Appendix

of

Imposing Curvature and Monotonicity on Flexible Functional Forms

H. Wolff, T. Heckelei and R.C. Mittelhammer

Computational Economics, 2010

Appendix also available online at http://hendrikwolff.com/research.html

Contents

Section 1: File Readme.txt

This set of files are associated with the article:

Imposing Curvature and Monotonicity on Flexible Functional Forms,

by H. Wolff, T. Heckelei and R.C. Mittelhammer

in *Computational Economics*

It is provided to help the practitioners to use the code as a template for their own applications.

The main code is written in GAUSS and the central file is “Monotonicity and Curvature.gss”.

In particular we wrote the code such that the interested reader can replicate our simulations performed in the paper.

Each simulation undertaken in “Monotonicity and Curvature.gss” automatically creates an output file named as “outputfile.out” such that the user can verify what the settings and the parameters of the simulation have been (i.g.

- number of draws within the Metropolis Hastings procedure,

- automatic calculation of the matrix of “Elasticity of Substitution” of the underlying true data generation process

- estimated “Elasticity of Substitution” matrix

- other parameters of interest of central features to the simulation.)

Note also the procedure files ending in “.g”. These files are needed in GAUSS in order to run the main file.

Section 2: Program Monotonicy and Curvature.gss

```
/*****************************************************************************
*****************

 This GAUSS program illustrates the estimation procedures in the article:


"Imposing Curvature and Monotonicity on Flexible Functional Forms" Computational Economics, by H.
Wolff, T. Heckelei and R.C. Mittelhammer.


*****************************************************************************
*****************/


new;

output file = outputfile.out reset;

datestart = date;

@ new x,y : clear global symbols, set maximum number of global symbols to x

        and allocate y bytes for main program code. @

cls;

print /flush "-----------------------------------------------------------------------------------------------------------------------
-------------------


  Estimation of AIM, DGP: CES and TL Technology


  Notation and examples from: Terrell 1995, Flexibility and

  Regularity Properties of the Asymptotically Ideal Production Model


  Programmed: by Hendrik Wolff
```

```
------------------------------------------------------------------------------------------------------------------------------------------
---";
/*****************************************************************************************
*****************
                    CHOOSE
              - DGP (CES or TL TECHNOLOGY)
              - PARAMETERS OF THE CES-TECHNOLOGY
              - FLEXIBLE FUNCTIONAL FORM
              - ORDER OF THE AIM(Order)
              - METHOD OF ESTIMATION
*****************************************************************************************
*****************/
 Sim         = 1;     @ if DGP = CES: Simulation Set 1 to 3 - see Tables in Terrell '95  @
 FFF         = "AIM";  @ Flexible Functional Form - see below                      @
 Order        = 2;     @ Order of Expansion                                    @
 concavity     = 0;     @ Imposes global concavity restrictions (for AIM)              @
 co_constrained = 0;     @ 1 --> uses CO Procedure                              @
 MCMC         = 2;     @ 0 --> without Resampling, 1 --> GIBBS, 2 --> METROPOLIS-HASTINGS @
 fcmptol_large  = 10e-15; @ for fuzzy comparisons                               @
 fcmptol_small  = 10e-5;  @ for fuzzy comparisons                               @
 breakn       = 10e-6;  @ To break Zellners SUR iterations                         @
 DGP          = "CES";  @ Data Generation Process: "CES", "TL"                      @
/*****************************************************************************************
*****************
 FFF    AIM  AIM Input Demand System
        TLS  Translog Share system: s1|s2
```

```
      TLC  Translog Cost Function & Share system: c|s1|s2

 Order: 1  --> AIM(1) (Generalized Leontief)

      2  --> AIM(2)

      3  --> AIM(3)
*************************************************************************************
*****************/

library co;

#include co.ext;

coset;

format 10,5;


/*************************************************************************************
*****************

*************************************************************************************
*****************

                    DATA GENERATION PROCESS

*************************************************************************************
*****************

*************************************************************************************
*****************/


/*************************************************************************************
*****************

                    Construction of Price Vectors w1, w2, w3

*************************************************************************************
*****************/

    k = 4;              /* k:    Number of Elements (Terrell: 4 prices in [0.5,1.5]       */

    m = 3;              /* m:    Number of Variables (Terrell: three inputs)             */
```

```
n = k^m;              /* n:    Number of Observation = number of possible combinations
                         (hyper-rightangle)                          */
init = seqa(0.5,1/3,k);  /* init:  Price Outcomes                    */
index = zeros(n,m);     /* index: combinations represented in the most general form: integers */
/* Here is the procedure to define the index (idea provided by Thomas Heckelei)          */
/*************************************************************************/
for i (1,n,1);
   for j (1,m,1);
      index[i,j] = trunc((i+k^j-1)/k^(j-1)) % k +1;
   endfor;
endfor;
/*************************************************************************/
w  = init[index];        /* maps init to w */
w  = reshape(w,n,m);
w1 = w[.,1]; w2 = w[.,2]; w3 = w[.,3];
clear index;
/* AES evaluated at input prices = 1 and output = 1 */
AESw1 = 1; AESw2 = 1; AESw3 = 1; AESw = AESw1~AESw2~AESw3;
AESy  = 1;


/*************************************************************************************
****************

                        DATA GENERATION OF CES


*************************************************************************************
****************/
```

```
if DGP $== "CES";
    y = ones(n,1);          @ output y @
    {a1,a2,a3,rho} = pset(sim); @ Procedure SET sets the parameter values of the CES function @

    @ CES Cost Function @
    c = y.*((a1^(1/(1-rho)))*(w1^(-rho/(1-rho)))
        + (a2^(1/(1-rho)))*(w2^(-rho/(1-rho)))
        + (a3^(1/(1-rho)))*(w3^(-rho/(1-rho))))^((1-rho)/(-rho));

    @ CES Input Demand Functions @
    x1 = (c^(1/(1-rho))).*(y^(-rho/(1-rho))).*(w1^(-1/(1-rho)))*(a1^(1/(1-rho)));
    x2 = (c^(1/(1-rho))).*(y^(-rho/(1-rho))).*(w2^(-1/(1-rho)))*(a2^(1/(1-rho)));
    x3 = (c^(1/(1-rho))).*(y^(-rho/(1-rho))).*(w3^(-1/(1-rho)))*(a3^(1/(1-rho)));

    @ control if y(x) = y @
    y  = (a1*(x1^rho) + a2*(x2^rho) + a3*(x3^rho))^(1/rho);

    x  =  x1~x2~x3;

    print /flush "True Allen Elasticities of Substitution of the CES";
    print /flush "AES12=AES13=AES23: " 1/(1-rho);
endif;

/*************************************************************************************
*****************
```

DATA GENERATION OF TRANSLOG

```
**********************************************************************************
****************/
if DGP $== "TL";


    d1 = .5; d2 = .25; d3 = .25;

    gamm = {-.2  .1   .1,

            .1 -.05 -.05,

            .1 -.05 -.05};

    alpha = zeros(10,1);

    alpha[2] = d1; alpha[3] = d2; alpha[4] = d3;

    alpha[5] = gamm[1,1]; alpha[6] = gamm[1,2]; alpha[7] = gamm[1,3];

                    alpha[8] = gamm[2,2]; alpha[9] = gamm[2,3];

                                    alpha[10]= gamm[3,3];


    zc = ones(n,1)~ln(w)        @ Regressors of the ln-cost function          @

       ~.5*ln(w1).*ln(w1) ~  ln(w1).*ln(w2)  ~   ln(w1).*ln(w3)

                     ~.5*ln(w2).*ln(w2) ~   ln(w2).*ln(w3)

                                    ~.5*ln(w3).*ln(w3);


    x1 = (alpha[2]/w1 + alpha[5]*ln(w1)./w1 + alpha[6]/w1.*ln(w2) + alpha[7]/w1.*ln(w3 )).*

         exp(zc*alpha);


    x2 = (alpha[3]/w2 + alpha[6]*ln(w1)./w2 + alpha[8]*ln(w2)./w2 + alpha[9]/w2.*ln(w3))

         .*exp(zc*alpha);
```

```
x3 = (alpha[4]/w3 +alpha[7]*ln(w1)./w3 + alpha[9]*ln(w2)./w3 + alpha[10]*ln(w3)./w3)
    .*exp(zc*alpha);


x = x1~x2~x3;

y = ones(n,1);

c = exp(zc*alpha);


/*****************************************************************************************************
                Calculation of the "true" elasticities of substitution
******************************************************************************************************/

@  The following checks the (partially incorrectly) Allen Elasticities given in Terrell 1995,p. 15 @

    /* Regressors of the ln-cost function*/

    AESzc = 1~ln(AESw1)~ln(AESw2)~ln(AESw3)

        ~.5*ln(AESw1).*ln(AESw1) ~  ln(AESw1).*ln(AESw2)  ~   ln(AESw1).*ln(AESw3)

                    ~.5*ln(AESw2).*ln(AESw2) ~   ln(AESw2).*ln(AESw3)

                                ~.5*ln(AESw3).*ln(AESw3);

    AESx1 = (alpha[2]/AESw1 + alpha[5]*ln(AESw1)./AESw1 + alpha[6]/AESw1.*ln(AESw2) +
alpha[7]/AESw1.*ln(AESw3 ))

        .*exp(AESzc*alpha);

    AESx2 = (alpha[3]/AESw2 + alpha[6]*ln(AESw1)./AESw2 + alpha[8]*ln(AESw2)./AESw2 +
alpha[9]/AESw2.*ln(AESw3))

        .*exp(AESzc*alpha);

    AESx3 = (alpha[4]/AESw3 +alpha[7]*ln(AESw1)./AESw3 + alpha[9]*ln(AESw2)./AESw3 +
alpha[10]*ln(AESw3)./AESw3)
```

```
        .*exp(AESzc*alpha);
    AESx = AESx1~AESx2~AESx3;
    y = ones(n,1);
    AESc = exp(AESzc*alpha);
    test = AESx1.*AESw1+AESx2.*AESw2+AESx3.*AESw3 - AESc;
    m1   = AESx1.*AESw1./AESc;  /*m : shares */
    m2   = AESx2.*AESw2./AESc;
    m3   = AESx3.*AESw3./AESc;
    mm   = m1~m2~m3;
    /* formula from Guilkby / Lovell, 1980, International Economic Review,
      "On the Flexibility of the Translog Approximation" */
    AES     = (gamm + mm'mm)./(mm'mm);      @ offdiagonal elements @
    AESdiag = (gamm + mm.*mm-mm)./(mm.*mm); @ diagonal elements @
    AES[1,1]= AESdiag[1,1];
    AES[2,2]= AESdiag[2,2];
    AES[3,3]= AESdiag[3,3];
    print /flush "True Allen Elasticities of Substitution of the TL: " AES;

endif;

/*************************************************************************
****************

        Construction of the Grid for Evaluation at the Price Space Psi (8000 Price Vectors)

*************************************************************************
*****************/

  f = 20;     @ f: Number of Prices for each input @
```

```
ngrid = f^m; @ n: Number of possible combinations (rightangle) @
initgrid = seqa(.5,1/19,20);
index = zeros(ngrid,3);
for i (1,ngrid,1);
   for j (1,m,1);
      index[i,j] = trunc((i+f^j-1)/f^(j-1)) % f +1;
   endfor;
endfor;
@ Psi is the hyperrightangle (for three = cube): region of the price space@
Psi = initgrid[index];
clear index;
Psi = reshape(Psi,ngrid,m);
/* Attention: the following helps in the case that f=20, m=3
  It should help speeding up the accept-reject algorithm,
  because we start checking first the corners of Psi */
Psi_interm    = Psi[2:8,.];
Psi[2,.]      = Psi[20,.];
Psi[3,.]      = Psi[381,.];
Psi[4,.]      = Psi[400,.];
Psi[5,.]      = Psi[7601,.];
Psi[6,.]      = Psi[7620,.];
Psi[7,.]      = Psi[7981,.];
Psi[8,.]      = Psi[8000,.];
Psi[20,.]     = Psi_interm[1,.];
Psi[381,.]    = Psi_interm[2,.];
```

```
    Psi[400,.]    = Psi_interm[3,.];
    Psi[7601,.]   = Psi_interm[4,.];
    Psi[7620,.]   = Psi_interm[5,.];
    Psi[7981,.]   = Psi_interm[6,.];
    Psi[8000,.]   = Psi_interm[7,.];
    clear Psi_interm;
    psi1 = psi[.,1];
    psi2 = psi[.,2];
    psi3 = psi[.,3];

/*************************************************************************************
*****************
                Calculate Input Quantities (x) at the Grid for the CES
*************************************************************************************
*****************/
if DGP $== "CES";
    ygrid = ones(ngrid,1);

    cgrid = ygrid.*((a1^(1/(1-rho)))*(psi1^(-rho/(1-rho)))
        + (a2^(1/(1-rho)))*(psi2^(-rho/(1-rho)))
        + (a3^(1/(1-rho)))*(psi3^(-rho/(1-rho))))^((1-rho)/(-rho));

    x1grid = (Cgrid^(1/(1-rho))).*(ygrid^(-rho/(1-rho))).*(psi1^(-1/(1-rho)))*(a1^(1/(1-rho)));
    x2grid = (Cgrid^(1/(1-rho))).*(ygrid^(-rho/(1-rho))).*(psi2^(-1/(1-rho)))*(a2^(1/(1-rho)));
    x3grid = (Cgrid^(1/(1-rho))).*(ygrid^(-rho/(1-rho))).*(psi3^(-1/(1-rho)))*(a3^(1/(1-rho)));
```

```
    xgrid = x1grid~x2grid~x3grid;


    Mean = meanc(xgrid);

    Std  = stdc(xgrid);

endif;


/*********************************************************************************
*****************

                    Calculate Input Quantities (x) at the Grid for the TRANSLOG

**********************************************************************************
*****************/

if DGP $== "TL";

    ygrid = ones(ngrid,1);


    zcgrid = ones(ngrid,1)~ln(psi)       @ Regressors of the ln-cost function          @

           ~.5*ln(psi1).*ln(psi1) ~  ln(psi1).*ln(psi2)  ~   ln(psi1).*ln(psi3)

                           ~.5*ln(psi2).*ln(psi2) ~   ln(psi2).*ln(psi3)

                                           ~.5*ln(psi3).*ln(psi3);

    x1grid = (alpha[2]/psi1 + alpha[5]*ln(psi1)./psi1 + alpha[6]/psi1.*ln(psi2) + alpha[7]/psi1.*ln(psi3 ))

           .*exp(zcgrid*alpha);


    x2grid = (alpha[3]/psi2 + alpha[6]*ln(psi1)./psi2 +alpha[8]*ln(psi2)./psi2 +alpha[9]/psi2.*ln(psi3))

           .*exp(zcgrid*alpha);


    x3grid = (alpha[4]/psi3 +alpha[7]*ln(psi1)./psi3 + alpha[9]*ln(psi2)./psi3 + alpha[10]*ln(psi3)./psi3)

           .*exp(zcgrid*alpha);
```

```
    xgrid = x1grid~x2grid~x3grid;

    cgrid = exp(zcgrid*alpha);


    Mean = meanc(xgrid);

    Std  = stdc(xgrid);

endif;



/**********************************************************************************************
*****************

**********************************************************************************************
*****************

                           ANALYTICAL SUR Estimation of the AIM

**********************************************************************************************
*****************

**********************************************************************************************
*****************/




if (FFF $== "TLS") or (FFF $== "TLC");

  goto translog;

endif;


if co_constrained == 1;

  goto coest;
```

```
endif;

    @ Procedure AIMdef:
        a) Defines the Regressors z1, z2, z3 of the AIM(order)Demand System
        b) Defines the Regressors ze1, ze2, ze3 for the Evaluation over the grid Psi
        c) Defines the Regressors AESz(order) for the Evaluation of the AES at prices AESw
        d) Defines the Restriction: RD*beta = R, whereby RD: Design Matrix @
    {z1,z2,z3,ze1,ze2,ze3,AESz1,AESz2,AESz3,RD,r} = pAIMdef(order);
    /* Number of Regressors in each single equations of the demand system */
    cs = cols(z1);
    ze  = (ze1~zeros(ngrid,2*cs))|
        (zeros(ngrid,cs)~ze2~zeros(ngrid,cs))|
        (zeros(ngrid,2*cs)~ze3);
    print /flush "Method: Analytical (symmetry constrained)";
    print /flush "mean(xgrid)   " mean';
    print /flush "Std dev(xgrid)" std';
    print /flush;

    /* Single Equation estimation */
    b_ols1 = invpd(z1'z1)*z1'x1;
    b_ols2 = invpd(z2'z2)*z2'x2;
    b_ols3 = invpd(z3'z3)*z3'x3;
    b_ols  = b_ols1|b_ols2|b_ols3;

    /* Define matrices for the stacked System */
```

```
z  =  (z1~zeros(n,2*cs))|
     (zeros(n,cs)~z2~zeros(n,cs))|
     (zeros(n,2*cs)~z3);
vx =  vec(x);
resols = reshape(vx - z*(b_ols),3,n)';

/* Estimation of Covariance Matrix, whereby omega = sigma(Kronecker)eye(n) */
invsigma = invpd(resols'resols/(n-cs));
invomega = invsigma.*.eye(n);
/* SUR Estimation unrestricted (betau)*/
betau = invpd(z'invomega*z)*z'invomega*vx;
/* SUR Estimation restricted coefficient over the equations
  - formula in Greene (2000): Econometric Analysis, p. 281 */
beta = betau - (invpd(z'invomega*z))*RD'*invpd(RD*invpd(z'invomega*z)*RD')*(RD*betau-R);

for i(1,1000,1);
   ressur   = reshape(vx - z*beta,3,n)';
   invsigma = invpd(ressur'ressur/(n-cs));
   invomega = invsigma.*.eye(n);
   bcovu = invpd(z'invomega*z);
   betau = bcovu*z'invomega*vx;
   betai = betau - bcovu*RD'*invpd(RD*bcovu*RD')*(RD*betau-R);
   if abs(betai-beta) < breakn;
        print /flush "converged at iteration" i;
        break;
```

```
        endif;
        beta = betai;
    endfor;
beta_sur = betai;
/*************************************************************************************
*****************

  Generating outcomes from posterior of beta based on Gibbs Sampler and/or Metropolis-Hastings
Algorithm


*************************************************************************************
*****************/
if MCMC > 0;          /* MCMC Resampling */
   if MCMC == 1;      /* Gibbs Sampler */
      print /flush "Resampling the parameters of the truncated posterior p(b|y)I(b)";
      print /flush "via Gibbs and Accept-Reject Algorithm.";
      vio = 0;
      ns = 100000;          /* number of draws from the posterior */
      nburn = 2000;       /* burn in period */
      bstar = zeros(3*cs,ns);
      betag = beta;
      print /flush "Number of draws:" ns;
      print /flush "Burn in:       " nburn;
      for i(1,nburn+ns,1);
         /*Drawing from Wishart */
         ressur = reshape(vx - z*betag,3,n)';
         sigmastar = reshape(piwishgen(n,ressur'ressur,1),3,3);
```

```
        invsigstar = invpd(sigmastar);
        /* unrestricted estimation */
        bcovstaru = invpd(z'(invsigstar.*.eye(n))*z);
        surbstaru = bcovstaru*z'(invsigstar.*.eye(n))*vx;
        /* resampling the unrestricted surbstaru */
        betagu = surbstaru + chol(bcovstaru)'rndn(3*cs,1);
        /* restricted estimation */
        invRbcovRt = invpd(RD*bcovstaru*RD');
        betag = betagu - bcovstaru*RD'*invRbcovRt*(RD*betagu-R);
        if betag[2] /= betag[5]; print i "wait"; waitc; endif;
        if i > nburn;
          {concavv,monov} = pRegularity(betag);
          if (concavv == 1) or (monov == 1);
              vio = vio + 1;
          else;
              bstar[.,i-vio-nburn] = betag;
          endif;
        endif;
      endfor;
   endif;
for exper(1,2,1);
   if MCMC == 2;          /* Metropolis-Hastings Algorithm */
      mhdiagnostic = 1;   /* 1: every drawn parameter vector shall be analyzed for diagnostic check of
MCMC
                          0: no diagnostics */
```

```
ns = 1000000 ;       /* number of draws from the posterior */
nburn = 2000;      /* burn in period */
if mhdiagnostic == 1;
  bstar = zeros(3*cs,ns+nburn);
endif;
adjustcov = 1;         /* Starting Scaling factor of Cov(beta) in MH-Algorithm */
highc = 250;           /* after highc draws set c to c2 */
adjustcov2 = 0.1;       /* Scaling factor of Cov(beta) in MH-Algorithm */
seed = 10;
/* Defining start values for bcovstar and betac */
{concavv,monov} = pRegularity(beta_sur);
if (concavv == 1) or (monov == 1);
   concavity = 1;
   p  = "ini";
   py = "ini";
   px = "ini";
   betac = pcoestaim(order);
   concavity = 0;
   artbetac = "BETA_SUR_nonnegative";
   print betac;
else;
   betac = beta;
   artbetac = "BETA_SUR";
endif;
print /flush "Resampling the parameters of the truncated posterior p(b|y)I(b)";
```

```
print /flush "via the Metropolis-Hastings Accept-Reject Algorithm: ";
print /flush "Number of draws: " ns;
print /flush "Burn in:         " nburn;
print /flush "Initial betac:   " artbetac;
print /flush "Factor c:        " adjustcov2;
ibeta = 0;
mhi_g_1 = 0;
mhi_ge_urnd = 0;
mhi_l_urnd = 0;
vio = 0;
mhmbstar = zeros(3*cs,1);
mhmbbstar = zeros(3*cs,1);
if mhdiagnostic == 1;
  concavv = 0;
  monov = 0;
endif;
bcovstar = bcovu;
for i(1,nburn+ns,1);
   if i == nburn+1;
      mhmbstar = zeros(3*cs,1);
      mhmbbstar = zeros(3*cs,1);
   endif;
   betau = betac + adjustcov*chol(bcovu)'rndns(3*cs,1,seed); /* bcovu = bcov since it is updated in
iteration - see above*/
   invRbcovRt = invpd(RD*bcovstar*RD');
```

```
betar = betau - bcovstar*RD'*invRbcovRt*(RD*betau-R);
ressurr  = reshape(vx - z*betar,3,n)';
ressurc  = reshape(vx - z*betac,3,n)';
invsigma = invpd(ressurr'ressurr/(n-cs));
invomega = invsigma.*.eye(n);
bcovstar = invpd(z'invomega*z);
if exper == 1;
@ if mhdiagnostic == 0; @
  {concavv,monov} = pRegularity(betar);
@ endif;              @
endif;
if (concavv == 1) or (monov == 1);
  if vio == 0;
    print "first vio at i: " i;
  endif;
  vio = vio + 1;
  @ print /flush "vio: " vio "in draw i:" i "(including nburn)"; @
  mhi = 0; /*mhi: Metropolis-Hastings Indicator */
else;
   mhi = (det(ressurr'ressurr)/det(ressurc'ressurc))^(-n/2);  /*this does the same as possur*/
endif;
if  mhi > 1;
   if mhdiagnostic == 1;
     bstar[.,i] = betar;
   endif;
```

```
    mhmbstar = mhmbstar + betar;
    mhmbbstar = mhmbbstar + betar.*betar;
    mhi_g_1 = mhi_g_1 + 1;
    oldb = 0;
else;
    urnd = rndu(1,1);
    if urnd <= mhi;
        if mhdiagnostic == 1;
          bstar[.,i] = betar;
        endif;
        mhmbstar = mhmbstar + betar;
        mhmbbstar = mhmbbstar + betar.*betar;
        mhi_ge_urnd = mhi_ge_urnd + 1;
        oldb = 0;
    else;
        if mhdiagnostic == 1;
          bstar[.,i] = betac;
        endif;
        mhmbstar = mhmbstar + betac;
        mhmbbstar = mhmbbstar + betac.*betac;
        mhi_l_urnd = mhi_l_urnd + 1;
        oldb = 1;
    endif;
endif;
if mhdiagnostic == 1; betac = bstar[.,i]; endif;
```

```
        if oldb == 0;
          betac = betar;
        endif;
        if (i > highc) and  (vio > 0) and (concavv == 0) and (monov == 0) and (ibeta == 0);
           ibeta = i;
           adjustcov = adjustcov2;
           betacc = betar;
        endif;
      endfor;
      count_accepted = (mhi_g_1 + mhi_ge_urnd) / (nburn+ns) *100;
      print /flush "accepted draws: " count_accepted "%";
      print /flush "mhi_g_1: " mhi_g_1;
      print /flush "mhi_ge_urnd: " mhi_ge_urnd;
      print /flush "mhi_l_urnd: " mhi_l_urnd;
      if mhdiagnostic == 1;
        bstarns = trim(bstar',nburn,0)';
      @   print bstarns'; @
      endif;
   endif;
   beta = mhmbstar./ns;
   stdbmh = sqrt((mhmbbstar./(ns) - beta.*beta)*(ns/(ns-1)));
if exper == 1; bexper1 = bstarns[1:11,.]|bstarns[18:21,.]; save bexper1; clear bexper1; endif;
if exper == 2; bexper2 = bstarns[1:11,.]|bstarns[18:21,.]; save bexper2; clear bexper2; endif;
endfor;
endif;
```

```
goto Eval;


/*****************************************************************************
*****************

*****************************************************************************
*****************

                        CO SUR Estimation of the AIM

*****************************************************************************
*****************

*****************************************************************************
*****************/


coest:


    @ Procedure AIMdef:

        a) Defines the Regressors z1, z2, z3 of the AIM(order)Demand System

        b) Defines the Regressors ze1, ze2, ze3 for the Evaluation over the grid Psi

        c) Defines the Restriction: RD*beta = R, whereby RD: Design Matrix @

    {z1,z2,z3,ze1,ze2,ze3,AESz1,AESz2,AESz3,RD,r} = pAIMdef(order);

    /* Number of Regressors in each single equations of the demand system */

    cs = cols(z1);


    print /flush "mean(xgrid)   " mean';

    print /flush "Std dev(xgrid)" std';

    print /flush;
```

```
    /* Single Equation estimation */

    beta = pcoestaim(order);

    print "beta" beta;

    p  = "ini"; /* Attention: I do this, because I cannot define p,py and px as locals in pcoestaim, */

    py = "ini"; /* since they are needed for the co - procedur */

    px = "ini";


goto Eval;




/************************************************************************************
*****************

*************************************************************************************
*****************

                    SUR Estimation of the Translog(2) s1s2

*************************************************************************************
*****************

*************************************************************************************
*****************/

Translog:


if FFF $== "TLS";


Print /flush "***************************** Translog *****************************";

print /flush;
```

```
Print /flush "*****************************  SUR s1-s2  *****************************";
print /flush;
    zs = ones(n,1)~ln(w);        @ Regressors of the share system            @
    zse= ones(ngrid,1)~ln(psi);   @ Regressors of the share system over the grid @
    s1 = w1.*x1./c;              @ Share of input one                        @
    s2 = w2.*x2./c;              @ share of input two                        @
    s  = s1|s2;                  @ stacked share vector                      @
    cs = cols(zs);

     /* Single Equation estimation of the translog*/
    beta_ols1 = invpd(zs'zs)*zs's1;
    beta_ols2 = invpd(zs'zs)*zs's2;
@      print /flush beta_ols1; @
@      print /flush beta_ols2; @
    res1   = s1-zs*beta_ols1;
    res2   = s2-zs*beta_ols2;
    resols = res1~res2;
    /* Define matrices for the stacked System */
    z    = (zs~zeros(n,cs))|
           (zeros(n,cs)~zs);

     /* Estimation of Covariance Matrix, whereby omega = sigma(Kronecker)eye(n) */
     invsigma = invpd(resols'resols/(n-cs));
 @     print /flush invsigma; @
     invomega = invsigma.*.eye(n);
```

```
/* SUR Estimation unrestricted (betau)*/
betau = invpd(z'invomega*z)*z'invomega*s;

RD ={0  1  1  1  0  0  0  0,
    0  0  0  0  0  1  1  1,
    0  0  1  0  0 -1  0  0};
R  = 0|0|0;

/* SUR Estimation restricted coefficient over the equations */
beta = betau - (invpd(z'invomega*z))*RD'*invpd(RD*invpd(z'invomega*z)*RD')*(RD*betau-R);

for i(1,1000,1);
   res1   = x1-zs*beta[1:cs];
   res2   = x2-zs*beta[cs+1:2*cs];
   ressur = res1~res2;

   invsigma = invpd(ressur'ressur/(n-cs));
   invomega = invsigma.*.eye(n);
   betau = invpd(z'invomega*z)*z'invomega*s;
   betai = betau - (invpd(z'invomega*z))*RD'*invpd(RD*invpd(z'invomega*z)*RD')*(RD*betau-R);
   if abs(betai-beta) < breakn;
        print /flush "converged at iteration" i;
        break;
   endif;
```

```
        beta = betai;
    endfor;

    betaf     = zeros(12,1);
    betaf[1:8] = beta;
    betaf[9]   = 1 - beta[1]-beta[5];
    betaf[10]  = beta[4];
    betaf[11]  = beta[8];
    betaf[12]  = 0 - betaf[11] - beta[4];
    beta      = betaf;
goto Eval;
endif;
/*****************************************************************************
*****************

*****************************************************************************
*****************

                        SUR Estimation of the Translog(2)  Cost-s1-s2

*****************************************************************************
*****************

*****************************************************************************
*****************/


if FFF $== "TLC";
   Print /flush "***************************** Translog ****************************";
   print /flush;
   Print /flush "*****************************  SUR Cost-s1-s2
*****************************";
```

```
print /flush;

/* Dependend Variables */

zs = ones(n,1)~ln(w);                @ Regressors of the share system            @

zse= ones(ngrid,1)~ln(psi);          @ Regressors of the share system over the grid @

zc = ones(n,1)~ln(w)                 @ Regressors of the ln-cost function        @

   ~.5*ln(w1).*ln(w1) ~  ln(w1).*ln(w2)  ~   ln(w1).*ln(w3)

                ~.5*ln(w2).*ln(w2) ~   ln(w2).*ln(w3)

                           ~.5*ln(w3).*ln(w3);

/* Inpendend Variables */

s1 = w1.*x1./c;                  @ Share of input one                    @

s2 = w2.*x2./c;                  @ share of input two                    @

s  = ln(c)|s1|s2;                @ stacked ln-cost & share vector            @

cs = cols(zs);

cc = cols(zc);

 /* Single Equation estimation of the translog*/

beta_ols1 = invpd(zc'zc)*zc'ln(c);

beta_ols2 = invpd(zs'zs)*zs's1;

beta_ols3 = invpd(zs'zs)*zs's2;

res1   = ln(c)-zc*beta_ols1;

res2   = s1-zs*beta_ols2;

res3   = s2-zs*beta_ols3;

resols = res1~res2~res3;

/* Define matrices for the stacked System */

z    = (zc~zeros(n,2*cs))|

      (zeros(n,cc)~zs~zeros(n,cs))|
```

```
     (zeros(n,cc)~zeros(n,cs)~zs);
/* Estimation of Covariance Matrix, whereby omega = sigma(Kronecker)eye(n) */
invsigma = invpd(resols'resols/(n-(cc+2*cs)));
invomega = invsigma.*.eye(n);

/* SUR Estimation unrestricted (betau)*/
betau = invpd(z'invomega*z)*z'invomega*s;

RD = {0  1  1  1  0  0  0  0  0  0  0  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  0  1  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0,
    0  0  0  0  1  0  0  0  0  0  0 -1  0  0  0  0  0  0,
    0  0  0  0  0  1  0  0  0  0  0  0 -1  0  0  0  0  0,
    0  0  0  0  0  0  1  0  0  0  0  0  0 -1  0  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  0  1  0  0  0  0  0  0  0  0 -1  0,
    0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0 -1,
    0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0,
    0  0  0  0  0  1  0  1  1  0  0  0  0  0  0  0  0  0,
    0  0  0  0  0  0  1  0  1  1  0  0  0  0  0  0  0  0};
R = 1|zeros(11,1);

/* SUR Estimation restricted coefficient over the equations */
beta = betau - (invpd(z'invomega*z))*RD'*invpd(RD*invpd(z'invomega*z)*RD')*(RD*betau-R);
for i(1,1000,1);
```

```
        resv    = s-z*beta;
        ressur  = resv[1:n]~resv[(n+1):(2*n)]~resv[(2*n+1):(3*n)];
        invsigma = invpd(ressur'ressur/(n-(cc+2*cs)));
        invomega = invsigma.*.eye(n);
        betau   = invpd(z'invomega*z)*z'invomega*s;
        betai   = betau - (invpd(z'invomega*z))*RD'*invpd(RD*invpd(z'invomega*z)*RD')*(RD*betau-R);
        if abs(betai-beta) < breakn;
            print /flush "converged at iteration" i;
            break;
        endif;
        beta = betai;
    endfor;

goto Eval;
endif;

/*************************************************************************************
*****************

*************************************************************************************
*****************

                    Evaluation of the Estimation at the Grid Psi

*************************************************************************************
*****************

*************************************************************************************
*****************/
```

Eval:

```
/*************************************************************************************
*****************

                    Evaluation 1: Check Average and Maximal Errors

*************************************************************************************
*****************/


if FFF $== "AIM";
    /* Calculate x at the grid */
    x1gridhat = ze1*beta[1:cs];
    x2gridhat = ze2*beta[(cs+1):(2*cs)];
    x3gridhat = ze3*beta[((2*cs)+1):(3*cs)];
    @ xgridhat = ze*beta;  ATTENTION: is this the same as equations above? @
endif;
if FFF $== "TLS";
    /* Calculate x at the grid */
    s1gridhat = zse*beta[1:cs];
    s2gridhat = zse*beta[cs+1:2*cs];
    s3gridhat = zse*beta[(cs*2)+1:3*cs];
    @ ATTENTION: in the case of C-s1-s2 I divide by the calculated cgrid - as it should be !
      However, in the case of s1-s2 I divide by the observed cgrid because it is not
      possible to calculate cgridhat. Is this correct?? @
    x1gridhat = s1gridhat.*cgrid./psi1;
    x2gridhat = s2gridhat.*cgrid./psi2;
    x3gridhat = s3gridhat.*cgrid./psi3;
```

```
endif;
if FFF $== "TLC";
    /* Calculate x at the grid */
    s1gridhat = zse*beta[(cc+1):(cc+cs)];
    s2gridhat = zse*beta[(cc+cs+1):(cc+(2*cs))];
    s3gridhat = zse[.,1]*beta[4] + zse[.,2]*beta[7] + zse[.,3]*beta[9] + zse[.,4]*beta[10];

    zcgrid = ones(ngrid,1)~ln(psi)       @ Regressors of the ln-cost function          @
        ~.5*ln(psi1).*ln(psi1) ~  ln(psi1).*ln(psi2)  ~   ln(psi1).*ln(psi3)
                    ~.5*ln(psi2).*ln(psi2) ~   ln(psi2).*ln(psi3)
                                ~.5*ln(psi3).*ln(psi3);
    x1gridhatnl = (beta[2]/psi1 + beta[5]*ln(psi1)./psi1 + beta[6]/psi1.*ln(psi2) + beta[7]/psi1.*ln(psi3 ))
            .*exp(zcgrid*beta[1:10]);
    x2gridhatnl = (beta[3]/psi2 + beta[6]*ln(psi1)./psi2 + beta[8]*ln(psi2)./psi2 + beta[9]/psi2.*ln(psi3))
            .*exp(zcgrid*beta[1:10]);
    x3gridhatnl = (beta[4]/psi3 + beta[7]*ln(psi1)./psi3 + beta[9]*ln(psi2)./psi3 + beta[10]*ln(psi3)./psi3)
            .*exp(zcgrid*beta[1:10]);
    cgridhat    = exp(zcgrid*beta[1:10]);

    x1gridhat = s1gridhat.*cgridhat./psi1;
    x2gridhat = s2gridhat.*cgridhat./psi2;
    x3gridhat = s3gridhat.*cgridhat./psi3;
endif;

xgridhat   = x1gridhat~x2gridhat~x3gridhat;
```

```
averror    = (sumc(abs(xgridhat - xgrid))/ngrid)';
maxerror1  = (maxc(xgrid - xgridhat))';
maxerror2  = -(maxc(xgridhat - xgrid))';
maxerror   = zeros(1,3);
for v(1,3,1);
   if abs(maxerror1[1,v]) >= abs(maxerror2[1,v]);
      maxerror[1,v] = maxerror1[1,v];
      else;
      maxerror[1,v] = maxerror2[1,v];
   endif;
endfor;
print /flush;
print /flush "averror:      " averror;
print /flush "maxerror:     " maxerror;
print /flush;



/*****************************************************************************
*****************
                    Evaluation 2: - Check for Violations of Curvature !!
                                - Euler Tests a) H*w = 0 b) Grad*w = c
                                - Estimate AES
*****************************************************************************
*****************/
        vee     = 0;            @ Initializing Counters for the for loop @
        zyy     = 0;            @ Initializing Counters for the for loop @
```

```
        cyy     = 0;            @ Initializing Counters for the for loop @
        mono    = 0;            @ Initializing Counters for the for loop @
        concavg  = zeros(ngrid,1);  @ Initializing vector for graphics over the grid @
        monovg   = zeros(ngrid,1);  @ Initializing vector for graphics over the grid @
/*************************************************************************************
*****************
                                For AIM(1)
*************************************************************************************
*****************/
if FFF $== "AIM";
 if order == 1;
     for i(1,ngrid,1);
       hfull = phaim(ygrid[i],psi1[i],psi2[i],psi3[i]);
       h = trim(hfull,1,0); h = trim(h',1,0);

       @ Check Curvature @
       v = eigrs(h);
       @ Fuzzy comparison functions feq, fle etc.:
         These functions use _fcmptol to fuzz the comparison operations to allow for round off error. @
       _fcmptol = fcmptol_small;
       ve = fle(v,0);  @ fuzzy operator y = 1 --> true, y = 0 --> false @
       if ve == 1;
          vee = vee + ve;
       else;
          concavg[i] = 1;
       endif;
```

```
@ Check Monotonocity @
 if xgridhat[i,.] > 0;
     mono = mono + 1;
 else;
     monovg[i] = 1;
 endif;

@ Check Euler Condition H*psi' = 0; (see Diewert, Wales, 1987, Econometrica) @
 zero = hfull*psi[i,.]';
 zy = feq(zero,0);
 if zy == 1;
    zyy = zyy + zy;
 endif;

@ Check Gradient*w = C (see Diewert, Wales, 1987, Econometrica) @

 costgrid = ygrid[i]*(beta[1]*psi1[i]  + beta[4]*psi2[i] + beta[7]*psi3[i]
 + beta[2]*psi1[i]^(1/2)*psi2[i]^(1/2) + beta[3]*psi1[i]^(1/2)*psi3[i]^(1/2)
 + beta[6]*psi2[i]^(1/2)*psi3[i]^(1/2));

 cost = xgridhat[i,.]*psi[i,.]';
 cy = feq(cost,costgrid);
 if cy == 1;
    cyy = cyy + cy;
```

```
        else;
          format 10,15;
        endif;
        _fcmptol = fcmptol_large;

    endfor;

    @ Calculate Allen Elasticities of Substitution @
    AEShat = pAES(beta);
    print /flush "Estimated Allen Elasticities of Substitution of the AIM1";
    print /flush "AEShat: " AEShat;

endif;

/*************************************************************************************
*****************

                            For AIM(2)

*************************************************************************************
*****************/

if order == 2;
    for i(1,ngrid,1);
        hfull = phaim(ygrid[i],psi1[i],psi2[i],psi3[i]);
        h = trim(hfull,1,0); h = trim(h',1,0);

        @ Check Curvature @
```

```
v = eigrs(h);
@ Fuzzy comparison functions feq, fle etc.:
  These functions use _fcmptol to fuzz the comparison operations to allow for round off error. @
_fcmptol = fcmptol_small;
ve  = fle(v,0); @ fuzzy operator y = 1 --> true, y = 0 --> false @
if ve == 1;
    vee = vee + ve;
else;
    concavg[i] = 1;
endif;

@ Check Monotonocity @
if xgridhat[i,.] > 0;
    mono = mono + 1;
else;
    monovg[i] = 1;
endif;

@ Check Euler Condition H*psi' = 0; (see Diewert, Wales, Econometrica) @
  zero = hfull*psi[i,.]';
  zy = feq(zero,0);
  if zy == 1;
    zyy = zyy + zy;
  endif;
```

```
@ Check Gradient*w = C (see Diewert, Wales, Econometrica) @

costgrid = ygrid[i]*(
 beta[1]  *psi1[i] + beta[11]*psi2[i] + beta[21]*psi3[i]
+ beta[2 ]*psi1[i]^.75*psi2[i]^.25
+ beta[3 ]*psi1[i]^.75*psi3[i]^.25
+ beta[4 ]*psi1[i]^.5 *psi2[i]^.5
+ beta[5 ]*psi1[i]^.5 *psi2[i]^.25 *psi3[i]^.25
+ beta[6 ]*psi1[i]^.5 *psi3[i]^.5
+ beta[7 ]*psi1[i]^.25*psi2[i]^.75
+ beta[8 ]*psi1[i]^.25*psi2[i]^.5  *psi3[i]^.25
+ beta[9 ]*psi1[i]^.25*psi2[i]^.25 *psi3[i]^.5
+ beta[10]*psi1[i]^.25*psi3[i]^.75
+ beta[18]*psi2[i]^.75*psi3[i]^.25
+ beta[19]*psi2[i]^.5 *psi3[i]^.5
+ beta[20]*psi2[i]^.25*psi3[i]^.75);

cost = xgridhat[i,.]*psi[i,.]';

cy = feq(cost,costgrid);
if cy == 1;
   cyy = cyy + 1;
   else;
   format 10,15;
```

```
        endif;
        _fcmptol = fcmptol_large;
    endfor;

    @ Calculate Allen Elasticities of Substitution @
    AEShat = pAES(beta);
    print /flush "Estimated Allen Elasticities of Substitution of the AIM1";
    print /flush "AEShat: " AEShat;

endif;

/*************************************************************************************
*****************
                            For AIM(3)
*************************************************************************************
*****************/

if order == 3;
    for i(1,ngrid,1);
        hfull = phaim(ygrid[i],psi1[i],psi2[i],psi3[i]);
        h = trim(hfull,1,0); h = trim(h',1,0);

        @ Check Curvature @
```

```
v = eigrs(h);
@ Fuzzy comparison functions feq, fle etc.:
  These functions use _fcmptol to fuzz the comparison operations to allow for round off error. @
_fcmptol = fcmptol_small;
ve  = fle(v,0); @ fuzzy operator y = 1 --> true, y = 0 --> false @
if ve == 1;
    vee = vee + ve;
else;
    concavg[i] = 1;
endif;

@ Check Monotonocity @
if xgridhat[i,.] > 0;
    mono = mono + 1;
else;
    monovg[i] = 1;
endif;

@ Check Euler Condition H*psi' = 0; (see Diewert, Wales, Econometrica) @
  zero = hfull*psi[i,.]';
  zy = feq(zero,0);
  if zy == 1;
    zyy = zyy + zy;
  endif;
```

```
@ Check Gradient*w = C (see Diewert, Wales, Econometrica) @


 costgrid = ygrid[i]*(
   beta[68]*psi2[i]^.625*psi3[i]^.375 +
   beta[67]*psi2[i]^.750*psi3[i]^.250 +
   beta[28]*psi1[i]^.250*psi3[i]^.750 +
   beta[69]*psi2[i]^.500*psi3[i]^.500 +
   beta[29]*psi1[i]^.125*psi2[i]^.875 +
   beta[7]*psi1[i]^.625*psi2[i]^.375 +
   beta[4]*psi1[i]^.750*psi2[i]^.250 +
   beta[21]*psi1[i]^.375*psi3[i]^.625 +
   beta[72]*psi2[i]^.125*psi3[i]^.875 +
   beta[36]*psi1[i]^.125*psi3[i]^.875 +
   beta[6]*psi1[i]^.750*psi3[i]^.250 +
   beta[15]*psi1[i]^.500*psi3[i]^.500 +
   beta[3]*psi1[i]^.875*psi3[i]^.125 +
   beta[71]*psi2[i]^.250*psi3[i]^.750 +
   beta[11]*psi1[i]^.500*psi2[i]^.500 +
   beta[70]*psi2[i]^.375*psi3[i]^.625 +
   beta[2]*psi1[i]^.875*psi2[i]^.125 +
   beta[66]*psi2[i]^.875*psi3[i]^.125 +
   beta[10]*psi1[i]^.625*psi3[i]^.375 +
   beta[22]*psi1[i]^.250*psi2[i]^.750 +
   beta[16]*psi1[i]^.375*psi2[i]^.625 +
   beta[5]*psi1[i]^.750*psi2[i]^.125*psi3[i]^.125 +
```

```
  beta[1]*psi1[i]^1.00 +
  beta[37]*psi2[i]^1.00 +
  beta[73]*psi3[i]^1.00 +
  beta[8]*psi1[i]^.625*psi2[i]^.250*psi3[i]^.125 +
  beta[9]*psi1[i]^.625*psi2[i]^.125*psi3[i]^.250 +
  beta[12]*psi1[i]^.500*psi2[i]^.375*psi3[i]^.125 +
  beta[13]*psi1[i]^.500*psi2[i]^.250*psi3[i]^.250 +
  beta[14]*psi1[i]^.500*psi2[i]^.125*psi3[i]^.375 +
  beta[17]*psi1[i]^.375*psi2[i]^.500*psi3[i]^.125 +
  beta[18]*psi1[i]^.375*psi2[i]^.375*psi3[i]^.250 +
  beta[19]*psi1[i]^.375*psi2[i]^.250*psi3[i]^.375 +
  beta[20]*psi1[i]^.375*psi2[i]^.125*psi3[i]^.500 +
  beta[23]*psi1[i]^.250*psi2[i]^.625*psi3[i]^.125 +
  beta[24]*psi1[i]^.250*psi2[i]^.500*psi3[i]^.250 +
  beta[25]*psi1[i]^.250*psi2[i]^.375*psi3[i]^.375 +
  beta[26]*psi1[i]^.250*psi2[i]^.250*psi3[i]^.500 +
  beta[27]*psi1[i]^.250*psi2[i]^.125*psi3[i]^.625 +
  beta[30]*psi1[i]^.125*psi2[i]^.750*psi3[i]^.125 +
  beta[31]*psi1[i]^.125*psi2[i]^.625*psi3[i]^.250 +
  beta[32]*psi1[i]^.125*psi2[i]^.500*psi3[i]^.375 +
  beta[33]*psi1[i]^.125*psi2[i]^.375*psi3[i]^.500 +
  beta[34]*psi1[i]^.125*psi2[i]^.250*psi3[i]^.625 +
  beta[35]*psi1[i]^.125*psi2[i]^.125*psi3[i]^.750);

cost = xgridhat[i,.]*psi[i,.]';
```

```
        cy = feq(cost,costgrid);
        if cy == 1;
          cyy = cyy + 1;
        endif;
        _fcmptol = fcmptol_large;
    endfor;

@ Calculate Allen Elasticities of Substitution @
    AEShat = pAES(beta);
    print /flush "Estimated Allen Elasticities of Substitution of the AIM3";
    print /flush "AEShat: " AEShat;

 endif;
endif;



/*************************************************************************************
*****************

                Curvature Check for the Translog(2) with the Jorgenson/Fraumeni-Method

*************************************************************************************
*****************/
if (FFF $== "TLS") or (FFF $=="TLC");
   if FFF $== "TLS";
     gammam = reshape(beta,3,4);
     gammam = trim(gammam',1,0);
```

```
   endif;

   if FFF $== "TLC";

      gammam = beta[12:14]'~beta[16:18]'~beta[14]~beta[18]~beta[10];

      gammam = reshape(gammam,3,3);

   endif;

   v     = eigrs(gammam);

   _fcmptol = fcmptol_small;

   ve = fle(v,0);

   if ve == 1;

      print /flush "curvature globally correct (Jorgenson/Fraumeni-Method) ";

   else;

      print /flush "curvature globally violated (Jorgenson/Fraumeni-Method)";

   endif;

endif;



/*********************************************************************************
*****************

                        Evaluation Check 2 for the Translog TLC

*********************************************************************************
*****************/

if FFF $== "TLC";

      for i(1,ngrid,1);

          h11 = (-beta[2]/psi1[i]^2+beta[5]/psi1[i]^2-beta[5]*ln(psi1[i])/psi1[i]^2-
beta[6]/psi1[i]^2*ln(psi2[i])-
beta[7]/psi1[i]^2*ln(psi3[i]))*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(psi3[i])+1/2*
beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8]*ln(psi2[i])
```

^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2)+(beta[2]/psi1[i]+beta[5]*ln(psi1[i])/psi1[i]+
beta[6]/psi1[i]*ln(psi2[i])+beta[7]/psi1[i]*ln(psi3[i]))^2*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i
])+beta[4]*ln(psi3[i])+1/2*beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3
[i])+1/2*beta[8]*ln(psi2[i])^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2);

h12 =
beta[6]/psi1[i]/psi2[i]*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(psi3[i])+1/2*beta[5
]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8]*ln(psi2[i])^2+be
ta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2)+(beta[2]/psi1[i]+beta[5]*ln(psi1[i])/psi1[i]+beta[
6]/psi1[i]*ln(psi2[i])+beta[7]/psi1[i]*ln(psi3[i]))*(beta[3]/psi2[i]+beta[6]*ln(psi1[i])/psi2[i]+beta[8]*ln(psi
2[i])/psi2[i]+beta[9]/psi2[i]*ln(psi3[i]))*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(psi
3[i])+1/2*beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8]
*ln(psi2[i])^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2);

h13 =
beta[7]/psi1[i]/psi3[i]*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(psi3[i])+1/2*beta[5
]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8]*ln(psi2[i])^2+be
ta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2)+(beta[2]/psi1[i]+beta[5]*ln(psi1[i])/psi1[i]+beta[
6]/psi1[i]*ln(psi2[i])+beta[7]/psi1[i]*ln(psi3[i]))*(beta[4]/psi3[i]+beta[7]*ln(psi1[i])/psi3[i]+beta[9]*ln(psi
2[i])/psi3[i]+beta[10]*ln(psi3[i])/psi3[i])*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(p
si3[i])+1/2*beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8
]*ln(psi2[i])^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2);

h22 = (-beta[3]/psi2[i]^2-beta[6]*ln(psi1[i])/psi2[i]^2+beta[8]/psi2[i]^2-
beta[8]*ln(psi2[i])/psi2[i]^2-
beta[9]/psi2[i]^2*ln(psi3[i]))*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(psi3[i])+1/2*
beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8]*ln(psi2[i])
^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2)+(beta[3]/psi2[i]+beta[6]*ln(psi1[i])/psi2[i]+
beta[8]*ln(psi2[i])/psi2[i]+beta[9]/psi2[i]*ln(psi3[i]))^2*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i
])+beta[4]*ln(psi3[i])+1/2*beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3
[i])+1/2*beta[8]*ln(psi2[i])^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2);

h23 =
beta[9]/psi2[i]/psi3[i]*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(psi3[i])+1/2*beta[5
]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8]*ln(psi2[i])^2+be
ta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2)+(beta[3]/psi2[i]+beta[6]*ln(psi1[i])/psi2[i]+beta[
8]*ln(psi2[i])/psi2[i]+beta[9]/psi2[i]*ln(psi3[i]))*(beta[4]/psi3[i]+beta[7]*ln(psi1[i])/psi3[i]+beta[9]*ln(psi
2[i])/psi3[i]+beta[10]*ln(psi3[i])/psi3[i])*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(p
si3[i])+1/2*beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8]
*ln(psi2[i])^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2);

```
h33 = (-beta[4]/psi3[i]^2-beta[7]*ln(psi1[i])/psi3[i]^2-
beta[9]*ln(psi2[i])/psi3[i]^2+beta[10]/psi3[i]^2-
beta[10]*ln(psi3[i])/psi3[i]^2)*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(psi2[i])+beta[4]*ln(psi3[i])+1/
2*beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(psi3[i])+1/2*beta[8]*ln(psi2[
i])^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2)+(beta[4]/psi3[i]+beta[7]*ln(psi1[i])/psi3[
i]+beta[9]*ln(psi2[i])/psi3[i]+beta[10]*ln(psi3[i])/psi3[i])^2*exp(beta[1]+beta[2]*ln(psi1[i])+beta[3]*ln(p
si2[i])+beta[4]*ln(psi3[i])+1/2*beta[5]*ln(psi1[i])^2+beta[6]*ln(psi1[i])*ln(psi2[i])+beta[7]*ln(psi1[i])*ln(
psi3[i])+1/2*beta[8]*ln(psi2[i])^2+beta[9]*ln(psi2[i])*ln(psi3[i])+1/2*beta[10]*ln(psi3[i])^2);


hfull = h11~h12~h13 |
        h12~h22~h23 |
        h13~h23~h33 ;
h = h11~h12|
    h12~h22;


@ Check Curvature @
v = eigrs(h);
@ Fuzzy comparison functions feq, fle etc.:
 These functions use _fcmptol to fuzz the comparison operations to allow for round off error. @
_fcmptol = fcmptol_small;
ve  = fle(v,0); @ fuzzy operator y = 1 --> true, y = 0 --> false @
if ve == 1;
    vee = vee + ve;
else;
    concavg[i] = 1;
endif;


@ Check Monotonocity @
```

```
    if xgridhat[i,.] > 0;
       mono = mono + 1;
    else;
       monovg[i] = 1;
    endif;

    @ Check Euler Condition H*psi' = 0; (see Diewert, Wales, 1987, Econometrica) @
    zero = hfull*psi[i,.]';
    zy = feq(zero,0);
    if zy == 1;
      zyy = zyy + zy;
    endif;

    @ Check Gradient*w = C (see Diewert, Wales, 1987, Econometrica) @

    cost = xgridhat[i,.]*psi[i,.]';
    cy = feq(cost,cgridhat[i]);    @ cgridhat for TLC already defined above @
    if cy == 1;
       cyy = cyy + cy;
    else;
       format 10,15;
    endif;
    _fcmptol = fcmptol_large;
endfor;
```

```
@ Calculation of the Allen Ellasticities of Substitution evaluated at w=1 @
AESzs = 1~ln(AESw1)~ln(AESw2)~ln(AESw3);   @ Regressors of the share system evaluated at w = 1 @
AESzc = AESzs                          @ Regressors of the ln-cost function evaluated at w = 1 @
   ~.5*ln(AESw1).*ln(AESw1) ~ln(AESw1).*ln(AESw2)    ~ln(AESw1).*ln(AESw3)
                  ~.5*ln(AESw2).*ln(AESw2) ~ln(AESw2).*ln(AESw3)
                                 ~.5*ln(AESw3).*ln(AESw3);
AESs1hat = AESzs*beta[(cc+1):(cc+cs)];
AESs2hat = AESzs*beta[(cc+cs+1):(cc+(2*cs))];
AESs3hat = AESzs[.,1]*beta[4] + AESzs[.,2]*beta[7] + AESzs[.,3]*beta[9] + AESzs[.,4]*beta[10];
AESchat  = exp(AESzc*beta[1:10]);
AESx1hat = AESs1hat.*AESchat./AESw1;
AESx2hat = AESs2hat.*AESchat./AESw2;
AESx3hat = AESs3hat.*AESchat./AESw3;
AESxhat = AESx1hat~AESx2hat~AESx3hat;

test =  AESx1hat.*AESw1+AESx2hat.*AESw2+AESx3hat.*AESw3 - AESchat;
m1hat = AESx1hat.*AESw1./AESchat;  /*m : shares */
m2hat = AESx2hat.*AESw2./AESchat;
m3hat = AESx3hat.*AESw3./AESchat;
/* formula from Guilkby / Lovell, 1980, International Economic Review,
  "On the Flexibility of the Translog Approximation" */
s12hat = (beta[6] + m1hat.*m2hat)./(m1hat.*m2hat);
s13hat = (beta[7] + m1hat.*m3hat)./(m1hat.*m3hat);
s23hat = (beta[9] + m2hat.*m3hat)./(m2hat.*m3hat);
print /flush "Estimated Allen Elasticities of Substitution of the TL";
```

```
    print /flush "s12: " s12hat; print /flush "s13: " s13hat; print /flush "s23: " s23hat;

endif;

if (FFF $== "AIM") or (FFF $== "TLC");
 vee = (vee/ngrid)*100;


 if ((FFF $== "AIM") and ((order == 1) or (order == 2))) or (FFF $=="TLC");
   mono = (mono/ngrid)*100;
 endif;

 zyy = (zyy/ngrid)*100;

 cyy = (cyy/ngrid)*100;
endif;



/*****************************************************/
/*****************************************************/
/*          PROCEDURES                   */
/*****************************************************/
/*****************************************************/
```

```
/*****************************************************/
/*            PROCEDURES FOR THE CO  (AIM only)  */
/*****************************************************/



/********* Objective Function */
    proc fct(beta);
     retp( (py - px*beta)'(py - px*beta) );
    endp;

/********* Equality Constraint */


    proc eqpaim(beta);
      local result;
      if order == 1;
        result    = zeros(3,1);
        result[1] = beta[2] - beta[5];
        result[2] = beta[3] - beta[8];
        result[3] = beta[6] - beta[9];
        retp(result);
      endif;
      if order == 2;
        result     = zeros(15,1);
        result[1]  = beta[2]  - beta[12];
        result[2]  = beta[3]  - beta[22];
```

```
  result[3]  = beta[4]  - beta[13];
  result[4]  = beta[5]  - beta[14];
  result[5]  = beta[5]  - beta[23];
  result[6]  = beta[6]  - beta[24];
  result[7]  = beta[7]  - beta[15];
  result[8]  = beta[8]  - beta[16];
  result[9]  = beta[8]  - beta[25];
  result[10] = beta[9]  - beta[17];
  result[11] = beta[9]  - beta[26];
  result[12] = beta[10] - beta[27];
  result[13] = beta[18] - beta[28];
  result[14] = beta[19] - beta[29];
  result[15] = beta[20] - beta[30];
  retp(result);
endif;
if order == 3;
  result    = zeros(63,1);
  result[1 ] =beta[2 ]  -   beta[38 ];
  result[2 ] =beta[3 ]  -   beta[74 ];
  result[3 ] =beta[4 ]  -   beta[39 ];
  result[4 ] =beta[5 ]  -   beta[40 ];
  result[5 ] =beta[6 ]  -   beta[76 ];
  result[6 ] =beta[7 ]  -   beta[41 ];
  result[7 ] =beta[8 ]  -   beta[42 ];
  result[8 ] =beta[9 ]  -   beta[43 ];
```

```
result[9 ] =beta[10]  -   beta[79 ];

result[10] =beta[11]  -   beta[44 ];

result[11] =beta[12]  -   beta[45 ];

result[12] =beta[13]  -   beta[46 ];

result[13] =beta[14]  -   beta[47 ];

result[14] =beta[15]  -   beta[83 ];

result[15] =beta[16]  -   beta[48 ];

result[16] =beta[17]  -   beta[49 ];

result[17] =beta[18]  -   beta[50 ];

result[18] =beta[19]  -   beta[51 ];

result[19] =beta[20]  -   beta[52 ];

result[20] =beta[21]  -   beta[88 ];

result[21] =beta[22]  -   beta[53 ];

result[22] =beta[23]  -   beta[54 ];

result[23] =beta[24]  -   beta[55 ];

result[24] =beta[25]  -   beta[56 ];

result[25] =beta[26]  -   beta[57 ];

result[26] =beta[27]  -   beta[58 ];

result[27] =beta[28]  -   beta[94 ];

result[28] =beta[29]  -   beta[59 ];

result[29] =beta[30]  -   beta[60 ];

result[30] =beta[31]  -   beta[61 ];

result[31] =beta[32]  -   beta[62 ];

result[32] =beta[33]  -   beta[63 ];

result[33] =beta[34]  -   beta[64 ];
```

```
result[34] =beta[35]  -   beta[65 ];

result[35] =beta[36]  -   beta[101];

result[36] =beta[66]  -   beta[102];

result[37] =beta[67]  -   beta[103];

result[38] =beta[68]  -   beta[104];

result[39] =beta[69]  -   beta[105];

result[40] =beta[70]  -   beta[106];

result[41] =beta[71]  -   beta[107];

result[42] =beta[72]  -   beta[108];

result[43] =beta[40 ] -   beta[75 ];

result[44] =beta[42 ] -   beta[77 ];

result[45] =beta[43 ] -   beta[78 ];

result[46] =beta[45 ] -   beta[80 ];

result[47] =beta[46 ] -   beta[81 ];

result[48] =beta[47 ] -   beta[82 ];

result[49] =beta[49 ] -   beta[84 ];

result[50] =beta[50 ] -   beta[85 ];

result[51] =beta[51 ] -   beta[86 ];

result[52] =beta[52 ] -   beta[87 ];

result[53] =beta[54 ] -   beta[89 ];

result[54] =beta[55 ] -   beta[90 ];

result[55] =beta[56 ] -   beta[91 ];

result[56] =beta[57 ] -   beta[92 ];

result[57] =beta[58 ] -   beta[93 ];

result[58] =beta[60 ] -   beta[95 ];
```

```
    result[59] =beta[61 ] -   beta[96 ];
    result[60] =beta[62 ] -   beta[97 ];
    result[61] =beta[63 ] -   beta[98 ];
    result[62] =beta[64 ] -   beta[99 ];
    result[63] =beta[65 ] -   beta[100];
    retp(result);
  endif;
endp;

/********* Gradient of Equality Constraint */
proc eqjaim(beta);
    local result;
    result =  RD;
    retp(result);
endp;

/********* Gradient of Objective Function */

 proc gp(beta);
   retp(-2*px'py + 2*px'px*beta);
 endp;

/********* Hessian of Objective Function */
 proc hsp(beta);
   retp(2*px'px);
```

```
    endp;

/*****************************************************/
/*            PROCEDURES FOR THE GIBBS          */
/*****************************************************/

/******** Check Regularity for each Gibbs Draw */
proc (2) = pRegularity(beta);
 local concav,concav1,concavv,monov,xgridhati,x1gridhati,x2gridhati,x3gridhati;
 concavv = 0; monov = 0;
 _fcmptol = fcmptol_small;
 /* Check Monotonicity */
 x1gridhati = ze1*beta[1:cs];
 x2gridhati = ze2*beta[(cs+1):(2*cs)];
 x3gridhati = ze3*beta[((2*cs)+1):(3*cs)];
 xgridhati  = x1gridhati~x2gridhati~x3gridhati;
 clear x1gridhati, x2gridhati, x3gridhati;
 monov = not fge(xgridhati,0);
 clear xgridhati;
 if monov == 1;
    goto breakregularity;
 endif;
 for i(1,ngrid,1);
    h = phaimmh(ygrid[i],psi1[i],psi2[i],psi3[i]);
    /* Check Curvature */
```

```
    concav = eigrs(h);
    /* Fuzzy comparison functions feq, fle etc.:
     These functions use _fcmptol to fuzz the comparison operations to allow for round off error. */
    concav1 = fle(concav,0);  @ fuzzy operator y = 1 --> true, y = 0 --> false @
    if concav1 /= 1;
        concavv = 1;
        goto breakregularity;
    endif;
  endfor;
  breakregularity:
  retp(concavv,monov);
endp;



proc (11) = pAIMDef(order);
local RD1, RD2, RD3;
/*************************************************************************************
******************
                        Definitions for the AIM(I)
*************************************************************************************
****************/
if order == 1;
   print "********** AIM(1) **********";
   print;
   /* Define Regressors of the AIM(1) (see, Terrell, 1995, p. 3) */
```

```
   z1 = y.*(ones(n,1) ~ 0.5*w1^(-0.5).*w2^(.5 ) ~ 0.5*w1^(-0.5).*w3^( .5));
   z2 = y.*(ones(n,1) ~ 0.5*w1^( 0.5).*w2^(-.5) ~ 0.5*w2^(-0.5).*w3^( .5));
   z3 = y.*(ones(n,1) ~ 0.5*w1^( 0.5).*w3^(-.5) ~ 0.5*w2^( 0.5).*w3^(-.5));

   /* Construction of the AIM(1) Regressor for the grid*/

   ze1 = ygrid.*(ones(ngrid,1) ~ 0.5*psi1^(-0.5).*psi2^(.5 ) ~ 0.5*psi1^(-0.5).*psi3^( .5));
   ze2 = ygrid.*(ones(ngrid,1) ~ 0.5*psi1^( 0.5).*psi2^(-.5) ~ 0.5*psi2^(-0.5).*psi3^( .5));
   ze3 = ygrid.*(ones(ngrid,1) ~ 0.5*psi1^( 0.5).*psi3^(-.5) ~ 0.5*psi2^( 0.5).*psi3^(-.5));

   /* Construction of the AIM(1) Regressor for the Evaluation at AESw*/

   AESz1 = AESy.*(ones(1,1) ~ 0.5*AESw1^(-0.5).*AESw2^(.5 ) ~ 0.5*AESw1^(-0.5).*AESw3^( .5));
   AESz2 = AESy.*(ones(1,1) ~ 0.5*AESw1^( 0.5).*AESw2^(-.5) ~ 0.5*AESw2^(-0.5).*AESw3^( .5));
   AESz3 = AESy.*(ones(1,1) ~ 0.5*AESw1^( 0.5).*AESw3^(-.5) ~ 0.5*AESw2^( 0.5).*AESw3^(-.5));

  /* Define design matrix R for the restrictions RD*Beta = R of the stacked System (for AIM(1) only !! ) */

   RD1 = { 0  1  0  0 -1  0  0  0  0 };
   RD2 = { 0  0  1  0  0  0  0 -1  0 };
   RD3 = { 0  0  0  0  0  1  0  0 -1 };
   RD = RD1|RD2|RD3;

   R = zeros(3,1);
endif;
```

```
/*************************************************************************************
*****************

                    Definitions for the AIM(2)

*************************************************************************************
*****************/

if order == 2;

   print "********** AIM(2) **********";

   print;


    /* Define Regressors of the AIM(2) (see, Terrell, 1995, p. 3) */

   z1 = y.*(ones(n,1) ~ .75*w1^(-.25).*w2^(.25)  ~ .75*w1^(-.25).*w3^(.25)

            ~ .5*w1^(-.5).*w2^(.5)    ~ .5*w1^(-.5).*w2^(.25).*w3^(.25)  ~ .5*w1^(-.5).*w3^(.5)

            ~ .25*w1^(-.75).*w2^(.75)  ~ .25*w1^(-.75).*w2^(.5).*w3^(.25) ~ .25*w1^(-.75).*w2^(.25
).*w3^(.5)

            ~ .25*w1^(-.75).*w3^(.75));


   z2 = y.*(ones(n,1) ~ .25*w1^(.75).*w2^(-.75)        ~ .5*w1^(.5).*w2^(-.5)

            ~ .25*w1^(.5).*w2^(-.75).*w3^(.25) ~ .75*w1^(.25).*w2^(-.25 ) ~ .5*w1^(.25).*w2^(-
.5).*w3^(.25)

            ~ .25*w1^(.25).*w2^(-.75).*w3^(.5) ~ .75*w2^(-.25).*w3^(.25)  ~ .5*w2^(-.5).*w3^(.5)

            ~ .25*w2^(-.75).*w3^(.75));


   z3 = y.*(ones(n,1) ~ .25*w1^(.75).*w3^(-.75) ~ .25*w1^(.5).*w2^(.25).*w3^(-.75)

            ~ .5*w1^(.5).*w3^(-.5)   ~ .25*w1^(.25).*w2^(.5).*w3^(-.75)  ~
.5*w1^(.25).*w2^(.25).*w3^(-.5)

            ~ .75*w1^(.25).*w3^(-.25) ~ .25*w2^(.75).*w3^(-.75)        ~ .5*w2^(.5).*w3^(-.5)
```

```
        ~ .75*w2^(.25).*w3^(-.25));

  /* Construction of the AIM(2) Regressor for the grid */

  ze1 = ygrid.*(ones(ngrid,1) ~ .75*psi1^(-.25).*psi2^(.25) ~ .75*psi1^(-.25).*psi3^(.25)

        ~ .5*psi1^(-.5).*psi2^(.5)    ~ .5*psi1^(-.5).*psi2^(.25).*psi3^(.25)  ~ .5*psi1^(-.5).*psi3^(.5)

        ~ .25*psi1^(-.75).*psi2^(.75)  ~ .25*psi1^(-.75).*psi2^(.5).*psi3^(.25) ~ .25*psi1^(-
.75).*psi2^(.25 ).*psi3^(.5)

        ~ .25*psi1^(-.75).*psi3^(.75));

  ze2 = ygrid.*(ones(ngrid,1) ~ .25*psi1^(.75).*psi2^(-.75)   ~ .5*psi1^(.5).*psi2^(-.5)

        ~ .25*psi1^(.5).*psi2^(-.75).*psi3^(.25) ~ .75*psi1^(.25).*psi2^(-.25 ) ~
.5*psi1^(.25).*psi2^(-.5).*psi3^(.25)

        ~ .25*psi1^(.25).*psi2^(-.75).*psi3^(.5) ~ .75*psi2^(-.25).*psi3^(.25)  ~ .5*psi2^(-
.5).*psi3^(.5)

        ~ .25*psi2^(-.75).*psi3^(.75));

  ze3 = ygrid.*(ones(ngrid,1) ~ .25*psi1^(.75).*psi3^(-.75) ~ .25*psi1^(.5).*psi2^(.25).*psi3^(-.75)

        ~ .5*psi1^(.5).*psi3^(-.5)    ~ .25*psi1^(.25).*psi2^(.5).*psi3^(-.75)  ~
.5*psi1^(.25).*psi2^(.25).*psi3^(-.5)

        ~ .75*psi1^(.25).*psi3^(-.25) ~ .25*psi2^(.75).*psi3^(-.75)        ~ .5*psi2^(.5).*psi3^(-.5)

        ~ .75*psi2^(.25).*psi3^(-.25));

  /* Construction of the AIM(1) Regressor for the Evaluation at AESw*/

  AESz1 = AESy.*(ones(1,1) ~ .75*AESw1^(-.25).*AESw2^(.25) ~ .75*AESw1^(-.25).*AESw3^(.25)
```

```
~ .5*AESw1^(-.5).*AESw2^(.5)    ~ .5*AESw1^(-.5).*AESw2^(.25).*AESw3^(.25)  ~
.5*AESw1^(-.5).*AESw3^(.5)

~ .25*AESw1^(-.75).*AESw2^(.75)  ~ .25*AESw1^(-.75).*AESw2^(.5).*AESw3^(.25) ~
.25*AESw1^(-.75).*AESw2^(.25 ).*AESw3^(.5)

~ .25*AESw1^(-.75).*AESw3^(.75));


 AESz2 = AESy.*(ones(1,1) ~ .25*AESw1^(.75).*AESw2^(-.75)   ~ .5*AESw1^(.5).*AESw2^(-.5)

~ .25*AESw1^(.5).*AESw2^(-.75).*AESw3^(.25) ~ .75*AESw1^(.25).*AESw2^(-.25 ) ~
.5*AESw1^(.25).*AESw2^(-.5).*AESw3^(.25)

~ .25*AESw1^(.25).*AESw2^(-.75).*AESw3^(.5) ~ .75*AESw2^(-.25).*AESw3^(.25)  ~
.5*AESw2^(-.5).*AESw3^(.5)

~ .25*AESw2^(-.75).*AESw3^(.75));


 AESz3 = AESy.*(ones(1,1) ~ .25*AESw1^(.75).*AESw3^(-.75) ~
.25*AESw1^(.5).*AESw2^(.25).*AESw3^(-.75)

~ .5*AESw1^(.5).*AESw3^(-.5)    ~ .25*AESw1^(.25).*AESw2^(.5).*AESw3^(-.75)  ~
.5*AESw1^(.25).*AESw2^(.25).*AESw3^(-.5)

~ .75*AESw1^(.25).*AESw3^(-.25) ~ .25*AESw2^(.75).*AESw3^(-.75)         ~
.5*AESw2^(.5).*AESw3^(-.5)

~ .75*AESw2^(.25).*AESw3^(-.25));



 /* Define design matrix R for the restrictions RD*Beta = R of the stacked System (for AIM(2) only !! ) */


 RD = {

 0  1  0  0  0  0  0  0  0  0  0  -1 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0,

 0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  -1 0  0  0  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  0  0  0  0  0  0  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  0  0  0  0  0  0  0  0,
  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0,
  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0,
  0  0  0  0  0  0  1  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0,
  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0  0  0,
  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0,
  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0,
  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0,
  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0,
  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1};


  R = zeros(15,1);
endif;


/*****************************************************************************
******************

                    Definitions for the AIM(3)

*****************************************************************************
*****************/
if order == 3;
   print "********** AIM(3) **********";
   print;

     /* Define Regressors of the AIM(3) (see, Barnett, Geweke, Wolfe, 1991, p. 45) */
```

```
z1 = y.*(ones(n,1)              ~ .875/w1^.125.*w2^.125        ~ .875/w1^.125.*w3^.125 ~
    .750/w1^.250.*w2^.250       ~ .750/w1^.250.*w2^.125.*w3^.125 ~
    .750/w1^.250.*w3^.250       ~ .625/w1^.375.*w2^.375        ~
    .625/w1^.375.*w2^.250.*w3^.125 ~ .625/w1^.375.*w2^.125.*w3^.250 ~
    .625/w1^.375.*w3^.375       ~ .500/w1^.500.*w2^.500        ~
    .500/w1^.500.*w2^.375.*w3^.125 ~ .500/w1^.500.*w2^.250.*w3^.250 ~
    .500/w1^.500.*w2^.125.*w3^.375 ~ .500/w1^.500.*w3^.500        ~
    .375/w1^.625.*w2^.625       ~ .375/w1^.625.*w2^.500.*w3^.125 ~
    .375/w1^.625.*w2^.375.*w3^.250 ~ .375/w1^.625.*w2^.250.*w3^.375 ~
    .375/w1^.625.*w2^.125.*w3^.500 ~ .375/w1^.625.*w3^.625        ~
    .250/w1^.750.*w2^.750       ~ .250/w1^.750.*w2^.625.*w3^.125 ~
    .250/w1^.750.*w2^.500.*w3^.250 ~ .250/w1^.750.*w2^.375.*w3^.375 ~
    .250/w1^.750.*w2^.250.*w3^.500 ~ .250/w1^.750.*w2^.125.*w3^.625 ~
    .250/w1^.750.*w3^.750       ~ .125/w1^.875.*w2^.875        ~
    .125/w1^.875.*w2^.750.*w3^.125 ~ .125/w1^.875.*w2^.625.*w3^.250 ~
    .125/w1^.875.*w2^.500.*w3^.375 ~ .125/w1^.875.*w2^.375.*w3^.500 ~
    .125/w1^.875.*w2^.250.*w3^.625 ~ .125/w1^.875.*w2^.125.*w3^.750 ~
    .125/w1^.875.*w3^.875);

    z2 = y.*(ones(n,1)          ~ .125*w1^.875./w2^.875        ~ .250*w1^.750./w2^.750 ~
    .125*w1^.750./w2^.875.*w3^.125  ~ .375*w1^.625./w2^.625        ~
    .250*w1^.625./w2^.750.*w3^.125  ~ .125*w1^.625./w2^.875.*w3^.250 ~
    .500*w1^.500./w2^.500       ~ .375*w1^.500./w2^.625.*w3^.125 ~
    .250*w1^.500./w2^.750.*w3^.250  ~ .125*w1^.500./w2^.875.*w3^.375 ~
    .625*w1^.375./w2^.375       ~ .500*w1^.375./w2^.500.*w3^.125 ~
```

```
.375*w1^.375./w2^.625.*w3^.250  ~ .250*w1^.375./w2^.750.*w3^.375 ~
.125*w1^.375./w2^.875.*w3^.500  ~ .750*w1^.250./w2^.250         ~
.625*w1^.250./w2^.375.*w3^.125  ~ .500*w1^.250./w2^.500.*w3^.250 ~
.375*w1^.250./w2^.625.*w3^.375  ~ .250*w1^.250./w2^.750.*w3^.500 ~
.125*w1^.250./w2^.875.*w3^.625  ~ .875*w1^.125./w2^.125         ~
.750*w1^.125./w2^.250.*w3^.125  ~ .625*w1^.125./w2^.375.*w3^.250 ~
.500*w1^.125./w2^.500.*w3^.375  ~ .375*w1^.125./w2^.625.*w3^.500 ~
.250*w1^.125./w2^.750.*w3^.625  ~ .125*w1^.125./w2^.875.*w3^.750 ~
.875/w2^.125.*w3^.125           ~ .750/w2^.250.*w3^.250         ~
.625/w2^.375.*w3^.375           ~ .500/w2^.500.*w3^.500         ~
.375/w2^.625.*w3^.625           ~ .250/w2^.750.*w3^.750         ~
.125/w2^.875.*w3^.875);
```

```
z3 =   y.*(ones(n,1)            ~ .125*w1^.875./w3^.875         ~ .125*w1^.750.*w2^.125./w3^.875 ~
.250*w1^.750./w3^.750           ~ .125*w1^.625.*w2^.250./w3^.875 ~
.250*w1^.625.*w2^.125./w3^.750  ~ .375*w1^.625./w3^.625         ~
.125*w1^.500.*w2^.375./w3^.875  ~ .250*w1^.500.*w2^.250./w3^.750 ~
.375*w1^.500.*w2^.125./w3^.625  ~ .500*w1^.500./w3^.500         ~
.125*w1^.375.*w2^.500./w3^.875  ~ .250*w1^.375.*w2^.375./w3^.750 ~
.375*w1^.375.*w2^.250./w3^.625  ~ .500*w1^.375.*w2^.125./w3^.500 ~
.625*w1^.375./w3^.375           ~ .125*w1^.250.*w2^.625./w3^.875 ~
.250*w1^.250.*w2^.500./w3^.750  ~ .375*w1^.250.*w2^.375./w3^.625 ~
.500*w1^.250.*w2^.250./w3^.500  ~ .625*w1^.250.*w2^.125./w3^.375 ~
.750*w1^.250./w3^.250           ~ .125*w1^.125.*w2^.750./w3^.875 ~
```

```
    .250*w1^.125.*w2^.625./w3^.750   ~ .375*w1^.125.*w2^.500./w3^.625 ~

    .500*w1^.125.*w2^.375./w3^.500   ~ .625*w1^.125.*w2^.250./w3^.375 ~

    .750*w1^.125.*w2^.125./w3^.250   ~ .875*w1^.125./w3^.125         ~

    .125*w2^.875./w3^.875          ~ .250*w2^.750./w3^.750         ~

    .375*w2^.625./w3^.625          ~ .500*w2^.500./w3^.500         ~

    .625*w2^.375./w3^.375          ~ .750*w2^.250./w3^.250         ~

    .875*w2^.125./w3^.125);



    /* Construction of the AIM(3) Regressor for the grid */


    ze1 = ygrid.*(ones(ngrid,1)         ~

     .875/psi1^.125.*psi2^.125          ~ .875/psi1^.125.*psi3^.125          ~ .750/psi1^.250.*psi2^.250
~

     .750/psi1^.250.*psi2^.125.*psi3^.125 ~ .750/psi1^.250.*psi3^.250          ~
.625/psi1^.375.*psi2^.375          ~

     .625/psi1^.375.*psi2^.250.*psi3^.125 ~ .625/psi1^.375.*psi2^.125.*psi3^.250 ~
.625/psi1^.375.*psi3^.375          ~

     .500/psi1^.500.*psi2^.500          ~ .500/psi1^.500.*psi2^.375.*psi3^.125 ~
.500/psi1^.500.*psi2^.250.*psi3^.250 ~

     .500/psi1^.500.*psi2^.125.*psi3^.375 ~ .500/psi1^.500.*psi3^.500          ~
.375/psi1^.625.*psi2^.625          ~

     .375/psi1^.625.*psi2^.500.*psi3^.125 ~ .375/psi1^.625.*psi2^.375.*psi3^.250 ~
.375/psi1^.625.*psi2^.250.*psi3^.375 ~

     .375/psi1^.625.*psi2^.125.*psi3^.500 ~ .375/psi1^.625.*psi3^.625          ~
.250/psi1^.750.*psi2^.750          ~

     .250/psi1^.750.*psi2^.625.*psi3^.125 ~ .250/psi1^.750.*psi2^.500.*psi3^.250 ~
.250/psi1^.750.*psi2^.375.*psi3^.375 ~
```

```
    .250/psi1^.750.*psi2^.250.*psi3^.500 ~ .250/psi1^.750.*psi2^.125.*psi3^.625 ~
.250/psi1^.750.*psi3^.750          ~

    .125/psi1^.875.*psi2^.875          ~ .125/psi1^.875.*psi2^.750.*psi3^.125 ~
.125/psi1^.875.*psi2^.625.*psi3^.250 ~

    .125/psi1^.875.*psi2^.500.*psi3^.375 ~ .125/psi1^.875.*psi2^.375.*psi3^.500 ~
.125/psi1^.875.*psi2^.250.*psi3^.625 ~

    .125/psi1^.875.*psi2^.125.*psi3^.750 ~ .125/psi1^.875.*psi3^.875);



    ze2 = ygrid.*(ones(ngrid,1)        ~

    .125*psi1^.875./psi2^.875          ~ .250*psi1^.750./psi2^.750          ~
.125*psi1^.750./psi2^.875.*psi3^.125 ~

    .375*psi1^.625./psi2^.625          ~ .250*psi1^.625./psi2^.750.*psi3^.125 ~
.125*psi1^.625./psi2^.875.*psi3^.250 ~

    .500*psi1^.500./psi2^.500          ~ .375*psi1^.500./psi2^.625.*psi3^.125 ~
.250*psi1^.500./psi2^.750.*psi3^.250 ~

    .125*psi1^.500./psi2^.875.*psi3^.375 ~ .625*psi1^.375./psi2^.375          ~
.500*psi1^.375./psi2^.500.*psi3^.125 ~

    .375*psi1^.375./psi2^.625.*psi3^.250 ~ .250*psi1^.375./psi2^.750.*psi3^.375 ~
.125*psi1^.375./psi2^.875.*psi3^.500 ~

    .750*psi1^.250./psi2^.250          ~ .625*psi1^.250./psi2^.375.*psi3^.125 ~
.500*psi1^.250./psi2^.500.*psi3^.250 ~

    .375*psi1^.250./psi2^.625.*psi3^.375 ~ .250*psi1^.250./psi2^.750.*psi3^.500 ~
.125*psi1^.250./psi2^.875.*psi3^.625 ~

    .875*psi1^.125./psi2^.125          ~ .750*psi1^.125./psi2^.250.*psi3^.125 ~
.625*psi1^.125./psi2^.375.*psi3^.250 ~

    .500*psi1^.125./psi2^.500.*psi3^.375 ~ .375*psi1^.125./psi2^.625.*psi3^.500 ~
.250*psi1^.125./psi2^.750.*psi3^.625 ~

    .125*psi1^.125./psi2^.875.*psi3^.750 ~ .875/psi2^.125.*psi3^.125          ~
.750/psi2^.250.*psi3^.250          ~
```

```
    .625/psi2^.375.*psi3^.375          ~ .500/psi2^.500.*psi3^.500          ~ .375/psi2^.625.*psi3^.625
~

    .250/psi2^.750.*psi3^.750          ~ .125/psi2^.875.*psi3^.875);



   ze3 =   ygrid.*(ones(ngrid,1)       ~

    .125*psi1^.875./psi3^.875          ~ .125*psi1^.750.*psi2^.125./psi3^.875 ~
.250*psi1^.750./psi3^.750 ~

    .125*psi1^.625.*psi2^.250./psi3^.875 ~ .250*psi1^.625.*psi2^.125./psi3^.750 ~
.375*psi1^.625./psi3^.625 ~

    .125*psi1^.500.*psi2^.375./psi3^.875 ~ .250*psi1^.500.*psi2^.250./psi3^.750 ~
.375*psi1^.500.*psi2^.125./psi3^.625 ~

    .500*psi1^.500./psi3^.500          ~ .125*psi1^.375.*psi2^.500./psi3^.875 ~
.250*psi1^.375.*psi2^.375./psi3^.750 ~

    .375*psi1^.375.*psi2^.250./psi3^.625 ~ .500*psi1^.375.*psi2^.125./psi3^.500 ~
.625*psi1^.375./psi3^.375 ~

    .125*psi1^.250.*psi2^.625./psi3^.875 ~ .250*psi1^.250.*psi2^.500./psi3^.750 ~
.375*psi1^.250.*psi2^.375./psi3^.625 ~

    .500*psi1^.250.*psi2^.250./psi3^.500 ~ .625*psi1^.250.*psi2^.125./psi3^.375 ~
.750*psi1^.250./psi3^.250 ~

    .125*psi1^.125.*psi2^.750./psi3^.875 ~ .250*psi1^.125.*psi2^.625./psi3^.750 ~
.375*psi1^.125.*psi2^.500./psi3^.625 ~

    .500*psi1^.125.*psi2^.375./psi3^.500 ~ .625*psi1^.125.*psi2^.250./psi3^.375 ~
.750*psi1^.125.*psi2^.125./psi3^.250 ~

    .875*psi1^.125./psi3^.125          ~ .125*psi2^.875./psi3^.875          ~ .250*psi2^.750./psi3^.750 ~

    .375*psi2^.625./psi3^.625          ~ .500*psi2^.500./psi3^.500          ~ .625*psi2^.375./psi3^.375 ~

    .750*psi2^.250./psi3^.250          ~ .875*psi2^.125./psi3^.125);


  /* Construction of the AIM(1) Regressor for the Evaluation at AESw*/
```

```
AESz1 = AESy.*(ones(1,1)        ~

    .875/AESw1^.125.*AESw2^.125         ~ .875/AESw1^.125.*AESw3^.125         ~
.750/AESw1^.250.*AESw2^.250         ~

    .750/AESw1^.250.*AESw2^.125.*AESw3^.125 ~ .750/AESw1^.250.*AESw3^.250         ~
.625/AESw1^.375.*AESw2^.375         ~

    .625/AESw1^.375.*AESw2^.250.*AESw3^.125 ~ .625/AESw1^.375.*AESw2^.125.*AESw3^.250 ~
.625/AESw1^.375.*AESw3^.375         ~

    .500/AESw1^.500.*AESw2^.500         ~ .500/AESw1^.500.*AESw2^.375.*AESw3^.125 ~
.500/AESw1^.500.*AESw2^.250.*AESw3^.250 ~

    .500/AESw1^.500.*AESw2^.125.*AESw3^.375 ~ .500/AESw1^.500.*AESw3^.500         ~
.375/AESw1^.625.*AESw2^.625         ~

    .375/AESw1^.625.*AESw2^.500.*AESw3^.125 ~ .375/AESw1^.625.*AESw2^.375.*AESw3^.250 ~
.375/AESw1^.625.*AESw2^.250.*AESw3^.375 ~

    .375/AESw1^.625.*AESw2^.125.*AESw3^.500 ~ .375/AESw1^.625.*AESw3^.625         ~
.250/AESw1^.750.*AESw2^.750         ~

    .250/AESw1^.750.*AESw2^.625.*AESw3^.125 ~ .250/AESw1^.750.*AESw2^.500.*AESw3^.250 ~
.250/AESw1^.750.*AESw2^.375.*AESw3^.375 ~

    .250/AESw1^.750.*AESw2^.250.*AESw3^.500 ~ .250/AESw1^.750.*AESw2^.125.*AESw3^.625 ~
.250/AESw1^.750.*AESw3^.750         ~

    .125/AESw1^.875.*AESw2^.875         ~ .125/AESw1^.875.*AESw2^.750.*AESw3^.125 ~
.125/AESw1^.875.*AESw2^.625.*AESw3^.250 ~

    .125/AESw1^.875.*AESw2^.500.*AESw3^.375 ~ .125/AESw1^.875.*AESw2^.375.*AESw3^.500 ~
.125/AESw1^.875.*AESw2^.250.*AESw3^.625 ~

    .125/AESw1^.875.*AESw2^.125.*AESw3^.750 ~ .125/AESw1^.875.*AESw3^.875);


AESz2 = AESy.*(ones(1,1)        ~

    .125*AESw1^.875./AESw2^.875         ~ .250*AESw1^.750./AESw2^.750         ~
.125*AESw1^.750./AESw2^.875.*AESw3^.125 ~
```

.375*AESw1^.625./AESw2^.625 ~ .250*AESw1^.625./AESw2^.750.*AESw3^.125 ~
.125*AESw1^.625./AESw2^.875.*AESw3^.250 ~

.500*AESw1^.500./AESw2^.500 ~ .375*AESw1^.500./AESw2^.625.*AESw3^.125 ~
.250*AESw1^.500./AESw2^.750.*AESw3^.250 ~

.125*AESw1^.500./AESw2^.875.*AESw3^.375 ~ .625*AESw1^.375./AESw2^.375 ~
.500*AESw1^.375./AESw2^.500.*AESw3^.125 ~

.375*AESw1^.375./AESw2^.625.*AESw3^.250 ~ .250*AESw1^.375./AESw2^.750.*AESw3^.375 ~
.125*AESw1^.375./AESw2^.875.*AESw3^.500 ~

.750*AESw1^.250./AESw2^.250 ~ .625*AESw1^.250./AESw2^.375.*AESw3^.125 ~
.500*AESw1^.250./AESw2^.500.*AESw3^.250 ~

.375*AESw1^.250./AESw2^.625.*AESw3^.375 ~ .250*AESw1^.250./AESw2^.750.*AESw3^.500 ~
.125*AESw1^.250./AESw2^.875.*AESw3^.625 ~

.875*AESw1^.125./AESw2^.125 ~ .750*AESw1^.125./AESw2^.250.*AESw3^.125 ~
.625*AESw1^.125./AESw2^.375.*AESw3^.250 ~

.500*AESw1^.125./AESw2^.500.*AESw3^.375 ~ .375*AESw1^.125./AESw2^.625.*AESw3^.500 ~
.250*AESw1^.125./AESw2^.750.*AESw3^.625 ~

.125*AESw1^.125./AESw2^.875.*AESw3^.750 ~ .875/AESw2^.125.*AESw3^.125 ~
.750/AESw2^.250.*AESw3^.250 ~

.625/AESw2^.375.*AESw3^.375 ~ .500/AESw2^.500.*AESw3^.500 ~
.375/AESw2^.625.*AESw3^.625 ~

.250/AESw2^.750.*AESw3^.750 ~ .125/AESw2^.875.*AESw3^.875);

AESz3 = AESy.*(ones(1,1) ~

.125*AESw1^.875./AESw3^.875 ~ .125*AESw1^.750.*AESw2^.125./AESw3^.875 ~
.250*AESw1^.750./AESw3^.750 ~

.125*AESw1^.625.*AESw2^.250./AESw3^.875 ~ .250*AESw1^.625.*AESw2^.125./AESw3^.750 ~
.375*AESw1^.625./AESw3^.625 ~

.125*AESw1^.500.*AESw2^.375./AESw3^.875 ~ .250*AESw1^.500.*AESw2^.250./AESw3^.750 ~
.375*AESw1^.500.*AESw2^.125./AESw3^.625 ~

```
    .500*AESw1^.500./AESw3^.500         ~ .125*AESw1^.375.*AESw2^.500./AESw3^.875 ~
.250*AESw1^.375.*AESw2^.375./AESw3^.750 ~

    .375*AESw1^.375.*AESw2^.250./AESw3^.625 ~ .500*AESw1^.375.*AESw2^.125./AESw3^.500 ~
.625*AESw1^.375./AESw3^.375 ~

    .125*AESw1^.250.*AESw2^.625./AESw3^.875 ~ .250*AESw1^.250.*AESw2^.500./AESw3^.750 ~
.375*AESw1^.250.*AESw2^.375./AESw3^.625 ~

    .500*AESw1^.250.*AESw2^.250./AESw3^.500 ~ .625*AESw1^.250.*AESw2^.125./AESw3^.375 ~
.750*AESw1^.250./AESw3^.250 ~

    .125*AESw1^.125.*AESw2^.750./AESw3^.875 ~ .250*AESw1^.125.*AESw2^.625./AESw3^.750 ~
.375*AESw1^.125.*AESw2^.500./AESw3^.625 ~

    .500*AESw1^.125.*AESw2^.375./AESw3^.500 ~ .625*AESw1^.125.*AESw2^.250./AESw3^.375 ~
.750*AESw1^.125.*AESw2^.125./AESw3^.250 ~

    .875*AESw1^.125./AESw3^.125         ~ .125*AESw2^.875./AESw3^.875         ~
.250*AESw2^.750./AESw3^.750 ~

    .375*AESw2^.625./AESw3^.625         ~ .500*AESw2^.500./AESw3^.500         ~
.625*AESw2^.375./AESw3^.375 ~

    .750*AESw2^.250./AESw3^.250         ~ .875*AESw2^.125./AESw3^.125);




RD = {

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,
```

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0 0 0 0 0 0 0

,

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  -1 0  0  0  0  0  0  0  0  0
,
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  -1 0  0  0  0  0  0  0
0};


R = zeros(63,1);


endif;




retp(z1,z2,z3,ze1,ze2,ze3,AESz1,AESz2,AESz3,RD,r);

endp;









/* Procedure for Constraint Optimization of the AIM

   possibly with or without regularity depending on flag: concavity */

proc pcoestaim(order);

      local beta_ols1,beta_ols2,beta_ols3,res1,res2,res3,resols,

           start,betai,func,grad,ret,sigma,omega,zs;
```

```
if order <= 2;

  beta_ols1 = invpd(z1'z1)*z1'x1;

  beta_ols2 = invpd(z2'z2)*z2'x2;

  beta_ols3 = invpd(z3'z3)*z3'x3;

endif;

if order == 3;

    @ results from Eviews: sim 1 -> SUR; sim 2 and 3 -> ones(108,1)  @

  if sim == 1;

    beta ={-247.67  443.02   443.00   -178.08  -343.08  -178.04  -416.77  -40.45   -40.57   -416.79
854.46   104.23   -281.91  104.25   854.49   -574.15  -15.50   313.73   313.97   -15.47   -574.26  167.27   -
556.88  234.35   102.22   234.27   -557.03  167.44   107.79   468.22   -823.28  286.69   286.69   -823.30
468.35   107.68   -791.37  443.02   -178.08  -343.08  -416.77  -40.45   -40.57   854.46   104.23   -281.91
104.25   -574.15  -15.50   313.73   313.97   -15.47   167.27   -556.88  234.35   102.22   234.27   -557.03
107.79   468.22   -823.28  286.69   286.69   -823.30  468.35   3074.98  -4357.94 1467.55  1235.50
1467.19  -4357.56 3074.79  -791.31  443.00   -343.08  -178.04  -40.45   -40.57   -416.79  104.23   -281.91
104.25   854.49   -15.50   313.73   313.97   -15.47   -574.26  -556.88  234.35   102.22   234.27   -557.03
167.44   468.22   -823.28  286.69   286.69   -823.30  468.35   107.68   3074.98  -4357.94 1467.55  1235.50
1467.19  -4357.56 3074.79};

    @ beta = ones(108,1)'; @

    beta = beta';

  endif;

  if sim >= 2;

    beta=ones(108,1);

  endif;

  beta_ols1 = beta[1:cs];

  beta_ols2 = beta[37:2*cs];

  beta_ols3 = beta[73:3*cs];

endif;
```

```
res1   = x1-z1*beta_ols1;
res2   = x2-z2*beta_ols2;
res3   = x3-z3*beta_ols3;
resols = res1~res2~res3;

/* Define matrices for the stacked System */
z  =  (z1~zeros(n,2*cs))|
      (zeros(n,cs)~z2~zeros(n,cs))|
      (zeros(n,2*cs)~z3);
vx =  vec(x);

/* Estimation of Covariance Matrix, whereby omega = sigma(Kronecker)eye(n) */
invsigma = invpd(resols'resols/(n-cs));
invomega = invsigma.*.eye(n);
p  = chol(invsigma).*.eye(n);
py = p*vx;
px = p*z;
/* SUR Estimation with CO */
if concavity == 1;
    @ Nonnegative Constraint for Global Concavity C*beta >= D @
    print /flush "Method: Constrained Optimization: Symmetry &";
    print /flush "Nonnegative Constraints for Global Concavity: I*beta >= 0";
    _co_C = eye(3*cs);
    _co_D = zeros(3*cs,1);
else;
```

```
    print /flush "Method: Constrained Optimization for Symmetry";
endif;
@ _co_Algorithm: 2 --> Davidon-Fletcher-Powell; 3 --> Newton-Raphson method @
if order == 3;
 _co_Algorithm     = 3;
else;
 _co_Algorithm     = 3;
endif;
_co_FeasibleTest    = 0;
string _co_Options  = {none};   @ to surpress the output @
_co_EqProc         = &eqpaim;
_co_EqJacobian      = &eqjaim;
_co_GradProc       = &gp;
_co_HessProc       = &hsp;
_co_MaxIters       = 10000;
start              = beta_ols1|beta_ols2|beta_ols3;
if order == 3;
  start          = beta;
endif;
print /flush "start iteration";
{beta,func,grad,ret}      = co(&fct,start);
@ call coprt(beta,f,g,ret); @
betai = beta;
for i(2,1000,1);
   res1   = x1-z1*beta[1:cs];
```

```
        res2   = x2-z2*beta[(cs+1):(2*cs)];
        res3   = x3-z3*beta[((2*cs)+1):(3*cs)];
        ressur = res1~res2~res3;

        sigma  = ressur'ressur/(n-cs);
        omega = sigma.*.eye(n);
        p  = chol(invpd(omega));
        py = p*vx;
        px = p*z;
        _co_MaxIters       = 10000;
        start              = beta;
        print /flush "start " i " iteration";
        {beta,func,grad,ret}     = co(&fct,start);
       @ call coprt(beta,f,g,ret); @
        if abs(betai-beta) < breakn;
              print /flush "converged at iteration" i;
              break;
        endif;
        betai = beta;
      endfor;
      retp(beta);
endp;
```

```
psic = concavg.*psi;
psim = monovg.*psi;

psic = sortc(psic,1);
for i(1,ngrid,1);
   if psic[i,1] /= 0;
   trimi = i;
   break;
   else; trimi = 8000;
   endif;
endfor;
psic = trim(psic,trimi-1,0);
/* print /flush "psic: " psic; */

psim = sortc(psim,1);
for i(1,ngrid,1);
   if psim[i,1] /= 0;
   trimi = i;
   break;
   else; trimi = 8000;
   endif;
endfor;
```

```
psim = trim(psim,trimi-1,0);

/* print /flush "psim: " psim; */



stopnow: print /flush "ende";

dateend = date;

totime = ethsec(datestart,dateend)/60/100;

print "duration of the estimation in minutes" totime;

load bexper1;

load bexper2;

saveall aims;


output off;
```

Section 3: Procedure pAES.g

```
@ PROCEDURE for the Definition of the Allen Elasticities of Substitution evaluated at AESw @


proc pAES(beta);
  local AESh11, AESh12, AESh13, AESh22, AESh23, AESh33, AEScosthat, AESx1hat,AESx2hat,AESx3hat,
      AESxhat, AEShat, AESh;
  if order == 1;
    AESh11 = AESy*(-1/4*beta[2]/AESw1^(3/2)*AESw2^(1/2)-1/4*beta[3]/AESw1^(3/2)*AESw3^(1/2));
    AESh12 = 1/4*AESy*beta[2]/AESw1^(1/2)/AESw2^(1/2);
    AESh13 = 1/4*AESy*beta[3]/AESw1^(1/2)/AESw3^(1/2);
    AESh22 = AESy*(-1/4*beta[2]*AESw1^(1/2)/AESw2^(3/2)-1/4*beta[6]/AESw2^(3/2)*AESw3^(1/2));
    AESh23 = 1/4*AESy*beta[6]/AESw2^(1/2)/AESw3^(1/2);
    AESh33 = AESy*(-1/4*beta[3]*AESw1^(1/2)/AESw3^(3/2)-1/4*beta[6]*AESw2^(1/2)/AESw3^(3/2));
    AESh   = AESh11~AESh12~AESh13 |
             AESh12~AESh22~AESh23 |
             AESh13~AESh23~AESh33 ;
    AEScosthat = AESy*(beta[1]*AESw1  + beta[4]*AESw2 + beta[7]*AESw3
    + beta[2]*AESw1^(1/2)*AESw2^(1/2) + beta[3]*AESw1^(1/2)*AESw3^(1/2)
    + beta[6]*AESw2^(1/2)*AESw3^(1/2));
    AESx1hat = AESz1*beta[1:cs];
    AESx2hat = AESz2*beta[(cs+1):(2*cs)];
    AESx3hat = AESz3*beta[((2*cs)+1):(3*cs)];
    AESxhat  = AESx1hat~AESx2hat~AESx3hat;
    @ Formula C[i,j]*C/(C[i]*C[j])  in e.g. Paolo Bertoletti:
```

```
      Elasticity of Substitution, one more time: gross versus net, direct versus inverse measures"
      p. 6 (paper found in the internet) @
    AEShat   = AESh.*AEScosthat./(AESxhat'AESxhat);
  endif;
  if order == 2;
    AESh11 = AESy * (-.1875 *beta[2]* AESw2^.25 / AESw1^1.25-.1875 *beta[3]* AESw3^.25 /
AESw1^1.25-1 /
          4 * beta[4] * AESw2^(1 / 2) / AESw1^(3 / 2)-.25 * beta[5] * AESw2^.25 * AESw3^.25 /
          AESw1^1.5-.25 * beta[6] * AESw3^.5 / AESw1^1.5-.1875 * beta[7] * AESw2^.75 / AESw1^1.75-
.1875 *
          beta[8] * AESw2^.5 * AESw3^.25 / AESw1^1.75-.1875 * beta[9] * AESw2^.25 * AESw3^.5 /
          AESw1^1.75-.1875 * beta[10] * AESw3^.75 / AESw1^1.75);
    AESh12 = AESy * (.1875 *beta[2]/ AESw1^.25 / AESw2^.75 + 1 / 4 * beta[4] / AESw1^(1 / 2) /
AESw2^(1 / 2)
          + .125 * beta[5] * AESw3^.25 / AESw1^.5 / AESw2^.75 + .1875 * beta[7] / AESw1^.75 /
AESw2^.25 +
          .125 * beta[8] * AESw3^.25 / AESw1^.75 / AESw2^.5 + .625e-1 * beta[9] * AESw3^.5
          / AESw1^.75 / AESw2^.75);
    AESh13 = AESy * (.1875 *beta[3]/ AESw1^.25 / AESw3^.75 + .125 * beta[5] * AESw2^.25 / AESw1^.5
/
          AESw3^.75 + .25 * beta[6] / AESw1^.5 / AESw3^.5 + .625e-1 * beta[8] * AESw2^.5 / AESw1^.75
          / AESw3^.75 + .125 * beta[9] * AESw2^.25 / AESw1^.75 / AESw3^.5 + .1875 * beta[10] /
          AESw1^.75 / AESw3^.25);
    AESh22 = AESy * (-.1875 *beta[2]* AESw1^.75 / AESw2^1.75-1 / 4 * beta[4] * AESw1^(1 / 2) /
          AESw2^(3 / 2)-.1875 * beta[5] * AESw1^.5 * AESw3^.25 / AESw2^1.75-.1875 * beta[7]
          * AESw1^.25 / AESw2^1.25-.25 * beta[8] * AESw1^.25 * AESw3^.25 / AESw2^1.5-.1875 *
beta[9] *
```

```
        AESw1^.25 * AESw3^.5 / AESw2^1.75-.1875 * beta[18] * AESw3^.25 / AESw2^1.25-.25 *
beta[19]

        * AESw3^.5 / AESw2^1.5-.1875 * beta[20] * AESw3^.75 / AESw2^1.75);

    AESh23 = AESy * (.625e-1 * beta[5] * AESw1^.5 / AESw2^.75 / AESw3^.75 + .125 * beta[8] *
AESw1^.25

        / AESw2^.5 / AESw3^.75 + .125 * beta[9] * AESw1^.25 / AESw2^.75 / AESw3^.5 + .1875 *
beta[18]

        / AESw2^.25 / AESw3^.75 + .25 * beta[19] / AESw2^.5 / AESw3^.5 + .1875 * beta[20] /
AESw2^.75

        / AESw3^.25);

    AESh33 = AESy * (-.1875 *beta[3]* AESw1^.75 / AESw3^1.75-.1875 * beta[5] * AESw1^.5 *
AESw2^.25 /

        AESw3^1.75-.25 * beta[6] * AESw1^.5 / AESw3^1.5-.1875 * beta[8] * AESw1^.25 * AESw2^.5 /

        AESw3^1.75-.25 * beta[9] * AESw1^.25 * AESw2^.25 / AESw3^1.5-.1875 * beta[10] *
AESw1^.25

        / AESw3^1.25-.1875 * beta[18] * AESw2^.75 / AESw3^1.75-.25 * beta[19] * AESw2^.5 /

        AESw3^1.5-.1875 * beta[20] * AESw2^.25 / AESw3^1.25);

    AESh = AESh11~AESh12~AESh13 |

        AESh12~AESh22~AESh23 |

        AESh13~AESh23~AESh33 ;

    AEScosthat = AESy*(

        beta[1]  *AESw1 + beta[11]*AESw2 + beta[21]*AESw3

       + beta[2 ]*AESw1^.75*AESw2^.25

       + beta[3 ]*AESw1^.75*AESw3^.25

       + beta[4 ]*AESw1^.5 *AESw2^.5

       + beta[5 ]*AESw1^.5 *AESw2^.25 *AESw3^.25

       + beta[6 ]*AESw1^.5 *AESw3^.5
```

```
      + beta[7 ]*AESw1^.25*AESw2^.75
      + beta[8 ]*AESw1^.25*AESw2^.5  *AESw3^.25
      + beta[9 ]*AESw1^.25*AESw2^.25 *AESw3^.5
      + beta[10]*AESw1^.25*AESw3^.75
      + beta[18]*AESw2^.75*AESw3^.25
      + beta[19]*AESw2^.5 *AESw3^.5
      + beta[20]*AESw2^.25*AESw3^.75);
  AESx1hat = AESz1*beta[1:cs];
  AESx2hat = AESz2*beta[(cs+1):(2*cs)];
  AESx3hat = AESz3*beta[((2*cs)+1):(3*cs)];
  AESxhat  = AESx1hat~AESx2hat~AESx3hat;
  @ Formula C[i,j]*C/(C[i]*C[j])  in e.g. Paolo Bertoletti:
    Elasticity of Substitution, one more time: gross versus net, direct versus inverse measures"
    p. 6 (paper found in the internet) @
  AEShat   = AESh.*AEScosthat./(AESxhat'AESxhat);
endif;
if order == 3;
  AESh11 = AESy*(
      -.109375*beta[5]/AESw1^1.125*AESw3^.125
      -.250000*beta[17]/AESw1^1.500*AESw3^.500
      -.187500*beta[8]/AESw1^1.250*AESw3^.250
      -.109375*beta[4]/AESw1^1.125*AESw2^.125
      -.234375*beta[9]/AESw1^1.375*AESw2^.375
      -.250000*beta[13]/AESw1^1.500*AESw2^.500
      -.109375*beta[38]/AESw1^1.875*AESw3^.875
```

-.187500*beta[30]/AESw1^1.750*AESw3^.750

-.109375*beta[31]/AESw1^1.875*AESw2^.875

-.234375*beta[12]/AESw1^1.375*AESw3^.375

-.234375*beta[18]/AESw1^1.625*AESw2^.625

-.187500*beta[24]/AESw1^1.750*AESw2^.750

-.234375*beta[23]/AESw1^1.625*AESw3^.625

-.187500*beta[6]/AESw1^1.250*AESw2^.250

-.187500*beta[7]/AESw1^1.250*AESw2^.125*AESw3^.125

-.234375*beta[10]/AESw1^1.375*AESw2^.250*AESw3^.125

-.234375*beta[11]/AESw1^1.375*AESw2^.125*AESw3^.250

-.250000*beta[14]/AESw1^1.500*AESw2^.375*AESw3^.125

-.250000*beta[15]/AESw1^1.500*AESw2^.250*AESw3^.250

-.250000*beta[16]/AESw1^1.500*AESw2^.125*AESw3^.375

-.234375*beta[19]/AESw1^1.625*AESw2^.500*AESw3^.125

-.234375*beta[20]/AESw1^1.625*AESw2^.375*AESw3^.250

-.234375*beta[21]/AESw1^1.625*AESw2^.250*AESw3^.375

-.234375*beta[22]/AESw1^1.625*AESw2^.125*AESw3^.500

-.187500*beta[25]/AESw1^1.750*AESw2^.625*AESw3^.125

-.187500*beta[26]/AESw1^1.750*AESw2^.500*AESw3^.250

-.187500*beta[27]/AESw1^1.750*AESw2^.375*AESw3^.375

-.187500*beta[28]/AESw1^1.750*AESw2^.250*AESw3^.500

-.187500*beta[29]/AESw1^1.750*AESw2^.125*AESw3^.625

-.109375*beta[32]/AESw1^1.875*AESw2^.750*AESw3^.125

-.109375*beta[33]/AESw1^1.875*AESw2^.625*AESw3^.250

-.109375*beta[34]/AESw1^1.875*AESw2^.500*AESw3^.375

```
    -.109375*beta[35]/AESw1^1.875*AESw2^.375*AESw3^.500
    -.109375*beta[36]/AESw1^1.875*AESw2^.250*AESw3^.625
    -.109375*beta[37]/AESw1^1.875*AESw2^.125*AESw3^.750);
AESh12 = AESy*(.250000*beta[13]/AESw1^.500/AESw2^.500 +
    .234375*beta[18]/AESw1^.625/AESw2^.375 +
    .109375*beta[31]/AESw1^.875/AESw2^.125 +
    .187500*beta[6]/AESw1^.250/AESw2^.750 +
    .187500*beta[24]/AESw1^.750/AESw2^.250 +
    .234375*beta[9]/AESw1^.375/AESw2^.625 +
    .109375*beta[4]/AESw1^.125/AESw2^.875 +
    .093750*beta[7]/AESw1^.250/AESw2^.875*AESw3^.125 +
    .156250*beta[10]/AESw1^.375/AESw2^.750*AESw3^.125 +
    .078125*beta[11]/AESw1^.375/AESw2^.875*AESw3^.250 +
    .187500*beta[14]/AESw1^.500/AESw2^.625*AESw3^.125 +
    .125000*beta[15]/AESw1^.500/AESw2^.750*AESw3^.250 +
    .062500*beta[16]/AESw1^.500/AESw2^.875*AESw3^.375 +
    .187500*beta[19]/AESw1^.625/AESw2^.500*AESw3^.125 +
    .140625*beta[20]/AESw1^.625/AESw2^.625*AESw3^.250 +
    .093750*beta[21]/AESw1^.625/AESw2^.750*AESw3^.375 +
    .046875*beta[22]/AESw1^.625/AESw2^.875*AESw3^.500 +
    .156250*beta[25]/AESw1^.750/AESw2^.375*AESw3^.125 +
    .125000*beta[26]/AESw1^.750/AESw2^.500*AESw3^.250 +
    .093750*beta[27]/AESw1^.750/AESw2^.625*AESw3^.375 +
    .062500*beta[28]/AESw1^.750/AESw2^.750*AESw3^.500 +
    .031250*beta[29]/AESw1^.750/AESw2^.875*AESw3^.625 +
```

```
    .093750*beta[32]/AESw1^.875/AESw2^.250*AESw3^.125 +
    .078125*beta[33]/AESw1^.875/AESw2^.375*AESw3^.250 +
    .062500*beta[34]/AESw1^.875/AESw2^.500*AESw3^.375 +
    .046875*beta[35]/AESw1^.875/AESw2^.625*AESw3^.500 +
    .031250*beta[36]/AESw1^.875/AESw2^.750*AESw3^.625 +
    .015625*beta[37]/AESw1^.875/AESw2^.875*AESw3^.750);
AESh13 = AESy*(.234375*beta[12]/AESw1^.375/AESw3^.625
    + .250000*beta[17]/AESw1^.500/AESw3^.500
    + .234375*beta[23]/AESw1^.625/AESw3^.375
    + .109375*beta[38]/AESw1^.875/AESw3^.125
    + .109375*beta[5]/AESw1^.125/AESw3^.875
    + .187500*beta[8]/AESw1^.250/AESw3^.750
    + .187500*beta[30]/AESw1^.750/AESw3^.250
    + .93750e-1*beta[7]/AESw1^.250*AESw2^.125/AESw3^.875
    + .78125e-1*beta[10]/AESw1^.375*AESw2^.250/AESw3^.875
    + .156250*beta[11]/AESw1^.375*AESw2^.125/AESw3^.750
    + .62500e-1*beta[14]/AESw1^.500*AESw2^.375/AESw3^.875
    + .125000*beta[15]/AESw1^.500*AESw2^.250/AESw3^.750
    + .187500*beta[16]/AESw1^.500*AESw2^.125/AESw3^.625
    + .46875e-1*beta[19]/AESw1^.625*AESw2^.500/AESw3^.875
    + .93750e-1*beta[20]/AESw1^.625*AESw2^.375/AESw3^.750
    + .140625*beta[21]/AESw1^.625*AESw2^.250/AESw3^.625
    + .187500*beta[22]/AESw1^.625*AESw2^.125/AESw3^.500
    + .31250e-1*beta[25]/AESw1^.750*AESw2^.625/AESw3^.875
    + .62500e-1*beta[26]/AESw1^.750*AESw2^.500/AESw3^.750
```

```
    + .93750e-1*beta[27]/AESw1^.750*AESw2^.375/AESw3^.625
    + .125000*beta[28]/AESw1^.750*AESw2^.250/AESw3^.500
    + .156250*beta[29]/AESw1^.750*AESw2^.125/AESw3^.375
    + .15625e-1*beta[32]/AESw1^.875*AESw2^.750/AESw3^.875
    + .31250e-1*beta[33]/AESw1^.875*AESw2^.625/AESw3^.750
    + .46875e-1*beta[34]/AESw1^.875*AESw2^.500/AESw3^.625
    + .62500e-1*beta[35]/AESw1^.875*AESw2^.375/AESw3^.500
    + .78125e-1*beta[36]/AESw1^.875*AESw2^.250/AESw3^.375
    + .93750e-1*beta[37]/AESw1^.875*AESw2^.125/AESw3^.250);
AESh22 = AESy*(
   - .234375*beta[18]*AESw1^.375/AESw2^1.375
   - .187500*beta[44]/AESw2^1.750*AESw3^.750
   - .109375*beta[31]*AESw1^.125/AESw2^1.125
   - .250000*beta[42]/AESw2^1.500*AESw3^.500
   - .109375*beta[45]/AESw2^1.875*AESw3^.875
   - .234375*beta[9]*AESw1^.625/AESw2^1.625
   - .109375*beta[39]/AESw2^1.125*AESw3^.125
   - .234375*beta[41]/AESw2^1.375*AESw3^.375
   - .187500*beta[6]*AESw1^.750/AESw2^1.750
   - .234375*beta[43]/AESw2^1.625*AESw3^.625
   - .187500*beta[24]*AESw1^.250/AESw2^1.250
   - .250000*beta[13]*AESw1^.500/AESw2^1.500
   - .187500*beta[40]/AESw2^1.250*AESw3^.250
   - .109375*beta[4]*AESw1^.875/AESw2^1.875
   - .109375*beta[7]*AESw1^.750/AESw2^1.875*AESw3^.125
```

```
    - .187500*beta[10]*AESw1^.625/AESw2^1.750*AESw3^.125
    - .109375*beta[11]*AESw1^.625/AESw2^1.875*AESw3^.250
    - .234375*beta[14]*AESw1^.500/AESw2^1.625*AESw3^.125
    - .187500*beta[15]*AESw1^.500/AESw2^1.750*AESw3^.250
    - .109375*beta[16]*AESw1^.500/AESw2^1.875*AESw3^.375
    - .250000*beta[19]*AESw1^.375/AESw2^1.500*AESw3^.125
    - .234375*beta[20]*AESw1^.375/AESw2^1.625*AESw3^.250
    - .187500*beta[21]*AESw1^.375/AESw2^1.750*AESw3^.375
    - .109375*beta[22]*AESw1^.375/AESw2^1.875*AESw3^.500
    - .234375*beta[25]*AESw1^.250/AESw2^1.375*AESw3^.125
    - .250000*beta[26]*AESw1^.250/AESw2^1.500*AESw3^.250
    - .234375*beta[27]*AESw1^.250/AESw2^1.625*AESw3^.375
    - .187500*beta[28]*AESw1^.250/AESw2^1.750*AESw3^.500
    - .109375*beta[29]*AESw1^.250/AESw2^1.875*AESw3^.625
    - .187500*beta[32]*AESw1^.125/AESw2^1.250*AESw3^.125
    - .234375*beta[33]*AESw1^.125/AESw2^1.375*AESw3^.250
    - .250000*beta[34]*AESw1^.125/AESw2^1.500*AESw3^.375
    - .234375*beta[35]*AESw1^.125/AESw2^1.625*AESw3^.500
    - .187500*beta[36]*AESw1^.125/AESw2^1.750*AESw3^.625
    - .109375*beta[37]*AESw1^.125/AESw2^1.875*AESw3^.750);
AESh23 = AESy*( .234375*beta[43]/AESw2^.625/AESw3^.375
    +  .109375*beta[45]/AESw2^.875/AESw3^.125
    +  .250000*beta[42]/AESw2^.500/AESw3^.500
    +  .187500*beta[44]/AESw2^.750/AESw3^.250
    +  .109375*beta[39]/AESw2^.125/AESw3^.875
```

```
    + .187500*beta[40]/AESw2^.250/AESw3^.750
    + .234375*beta[41]/AESw2^.375/AESw3^.625
    + .15625e-1*beta[7]*AESw1^.750/AESw2^.875/AESw3^.875
    + .31250e-1*beta[10]*AESw1^.625/AESw2^.750/AESw3^.875
    + .31250e-1*beta[11]*AESw1^.625/AESw2^.875/AESw3^.750
    + .46875e-1*beta[14]*AESw1^.500/AESw2^.625/AESw3^.875
    + .62500e-1*beta[15]*AESw1^.500/AESw2^.750/AESw3^.750
    + .46875e-1*beta[16]*AESw1^.500/AESw2^.875/AESw3^.625
    + .62500e-1*beta[19]*AESw1^.375/AESw2^.500/AESw3^.875
    + .93750e-1*beta[20]*AESw1^.375/AESw2^.625/AESw3^.750
    + .93750e-1*beta[21]*AESw1^.375/AESw2^.750/AESw3^.625
    + .62500e-1*beta[22]*AESw1^.375/AESw2^.875/AESw3^.500
    + .78125e-1*beta[25]*AESw1^.250/AESw2^.375/AESw3^.875
    + .125000*beta[26]*AESw1^.250/AESw2^.500/AESw3^.750
    + .140625*beta[27]*AESw1^.250/AESw2^.625/AESw3^.625
    + .125000*beta[28]*AESw1^.250/AESw2^.750/AESw3^.500
    + .78125e-1*beta[29]*AESw1^.250/AESw2^.875/AESw3^.375
    + .93750e-1*beta[32]*AESw1^.125/AESw2^.250/AESw3^.875
    + .156250*beta[33]*AESw1^.125/AESw2^.375/AESw3^.750
    + .187500*beta[34]*AESw1^.125/AESw2^.500/AESw3^.625
    + .187500*beta[35]*AESw1^.125/AESw2^.625/AESw3^.500
    + .156250*beta[36]*AESw1^.125/AESw2^.750/AESw3^.375
    + .93750e-1*beta[37]*AESw1^.125/AESw2^.875/AESw3^.250);
AESh33 = AESy*(
    - .234375*beta[41]*AESw2^.625/AESw3^1.625
```

- .234375*beta[12]*AESw1^.625/AESw3^1.625

- .187500*beta[44]*AESw2^.250/AESw3^1.250

- .250000*beta[17]*AESw1^.500/AESw3^1.500

- .109375*beta[45]*AESw2^.125/AESw3^1.125

- .234375*beta[23]*AESw1^.375/AESw3^1.375

- .187500*beta[8]*AESw1^.750/AESw3^1.750

- .250000*beta[42]*AESw2^.500/AESw3^1.500

- .187500*beta[30]*AESw1^.250/AESw3^1.250

- .109375*beta[5]*AESw1^.875/AESw3^1.875

- .234375*beta[43]*AESw2^.375/AESw3^1.375

- .109375*beta[39]*AESw2^.875/AESw3^1.875

- .109375*beta[38]*AESw1^.125/AESw3^1.125

- .187500*beta[40]*AESw2^.750/AESw3^1.750

- .109375*beta[7]*AESw1^.750*AESw2^.125/AESw3^1.875

- .109375*beta[10]*AESw1^.625*AESw2^.250/AESw3^1.875

- .187500*beta[11]*AESw1^.625*AESw2^.125/AESw3^1.750

- .109375*beta[14]*AESw1^.500*AESw2^.375/AESw3^1.875

- .187500*beta[15]*AESw1^.500*AESw2^.250/AESw3^1.750

- .234375*beta[16]*AESw1^.500*AESw2^.125/AESw3^1.625

- .109375*beta[19]*AESw1^.375*AESw2^.500/AESw3^1.875

- .187500*beta[20]*AESw1^.375*AESw2^.375/AESw3^1.750

- .234375*beta[21]*AESw1^.375*AESw2^.250/AESw3^1.625

- .250000*beta[22]*AESw1^.375*AESw2^.125/AESw3^1.500

- .109375*beta[25]*AESw1^.250*AESw2^.625/AESw3^1.875

- .187500*beta[26]*AESw1^.250*AESw2^.500/AESw3^1.750

```
  - .234375*beta[27]*AESw1^.250*AESw2^.375/AESw3^1.625
  - .250000*beta[28]*AESw1^.250*AESw2^.250/AESw3^1.500
  - .234375*beta[29]*AESw1^.250*AESw2^.125/AESw3^1.375
  - .109375*beta[32]*AESw1^.125*AESw2^.750/AESw3^1.875
  - .187500*beta[33]*AESw1^.125*AESw2^.625/AESw3^1.750
  - .234375*beta[34]*AESw1^.125*AESw2^.500/AESw3^1.625
  - .250000*beta[35]*AESw1^.125*AESw2^.375/AESw3^1.500
  - .234375*beta[36]*AESw1^.125*AESw2^.250/AESw3^1.375
  - .187500*beta[37]*AESw1^.125*AESw2^.125/AESw3^1.250);
AESh = AESh11~AESh12~AESh13 |
    AESh12~AESh22~AESh23 |
    AESh13~AESh23~AESh33;
AEScosthat = AESy*(
   beta[68]*AESw2^.625*AESw3^.375 +
   beta[67]*AESw2^.750*AESw3^.250 +
   beta[28]*AESw1^.250*AESw3^.750 +
   beta[69]*AESw2^.500*AESw3^.500 +
   beta[29]*AESw1^.125*AESw2^.875 +
   beta[7]*AESw1^.625*AESw2^.375 +
   beta[4]*AESw1^.750*AESw2^.250 +
   beta[21]*AESw1^.375*AESw3^.625 +
   beta[72]*AESw2^.125*AESw3^.875 +
   beta[36]*AESw1^.125*AESw3^.875 +
   beta[6]*AESw1^.750*AESw3^.250 +
   beta[15]*AESw1^.500*AESw3^.500 +
```

beta[3]*AESw1^.875*AESw3^.125 +

beta[71]*AESw2^.250*AESw3^.750 +

beta[11]*AESw1^.500*AESw2^.500 +

beta[70]*AESw2^.375*AESw3^.625 +

beta[2]*AESw1^.875*AESw2^.125 +

beta[66]*AESw2^.875*AESw3^.125 +

beta[10]*AESw1^.625*AESw3^.375 +

beta[22]*AESw1^.250*AESw2^.750 +

beta[16]*AESw1^.375*AESw2^.625 +

beta[5]*AESw1^.750*AESw2^.125*AESw3^.125 +

beta[1]*AESw1^1.00 +

beta[37]*AESw2^1.00 +

beta[73]*AESw3^1.00 +

beta[8]*AESw1^.625*AESw2^.250*AESw3^.125 +

beta[9]*AESw1^.625*AESw2^.125*AESw3^.250 +

beta[12]*AESw1^.500*AESw2^.375*AESw3^.125 +

beta[13]*AESw1^.500*AESw2^.250*AESw3^.250 +

beta[14]*AESw1^.500*AESw2^.125*AESw3^.375 +

beta[17]*AESw1^.375*AESw2^.500*AESw3^.125 +

beta[18]*AESw1^.375*AESw2^.375*AESw3^.250 +

beta[19]*AESw1^.375*AESw2^.250*AESw3^.375 +

beta[20]*AESw1^.375*AESw2^.125*AESw3^.500 +

beta[23]*AESw1^.250*AESw2^.625*AESw3^.125 +

beta[24]*AESw1^.250*AESw2^.500*AESw3^.250 +

beta[25]*AESw1^.250*AESw2^.375*AESw3^.375 +

```
            beta[26]*AESw1^.250*AESw2^.250*AESw3^.500 +
            beta[27]*AESw1^.250*AESw2^.125*AESw3^.625 +
            beta[30]*AESw1^.125*AESw2^.750*AESw3^.125 +
            beta[31]*AESw1^.125*AESw2^.625*AESw3^.250 +
            beta[32]*AESw1^.125*AESw2^.500*AESw3^.375 +
            beta[33]*AESw1^.125*AESw2^.375*AESw3^.500 +
            beta[34]*AESw1^.125*AESw2^.250*AESw3^.625 +
            beta[35]*AESw1^.125*AESw2^.125*AESw3^.750);
      AESx1hat = AESz1*beta[1:cs];
      AESx2hat = AESz2*beta[(cs+1):(2*cs)];
      AESx3hat = AESz3*beta[((2*cs)+1):(3*cs)];
      AESxhat  = AESx1hat~AESx2hat~AESx3hat;
      @ Formula C[i,j]*C/(C[i]*C[j])  in e.g. Paolo Bertoletti:
        Elasticity of Substitution, one more time: gross versus net, direct versus inverse measures"
        p. 6 (paper found in the internet) @
      AEShat   = AESh.*AEScosthat./(AESxhat'AESxhat);
    endif;
    retp(AEShat);
endp;
```

Section 4: Procedure phaim.g

```
/* Procedure haim to calculate the Hessian of the AIM(order) */

proc phaim(ygrid,psi1,psi2,psi3);

    local h11, h12, h13, h22, h23, h33;

    if order == 1;

      h11 = ygrid*(-1/4*beta[2]/psi1^(3/2)*psi2^(1/2)-1/4*beta[3]/psi1^(3/2)*psi3^(1/2));

      h12 = 1/4*ygrid*beta[2]/psi1^(1/2)/psi2^(1/2);

      h13 = 1/4*ygrid*beta[3]/psi1^(1/2)/psi3^(1/2);

      h22 = ygrid*(-1/4*beta[2]*psi1^(1/2)/psi2^(3/2)-1/4*beta[6]/psi2^(3/2)*psi3^(1/2));

      h23 = 1/4*ygrid*beta[6]/psi2^(1/2)/psi3^(1/2);

      h33 = ygrid*(-1/4*beta[3]*psi1^(1/2)/psi3^(3/2)-1/4*beta[6]*psi2^(1/2)/psi3^(3/2));

    endif;

    if order == 2;

      h11 = ygrid * (-.1875 *beta[2]* psi2^.25 / psi1^1.25-.1875 *beta[3]* psi3^.25 / psi1^1.25-1 / 4 *
beta[4] * psi2^(1 / 2) / psi1^(3 / 2)-.25 * beta[5] * psi2^.25 * psi3^.25 / psi1^1.5-.25 * beta[6] * psi3^.5 /
psi1^1.5-.1875 * beta[7] * psi2^.75 / psi1^1.75-.1875 *

        beta[8] * psi2^.5 * psi3^.25 / psi1^1.75-.1875 * beta[9] * psi2^.25 * psi3^.5 / psi1^1.75-.1875 *
beta[10] * psi3^.75 / psi1^1.75);

      h12 = ygrid * (.1875 *beta[2]/ psi1^.25 / psi2^.75 + 1 / 4 * beta[4] / psi1^(1 / 2) / psi2^(1 / 2) +
.125 * beta[5] * psi3^.25 / psi1^.5 / psi2^.75 + .1875 * beta[7] / psi1^.75 / psi2^.25 + .125 * beta[8] *
psi3^.25 / psi1^.75 / psi2^.5 + .625e-1 * beta[9] * psi3^.5

        / psi1^.75 / psi2^.75);

      h13 = ygrid * (.1875 *beta[3]/ psi1^.25 / psi3^.75 + .125 * beta[5] * psi2^.25 / psi1^.5 / psi3^.75 +
.25 * beta[6] / psi1^.5 / psi3^.5 + .625e-1 * beta[8] * psi2^.5 / psi1^.75 / psi3^.75 + .125 * beta[9] *
psi2^.25 / psi1^.75 / psi3^.5 + .1875 * beta[10] / psi1^.75 / psi3^.25);
```

```
        h22 = ygrid * (-.1875 *beta[2]* psi1^.75 / psi2^1.75-1 / 4 * beta[4] * psi1^(1 / 2) / psi2^(3 / 2)-
.1875 * beta[5] * psi1^.5 * psi3^.25 / psi2^1.75-.1875 * beta[7] * psi1^.25 / psi2^1.25-.25 * beta[8] *
psi1^.25 * psi3^.25 / psi2^1.5-.1875 * beta[9] * psi1^.25 * psi3^.5

          / psi2^1.75-.1875 * beta[18] * psi3^.25 / psi2^1.25-.25 * beta[19] * psi3^.5 / psi2^1.5-.1875 *
beta[20] * psi3^.75 / psi2^1.75);

        h23 = ygrid * (.625e-1 * beta[5] * psi1^.5 / psi2^.75 / psi3^.75 + .125 * beta[8] * psi1^.25 /
psi2^.5 / psi3^.75 + .125 * beta[9] * psi1^.25 / psi2^.75 / psi3^.5 + .1875 * beta[18] / psi2^.25 / psi3^.75
+ .25 * beta[19] / psi2^.5 / psi3^.5 + .1875 * beta[20] / psi2^.75 / psi3^.25);

        h33 = ygrid * (-.1875 *beta[3]* psi1^.75 / psi3^1.75-.1875 * beta[5] * psi1^.5 * psi2^.25 /
psi3^1.75-.25 * beta[6] * psi1^.5 / psi3^1.5-.1875 * beta[8] * psi1^.25 * psi2^.5 / psi3^1.75-.25 * beta[9]
* psi1^.25 * psi2^.25 / psi3^1.5-.1875 * beta[10] * psi1^.25

          / psi3^1.25-.1875 * beta[18] * psi2^.75 / psi3^1.75-.25 * beta[19] * psi2^.5 / psi3^1.5-.1875 *
beta[20] * psi2^.25 / psi3^1.25);

      endif;

      if order == 3;

        h11 = ygrid*(

            -.109375*beta[5]/psi1^1.125*psi3^.125

            -.250000*beta[17]/psi1^1.500*psi3^.500

            -.187500*beta[8]/psi1^1.250*psi3^.250

            -.109375*beta[4]/psi1^1.125*psi2^.125

            -.234375*beta[9]/psi1^1.375*psi2^.375

            -.250000*beta[13]/psi1^1.500*psi2^.500

            -.109375*beta[38]/psi1^1.875*psi3^.875

            -.187500*beta[30]/psi1^1.750*psi3^.750

            -.109375*beta[31]/psi1^1.875*psi2^.875

            -.234375*beta[12]/psi1^1.375*psi3^.375

            -.234375*beta[18]/psi1^1.625*psi2^.625

            -.187500*beta[24]/psi1^1.750*psi2^.750
```

```
    -.234375*beta[23]/psi1^1.625*psi3^.625
    -.187500*beta[6]/psi1^1.250*psi2^.250
    -.187500*beta[7]/psi1^1.250*psi2^.125*psi3^.125
    -.234375*beta[10]/psi1^1.375*psi2^.250*psi3^.125
    -.234375*beta[11]/psi1^1.375*psi2^.125*psi3^.250
    -.250000*beta[14]/psi1^1.500*psi2^.375*psi3^.125
    -.250000*beta[15]/psi1^1.500*psi2^.250*psi3^.250
    -.250000*beta[16]/psi1^1.500*psi2^.125*psi3^.375
    -.234375*beta[19]/psi1^1.625*psi2^.500*psi3^.125
    -.234375*beta[20]/psi1^1.625*psi2^.375*psi3^.250
    -.234375*beta[21]/psi1^1.625*psi2^.250*psi3^.375
    -.234375*beta[22]/psi1^1.625*psi2^.125*psi3^.500
    -.187500*beta[25]/psi1^1.750*psi2^.625*psi3^.125
    -.187500*beta[26]/psi1^1.750*psi2^.500*psi3^.250
    -.187500*beta[27]/psi1^1.750*psi2^.375*psi3^.375
    -.187500*beta[28]/psi1^1.750*psi2^.250*psi3^.500
    -.187500*beta[29]/psi1^1.750*psi2^.125*psi3^.625
    -.109375*beta[32]/psi1^1.875*psi2^.750*psi3^.125
    -.109375*beta[33]/psi1^1.875*psi2^.625*psi3^.250
    -.109375*beta[34]/psi1^1.875*psi2^.500*psi3^.375
    -.109375*beta[35]/psi1^1.875*psi2^.375*psi3^.500
    -.109375*beta[36]/psi1^1.875*psi2^.250*psi3^.625
    -.109375*beta[37]/psi1^1.875*psi2^.125*psi3^.750);
h12 = ygrid*(.250000*beta[13]/psi1^.500/psi2^.500 +
    .234375*beta[18]/psi1^.625/psi2^.375 +
```

.109375*beta[31]/psi1^.875/psi2^.125 +

.187500*beta[6]/psi1^.250/psi2^.750 +

.187500*beta[24]/psi1^.750/psi2^.250 +

.234375*beta[9]/psi1^.375/psi2^.625 +

.109375*beta[4]/psi1^.125/psi2^.875 +

.093750*beta[7]/psi1^.250/psi2^.875*psi3^.125 +

.156250*beta[10]/psi1^.375/psi2^.750*psi3^.125 +

.078125*beta[11]/psi1^.375/psi2^.875*psi3^.250 +

.187500*beta[14]/psi1^.500/psi2^.625*psi3^.125 +

.125000*beta[15]/psi1^.500/psi2^.750*psi3^.250 +

.062500*beta[16]/psi1^.500/psi2^.875*psi3^.375 +

.187500*beta[19]/psi1^.625/psi2^.500*psi3^.125 +

.140625*beta[20]/psi1^.625/psi2^.625*psi3^.250 +

.093750*beta[21]/psi1^.625/psi2^.750*psi3^.375 +

.046875*beta[22]/psi1^.625/psi2^.875*psi3^.500 +

.156250*beta[25]/psi1^.750/psi2^.375*psi3^.125 +

.125000*beta[26]/psi1^.750/psi2^.500*psi3^.250 +

.093750*beta[27]/psi1^.750/psi2^.625*psi3^.375 +

.062500*beta[28]/psi1^.750/psi2^.750*psi3^.500 +

.031250*beta[29]/psi1^.750/psi2^.875*psi3^.625 +

.093750*beta[32]/psi1^.875/psi2^.250*psi3^.125 +

.078125*beta[33]/psi1^.875/psi2^.375*psi3^.250 +

.062500*beta[34]/psi1^.875/psi2^.500*psi3^.375 +

.046875*beta[35]/psi1^.875/psi2^.625*psi3^.500 +

.031250*beta[36]/psi1^.875/psi2^.750*psi3^.625 +

```
     .015625*beta[37]/psi1^.875/psi2^.875*psi3^.750);
h13 = ygrid*(.234375*beta[12]/psi1^.375/psi3^.625
     +  .250000*beta[17]/psi1^.500/psi3^.500
     +  .234375*beta[23]/psi1^.625/psi3^.375
     +  .109375*beta[38]/psi1^.875/psi3^.125
     +  .109375*beta[5]/psi1^.125/psi3^.875
     +  .187500*beta[8]/psi1^.250/psi3^.750
     +  .187500*beta[30]/psi1^.750/psi3^.250
     +  .93750e-1*beta[7]/psi1^.250*psi2^.125/psi3^.875
     +  .78125e-1*beta[10]/psi1^.375*psi2^.250/psi3^.875
     +  .156250*beta[11]/psi1^.375*psi2^.125/psi3^.750
     +  .62500e-1*beta[14]/psi1^.500*psi2^.375/psi3^.875
     +  .125000*beta[15]/psi1^.500*psi2^.250/psi3^.750
     +  .187500*beta[16]/psi1^.500*psi2^.125/psi3^.625
     +  .46875e-1*beta[19]/psi1^.625*psi2^.500/psi3^.875
     +  .93750e-1*beta[20]/psi1^.625*psi2^.375/psi3^.750
     +  .140625*beta[21]/psi1^.625*psi2^.250/psi3^.625
     +  .187500*beta[22]/psi1^.625*psi2^.125/psi3^.500
     +  .31250e-1*beta[25]/psi1^.750*psi2^.625/psi3^.875
     +  .62500e-1*beta[26]/psi1^.750*psi2^.500/psi3^.750
     +  .93750e-1*beta[27]/psi1^.750*psi2^.375/psi3^.625
     +  .125000*beta[28]/psi1^.750*psi2^.250/psi3^.500
     +  .156250*beta[29]/psi1^.750*psi2^.125/psi3^.375
     +  .15625e-1*beta[32]/psi1^.875*psi2^.750/psi3^.875
     +  .31250e-1*beta[33]/psi1^.875*psi2^.625/psi3^.750
```

```
      + .46875e-1*beta[34]/psi1^.875*psi2^.500/psi3^.625
      + .62500e-1*beta[35]/psi1^.875*psi2^.375/psi3^.500
      + .78125e-1*beta[36]/psi1^.875*psi2^.250/psi3^.375
      + .93750e-1*beta[37]/psi1^.875*psi2^.125/psi3^.250);
h22 = ygrid*(
     - .234375*beta[18]*psi1^.375/psi2^1.375
     - .187500*beta[44]/psi2^1.750*psi3^.750
     - .109375*beta[31]*psi1^.125/psi2^1.125
     - .250000*beta[42]/psi2^1.500*psi3^.500
     - .109375*beta[45]/psi2^1.875*psi3^.875
     - .234375*beta[9]*psi1^.625/psi2^1.625
     - .109375*beta[39]/psi2^1.125*psi3^.125
     - .234375*beta[41]/psi2^1.375*psi3^.375
     - .187500*beta[6]*psi1^.750/psi2^1.750
     - .234375*beta[43]/psi2^1.625*psi3^.625
     - .187500*beta[24]*psi1^.250/psi2^1.250
     - .250000*beta[13]*psi1^.500/psi2^1.500
     - .187500*beta[40]/psi2^1.250*psi3^.250
     - .109375*beta[4]*psi1^.875/psi2^1.875
     - .109375*beta[7]*psi1^.750/psi2^1.875*psi3^.125
     - .187500*beta[10]*psi1^.625/psi2^1.750*psi3^.125
     - .109375*beta[11]*psi1^.625/psi2^1.875*psi3^.250
     - .234375*beta[14]*psi1^.500/psi2^1.625*psi3^.125
     - .187500*beta[15]*psi1^.500/psi2^1.750*psi3^.250
     - .109375*beta[16]*psi1^.500/psi2^1.875*psi3^.375
```

```
    - .250000*beta[19]*psi1^.375/psi2^1.500*psi3^.125
    - .234375*beta[20]*psi1^.375/psi2^1.625*psi3^.250
    - .187500*beta[21]*psi1^.375/psi2^1.750*psi3^.375
    - .109375*beta[22]*psi1^.375/psi2^1.875*psi3^.500
    - .234375*beta[25]*psi1^.250/psi2^1.375*psi3^.125
    - .250000*beta[26]*psi1^.250/psi2^1.500*psi3^.250
    - .234375*beta[27]*psi1^.250/psi2^1.625*psi3^.375
    - .187500*beta[28]*psi1^.250/psi2^1.750*psi3^.500
    - .109375*beta[29]*psi1^.250/psi2^1.875*psi3^.625
    - .187500*beta[32]*psi1^.125/psi2^1.250*psi3^.125
    - .234375*beta[33]*psi1^.125/psi2^1.375*psi3^.250
    - .250000*beta[34]*psi1^.125/psi2^1.500*psi3^.375
    - .234375*beta[35]*psi1^.125/psi2^1.625*psi3^.500
    - .187500*beta[36]*psi1^.125/psi2^1.750*psi3^.625
    - .109375*beta[37]*psi1^.125/psi2^1.875*psi3^.750);
h23 = ygrid*( .234375*beta[43]/psi2^.625/psi3^.375
    + .109375*beta[45]/psi2^.875/psi3^.125
    + .250000*beta[42]/psi2^.500/psi3^.500
    + .187500*beta[44]/psi2^.750/psi3^.250
    + .109375*beta[39]/psi2^.125/psi3^.875
    + .187500*beta[40]/psi2^.250/psi3^.750
    + .234375*beta[41]/psi2^.375/psi3^.625
    + .15625e-1*beta[7]*psi1^.750/psi2^.875/psi3^.875
    + .31250e-1*beta[10]*psi1^.625/psi2^.750/psi3^.875
    + .31250e-1*beta[11]*psi1^.625/psi2^.875/psi3^.750
```

```
    + .46875e-1*beta[14]*psi1^.500/psi2^.625/psi3^.875
    + .62500e-1*beta[15]*psi1^.500/psi2^.750/psi3^.750
    + .46875e-1*beta[16]*psi1^.500/psi2^.875/psi3^.625
    + .62500e-1*beta[19]*psi1^.375/psi2^.500/psi3^.875
    + .93750e-1*beta[20]*psi1^.375/psi2^.625/psi3^.750
    + .93750e-1*beta[21]*psi1^.375/psi2^.750/psi3^.625
    + .62500e-1*beta[22]*psi1^.375/psi2^.875/psi3^.500
    + .78125e-1*beta[25]*psi1^.250/psi2^.375/psi3^.875
    + .125000*beta[26]*psi1^.250/psi2^.500/psi3^.750
    + .140625*beta[27]*psi1^.250/psi2^.625/psi3^.625
    + .125000*beta[28]*psi1^.250/psi2^.750/psi3^.500
    + .78125e-1*beta[29]*psi1^.250/psi2^.875/psi3^.375
    + .93750e-1*beta[32]*psi1^.125/psi2^.250/psi3^.875
    + .156250*beta[33]*psi1^.125/psi2^.375/psi3^.750
    + .187500*beta[34]*psi1^.125/psi2^.500/psi3^.625
    + .187500*beta[35]*psi1^.125/psi2^.625/psi3^.500
    + .156250*beta[36]*psi1^.125/psi2^.750/psi3^.375
    + .93750e-1*beta[37]*psi1^.125/psi2^.875/psi3^.250);
h33 = ygrid*(
    - .234375*beta[41]*psi2^.625/psi3^1.625
    - .234375*beta[12]*psi1^.625/psi3^1.625
    - .187500*beta[44]*psi2^.250/psi3^1.250
    - .250000*beta[17]*psi1^.500/psi3^1.500
    - .109375*beta[45]*psi2^.125/psi3^1.125
    - .234375*beta[23]*psi1^.375/psi3^1.375
```

- .187500*beta[8]*psi1^.750/psi3^1.750

- .250000*beta[42]*psi2^.500/psi3^1.500

- .187500*beta[30]*psi1^.250/psi3^1.250

- .109375*beta[5]*psi1^.875/psi3^1.875

- .234375*beta[43]*psi2^.375/psi3^1.375

- .109375*beta[39]*psi2^.875/psi3^1.875

- .109375*beta[38]*psi1^.125/psi3^1.125

- .187500*beta[40]*psi2^.750/psi3^1.750

- .109375*beta[7]*psi1^.750*psi2^.125/psi3^1.875

- .109375*beta[10]*psi1^.625*psi2^.250/psi3^1.875

- .187500*beta[11]*psi1^.625*psi2^.125/psi3^1.750

- .109375*beta[14]*psi1^.500*psi2^.375/psi3^1.875

- .187500*beta[15]*psi1^.500*psi2^.250/psi3^1.750

- .234375*beta[16]*psi1^.500*psi2^.125/psi3^1.625

- .109375*beta[19]*psi1^.375*psi2^.500/psi3^1.875

- .187500*beta[20]*psi1^.375*psi2^.375/psi3^1.750

- .234375*beta[21]*psi1^.375*psi2^.250/psi3^1.625

- .250000*beta[22]*psi1^.375*psi2^.125/psi3^1.500

- .109375*beta[25]*psi1^.250*psi2^.625/psi3^1.875

- .187500*beta[26]*psi1^.250*psi2^.500/psi3^1.750

- .234375*beta[27]*psi1^.250*psi2^.375/psi3^1.625

- .250000*beta[28]*psi1^.250*psi2^.250/psi3^1.500

- .234375*beta[29]*psi1^.250*psi2^.125/psi3^1.375

- .109375*beta[32]*psi1^.125*psi2^.750/psi3^1.875

- .187500*beta[33]*psi1^.125*psi2^.625/psi3^1.750

```
        - .234375*beta[34]*psi1^.125*psi2^.500/psi3^1.625
        - .250000*beta[35]*psi1^.125*psi2^.375/psi3^1.500
        - .234375*beta[36]*psi1^.125*psi2^.250/psi3^1.375
        - .187500*beta[37]*psi1^.125*psi2^.125/psi3^1.250);
  endif;
  hfull = h11~h12~h13 |
      h12~h22~h23 |
      h13~h23~h33 ;
  retp(hfull);
endp;
```

Section 5: Procedure phaimmh.g

```
/* Procedure haim to calculate the Hessian of the AIM(order) */

proc phaimmh(ygrid,psi1,psi2,psi3);

    local h11, h12, h22;

    if order == 1;

      h11 = ygrid*(-1/4*beta[2]/psi1^(3/2)*psi2^(1/2)-1/4*beta[3]/psi1^(3/2)*psi3^(1/2));

      h12 = 1/4*ygrid*beta[2]/psi1^(1/2)/psi2^(1/2);

      h22 = ygrid*(-1/4*beta[2]*psi1^(1/2)/psi2^(3/2)-1/4*beta[6]/psi2^(3/2)*psi3^(1/2));

    endif;

    if order == 2;

      h11 = ygrid * (-.1875 *beta[2]* psi2^.25 / psi1^1.25-.1875 *beta[3]* psi3^.25 / psi1^1.25-1 / 4 *
beta[4] * psi2^(1 / 2) / psi1^(3 / 2)-.25 * beta[5] * psi2^.25 * psi3^.25 / psi1^1.5-.25 * beta[6] * psi3^.5 /
psi1^1.5-.1875 * beta[7] * psi2^.75 / psi1^1.75-.1875 *

        beta[8] * psi2^.5 * psi3^.25 / psi1^1.75-.1875 * beta[9] * psi2^.25 * psi3^.5 / psi1^1.75-.1875 *
beta[10] * psi3^.75 / psi1^1.75);

      h12 = ygrid * (.1875 *beta[2]/ psi1^.25 / psi2^.75 + 1 / 4 * beta[4] / psi1^(1 / 2) / psi2^(1 / 2) +
.125 * beta[5] * psi3^.25 / psi1^.5 / psi2^.75 + .1875 * beta[7] / psi1^.75 / psi2^.25 + .125 * beta[8] *
psi3^.25 / psi1^.75 / psi2^.5 + .625e-1 * beta[9] * psi3^.5

        / psi1^.75 / psi2^.75);

      h22 = ygrid * (-.1875 *beta[2]* psi1^.75 / psi2^1.75-1 / 4 * beta[4] * psi1^(1 / 2) / psi2^(3 / 2)-
.1875 * beta[5] * psi1^.5 * psi3^.25 / psi2^1.75-.1875 * beta[7] * psi1^.25 / psi2^1.25-.25 * beta[8] *
psi1^.25 * psi3^.25 / psi2^1.5-.1875 * beta[9] * psi1^.25 * psi3^.5

        / psi2^1.75-.1875 * beta[18] * psi3^.25 / psi2^1.25-.25 * beta[19] * psi3^.5 / psi2^1.5-.1875 *
beta[20] * psi3^.75 / psi2^1.75);

    endif;

    if order == 3;

      h11 = ygrid*(
```

-.109375*beta[5]/psi1^1.125*psi3^.125

-.250000*beta[17]/psi1^1.500*psi3^.500

-.187500*beta[8]/psi1^1.250*psi3^.250

-.109375*beta[4]/psi1^1.125*psi2^.125

-.234375*beta[9]/psi1^1.375*psi2^.375

-.250000*beta[13]/psi1^1.500*psi2^.500

-.109375*beta[38]/psi1^1.875*psi3^.875

-.187500*beta[30]/psi1^1.750*psi3^.750

-.109375*beta[31]/psi1^1.875*psi2^.875

-.234375*beta[12]/psi1^1.375*psi3^.375

-.234375*beta[18]/psi1^1.625*psi2^.625

-.187500*beta[24]/psi1^1.750*psi2^.750

-.234375*beta[23]/psi1^1.625*psi3^.625

-.187500*beta[6]/psi1^1.250*psi2^.250

-.187500*beta[7]/psi1^1.250*psi2^.125*psi3^.125

-.234375*beta[10]/psi1^1.375*psi2^.250*psi3^.125

-.234375*beta[11]/psi1^1.375*psi2^.125*psi3^.250

-.250000*beta[14]/psi1^1.500*psi2^.375*psi3^.125

-.250000*beta[15]/psi1^1.500*psi2^.250*psi3^.250

-.250000*beta[16]/psi1^1.500*psi2^.125*psi3^.375

-.234375*beta[19]/psi1^1.625*psi2^.500*psi3^.125

-.234375*beta[20]/psi1^1.625*psi2^.375*psi3^.250

-.234375*beta[21]/psi1^1.625*psi2^.250*psi3^.375

-.234375*beta[22]/psi1^1.625*psi2^.125*psi3^.500

-.187500*beta[25]/psi1^1.750*psi2^.625*psi3^.125

```
    -.187500*beta[26]/psi1^1.750*psi2^.500*psi3^.250
    -.187500*beta[27]/psi1^1.750*psi2^.375*psi3^.375
    -.187500*beta[28]/psi1^1.750*psi2^.250*psi3^.500
    -.187500*beta[29]/psi1^1.750*psi2^.125*psi3^.625
    -.109375*beta[32]/psi1^1.875*psi2^.750*psi3^.125
    -.109375*beta[33]/psi1^1.875*psi2^.625*psi3^.250
    -.109375*beta[34]/psi1^1.875*psi2^.500*psi3^.375
    -.109375*beta[35]/psi1^1.875*psi2^.375*psi3^.500
    -.109375*beta[36]/psi1^1.875*psi2^.250*psi3^.625
    -.109375*beta[37]/psi1^1.875*psi2^.125*psi3^.750);
h12 = ygrid*(.250000*beta[13]/psi1^.500/psi2^.500 +
    .234375*beta[18]/psi1^.625/psi2^.375 +
    .109375*beta[31]/psi1^.875/psi2^.125 +
    .187500*beta[6]/psi1^.250/psi2^.750 +
    .187500*beta[24]/psi1^.750/psi2^.250 +
    .234375*beta[9]/psi1^.375/psi2^.625 +
    .109375*beta[4]/psi1^.125/psi2^.875 +
    .093750*beta[7]/psi1^.250/psi2^.875*psi3^.125 +
    .156250*beta[10]/psi1^.375/psi2^.750*psi3^.125 +
    .078125*beta[11]/psi1^.375/psi2^.875*psi3^.250 +
    .187500*beta[14]/psi1^.500/psi2^.625*psi3^.125 +
    .125000*beta[15]/psi1^.500/psi2^.750*psi3^.250 +
    .062500*beta[16]/psi1^.500/psi2^.875*psi3^.375 +
    .187500*beta[19]/psi1^.625/psi2^.500*psi3^.125 +
    .140625*beta[20]/psi1^.625/psi2^.625*psi3^.250 +
```

```
    .093750*beta[21]/psi1^.625/psi2^.750*psi3^.375 +
    .046875*beta[22]/psi1^.625/psi2^.875*psi3^.500 +
    .156250*beta[25]/psi1^.750/psi2^.375*psi3^.125 +
    .125000*beta[26]/psi1^.750/psi2^.500*psi3^.250 +
    .093750*beta[27]/psi1^.750/psi2^.625*psi3^.375 +
    .062500*beta[28]/psi1^.750/psi2^.750*psi3^.500 +
    .031250*beta[29]/psi1^.750/psi2^.875*psi3^.625 +
    .093750*beta[32]/psi1^.875/psi2^.250*psi3^.125 +
    .078125*beta[33]/psi1^.875/psi2^.375*psi3^.250 +
    .062500*beta[34]/psi1^.875/psi2^.500*psi3^.375 +
    .046875*beta[35]/psi1^.875/psi2^.625*psi3^.500 +
    .031250*beta[36]/psi1^.875/psi2^.750*psi3^.625 +
    .015625*beta[37]/psi1^.875/psi2^.875*psi3^.750);
h22 = ygrid*(
    - .234375*beta[18]*psi1^.375/psi2^1.375
    - .187500*beta[44]/psi2^1.750*psi3^.750
    - .109375*beta[31]*psi1^.125/psi2^1.125
    - .250000*beta[42]/psi2^1.500*psi3^.500
    - .109375*beta[45]/psi2^1.875*psi3^.875
    - .234375*beta[9]*psi1^.625/psi2^1.625
    - .109375*beta[39]/psi2^1.125*psi3^.125
    - .234375*beta[41]/psi2^1.375*psi3^.375
    - .187500*beta[6]*psi1^.750/psi2^1.750
    - .234375*beta[43]/psi2^1.625*psi3^.625
    - .187500*beta[24]*psi1^.250/psi2^1.250
```

```
      - .250000*beta[13]*psi1^.500/psi2^1.500
      - .187500*beta[40]/psi2^1.250*psi3^.250
      - .109375*beta[4]*psi1^.875/psi2^1.875
      - .109375*beta[7]*psi1^.750/psi2^1.875*psi3^.125
      - .187500*beta[10]*psi1^.625/psi2^1.750*psi3^.125
      - .109375*beta[11]*psi1^.625/psi2^1.875*psi3^.250
      - .234375*beta[14]*psi1^.500/psi2^1.625*psi3^.125
      - .187500*beta[15]*psi1^.500/psi2^1.750*psi3^.250
      - .109375*beta[16]*psi1^.500/psi2^1.875*psi3^.375
      - .250000*beta[19]*psi1^.375/psi2^1.500*psi3^.125
      - .234375*beta[20]*psi1^.375/psi2^1.625*psi3^.250
      - .187500*beta[21]*psi1^.375/psi2^1.750*psi3^.375
      - .109375*beta[22]*psi1^.375/psi2^1.875*psi3^.500
      - .234375*beta[25]*psi1^.250/psi2^1.375*psi3^.125
      - .250000*beta[26]*psi1^.250/psi2^1.500*psi3^.250
      - .234375*beta[27]*psi1^.250/psi2^1.625*psi3^.375
      - .187500*beta[28]*psi1^.250/psi2^1.750*psi3^.500
      - .109375*beta[29]*psi1^.250/psi2^1.875*psi3^.625
      - .187500*beta[32]*psi1^.125/psi2^1.250*psi3^.125
      - .234375*beta[33]*psi1^.125/psi2^1.375*psi3^.250
      - .250000*beta[34]*psi1^.125/psi2^1.500*psi3^.375
      - .234375*beta[35]*psi1^.125/psi2^1.625*psi3^.500
      - .187500*beta[36]*psi1^.125/psi2^1.750*psi3^.625
      - .109375*beta[37]*psi1^.125/psi2^1.875*psi3^.750);
endif;
```

```
    h = h11~h12|
        h12~h22;
    retp(h);
endp;
```

Section 6: Procedure piwishgen.g

```
/*
  OUTPUT: dim^2 * "NSAMP" MATRIX "IWISH" REPRESENTING NSAMP OUTCOMES FROM
          THE INVERTED WISHART DISTRIBUTION WITH "V" DEGREES OF FREEDOM AND
          PARAMETER MATRIX S.

   INPUT: "V" = DEGREES OF FREEDOM (SCALAR)
          "S" = PARAMETER MATRIX (DIM*DIM)
          "NSAMP" = SAMPLE SIZE (SCALAR)
*/

proc piwishgen(v,s,nsamp);
   local dim,invs,cinvs,iwish,state,multn,wish,omega,theta,matt,norrnd;

   dim = cols(s);
   invs = inv(s);
   cinvs = chol(invs)';
   iwish = zeros(dim^2,nsamp);
   norrnd = rndn(1,1);
   for i(1,nsamp,1);
      norrnd = rndn(dim,v);
      multn = (cinvs*norrnd)';
      wish = multn'multn;
      iwish[.,i] = vec(inv(wish));
```

```
    endfor;
    retp(iwish);
endp;
```

Section 7: Procedure ppossur.g

```
/* CALCULATES THE ORDINATE VALUE OF THE MARGINAL POSTERIOR PDF P(BETA|Y) =
INTEGRAL[f(BETA,SIGMA|Y)dSIGMA]

  PROPORTIONAL TO |V'V|^(N/2), WHEREBY V = reshape(ERROR,neq,n)' ;*/




proc ppossur(bstar,y,x,neq);

  local n,nb,posw,v;


  n = rows(y)/neq;

  nb = cols(bstar);    /* normally: nb = 1 */

  posw = zeros(1,nb);

  for i(1,nb,1);

      v = y-x*bstar[.,i];

      v = reshape(v,neq,n)';

      posw[i] = (det(v'v))^(-n/2);

  endfor;

  retp(posw);

endp;
```

Section 8: Procedure pset.g

```
/*************************************************************************************
*****************

                    PARAMETERS OF THE CES-TECHNOLOGY:  (Procedure for the AIMSIM)

*************************************************************************************
******************

-------------------------------------------------------------------------------------------------------------------------
---


  Estimation of AIM, DGP: CES-Technology


  Notation and examples from: Terrell 1995, Flexibility and

  Regularity Properties of the Asymptotically Ideal Production Model


  Programmed: April 2002 - Hendrik Wolff

-------------------------------------------------------------------------------------------------------------------------
---


  Choose Simulation of the AIM(order) */



proc (4) =  pset(sim);

 if sim == 1;

      a1  = 1;

      a2  = 1;
```

```
    a3  = 1;
    rho  = 0.75;
    print "1. Simulation  (Table 1-2, p. 9f): ";
    Print;
 endif;

 if sim == 2;
    a1  = .1;
    a2  = .4;
    a3  = .5;
    rho = -0.5;
    print "2. Simulation (Table 3-4, p. 11f): ";
    Print;
 endif;
 if sim == 3;
    a1  = 0.25;
    a2  = 0.25;
    a3  = 0.5;
    rho = -4.0;
    print "3. Simulation  (Table 5-6, p. 13f): ";
    Print;
 endif;

 retp(a1,a2,a3,rho);
endp;
```