Interdyme Basics

VAM.CFG File for the TINY Model

```
1995 2010
#name   Rows Cols Lags Row names Column names Comment
FM         8    8  0  sectors.ttl sectors.ttl #Input-output flow matrix
AM         8    8  0  sectors.ttl sectors.ttl #Input-output coeff matrix
LINV       8    8  0  sectors.ttl sectors.ttl # Leontief inverse
out        8    1  0  sectors.ttl # Output
pce        8    1  0  sectors.ttl # Personal consumption expenditure
gov        8    1  0  sectors.ttl # Government spending
inv        8    1  0  sectors.ttl # Investment
ex         8    1  0  sectors.ttl # Exports
im         8    1  0  sectors.ttl # Imports
fd         8    1  0  sectors.ttl # Total final demand
dep        8    1  0  sectors.ttl # Depreciation
lab        8    1  0  sectors.ttl # Labor income
cap        8    1  0  sectors.ttl # Capital income
ind        8    1  0  sectors.ttl # Indirect taxes
depc       8    1  0  sectors.ttl # Depreciation coef
labc       8    1  0  sectors.ttl # Labor income coef
capc       8    1  0  sectors.ttl # Capital income coef
indc       8    1  0  sectors.ttl # Indirect taxes coef
x          8    1  0  sectors.ttl # Working space
y          8    1  0  sectors.ttl # Working space
```

To create a vam file from a vam configuration file the command in G is

vamcreate <vam configuration file> <vam file>

To create the vam file HIST.VAM from the configuration file VAM.CFG, the command is

vamcreate vam.cfg hist

The newly created vam file is all zero. To work with it, we assign it as a bank:

vam <filename> <letter name of bank>

For example,

vam hist b

Most commands for working with VAM files use the default VAM file.
It is specified by the "dvam" command

dvam <letter name of bank>

For example

dvam b

The usual ways to introduce data into a VAM file are with the

matin command for matrices

and the

vmatdat command for vectors.

The next three slides show examples for TINY.

The Flows.dat File for Introducing the Input-Output Flow Matrix into the VAM File

matin FM 1995 1 8 1 8 15

#	Agricul	Mining	Elect	Mfg	Commerce	Transp	Services	Govt
Agriculture	20	1	0	100	5	0	2	0
Mining	4	3	20	15	2	1	2	0
Electricity	6	4	10	40	20	10	25	0
Manufacturing	20	10	4	60	25	18	20	0
Commerce	2	1	1	10	2	3	6	0
Transportation	2	1	5	17	3	2	5	0
Services	6	3	8	45	20	5	20	0
Government	0	0	0	0	0	0	0	0

The FD.dat File for Introducing the Final Demands into the VAM File

vmatdata c 5 1 1 8 15

c means the vectors are in columns
there are 5 columns
there is 1 year
each column runs from sector 1 to 8
skip 15 spaces before data

1995 pce gov inv ex im

The year is 1995
The vector names are pce, gov, inv, ex, im

#	PersCon	Gov	Invest	Exports	Imports
Agriculture	**15**	**1**	**0**	**40**	**-20**
Mining	**2**	**1**	**0**	**10**	**-10**
Electricity	**80**	**10**	**0**	**0**	**0**
Manufacturing	**400**	**80**	**200**	**120**	**-170**
Commerce	**350**	**10**	**6**	**10**	**0**
Transportation	**130**	**20**	**8**	**5**	**0**
Services	**500**	**40**	**10**	**30**	**-20**
Government	**0**	**150**	**0**	**0**	**0**

The VA.DAT File for Introducing the Value-added Vectors

```
vmatdata r 4 1  1 8 15
# r means the vectors are in rows
# There are 4 vectors
# For 1 year
# The first sector is 1 and the last 8
# Skip 15 places before beginning to read free-form data
1995  dep lab cap ind
#                 1   2      3      4      5      6      7      8
Depreciation     11   5     60    130     35     40     25      0
Labor            65  20     21    260    140     97    485    150
Capital          20   2     56     60     40     12     59      0
Indirect tax      8   0     20     50    109     10     18      0
```

G commands to create the VAM file and load the data into it:

```
# Create and load the VAM file for TINY
vamcreate vam.cfg hist
vam hist b
dvam b
# Bring in the intermediate flow matrix
add flows.dat
# Bring in the final demand vectors
add fd.dat
# Bring in the value added vectors
add va.dat
```

DISPLAY OF MATRIX DATA

```
# Show the FM Matrix for year 1995
show FM   y   1995
# Show FM matrix row 2 for all years
show FM r 2
# show FM matrix column 5 for all years
show FM c 5
# Show ind (indirect tax) vector for all years
show ind     # Display the indirect tax vector
#Show pce vector from bank b
show b.pce   # Display the personal consumption expenditure vector
```

CALCULATE INPUT-OUTPUT COEFFICIENTS

Add up the intermediate rows

getsum FM r out

Add on the final demand vectors to get total output

vc out = out+pce+gov+inv+ex+im

show b.out

Copy the flow matrix, stored in FM, to AM

fdates 1995 1995

mcopy b.AM = b.FM

Can be simplified to

mcopy AM FM

Calculate input-output coefficients

coef AM out

show AM y 1995

Create value-added coefficient vectors.

vc labc = lab/out

vc capc = cap/out

vc indc = ind/out

Set fdates back to the entire range of the VAM file.

fdates 1995 2010

COPY THE COEFFICIENT MATRICES TO OTHER YEARS

We use the *index* command to move these coefficient matrix
and vectors to other years.

index <base year> <guide series> <matrix or vector>

It operates over the range specified by the current value of the fdates.
Since we just want to copy the coefficients to all the years, our guide series
will be simply a series of 1's, which we shall call *one*. Here are the commands:

```
# Copy the 1995 AM matrix into 1996 -  2010
dfreq 1
f one = 1.
index 1995 one AM
        index 1995 one depc
        index 1995 one labc
        index 1995 one capc
        index 1995 one indc
show AM c 1
```

MAKE UP A PATH FOR FUTURE FINAL DEMAND

```
# Create a time trend
f time = @cum(time,one,0)
# Make all final demand vectors grow by 3 percent per year
f g03 = @exp(.03*(time-1))
gr  g03
# Create waves, the guide series for the investment vector
f  waves = g03 + .3*@sin(time-1)
gr  waves
index 1995 g03 pce
index 1995 g03 gov
index 1995 waves inv
index 1995 g03 ex
index 1995 g03 im
show inv
# Add up the final demands
vc fd = pce+gov+inv+ex+im
show fd
```

THE LEONTIEF INVERSE COMMAND

The new command

linv <square matrix> [year]

converts the square matrix into its Leontief inverse. If the optional year is omitted, it works over the *fdates* range.

USE THE NEW LINV COMMAND TO TAKE THE LEONTIEF INVERSE

Take the Leontief inverse of the A matrix
mcopy LINV = AM
linv LINV
show LINV y 1995

Compute total outputs
vc out = LINV*fd
show b.out

With the outputs known, we can compute the implied value-added of each type by each industry with the following commands. In them, the *vc* command will recognize that the dimensions of the vectors on the right are such that element-by-element multiplication makes sense and perform it.

Compute Value added
The following are element-by-element multiplication
vc dep = depc*out
vc lab = labc*out
vc cap = capc*out
vc ind = indc*out
show lab

MAKE INDUSTRY LEVEL GRAPHS WITH fadd COMMAND

fadd graphs.fad sectors.ttl

Where graphs.fad is

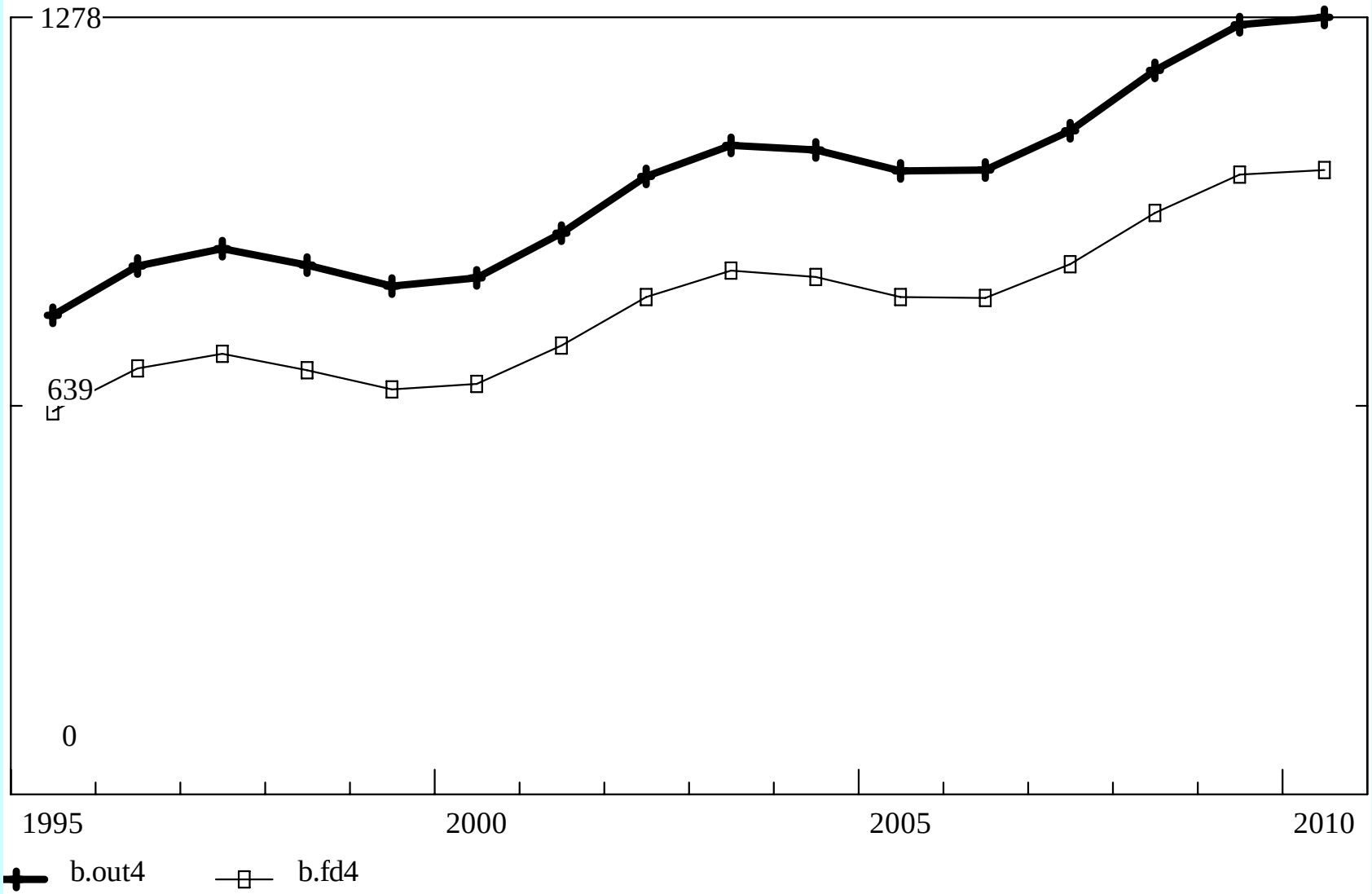
vr 0
ti %3 %5
subti Output and Final demand
gname out%3
gr b.out%3 b.fd%3
subti Depreciation,Labor income, Capital income, Indirect taxes
gname va%3
gr b.dep%3 b.lab%3 b.cap%3 b.ind%3
ti
subti

And sectors is:

```
Agricul     ;1   e   "Agriculture"
Mining      ;2   e   "Mining and quarrying"
Elect       ;3   e   "Electricity and gas"
Mfg         ;4   e   "Manufacturing"
Commerce    ;5   e   "Commerce"
Transport   ;6   e   "Transportation"
Services    ;7   e   "Services"
Government  ;8   e   "Government"
```

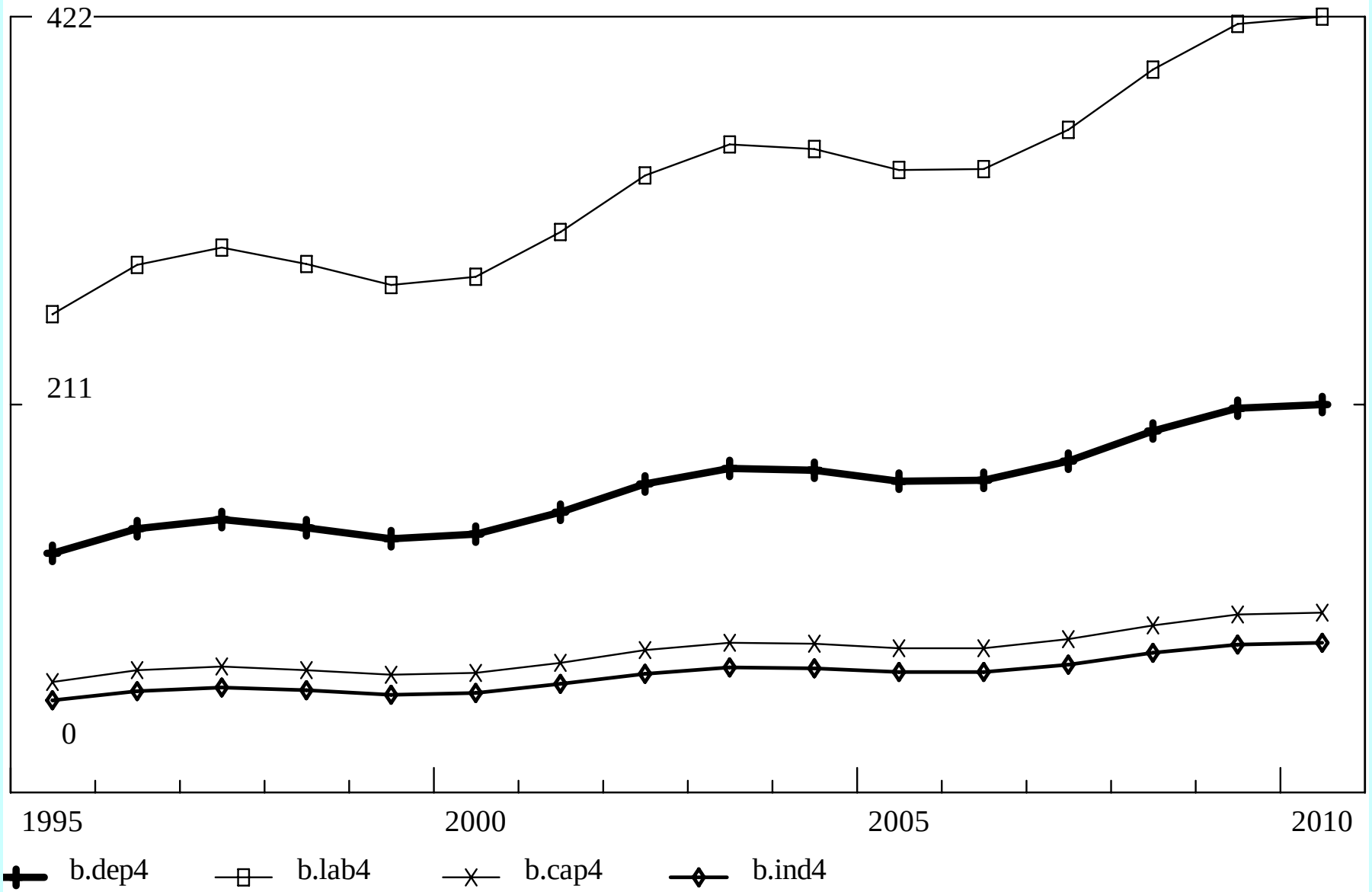
4 Manufacturing

Output and Final demand



4 Manufacturing

Depreciation, Labor income, Capital income, Indirect taxes



OTHER MATRIX COMMANDS NOW IN G

minv A	converts A into its inverse
madd A = B + C	adds B and C and stores in A
madd A = B - C	subtracts C from B and stores result in A
mmul A = B*C	multiply B and C and store result in A
mmul A = B’C	multiplies B transpose by C and stores result in A
mmul A = B&C	does element-by-element multiplication of B and C and stores in A
mmul A = B/C	element-by-element division of B by C stored in A
mtrans A B	the transpose of B is stored in A

MAKING TABLES WITH A MULTISECTORAL MODEL

The TINY.STB File – top third

```
\dates 1995 2000 2005 2010 1995-2000 2000-2005 2005-2010
\pages off
\noformat
\title TINY G-ONLY MODEL, ILLUSTRATIVE FORECAST

; out  Output of Industries
&
out1 ;1 Agriculture
out2 ;2 Mining and quarrying
out3 ;3 Electricity and gas
out4 ;4 Manufacturing
out5 ;5 Commerce
out6 ;6 Transportation
out7 ;7 Services
out8 ;8 Government
;
```

MAKING TABLES WITH A MULTISECTORAL MODEL

The TINY.STB File – middle third

```
\add tiny.tab pce "Personal Consumption Expenditure"
;
\add tiny.tab gov "Government Expenditures"
;
\add tiny.tab inv "Investment by Supplying Industry"
;
```

The TINY.TAB File

```
; %1 %2
&
%11 ;1 Agriculture
%12 ;2 Mining and quarrying
%13 ;3 Electricity and gas
%14 ;4 Manufacturing
%15 ;5 Commerce
%16 ;6 Transportation
%17 ;7 Services
%18 ;8 Government
```

MAKING TABLES WITH A MULTISECTORAL MODEL

The TINY.STB File – bottom third

```
# The next line forces a new page
*
\matcfg Matlist.cfg
\center Matrix Listing
\row
\cutoff .001
\matlist 1-8
```

TINY G-ONLY MODEL, ILLUSTRATIVE FORECAST

Matrix Listing

Seller: 1 Agriculture

	1995	2000	2005	2010	95-00	00-05	05-10
Sales to Intermediate							
1 Agriculture	20.0	22.1	26.3	32.2	2.0	3.5	4.0
2 Mining and quarrying	1.0	1.1	1.3	1.6	2.3	3.4	3.7
4 Manufacturing	100.0	107.9	130.3	162.4	1.5	3.8	4.4
5 Commerce	5.0	5.8	6.7	7.9	2.9	3.1	3.1
7 Services	2.0	2.3	2.7	3.2	2.8	3.1	3.2
SUM: Intermediate	128.0	139.2	167.4	207.2	1.7	3.7	4.3
Sales to Final Demand							
Personal consumption expenditure	15.0	17.4	20.2	23.5	3.0	3.0	3.0
Government consumption	1.0	1.2	1.3	1.6	3.0	3.0	3.0
Exports	40.0	46.5	54.0	62.7	3.0	3.0	3.0
Imports	-20.0	-23.2	-27.0	-31.4	3.0	3.0	3.0
Output	164.0	181.0	216.0	263.7	2.0	3.5	4.0

Seller: 2 Mining and quarrying

	1995	2000	2005	2010	95-00	00-05	05-10
Sales to Intermediate							
1 Agriculture	4.0	4.4	5.3	6.4	2.0	3.5	4.0
2 Mining and quarrying	3.0	3.4	4.0	4.8	2.3	3.4	3.7
3 Electricity and gas	20.0	22.8	26.7	31.7	2.6	3.2	3.4
4 Manufacturing	15.0	16.2	19.5	24.4	1.5	3.8	4.4
5 Commerce	2.0	2.3	2.7	3.1	2.9	3.1	3.1
6 Transportation	1.0	1.1	1.3	1.6	2.6	3.2	3.4
7 Services	2.0	2.3	2.7	3.2	2.8	3.1	3.2
SUM: Intermediate	47.0	52.5	62.2	75.1	2.2	3.4	3.8
Sales to Final Demand							
Personal consumption expenditure	2.0	2.3	2.7	3.1	3.0	3.0	3.0
Government consumption	1.0	1.2	1.3	1.6	3.0	3.0	3.0
Exports	10.0	11.6	13.5	15.7	3.0	3.0	3.0
Imports	-10.0	-11.6	-13.5	-15.7	3.0	3.0	3.0
Output	50.0	56.0	66.3	79.8	2.3	3.4	3.7

The MATLIST.CFG File for TINY

```
Matrix listing identity;out=AM*out+pce+gov+inv+ex+im
# Title file name for the rows of out, the lefthand side vector
out; "sectors.ttl"
# Title file names for matrix columns
AM; "sectors.ttl"
# headers for each term
header for out;    "Output"
header for AM*out; "Intermediate"
header for pce;    "Personal consumption expenditure"
header for gov;    "Government consumption"
header for inv;    "Investment"
header for ex;     "Exports"
header for im;     "Imports"
```

And finally,

Interdyme ….

USER.H

```
// USER.H -- Put here any includes that refer to the user model
// From DYME.CFG and opening screen:
GLOBAL char RunTitle[80],CfgFileName[80],VamFileName[80],
    GbankName[80],VecFixFileName[80],MacFixFileName[80];
GLOBAL char* outfix; // Determines how Seidel will determine output
// Vector declaration:
    GLOBAL Vector out,pce,gov,inv,ex,im,fd,
    dep,depc,lab,labc,cap,capc,ind,indc,x,y;

// Matrix declaration
    GLOBAL Matrix AM;

    GLOBAL IVector triang;
```

In LOOP() Function of Model.cpp

// Vectors

out.r("out");pce.r("pce");gov.r("gov");
inv.r("inv");ex.r("ex");im.r("im");fd.r("fd");dep.r("dep");
depc.r("depc"),lab.r("lab");labc.r("labc");cap.r("cap");
capc.r("capc"),ind.r("ind");indc.r("indc");x.r("x");
y.r("y");

// Matrices
AM.r("AM");

INTERDYME CODE FOR TINY

```
for (t = godate; t<= stopdate; t++) {
        // Load all vectors and matrices.
        load(t);
        // Particular to TINY:
        fd = pce + gov + inv + ex + im;
        Seidel(AM, out, fd, triang, toler);
        dep = ebemul(depc,out);
        lab = ebemul(labc,out);
        cap = ebemul(capc,out);
        ind = ebemul(indc,out);

        // General end of the spin() function:
        if(MaxFlag == 'y')
            shiftback(t);
        else{
            // Store the values of vectors and matrices.
            store(t);
            printf("\n");
            }
        }
```