%=========================================================================

%

% Program to demonstrate the Law of Large Numbers

% (Exponential distribution example)

%

%=========================================================================

clear all

clc

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12) )

mu = 5;

t = 500;

% Generate exponential random numbers from a Gamma distribution

e = randg(1,[t 1]);

y = mu.\*e;

% Generate sample means from sample of size t=1,2,3,...,tmax

ybar = zeros(t,1);

for i = 1:t

ybar(i) = mean(y(1:i));

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*

%\*\* Generate graph

%\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

tt = (1:1:t)';

plot(tt,mu\*ones(t,1),'-k',...

tt,ybar,'-k',...

tt,4.80\*ones(t,1),':k',...

tt,5.20\*ones(t,1),':k',...

'LineWidth',0.75);

ylabel('$\overline{y}\_T$');

xlabel('T');

set(gca,'XTick',0:100:500);

axis( [0 500 3 7 ]);

set(gca,'YTick',3:1:7);

%laprint(1,'wln','options','factory');

%=========================================================================

%

% Program to demonstrate by simulation the necessary and sufficient

% conditions of the weak law of large numbers

%

%=========================================================================

clear all

clc

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12) )

% Choose parameters

t = 100;

n = 50000;

m1 = zeros(n,1);

m2 = zeros(n,1);

m3 = zeros(n,1);

m4 = zeros(n,1);

for i = 1:n

y = rand(t,1) - 0.5; % Simulate y from a uniform distribution (-0.5,0.5).

%y = 2 + trnd(3, [t 1]); % Simulate Student t (mue,dof=3) random numbers

m1(i) = sum(y.^1)/t;

m2(i) = sum(y.^2)/t;

m3(i) = sum(y.^3)/t;

m4(i) = sum(y.^4)/t;

end

format ShortG

disp( [' Sample size = ' num2str(t) ] );

disp( ' ' );

disp( ['Mean of m1 = ' num2str(mean(m1)) ] );

disp( ['Variance of m1 = ' num2str(std(m1)^2) ] );

disp( ' ' );

disp( ['Mean of m2 = ' num2str(mean(m2)) ] );

disp( ['Variance of m2 = ' num2str(std(m2)^2) ] );

disp( ' ' );

disp( ['Mean of m3 = ' num2str(mean(m3)) ] );

disp( ['Variance of m3 = ' num2str(std(m3)^2) ] );

disp( ' ' );

disp( ['Mean of m4 = ' num2str(mean(m4)) ] );

disp( ['Variance of m4 = ' num2str(std(m4)^2) ] );

%=========================================================================

%

% Program to demonstrate by simulation the necessary and sufficient

% conditions of the weak law of large numbers

%

%=========================================================================

clear all

clc

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12) )

% Choose parameters

t = 100;

n = 50000;

m1 = zeros(n,1);

m2 = zeros(n,1);

m3 = zeros(n,1);

m4 = zeros(n,1);

for i = 1:n

y = rand(t,1) - 0.5; % Simulate y from a uniform distribution (-0.5,0.5).

%y = 2 + trnd(3, [t 1]); % Simulate Student t (mue,dof=3) random numbers

m1(i) = sum(y.^1)/t;

m2(i) = sum(y.^2)/t;

m3(i) = sum(y.^3)/t;

m4(i) = sum(y.^4)/t;

end

format ShortG

disp( [' Sample size = ' num2str(t) ] );

disp( ' ' );

disp( ['Mean of m1 = ' num2str(mean(m1)) ] );

disp( ['Variance of m1 = ' num2str(std(m1)^2) ] );

disp( ' ' );

disp( ['Mean of m2 = ' num2str(mean(m2)) ] );

disp( ['Variance of m2 = ' num2str(std(m2)^2) ] );

disp( ' ' );

disp( ['Mean of m3 = ' num2str(mean(m3)) ] );

disp( ['Variance of m3 = ' num2str(std(m3)^2) ] );

disp( ' ' );

disp( ['Mean of m4 = ' num2str(mean(m4)) ] );

disp( ['Variance of m4 = ' num2str(std(m4)^2) ] );

%=========================================================================

%

% Program to demonstrate Slutsky's theorem by simulation

%

%=========================================================================

clear all

clc

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12) )

% Choose parameters

t = 10;

n = 50000;

mu = 2;

m1 = zeros(n,1);

m2 = zeros(n,1);

for i = 1:n

% Generate exponential random numbers from a Gamma distribution

y = mu\*randg(1,[t 1]);

m1(i) = ( mean(y)/std(y) )^2;

m2(i) = ( sqrt(t)\*mean(y) )^2;

end

format ShortG

disp( [' Sample size = ' num2str(t) ] );

disp( ' ' );

disp( ' Moment results for m1 ');

disp( ['Mean of m1 = ' num2str(mean(m1)) ] );

disp( ['Variance of m1 = ' num2str(std(m1)^2) ] );

disp( ' ' );

disp( ' Moment results for m2 ');

disp( ['Mean of m2 = ' num2str(mean(m2)) ] );

disp( ['Variance of m2 = ' num2str(std(m2)^2) ] );

%=========================================================================

%

% Program to compute gradient, Hessian, information matrix and OPG

%=========================================================================

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12) )

t = 5000;

% Normal distribution

mu = 1;

sig2 = 1;

u = randn(t,1);

y = mu + sqrt(sig2 )\*u;

m = mean(y); % maximum likelihood estimate

g = (y - m)/sig2; % gradient at t

h = -1/sig2; % hessian at t

j = g'\*g; % opg

disp(' Normal distribution results ')

disp(['Gradient vector = ',num2str(mean(g)) ]);

disp(['Hessian = ',num2str(mean(h)) ]);

disp(['Information matrix = ',num2str(-mean(h)) ]);

disp(['OPG matrix = ',num2str(j/t) ]);

disp(' ')

% Exponential distribution

theta = 2;

y = theta\*gamrnd(1,1,[t 1]);

th = 1/mean(y); % maximum likelihood estimate

g = 1/th - y; % gradient at t

h = -1/th^2; % hessian at t

j = g'\*g; % opg

disp(' Exponential distribution results ')

disp(['Gradient vector = ',num2str(mean(g)) ]);

disp(['Hessian = ',num2str(mean(h)) ]);

disp(['Information matrix = ',num2str(-mean(h)) ]);

disp(['OPG matrix = ',num2str(j/t) ]);

disp(' ')

%=========================================================================

%

% Program to demonstrate two aspects of consistency

%

%=========================================================================

clear all

clc

RandStream.setDefaultStream( RandStream('mt19937ar','seed',66) )

mu = 0:0.1:20;

sig = 4.0;

lnl1 = zeros( 0,length(mu));

lnl2 = zeros( 0,length(mu));

lnl3 = zeros( 0,length(mu));

% Population parameters

mu\_0 = 10.0;

sig\_0 = 4.0;

% Sample T=5

t = 5;

y = mu\_0 + sig\_0\*randn(t,1);

disp( ['Sample mean (T=5) =' num2str(mean(y)) ]);

for i = 1:length(mu)

lnl1(i) = mean(-0.5\*log(2\*pi\*sig^2) - 0.5\*(y-mu(i)).^2./sig^2);

end

% Sample T=20

t = 20;

y = mu\_0 + sig\_0\*randn(t,1);

disp( ['Sample mean (T=20) =' num2str(mean(y)) ]);

for i = 1:length(mu)

lnl2(i) = mean(-0.5\*log(2\*pi\*sig^2) - 0.5\*(y-mu(i)).^2./sig^2);

end

% Sample T=500

t = 500;

y = mu\_0 + sig\_0\*randn(t,1);

disp( ['Sample mean (T=500) =' num2str(mean(y)) ]);

for i = 1:length(mu)

lnl3(i) = mean(-0.5\*log(2\*pi\*sig^2) - 0.5\*(y-mu(i)).^2./sig^2);

end

% Compute population log-likelihood

e\_lnl = -0.5\*log(2\*pi\*sig\_0^2) - 0.5 - 0.5\*(mu - mu\_0).^2/sig\_0^2;

tmp = -0.5\*(log(2\*pi\*sig\_0^2) + 1);

disp( ['Maximum value at theta = theta\_0 =' num2str(tmp) ] );

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*

%\*\*\* Generate graph

%\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

plot(mu,e\_lnl,'-k',mu,lnl1,':k',mu,lnl2,'-.k',mu,lnl3,'--k','LineWidth',0.75);

axis([2 15 -3.5 -2.5])

ylabel('$A(\theta)$')

xlabel('$\mu$')

% Print TEX file

laprint(1,'twotypesconsistency','options','factory');

%=========================================================================

%

% Program to demonstrate the consistency property of MLE for the

% mean of the normal distribution.

%

%=========================================================================

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',123) )

t = 500;

mu = 1; % Population mean

sig2 = 2; % Population variance

% Generate sample means from sample of size t=1,2,3,...,t

yb = zeros(t,1);

for t = 1:t

y = mu + sqrt(sig2)\*randn(t,1);

yb(t) = mean(y);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*

%\*\* Generate graph

%\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

plot(1:1:t,mu.\*ones(t,1),'-k',1:1:t,yb,'-k');

ylabel('$\bar{y}$');

xlabel('T');

box off;

axis tight;

laprint(1,'normconsist','options','factory');

%=========================================================================

%

% Program to demonstrate the consistency property of MLE for the

% location parameter (theta) of the Cauchy distribution.

%

% For this example the median is the MLE and is thus a consistent estimator

% but the sample mean is an inconsistent estimator.

%

%=========================================================================

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',1234) )

theta = 1; % Loaction parameter

nu = 1; % Cauchy = Student t nu =1 dof

t = 500;

% Generate sample means and medians from sample of size t=1,2,3,...,t

ybar = zeros( t,1 );

ymed = zeros( t,1 );

for i = 1:t

y = theta + trnd(nu,[i 1]);

ybar(i) = mean(y);

ymed(i) = median(y);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*

%\*\* Generate graph

%\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

subplot(1,2,1)

plot(1:1:t,theta.\*ones(t,1),'-k',1:1:t,ybar,'-k');

title('(a) Mean');

ylabel('$\widehat{\theta}$');

xlabel('Progressive Sample Size');

box off;

axis tight;

subplot(1,2,2)

plot(1:1:t,theta.\*ones(t,1),'-k',1:1:t,ymed,'-k');

title('(b) Median');

ylabel('$\widehat{\theta}$');

xlabel('Progressive Sample Size');

box off;

axis tight;

%laprint(1,'cauchyconsist','options','factory');

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*

%\*\*\* Program to demonstrate the efficiency property of MLE for the normal distribution

%\*\*\*

%\*\*\* \*\* Note that the results will not match the numbers reported in the

%\*\*\* text exactly because of differences in the Gauss and Matlab

%\*\*\* random number generation.\*\*

%\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

clear all;

clc;

state = 123457;

rand('state', state);

randn('state', state);

mue = 1; % Population mean

sig2 = 2; % Population variance

r = 10000; % Number of replications

t = 100; % Sample size

% Generate sample means

u = randn(t,r); % Generate N(0,1) random numbers

y = mue + sqrt(sig2)\*u; % Generate N(mue,sig2) random numbers

meany = mean(y); % Compute the means of the r samples

mediany = median(y);

var\_meany = mean( (meany - mue).^2 );

var\_mediany = mean( (mediany - mue).^2 );

fprintf('Theoretical variance of the sample mean = %f\n', sig2/t);

fprintf('Simulated variance of the sample mean = %f\n\n', var\_meany);

fprintf('Theoretical variance of the sample median = %f\n', pi\*sig2/(2\*t));

fprintf('Simulated variance of the sample median = %f\n\n', var\_mediany);

%=========================================================================

%

% Monte Carlo program to demonstrate the asymptotical normality

% of MLE with R = 5000 replications based on an exponential distribution

% with theta = 1.

%

%=========================================================================

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',1234) )

R = 5000;

theta = 1;

% For the sample size of T = 5

%

t = 5; % Sample size

u = rand(t,R); % Generate uniform random numbers

y = -theta.\*log(1 - u); % Generate realizations of y

ybar = mean(y);

z1 = sqrt(t).\*(ybar - theta)./sqrt(theta); % Standardized random variable

% For the sample size of T = 100

%

t = 100; % Sample size

u = rand(t,R); % Generate uniform random numbers

y = -theta.\*log(1 - u); % Generate realizations of y

ybar = mean(y);

z2 = sqrt(t)\*(ybar - theta)./sqrt(theta); % Standardized random variable

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*

%\*\* Generate graph

%\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

subplot(2,2,1:2)

s = 0:0.001:12;

fy = exp(-s./theta)./theta;

plot(s,fy,'-k','LineWidth',0.75);

title('(a) Exponential distribution');

ylabel('$f(y)$');

xlabel('$y$');

ylim( [0, 1.5] )

xlim([0,6]);

set(gca,'XTick',0:2:6);

set(gca,'YTick',0:0.5:1.5);

box off;

%axis tight;

subplot(2,2,3)

hist(z1,21)

title('(b) $T=5$');

ylabel('$f(z)$');

xlabel('$z$');

xlim([-5,5]);

set(gca,'XTick',-4:1:4);

h = findobj(gca,'Type','patch');

set(h,'FaceColor','w','EdgeColor','k')

box off;

%axis tight;

subplot(2,2,4)

hist(z2,21)

title('(c) $T=100$');

ylabel('$f(z)$');

xlabel('$z$');

xlim([-5,5]);

set(gca,'XTick',-4:1:4);

h = findobj(gca,'Type','patch');

set(h,'FaceColor','w','EdgeColor','k')

box off;

%axis tight;

%laprint(1,'aympnorm','options','factory');

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*

%\*\*\* Monte Carlo program to demonstrate the Central Limit Theorem

%\*\*\* where 10000 numbers are drawn from a Chi-squared distribution

%\*\*\* with one degree of freedom.

%\*\*\*

%\*\*\* For each sample of size 5, the sample mean is computed and the

%\*\*\* standardized random variable constructed where the population

%\*\*\* mean and variance are equal to 1 and 2 respectively for the

%\*\*\* Chi-squared distribution with one degree of freedom

%\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

clear all;

clc;

clf;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',123457) )

t = 5; % Sample size

r = 1000; % Number of draws

rnorm = randn(t,r); % Generate N(0,1) random numbers

rchi1 = rnorm.^2; % Chi-squared (1) random numbers

z = sqrt(t)\*(mean(rchi1)' - 1)/sqrt(2); % Standardized random variable

hist(z,21); % Plot the histogram with 21 bars

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*

%\*\*\* Note that the last four lines can be written on one line as:

%\*\*\*

%\*\*\* hist( (mean(rnorm.^2)' -1 )/sqrt(2/n), 21);

%\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fprintf('Replications = %f\n', r);

fprintf('Sample size = %f\n', t);

fprintf('Mean = %f\n', mean(z)');

fprintf('Variance = %f\n', std(z)'^2);

fprintf('Standard deviation = %f\n\n', std(z)');

zsort = sort(z,1); % Sort the data

fprintf('The empirical distribution\n');

fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n');

fprintf(' 1.0 per cent = %f\n', zsort(0.01\*r,1));

fprintf(' 2.5 per cent = %f\n', zsort(0.025\*r,1));

fprintf(' 5.0 per cent = %f\n', zsort(0.05\*r,1));

fprintf('10.0 per cent = %f\n', zsort(0.10\*r,1));

fprintf('50.0 per cent = %f\n', zsort(0.50\*r,1));

fprintf('90.0 per cent = %f\n', zsort(0.90\*r,1));

fprintf('95.0 per cent = %f\n', zsort(0.95\*r,1));

fprintf('97.5 per cent = %f\n', zsort(0.975\*r,1));

fprintf('99.0 per cent = %f\n\n', zsort(0.99\*r,1));

%=========================================================================

%

% Program to demonstrate the distribution of the t-statistic

% from a Student t distribution with degrees of freedom of nu={1,2,3}.

%

%=========================================================================

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12) )

t = 500;

r = 5000;

% Generate Student t (nu=1) ie Cauchy distribution

nu = 1;

chi1 = randn(t,r).^2;

rstud1 = randn(t,r)./sqrt( chi1./nu );

z1 = sqrt(t)\*mean(rstud1)./std(rstud1);

figure(1)

histfit(z1,41);

title('Student t with nu=1')

xlabel('Midpoint')

ylabel('Frequency')

% Generate Student t (nu=2)

nu = 2;

chi2 = randn(t,r).^2 + randn(t,r).^2;

rstud2 = randn(t,r)./sqrt( chi2./nu );

z2 = sqrt(t)\*mean(rstud2)./std(rstud2);

figure(2)

histfit(z2,41);

title('Student t with nu=2')

xlabel('Midpoint')

ylabel('Frequency')

% Generate Student t (nu=3)

nu = 3;

chi3 = randn(t,r).^2 + randn(t,r).^2 + randn(t,r).^2;

rstud3 = randn(t,r)./sqrt( chi3./nu );

z3 = sqrt(t)\*mean(rstud3)./std(rstud3);

figure(3)

histfit(z3,41);

title('Student t with nu=3')

xlabel('Midpoint')

ylabel('Frequency')

%=========================================================================

%

% Program to compare the edgeworth expansion of the finite sample distribution

% and the asymptotic distribution (exponential distribution)

%

%=========================================================================

clear all

clc

t = 5;

s = -2:1:2;

% Finite sample value (based on a gamma distribution)

f\_finite = 1 - gamcdf( t./(1 + s/sqrt(t)),t,1 );

% Asymptotic value (based on the normal distribution)

f\_asy = normcdf( s );

% Compute Edgeworth expansion values

h2 = s.^2 - 1;

h3 = s.^3 - 3\*s;

h5 = s.^5 - 10\*s.^3 + 15\*s;

f\_ew = normcdf(s) - normpdf(s).\*( (1 + (2/3)\*h2)/sqrt(t) + ( (5/2)\*s + (11/12)\*h3 + (2/9)\*h5 )/t );

% Print results

format ShortG

disp( [' Sample size = ' num2str(t) ] );

disp( ' ' );

disp( ' s Finite Edgeworth Asymptotic ');

disp([ s' f\_finite' f\_ew' f\_asy'] );

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*

%\*\*\* Program to demonstrate the biasedness property of MLE for the

%\*\*\* normal distribution and an adjustment to achieve unbiasedness.

%\*\*\*

%\*\*\* \*\* Note that the results will not match the numbers reported in the

%\*\*\* text exactly because of differences in the Gauss and Matlab

%\*\*\* random number generation.\*\*

%\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

clear all;

clc;

state = 123457;

rand('state', state);

randn('state', state);

mue = 1; % Population mean

sig2 = 2; % Population variance

r = 20000; % Number of replications

t = 5; % Sample size

% Generate sample means assuming population mean is unknown

u = randn(t,r); % Generate N(0,1) random numbers

y = mue + sqrt(sig2)\*u; % Generate N(mue,sig2) random numbers

tmp = repmat(mean(y),size(y,1),1);

vary = sum( (y - tmp).^2 )/t; % Compute the MLEs of the r samples

vary\_unbiased = sum( (y - tmp).^2 )/(t-1); % Compute the unbiased estimates of the r samples

fprintf('Results based on unknown population mean\n');

fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n');

fprintf('Theoretical value = %f\n', sig2);

fprintf('Simulated expectation of the MLE = %f\n', mean(vary));

fprintf('Simulated expectation of the unbiased estimator = %f\n\n\n', mean(vary\_unbiased));

% Generate sample means assuming population mean is known

u = randn(t,r); % Generate N(0,1) random numbers

y = mue + sqrt(sig2)\*u; % Generate N(mue,sig2) random numbers

vary = sum( (y - mue).^2 )/t; % Compute the MLEs of the r samples

vary\_unbiased = sum( (y - mue).^2 )/(t-1); % Compute the unbiased estimates of the r samples

fprintf('Results based on known population mean\n');

fprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n');

fprintf('Theoretical value = %f\n', sig2);

fprintf('Simulated expectation of the MLE = %f\n', mean(vary));

fprintf('Simulated expectation of the unbiased estimator = %f\n\n\n', mean(vary\_unbiased));

%=========================================================================

%

% Program to compute the maximum likelihood estimates of the portfolio

% diversification model

%

% Asset price data from 6 August 2010 to 2 January 2001 (note that the

% data are in reverse order ie from recent to past)

%

%=========================================================================

clear all

clc

% Load data

load diversify.mat

% Select appropriate sample

pt\_apple = pt\_apple(1:2413);

pt\_ford = pt\_ford(1:2413);

% Compute percentage returns

r\_apple = 100\*diff(log(pt\_apple));

r\_ford = 100\*diff(log(pt\_ford));

% Compute statistics

m1 = mean(r\_apple);

m2 = mean(r\_ford);

s11 = mean((r\_apple - mean(r\_apple)).^2);

s22 = mean((r\_ford - mean(r\_ford)).^2);

c = corrcoef(r\_apple,r\_ford);

r = c(1,2);

disp(['Sample mean (Apple) = ' num2str(m1) ]);

disp(['Sample mean (Ford) = ' num2str(m2) ]);

disp(['Sample variance (Apple) = ' num2str(s11) ]);

disp(['Sample variance (Ford) = ' num2str(s22) ]);

disp(['Sample correlation = ' num2str(r) ]);

% Compute weights and risk of optimal portfolio

w1 = ( s22 - r\*sqrt(s11\*s22) ) / ( s11 + s22 - 2\*r\*sqrt(s11\*s22) );

w2 = 1 - w1;

s2\_port = w1^2\*s11 + w2^2\*s22 + 2\*w1\*w2\*r\*sqrt(s11\*s22);

disp(['Optimal weight (Apple) = ' num2str(w1) ]);

disp(['Optimal weight (Ford) = ' num2str(w2) ]);

disp(['Risk of optimal portfolio = ' num2str(s2\_port) ]);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*

%\*\*\* Generate graph

%\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

scatter(r\_ford,r\_apple,'.','k')

xlabel('Ford');

ylabel('Apple');

% Print TEX file

laprint(1,'diversify','options','factory');

%=========================================================================

%\*\*

%\*\* Program to demonstrate multiple roots of the bivariate normal model

%\*\*

%=========================================================================

clear all;

clc;

% This commented out code generates the data from scratch

% RandStream.setDefaultStream( RandStream('mt19937ar','seed',1234) )

% t = 4; % Sample size

% rho = 0.6;

% u1 = randn(t,1); % Generate independent N(0,1)

% u2 = randn(t,1);

% x = u1; % Generate dependent normal random numbers

% y = rho\*u1 + sqrt(1 - rho^2)\*u2;

% Take the GAUSS data for consistency

x = [-0.60303846 ;

-0.098331502;

-0.15897445 ;

-0.65344600 ];

y = [ 0.15367001;

-0.22971064;

0.66821992;

-0.44328369 ];

sxy = mean(x.\*y);

sxx = mean(x.^2);

syy = mean(y.^2);

lnl = zeros(199,1); % log-likelihood

g = zeros(199,1); % gradients

rho = -0.99:0.01:0.99; % grid values of rho

tg = length( rho );

for i = 1:tg

lnl(i) = -log(2\*pi) - 0.5\*log(1 - rho(i)^2) ...

- 0.5\*(sxx - 2\*rho(i)\*sxy + syy)/(1 - rho(i)^2);

g(i) = rho(i)\*(1 - rho(i)^2) + (1 + rho(i)^2)\*sxy - rho(i)\*(sxx + syy);

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*

%\*\* Generate graph

%\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Switch off TeX interpreter and clear figure

set(0,'defaulttextinterpreter','none');

figure(1);

clf;

subplot(1,2,1)

plot(rho,zeros(tg,1),'-k',rho,g,'-k','LineWidth',1);

title('(a) Gradient');

ylabel('$G\_T(\rho)$');

xlabel('$\rho$');

axis([-1.0 1.0 -0.6 0.6])

set(gca,'YTick',[-0.6 -0.4 -0.2 0.0 0.2 0.4 0.6]);

set(gca,'XTick',[-1.0 -0.5 0.0 0.5 1.0]);

box off;

subplot(1,2,2)

plot(rho(5:end-5),lnl(5:end-5),'-k','LineWidth',1);

title('(b) Log-likelihood function');

ylabel('$L\_T(\rho)$');

xlabel('$\rho$');

axis([-1.0 1.0 -3.0 -1.5])

set(gca,'YTick',[-3.0 -2.5 -2.0 -1.5]);

box off;

%laprint(1,'binormal','options','factory');

%=========================================================================

%

% Program to demonstrate multiple roots of a regression model

% with nonlinear parameterisation

%

%=========================================================================

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12) )

t = 100;

% Parameters

beta0 = 2.5;

beta1 = -1.5;

sig2 = 1.0;

% Generate data

u = randn(t,1);

x = rand(t,2);

y = beta0 + beta1\*x(:,1) + beta1^2\*x(:,2) + sqrt(sig2)\*u;

ngrid = 299;

lnl = zeros(ngrid,1);

g = zeros(ngrid,2);

b1 = seqa(-2.1,0.01,ngrid);

for i = 1:ngrid

b0 = beta0;

z = (y - b0 - b1(i)\*x(:,1) - b1(i)^2\*x(:,2))/sig2;

lnl(i) = - log(2\*pi) - 0.5\*log(sig2) - 0.5\*mean(z.^2);

g(i,1) = mean( (y - b0 - b1(i)\*x(:,1) - b1(i)^2\*x(:,2)) );

g(i,2) = mean( (y - b0 - b1(i)\*x(:,1) - b1(i)^2\*x(:,2)).\*(x(:,1) + 2\*b1(i)\*x(:,2)) );

end

figure(1)

subplot(1,2,1)

plot(b1,[g(:,2) zeros(ngrid,1)]);

title('(a) Gradient');

ylabel('$G\_T(\theta)$')

xlabel('$\theta$')

subplot(1,2,2)

plot(b1,lnl)

title('(b) Log-likelihood');

ylabel('$L\_T(\theta)$');

xlabel('$\theta$')