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

%

% Simulation of nonlinear and linear exponential models.

%

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

clear all

clc

RandStream.setDefaultStream( RandStream('mt19937ar','seed',1) )

% Simulate data

t = 50;

b0 = 1.0;

b1 = 0.05;

sig = 0.5;

u = sig\*randn(t,1);

x = (1:1:t)';

y1 = b0\*exp( b1\*x + u);

y2 = b0\*exp( b1\*x ) + u;

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

%\*\*\*

%\*\*\* Generate graphs

%\*\*\*

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

% Switch off TeX interpreter and clear figure

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

figure(1);

clf;

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

% Panel (a)

subplot(1,2,1);

plot(x,y1,'-k',...

x,y2,'-.k',...

'LineWidth',0.75);

title('(a) Levels');

xlabel('$x\_t$');

ylabel('$y\_t$');

set(gca,'XTick',0:10:50);

set(gca,'YTick',0:5:20);

xlim([0,50]);

ylim([0,20]);

%set(gca,'LineWidth',1);

%legend('$y\_{1,t}$','$y\_{2,t}$','Location','NorthWest');

box off;

%legend('boxoff');

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

% Panel (b)

subplot(1,2,2);

plot(x,log(y1),'-k',...

x,log(y2),'-.k',...

'LineWidth',0.75);

title('(b) Logs');

xlabel('$x\_t$');

ylabel('$\log y\_t$');

set(gca,'XTick',0:10:50);

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

xlim([0,50]);

ylim([-1,4]);

%set(gca,'LineWidth',1);

%legend('$\log y\_{1,t}$','$\log y\_{2,t}$','Location','NorthWest');

box off;

legend('boxoff');

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

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

%

% Estimate the exponential model by maximum likelihood.

%

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

function nls\_exponential()

clear all;

clc;

t = 50;

%flag = 1; % 1 = simulate data

flag = 0; % 0 = use GAUSS data to reproduce text

if flag

[ y,x ] = simulatedata( t );

else

% Aternatively load the GAUSS data

load nls\_expdata.dat

y = nls\_expdata(:,1);

x = nls\_expdata(:,2);

end

% Estimate the model and compute Hessian se

start = [0.1,0.1];

theta = fminunc(@(theta) neglog(theta,y,x),start);

ht = numhess(@neglog,theta',y,x);

vcov = (1/t)\*inv(ht);

disp(['Parameter estimates = ',num2str(theta) ]);

disp('Negative Hessian matrix');

disp( ht );

disp('Covariance matrix');

disp( vcov );

end

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

% Simulate data

function [ y,x ] = simulatedata( t )

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

b0 = 1.0;

b1 = 0.05;

sig = 0.5;

u = sig\*randn(t,1);

x = [1:1:t]';

y = b0\*exp( b1\*x ) + u;

end

% Log-likelihood function

function lf = neglog(b,y,x)

lf = - mean( lnlt(b,y,x) );

end

% Log-likelihood (concentrated) at each observation

function lf = lnlt(b,y,x)

e = y - b(1)\*exp( b(2)\*x );

s2 = e'\*e/length(e);

lf = - 0.5\*log(2\*pi) - 0.5\*log(s2) - 0.5\*e.^2/s2 ;

end

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

% Define the log of the likelihood (unconcentrated) at each observation

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

function val = lnlt\_unc(b,y,x,t)

e = y - b(1)\*exp( b(2)\*x ); % Residual error

s2 = b(3); % Do not concentrate out the residual variance

val = - 0.5\*log(2\*pi) - 0.5\*log(s2) - 0.5\*e.^2/s2 ;

end

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

%

% Program to estimate an exponential model

% using the GAUSS-NEWTON algorithm

%

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

clear all;

clc;

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

% Simulate data

t = 50;

b0 = 1.0;

b1 = 0.05;

sig = 0.5;

u = sig\*randn(t,1);

x = (1:1:t)';

y = b0\*exp( b1\*x ) + u;

% GAUSS-NEWTON algorithm

b = [ 0.1; 0.1]; % Choose staring values

crit = 0.00001; % Convergence criterion

maxit = 20; % Maximum number of iterations

for i=1:maxit

e = y - b(1)\*exp(b(2)\*x);

z1 = exp(b(2)\*x);

z2 = b(1)\*x.\*exp(b(2)\*x);

z = [z1 z2];

disp( ['Iteration = ', num2str(i) ] );

bchange = z\e;

if (bchange'\*bchange) < crit;

break;

else

b = b + bchange; % Update new parameters

if i==maxit;

fprintf(1,'Failed to converge after iteration number %d\n', maxit);

end

end

end

s2 = (e'\*e)/t; % Residual variance

omega = s2\*inv(z'\*z);

disp( 'Parameter estimates, std errors and t-stats' );

disp( [b sqrt(diag(omega)) b./sqrt(diag(omega)) ] );

disp(' ');

disp('Estimated asymptotic covariance matrix')

disp( omega );

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

%

% Program to estimate a nonlinear consumption function

% using the GAUSS-NEWTON algorithm

%

% U.S. data on real consumption and real disposable income(2005 $)

% 1960:Q1 to 2009:Q4 200 observations

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

clear all;

clc;

% Load data

load USdata.mat

yt = inc;

ct = cons;

t = length( yt );

% Estimate the linear model to use as initial starting values

b = [ones(t,1) yt]\ct;

% GAUSS-NEWTON algorithm

alpha = b(1);

beta = b(2);

gam = 1.00;

crit = 0.00001; % Convergence criterion

maxit = 20; % Maximum number of iterations

for i=1:maxit

et = ct - alpha - beta\*yt.^gam;

z1t = ones(t,1);

z2t = yt.^gam;

z3t = beta\*log(yt).\*yt.^gam;

disp( ' ' )

disp(['Iteration = ' num2str(i)]) ;

disp(['Parameters = ' num2str([alpha beta gam]) ] );

disp('---------------------------------------------------------------');

y = et;

x = [z1t z2t z3t];

bchange = x\y;

if (bchange'\*bchange) < crit

break;

else

alpha = alpha + bchange(1); % Update new parameters

beta = beta + bchange(2);

gam = gam + bchange(3);

if i == maxit

disp(['Failed to converge after iteration: ', num2str(maxit)] );

end

end

end

ssr = y'\*y;

sig2 = ssr/t;

disp( ['Sum squared residuals = ' num2str(ssr)] );

disp( ['Residual variance = ' num2str(sig2) ] );

disp( 'Information matrix' );

disp('---------------------');

z = [z1t z2t z3t];

im = z'\*z/sig2;

disp( im );

disp('Estimated asymptotic covariance matrix:\n');

disp( inv(im) );

bhat = [alpha beta gam]';

se = sqrt(diag(inv(im)));

disp('Parameter estimates, standard errors and t-stats');

disp( [bhat se bhat./se]);

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

%

% Program to perform tests on a nonlinear consumption function

%

% U.S. data on real consumption and real disposable income(2005 $)

% 1960:Q1 to 2009:Q4 200 observations

%

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

function nls\_contest( )

clear all;

clc;

% Load data

load USdata.mat

yt = inc;

ct = cons;

t = length( yt );

% Estimate the constrained model

b0 = [-228; 0.9] ;

[theta0,f0] = fminunc(@(b) neglog0(b,ct,yt),b0);

disp('Restricted Parameter Estimates');

theta0 = [ theta0' 1.000];

disp( theta0');

% Estimate the unconstrained model

b0 = [theta0 ];

[theta1,f1] = fminsearch(@(b) neglog1(b,ct,yt),b0);

f0 = -f0;

f1 = -f1;

% Compute relevant matrices

g = numgrad(@lnlt1,theta0',ct,yt);

G = mean( g );

J = g'\*g/t;

invH = inv(numhess(@neglog1,theta1',ct,yt));

disp('Unestricted Parameter Estimates');

disp( theta1 );

disp('Covariance Matrix of the Parameters');

disp( invH/t );

% Perform likelihood ratio test

lr = -2\*t\*(f0 - f1);

disp('Restricted Likelihood Function');

disp( t\*f0 );

disp('Unrestricted Likelihood Function');

disp( t\*f1 );

disp(' ');

disp('LR test and p-value');

disp( [lr 1-cdf('chi2',lr,1)] );

% Perform Wald test

R = [0 0 1];

Q = 1;

W = t\*(R\*theta1' - Q)'\*inv(R\*invH\*R')\*(R\*theta1' - Q);

disp(' ');

disp('Wald statistic and p-value');

disp( [ W 1-cdf('chi2',W,1) ] );

% Perform Lagrange multiplier test (OPG)

disp('');

disp( 'Outer product of gradients matrix')

disp( J );

LM = t\*(G\*inv(J)\*G');

pv = 1-cdf('chi2',LM,1);

disp(' ');

disp('LM statistic and p-value');

disp( [ LM 1-cdf('chi2',LM,1) ] );

% Perform 2-step LM test

x = [ ones(t,1),inc ];

y = cons;

% Estimate constrained model

b = x\y;

e = y - x\*b;

b = [b ; 1.0];

% Evaluate derivatives at constrained estimates

z1 = ones(t,1);

z2 = inc.^b(3);

z3 = b(2)\*log(inc).\*inc.^b(3);

z = [z1 , z2 , z3];

% Second step regression

v = e - z\*(z\e);

r2 = 1 - e'\*e\v'\*v;

% LM statistic

lm = t\*r2;

disp('2- step LM statistic and p-value');

disp( [ lm 1-cdf('chi2',lm,1) ] );

end

%

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

%

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

% Log-likelihood function of constrained model

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

function lf = neglog0(b,ct,yt)

t = length( ct );

e = ct - b(1) - b(2)\*yt;

s2 = e'\*e/t; % Concentrate the variance

f = - 0.5\*log(2\*pi) - 0.5\*log(s2) - 0.5\*e.^2/s2 ;

lf = -mean( f );

end

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

% Log-likelihood function of unconstrained model

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

function lf = neglog1(b,ct,yt)

t = length(ct);

e = ct - b(1) - b(2)\*yt.^b(3);

s2 = e'\*e/t; % Concentrate the variance

f = - 0.5\*log(2\*pi) - 0.5\*log(s2) - 0.5\*e.^2/s2;

lf = -mean( f );

end

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

% Log-likelihood function of constrained model at each observation

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

function lf = lnlt1(b,ct,yt)

t = length(ct);

e = ct - b(1) - b(2)\*yt.^b(3);

s2 = e'\*e/t; % Concentrate the variance

lf = - 0.5\*log(2\*pi) - 0.5\*log(s2) - 0.5\*e.^2/s2;

end

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

%

% Program to estimate a nonlinear regression model

%

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

function nls\_regression1( )

clear all

clc

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

t = 100;

% Generate the data

theta = 2;

x = 2 + rand(t,1);

u = randn(t,1);

y = 1./( x - theta) + u;

% Method of Scoring

start = 1.5;

g = mean( (y - 1./(x - start)) .\* (1./(x - start).^2) );

i = mean( (1./(x - start).^4) );

theta1 = start + inv(i)\*g;

disp('Method of Scoring')

disp(['Starting value of theta = ',num2str(start) ]);

disp(['Updated value of theta = ',num2str(theta1) ]);

% Estimate the model using BGS and compute Hessian se

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

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

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

disp(' ');

disp( ['BFGS estimate of theta = ',num2str(bhat) ])

disp( ['Std. error of theta = ',num2str(sqrt(vc)) ])

end

%

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

%

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

% Log-likelihood function

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

function lf = neglog(b,y,x)

m = 1./(x - b);

s2 = 1;

lt = - 0.5\*log(2\*pi) - 0.5\*log(s2) - 0.5\*((y - m).^2)/s2;

lf = -mean(lt);

end

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

%

% a Program to estimate a nonlinear regression model

%

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

function nls\_regression2( )

clear all

clc

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

t = 100;

% Parameters

beta0 = 10;

beta1 = 2;

beta2 = 0.5;

sig2 = 0.1;

% Generate the data

x = rand(t,1).^2;

u = randn(t,1);

y = (beta0 + beta1\*x + sqrt(sig2)\*u).^(1/beta2);

% Estimate the model using BGS and compute Hessian se

start = [beta0 ; beta1; beta2 ];

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

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

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

disp(' ');

disp( ' Estimates ' )

disp( bhat )

end

%

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

%

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

% Log-likelihood function

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

function lf = neglog(b,y,x)

t = length(y);

m = b(1) + b(2)\*x;

u = y.^b(3) - m;

% Concentrate the likelihood

s2 = u'\*u/t;

lt = - 0.5\*log(2\*pi) - 0.5\*log(s2) + log(b(3)) + (b(3)-1)\*log(y) - 0.5\*(u.^2)/s2;

lf = -mean(real(lt));

end

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

%

% Program to test a liquidity trap for the United States

%

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

function nls\_liquiditytrap( )

clear all

clc

% Load data for the United States: January 1959 to December 2011

% Variables are

% m2

% gdp (real)

% cpi

% interest

load us\_liquiditytrap

% Construct variables

y = log(m2./cpi);

x1 = log(gdp);

x2 = interest/100;

t = length(y);

% Estimate linear model by OLS

x = [ones(t,1) x1 x2 ];

bols = x\y;

u = y - x\*bols;

s2 = u'\*u/t;

% LM test (2-step)

% Intercept is used in the construction of the test

% so the restricted model has an intercept. \*\*/

% Stage 1 regression

x = [ones(t,1) x1 1./x2 ];

b1 = x\y;

u = y - x\*b1;

% Stage 2 regression

z = [x 1./(x2.^2) ];

b2 = z\u;

v = u - z\*b2;

r2 = 1 - (v'\*v)/(u'\*u);

lm = t\*r2;

disp(' ')

disp(['LM test (2-step) with intercept = ',num2str(lm) ]);

disp(['p-value = ',num2str(1-chi2cdf(lm,1)) ]);

% Estimate model by MLE

start = bols([2 1 3]); % Note change of order of OLS estimates

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

[ bhat,~,~,~,~,hess ] = fminunc(@(b) neglog(b,y,x1,x2),start);

% fminunc hess doesn't seem correct

hess = numhess(@neglog,bhat,y,x1,x2 );

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

% Wald test

wd = (bhat(3) - 0)^2/vc(3,3);

disp(' ')

disp(['Wald test = ',num2str(wd) ]);

disp(['p-value = ',num2str(1-chi2cdf(wd,1)) ]);

end

%

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

%

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

% Negative log-likelihood function

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

function lf = neglog(b,y,x1,x2)

t = length(y);

m = b(1)\*x1 + b(2)./(x2 - b(3));

u = y - m;

s2 = u'\*u/t;

lnl = - 0.5\*log(2\*pi) - 0.5\*log(s2) - 0.5\*((y - m).^2)/s2;

lf = -mean( lnl );

end

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

%

% Vuoung's Nonnested Test

%

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

function nls\_money( )

clear all;

clc;

% Load data

load moneydemand.mat

mt = m2./cpi;

yt = gdp./cpi;

rt = tbill/100;

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

% Estimate model 1

y = mt; % Estimate Model 1

x = [ones(t,1) rt yt];

b = x\y;

sig2 = ((y - x\*b)'\*(y - x\*b))/t;

lf1 = model1( [ b; sig2],y,x );

% Estimate Model 2

y = log(mt);

x = [ones(t,1) log(rt) log(yt)];

b = x\y;

sig2 = ((y - x\*b)'\*(y - x\*b))/t;

lf2 = model2( [ b; sig2],y,x );

% Perform Vuong's test

dt = lf1 - lf2;

dbar = mean( dt );

sbar = std( dt,1 );

V = sqrt(t)\*(dbar/sbar);

disp('Vuongs test and p-value');

disp('------------------------');

disp([ V normcdf(V,0,1) ] );

end

%

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

%

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

% Likelihood function model 1

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

function lf = model1( b,y,x )

u = y-b(1)\*x(:,1)-b(2)\*x(:,2)-b(3)\*x(:,3);

lf = -0.5\*log(2\*pi) - 0.5\*log( b(4) ) - u.^2/(2\*b(4)) + log(abs(1.0));

end

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

% Likelihood function model 2

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

function lf = model2( b,y,x )

u = y - b(1)\*x(:,1) - b(2)\*x(:,2) - b(3)\*x(:,3);

lf = -0.5\*log(2\*pi) - 0.5\*log( b(4) ) - u.^2/(2\*b(4)) ...

+ log( abs( 1.0./exp( y ) ) );

end

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

%

% Robust estimation of the CAPM model

% Monthly excess returns for the company Martin-Marietta adn the value

% weighted CRSP index for the period Jan. 1982 to Dec. 1986 are taken from

% Butler et. al. Review of Economics and Statistics (1990), Table A1.

%

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

function nls\_capm( )

clear all;

clc;

% Load and plot the data

data = [ -0.1365 -0.0300;

-0.0769 -0.0584;

-0.0575 -0.0181;

0.0526 0.0306;

-0.0449 -0.0397;

-0.0859 -0.0295;

-0.0742 -0.0316;

0.6879 0.1176;

-0.0770 0.0075;

0.0850 0.1098;

0.0030 0.0408;

0.0754 0.0095;

-0.0412 0.0301;

-0.0890 0.0221;

0.2319 0.0269;

0.1087 0.0655;

0.0375 -0.0030;

0.0958 0.0325;

0.0174 -0.0374;

-0.0724 0.0049;

0.0750 0.0105;

-0.0588 -0.0257;

-0.0620 0.0186;

-0.0378 -0.0155;

0.0169 -0.0165;

-0.0799 -0.0440;

-0.0147 0.0094;

0.0106 -0.0028;

-0.0421 -0.0591;

-0.0036 0.0158;

0.0876 -0.0238;

0.1025 0.1031;

-0.0499 -0.0065;

0.1953 -0.0067;

-0.0714 -0.0167;

0.0469 0.0188;

0.1311 0.0733;

0.0461 0.0105;

-0.0328 -0.0070;

-0.0096 -0.0099;

0.1272 0.0521;

-0.0077 0.0117;

0.0165 -0.0099;

-0.0150 -0.0102;

-0.1479 -0.0428;

-0.0065 0.0376;

0.0390 0.0628;

0.0223 0.0391;

-0.0690 0.0002;

0.1338 0.0688;

0.1458 0.0486;

0.0063 -0.0174;

0.0692 0.0460;

-0.0239 0.0100;

-0.0568 -0.0594;

0.0814 0.0680;

-0.0889 -0.0839;

-0.0887 0.0481;

0.1037 0.0136;

-0.1163 -0.0322; ];

t = length(data);

r = data(:,1);

m = data(:,2);

set(0,'defaulttextinterpreter','latex');

figure(1);

clf;

plot(m,r,'.k')

title('Excess returns: Martin-Marietta and CRSP index');

xlabel('$m\_t$');

ylabel('$r\_t$');

box off;

% Estimate the model by OLS

Y = r;

X = [ ones(t,1) m ] ; % Create X matrix by concatenation

bols = X\Y;

s2ols = mean((Y - X\*bols).^2);

omols = s2ols\*inv(X'\*X);

seols = sqrt(diag(omols));

disp('OLS Estimates and std errors');

disp( [ bols seols ] );

disp( ['Residual variance = ', num2str(s2ols)] );

% Wald test of beta2 = 1

R = [ 0 1 ];

Q = 1;

W = (R\*bols - Q)\*inv(R\*omols\*R')\*(R\*bols - Q);

disp( ' ' );

disp('Wald statistic and p-value -- OLS')

disp('---------------------------------')

disp( [ W 1-chi2cdf(W,1) ] );

% Estimate the model by ML

theta0 = [bols; s2ols; 3 ];

[bhat,~,~,~,~,H] = fminunc( @(b) neglog( b,r,m ),theta0);

invH = inv( H );

format short

disp('Parameter Estimates and Std.Errors (Hessian)');

disp('---------------------------------------------');

disp( [bhat sqrt( diag( invH/t ) ) ] );

% Wald test that beta2 = 1 using ML results

R = [0 1 0 0 ];

Q = 1;

W = t\*(R\*bhat - Q)\*inv(R\*H\*R')\*(R\*bhat - Q);

disp( ' ' );

disp('Wald statistic and p-value -- MLE')

disp('---------------------------------')

disp( [ W 1-chi2cdf(W,1) ] );

end

%

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

%

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

% Negative log-likelihood function

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

function logl = neglog(b,y,x)

logl = -mean( lnltst(b,y,x) );

end

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

% Log-likelihood function for a Student t disturbance

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

function loglt = lnltst( b,y,x )

u = y - (b(1) + b(2)\*x);

s2 = abs( b(3) );

gam = abs( b(4) );

z = u./sqrt(s2);

const = gamma( (gam+1)/2 ) / ( sqrt(pi\*(gam-2)) \* gamma( gam/2 ) );

loglt = log(const) - 0.5\*log(s2) - 0.5\*(gam+1)\*log( 1 + (z.^2)/(gam-2) );

end

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

%

% Program to estimate a stochastic frontier model

%

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

function nls\_frontier( )

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',1) )

% Set parameter values

beta0 = 1;

beta1 = 0.5;

sig1 = 1;

sig2 = 1.5;

t = 1000;

x = randn(t,1);

% Compute density and cumulative distribution function

h = 0.01;

u = seqa(-10,h,1501);

f = (1/sig2)\*exp( sig1^2/(2\*sig2^2) + u/sig2 ).\*normcdf( -(u + sig1^2/sig2)/sig1 );

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

%\*\*\*

%\*\*\* Generate graph

%\*\*\*

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

% Switch off TeX interpreter and clear figure

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

figure(1);

clf;

plot(u,f,'-k')

ylabel('$f(u)$');

xlabel('$u$');

set(gca,'YTick',[0.0 0.1 0.2 0.3]');

box off

% Print the tex file to the relevant directory

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

% Monte Carlo simulation

ndraws = 5000;

theta = zeros(ndraws,4);

% cdf to be used as a look-up table in inverse cdf method

cdf = h\*cumsum(f);

y = zeros(t,1);

for i = 1:ndraws

% Generate random numbers for y

for j = 1:t

[~,ind] = min(abs(cdf - rand));

y(j) = beta0 + beta1\*x(j) + u(ind);

end

% Estimate the model

theta0 = [beta0 ; beta1 ; sig1 ; sig2 ];

bhat = fminsearch(@(b) neglog(b,y,x),theta0);

theta(i,:) = abs(bhat);

end

disp( ' beta0 beta1 sig1 sig2');

disp( ['Population parameters = ', num2str(theta0') ]);

disp( ['Mean = ', num2str(mean(theta)) ]);

%print "Bias = " meanc(theta\_mle)'-theta0';

%print "MSE = " meanc((theta\_mle-theta0').^2)';

end

%

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

%

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

% Log-likelihood function of the stochastic frontier model

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

function lnf = neglog(b,y,x)

u = y - b(1) - b(2)\*x;

sig1 = abs(b(3));

sig2 = abs(b(4));

% Normal - exponential likelihood

lf = - log(sig2) + 0.5\*sig1^2/sig2^2 + u/sig2 ...

+ log(normcdf( -(u + sig1^2/sig2 )/sig1 ) );

% Normal - halfnormal likelihood

% sigs = sqrt( sig2^2 + sig1^2 );

% lam = sig2/sig1;

% lf = 0.5\*log(2/pi) - log(sigs) + log(cdfn( -s\*e\*lam/sigs )) ...

% - 0.5\*e.^2/sigs^2 ;

lnf = -mean( lf );

end