (level of significance) = '); % sign. level
% Generate a sample of such variables
N =  ceil(0.25*(norminv(alpha/2,0,1)/err)^2); % MC size to ensure
% that the error is < err, with confidence level 1 - alpha
fprintf('Nr. of simulations N = %d \n\n', N)
```

It would be a good idea to track the number of simulations and proceed with caution! For instance, an error $\varepsilon = 1e - 3$ and a significance level $\alpha = 0.01$ produce a number of $N = 1658725$ simulations! However, α should never be bigger than 0.05 and the error should be at least $1e - 2$ (so, you should try $\varepsilon = 1e - 2, 5e - 3, 1e - 3, 5e - 4$, $\alpha = 0.05, 0.01, 0.001$).

Now, that we generated one variable and we determined the number of simulations, generate a sample.

```
X = zeros(1, N);
for i = 1 : N
    X(i) = ceil(log(1 - rand)/log(1 - p) - 1); % the Geo variables
end
```

Like last time, compare the estimates with the true values.

```
% Application/Comparison
fprintf('simulated probab. P(X = 2) = %1.5f\n', mean(X == 2))
fprintf('true probab. P(X = 2) = %1.5f\n', geopdf(2, p))
fprintf('error = %e\n\n', abs(geopdf(2, p) - mean(X == 2)))

fprintf('simulated probab. P(X <= 2) = %1.5f\n', mean(X <= 2))
fprintf('true probab. P(X <= 2) = %1.5f\n', geocdf(2, p))
fprintf('error = %e\n\n', abs(geocdf(2, p) - mean(X <= 2)))

fprintf('simulated probab. P(X < 2) = %1.5f\n', mean(X < 2))
fprintf('true probab. P(X < 2) = %1.5f\n', geocdf(1, p))
fprintf('error = %e\n\n', abs(geocdf(1, p) - mean(X < 2)))

fprintf('simulated mean E(X) = %5.5f\n', mean(X))
fprintf('true mean E(X) = %5.5f\n', (1 - p)/p)
fprintf('error = %e\n\n', abs((1 - p)/p - mean(X)))
```

For part **b)**, use again the fact that a Negative Binomial $NB(n,p)$ variable is the sum of n independent Geometric $Geo(p)$ variables. Take advantage of Matlab and generate *all* n $Geo(p)$ variables at once! For one variable:

```
% Simulate distr.  NBin(n, p) variables, using the Discrete
% Inverse transform method.
clear all
n = input('n (in N) = '); %
p = input('p (in (0,1)) = '); % the parameters of the NBin distr.
% A NBin(n,p) variable is the sum of n indep. Geo variables (and
% repr. the number of failures occurred before the nth success)

% Generate one variable
Y = ceil(log(1 - rand(n, 1))/log(1 - p) - 1);
X = sum(Y);
```

Then put it in a "for" loop to generate N variables. For the comparison, DO NOT forget to change the name of the distribution "geo" to "nbin", the parameters (two, n and p) and the expected value $E(X) = \dfrac{nq}{p}$.

2.

a) Use the ITM to generate an $Exp(\lambda), \lambda > 0$, variable.

b) Then use that to generate a $Gam(\alpha, \lambda), \alpha \in \mathbb{N}, \lambda > 0$, variable (a Gamma $Gam(\alpha, \lambda)$ variable is the sum of α independent $Exp(1/\lambda)$ variables).

Solution:

For an $Exp(\lambda), \lambda > 0$, variable, the ITM yields

$$X = -\frac{1}{\lambda} \ln(U).$$

(Example 3.3., formula (3.4) in Lecture 3)

So, first, generate just one variable and then a sample of N such variables. Recall that "our" parameter λ is $\frac{1}{\lambda}$ in Matlab!

```
X = zeros(1, N);
for i = 1 : N
    X(i) = -1/lambda*log(rand); % the Exp variables
end
```

For the last part, the comparison with true values, the expected value is $\frac{1}{\lambda}$.

Also, since now we simulate a *continuous* variable, it will take *single* values with probability ZERO! (so omit that part in the comparison)

```
% Application/Comparison

fprintf('simulated probab. P(X <= 2) = %1.5f\n', mean(X <= 2))
fprintf('true probab. P(X <= 2) = %1.5f\n', expcdf(2, 1/lambda))
fprintf('error = %e\n\n', abs(expcdf(2, 1/lambda) - mean(X <= 2)))

fprintf('simulated probab. P(X < 2) = %1.5f\n', mean(X < 2))
fprintf('true probab. P(X < 2) = %1.5f\n', expcdf(2, 1/lambda))
fprintf('error = %e\n\n', abs(expcdf(2, 1/lambda) - mean(X < 2)))

fprintf('simulated mean E(X) = %5.5f\n', mean(X))
fprintf('true mean E(X) = %5.5f\n', 1/lambda)
fprintf('error = %e\n\n', abs(1/lambda - mean(X)))
```

For part **b)**, for $a \in \mathbb{N}$ a $Gamma(a, \lambda)$ variable is the sum of a independent $Exp(1/\lambda)$ variables.

```
X = zeros(1, N);
for i = 1 : N
    X(i) = sum(-lambda*log(rand(a, 1))); % the Gamma variables
end
```

3. Use a special method to generate a $Poiss(\lambda), \lambda > 0$, variable.

<u>**Solution:**</u>

The special method for Poisson variables uses the relationship with Exponential variables. A $Poiss(\lambda)$ variable counts the number of "rare" events that occur during one unit of time when the time elapsed between any two such events has $Exp(\lambda)$ distribution.

$$X \;=\; \max\{n \mid U_1 \cdot \ldots \cdot U_n \geq e^{-\lambda}\}.$$

Recall Algorithm 5.1 (Lecture 4)

Algorithm

1. Generate $U_1, U_2, \ldots \in U(0,1)$.
2. $X = \max\{n \mid U_1 \cdot U_2 \cdot \ldots \cdot U_n \geq e^{-\lambda}\}$.

```
% Generate Poisson distr. P(lambda), using a special method
% (related to the Exp distr.).
clear all
lambda = input('lambda ( > 0) = '); % param. of the Poisson distr.
% Generate one variable
U = rand; % generated U(0,1) variable
X = 0; % initial value
while U >= exp(- lambda) % check that U1*...*Un >= exp(-lambda),
                         % to get the max n
    U = U * rand; % go further to n + 1 (i.e. X + 1)
    X = X + 1; % the Poisson variable
end
```

Generate N such variables and do the comparison (the mean of the Poisson variable is λ). Here, again we can look at probabilities of the type $P(X = 2)$, since this is a discrete random variable.

4. Use the rejection method to approximate π (see Example 7.2, Lecture 4).

Solution:

From Example 7.2, Lecture 4, we have the algorithm

Algorithm

1. Generate $X_1, \dots, X_N, Y_1, \dots, Y_N \in U(-1, 1)$.
2. Compute the number of pairs (X_i, Y_i) for which $X_i^2 + Y_i^2 \le 1$, say N_π.
3. Approximate $\pi \approx 4\dfrac{N_\pi}{N}$.

In Matlab π is "pi".

5. Application: Forecasting for new software release

An IT company is testing a new software to be released. Every day, software engineers find a random number of errors and correct them. On each day t, the number of errors found, X_t, has a Poisson(λ_t) distribution, where the parameter λ_t is the lowest number of errors found during the previous k days,

$$\lambda_t = \min\{X_{t-1}, X_{t-2}, \dots, X_{t-k}\}.$$

If some errors are still undetected after $tmax$ days (i.e. if not all errors are found in $tmax - 1$ days), the software is withdrawn and goes back to development. Generate a Monte Carlo study to estimate

a) the time it will take to find all errors;

b) the total number of errors found in this new release;

c) the probability that the software will be sent back to development.

(Try $k = 4, [X_{t-1}, X_{t-2}, X_{t-3}, X_{t-4}] = [10, 5, 7, 6], tmax = 10.$)

Solution:

```
% Probl. 5, Sem.3, Forecasting errors in new software release.
clear all
err = input('error  = '); % maximum error
alpha = input('alpha (level of significance) = '); % sign. level
% Generate a sample of  variables
N =  ceil(0.25*(norminv(alpha/2,0,1)/err)^2); % MC size to ensure
% that the error is < err, with confidence level 1 - alpha
fprintf('Nr. of simulations N = %d \n\n', N)
```

```
k = input('number of previous days considered = ');
in_last = input('numbers of errors in the last k days
(vector of length k) = '); % initial number of errors in
                           % the last k days
tmax = input('max time after which the new software is
 withdrawn (in days) = ');
```

```
Ttotal = zeros(1, N);
% Ttotal is the time it takes to find all the errors (in days)
Ntotalerr = zeros(1, N);
% Ntotalerr is the total number of errors that are detected
```

We need to keep track of time T, of number of errors on day T, number of errors detected so far and the numbers of errors in the last k days.

```
for i = 1 : N
    % T is time from now on (in days), X is nr. of errors on day T
    % nrerr is the number of errors detected so far

    T = 0;
    last = in_last; % number of errors in the last k days
    X = in_last(k); % nr. of errors in day T
    nrerr = sum(in_last);
```

While there still are errors, we generate the new number of errors, as a Poisson variable with parameter $\lambda_t = \min\{X_{t-1}, X_{t-2}, \ldots, X_{t-k}\}$. We use the special method discussed previously.

```
while X > 0; % while loop until no errors are found
        lambda = min(last); % parameter for var X
        % Simulate the number of errors on day T,
        % Poisson (lambda), special method
        U = rand; % generated U(0,1) variable
        X = 0; % initial value
        while U >= exp(- lambda);
            U = U * rand;
            X = X + 1; % the Poisson variable
        end;
```

Then we update.

```
        T = T + 1; % update: next day
        nrerr = nrerr + X; % update: new nr. of errors
        last = [last(2:k), X]; % update: new nrs of errors
                       % in the last k days
    end;  % the while loop ends when X = 0 on day T, that means
          %  that all errors were found on previous day, T - 1
Ttotal(i) = T - 1; % the day all errors were found
Ntotalerr(i) = nrerr; % total nr. of errors found
end; % end of for
```

Last, we get our estimates.

```
fprintf('\n')
fprintf('a) The time it will take to find all errors is
         %3.3f days \n', mean(Ttotal))
fprintf('b) Total number of errors in the new release is
         %5.3f \n', mean(Ntotalerr))
fprintf('c) Prob. that some errors will still be undetected
         after %d days, \n', tmax)
fprintf('after which the software will be withdrawn is
         %3.3f \n\n', mean(Ttotal > tmax))
```

Here are several runs:

```
>> problem5_sem3_forecasting_errors
error  = 5e-3
alpha (level of significance) = 0.01
Nr. of simulations N = 66349

number of previous days considered = 4
numbers of errors in the last k days (vector of length k) =
[10, 5, 7, 6]
max time after which the new software is withdrawn (in days) = 10

a) The time it will take to find all errors is 6.984 days
```

```
b) Total number of errors in the new release is 52.900
c) Prob. that some errors will still be undetected after 10 days,
after which the software will be withdrawn is 0.181

>> problem5_sem3_forecasting_errors
error  = 1e-3
alpha (level of significance) = 0.01
Nr. of simulations N = 1658725

number of previous days considered = 4
numbers of errors in the last k days (vector of length k) =
[10, 5, 7, 6]
max time after which the new software is withdrawn (in days) = 10

a) The time it will take to find all errors is 6.971 days
b) Total number of errors in the new release is 52.855
c) Prob. that some errors will still be undetected after 10 days,
after which the software will be withdrawn is 0.179
```

This Monte Carlo study should predict an expected time of about 7 days to detect all the errors, about 53 errors overall and a probability of approximately 0.18 that errors remain after 10 days.