5-5-2014

Matlab Code for Structural Decomposition Analysis

Juan Tomas Sayago Gomez
West Virginia University, Juan.Sayago@mail.wvu.edu

Regional Research Institute
West Virginia University

Technical Document Series

Matlab Code for Structural Decomposition Analysis

Juan Tomás Sayago-Gómez

RRI TechDoc 2014-01

Date submitted:3/14/2014
Date revised: 5/5/2014
Key words/Codes: : Structural Decomposition Analysis,
Input Output, RAS procedure

Matlab Code for Structural Decomposition Analysis*

Abstract

This TechDoc describes the steps necessary to apply the Structural Decomposition Analysis (SDA) using Matlab. The code has two stages. The first stage, which comprises PrepSDA.m and RAS_SDA.m, prepares the data and the input required for SDA based on the accounting identities defined in Miller and Blair (2009) and Jackson and Schwarm (2011). The second stage (SDA.m) carries out the analysis and estimates the results based on the mathematical procedure in Yang and Lahr (2010) and Zhang and Lahr (2014). The end results can be exported into a csv file.

Stage One: Prepare the Data and Obtain the Inputs

1. PrepSDA.m

PrepSDA.m applies the transformations to estimate the total input and output by commodity (q) and industry (g) and the technical coefficients matrix (A), the final demand by industry (B), the structure of final demand (y), the value added per unit of output (v), and input per industry output for each selected industry (e). The code uses the identities given in table 1:

The procedure requires the following information:

- Use matrix: Includes domestic use of commodities by industries and the industry foreign and domestic imports use of commodities by industry.

*Acknowledgements: This material is based upon work supported by the National Science Foundation under Grant No. 1235684.

	Commodity	Industry	FD	Total Output
Commodity		U	E	q
Industry	V			g
FD		W		
Total Output				

Table 1: Single-region commodity by industry framework. Source: Jackson and Schwarm (2011).

- Make matrix: Includes domestic industry make and the industry foreign and domestic export make.
- Final demand by Commodity: Includes domestic final demand by commodity by institution and institutional domestic and foreign imports.
- Value added by industry: Includes value added elements.
- Sales by Commodity: Includes domestic institutional make and institutional domestic and foreign export make.
- Exports by commodity: Includes domestic and foreign exports by commodity.
- Imports by Commodity: Includes domestic and foreign imports by commodity.

2. RAS_SDA.m

This code applies the RAS procedure to convert money values from current prices to prices indexed to a specific year (Dietzenbacher and Hoen, 1998) and these converted variables are the intermediate industry demand (B) and final demand (y). The procedure also requires price deflators by industry or industry groups, the index list of industries for the deflators, and the list of industries resulting from the PrepSDA.m.

Stage Two: Estimate the Structural Decomposition Analysis

3. SDA.m

The Structural Decomposition Analysis (SDA) uses the variation of the different Input-Output components to estimate the change in the intensity of input use. This analysis requires information from both periods to assess the change.
The procedure first decomposes specific total input consumption, as can be seen in equation (1), and then the value added from each industrial sector in equation (2). It is at this point that the value added captures the input intensity in equation (3), highlighting its use. The change in the use of the input intensity is the component change that is measured in the SDA (see equations (4), (5), (6)).

$$\Delta E = \frac{E_1}{E_0} = \frac{e_1' L_1 B_1 y_1}{e_0' L_0 B_0 y_0} \tag{1}$$

$$V = v'(I - A)^{-1} B y = v' L B y \tag{2}$$

$$\varepsilon = \frac{E}{V} = \frac{e' L B y}{v' L B y} \tag{3}$$

$$\frac{\varepsilon_1}{\varepsilon_0} = \frac{\frac{e_1' L_1 B_1 y_1}{v_1' L_1 B_1 y_1}}{\frac{e_0' L_0 B_0 y_0}{v_0' L_0 B_0 y_0}} \tag{4}$$

The analysis of the total production related to a specific input (energy, E) includes the specific (energy) input per unit of output (e'), the Leontief inverse matrix (L), the final demand structure (B), and the aggregate final demand for category or institutions as defined by IMPLAN. On the other hand, the Total value added (V) uses the same variables and the value added per unit of output per industry (v).

$$\begin{aligned}\frac{\varepsilon_1}{\varepsilon_0} &= \left(\frac{e_1' L_0 B_0 y_0}{e_0' L_0 B_0 y_0}\right) \\ &\times \left(\frac{v_0' L_1 B_1 y_1}{v_1' L_1 B_1 y_1}\right) \\ &\times \left(\frac{e_1' L_1 B_0 y_0}{e_1' L_0 B_0 y_0}\right)\left(\frac{v_0' L_0 B_1 y_1}{v_0' L_1 B_1 y_1}\right) \\ &\times \left(\frac{e_1' L_1 B_1 y_0}{e_1' L_1 B_0 y_0}\right)\left(\frac{v_0' L_0 B_0 y_1}{v_0' L_0 B_1 y_1}\right) \\ &\times \left(\frac{e_1' L_1 B_1 y_1}{e_1' L_1 B_1 y_0}\right)\left(\frac{v_0' L_0 B_0 y_0}{v_0' L_0 B_0 y_1}\right)\end{aligned} \tag{5}$$

$$\frac{\varepsilon_1}{\varepsilon_0} = \Delta e \times \Delta v^{-1} \times \Delta L \times \Delta B \times \Delta y \tag{6}$$

Following Zhang and Lahr (2014), the results of the SDA represent the following changes: input per unit of gross output by industry (7a), the inversed effects of changes in value added (7b), the changes in sub industrial structure (7c), the changes of final demand structure (7d), and the changes in final demand levels (7e).

$$\Delta e' = \frac{e_1' L_0 B_0 y_0}{e_0' L_0 B_0 y_0} \tag{7a}$$

$$\Delta v^{-T} = \frac{v_0' L_1 B_1 y_1}{v_1' L_1 B_1 y_1} \tag{7b}$$

$$\Delta L = \left(\frac{e_1' L_1 B_0 y_0}{e_1' L_0 B_0 y_0}\right)\left(\frac{v_0' L_0 B_1 y_1}{v_0' L_1 B_1 y_1}\right) \tag{7c}$$

$$\Delta B = \left(\frac{e_1' L_1 B_1 y_0}{e_1' L_1 B_0 y_0}\right)\left(\frac{v_0' L_0 B_0 y_1}{v_0' L_0 B_1 y_1}\right) \tag{7d}$$

$$\Delta y = \left(\frac{e_1' L_1 B_1 y_1}{e_1' L_1 B_1 y_0}\right)\left(\frac{v_0' L_0 B_0 y_0}{v_0' L_0 B_0 y_1}\right) \tag{7e}$$

Instructions

The zip file SDA_example.zip includes two sets of files: Make1.mat, Make2.mat, Use1.mat, Use2.mat, Sales1.mat, Sales2.mat, Finaldemand1.mat, Finaldemand2.mat, Imports1.mat, Imports2.mat, Exports1.mat, Exports2.mat, ValueAdded1.mat, ValueAdded2.mat, PPindustries.mat, and Deflator.mat. The number in each file name indicates a posterior period for the same regional unit. The two non-numbered files include the price deflator (Deflator.mat) and the industries by number in groups (PPindustries.mat) where the number in each line identifies the last sector included in the model. The zip file also includes the codes Prep_SDA.m, RAS_SDA.m, SDA.m, and test_file.m. To run the example below, perform the following steps:

1. Unzip the file SDA_example.zip and save the files in a folder in the computer.
2. Open matlab and change the folder to where saved the files in the computer.
3. Open the matlab test_file.m and run it.
4. To view the results open the file SDA_results.csv

Supporting Algorithm(s)/Code

PrepSDA.m

```
function [A,e,va,F,Y,filtg] = PrepSDA(U,V,E,VA,Sales,EXP,IMP,sec,scrap)
% PURPOSE: Computes the A matrix and other required vectors and matrixes
% by using the identities in the IO matrix
%
%
% This program has been developed to create the matrixes A and L which are
% used on the Structural Decomposition Analysis. The program uses matrixes
% Use(U) and Make (V) to
%
% ---------------------------------------------------
%  USAGE: results = makeA(U,V)
%  where: U = Use matrix
```

```
%         V = Make matrix
%         E = Final Demand by commodity
%         VA = Value Added
%         Sales = Sales by Commodity
%         EXP = Total Exports by Commodity
%         IMP = Total Imports by Commodity
%         sec = industrial sector to estimate the use intensity or vector of
%         employment per industry
%         scrap = row in the Make matrix where scrap is accounted
%
% ------------------------------------------------
%  RETURNS: a matrixes and vectors
%         A  = Technical coefficients matrix
%         e  = selected input per unit of output by industry (input
%         intensity)
%         v  = value added per unit of output by industry
%         F  = Final demand by industry
%         Y  = Aggregated final demand per categories.
%         filtg = Industry codes for the industries included.
%  ------------------------------------------------
%
%  Function created by J. T. Sayago-Gomez (Spring 2014) to prepare the data
%  to apply the Structural Decomposition analysis.
%
% REFERENCES:
% Miller and Blair. (2009). The Commodity-by-Industry Approach in
% Input-Output Models. In Input-Output Analysis: Foundations and Extensions,
%Ch. 5. Cambridge.
%
% Jackson and Schwarm. (2007). Issues in the implementation of interregional
% commodity by industry input-output models.
%
%Calculates the outputs and inputs and uses the size of the Use and Make
% matrixes to create the D and B matrixes
%
% With D and B calculate A after correcting for scrap

% First the code checks the number of industries and commodities to match.

if size(U)==size(V')
disp('Number of industries and commodities in Use and Make match');
else
errordlg('Number of industries and commodities in Use and Make do not match') ;
end
```

```
v=sum(VA)
gg=sum(U)+v; %sum of column elements to get output

g=sum(V'); %sum of row elements of V to get output

% The code checks the industry outputs to ensure they are equal and
% balances the model.

T=g./gg;
T(isnan(T)) = [];
if all((0.999<T)&(T<1.001))
    disp('Total industry output matches');
else
    errordlg('Total industry output does not match') ;
end

% The code checks the industry outputs to ensure they are equal and
% balances the model.

q=sum(V)+sum(Sales)+IMP; %sum of column elements of V to get inputs
qq=sum(U')+sum(E')+EXP;
TT=q./qq;
TT(isnan(TT)) = [];
if all((0.999<TT)&(TT<1.001))
    disp('Total commodity output matches');
else
    errordlg('Total commodity output does not match') ;
end

% The code corrects for scrap production, adding it to the D of the other
% sectors. In addition, itcorrects for industries where no production has
% taken place, dropping them from the analysis.

h=V(:,scrap);

location0hs=find(h==0); % the industries that do not produce are identified
location0gs=find(g==0);
location0qs=find(q==0);
SV=size(V);
filtg=1:SV(1);
filtq=1:SV(1);
filtg(location0gs)=[];%industries that do not produce are removed
g(location0gs)=[]; % filters out industrial sectors without output
h(location0gs)=[]; % filters out industrial sectors without output
```

```
q(location0qs)=1; % Commodity sectors without production are not dropped.
V1=V(filtg,:); % filters out industrial sectors without output
U1=U(:,filtg); % filters out industrial sectors without output


sU=size(U1);
sV=size(V1);

% The code then estimates the B and D, which are necessary to
% estimate A
B=U1*diag(1./g);
D=V1*diag(1./q);
[Shr,Shc]=size(h);

% Then the code estimates a new D where it corrects for production of scrap.
D1=((eye(Shr)-diag(h'./g))^(-1))*D;

% After having B and D1, it proceeds to estimate final demand and the
% Technical coefficients matrix. The code also estimates the output
% variables required for the SDA analysis.
f=D1*E;
Y=sum(f);
location0Ys=find(Y==0);
Y(location0Ys)=1;
Sf=size(f);
F=f./repmat(Y,Sf(1),1);
Y(location0Ys)=0;
v(location0gs)=[];
va=v./g;
A=D1*B;
[dsr,dsc]=size(sec)
if dsr==1
locationindse=find(filtg==sec);
e=A(locationindse,:);
else
    sec(location0gs)=[];
    e=sec./g
end
end


RAS_SDA.m
function [FP,YP] = RAS(F,y,ind,P,index,time)
% PURPOSE: RAS procedure to deflate SDA's arguments to constant prices.
```

```
% ------------------------------------------------
%  USAGE: [FP,YP] = RAS(F,y,ind,P,index,time)
%  where: F   = Final demand by industry
%         Y   = Aggregated final demand per categories.
%         ind = industries included
%         P = Price deflator by industry or group of aggregated industries
%         index = index for the deflator industries or group of industries.
%         time = prices deflated forward in time if set is equal to 1 or backward
% in time if set is equal to 2.
%
% ------------------------------------------------
%  RETURNS: a structure
%         FP = Changes in input efficiency deflated
%         YP = Changes in value added of production structure deflated
%
%  ------------------------------------------------
%
%  Function created by J. T. Sayago-Gomez (fall 2013) to implement the
%  structural decomposition analysis
%
%REFERENCES:
%
% Dietzenbacher E. & A. R. Hoen. (1998). Deflation of input-output tables
% from the user's point of view: A heuristic approach. Review of Income
% and Wealth. (44, 111-122.)
%
% The code begins here:

% The data is from the SDA's F and Y; these matrix and vector are
% reconverted to total values of final demand by industry and institution
% so that it can apply the price index to the price variation to convert money
% prices into constant prices of a specific year. If prices are indexed by
% groups, then the same price index applies to more than one industrial
% sector; and the code applies the same index to each of those industries.

[j,sz]=size(F);
B2=repmat(y,j,1).*F;
[n,si]=size(index);
rep=1:(n+1);
value=zeros(j,1);
for i=si:-1:1
value(ind<=index(i))=i;
end

value(find(value==0))=si+1;
```

```
Pindex=P(value);
if time==1
BN=B2./repmat(Pindex,1,sz);
else
BN=B2.*repmat(Pindex,1,sz) ;
end

%After applying the price indexing procedure, the code recalculates
% the F and Y to be used in the SDA.

YP=sum(BN);
location0Ys=find(YP==0);
YP(location0Ys)=1;
FP=BN./repmat(YP,j,1);
YP(location0Ys)=0;

end
```

SDA.m

```
function [DeltaE,DeltaV,DeltaL,DeltaB,DeltaY ] = SDA(e1,v1,A1,B1,Y1,ind1,e2,v2,A2,B2
% PURPOSE: Compute the Structural Decomposition Analysis by using
% the identities in the IO matrix and the specific input per unit of output to be
% analyzed.
%
% This program has been developed to apply the Structural Decomposition
% Analysis. The program uses matrices of technical coefficients(A) and final
% demand structure by category (B) and the vectors of value added (v), input
% per unit of output (e), and aggregated final demand per category (Y).
%
% ---------------------------------------------------
%  USAGE: results = SDA(e0,v0,A0,B0,Y0,ind0,e1,v1,A1,B1,Y1,ind1 )
%  where: e = specific input per unit of output (1 by n)
%         A  = Technical coefficients matrix (n by n)
%         v = value added per unit of output (1 by n)
%         B = final demand structure industries by category (n by k)
%         Y = Aggregated final demand for each category (k by 1)
%         ind = list of industries (1 by n).
%         1,2 = are indexes for region or time periods of the different components,
% that are included.
%
% ---------------------------------------------------
%  RETURNS: a structure
%         DeltaE = Changes in input efficiency
%         DeltaV = Changes in value added of production structure
```

```
%         DeltaL = Changes in input structure
%         DeltaB = Changes in Consumption structure
%         DeltaY = Changes in consumption volume
%  ------------------------------------------------
%
%  Function created by J. T. Sayago-Gomez (fall 2013) to implement the
%  structural decomposition analysis
%
% REFERENCES:
%  Miller and Blair (2009) China's Energy Intensity Change from 1987 to 2007:
% A Structural Decomposition Analysis
%
% The code begins here:
%
%
%
% The following  i f  conditional code compares the industries included in the
%  two moments or locations. There are two courses of action: First, if
% the industries have the same size and are the same type, then it proceeds
% to create the L matrices, and do the Structural decomposition Analysis.



if (length(ind1)==length(ind2))
if (ind1==ind2)
L1=inv(eye(length(ind1))-A1);
L2=inv(eye(length(ind1))-A2);

% Each component of the SDA is calculated and stored.

DeltaE=(e2'*L1*B1*Y1)/(e1'*L1*B1*Y1);
DeltaV1=(v1'*L2*B2*Y2)/(v2'*L2*B2*Y2);
DeltaV=1/DeltaV1;
DeltaL=((e2'*L2*B1*Y1)/(e2'*L1*B1*Y1))*((v1'*L1*B2*Y2)/(v1'*L2*B2*Y2));
DeltaB=((e2'*L2*B2*Y1)/(e2'*L2*B1*Y1))*((v1'*L1*B1*Y2)/(v1'*L1*B2*Y2));
DeltaY=((e2'*L2*B2*Y2)/(e2'*L2*B2*Y1))*((v1'*L1*B1*Y1)/(v2'*L1*B1*Y2));
else
% Second, if the list of industries are the same in size but are different in type,
% then the code will create a new vector that includes all the
% industries of both locations or moments. After it creates the new
% vector named indtot, it updates the two sets of data for matching industries
% and places zeros inthe empty values vectors that allow for the same industrial
% sectors, matching between units to perform the SDA. The new matrices and
% vectors use the same names and the prefix "nw"; for example, "A1" will
% become "nwA1".
```

```
indtot=union(ind1,ind2);
locindtot=1:length(indtot);
in1=(ismember(indtot,ind1)).*(locindtot);
in2=(ismember(indtot,ind2)).*(locindtot);

sFA1=size(B1);
sFA2=size(B2);

nwV1=zeros(length(indtot),1);
nwV2=zeros(length(indtot),1);
nwB1=zeros(length(indtot),sFA1(2));
nwB2=zeros(length(indtot),sFA2(2));
nwe1=zeros(length(indtot),1);
nwe2=zeros(length(indtot),1);
nwA1=zeros(length(indtot),length(indtot));
nwA2=zeros(length(indtot),length(indtot));

in1(in1==0)=[];
in2(in2==0)=[];


SM1=size(A1);
SM2=size(A2);

nwA1(in1,in1)=A1;
nwA2(in2,in2)=A2;
nwv1(in1,1)=v1;
nwv2(in2,1)=v2;
nwB1(in1,:)=B1;
nwB2(in2,:)=B2;
nwe1(in1,1)=e1;
nwe2(in2,1)=e2;

L1=inv(eye(length(indtot))-nwA1);

L2=inv(eye(length(indtot))-nwA2);


% Each component of the SDA is calculated and stored.

DeltaE=(nwe2'*L1*nwB1*Y1')/(nwe1'*L1*nwB1*Y1');
DeltaV1=(nwv1'*L2*nwB2*Y2')/(nwv2'*L2*nwB2*Y2');
DeltaV=1/DeltaV1;
DeltaL=((nwe2'*L2*nwB1*Y1')/(nwe2'*L1*nwB1*Y1'))*((nwv1'*L1*nwB2*Y2')/(nwv1'*L2*nwB2
```

```
DeltaB=((nwe2'*L2*nwB2*Y1')/(nwe2'*L2*nwB1*Y1'))*((nwv1'*L1*nwB1*Y2')/(nwv1'*L1*nwB2
DeltaY=((nwe2'*L2*nwB2*Y2')/(nwe2'*L2*nwB2*Y1'))*((nwv1'*L1*nwB1*Y1')/(nwv2'*L1*nwB1

end

else
% if the industries are not the same size, they will not be classified as the same i
% a new vector will be created that includes all the industries
% of both locations or moments. After it creates the new
% vector named indtot, it updates the two sets of data with matching industries
% and places zeros in the empty values vectors that allow us to have the same indust
% sectors, matching between units to perform the SDA. The new matrices and
% vectors use the same names and the prefix "nw", for example "A1" will
% become "nwA1".

indtot=union(ind1,ind2);
locindtot=1:length(indtot);
in1=(ismember(indtot,ind1)).*(locindtot);
in2=(ismember(indtot,ind2)).*(locindtot);

sFA1=size(B1);
sFA2=size(B2);

nwV1=zeros(length(indtot),1);
nwV2=zeros(length(indtot),1);
nwB1=zeros(length(indtot),sFA1(2));
nwB2=zeros(length(indtot),sFA2(2));
nwe1=zeros(length(indtot),1);
nwe2=zeros(length(indtot),1);
nwA1=zeros(length(indtot),length(indtot));
nwA2=zeros(length(indtot),length(indtot));

in1(in1==0)=[];
in2(in2==0)=[];


SM1=size(A1);
SM2=size(A2);

nwA1(in1,in1)=A1;
nwA2(in2,in2)=A2;
nwv1(in1,1)=v1;
nwv2(in2,1)=v2;
nwB1(in1,:)=B1;
nwB2(in2,:)=B2;
```

```
nwe1(in1,1)=e1;
nwe2(in2,1)=e2;

L1=inv(eye(length(indtot))-nwA1);

L2=inv(eye(length(indtot))-nwA2);

% Each component of the SDA is calculated and stored.

DeltaE=(nwe2'*L1*nwB1*Y1')/(nwe1'*L1*nwB1*Y1');
DeltaV1=(nwv1'*L2*nwB2*Y2')/(nwv2'*L2*nwB2*Y2');
DeltaV=1/DeltaV1;
DeltaL=((nwe2'*L2*nwB1*Y1')/(nwe2'*L1*nwB1*Y1'))*((nwv1'*L1*nwB2*Y2')/(nwv1'*L2*nwB2
DeltaB=((nwe2'*L2*nwB2*Y1')/(nwe2'*L2*nwB1*Y1'))*((nwv1'*L1*nwB1*Y2')/(nwv1'*L1*nwB2
DeltaY=((nwe2'*L2*nwB2*Y2')/(nwe2'*L2*nwB2*Y1'))*((nwv1'*L1*nwB1*Y1')/(nwv2'*L1*nwB1

end
```

References

Dietzenbacher, E. and Hoen, A. R. (1998). Deflation of input-output tables from the user's point of view: A heuristic approach. *Review of Income and Wealth*, 44(1):111–122.

Jackson, R. and Schwarm, W. (2011). Accounting foundations for interregional commodity-by-industry input-output models. *Letters in Spatial and Resource Sciences*, 4(3):187–196.

Miller, R. and Blair, P. (2009). *Input-Output Analysis: Foundations and Extensions*. Cambridge University Press.

Yang, L. and Lahr, M. L. (2010). Sources of Chinese labor productivity growth: A structural decomposition analysis, 1987-2005. *China Economic Review*, 21(4):557–570.

Zhang, H. and Lahr, M. L. (2014). China's energy consumption change from 1987 to 2007: A multi-regional structural decomposition analysis. *Energy Policy*, 67(C):682–693.