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

%

% Simulating a regression model with autocorrelation

%

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

clear all;

clc;

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

t = 200;

% Model parameters

beta0 = 2;

beta1 = 1;

rho1 = 0.95;

delta1 = 0.95;

sigma = 3;

% Generate the exogenous variable

x = 0.5\*(1:1:t+100)' + randn(t+100,1);

% Simulate a regression model with an AR(1) disturbance term

v = sigma\*randn(t+100,1);

u = zeros(t+100,1);

y = zeros(t+100,1);

for i=2:t+100

u(i) = rho1\*u(i-1) + v(i);

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

end

y\_ar1 = y;

% Simulate a regression model with a MA(1) disturbance term

v = sigma\*randn(t+100,1);

u = zeros(t+100,1);

y = zeros(t+100,1);

for i=2:t+100

u(i) = v(i) + delta1\*v(i-1);

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

end

y\_ma1 = y;

% Trim data to overcome startup problems

y\_ar1 = y\_ar1(101:end,:);

y\_ma1 = y\_ma1(101:end,:);

x = x(101:end,:);

mu = beta0 + beta1\*x;

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

%\*\*\*

%\*\*\* Plot the series

%\*\*\*

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

% Switch off TeX interpreter and clear figure

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

figure(1);

clf;

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

% Panel (a)

subplot(1,2,1);

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

x,y\_ar1,'.k','MarkerSize',6);

title('(a) AR(1) Regression Model');

ylabel('$y\_t$');

xlabel('$x\_t$');

set(gca,'XTick',40:20:160);

set(gca,'YTick',20:20:160);

xlim([40,160]);

ylim([20,160]);

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

box off;

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

% Panel (b)

subplot(1,2,2);

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

x,y\_ma1,'.k','MarkerSize',6);

title('(b) MA(1) Regression Model');

ylabel('$y\_t$');

xlabel('$x\_t$');

set(gca,'XTick',40:20:160);

set(gca,'YTick',20:20:160);

xlim([40,160]);

ylim([20,160]);

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

box off;

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

% Simulate a regression model with an AR(2) disturbance term

t = 200;

beta0 = 2;

beta1 = 1;

rho1 = 0.1;

rho2 = -0.9;

sigma = 3;

x = 0.5\*(1:1:t+100)' + randn(t+100,1);

v = sigma\*randn(t+100,1);

u0 = [0;0];

rho = [rho1;rho2];

for i = 3:length(v+1)

u0(i,1) = v(i-1) + rho(1)\*u0(i-1) + rho(2)\*u0(i-2);

end

u = u0;

y = beta0 + beta1\*x + u;

y\_ar2 = y(101:end,:);

% Simulate a regression model with an ARMA(2,2) disturbance term

t = 200;

beta0 = 2;

beta1 = 1;

rho1 = 0.1;

rho2 = -0.9;

delta1 = 0.3;

delta2 = 0.2;

sigma = 3;

x = 0.5\*(1:1:t+100)' + randn(t+100,1); % xt is generated from a trend with normal additive errors

v = sigma\*randn(t+100,1); % vt is N(0,sigma^2)

tmp = [0; 0; (v(3:end,:) + delta1\*v(2:(end-1),:) + delta2\*v(1:(end-2),:))];

u0 = [0;0];

rho = [rho1;rho2];

for i = 3:length(tmp+1)

u0(i,1) = tmp(i-1) + rho(1)\*u0(i-1) + rho(2)\*u0(i-2);

end

u = u0;

y = beta0 + beta1\*x + u;

y\_arma = y(101:end,:);

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

%

% Program to estimate a dynamic investment model

% U.S. quarterly data for period 1957 to 2007

%

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

function auto\_invest( )

clear all;

clc;

% Load data

load usinvest.txt

cpi = usinvest(:,1);

gdp = usinvest(:,2);

invest = usinvest(:,3);

r10yr = usinvest(:,4);

r3yr = usinvest(:,5);

tbill = usinvest(:,6);

% Generate variables: data start in 1958Q1

gfc = [zeros(length(usinvest)-13,1) ; ones(13,1) ] ;

dri = 100\*(log(trimr(invest./cpi,1,0)) - log(trimr(invest./cpi,0,1)));

inf = 100\*log(trimr(cpi,1,0)./trimr(cpi,0,1));

rint = trimr(r10yr/4,1,0) - inf;

dry = 100\*( log(trimr(gdp./cpi,1,0)) - log(trimr(gdp./cpi,0,1)) );

gfc = trimr(gfc,1,0);

% OLS regression

t = length(dry);

y = dri;

x = [ones(t,1),dry,rint];

b = x\y;

e = y - x\*b;

s2 = mean(e.^2);

% Exact MLE

theta0 = [b ; 0.02 ; sqrt(s2)] ;

[theta,fe,~,~,~,H] = fminunc(@(p) neglog(p,dri,dry,rint),theta0);

invH = inv(H);

disp('Results for exact MLE');

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

disp('Parameter estimates and std. errors');

disp( [theta diag(invH)/t] );

% Conditional MLE

[theta,fc,~,~,~,H] = fminunc(@(p) neglogc(p,dri,dry,rint),theta);

invH = inv(H);

disp('Results for conditional MLE');

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

disp('Parameter estimates and std. errors');

disp( [theta diag(invH)/t] );

end

%

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

%

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

% Exact log-likelihood function

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

function lf = neglog(b,ri,ry,rint)

beta0 = b(1);

beta1 = b(2);

beta2 = b(3);

rho1 = b(4);

sig2 = b(5);

u = ri - beta0 - beta1\*ry - beta2\*rint;

v = trimr(u,1,0) - rho1\*trimr(u,0,1);

% Log-likelihood for t=1

lnl\_0 = - 0.5\*log(2\*pi) - 0.5\*log(sig2) + 0.5\*log(1 - rho1^2) ...

- 0.5\*(u(1) - 0).^2/(sig2/(1 - rho1^2));

% Log-likelihood for t=2,3,...

lnl\_1 = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

lf = -mean( [lnl\_0 ; lnl\_1] );

end

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

% Conditional log-likelihood function

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

function lf = neglogc(b,ri,ry,rint)

beta0 = b(1);

beta1 = b(2);

beta2 = b(3);

rho1 = b(4);

sig2 = b(5);

u = ri - beta0 - beta1\*ry - beta2\*rint;

v = trimr(u,1,0) - rho1\*trimr(u,0,1);

% Log-likelihood for t=2,3,...

lnl\_1 = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

lf = -mean( lnl\_1 );

end

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

%

% Program to estimate and test a dynamic investment model

% U.S. quarterly data for period 1957 to 2007

%

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

function auto\_test( )

clear all;

clc;

% Load data

load usinvest.txt

cpi = usinvest(:,1);

gdp = usinvest(:,2);

invest = usinvest(:,3);

r10yr = usinvest(:,4);

r3yr = usinvest(:,5);

tbill = usinvest(:,6);

% Generate variables: data start in 1958Q1

gfc = [zeros(length(usinvest)-13,1) ; ones(13,1) ] ;

dri = 100\*(log(trimr(invest./cpi,1,0)) - log(trimr(invest./cpi,0,1)));

inf = 100\*log(trimr(cpi,1,0)./trimr(cpi,0,1));

rint = trimr(r10yr/4,1,0) - inf;

dry = 100\*( log(trimr(gdp./cpi,1,0)) - log(trimr(gdp./cpi,0,1)) );

gfc = trimr(gfc,1,0);

% OLS regression

t = length(dry);

y = dri;

x = [ones(t,1),dry,rint];

b = x\y;

e = y - x\*b;

s2 = mean(e.^2);

% Unconstrained model

theta = [b ; 0.02 ; sqrt(s2)] ;

[theta1,f1,~,~,~,H1] = fminunc(@(p) neglog1(p,dri,dry,rint),theta);

lnl1 = -f1;

invH1 = inv(H1);

disp('Results for conditional MLE');

disp(['Log-likelihood function = ', num2str(lnl1)]);

disp('Parameter estimates and std. errors');

disp( [theta diag(invH1)/t] );

% Constrained model

theta = [b ; sqrt(s2)] ;

[theta0,f0,~,~,~,H0] = fminunc(@(p) neglog0(p,dri,dry,rint),theta);

invH0 = inv(H0);

lnl0 = -f0;

disp('Results for conditional MLE');

disp(['Log-likelihood function = ', num2str(lnl0)]);

disp('Parameter estimates and std. errors');

disp( [theta diag(invH0)/t] );

% LR test

lr = -2\*t\*(lnl0 - lnl1);

disp(['LR statistic = ',num2str(lr) ]);

disp(['p-value = ',num2str(1-cdf('chi2',lr,1)) ]);

% Wald test

r = [ 0 , 0 , 0 , 1 , 0 ];

q = 0;

wd = t\*(r\*theta1 - q)'\*inv(r\*inv(H1)\*r')\*(r\*theta1 - q);

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

disp(['p-value = ',num2str(1-cdf('chi2',wd,1)) ]);

% LM test

theta = [ theta0(1:3) ; 0.0 ; theta0(4) ];

gmat = numgrad(@lnlt1,theta,dri,dry,rint);

g = mean(gmat)';

j = gmat'\*gmat/t;

lm = t\*g'\*inv(j)\*g;

disp('Gradient evaluated at contrained estimates');

disp( g );

disp('Outer product of gradients matrix');

disp( j );

disp(['LM statistic = ',num2str(lm) ]);

disp(['p-value = ',num2str(1-cdf('chi2',lm,1)) ]);

% LM test (regression form)

% Stage 1 regression

b = x\y;

u = y - x\*b;

% Stage 2 regression

y = trimr(u,1,0);

z = [trimr(x,1,0) , trimr(e,0,1)];

b = z\y;

e = y - z\*b;

r2= 1 - ((y-mean(y))'\*(y-mean(y)))\e'\*e;

lm = (t-1)\*r2;

disp(['LM statistic (regression) = ',num2str(lm) ]);

disp(['p-value = ',num2str(1-cdf('chi2',lm,1)) ]);

% LM test (first-order autocorrelation of residuals)

y = dri;

b = x\y;

u = y - x\*b;

r1 = autocorr(u,1,0);

lm = (t-1)\*r1(2)^2;

disp(['LM statistic (alternative) = ',num2str(lm) ]);

disp(['p-value = ',num2str(1-cdf('chi2',lm,1)) ]);

end

%

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

%

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

% Negative unconstrained log-likelihood

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

function lf = neglog1(b,ri,ry,rint)

lf = -mean( lnlt1(b,ri,ry,rint) );

end

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

% Unconstrained log-likelihood function at each observation

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

function lnl = lnlt1(b,ri,ry,rint)

beta0 = b(1);

beta1 = b(2);

beta2 = b(3);

rho1 = b(4);

sig2 = b(5);

u = ri - beta0 - beta1\*ry - beta2\*rint;

v = trimr(u,1,0) - rho1\*trimr(u,0,1);

% Log-likelihood for t=2,3,...

lnl = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

end

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

% Negative constrained log-likelihood function

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

function lf = neglog0(b,ri,ry,rint)

beta0 = b(1);

beta1 = b(2);

beta2 = b(3);

rho1 = 0.0;

sig2 = b(4);

u = ri - beta0 - beta1\*ry - beta2\*rint;

v = trimr(u,1,0) - rho1\*trimr(u,0,1);

% Log-likelihood for t=2,3,...

lnl = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

lf = - mean( lnl );

end

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

%

% Simulation example to reproduce the asymptotic distribution of the

% MLE estimator for the regression model with autocorrelation.

%

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

function auto\_distribution( )

clear all;

clc;

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

% Set parameter values

beta0 = 1.0;

beta1 = 1.0;

rho1 = 0.6;

sig2 = 10;

theta0 = [ beta0 ; beta1 ; rho1 ; sig2 ]; % Start values

t = 500;

ndraws = 5000;

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

% Simulate a regression model with an AR(1) disturbance term

x = rand(t,1) - 0.5; % x fixed in repeated samples

%exact = zeros(ndraws,1);

cond = zeros(ndraws,1);

for k = 2:ndraws

v = sqrt(sig2)\*randn(t,1);

u = recserar( v , sqrt(1/(1-rho1^2))\*v(1) , rho1 );

y = beta0 + beta1\*x + u;

% Exact MLE

%[ tmp ] = fminunc(@(p) negloge(p,y,x),theta0,options);

%exact(k) = tmp(2);

% Conditional MLE

[ tmp ] = fminunc(@(p) neglogc(p,y,x),theta0,options);

cond(k) = tmp(2);

end

mse = mean( (cond - beta1).^2 );

ssq = sum( ( trimr(x,1,0) - rho1\*trimr(x,0,1) ).^2 );

i\_beta = ssq/sig2; % Inforation matrix of beta hat

var\_beta1 = inv(i\_beta); % Asymptotic variance

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

disp( [ 'Sum of squares = ' num2str(ssq) ] );

disp( [ 'Asymptotic variance (theoretical) = ' num2str(var\_beta1) ] );

disp( [ 'Asymptotic variance (simulated) = ' num2str(mse) ] );

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

%\*\*\*

%\*\*\* Generate graph

%\*\*\*

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

% Switch off TeX interpreter and clear figure

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

figure(1);

clf;

z = ( cond - beta1 )/sqrt( mse );

tt = -5:0.1:5;

[fcdf,xx] = ecdf(z);

[f,bins] = ecdfhist(fcdf,xx,31);

bar(bins,f,'hist');

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

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

box off

hold on

plot(tt,normpdf(tt),'-k','LineWidth',0.75)

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

xlabel('$z$');

set(gca,'YTick',[0.1 0.2 0.3 0.4 0.5])

set(gca,'XTick',[-5 -4 -3 -2 -1 0 1 2 3 4 5])

hold off

% Print the tex file to the relevant directory

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

end

%

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

%

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

% Exact log-likelihood function

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

function lf = negloge(b,y,x)

beta0 = b(1);

beta1 = b(2);

rho1 = tanh(b(3)); % Stay in the unit circle

sig2 = abs(b(4)); % Variance is positive

fac = 1 - rho1^2;

u = y - beta0 - beta1\*x;

v = trimr(u,1,0) - rho1\*trimr(u,0,1);

tmp = -0.5\*log(2\*pi)-0.5\*log(sig2)+0.5\*log(fac)-0.5\*(u(1))^2/(sig2/fac);

tmp1 = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

lf = - sum( [ tmp; tmp1 ] );

end

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

% Conditional log-likelihood function

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

function lf = neglogc(b,y,x)

beta0 = b(1);

beta1 = b(2);

rho1 = tanh(b(3)); % Stay in the unit circle

sig2 = abs(b(4)); % Variance is positive

u = y - beta0 - beta1\*x;

v = trimr(u,1,0) - rho1\*trimr(u,0,1);

tmp = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

lf = - sum( tmp );

end

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

%

% Program to compare the efficiency properties of MLE and OLS

% in the AR(1) regression model in the case where the

% explanatory variable is a constant. In this case the OLS estimator is

% the sample mean

%

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

clear all;

clc;

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

rho = 0.6;

sig2 = 10;

T = [5, 50, 500];

% Compute OLS variance

for k = 1:length(T)

t = T(k);

sum = 0.0;

for i = 1:t-1

for j = 1:t-i

sum = sum + rho^i;

end

end

var\_ols = sig2\*(t + 2\*sum)/((1-rho^2)\*t^2);

% OLS estimator asymptotic variance based on Harvey (1990, p.197)

sum\_h = (t\*(1-rho^2) - 2\*rho\*(1-rho^t))/((1-rho)^2);

var\_ols\_harvey = sig2\*(sum\_h)/( (1-rho^2)\*t^2 );

% Compute MLE variance

var\_mle = sig2/((t-1)\*(1-rho)^2);

% Print results

disp(['Sample size = ', num2str(T(k)) ]);

disp(['Variance of OLS = ', num2str(var\_ols) ]);

disp(['Variance of OLS (Harvey) = ', num2str(var\_ols) ]);

disp(['Variance of MLE = ', num2str(var\_mle) ]);

disp(['Efficiency (OLS/MLE) = ', num2str(var\_ols/var\_mle) ]);

end

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

%

% Simulation example to compare the asymptotic distribution of the MLE

% and the OLS estimators of the autocorrelated regression model for

% alternative assumptions about the explanatory variable xt

%

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

function auto\_efficiency( )

clear all;

clc;

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

% Simulate a regression model with an AR(1) disturbance term

t = 200;

beta0 = 1.0;

beta1 = 1.0;

rho1 = 0.6;

sigma2 = 0.0036;

theta0 = [beta0; beta1; rho1; sigma2];

phi1 = 0.6;

sigma2\_w = 0.0036;

% Two instances where xt is treated as fixed

%x = (1:1:t)'%.\*randn(t,1);

% x = sin(2\*pi\*[0:1:t-1]'/t);

ndraws = 500;

theta\_exact = zeros(ndraws,4);

theta\_cond = zeros(ndraws,4);

theta\_ols = zeros(ndraws,4);

for k = 1:ndraws

% Generate data

v = sqrt(sigma2)\*randn(t,1);

u = recserar( v , sqrt(1/(1-rho1^2))\*v(1) , rho1 );

w = sqrt(sigma2\_w)\*randn(t,1);

% For the case where xt is not fixed

x = recserar( w , sqrt(1/(1-phi1^2))\*w(1) , phi1 );

y = beta0 + beta1\*x + u;

% Exact MLE

flag = 0;

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

theta\_exact(k,:) = [ theta(1:2) ; tanh(theta(3)) ; abs(theta(4))]';

% Conditional MLE

flag = 1;

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

theta\_cond(k,:) = [ theta(1:2) ; tanh(theta(3)) ; abs(theta(4)) ]';

% OLS

b\_ols = [ones(t,1),x]\y;

e = y - [ones(t,1),x]\*b\_ols;

sig2\_ols = mean(e.^2);

rho\_ols = e(1:end-1)\e(2:end);

theta\_ols(k,:) = [b\_ols' , sig2\_ols , rho\_ols];

end

% Compute statistics of sampling distributions and print results

mse\_exact = mean(((theta\_exact - repmat(theta0',ndraws,1)).^2));

mse\_cond = mean(((theta\_cond - repmat(theta0',ndraws,1)).^2));

mse\_ols = mean(((theta\_ols - repmat(theta0',ndraws,1)).^2));

disp('Beta0 Beta1 Rho1 Sigma^2');

disp(['Population parameter = ' num2str(theta0')]);

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

disp(['Mean (exact MLE) = ' num2str(mean(theta\_exact))]);

disp(['Bias (exact MLE) = ' num2str(100\*(mean(theta\_exact)-theta0'))]);

disp(['MSE (exact MLE) = ' num2str(100\*mse\_exact)]);

disp(['RMSE (exact MLE) = ' num2str(sqrt(mse\_exact))]);

disp(' ');

disp(['Mean (cond. MLE) = ' num2str(mean(theta\_cond))]);

disp(['Bias (cond. MLE) = ' num2str(100\*(mean(theta\_cond)-theta0'))]);

disp(['MSE (cond. MLE) = ' num2str(100\*mse\_cond)]);

disp(['RMSE (cond. MLE) = ' num2str(sqrt(mse\_cond))]);

disp(' ');

disp(['Mean (OLS) = ' num2str(mean(theta\_ols))]);

disp(['Bias (OLS) = ' num2str(100\*(mean(theta\_ols)-theta0'))]);

disp(['MSE (OLS) = ' num2str(100\*mse\_ols)]);

disp(['RMSE (OLS) = ' num2str(sqrt(mse\_ols))]);

disp(' ');

disp(['Efficiency (cond/exact) = ' num2str((mse\_cond./mse\_exact))]);

disp(['Efficiency (ols/exact) = ' num2str((mse\_ols./mse\_exact))]);

end

%

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

%

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

% Negative log-likelihood function

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

function lnl = neglog(b,y,x,flag)

beta0 = b(1);

beta1 = b(2);

rho1 = tanh(b(3)); % rho1 stays in the unit circle

sig2 = abs(b(4)); % variance is positive

u = y - beta0 - beta1\*x;

v = u(2:end) - rho1\*u(1:end-1);

lnl\_0 = - 0.5\*log(2\*pi) - 0.5\*log(sig2) + 0.5\*log(1 - rho1^2) ...

- 0.5\*(u(1) - 0).^2/(sig2/(1 - rho1^2));

lnl\_1 = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

if flag

lnl = -mean(lnl\_1);

else

lnl = -mean( [lnl\_0 ; lnl\_1] );

end

end

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

%

% Simulation example to compure the asymptotic distribution of the MLE

% estimator and the Hatanaka estimator for the regression model with

% autocorrelation.

%

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

function auto\_hatanaka( )

clear all;

clc;

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

t = 1000000;

beta0 = 1.0;

beta1 = 1.0;

alpha1 = 0.5;

rho1 = 0.6;

sigma2 = 0.1;

theta0 = [beta0, beta1, alpha1, rho1, sigma2];

x = rand(t,1) - 0.5;

v = sqrt(sigma2)\*randn(t,1);

u = recserar(v, sqrt(1/(1-rho1^2))\*v(1), rho1); % Beach and MacKinnon, fn3

y = recserar(beta0 + beta1\*x + u , 0.0, alpha1);

cov\_theoretical = rho1\*sigma2/((1-alpha1\*rho1)\*(1-rho1^2));

cov\_simulated = cov(trimr(y,0,1), trimr(u,1,0));

disp('Theoretical and simulated covariance');

disp([ cov\_theoretical, cov\_simulated(1,2) ]);

% Monte Carlo replications

t = 1000;

ndraws = 1000;

x = rand(t,1) - 0.5;

theta\_cond = zeros(ndraws,5);

theta\_hatanaka = zeros(ndraws,5);

theta\_ols = zeros(ndraws,5);

for k = 1:ndraws

% Generate data

v = sqrt(sigma2)\*randn(t,1);

u = recserar( v , sqrt(1/(1-rho1^2))\*v(1) , rho1 );

y = recserar( beta0 + beta1\*x + u , 0.0, alpha1 );

% Conditional MLE

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

theta\_cond(k,:) = [theta(1) theta(2) tanh(theta(3)) tanh(theta(4)) abs(theta(5))];

% Hatanaka 2-step efficient estimator

theta = hatanaka(y,x,t);

theta\_hatanaka(k,:) = theta;

% OLS

b\_ols = [ones(t-1,1), trimr(x,1,0), trimr(y,0,1)]\trimr(y,1,0);

u\_hat = trimr(y,1,0) - [ones(t-1,1), trimr(x,1,0), trimr(y,0,1)]\*b\_ols;

rho\_ols = trimr(u\_hat,0,1)\trimr(u\_hat,1,0);

sig2\_ols = mean(u\_hat.^2);

theta\_ols(k,:) = [b\_ols', rho\_ols, sig2\_ols];

end

% Compute statistics of sampling distributions and print results

mse\_cond = mean((theta\_cond - repmat(theta0,ndraws,1)).^2);

mse\_hatanaka = mean((theta\_hatanaka - repmat(theta0,ndraws,1)).^2);

mse\_ols = mean((theta\_ols - repmat(theta0,ndraws,1)).^2);

disp(' Beta0 Beta1 Alpha1 Rho1 Sigma^2');

disp(['Population parameter ' num2str(theta0)]);

disp(['Mean (cond. MLE) ' num2str(mean(theta\_cond))]);

disp(['Bias(x100) (cond. MLE) ' num2str(100\*(mean(theta\_cond)-theta0))]);

disp(['MSE(x100) (cond. MLE) ' num2str(100\*mse\_cond)]);

disp(['RMSE(x100) (cond. MLE) ' num2str(100\*sqrt(mse\_cond))]);

disp(['Mean (Hatanaka) ' num2str(mean(theta\_hatanaka))]);

disp(['Bias(x100) (Hatanaka) ' num2str(100\*(mean(theta\_hatanaka)-theta0))]);

disp(['MSE(x100) (Hatanaka) ' num2str(100\*mse\_hatanaka)]);

disp(['RMSE(x100) (Hatanaka) ' num2str(100\*sqrt(mse\_hatanaka))]);

disp(['Mean (OLS) ' num2str(mean(theta\_ols))]);

disp(['Bias(x100) (OLS) ' num2str(100\*(mean(theta\_ols)-theta0))]);

disp(['MSE(x100) (OLS) ' num2str(100\*mse\_ols)]);

disp(['RMSE(x100) (OLS) ' num2str(100\*sqrt(mse\_ols))]);

disp(['Efficiency (hatanaka/cond) ' num2str((sqrt(mse\_hatanaka)./sqrt(mse\_cond)))]);

disp(['Efficiency (ols/cond) ' num2str((sqrt(mse\_ols)./sqrt(mse\_cond)))]);

end

%

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

%

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

% Log-likelihood function at each observation

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

function lf = neglog(b,y,x)

beta0 = b(1);

beta1 = b(2);

alpha1 = tanh(b(3)); % rho1 stays in the unit circle

rho1 = tanh(b(4)); % rho2 stays in the unit circle

sig2 = abs(b(5)); % variance is positive

u = trimr(y,1,0) - beta0 - beta1\*trimr(x,1,0) - alpha1\*trimr(y,0,1);

v = trimr(u,1,0) - rho1\*trimr(u,0,1);

lnl = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

lf = -mean( lnl );

end

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

% Hatanaka 2-step efficient estimator

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

function h1 = hatanaka(y,x,t)

yvar = trimr(y,1,0);

xvar = [ones(t-1,1), trimr(x,1,0), trimr(y,0,1)];

zvar = [ones(t-1,1), trimr(x,1,0), trimr(x,0,1)];

% IV initial estimates of mean parameters

biv = inv(zvar'\*xvar)\*(zvar'\*yvar);

u = yvar - xvar\*biv;

% Estimate rho

rho = trimr(u,0,1)\trimr(u,1,0);

yvar = trimr(y,2,0) - rho.\*trimr(y,1,1);

xvar = [ones(t-2,1), (trimr(x,2,0) - rho.\*trimr(x,1,1)), (trimr(y,1,1) - rho.\*trimr(y,0,2)), trimr(u,0,1)];

% Regression on transformed variables and update rho

b = xvar\yvar;

rho = rho + b(4);

% Compute residual variance

v = trimr(y,2,0)-([ones(t-2,1), trimr(x,2,0), trimr(y,1,1)]\*b(1:3) -rho\*(trimr(y,1,1)-[ones(t-2,1), trimr(x,1,1), trimr(y,0,2)]\*b(1:3)));

sig2 = mean(v.^2);

h1 = [(b(1:3))', rho, sig2];

end

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

%

% Program to estimate a simultaneous model with first order vector

% autocorrelation. The set of equations is defined as yt\*b + xt\*a = u

% where

% u = ru(-1) + v

% where yt is a (1xn) set of dependent variables at time t

% xt is a (1xk) set of explanatory variables at time t

%

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

function auto\_system( )

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',123456) )

t = 500;

% Simulate the data

%[ y,x ] = simulatedata( t );

% Load GAUSS data to reproduce results

load system.mat

y = gaussdata(:,[1 2]);

x = gaussdata(:,[3 4]);

% Estimate the unconstrained model

theta = [0.6; 0.4; 0.2; -0.5; 0.8; 0.1;-0.2; 0.6];

[theta1,a1,~,~,~,H] = fminunc(@(theta) neglog1(theta,y,x),theta);

lnl1 = -(t-1)\*a1; % Unconstrained log-likelihood

vcov1 = inv(H);

% Estimate the constrained model

theta = [0.6; 0.4; 0.2; -0.5];

[theta0,a0] = fminunc(@(theta) neglog0(theta,y,x),theta);

lnl0 = -(t-1)\*a0; % Constrained log-likelihood

% Estimate the constrained model with independent disturbances

theta = [0.6; 0.4; 0.2; -0.5; 0.8; 0.1];

[theta2,a2] = fminunc(@(theta) neglog2(theta,y,x),theta);

lnl2 = -(t-1)\*a2; % Constrained log-likelihood

disp(' ');

disp(['Unconstrained log-likelihood function = ', num2str(lnl1) ] );

disp(['Constrained log-likelihood function = ', num2str(lnl0) ] );

disp(['Constrained log-likelihood function (independent auto) = ', num2str(t\*lnl2) ] );

disp(' ');

% LR test of no autocorrelation

lr = -2\*(lnl0 - lnl1);

dof = length(theta1) - length(theta0);

disp(['LR test (no autocorrelation) = ', num2str(lr) ]);

disp(['Degrees of freedom = ', num2str(dof) ]);

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

disp(' ');

% LR test of independent autocorrelation

lr = -2\*(lnl2 - lnl1);

dof = length(theta1) - length(theta2);

disp(['LR test (indep autocorrelation) = ', num2str(lr) ]);

disp(['Degrees of freedom = ', num2str(dof) ]);

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

disp(' ');

% Wald test of no autocorrelation

r = [0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 ;

0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 ;

0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 ;

0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 ];

q = [ 0 ; 0 ; 0 ; 0];

wd = t\*(r\*theta1 - q)'\*inv(r\*vcov1\*r')\*(r\*theta1 - q);

dof = size(r,1);

disp(' ');

disp(['Wald test (no autocorrelation) = ', num2str(wd) ]);

disp(['Number of degrees of freedom = ', num2str(dof) ]);

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

% Wald test of common autocorrelation

r = [ 0 , 0 , 0 , 0 , 1 , 0 , 0 , -1 ;

0 , 0 , 0 , 0 , 0 , 1 , -1 , 0 ];

q = [0 ; 0] ;

wd = t\*(r\*theta1 - q)'\*inv(r\*vcov1\*r')\*(r\*theta1 - q);

dof = size(r,1);

disp(' ');

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

disp(['Number of degrees of freedom = ', num2str(dof) ]);

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

% Lagrange Multiplier test (based on numerical opg matix)

dof = length(theta1) - length(theta0);

theta = [theta0 ; zeros(4,1)];

gmat = numgrad(@lnlt1,theta,y,x);

g = mean(gmat)';

j = gmat'\*gmat/t;

lm = t\*g'\*inv(j)\*g;

disp(' ');

disp(['LM test (no autocorrelation) = ', num2str(lm) ]);

disp(['Number of degrees of freedom = ', num2str(dof) ]);

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

end

%

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

%

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

% Simulate the data

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

function [ y,x ] = simulatedata( t )

% Population paramaters

beta1 = 0.6;

alpha1 = 0.4;

beta2 = 0.2;

alpha2 = -0.5;

rho11 = 0.8;

rho12 = 0.1;

rho21 = -0.2;

rho22 = 0.6;

omega = [ 1 0.5 ;

0.5 1.0 ];

b = [ 1 -beta2 ;

-beta1 1 ];

a = [ -alpha1 0 ;

0 -alpha2 ];

% Exogenous variables

x = [10\*rand(t,1), 3\*randn(t,1)];

% Disturbances

v = randn(t,2)\*chol(omega);

u = zeros(t,2);

for i = 2:t;

u(i,1) = rho11\*u(i-1,1) + rho12\*u(i-1,2) + v(i,1);

u(i,2) = rho21\*u(i-1,1) + rho22\*u(i-1,2) + v(i,2);

end

% Simulate the model by simulating the reduced form

y = zeros(t,2);

i = 1;

for i = 1:t

y(i,:) = -x(i,:)\*a\*inv(b) + u(i,:)\*inv(b);

end

end

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

% Negative unconstrained log-likelihood

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

function lf = neglog1(theta,y,x)

lf = -mean( lnlt1(theta,y,x) );

end

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

% Unconstrained log-likelihood function

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

function lnl = lnlt1(theta,y,x)

[t,n] = size(y);

b = [ 1 , -theta(3) ;

-theta(1) , 1 ];

a = [ -theta(2) , 0 ;

0 , -theta(4) ];

rho = [theta(5) , theta(7) ;

theta(6) , theta(8) ];

% Construct residuals and concentrate the covariance matrix

u = zeros(t,n);

v = zeros(t,n);

for i = 2:t

u(i,:) = y(i,:)\*b + x(i,:)\*a;

v(i,:) = u(i,:) - u(i-1,:)\*rho;

end

omega = v'\*v/t;

lnl = zeros(t,1);

for i = 2:t

lnl(i) = - n\*0.5\*log(2\*pi) + log(abs(det(b))) - 0.5\*log(det(omega)) ...

- 0.5\*v(i,:)\*inv(omega)\*v(i,:)';

end

lnl = trimr(lnl,1,0);

end

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

% Negative constrained log-likelihood function

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

function lf = neglog0(theta,y,x)

lf = -mean( lnlt0(theta,y,x) );

end

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

% Constrained log-likelihood function

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

function lnl = lnlt0(theta,y,x)

[t,n] = size(y);

b = [ 1 , -theta(3) ;

-theta(1) , 1 ];

a = [ -theta(2) , 0 ;

0 , -theta(4) ];

rho = [0 , 0 ;

0 , 0 ];

u = zeros(t,n);

v = zeros(t,n);

% Construct residuals and concentrate covariance matrix

for i = 2:t

u(i,:) = y(i,:)\*b + x(i,:)\*a;

v(i,:) = u(i,:) - u(i-1,:)\*rho;

end

omega = v'\*v/t;

lnl = zeros(t,1);

for i = 2:t

lnl(i) = - n\*0.5\*log(2\*pi) + log(abs(det(b))) - 0.5\*log(det(omega)) ...

- 0.5\*v(i,:)\*inv(omega)\*v(i,:)';

end

lnl = trimr(lnl,1,0);

end

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

% Negative constrained log-likelihood function (independent auto)

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

function lf = neglog2(theta,y,x)

lf = -mean( lnlt2(theta,y,x) );

end

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

% Constrained log-likelihood function at each observation

% with independent autocorrelation

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

function lnl = lnlt2(theta,y,x)

[t,n] = size(y);

b = [ 1 , -theta(3) ;

-theta(1), 1 ];

a = [ -theta(2), 0 ;

0 , -theta(4) ];

rho = [ theta(5) , 0 ;

0 , theta(6) ];

% Construct residuals and concentrate covariance matrix

u = zeros(t,n);

v = zeros(t,n);

for i = 2:t

u(i,:) = y(i,:)\*b + x(i,:)\*a;

v(i,:) = u(i,:) - u(i-1,:)\*rho;

end

omega = v'\*v/t;

lnl = zeros(t,1);

for i = 2:t

lnl(i) = - n\*0.5\*log(2\*pi) + log(abs(det(b))) - 0.5\*log(det(omega)) ...

- 0.5\*v(i,:)\*inv(omega)\*v(i,:)';

end

lnl = trimr(lnl,1,0);

end

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

%

% Program to compute the MIC model using the hedge data

%

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

function auto\_hedge( )

clear all;

clc;

% Load daily hedge fund data (1 April 2003 - 28 May 2010)

load hedge.mat

% 1 - 7 daiy hedge fund returns (not excess returns)

% 8. Market excess return (this IS adjusted for risk free)

% 9. Risk free rate (expressed on a daily basis)

% 10. Market excess rets with gaps filled in (this IS adjusted for risk free)

% 11. Risk free rate with gaps filled in (expressed on a daily basis)

% 12. Dow daily returns in percentage (not excess)

% 13. NASDAQ daily returns in percentage (not excess)

% 14. SP500 daily returns in percentage (not excess)

% 15. Tuesday dummy

% 16. Wednesday dummy

% 17. Thursday dummy

% 18. Friday dummy

% 19. Holiday dummy

% Hedge fund excess ret

hedge = hedgedata(:,1:7) - repmat(hedgedata(:,11),1,7);

% Market excess returns

m\_dow = hedgedata(:,12) - hedgedata(:,11);

m\_nasdaq = hedgedata(:,13) - hedgedata(:,11);

m\_sp500 = hedgedata(:,14) - hedgedata(:,11);

% Dummy variables

d\_season = hedgedata(:,15:18);

d\_hol = hedgedata(:,19);

% Choose a hedge fund

y = hedge(:,3);

m = m\_sp500;

t = length(y);

% LM test applied to the CAPM without AR(1) disturbances

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

b1 = x\y;

v = y - x\*b1;

z = [ones(t-1,1), trimr(m,1,0), trimr(v,0,1)];

v = trimr(v,1,0);

b2 = z\v;

e = v - z\*b2;

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

lm = (t-1)\*r2;

disp(['LM test = ' num2str(lm)]);

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

% Estimate a CAPM model with AR(1) disturbances

theta0 = [0.1 , 0.1 , 0.0];

[theta,~,~,~,~,H] = fminunc(@(b) neglog(b,y,m),theta0);

% Compute residuals

u = y - theta(1) - theta(2)\*m;

v = trimr(u,1,0) - theta(3)\*trimr(u,0,1);

z = [ones(t-2,1), trimr(m,2,0), trimr(v,0,1)];

v = trimr(v,1,0);

b2 = z\v;

e = v - z\*b2;

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

lm = (t-1)\*r2;

disp(['LM test = ' num2str(lm)]);

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

% Wald test (applied to the model with AR(1) disturbances)

r = [0, 0, 1];

q = 0;

w = t\*(r\*theta' - q)'\*inv(r\*(H)\*r')\*(r\*theta' - q);

disp(['Wald test = ' num2str(w)]);

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

end

%

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

%

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

% Negative log-likelihood function

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

function lf = neglog(b,y,m)

beta0 = b(1);

beta1 = b(2);

rho1 = b(3);

u = y - beta0 - beta1\*m;

v = trimr(u,1,0) - rho1\*trimr(u,0,1);

sig2 = mean(v.^2);

lnl = -0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

lf = -mean( lnl );

end

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

%

% Simulation example to reproduce the Beach and MacKinnon (1978)

% Econometrica, pp.51-58 study which derives the sampling

% distributions MLE estimators of regression models with

% autocorrelation.

%

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

function auto\_beachmack( )

clear all;

clc;

RandStream.setDefaultStream( RandStream('mt19937ar','seed',12345) )

% Simulate a regression model with an AR(1) disturbance term

t = 20;

beta0 = 1.0;

beta1 = 1.0;

rho1 = 0.6;

sigma2 = 0.0036;

theta0= [beta0 ; beta1 ; rho1 ; sigma2];

% xt is fixed in repeated samples

x = exp(0.04\*(1:1:t)') + sqrt(0.0009)\*randn(t,1);

ndraws = 200; % used by Beach and MacKinnon

theta\_exact = zeros(ndraws,4);

theta\_cond = zeros(ndraws,4);

theta\_ols = zeros(ndraws,4);

for k = 1:ndraws

% Generate data

v = sqrt(sigma2)\*randn(t,1);

u = recserar( v , sqrt(1/(1-rho1^2))\*v(1) , rho1 );

y = beta0 + beta1\*x + u;

% Exact MLE

flag = 0;

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

theta\_exact(k,:) = [ theta(1:2) ; tanh(theta(3)) ; abs(theta(4))]';

% Conditional MLE

flag = 1;

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

theta\_cond(k,:) = [ theta(1:2) ; tanh(theta(3)) ; abs(theta(4)) ]';

% OLS

b\_ols = [ones(t,1),x]\y;

e = y - [ones(t,1),x]\*b\_ols;

sig2\_ols = mean(e.^2);

rho\_ols = e(1:end-1)\e(2:end);

theta\_ols(k,:) = [b\_ols' , sig2\_ols , rho\_ols];

end

% Compute statistics of sampling distributions and print results

rmse\_exact = sqrt( mean( (mean(theta\_exact) - theta0').^2 ) );

rmse\_cond = sqrt( mean( (mean(theta\_cond) - theta0').^2 ) );

rmse\_ols = sqrt( mean( (mean(theta\_ols) - theta0').^2 ) );

fprintf (' Beta0 Beta2 Rho1 Sigma^2\n');

fprintf ('Population parameter %2.3f %2.3f %2.3f %2.3f\n\n\n', theta0');

fprintf ('Mean (exact MLE) %2.3f %2.3f %2.3f %2.3f\n', mean(theta\_exact)');

fprintf ('Bias (exact MLE) %2.3f %2.3f %2.3f %2.3f\n', theta0'-mean(theta\_exact));

disp(' ')

fprintf ('Mean (cond. MLE) %2.3f %2.3f %2.3f %2.3f \n', mean(theta\_cond)');

fprintf ('Bias (cond. MLE) %2.3f %2.3f %2.3f %2.3f \n', theta0'-mean(theta\_cond));

disp(' ');

fprintf ('Mean (OLS) %2.3f %2.3f %2.3f %2.3f \n', mean(theta\_ols)');

fprintf ('Bias (OLS) %2.3f %2.3f %2.3f %2.3f \n', theta0'-mean(theta\_ols));

disp(' ');

fprintf ('\nRMSE (exact MLE) %f \n', rmse\_exact');

fprintf ('\nRMSE (cond. MLE) %f \n', rmse\_cond');

fprintf ('\nRMSE (OLS) %f \n\n', rmse\_ols');

disp(' ')

fprintf ('Efficiency (cond/exact) %f \n', (rmse\_cond./rmse\_exact)');

fprintf ('Efficiency (ols/exact) %f \n', (rmse\_ols./rmse\_exact)');

end

%

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

%

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

% Negative log-likelihood function

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

function lnl = neglog(b,y,x,flag)

beta0 = b(1);

beta1 = b(2);

rho1 = tanh(b(3)); % rho1 stays in the unit circle

sig2 = abs(b(4)); % variance is positive

u = y - beta0 - beta1\*x;

v = u(2:end) - rho1\*u(1:end-1);

lnl\_0 = - 0.5\*log(2\*pi) - 0.5\*log(sig2) + 0.5\*log(1 - rho1^2) ...

- 0.5\*(u(1) - 0).^2/(sig2/(1 - rho1^2));

lnl\_1 = - 0.5\*log(2\*pi) - 0.5\*log(sig2) - 0.5\*v.^2/sig2;

if flag

lnl = -mean(lnl\_1);

else

lnl = -mean( [lnl\_0 ; lnl\_1] );

end

end