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

%

% Program to find the MLEs using graphical methods

%

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

clear all;

clc;

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

% Simulate the model

x = [ 1; 2; 4; 5; 8 ];

beta = 1.0;

sig2 = 4.0;

t = length( x );

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

% Plot log-likelihood sig2 = 4

sig2 = 4.0;

beta = (0.0:0.1:2.0)';

a1 = zeros(21,1);

for i=1:length(beta)

z = ( y - beta(i)\*x )/sqrt(sig2);

a1(i) = -0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*mean( z.^2 );

end

figure;

subplot(2,2,1);

plot(beta,a1);

title('Log-likelihood: sig2 = 4.0');

% Plot log-likelihood sig2 = 3.5

sig2 = 3.5;

beta = (0.0:0.1:2.0)';

a2 = zeros(21,1);

for i=1:21

z = ( y - beta(i)\*x)/sqrt(sig2);

a2(i) = -0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*mean( z.^2 );

end

subplot(2,2,2);

plot(beta,a2);

title('Log-likelihood: sig2 = 3.5');

% Plot log-likelihood beta = 0.9

beta = 1.0;

sig2 = (1.0:0.5:11)';

a3 = zeros(21,1);

for i=1:21

z = ( y - beta\*x)/sqrt(sig2(i));

a3(i) = -0.5\*log(2\*pi) - 0.5\*log(sig2(i)) - 0.5\*mean( z.^2 );

end

subplot(2,2,3);

plot(sig2,a3);

title('Log-likelihood: beta = 1.0');

axis tight

% Plot log-likelihood beta = 0.9

beta = 0.9;

sig2 = (1.0:0.5:11)';

a4 = zeros(21,1);

for i=1:21

z = ( y - beta\*x)/sqrt(sig2(i));

a4(i) = -0.5\*log(2\*pi) - 0.5\*log(sig2(i)) - 0.5\*mean( z.^2 );

end

subplot(2,2,4);

plot(sig2,a4);

title('Log-likelihood: beta = 0.9');

axis tight

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

%

% Program to find the MLEs using grid search methods

%

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

clear all;

clc;

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

% Simulate the model

x = [ 1; 2; 4; 5; 8; ];

beta = 1.0;

sig2 = 4.0;

t = size(x,1);

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

% Grid search on gradient sig2 = 4

sig2 = 4.0;

beta = (0.5:0.1:1.5)';

g1 = zeros(11,1);

for i=1:11

g1(i) = mean( (y - beta(i)\*x).\*x )' / sig2;

end

figure(1)

subplot(1,2,1);

plot(beta,g1,beta,zeros(length(beta)),'-k');

title('Gradient: sig2 = 4.0');

% Grid search on gradient sig2 = 3.5

sig2 = 3.5;

beta = (0.5:0.1:1.5)';

g2 = zeros(11,1);

for i=1:11

g2(i) = mean( (y - beta(i)\*x).\*x )' / sig2;

end

subplot(1,2,2)

plot(beta,g2,beta,zeros(length(beta)),'-k');

title('Gradient: sig2 = 3.5');

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

%\*\*\*

%\*\*\* Program to find the MLEs using one iteration of Newton-Raphson

%\*\*\*

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

clear all;

clc;

state = 123;

rand('twister', state);

randn('state', state);

% (1) - Simulate the model

x = [ 1; 2; 4; 5; 8 ];

beta = 1.0;

sig2 = 4.0;

t = size(x,1);

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

% (3) - Evaluate gradient and Hessian at theta0

beta\_0 = 1.0;

sig2\_0 = 4.0;

theta\_0 = beta\_0 | sig2\_0;

g = zeros(2,1);

g(1,1) = sum( (y - beta\_0\*x).\*x )/sig2\_0;

g(2,1) = -0.5\*t/sig2\_0 + 0.5\*sum( (y - beta\_0\*x).^2 )/sig2\_0^2;

h = zeros(2,2);

h(1,1) = -sum(x.^2)/sig2\_0;

h(1,2) = -sum( (y - beta\_0\*x).\*x )/sig2\_0^2;

h(2,1) = -sum( (y - beta\_0\*x).\*x )/sig2\_0^2;

h(2,2) = 0.5\*t/sig2\_0^2 - sum( (y - beta\_0\*x).^2 )/sig2\_0^3;

% (4) - Average log-likelihood at theta0

a\_0 = -0.5\*log(2\*pi) - 0.5\*log(sig2\_0) - 0.5\*mean( (y - beta\_0\*x).\*2 )/sig2\_0;

% (5) - Newton-Raphson update

theta\_1 = theta\_0 - inv(h)\*g;

beta\_1 = theta\_1(1);

sig2\_1 = theta\_1(2);

% (6) - Average log-likelihood at theta1

a\_1 = -0.5\*log(2\*pi) - 0.5\*log(sig2\_1) - 0.5\*mean( (y - beta\_1\*x).\*2 )/sig2\_1;

fprintf(1,'Average log-likelihood at theta0 = %f\n', a\_0);

fprintf(1,'Average log-likelihood at theta1 = %f\n\n', a\_1);

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

%\*\*\*

%\*\*\* Program to find the MLEs using the Newton-Raphson algorithm

%\*\*\*

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

clear all;

clc;

state = 123;

randn('state', state);

% Part (a): Simulate the model

x = [ 1; 2; 4; 5; 8; ];

beta = 1.0;

sig2 = 4.0;

t = size(x,1);

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

% (3) - Evaluate gradient and Hessian at theta0

beta = 1.0;

sig2 = 4.0;

theta = beta | sig2;

g = zeros(2,1);

g(1,1) = sum( (y - beta\*x).\*x )/sig2;

g(2,1) = -0.5\*t/sig2 + 0.5\*sum( (y - beta\*x).^2 )/sig2^2;

h = zeros(2,2);

h(1,1) = -sum(x.^2)/sig2;

h(1,2) = -sum( (y - beta\*x).\*x )/sig2^2;

h(2,1) = -sum( (y - beta\*x).\*x )/sig2^2;

h(2,2) = 0.5\*t/sig2^2 - sum( (y - beta\*x).^2 )/sig2^3;

% Average log-likelihood at theta0

a\_0 = -0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*mean( (y - beta\*x).\*2 )/sig2;

% Newton-Raphson

while g'\*g > 0.001

theta = theta - inv(h)\*g;

beta = theta(1);

sig2 = theta(2);

g = zeros(2,1);

g(1,1) = sum( (y - beta\*x).\*x )/sig2;

g(2,1) = -0.5\*t/sig2 + 0.5\*sum( (y - beta\*x).^2 )/sig2^2;

h = zeros(2,2);

h(1,1) = -sum(x.^2)/sig2;

h(1,2) = -sum( (y - beta\*x).\*x )/sig2^2;

h(2,1) = -sum( (y - beta\*x).\*x )/sig2^2;

h(2,2) = 0.5\*t/sig2^2 - sum( (y - beta\*x).^2 )/sig2^3;

end

% Optimum Average log-likelihood at final theta

a\_1 = -0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*mean( (y - beta\*x).\*2 )/sig2;

fprintf(1,'Average log-likelihood at theta0 = %f\n\n', a\_0);

fprintf(1,'After applying newton raphson\n');

fprintf(1,'Final value of beta = %f\n', beta);

fprintf(1,'Final value of variance = %f\n', sig2);

fprintf(1,'Final average log-likelihood = %f\n', a\_1);

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

%

% Program to estimate a exponential model

%

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

clear all

clc

% Data

y = [ 3.5, 1.0, 1.5 ]';

t = length(y);

niter = 10;

% Newton-Raphson algorithm

disp('Newton Raphson algorithm');

disp(' ');

theta = 1.0;

for k = 1:niter

g = -1/theta + mean(y)/theta^2;

h = 1/theta^2 - 2\*mean(y)/theta^3;

lnl = -log(theta) - mean(y)/theta;

theta = theta - inv(h)\*g;

disp( ' Iter G J logL th(k)') ;

disp( [k g g'\*g lnl theta ] );

end

disp(' ');

% Scoring algorithm

disp('Scoring algorithm');

disp(' ');

theta = 1.0;

for k = 1:niter

g = -1/theta + mean(y)/theta^2;

i = 1/theta^2;

lnl = -log(theta) - mean(y)/theta;

theta = theta + inv(i)\*g;

disp( ' Iter G J logL th(k)') ;

disp( [k g g'\*g lnl theta ] );

end

disp(' ');

% BHHH algorithm

disp('BHHH algorithm');

disp(' ');

theta = 1.0;

for k = 1:niter

gt = -1/theta + y/theta^2;

g = mean(gt);

j = gt'\*gt/t;

lnl = -log(theta) - mean(y)/theta;

theta = theta + inv(j)\*g;

disp( ' Iter G J logL th(k)') ;

disp( [k g j lnl theta ] );

end

% BHHH algorithm with squeezing

disp('BHHH algorithm with squeezing');

disp(' ');

theta = 1.0;

for k = 1:niter

gt = -1/theta + y/theta^2;

g = mean(gt);

j = gt'\*gt/t;

lnl = -log(theta) - mean(y)/theta;

thetaold = theta;

lnlold = lnl;

theta = theta + inv(j)\*g;

lnl = -log(theta) - mean(y)/theta;

if lnl > lnlold;

disp( ['Full iteration step successful at iteration k =' num2str(k) ]);

else

disp( ['Full iteration step not successful at iteration k =' num2str(k) ]);

for m = 2:10

lambda = 1/m;

disp( ['Squeezing with step = ' num2str(lambda) ]);

theta = thetaold + lambda\*inv(j)\*g;

lnl = -log(theta) - mean(y)/theta;

if lnl > lnlold

break

end

end

end

disp( ' Iter G J logL th(k)') ;

disp( [k g j lnl theta ] );

end

% BFGS algorithm

disp('BFGS algorithm');

disp(' ');

theta = 1.5;

h = -1;

niter = 9;

for k = 1:niter

g = -1/theta + mean(y)/theta^2;

lnl = -log(theta) - mean(y)/theta;

theta = theta - inv(h)\*g;

disp( ' Iter g h logL th(k)') ;

disp( [k g h lnl theta ] );

dtheta = -inv(h)\*g;

gold = g;

g = -1/theta + mean(y)/theta^2;

dg = g - gold;

h = h - ( h\*dtheta\*dg' + dg\*dtheta'\*h )/(dg'\*dtheta) ...

+ ( 1 + (dtheta'\*h\*dtheta)/(dg'\*dtheta) )\*(dg\*dg')/(dg'\*dtheta);

end

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

%

% Program to estimate a Cauchy model (one iteration)

% using the NEWTON-RAPHSON, SCORING and BHHH algorithms

%

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

clear all

clc

% Load data

yt = [ 2, 5, -2, 3, 3 ];

t = length(yt);

theta0 = 3;

% Newton-Raphson

g = 2\*sum( (yt - theta0)./(1 + (yt - theta0).^2) );

h = 2\*sum( ((yt - theta0).^2 - 1)./(1 + (yt - theta0).^2).^2 );

thetaNR = theta0 - inv(h)\*g;

% Method of Scoring

g = 2\*sum( (yt - theta0)./(1 + (yt - theta0).^2) );

i = t/2;

thetaSC = theta0 + inv(i)\*g;

% BHHH

gt = 2\*( (yt - theta0)./(1 + (yt - theta0).^2) );

thetaBH = theta0 + inv(gt\*gt')\*sum(gt);

disp( ['Newton-Raphson ' num2str(thetaNR) ] );

disp( ['Method of Scoring ' num2str(thetaSC) ] );

disp( ['BHHH ' num2str(thetaBH) ] );

disp( ' ' );

disp( ['Gradient ' num2str(g) ]);

disp( ['Standard error (Hessian) ' num2str(sqrt(-inv(h))) ] );

disp( ['Standard error (Information) ' num2str(sqrt(inv(i))) ] );

disp( ['Standard error (OPG) ' num2str(sqrt(inv(gt\*gt'))) ] );

disp( ' ' );

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

%

% Program to estimate the parameters of a Weibull distribution using

% the Newton-Raphson and BHHH algorithms

%

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

function max\_weibull( )

clear all;

clc;

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

t = 20;

% Generate Weibull random numbers

alpha = 1.0;

beta = 2.0;

z = rand(t,1);

x = -log(1 - z)/alpha; % Generate exponential random numbers

y = x.^(1/beta); % Generate Weibull random numbers

% or use Matlab function

y = wblrnd((1/alpha)^(1/beta),beta,[t,1]); % Generate Weibull random numbers

% or load data

y = [0.293, 0.589, 1.374, 0.954, 0.608, 1.199, 1.464, ...

0.383, 1.743, 0.022, 0.719, 0.949, 1.888, 0.754, ...

0.873, 0.515, 1.049, 1.506, 1.090, 1.644]';

disp(' t y');

disp([(1:1:t)',y]);

% Starting values

theta0 = [0.5;1.5];

alpha = theta0(1);

beta = theta0(2);

g = gradvec(theta0,y);

h = hessmat(theta0,y);

j = opgmat(theta0,y);

disp( 'Estimates at iteration 0 ' );

disp( theta0 );

disp(['Value of log-likelihood at iteration 0 = ' num2str(-lnl(theta0,y))] );

disp('Gradient at iteration 0' )

disp( g );

disp('Hessian at iteration 0');

disp( h );

disp('OPG at iteration 0');

disp(j);

% Newton Raphson update

theta1 = theta0 - inv(h)\*g;

disp( 'Estimates at iteration 1 (Newton Raphson) ' );

disp( theta1 );

disp(['Value of log-likelihood at iteration 1 = ' num2str(-lnl(theta1,y))] );

% BHHH

theta1 = theta0 + inv(j)\*g;

disp( 'Estimates at iteration 1 (BHHH) ' );

disp( theta1 );

disp(['Value of log-likelihood at iteration 1 = ' num2str(-lnl(theta1,y))] );

% Call fminunc to optimize function

theta = fminunc(@(theta) lnl(theta,y),theta1);

disp( 'MLE of theta ' );

disp( theta );

g = gradvec(theta,y);

h = hessmat(theta,y);

j = opgmat(theta,y);

disp( 'Covariance matrix (Hessian) ' );

disp( (1/t)\*inv(h) );

disp( 'Covariance matrix (OPG) ' );

disp( (1/t)\*inv(j) );

% Call fminunc to optimize the concentrated likelihood

beta = fminunc(@(b) lnlc(b,y),theta1(2));

alpha = 1/mean(y.^beta);

disp( 'Results for concentrated likelihood' );

disp(['MLE of alpha = ' num2str(alpha) ] );

disp(['MLE of beta = ' num2str(beta) ] );

% Call fminunc to optimize the transformed likelihood

[theta,~,~,~,~,h] = fminunc(@(theta) lnlt(theta,y),theta1);

disp( 'Results for transformed likelihood' );

disp(['MLE of lambda = ' num2str(theta(1)) ] );

disp(['MLE of beta = ' num2str(theta(2)) ] );

% Std error by delta method

d = [ -(1/theta(2))\*theta(1)^(-1/theta(2)-1), ...

(log(theta(1))/theta(2)^2)\*theta(1)^(-1/theta(2)) ];

disp( 'Standard error of lambda by delta method' ) ;

disp( -d\*inv(h)\*d' );

end

%------------------------- Functions ----------------------------- %

% Log-likelihood

function lf = lnl(theta,y)

a = theta(1);

b = theta(2);

f = log(a) + log(b) + (b-1)\*log(y) - a\*y.^b;

lf = -mean( f );

end

% Concentrated log-likelihood

function lf = lnlc(b,y)

a = 1/mean(y.^b);

f = log(a) + log(b) + (b-1)\*log(y) - a\*y.^b;

lf = -mean( f );

end

% Transformed log-likelihood function

function lf = lnlt(theta,y)

l = theta(1);

b = theta(2);

f = log(b) - log(l ) + (b-1)\*log(y/l) - (y/l).^b;

lf = -mean(f);

end

% Return gradient vector

function g = gradvec(theta,y)

alpha = theta(1);

beta = theta(2);

g1 = 1/alpha - mean(y.^beta);

g2 = 1/beta + mean(log(y)) - alpha\*mean(log(y).\*y.^beta );

g = [g1; g2];

end

% Return Hessian matrix

function h = hessmat(theta,y)

alpha = theta(1);

beta = theta(2);

h11 = - 1/alpha^2 ;

h12 = - mean(log(y).\*y.^beta);

h21 = h12;

h22 = -1/beta^2 - alpha\*mean( (log(y).^2).\*(y.^beta) );

h = [h11, h12; h21, h22];

end

% Return OPG matrix

function j = opgmat(theta,y)

alpha = theta(1);

beta = theta(2);

g22 = 1/beta + log(y) - alpha\*log(y).\*y.^beta;

j11 = mean( ( 1/alpha - y.^beta).^2 );

j12 = mean( ( 1/alpha - y.^beta) .\* g22 );

j21 = j12;

j22 = mean( g22.^2 );

j = [j11, j12; j21, j22];

end

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

%

% Program to Program to estimate the distribution of asset returns

% using a standardised Student t distribution

%

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

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

%

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

function max\_returns( )

clear all

clc

% Load data

load diversify.mat

t = 2413;

% Select appropriate sample

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

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

% Compute percentage returns

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

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

% Select a return

y = r\_apple;

% Compute desrciptive statistics

m = mean(y);

s = std(y);

z = (y - m)/s;

skew = mean( z.^3);

kurt = mean( 4.^3);

disp(['Mean = ',num2str(m) ]);

disp(['Std dev. = ',num2str(s) ]);

disp(['Skewness = ',num2str(skew) ]);

disp(['Kurtosis = ',num2str(kurt) ]);

disp(['Maximum = ',num2str(max(z)) ]);

disp(['Minimum = ',num2str(max(z)) ]);

figure(1)

histfit( z, 21);

% Estimate the model by MLE with Student t distribution

start = [m ; s ; 5 ];

ops = optimset('LargeScale','off','Display','off');

[ bhat,lf,a,a,a,hess] = fminunc(@(b) lnl( b,z ),start,ops );

lf = -lf;

vc = 1/(t-1)\*inv(hess);

disp('Unrestricted model results')

disp(['Log-likelihood function = ',num2str(lf) ]);

disp(' Param Std err.')

disp( [ bhat sqrt(diag(vc))] )

% Estimate the model by MLE with normal distribution

[ bhat,lf,a,a,a,hess] = fminunc(@(b) lnl1( b,z ),start(1:2),ops );

lf = -lf;

vc = 1/(t-1)\*inv(hess);

disp('Restricted model results')

disp(['Log-likelihood function = ',num2str(lf) ]);

disp(' Param Std err.')

disp( [ bhat sqrt(diag(vc))] )

end

% ------------------------ Functions ------------------------------------%

%

%-------------------------------------------------------------------------

% Log-likelihood of a Student t distribution

%-------------------------------------------------------------------------

function lf = lnl(b,z)

m = b(1);

s = abs(b(2));

v = abs(b(3));

s2 = s^2;

tmp = log( gamma((v+1)/2) ) - 0.5\*log(pi) - log(gamma(v/2)) - 0.5\*log(v-2) ...

- 0.5\*log(s2) - ((v+1)/2)\*log( 1 + ((z - m).^2)./(s2\*(v-2)) );

lf = -mean(tmp);

end

%-------------------------------------------------------------------------

% Log-likelihood of a normal distribution

%-------------------------------------------------------------------------

function lf = lnl1(b,z)

m = b(1);

s = abs(b(2));

tmp = log(normpdf(z,m,s));

lf = -mean(tmp);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%

%% Compute the steps of simplex and find the step thatis used

%% in first iteration.

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function ch3\_7

clear all;

clc;

% Read in the data for y

y = [ 3.5; 1.0; 1.5; ];

t = length( y ); % Define the sample size

th = [1 ; 3 ]; % Initialize the estimates

neglog = lnl\_neg(th,y,t);

if(neglog(1) > neglog(2))

flipud (th);

flipud(neglog);

end

th\_avg = mean(th);

% The three steps that may be followed

% Reflect

alpha = 0.5;

th\_r = th\_avg + alpha\*(th\_avg - th(2));

% Expand

beta = 1.1;

th\_e = th\_avg + beta\*(th\_avg - th(2));

% Contract

gamma = 0.5;

th\_c = th\_avg + gamma\*(th\_avg - th(2));

% Decide which step is followed

llen\_r = lnl\_neg(th\_r,y,t);

if(llen\_r < neglog(2))

if(llen\_r < neglog(1))

fprintf('Expanding the simplex \n');

llen\_e = lnl\_neg(th\_e,y,t);

if (llen\_e < llen\_r)

fprintf('Theta2 is replaced by theta\_e \n');

th(2) = th\_e;

else

fprintf('Theta2 is replaced by theta\_r \n');

th(2) = th\_r;

end

else

fprintf('Reflecting the simplex \n');

fprintf('Theta2 is replaced by theta\_r \n');

th(2) = th\_r;

end

else

llen\_c = lnl\_neg(th\_c,y,t);

if(llen\_c < neglog(2))

fprintf('Contracting the simplex \n');

fprintf('Theta2 is replaced by theta\_c \n');

th(2) = th\_c;

else

fprintf('Shrinking the simplex \n');

th = (th + th\_avg) / 2;

end

end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SUBROUTINES

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Negative log likelihood evaluator

function lnl\_neg = lnl\_neg(theta,y,t)

lnl\_neg = t\*log(theta) + sum(y)./theta;

end

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

%

% Program to plot the profile likelihood for the portfolio

% diversification model

%

%

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

clear all;

clc;

% Asset price data from 6 August 2010 to 2 January 2001

%(note that the data are in reverse order ie from recent to past

%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\*( log(pt\_apple(2:end,:)) - log(pt\_apple(1:(end-1),:)) );

r\_ford = 100\*( log(pt\_ford(2:end,:)) - log(pt\_ford(1:(end-1),:)) );

% Compute maximum likelihood estimates

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);

rho = c(1,2);

disp(['Value of rho = ' num2str(rho)]);

% Generate the profile log-likelihood

y = [r\_apple, r\_ford];

r = ( -0.9 :0.01:-0.9+0.01\*(186-1) )';

a = zeros(length(r),1);

mean = [ m1, m2];

for i = 1:length(r)

s12 = r(i)\*sqrt(s11)\*sqrt(s22); % covariance between apple and ford

covariance = [s11, s12; s12, s22]; % s12 = s21

a(i) = -ecmnobj(y, mean, covariance)/length(y); % Compute average log-likelihood value for alternative values of rho

end

% Plot the profile log-likelihood

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

%\*\*\*

%\*\*\* Generate graph

%\*\*\*

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

% Switch off TeX interpreter and clear figure

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

figure(1);

clf;

plot(r,a,'k');

xlabel('theta1');

ylabel('Average lnl');

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

%

% Maximum likelihood estimation of the stationary distribution of the

% CIR model of interest rates using Ait Sahalia's (1996) data.

%

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

function max\_stationary( )

clear all;

clc;

% Load data (5505x4 array called eurodata, 1 Jun 1973 - 25 Feb 1995)

% 1. year

% 2. day

% 3. date stamp

% 4. interest rates

load eurodollar.mat

% Load data: time series object 'tbr'

%load FREDdata

rt = sort( eurodata(:,4)\*100 );

%rt = tbr.Data;

t = length( rt );

pstart = [ 5.6556 0.6763];

options = optimset('LargeScale', 'off', 'Display', 'iter');

[phat,~,~,~,~,hessian] = fminunc(@(p) neglog(p,rt),pstart,options);

%[phat] = fminsearch(@(p) neglog(p,rt),pstart,options);

hessinv = inv(hessian);

disp( 'Parameter estimates')

disp( phat )

disp( 'Standard errors based on inverse Hessian')

disp( [sqrt( hessinv(1,1) ) sqrt( hessinv(2,2) )] )

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

%\*\*\*

%\*\*\* Generate graph

%\*\*\*

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

% Switch off TeX interpreter and clear figure

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

figure(1);

clf;

[fcdf,x] = ecdf(rt);

[f,bins] = ecdfhist(fcdf,x,51);

bar(bins,f,'hist');

%[n,xout]=hist(rt,51);

%bar(xout,n/t)

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

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

box off

hold on

plot(rt,gampdf(rt,phat(1),1/phat(2)),'-k','LineWidth',0.75)

ylabel('$f(r)$');

xlabel('$r$');

set(gca,'YTick',[] );

axis tight

hold off

% Print the tex file to the relevant directory

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

end

%

%--------------------------- Functions -----------------------------------

%

%-------------------------------------------------------------------------

% Likelihood wrapper function

%-------------------------------------------------------------------------

function f = neglog( p,data )

f = -sum( lnlt( p,data) );

end

%-------------------------------------------------------------------------

% Likelihood function for stationary distribution of CIR model

%-------------------------------------------------------------------------

function f = lnlt(p,data)

v = abs(p(1));

w = abs(p(2));

%f = log( gampdf(data,v,w ) );

f = v\*log( w ) - gammaln( v ) + (v-1)\*log(data) - w\*data;

end

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

%

% Maximum likelihood estimation of the transitional distribution of the

% CIR model of interest rates using Ait Sahalia's (1996) data.

%

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

function max\_transitional( )

clc

clear all

% Load data (5505x4 array called eurodata, 1 Jun 1973 - 25 Feb 1995)

% 1. year

% 2. day

% 3. date stamp

% 4. interest rates

load eurodollar.mat

dt = 1/250;

rt = eurodata(:,4);

% Starting values

x = rt(1:end-1);

dx = diff(rt);

dx = dx./x.^0.5;

regressors = [dt./x.^0.5, dt\*x.^0.5];

drift = regressors\dx;

res = regressors\*drift - dx;

alpha = -drift(2);

mu = -drift(1)/drift(2);

sigma = sqrt(var(res, 1)/dt);

p0 = [abs(alpha) abs(mu) abs(sigma)];

% Estimation based on scaled Bessel function

options = optimset('LargeScale', 'off', 'Display', 'iter');

[phat1,~,~,~,~,hessian] = fminunc(@(p) cir1(p,rt),p0,options);

hessinv = inv(hessian);

disp( 'Parameter estimates')

disp( phat1 )

disp( 'Standard errors based on inverse Hessian')

disp( [sqrt( hessinv(1,1) ) sqrt( hessinv(2,2) ) sqrt( hessinv(3,3) )] )

% Estimation based on ncx2pdf function

options = optimset('LargeScale', 'off', 'Display', 'iter');

[phat,~,~,~,~,hessian] = fminunc(@(p) cir2(p,rt),phat1,options);

hessinv = inv(hessian);

disp( 'Parameter estimates')

disp( phat )

disp( 'Standard errors based on inverse Hessian')

disp( [sqrt( hessinv(1,1) ) sqrt( hessinv(2,2) ) sqrt( hessinv(3,3) )] )

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

%\*\*\*

%\*\*\* Generate graph

%\*\*\*

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

% Switch off TeX interpreter and clear figure

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

figure(1);

clf;

rlag = rt(1:end-1);

rnow = rt(2:end);

tmp0 = 0.03:0.01:0.25;

tmp1 = (phat(3).^2)\*tmp0;

plot(tmp0,tmp1,'-k','LineWidth',0.75);

hold on

scatter(rlag,rnow.^2,'.k')

hold off

ylabel('$r^2\_{t+\Delta t}$');

xlabel('$r\_t$');

set(gca,'YTick',[] );

axis tight

hold off

% Print the tex file to the relevant directory

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

end

%

%--------------------------- Functions -----------------------------------

%

%-------------------------------------------------------------------------

% Likelihood function for transitional distribution of CIR model (Bessel)

%-------------------------------------------------------------------------

function f = cir1( p,data )

alpha = abs(p(1));

mu = abs(p(2));

sigma = abs(p(3));

rnow = data(2:end);

rlag = data(1:end-1);

dt = 1/250;

c = 2\*alpha/(sigma^2\*(1-exp(-alpha\*dt)));

q = 2\*alpha\*mu/sigma^2-1;

u = c\*exp(-alpha\*dt)\*rlag;

v = c\*rnow;

lf = log(c)-u-v+0.5\*q\*log(v./u)+log(besseli(q,2\*sqrt(u.\*v),1))+2\*sqrt(u.\*v);

f = -sum( lf );

end

%-------------------------------------------------------------------------

% Likelihood function for transitional distribution of CIR model

% (Chi-square)

%-------------------------------------------------------------------------

function f = cir2( p,data )

alpha = abs(p(1));

mu = abs(p(2));

sigma = abs(p(3));

rnow = data(2:end);

rlag = data(1:end-1);

dt = 1/250;

c = 2\*alpha/(sigma^2\*(1-exp(-alpha\*dt)));

q = 2\*alpha\*mu/sigma^2-1;

u = c\*exp(-alpha\*dt)\*rlag;

v = c\*rnow;

nc = 2\*u;

df = 2\*q+2;

s = 2\*v;

gpdf = ncx2pdf( s,df,nc );

ppdf = 2\*c\*gpdf;

f = -sum(log( ppdf ) );

end

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

%

% Program to estimate a threshold autoregressive model based on

% the stationary and transitional distributions

%

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

function max\_tar( )

clear all

clc

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12357) )

t = 250;

theta = 0.5;

% Simulate data

y = zeros(t,1);

for i=2:t

y(i) = theta\*abs(y(i-1)) + randn;

end

% Estimate by least squares

xvar = trimr(abs(y),0,1);

yvar = trimr(y,1,0);

bols = xvar\yvar;

% Estimate by Ml applied to the stationary distribution

start = bols;

ops = optimset('LargeScale','off','Display','off');

[ bhat,~,~,~,~,hess] = fminunc(@(b) lnlstat( b,y ),start,ops );

disp('True value of theta')

disp( theta )

disp('OLS Results')

disp( bols );

disp('Stationary distribution results')

disp( bhat )

% Estimate by Ml applied to the transtional distribution

start = bols;

ops = optimset('LargeScale','off','Display','off');

[ bhat,~,~,~,~,hess] = fminunc(@(b) lnltrans( b,y ),start,ops );

disp('Transitional distribution results')

disp( bhat )

end

% ------------------------ Functions ------------------------------------%

%

%-------------------------------------------------------------------------

% Log-likelihood of a tar model: transitional distribution

%-------------------------------------------------------------------------

function lf = lnltrans(b,y)

m = b\*abs(trimr(y,0,1));

s = 1;

z = ( trimr(y,1,0) - m ) ./ s;

f = log( normpdf(z)/s );

lf = -mean(f);

end

%-------------------------------------------------------------------------

% Log-likelihood of a tar model: stationary distribution

%-------------------------------------------------------------------------

function lf = lnlstat(b,y)

m = 0.0;

s = 1/sqrt(1 - b^2);

z = ( y - m ) ./ s;

f = 2\*normpdf(z)\*(1/s).\*normcdf(b\*y);

lf = -mean( log(f) );

end