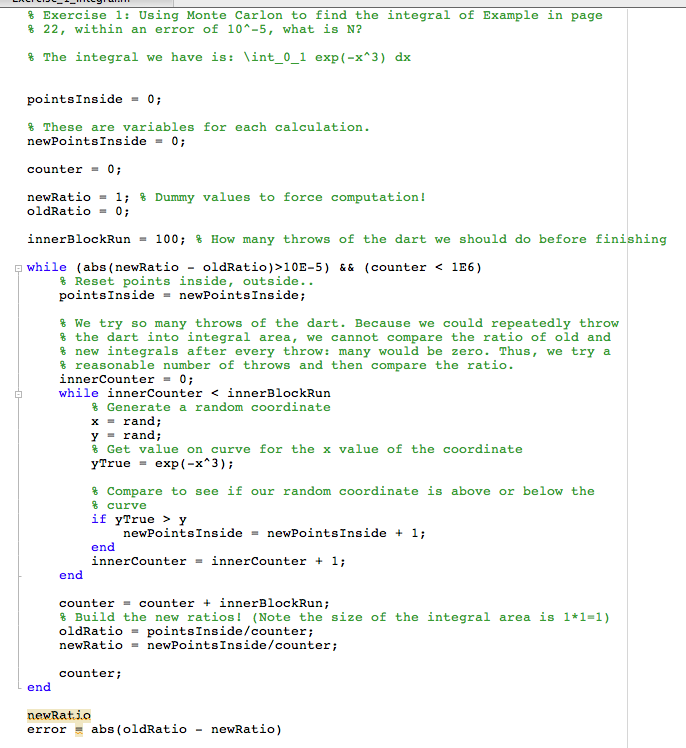
MATH 6060

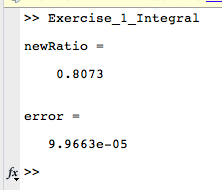
Vance Turnewitsch

Chapter 2 Homework

Exercise 1: Below we present the code for this exercise.



Next we present the results from running the code and find N=682000



Exercise 2:

Part A and B: First, we present our code: The code was too long for a single screenshot, thus we copy and paste:

% Exercise 2.2: Run MC Routine in page 23. Then write your own routine for

% the following cases.

% Routine 2.2

N = 10000; % Numer of trials

m = 0; % Mean failure time

m2 = 0; % The second moment

% We calculate standard deviation by using:

% (std dev)^2 = E[X^2] - E[X]

% Thus we simply average up the X^2 at each trial.

for i = 1:N

% randn here generates normally distributed values with mean 0 and

% std dev 1. Thus our mean times of device failures are 11, 12, 13

% years with std dev of 1 2 and 3 years. Thus we are effectively

% creating making our mean failure times normally distributed.

x = 11 + randn;

y = 12 + 2\*randn;

z = 13 + 3\*randn;

w = min([x,y,z]); % Get the minimum failure time of the components.

m = m + w/N; % Add trial to mean fail time, note true mean is: (t1 + t2 + t3 + ... tn)/N

m2 = m2 + w\*w/N; % Add trial to second moment

end;

disp('Mean and Standard Deviation for book code');

m

sd = sqrt(m2 - m\*m)

% Case A: There are five devices T1, ...,T5 which have normal distributions

% with mean 11,12,13,15,16, and standard deviation 1,2,3,4,5, respectively.

% Solution: We reuse the approach above but use more devices and varying

% means and std dev.

N = 10000

m = 0;

m2 = 0;

for i = 1:N

% Generate all the failure times.

t1 = 11 + randn;

t2 = 12 + 2\*randn;

t3 = 13 + 3\*randn;

t4 = 14 + 4\*randn;

t5 = 15 + 5\*randn;

% Determine which device caused us to fail

w = min([t1,t2,t3,t4,t5]);

% Alter the mean based on this failure time

m = m + w/N;

% Alter the Expected value of our failure times squared

m2 = m2 + w\*w/N;

end

disp('Mean and Standard Deviation for Case A');

m

sd = sqrt(m2 - m\*m) % std dev = sqrt{E[X^2] - mean^2}

% Case B: Repeat (a) with uniform distribution instead.

% Solution: We use rand which generates uniformly distributed random

% numbers on the interval 0 to 1. Thus their mean is: (0+1)/2 = 0.5. Their

% variance is (1-0)^2/12) = 1/12. Std dev is thus 1/sqrt{12} = 1/(2sqrt{3})

% We can transform this distribution into what we want by the following:

%

% (b-a)X + a ~ U(a,b) where a and b are the mean and std dev.

N = 10000;

m = 0;

m2 = 0;

for i = 1:N

% Generate uniformly distributed failure times

t1 = (1-11)\*rand + 11;

t2 = (2-12)\*rand + 12;

t3 = (3-13)\*rand + 13;

t4 = (4-14)\*rand + 14;

t5 = (5-15)\*rand + 15;

% Determine which device failued

w = min([t1,t2,t3,t4,t5]);

% Alter mean

m = m + w/N;

% Alter expected value of failuter times squared

m2 = m2 + w\*w/N;

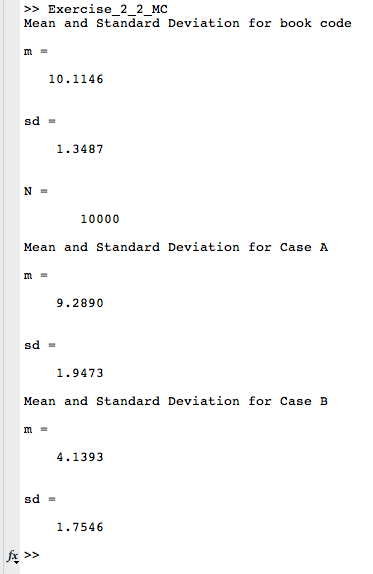
end

disp('Mean and Standard Deviation for Case B');

m

sd = sqrt(m2 - m\*m)

We now present the results from running the code above:



Exercise 3: Again, the code was unwieldy for a screenshot, thus we copy and paste:

% Exercise 2.3: Write a MC routine to compute (see Routine 2.1 in page 25)

% E[X] and sigma^2[X] with given CDF of X. Then test it using exponential

% distribution with various lambda. Run also Routine 2.1 in page 25.

% Compare the difference in results.

% Solution: We assume our given CDF is strictly increasing whenever it is

% nonzero and less than one. Then given the mean of this CDF we can find

% distributed values for it by the following:

%

% X = -aln(1-Y) where a is the mean and Y is a uniformly distributed

% variable on the interval: [0,1]

%

% We note that the exponential distribution is strictly increasing whenever

% it is nonzero and less than one. Thus, we can use our transformation if

% given the mean of the distribution.

N = 1E6;

meanCalc = 0;

moment = 0;

maxLambda = 1;

for lambda = 0.1:0.1:maxLambda

mean = 1/lambda;

meanCalc = 0;

moment = 0;

for i = 1:N

% Get uniformly distributed value

y = rand;

% Transform this uniformly distributed value into another form

x = -mean\*log(1-y);

% Determine mean

meanCalc = meanCalc + x/N;

% Determine moment used for calculating variance

moment = moment + x\*x/N;

end

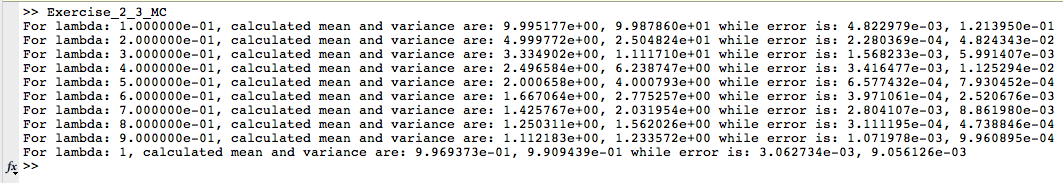
variance = moment - meanCalc\*meanCalc;

txt = sprintf('For lambda: %d, calculated mean and variance are: %d, %d while error is: %d, %d',lambda,meanCalc,variance,abs(meanCalc-mean),abs(variance-1/lambda^2));

disp(txt);

end

We present our results below:



Notice the error is quite small.

Exercise 4: Again, we present our code as copy and paste since it is too long for a screenshot:

% Exercise 2.4: Write a Matlab routine to generate N(0,1) (e.g., find mean

% and variance) using "rand", and then compare your results with "randn".

% Test also your results in 2(a).

% Solution: Our usual solution of using the inverse transformation isn't

% going to work simply because we have here a rather nasty cumulative

% distribution function for the normal distribution.

%

% We thus proceed to use the Box-Muller transformation:

% Given two uniformly distributed random variables: u1, u2 we can find to

% normaly distributed random variables:

%

% z1 = sqrt{-2\*ln(u1)}\*cos(2\*pi\*u2)

% z2 = sqrt{-2\*ln(u1)}\*sin(2\*pi\*u2)

N = 1E7;

mean = 0;

moment = 0;

% The calculated mean and moments using randn function

meanRN = 0;

momentRN = 0;

for i = 1:N

% Generate to uniformly distributed random numbers

u1 = rand;

u2 = rand;

% Transform to normaly distributed random numbers via Box-Muller

z1 = sqrt(-2\*log(u1))\*cos(2\*pi\*u2);

z2 = sqrt(-2\*log(u1))\*sin(2\*pi\*u2);

% Alter mean

mean = mean + z1/N;

% Alter moment

moment = moment + z1\*z1/N;

% Now do it again using the randn function

x = randn;

meanRN = meanRN + x/N;

momentRN = momentRN + x\*x/N;

end

variance = moment - mean\*mean;

varianceRN = momentRN - meanRN\*meanRN;

txt = sprintf('Calculated mean, variance vs randn mean variance:(%d,%d) vs (%d,%d)',mean,variance,meanRN,varianceRN);

txt2 = sprintf('Error using transformation: (%d,%d)',abs(mean-meanRN),abs(variance-varianceRN));

disp(txt);

disp(txt2);

% Now we test our results from 2A. We simply transform the results. :)

% We copy over the code from 2A but now use our code from above (slightly

% alterted) generate the normal numbers.

N = 1E4;

m = 0;

m2 = 0;

mRN = 0;

m2RN = 0;

for i = 1:N

% Generate all the failure times.

t1 = 11 + boxMullerTrans;

t2 = 12 + 2\*boxMullerTrans;

t3 = 13 + 3\*boxMullerTrans;

t4 = 14 + 4\*boxMullerTrans;

t5 = 15 + 5\*boxMullerTrans;

% Determine which device caused us to fail

w = min([t1,t2,t3,t4,t5]);

% Alter the mean based on this failure time

m = m + w/N;

% Alter the Expected value of our failure times squared

m2 = m2 + w\*w/N;

% Now we do it again using randn this time for comparison

% Generate all the failure times.

t1 = 11 + randn;

t2 = 12 + 2\*randn;

t3 = 13 + 3\*randn;

t4 = 14 + 4\*randn;

t5 = 15 + 5\*randn;

% Determine which device caused us to fail

w = min([t1,t2,t3,t4,t5]);

% Alter the mean based on this failure time

mRN = mRN + w/N;

% Alter the Expected value of our failure times squared

m2RN = m2RN + w\*w/N;

end

disp('Mean failure time results>');

m;

sd = sqrt(m2 - m\*m); % std dev = sqrt{E[X^2] - mean^2}

sdRN = sqrt(m2RN - mRN\*mRN);

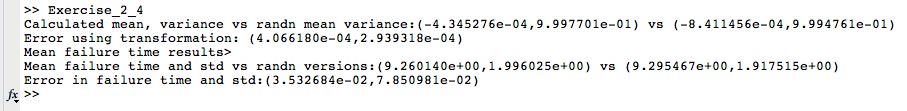
txt = sprintf('Mean failure time and std vs randn versions:(%d,%d) vs (%d,%d)',m,sd,mRN,sdRN);

txt2 = sprintf('Error in failure time and std:(%d,%d)',abs(m-mRN),abs(sd-sdRN));

disp(txt);

disp(txt2);

Now we present the results from running the code:



Notice we are quite close!