```
/* Transforms raw data to make them stationary and standardized */
/*==============================================================*/
proc(1) = No_Transform(vecDat);
    local vecX;
    vecX = (vecDat - meanc(vecDat)) ./ stdc(vecDat);
    vecX = trimr(vecX,2,0);
retp(vecX);
endp;

proc(1) = First_Diff(vecDat);
    local vecYt,vecYt1,vecX;
    vecYt = trimr(vecDat,1,0);
    vecYt1 = trimr(vecDat,0,1);
    vecX = vecYt - vecYt1;
    vecX = ((vecX - meanc(vecX)) ./ stdc(vecX));
    vecX = trimr(vecX,1,0);
retp(vecX);
endp;

proc(1) = Logthm(vecDat);
    local vecX;
    vecX = ln(vecDat);
    vecX = ((vecX - meanc(vecX)) ./ stdc(vecX));
    vecX = trimr(vecX,2,0);
retp(vecX);
endp;

proc(1) = First_Diff_Logthm(vecDat);
    local vecX,Yt,Yt1;
    vecX = ln(vecDat);
    Yt = trimr(vecX,1,0);
    Yt1 = trimr(vecX,0,1);
    vecX = Yt - Yt1;
    vecX = ((vecX - meanc(vecX)) ./ stdc(vecX));
    vecX = trimr(vecX,1,0);
retp(vecX);
endp;

proc(1) = Sec_Diff_Logthm(vecDat);
    local vecTemp,Yt,Yt1,vecX;
    vecX = ln(vecDat);
    Yt = trimr(vecX,1,0);
    Yt1 = trimr(vecX,0,1);
    vecTemp = Yt - Yt1;
    Yt = trimr(vecTemp,1,0);
    Yt1 = trimr(vecTemp,0,1);
    vecX = Yt - Yt1;
    vecX = ((vecX - meanc(vecX)) ./ stdc(vecX));
retp(vecX);
endp;

proc(1) = Stn_Std_Matrix(matDat,vecCode);
    local inc,nvars,matX,vecTemp;

    nvars = cols(matDat);
    matX = {};
    for inc (1,nvars,1);
        if vecCode[inc] == 1;
            vecTemp = No_Transform(matDat[.,inc]);
        elseif vecCode[inc] == 2;
            vecTemp = First_Diff(matDat[.,inc]);
        elseif vecCode[inc] == 4;
```

```
            vecTemp = Logthm(matDat[.,inc]);
        elseif vecCode[inc] == 5;
            vecTemp = First_Diff_Logthm(matDat[.,inc]);
        elseif vecCode[inc] == 6;
            vecTemp = Sec_Diff_Logthm(matDat[.,inc]);
        endif;
        matX = matX~vecTemp;
    endfor;
    retp(matX);
endp;

/*===================================================================*/

/* Estimates the number of factors in the data */
/*=============================================*/
proc(2) = Factor_Load(matDat,nFacs);
local matFacs,matLoads,matTemp,nTS,nCS,matEigVecs,
        vecEigVals;
nTS = rows(matDat);
nCS = cols(matDat);

if nTS > nCS;
    {vecEigVals,matEigVecs} = eighv(matDat'matDat);
    matEigVecs = matEigVecs';
    matTemp = rev(matEigVecs);
    matTemp = matTemp';
    matLoads = matTemp[.,1:nFacs] .* sqrt(nCS);
    matFacs = (matDat * matLoads) ./ nCS;
    matLoads = matLoads';
else;
    {vecEigVals,matEigVecs} = eighv(matDat * (matDat'));
    matEigVecs = matEigVecs';
    matTemp = rev(matEigVecs);
    matTemp = matTemp';
    matFacs = matTemp[.,1:nFacs] .* sqrt(nTS);
    matLoads = (matFacs' * matDat) ./ nTS;
endif;
retp(matFacs,matLoads);
endp;

proc(1) = Num_Of_Facs(matDat);
local numFacs,matTemp,matErrs,matFacs,matLoads,nTS,nCS,
        vecSumSqErrs,tempVal,PenlWt,inc,crtVal,vecX,cNTSq;
nTS = rows(matDat);
nCS = cols(matDat);
cNTSq = minc(nCS|nTS);

{matFacs,matLoads} = Factor_Load(matDat,1);
matTemp = matFacs * matLoads;
matErrs = matDat - matTemp;
vecSumSqErrs = (diag(matErrs'matErrs)) ./ nTS;
tempVal = (sumc(vecSumSqErrs)) ./ nCS;
PenlWt = ((nCS + nTS) / (nCS * nTS)) * ln(cNTSq);
vecX = ln(tempVal) + PenlWt;

for inc(2,20,1);
    {matFacs,matLoads} = Factor_Load(matDat,inc);
    matTemp = matFacs * matLoads;
    matErrs = matDat - matTemp;
    vecSumSqErrs = (diag(matErrs'matErrs)) ./ nTS;
    tempVal = (sumc(vecSumSqErrs)) ./ nCS;
    PenlWt = inc * (((nCS + nTS) / (nCS * nTS)) * ln(cNTSq));
    crtVal = ln(tempVal) + PenlWt;
    vecX = vecX|crtVal;
endfor;
numFacs = minindc(vecX);
retp(numFacs);
endp;

/*================================================================*/

/* Estimates the variance structures needed to select the proxies */
/*================================================================*/
proc(3) = Pap_Factor_Load(matDat,nFac);
local matFac,matLoad,vecEigVals,matEigVecs,nCS,nTS,matTemp;
nCS = cols(matDat);
nTS = rows(matDat);
{vecEigVals,matEigVecs} = eighv((matDat * (matDat')) ./ (nTS * nCS));
matEigVecs = matEigVecs';
matTemp = rev(matEigVecs);
matTemp = matTemp';
matFac = matTemp[.,1:nFac] .* sqrt(nTS);
matLoad = (matFac' * matDat) ./ nTS;
vecEigVals = rev(vecEigVals);
vecEigVals = vecEigVals[1:nFac];
retp(matFac,matLoad,vecEigVals);
endp;


proc(4) = TStat(matDat,matPxyDat,nFacs,sigLev,indc);
local matResid,matFac,matLoad,matEstPxyVar,nObs,nCS,nTS,temp,
        matSumSqResid,incI,incJ,matTemp,matGamma,matLamda,
        matZeros,matEigVal,sigsq,gammaOLS,nPxyCS,incPxy,
        vecEigVal,matTStat,crtVal,vecMax,vecTest,vecPxyIndex,
        crtValFrq,matBool,vecFrq,vecFrqPxyIndex,vecNSRatioSrt,
        vecNSRatio,vecNSIndex,vecRSqRatio,vecRSqRatioSrt,
        vecRSqIndex,vecFrqSrt,vecPos,matEpsn,incCIj,incCIs,
        matCITemp,vecCIFcT,matAdjS,matCILtBd,matCIRtBd,
        matCIFcS,matCIFcT,vecCIPxyFrq,vecCIFrqSrt,vecCIindex;
nTS = rows(matDat);
nCS = cols(matDat);
nPxyCS = cols(matPxyDat);
temp = sqrt(nTS)|sqrt(nCS);
nObs = round(minc(temp));
crtVal = cdfni((((1 - (sigLev / 2))^(1/nTS)) + 1) / 2);
crtValFrq = cdfni(1 - (sigLev / 2));
vecPos = seqa(1,1,nPxyCS);
{matFac,matLoad,vecEigVal} = Pap_Factor_Load(matDat,nFacs);
matZeros = zeros(nFacs,nFacs);
matEigVal = diagrv(matZeros,vecEigVal);
gammaOLS = inv(matFac'matFac) * (matFac'matPxyDat);
matResid = matDat - (matFac * matLoad);
matSumSqResid = (matResid'matResid) ./ nTS;
matGamma = zeros(nFacs,nFacs);
matCIFcT = {};

matEpsn = (matPxyDat - (matFac * gammaOLS));

if indc == 1;
    for incI(1,nObs,1);
        matLamda = (matLoad[.,incI] * matLoad[.,1]') .* matSumSqResid[incI,1];
        for incJ(2,nObs,1);
            matTemp = (matLoad[.,incI] * matLoad[.,incJ]') .*
matSumSqResid[incI,incJ];
            matLamda = matLamda + matTemp;
        endfor;
        matGamma = matGamma + matLamda;
    endfor;
    matGamma = matGamma ./ nObs;
    matEstPxyVar = (gammaOLS' * inv(matEigVal) * matGamma * inv(matEigVal) *
gammaOLS) ./ nCS;
    matTStat = ((matFac * gammaOLS) - matPxyDat) ./ (sqrt(diag(matEstPxyVar)))';
    vecMax = maxc(abs(matTStat));
    vecTest = vecMax .> crtVal;
    vecPxyIndex = indexcat(vecTest,0);
    matBool = abs(matTStat) .> crtValFrq;
    vecFrq = abs((sumc(matBool) ./ nTS) - sigLev);
    vecFrqSrt = sortc(vecPos~vecFrq,2);
    vecFrqPxyIndex = vecFrqSrt[1:nFacs,1];

elseif indc == 2;
    matEstPxyVar = zeros(nTS,nPxyCS);
    for incJ(1,nTS,1);
        matLamda = zeros(nFacs,nFacs);
        for incI(1,nCS,1);
            matLamda = matLamda + (matLoad[.,incI] * matLoad[.,incI]') .*
(matResid[incJ,incI]^2);
        endfor;
        matGamma = matLamda ./ nCS;
        for incPxy (1,nPxyCS,1);
            matEstPxyVar[incJ,incPxy] = (gammaOLS[.,incPxy]' * inv(matEigVal) *
matGamma * inv(matEigVal) * gammaOLS[.,incPxy]) ./ nCS;
        endfor;
    endfor;
    matTStat = ((matFac * gammaOLS) - matPxyDat) ./ sqrt(matEstPxyVar);
    vecMax = maxc(abs(matTStat));
    vecTest = vecMax .> crtVal;
    vecPxyIndex = indexcat(vecTest,0);
    matBool = abs(matTStat) .> crtValFrq;
    vecFrq = abs((sumc(matBool) ./ nTS) - sigLev);
    vecFrqSrt = sortc(vecPos~vecFrq,2);
    vecFrqPxyIndex = vecFrqSrt[1:nFacs,1];

elseif indc == 3;
    matLoad = matLoad';
    temp = diag(matSumSqResid);
    sigsq = sumc(temp) ./ nCS;
    matGamma = ((matLoad'matLoad) ./ nCS) .* sigsq;
    matEstPxyVar = (gammaOLS' * inv(matEigVal) * matGamma * inv(matEigVal) *
gammaOLS) ./ nCS;
    matTStat = ((matFac * gammaOLS) - matPxyDat) ./ (sqrt(diag(matEstPxyVar)))';
    vecMax = maxc(abs(matTStat));
    vecTest = vecMax .> crtVal;
    vecPxyIndex = indexcat(vecTest,0);
    matBool = abs(matTStat) .> crtValFrq;
    vecFrq = abs((sumc(matBool) ./ nTS) - sigLev);
    vecFrqSrt = sortc(vecPos~vecFrq,2);
    vecFrqPxyIndex = vecFrqSrt[1:nFacs,1];
endif;
retp(vecPxyIndex,vecFrqPxyIndex,matEstPxyVar,matFac);
endp;

/*=============================================================================*/

/* The first procedure computes the SIC and the second generates an AR for a given
lag */
/*==================================================================================
===*/
proc(1) = ARX_SIC3(vecVabl1,hrzn,pMax);
```

```
local nTS,nSmpSz,vecYt1,matDat,incJ,incTemp,vecTemp1,vecOlsY,
        matX,vecOlsPar,vecResid,nARTS,vecSIC,SICcrt,ARLag,
        vecYt2,vecYth,vecTemp2;

vecYth = ln(trimr(vecVabl1,2,0) ./ trimr(vecVabl1,1,1));

vecYt1 = trimr(vecYth,0,hrzn);
nTS = rows(vecYt1);
nSmpSz = nTS - pMax;
vecSIC = {};

for incJ(0,pMax,1);
    matDat = {};
    for incTemp(0,incJ,1);
        vecTemp1 = trimr(vecYt1,incJ-incTemp,incTemp);
        matDat = matDat~vecTemp1;
    endfor;
    vecOlsY = trimr(vecYth,hrzn + incJ,0);
    matDat = vecOlsY~matDat;
    nARTS = rows(matDat);
    if nARTS > nSmpSz;
        matDat = trimr(matDat,nARTS - nSmpSz,0);
    endif;
    vecOlsY = matDat[.,1];
    matX = matDat[.,2:cols(matDat)]~ones(rows(matDat),1);
    vecOlsPar = inv(matX'matX) * matX'vecOlsY;
    vecResid = vecOlsY - (matX * vecOlsPar);
    SICcrt = ln((vecResid'vecResid) / nSmpSz) + ((incJ + 2) * (ln(nSmpSz) /
nSmpSz));
    vecSIC = vecSIC|SICcrt;
endfor;
ARLag = minindc(vecSIC) - 1;
retp(ARLag);
endp;

proc(2) = AR_Spawn3(vecVabl1,hrzn,lagP);
local incTemp, matDat,vecTemp1,vecYth,vecYt1,nTS,vecOlsY,
        futVal;

vecYth = ln(trimr(vecVabl1,2,0) ./ trimr(vecVabl1,1,1));
vecYt1 = trimr(vecYth,0,hrzn);
nTS = rows(vecYth);
futVal = vecYth[(nTS - hrzn + 1)];

matDat = {};
    for incTemp(0,lagP,1);
        vecTemp1 = trimr(vecYt1,lagP-incTemp,incTemp);
        matDat = matDat~vecTemp1;
    endfor;
vecOlsY = trimr(vecYth,hrzn + lagP,0);
matDat = vecOlsY~matDat~ones(rows(matDat),1);
retp(matDat,futVal);
endp;

/*-------------------------------*/

proc(1) = ARX_SIC(vecVabl1,vecVabl2,hrzn,pMax);
local nTS,nSmpSz,vecYt1,matDat,incJ,incTemp,vecTemp1,vecOlsY,
        matX,vecOlsPar,vecResid,nARTS,vecSIC,SICcrt,ARLag,
        vecYt2,vecYth,vecTemp2;

vecYth = ln(trimr(vecVabl1,2,0) ./ trimr(vecVabl1,1,1));
```

```
vecYt1 = trimr(vecYth,0,hrzn);
vecYt2 = trimr(vecVabl2,0,hrzn);
nTS = rows(vecYt1);
nSmpSz = nTS - pMax;
vecSIC = {};

for incJ(0,pMax,1);
    matDat = {};
    for incTemp(0,incJ,1);
        vecTemp1 = trimr(vecYt1,incJ-incTemp,incTemp);
        vecTemp2 = trimr(vecYt2,incJ-incTemp,incTemp);
        matDat = matDat~vecTemp1~vecTemp2;
    endfor;
    vecOlsY = trimr(vecYth,hrzn + incJ,0);
    matDat = vecOlsY~matDat;
    nARTS = rows(matDat);
    if nARTS > nSmpSz;
        matDat = trimr(matDat,nARTS - nSmpSz,0);
    endif;
    vecOlsY = matDat[.,1];
    matX = matDat[.,2:cols(matDat)]~ones(rows(matDat),1);
    vecOlsPar = inv(matX'matX) * matX'vecOlsY;
    vecResid = vecOlsY - (matX * vecOlsPar);
    SICcrt = ln((vecResid'vecResid) / nSmpSz) +
    (((2 * (incJ + 1)) + 1) * (ln(nSmpSz) / nSmpSz));
    vecSIC = vecSIC|SICcrt;
endfor;
ARLag = minindc(vecSIC) - 1;
retp(ARLag);
endp;

proc(3) = AR_Spawn(vecVabl1,vecVabl2,hrzn,lagP);
local incTemp, matDat,vecTemp2,vecYth,vecYt1,nTS,vecOlsY,
        futVal1,futVal2,vecYt2,vecTemp1;

vecYth = ln(trimr(vecVabl1,2,0) ./ trimr(vecVabl1,1,1));
vecYt1 = trimr(vecYth,0,hrzn);
vecYt2 = trimr(vecVabl2,0,hrzn);
nTS = rows(vecVabl2);
futVal1 = vecYth[(rows(vecYth) - hrzn + 1)];
futVal2 = vecVabl2[(nTS - hrzn + 1)];

matDat = {};
    for incTemp(0,lagP,1);
        vecTemp1 = trimr(vecYt1,lagP-incTemp,incTemp);
        vecTemp2 = trimr(vecYt2,lagP-incTemp,incTemp);
        matDat = matDat~vecTemp1~vecTemp2;
    endfor;
vecOlsY = trimr(vecYth,hrzn + lagP,0);
matDat = vecOlsY~matDat~ones(rows(matDat),1);
retp(matDat,futVal1,futVal2);
endp;

/*------------------------------*/

proc(1) = ARX_SIC1(vecVabl1,vecVabl2,hrzn,pMax);
local nTS,nSmpSz,vecYt1,matDat,incJ,incTemp,vecTemp1,vecOlsY,
        matX,vecOlsPar,vecResid,nARTS,vecSIC,SICcrt,ARLag,
        vecYt2,vecYth,vecTemp2;

vecYth = ln(trimr(vecVabl1,2,0) ./ trimr(vecVabl1,1,1));

vecYt1 = trimr(vecYth,0,hrzn);
```

```
nTS = rows(vecYt1);
nSmpSz = nTS - pMax;
vecSIC = {};

for incJ(0,pMax,1);
    matDat = {};
    for incTemp(0,incJ,1);
        vecTemp1 = trimr(vecYt1,incJ-incTemp,incTemp);
        matDat = matDat~vecTemp1;
    endfor;
    vecOlsY = trimr(vecYth,hrzn + incJ,0);
    vecYt2 = trimr(vecVabl2,incJ,hrzn);
    matDat = vecOlsY~matDat~vecYt2;
    nARTS = rows(matDat);
    if nARTS > nSmpSz;
        matDat = trimr(matDat,nARTS - nSmpSz,0);
    endif;
    vecOlsY = matDat[.,1];
    matX = matDat[.,2:cols(matDat)]~ones(rows(matDat),1);
    vecOlsPar = inv(matX'matX) * matX'vecOlsY;
    vecResid = vecOlsY - (matX * vecOlsPar);
    SICcrt = ln((vecResid'vecResid) / nSmpSz) + ((incJ + 3) * (ln(nSmpSz) /
nSmpSz));
    vecSIC = vecSIC|SICcrt;
endfor;
ARLag = minindc(vecSIC) - 1;
retp(ARLag);
endp;

proc(3) = AR_Spawn1(vecVabl1,vecVabl2,hrzn,lagP);
local incTemp, matDat,vecTemp1,vecYth,vecYt1,nTS,vecOlsY,
        futVal1,futVal2,vecYt2;

vecYth = ln(trimr(vecVabl1,2,0) ./ trimr(vecVabl1,1,1));
vecYt1 = trimr(vecYth,0,hrzn);
nTS = rows(vecYth);
futVal1 = vecYth[(nTS - hrzn + 1)];

matDat = {};
    for incTemp(0,lagP,1);
        vecTemp1 = trimr(vecYt1,lagP-incTemp,incTemp);
        matDat = matDat~vecTemp1;
    endfor;
vecOlsY = trimr(vecYth,hrzn + lagP,0);
vecYt2 = trimr(vecVabl2,lagP,hrzn);
matDat = vecOlsY~matDat~vecYt2~ones(rows(matDat),1);
futVal2 = vecVabl2[(rows(vecVabl2) - hrzn + 1)];
retp(matDat,futVal1,futVal2);
endp;

/*--------------------------------*/

proc(1) = ARX_SIC2(vecVabl1,vecVabl2,hrzn,pMax);
local nTS,nSmpSz,vecYt1,matDat,incJ,incTemp,vecTemp1,vecOlsY,
        matX,vecOlsPar,vecResid,nARTS,vecSIC,SICcrt,ARLag,
        vecYt2,vecYth,vecTemp2;

vecYth = ln(trimr(vecVabl1,2,0) ./ trimr(vecVabl1,1,1));
vecYt2 = trimr(vecVabl2,0,hrzn);
nTS = rows(vecYt2);
nSmpSz = nTS - pMax;
vecSIC = {};
```

```
for incJ(0,pMax,1);
    matDat = {};
    for incTemp(0,incJ,1);
        vecTemp2 = trimr(vecYt2,incJ-incTemp,incTemp);
        matDat = matDat~vecTemp2;
    endfor;
    vecOlsY = trimr(vecYth,hrzn + incJ,0);
    vecYt1 = trimr(vecYth,incJ,hrzn);
    matDat = vecOlsY~matDat~vecYt1;
    nARTS = rows(matDat);
    if nARTS > nSmpSz;
        matDat = trimr(matDat,nARTS - nSmpSz,0);
    endif;
    vecOlsY = matDat[.,1];
    matX = matDat[.,2:cols(matDat)]~ones(rows(matDat),1);
    vecOlsPar = inv(matX'matX) * matX'vecOlsY;
    vecResid = vecOlsY - (matX * vecOlsPar);
    SICcrt = ln((vecResid'vecResid) / nSmpSz) + ((incJ + 3) * (ln(nSmpSz) /
nSmpSz));
    vecSIC = vecSIC|SICcrt;
endfor;
ARLag = minindc(vecSIC) - 1;
retp(ARLag);
endp;

proc(3) = AR_Spawn2(vecVabl1,vecVabl2,hrzn,lagP);
local incTemp, matDat,vecTemp2,vecYth,vecYt1,nTS,vecOlsY,
        futVal1,futVal2,vecYt2;

vecYth = ln(trimr(vecVabl1,2,0) ./ trimr(vecVabl1,1,1));
vecYt2 = trimr(vecVabl2,0,hrzn);
nTS = rows(vecVabl2);
futVal2 = vecVabl2[(nTS - hrzn + 1)];

matDat = {};
    for incTemp(0,lagP,1);
        vecTemp2 = trimr(vecYt2,lagP-incTemp,incTemp);
        matDat = matDat~vecTemp2;
    endfor;
vecOlsY = trimr(vecYth,hrzn + lagP,0);
vecYt1 = trimr(vecYth,lagP,hrzn);
matDat = vecOlsY~matDat~vecYt1~ones(rows(matDat),1);
futVal1 = vecYth[(rows(vecYth) - hrzn + 1)];
retp(matDat,futVal1,futVal2);
endp;

/*==============================================================*/

/* The first procedure computes the Newey-West HAC estimator and the second
computes the DM test statistic */
/*======================================================================*/
proc (1) = NwyWst(vecD);
local vecZrMn,varJbar,incI,varShat,vecZrMn1,vecZrMn2;

vecZrMn = vecD - meanc(vecD);
varJbar = int(rows(vecZrMn)^(1/6));
varShat = 0;
for incI(1,varJbar - 1,1);
    vecZrMn1 = trimr(vecZrMn,incI,0);
    vecZrMn2 = trimr(vecZrMn,0,incI);
    varShat = varShat + (((varJbar - incI)/varJbar) * 2 *
(vecZrMn1'vecZrMn2)/(rows(vecZrMn1)));
endfor;
```

```
varShat = varShat + (vecZrMn'vecZrMn)/(rows(vecZrMn));

retp(varShat);
endp;

proc(1) = DMTest(vecLssErs1,vecLssErs2,hrzn);
local DMstat,dbar,stDevD,valD,Pbig;

if hrzn == 1;
    valD = vecLssErs1 - vecLssErs2;
    Pbig = rows(valD);
    dbar = meanc(valD);
    stDevD = stdc(valD);
    DMstat = (sqrt(Pbig) * dbar) / stDevD;
else;
    valD = vecLssErs1 - vecLssErs2;
    Pbig = rows(valD);
    dbar = meanc(valD);
    stDevD = sqrt(NwyWst(valD));
    DMstat = (sqrt(Pbig) * dbar) / stDevD;
endif;

retp(DMstat);
endp;

/*==================================================================================
=====================*/

/* Computes the FSE of the AR, Factor and RW w/Drift Price models */
/*================================================================*/
proc(3) = Factor_AR_FcstRtP_DM(matRawDat,vecRawCodes,vecVabl,cutPt,step);
local matFac,matLoad,matStnStdDat,nTS,nCS,cutVal,matInitDat,
        matParFac,vecYt,vecYth,matX,olsPar,nFac,vecStdVabl,
        cutValH,vecTempT,vecTempT1,vecTempH,nParFacTS,nTempTTS,
        nFacTS,vecDat,FcstVal,cutFacVal,indI,cutActH,cutAct,
        vecActT,vecActT1,vecActVal,vecActPts,lpLimit,vecFcstPts,
        RndWlkMnVal,vecRndWlkPts,RndWlkFcstVal,vecNFac,
        fcstSE,rndWlkSE,vecTempH1,matXAR,vecYthAR,lagAR,
        olsParAR,nXAR,FcstValAR,vecARFcstPts,ARfcstSE,matRpt,
        nxtVal;

nCS = cols(matRawDat);
nTS = rows(matRawDat);
cutValH = round(cutPt * nTS);
cutActH = nTS;
cutAct = cutActH - step;
vecFcstPts = {};
vecRndWlkPts = {};
vecNFac = {};
vecARFcstPts = {};

vecActT = vecVabl[2:cutActH];
vecActT1 = vecVabl[1:(cutActH - 1)];
vecActVal = ln(vecActT ./ vecActT1);
vecActPts = vecActVal[(cutValH):(nTS - 1)];
lpLimit = rows(vecActPts);

for indI(1,lpLimit,1);
cutVal = cutValH - step;
cutFacVal = cutVal + 1;
matInitDat = matRawDat[1:cutValH,.];
matStnStdDat = Stn_Std_Matrix(matInitDat,vecRawCodes);
nFac = Num_Of_Facs(matStnStdDat);
```

```
vecNFac = vecNFac|nFac;
{matFac,matLoad} = Factor_Load(matStnStdDat,nFac);
nFacTS = rows(matFac);
matParFac = trimr(matFac,0,step);
vecStdVabl = vecVabl;
vecTempT1 = vecStdVabl[1:(cutVal - 1)];
vecTempT = vecStdVabl[2:cutVal];
vecTempH = vecStdVabl[(step + 2):cutValH];
vecTempH1 = vecStdVabl[(step + 1):(cutValH - 1)];

nParFacTS = rows(matParFac);
nTempTTS = rows(vecTempT);
if nParFacTS > nTempTTS;
    matParFac = trimr(matParFac,1,0);
elseif nParFacTS < nTempTTS;
    vecTempT1 = trimr(vecTempT1,1,0);
    vecTempT = trimr(vecTempT,1,0);
    vecTempH = trimr(vecTempH,1,0);
    vecTempH1 = trimr(vecTempH1,1,0);
endif;
nTempTTS = rows(vecTempT);
vecYt = ln(vecTempT ./ vecTempT1);
vecYth = ln(vecTempH ./ vecTempH1);
RndWlkMnVal = meanc(vecYth - vecYt);

lagAR = ARX_SIC3(vecVabl[1:cutValH],step,12);
{matXAR,nxtVal} = AR_Spawn3(vecVabl[1:cutValH],step,lagAR);
vecYthAR = matXAR[.,1];
matXAR = matXAR[.,2:cols(matXAR)];
olsParAR = inv(matXAR'matXAR) * matXAR'vecYthAR;
nXAR = rows(matXAR);
if lagAR == 0;
    FcstValAR = (nxtVal~1) * olsParAR;
else;
FcstValAR = (nxtVal~matXAR[nXAR,1:lagAR]~1) * olsParAR;
endif;
vecARFcstPts = vecARFcstPts|FcstValAR;

matX = matParFac~vecYt~ones(nTempTTS,1);
olsPar = inv(matX'matX) * matX'vecYth;
vecDat = matFac[(nFacTS - step + 1),.]~(ln(vecStdVabl[cutFacVal] /
vecStdVabl[cutVal]))~1;
FcstVal = vecDat * olsPar;
vecFcstPts = vecFcstPts|FcstVal;
RndWlkFcstVal = vecDat[(nFac + 1)];
vecRndWlkPts = vecRndWlkPts|RndWlkFcstVal;
cutValH = cutValH + 1;
endfor;

fcstSE = (vecActPts - vecFcstPts)^2;
rndWlkSE = (vecActPts - vecRndWlkPts)^2;
ARfcstSE = (vecActPts - vecARFcstPts)^2;

retp(fcstSE,ARfcstSE,rndWlkSE);
endp;

/*============================================================================*/

/* Computes the FSE for the ordinary proxy forecasts of Price models */
/*=================================================================*/
proc(2) = Forecast_PxyA_DMp(matRawDat,vecRawCodes,vecVabl,cutPt,step,varStr);
local matFac,matLoad,matStnStdDat,nTS,nCS,cutVal,matInitDat,
        matParFac,vecYt,vecYth,matX,olsPar,nFac,vecStdVabl,
        cutValH,vecTempT,vecTempT1,vecTempH,nParFacTS,nTempTTS,
        nFacTS,vecDat,FcstVal,cutFacVal,indI,cutActH,cutAct,
        vecActT,vecActT1,vecActVal,vecActPts,lpLimit,vecFcstPts,
        vecNFac,fcstSE,vecTempH1,matRpt,numOfFactors,MIndex,
        AIndex,NSIndex,estProxyVar,factors,CIIndex,
        matPxyStnStdDat,vecPxyAIndex,vecUnqs,matPxyACnt;

nCS = cols(matRawDat);
nTS = rows(matRawDat);
cutValH = round(cutPt * nTS);
cutActH = nTS;
cutAct = cutActH - step;
vecFcstPts = {};
vecNFac = {};
vecPxyAIndex = {};

vecActT = vecVabl[2:cutActH];
vecActT1 = vecVabl[1:(cutActH - 1)];
vecActVal = ln(vecActT ./ vecActT1);
vecActPts = vecActVal[(cutValH):(nTS - 1)];
lpLimit = rows(vecActPts);

for indI(1,lpLimit,1);
cutVal = cutValH - step;
cutFacVal = cutVal + 1;
matInitDat = matRawDat[1:cutValH,.];
matStnStdDat = Stn_Std_Matrix(matInitDat,vecRawCodes);
matPxyStnStdDat = matStnStdDat[.,1:114 116:132];

numOfFactors = Num_Of_Facs(matStnStdDat);
{MIndex,AIndex,estProxyVar,factors} =
TStat(matStnStdDat,matPxyStnStdDat,numOfFactors,0.05,varStr);
matFac = matPxyStnStdDat[.,AIndex'];
vecPxyAIndex = vecPxyAIndex|AIndex;

nFac = cols(matFac);
nFacTS = rows(matFac);
matParFac = trimr(matFac,0,step);
vecStdVabl = vecVabl;
vecTempT1 = vecStdVabl[1:(cutVal - 1)];
vecTempT = vecStdVabl[2:cutVal];
vecTempH = vecStdVabl[(step + 2):cutValH];
vecTempH1 = vecStdVabl[(step + 1):(cutValH - 1)];

nParFacTS = rows(matParFac);
nTempTTS = rows(vecTempT);
if nParFacTS > nTempTTS;
    matParFac = trimr(matParFac,1,0);
elseif nParFacTS < nTempTTS;
    vecTempT1 = trimr(vecTempT1,1,0);
    vecTempT = trimr(vecTempT,1,0);
    vecTempH = trimr(vecTempH,1,0);
    vecTempH1 = trimr(vecTempH1,1,0);
endif;
nTempTTS = rows(vecTempT);
vecYt = ln(vecTempT ./ vecTempT1);
vecYth = ln(vecTempH ./ vecTempH1);

matX = matParFac~vecYt~ones(nTempTTS,1);
olsPar = inv(matX'matX) * matX'vecYth;
vecDat = matFac[(nFacTS - step + 1),.]~(ln(vecStdVabl[cutFacVal] /
vecStdVabl[cutVal]))~1;
FcstVal = vecDat * olsPar;
```

```
vecFcstPts = vecFcstPts|FcstVal;
cutValH = cutValH + 1;
endfor;

vecUnqs = unique(vecPxyAIndex,1);
matPxyACnt = rev(sortc(vecUnqs~(counts(vecPxyAIndex,vecUnqs) ./ lpLimit),2));
matPxyACnt = matPxyACnt[1:13,.];

fcstSE = (vecActPts - vecFcstPts)^2;

retp(fcstSE,matPxyACnt);
endp;

/*------------------------------*/

proc(2) = Forecast_PxyM_DMp(matRawDat,vecRawCodes,vecVabl,cutPt,step,varStr);
local matFac,matLoad,matStnStdDat,nTS,nCS,cutVal,matInitDat,
        matParFac,vecYt,vecYth,matX,olsPar,nFac,vecStdVabl,
        cutValH,vecTempT,vecTempT1,vecTempH,nParFacTS,nTempTTS,
        nFacTS,vecDat,FcstVal,cutFacVal,indI,cutActH,cutAct,
        vecActT,vecActT1,vecActVal,vecActPts,lpLimit,vecFcstPts,
        vecNFac,fcstSE,vecTempH1,matRpt,numOfFactors,MIndex,
        AIndex,NSIndex,estProxyVar,factors,CIIndex,misFlg,
        matPxyStnStdDat,vecPxyMIndex,vecUnqs,matPxyMCnt;

nCS = cols(matRawDat);
nTS = rows(matRawDat);
cutValH = round(cutPt * nTS);
cutActH = nTS;
cutAct = cutActH - step;
vecFcstPts = {};
vecNFac = {};
vecPxyMIndex = {};

vecActT = vecVabl[2:cutActH];
vecActT1 = vecVabl[1:(cutActH - 1)];
vecActVal = ln(vecActT ./ vecActT1);
vecActPts = vecActVal[(cutValH):(nTS - 1)];
lpLimit = rows(vecActPts);

for indI(1,lpLimit,1);
cutVal = cutValH - step;
cutFacVal = cutVal + 1;
matInitDat = matRawDat[1:cutValH,.];
matStnStdDat = Stn_Std_Matrix(matInitDat,vecRawCodes);
matPxyStnStdDat = matStnStdDat[.,1:114 116:132];

numOfFactors = Num_Of_Facs(matStnStdDat);
{MIndex,AIndex,estProxyVar,factors} =
TStat(matStnStdDat,matPxyStnStdDat,numOfFactors,0.05,varStr);
matFac = matPxyStnStdDat[.,MIndex'];
vecPxyMIndex = vecPxyMIndex|MIndex;
misFlg = ismiss(MIndex);

nFac = cols(matFac);
nFacTS = rows(matFac);
matParFac = trimr(matFac,0,step);
vecStdVabl = vecVabl;
vecTempT1 = vecStdVabl[1:(cutVal - 1)];
vecTempT = vecStdVabl[2:cutVal];
vecTempH = vecStdVabl[(step + 2):cutValH];
vecTempH1 = vecStdVabl[(step + 1):(cutValH - 1)];
```

```
nParFacTS = rows(matParFac);
nTempTTS = rows(vecTempT);
if nParFacTS > nTempTTS;
    matParFac = trimr(matParFac,1,0);
elseif nParFacTS < nTempTTS;
    vecTempT1 = trimr(vecTempT1,1,0);
    vecTempT = trimr(vecTempT,1,0);
    vecTempH = trimr(vecTempH,1,0);
    vecTempH1 = trimr(vecTempH1,1,0);
endif;
nTempTTS = rows(vecTempT);
vecYt = ln(vecTempT ./ vecTempT1);
vecYth = ln(vecTempH ./ vecTempH1);

if misFlg == 1;
    matX = vecYt~ones(nTempTTS,1);
    olsPar = inv(matX'matX) * matX'vecYth;
    vecDat = (ln(vecStdVabl[cutFacVal] / vecStdVabl[cutVal]))~1;
else;
    matX = matParFac~vecYt~ones(nTempTTS,1);
    olsPar = inv(matX'matX) * matX'vecYth;
    vecDat = matFac[(nFacTS - step + 1),.]~(ln(vecStdVabl[cutFacVal] /
vecStdVabl[cutVal]))~1;
endif;

FcstVal = vecDat * olsPar;
vecFcstPts = vecFcstPts|FcstVal;
cutValH = cutValH + 1;
endfor;

vecUnqs = unique(vecPxyMIndex,1);
matPxyMCnt = rev(sortc(vecUnqs~(counts(vecPxyMIndex,vecUnqs) ./ lpLimit),2));

fcstSE = (vecActPts - vecFcstPts)^2;

retp(fcstSE,matPxyMCnt);
endp;

/*-----------------------*/


/*=================================================================================
*/

/* Computes the FSE for the ex-ante proxy forecasts of Price models */
/*==================================================================*/
proc(4) = Forecast_PxyExAnA_DMp(matRawDat,vecRawCodes,vecVabl,cutPt,step,varStr);
local matFac,matLoad,matStnStdDat,nTS,nCS,cutVal,matInitDat,
        matParFac,vecYt,vecYth,matX,olsPar,nFac,vecStdVabl,
        cutValH,vecTempT,vecTempT1,vecTempH,nParFacTS,nTempTTS,
        nFacTS,vecDat,FcstVal,cutFacVal,indI,cutActH,cutAct,
        vecActT,vecActT1,vecActVal,vecActPts,lpLimit,vecFcstPts,
        numOfFactors,MIndex,AIndex,NSIndex,estProxyVar,factors,
        fcstSE,vecAIndex,vecUnion,vecB,vecM,vecFreq,
        vecExAntePxy,vecBool,vecTtlExAnPxy,nExAnPxy,incExAn,
        vecFcstMSE,vecPxyFrq,matExAnTemp,matExPxyFrq,
        CIIndex,vecTempH1,lagAR,matXAR,nxtVal1,nxtVal2,vecYthAR,
        olsParAR,nXAR,FcstValAR,vecARFcstPts,FcstValAR1,
        vecARFcstPts1,FcstValAR2,vecARFcstPts2,fcstARSE,
        fcstARSE1,fcstARSE2,matPxyStnStdDat,matRawExDat,
        vecRawExCodes;

nCS = cols(matRawDat);
```

```
nTS = rows(matRawDat);
cutValH = round(cutPt * nTS);
lpLimit = round((nTS - cutValH) / 2);
cutActH = nTS;
cutAct = cutActH - step;
vecAIndex = {};
vecFcstMSE = {};
vecARFcstPts = {};
vecARFcstPts1 = {};
vecARFcstPts2 = {};

vecActT = vecVabl[2:cutActH];
vecActT1 = vecVabl[1:(cutActH - 1)];
vecActVal = ln(vecActT ./ vecActT1);
vecActPts = vecActVal[(2 * cutValH):(nTS - 1)];

for indI(1,lpLimit,1);
cutVal = cutValH - step;
cutFacVal = cutVal + 1;
matInitDat = matRawDat[1:cutValH,.];
matStnStdDat = Stn_Std_Matrix(matInitDat,vecRawCodes);
matPxyStnStdDat = matStnStdDat[.,1:114 116:132];

numOfFactors = Num_Of_Facs(matStnStdDat);
{MIndex,AIndex,estProxyVar,factors} =
TStat(matStnStdDat,matPxyStnStdDat,numOfFactors,0.05,varStr);
vecAIndex = vecAIndex|AIndex;
cutValH = cutValH + 1;
endfor;

vecAIndex = packr(vecAIndex);
vecUnion = unique(vecAIndex,1);
vecExAntePxy = vecUnion[maxindc(counts(vecAIndex,vecUnion))];

cutValH = round(2 * cutPt * nTS);
vecFcstPts = {};
matRawExDat = matRawDat[.,1:114 116:132];
vecRawExCodes = vecRawCodes[1:114 116:132];

for indI(1,lpLimit,1);
cutVal = cutValH - step;
cutFacVal = cutVal + 1;
matInitDat = matRawExDat[1:cutValH,vecExAntePxy];
matStnStdDat = Stn_Std_Matrix(matInitDat,vecRawExCodes[vecExAntePxy]);
matParFac = trimr(matStnStdDat,0,step);
vecStdVabl = vecVabl;
vecTempT1 = vecStdVabl[1:(cutVal - 1)];
vecTempT = vecStdVabl[2:cutVal];
vecTempH = vecStdVabl[(step + 2):cutValH];
vecTempH1 = vecStdVabl[(step + 1):(cutValH - 1)];

nParFacTS = rows(matParFac);
nTempTTS = rows(vecTempT);
if nParFacTS > nTempTTS;
    matParFac = trimr(matParFac,1,0);
elseif nParFacTS < nTempTTS;
    vecTempT1 = trimr(vecTempT1,1,0);
    vecTempT = trimr(vecTempT,1,0);
    vecTempH = trimr(vecTempH,1,0);
    vecTempH1 = trimr(vecTempH1,1,0);
endif;
nTempTTS = rows(vecTempT);
vecYt = ln(vecTempT ./ vecTempT1);
```

```
vecYth = ln(vecTempH ./ vecTempH1);

matX = matParFac~vecYt~ones(nTempTTS,1);
olsPar = inv(matX'matX) * matX'vecYth;
vecDat = matStnStdDat[(rows(matStnStdDat) - step + 1)]~(ln(vecStdVabl[cutFacVal] /
vecStdVabl[cutVal]))~1;
FcstVal = vecDat * olsPar;
vecFcstPts = vecFcstPts|FcstVal;

lagAR = ARX_SIC(vecVabl[1:cutValH],matStnStdDat,step,12);
{matXAR,nxtVal1,nxtVal2} = AR_Spawn(vecVabl[1:cutValH],matStnStdDat,step,lagAR);
vecYthAR = matXAR[.,1];
matXAR = matXAR[.,2:cols(matXAR)];
olsParAR = inv(matXAR'matXAR) * matXAR'vecYthAR;
nXAR = rows(matXAR);
if cols(matXAR) == 3;
    FcstValAR = (nxtVal1~nxtVal2~1) * olsParAR;
else;
    FcstValAR = (nxtVal1~nxtVal2~matXAR[nXAR,1:(cols(matXAR) - 3)]~1) * olsParAR;
endif;
vecARFcstPts = vecARFcstPts|FcstValAR;

lagAR = ARX_SIC1(vecVabl[1:cutValH],matStnStdDat,step,12);
{matXAR,nxtVal1,nxtVal2} = AR_Spawn1(vecVabl[1:cutValH],matStnStdDat,step,lagAR);
vecYthAR = matXAR[.,1];
matXAR = matXAR[.,2:cols(matXAR)];
olsParAR = inv(matXAR'matXAR) * matXAR'vecYthAR;
nXAR = rows(matXAR);
if cols(matXAR) == 3;
    FcstValAR1 = (nxtVal1~nxtVal2~1) * olsParAR;
else;
    FcstValAR1 = (nxtVal1~matXAR[nXAR,1:(cols(matXAR) - 3)]~nxtVal2~1) * olsParAR;
endif;
vecARFcstPts1 = vecARFcstPts1|FcstValAR1;

lagAR = ARX_SIC2(vecVabl[1:cutValH],matStnStdDat,step,12);
{matXAR,nxtVal1,nxtVal2} = AR_Spawn2(vecVabl[1:cutValH],matStnStdDat,step,lagAR);
vecYthAR = matXAR[.,1];
matXAR = matXAR[.,2:cols(matXAR)];
olsParAR = inv(matXAR'matXAR) * matXAR'vecYthAR;
nXAR = rows(matXAR);
if cols(matXAR) == 3;
    FcstValAR2 = (nxtVal2~nxtVal1~1) * olsParAR;
else;
    FcstValAR2 = (nxtVal2~matXAR[nXAR,1:(cols(matXAR) - 3)]~nxtVal1~1) * olsParAR;
endif;
vecARFcstPts2 = vecARFcstPts2|FcstValAR2;

cutValH = cutValH + 1;
endfor;

fcstSE = (vecActPts - vecFcstPts)^2;
fcstARSE = (vecActPts - vecARFcstPts)^2;
fcstARSE1 = (vecActPts - vecARFcstPts1)^2;
fcstARSE2 = (vecActPts - vecARFcstPts2)^2;

retp(fcstSE,fcstARSE,fcstARSE1,fcstARSE2);
endp;

/*--------------------------*/

proc(4) = Forecast_PxyExAnM_DMp(matRawDat,vecRawCodes,vecVabl,cutPt,step,varStr);
local matFac,matLoad,matStnStdDat,nTS,nCS,cutVal,matInitDat,
        matParFac,vecYt,vecYth,matX,olsPar,nFac,vecStdVabl,
        cutValH,vecTempT,vecTempT1,vecTempH,nParFacTS,nTempTTS,
        nFacTS,vecDat,FcstVal,cutFacVal,indI,cutActH,cutAct,
        vecActT,vecActT1,vecActVal,vecActPts,lpLimit,vecFcstPts,
        numOfFactors,MIndex,AIndex,NSIndex,estProxyVar,factors,
        fcstSE,vecMIndex,vecUnion,vecB,vecM,vecFreq,
        vecExAntePxy,vecBool,vecTtlExAnPxy,nExAnPxy,incExAn,
        vecFcstMSE,vecPxyFrq,matExAnTemp,matExPxyFrq,
        CIIndex,vecTempH1,lagAR,matXAR,nxtVal1,nxtVal2,vecYthAR,
        olsParAR,nXAR,FcstValAR,vecARFcstPts,FcstValAR1,
        vecARFcstPts1,FcstValAR2,vecARFcstPts2,fcstARSE,
        fcstARSE1,fcstARSE2,matPxyStnStdDat,matRawExDat,
        vecRawExCodes;

nCS = cols(matRawDat);
nTS = rows(matRawDat);
cutValH = round(cutPt * nTS);
lpLimit = round((nTS - cutValH) / 2);
cutActH = nTS;
cutAct = cutActH - step;
vecMIndex = {};
vecFcstMSE = {};
vecARFcstPts = {};
vecARFcstPts1 = {};
vecARFcstPts2 = {};

vecActT = vecVabl[2:cutActH];
vecActT1 = vecVabl[1:(cutActH - 1)];
vecActVal = ln(vecActT ./ vecActT1);
vecActPts = vecActVal[(2 * cutValH):(nTS - 1)];

for indI(1,lpLimit,1);
cutVal = cutValH - step;
cutFacVal = cutVal + 1;
matInitDat = matRawDat[1:cutValH,.];
matStnStdDat = Stn_Std_Matrix(matInitDat,vecRawCodes);
matPxyStnStdDat = matStnStdDat[.,1:114 116:132];

numOfFactors = Num_Of_Facs(matStnStdDat);
{MIndex,AIndex,estProxyVar,factors} =
TStat(matStnStdDat,matPxyStnStdDat,numOfFactors,0.05,varStr);
vecMIndex = vecMIndex|MIndex;
cutValH = cutValH + 1;
endfor;

vecMIndex = packr(vecMIndex);
vecUnion = unique(vecMIndex,1);
vecExAntePxy = vecUnion[maxindc(counts(vecMIndex,vecUnion))];

cutValH = round(2 * cutPt * nTS);
vecFcstPts = {};
matRawExDat = matRawDat[.,1:114 116:132];
vecRawExCodes = vecRawCodes[1:114 116:132];

for indI(1,lpLimit,1);
cutVal = cutValH - step;
cutFacVal = cutVal + 1;
matInitDat = matRawExDat[1:cutValH,vecExAntePxy];
matStnStdDat = Stn_Std_Matrix(matInitDat,vecRawExCodes[vecExAntePxy]);
matParFac = trimr(matStnStdDat,0,step);
vecStdVabl = vecVabl;
vecTempT1 = vecStdVabl[1:(cutVal - 1)];
vecTempT = vecStdVabl[2:cutVal];
```

```
vecTempH = vecStdVabl[(step + 2):cutValH];
vecTempH1 = vecStdVabl[(step + 1):(cutValH - 1)];

nParFacTS = rows(matParFac);
nTempTTS = rows(vecTempT);
if nParFacTS > nTempTTS;
    matParFac = trimr(matParFac,1,0);
elseif nParFacTS < nTempTTS;
    vecTempT1 = trimr(vecTempT1,1,0);
    vecTempT = trimr(vecTempT,1,0);
    vecTempH = trimr(vecTempH,1,0);
    vecTempH1 = trimr(vecTempH1,1,0);
endif;
nTempTTS = rows(vecTempT);
vecYt = ln(vecTempT ./ vecTempT1);
vecYth = ln(vecTempH ./ vecTempH1);

matX = matParFac~vecYt~ones(nTempTTS,1);
olsPar = inv(matX'matX) * matX'vecYth;
vecDat = matStnStdDat[(rows(matStnStdDat) - step + 1)]~(ln(vecStdVabl[cutFacVal] /
vecStdVabl[cutVal]))~1;
FcstVal = vecDat * olsPar;
vecFcstPts = vecFcstPts|FcstVal;

lagAR = ARX_SIC(vecVabl[1:cutValH],matStnStdDat,step,12);
{matXAR,nxtVal1,nxtVal2} = AR_Spawn(vecVabl[1:cutValH],matStnStdDat,step,lagAR);
vecYthAR = matXAR[.,1];
matXAR = matXAR[.,2:cols(matXAR)];
olsParAR = inv(matXAR'matXAR) * matXAR'vecYthAR;
nXAR = rows(matXAR);
if cols(matXAR) == 3;
    FcstValAR = (nxtVal1~nxtVal2~1) * olsParAR;
else;
    FcstValAR = (nxtVal1~nxtVal2~matXAR[nXAR,1:(cols(matXAR) - 3)]~1) * olsParAR;
endif;
vecARFcstPts = vecARFcstPts|FcstValAR;

lagAR = ARX_SIC1(vecVabl[1:cutValH],matStnStdDat,step,12);
{matXAR,nxtVal1,nxtVal2} = AR_Spawn1(vecVabl[1:cutValH],matStnStdDat,step,lagAR);
vecYthAR = matXAR[.,1];
matXAR = matXAR[.,2:cols(matXAR)];
olsParAR = inv(matXAR'matXAR) * matXAR'vecYthAR;
nXAR = rows(matXAR);
if cols(matXAR) == 3;
    FcstValAR1 = (nxtVal1~nxtVal2~1) * olsParAR;
else;
    FcstValAR1 = (nxtVal1~matXAR[nXAR,1:(cols(matXAR) - 3)]~nxtVal2~1) * olsParAR;
endif;
vecARFcstPts1 = vecARFcstPts1|FcstValAR1;

lagAR = ARX_SIC2(vecVabl[1:cutValH],matStnStdDat,step,12);
{matXAR,nxtVal1,nxtVal2} = AR_Spawn2(vecVabl[1:cutValH],matStnStdDat,step,lagAR);
vecYthAR = matXAR[.,1];
matXAR = matXAR[.,2:cols(matXAR)];
olsParAR = inv(matXAR'matXAR) * matXAR'vecYthAR;
nXAR = rows(matXAR);
if cols(matXAR) == 3;
    FcstValAR2 = (nxtVal2~nxtVal1~1) * olsParAR;
else;
    FcstValAR2 = (nxtVal2~matXAR[nXAR,1:(cols(matXAR) - 3)]~nxtVal1~1) * olsParAR;
endif;
vecARFcstPts2 = vecARFcstPts2|FcstValAR2;
```

```
cutValH = cutValH + 1;
endfor;

fcstSE = (vecActPts - vecFcstPts)^2;
fcstARSE = (vecActPts - vecARFcstPts)^2;
fcstARSE1 = (vecActPts - vecARFcstPts1)^2;
fcstARSE2 = (vecActPts - vecARFcstPts2)^2;

retp(fcstSE,fcstARSE,fcstARSE1,fcstARSE2);
endp;

/*---------------------*/



/*==============================================================================*/
```