# THE DEMOGRAPHIC PROJECTIONS MODEL

**INFORUM**

**Interindustry Forecasting Project**

**University of Maryland**

**College Park, MD**

**July 1996**

# The Demographic Projections Model
# Version 2.0

## Introduction

The Demographic Projections Model (DPM) essentially consists of three parts. There is the core part of the model where the actual yearly population projections are made. There is the pure aggregation of the data to form macroeconomic variables that can be used in LIFT. And, there is the creation of the ancillary variables that use DPM projections in order to obtain forecasts of demographic variables other than population.

## Core Element of DPM

The core element of DPM is the actual projections from 1995 to 2050 of the population by gender and age. To create these projections there are a number of exogenous variables that need to be specified:

- population by gender and age for a base year;
- fertility rates for females ages fourteen to forty-nine from 1995 to 2050;
- survival rates for males and females for each age from 1995 to 2050;
- net immigration rates for males and females by age from 1995 to 2050.

We used Census Bureau estimates of these four sets of exogenous variables. Combining them and using the cohort component technique of demographic projections implies a set of equations projecting the number of people at any age for both males and females:

- female2 = female1[t-1]*srtf2 + 0.5*(imf1 + imf2)*(1 + srtf2)*0.5.

Female2 is the population of females that are one year old. Female1[t-1] is the population of females between the ages zero and one last year. Srtf2 is the survival rate of one-year old females. Imf1, and imf2 are the net immigrations of children less than one year old and one-year olds, respectively.

The last term needs a bit of explaining. The influx of immigrants is assumed to be evenly distributed over the year, and the data correspond to the age when entering the country. Thus, some of the immigrants who enter and are categorized as imf1, female immigrant under the age of one, will actually be imf2, female immigrant over the age one, when July first rolls around. Therefore, we take the average of the two groups. Since the influx is assumed to be evenly distributed over the year,

the survival rate needs to be adjusted in order to account for immigrants who enter mid year. When the immigrants enter they obviously are alive, hence the one plus the survival rate. The term (1+srtf2)*.05 can be thought of as the survival rate of immigrants, who, on average, enter the country at mid-year.

Finally, we needed to obtain a projection of the population of zero-year olds (that is infants):

- female1 = (0.488)*[fert15*female15+...+fert50*female50]*(0.001)*srtf1.

Where fert15*female15 is the fertility rate of fourteen-year olds multiplied by the population of fourteen-year old females. Summing the births to each age group gives us the total number of births. However, the fertility rates are per thousand women, hence we needed to divide by 1000. Finally, multiplying the total number of births by the percentage that are female and the survival rate for zero year old girls yields the population of females age zero.

**Aggregation of Variables**

To use the data that is obtained from the core part of the model it is necessary to aggregate the variables into groups that can be used in a model such as LIFT. There are essentially three different types of aggregate variables that are necessary.

- Total population, pt, is equal to the sum over age and gender.

- Total population for each gender, ftot and mtot, is equal to the sum over the ages for each gender.

- Population groups for each gender, pgf1-18 and pgm1-18, are the sum of five-year age groupings until the last, which is the group of people eighty-five years and older: (note: these groupings correspond to the current gpop groupings only there is more detail),

```
pgm1 = male1 + male2 + male3 + male4 + male5;
.    = .;
.    = .;
.    = .;
pgm17= male81 + male82 + male83 + male84 + male85;
pgm18= [male86 +...+male110].
```

Finally in the aggregation part of DPM, we put together the historical populations from 1950 to 1994 with the projections. We now have a data bank from 1950 to 2050 of population by age and gender.

**Ancillary Variables**

There are other variables that need to be estimated for LIFT 2050. These variables are the

demographic ones about households: the number of households, the number of household heads, number of households with two earners, etc.. Also, demographic variables such as the total working age population need to be estimated.

**Changes to the Model**

This version of the model has a number of new features. The model is made with the InterDyme modeling framework. This new framework allows the user to apply fixes more easily. The model is also far easier to run. The new framework allows the user to use the VAM software, similar to G, to manipulate and view the data in a spread-sheet format. Finally, the new version of the model has been updated with the Census estimates for population through 1994 and the new survival rates, fertility rates, and immigration levels.

## HOW TO USE THE DPM

Running the Demographic Projections Model and preparing data for use in the U.S. model LIFT can be done in four major steps. They are:

Step 1. Set up simulation scenario

Step 2. Run DPM

Step 3. Check results using the Compare program or VAM

Step 4. Put the DPM variables in the proper files for LIFT

The following are detailed procedures for each step. (Note that if you just want to run the base forecast, proceed to Step 2. Otherwise, follow Step 1 to set up an alternative scenario.)

### Step 1. Set up simulation scenario

Setting up a simulation in which you change the survival rate, fertility rate and immigration populations can be done in a variety of ways. The vectors.vfx file contains the data which can be varied for simulations. The file may be edited directly using any program editor. Simply replace any piece of data "by hand" with the desired simulation data. There are a number of different ways to apply fixes. The base version of the vectors.vfx file only uses multiplicative fixes and the fixes are unnecessary- they multiply the variable by one.

The following is a portion of the vectors.vfx file.

```
# GROUP DEFINITION
# Groups are defined in the following way. Note: age 1 is under age less than one.
grp allofem
     1-110
grp child
     1-16
```

The "#" sign indicates that the line is a comment line. The first step in making a simulation is to determine which ages you will apply the fix to. Then you must define a group that encompasses the ages you are interested in. In the example in the vectors.vfx file we have defined the group allofem to be all ages. Similarly, we have defined the group child to be all children under the age of 16. The grp command allows you to exempt ages from the group. For example the following group command will create a group called depend which has all the population under 18 and over 85.

```
grp depend
     1-110 (19-86)
```

**N.B.** The individual ages are referenced by the age plus one; the male population age zero is male1. Similarly, 18 year old males are male19.

Groups can also be defined by referencing the individual ages. For example, the following will create a group that is of 0, 2, and 4 year olds.

```
grp even
    1 3 5
```

Once you have defined the groups of interest you can apply the fixes to the variables. The following is an example of a multiplicative fix to the survival rates for males and females of all ages, where the group all was defined above. The fix in the example below will reduce the survival rate for all males and all women except those of childbearing age by 10 percent in 1994 and 30 percent in 2050, where the multiplier is linearly interpolated between 1994 and 2050. In order to avoid a survival rate greater than one the fix is actually applied to one minus the survival rate (1-srtm). Therefore a multiplier greater than one will reduce the survival rate.

```
mul srtm :allofem
    1994 1.1
    2050 1.3;
grp nchld
    :all (16-50)
mul srtf :nchld
    1994 1.1
    2050 1.3;
```

Another way to apply a fix in the vectors.vfx file is to use the override command which will replace the value with the value you supply. For example the following will override the value for immigration of female 20 year olds with the value you supply.

```
ovr immf 21
    1994   80.000
    2010 110.000
    2050 100.000;
```

Alternatively we could have added a constant term to the value of immf21.

```
cta immf 21
    1994 20.000
    2050 20.000;
```

In the next section the macro-variable fixes are illustrated. The fixes can be used in the same way for veectors but they must reference a group. In order to fix the macro-variables we need to edit the

file macrofix.mfx. We can use an index to adjust the values in the forecast. The following will move the share of the military population that is male by an index starting in 1994.

```
ind mshmil
    1994 1.0 1.2  1.1  1.15  1.2
    1999 1.2 1.3 1.25 1.25 1.3;
```

We can also fix a variable by using a growth rate fix. The following will fix the total military population using a growth rate fix starting in 1994 and linearly interpolating the values before 2050..

```
gro tmilpop
    1995 3.5
    2050 2.1;
```

Finally, we can apply a step wise growth rate fix (stp). In this case the growth rate in the total military population is 3.5 from 1994 until 2010 when it steps down to 2.9 until 2050, where it steps down to 2.1. It should be noted that the stp function requires a value for the final period.

```
stp tmilpop
    1995 3.5
    2010 2.9
    2050 2.1;
```

One last note on using the fix software commands. There must be a colon before the group name and a semicolon after the last fix on that variable, also the group name "all" is illegal. There need not be any fixes on a variable for the model to run. Vectors.vfx is provided to be illustrative and list some of the variables that can be fixed, with the exception of the fix on mshmil in the macrofix.mfx file the fixes can be ignored. Once the input file is ready you can create the fixes by typing "fixer -z" from the DOS prompt. For more details on the fixer programs see Appendix B.

Before you begin, you may want to make a back-up copy of the following files in case any mistakes are made in editing them for various scenarios.

vectors.vfx
macrofix.mfx

**Step 2. Run DPM**

Type "run" at the dos prompt, and the model will run and create five files. The five files are: dyme.bnk, dyme.ind, dyme.vam, dpmmac.fix, and dpmfdv.fix. Then copy the five files made by DPM and the hist.stb stub file to the name of the simulation. For example:

```
copy dyme.bnk    base.bnk
copy dyme.ind    base.ind
copy dyme.vam    base.vam
copy dpmmac.fix basemac.fix
copy dpmfdv.fix   basefdv.fix
copy hist.stb        base.stb       (The stub file that allows you to use the look command.)
```

**Step 3. Check results using Compare program or VAM**

**Compare**

The following files with the extension .stb can be used with the Compare program to make tables. You may make a table for just one scenario, such as the base case, or compare the base case to an alternative scenario. (Compare can "compare" up to 4 scenarios.) Currently the .stb files are set up to print data for the years 1994, 2000, 2010, 2020, 2030, 2040, and 2050 with growth rates for those periods. However, they may be changed easily by editing the .stb file with any editor program.

| | |
|---|---|
| popmft.stb | population for males, females, and the total |
| groupop.stb | population groups for males and females |
| popimmf.stb | immigration for males and females |
| fertrate.stb | fertility rates |
| survrate.stb | survival rates for males and females |
| arminst.stb | population in armed forces and institutions |
| summary.stb | table that summarizes simulation results |

For example, if you want to make a table of fertility rates, we will use the fertrate.stb file as follows:

Type <compare> at the DOS prompt to begin. You will then be asked a series of questions (denoted in bold print):

**How many alternatives?** Enter **1** if you want tables from only 1 bank, enter **2** if you want to compare data from 2 different banks.

**Bank type (w=workspace, c=compressed, h=hashed bank, d=dirfor, v=vam):** Enter **w** for a workspace bank.

**Rootname of workspace bank:** enter the name of your bank for example, **run1**

**Show alternatives as (a) actual, (d) deviations, or (p) percentages?** Enter **a** if you want the actual data, **d** if you want the actual change from the previous period, or **p** if

you want the percentage change from the previous period.

**With what stub?** Enter **fertrate.stb**

**Name of output file:** enter the name under which your table shall be stored, for example, **frate.out**

**Now making table as file frate.out.** The table will be complete when the DOS prompt returns to the screen. Then simply type <**print frate.out**>, for example.

**VAM**

The Vam program is used to modify and look at data. The data that is contained in a Vam file are listed in the Vam.cfg file. The vamfile currently contains all the vectors listed in the vam.cfg file. This set of vectors is not all the variables that the model creates. Some of the variables that the model creates are put into the g-bank dyme.bnk. Therefore to see all the variables you must use VAM. The Vam program can be invoked with the following command:

vam [-v<Vam file>] [-i<Input File>] [-a<Action>]

Items inclosed in < > are mandatory; items enclosed in [ ] are optional. Items not enclosed in either must appear exactly as written; the words inside < > and [ ] are intended to describe a class of entries. In no case would the < > or [ ] appear on the actual command. The name of the vam file to be worked on or displayed is specified (without the .vam extension) by the -v option. The default value of VamFile is "hist", so that "hist.vam" is the default DOS name of the vam file worked on or displayed. The InputFile option allows one to specify on the command line the name of a file containing commands to Vam. If no input file is named, Vam defaults to a "|" prompt from which it will accept any command. The <action> may be 'c' for "create" or 'u' for update. The default action is update if the output file exists and create if it does not. Examples of invocation are

| | |
|---|---|
| vam | Opens hist.vam if it exists, otherwise creates it. Gives \| prompt. |
| vam -vDyme | Opens dyme.vam if it exists, otherwise creates it. Gives \| prompt. |
| vam -iInitial | Opens .vam; executes commands in Initial file. |

The hist.vam file contains the historical and exogenous forecast values of all of the vectors and matrices in the model. This file will NOT be a G databank but rather the transpose of one. That is, it will not have the time series for a particular variable all together; rather it will have first the values of all variables in the first year, then the values of all variables in the second year. This order speeds the operation of the forecasting program. Creating, updating and modifying of this file is the main work of Vam.

**G Commands in Vam**

Like G, Vam has a workspace bank and may have assigned G banks as well. Nearly all G commands not directly concerned with regression are available in Vam. Thus Vam has such G commands as

Data-bank related commands
bank, wsb, hbk, cbk, look, btitle
graphics commands
gr, lgr, mgr, sgr, gdates, title, subtitle, vaxtitle, vaxlabel, legend, line, annline, anntext, annsave, printer, and autoprint
utility
dos, ed, save, tdates

All of these work in Vam just as they do in G. They will not be described further here. Some other commands are the same or similar to G commands but are described here because of minor differences or for emphasis.

Vam does NOT have such G regression commands as
r, recur, hl, bj, sur, nl, nlp, con, sma, norm, chow, comcoef.

**The Vam Configuration File, Vam.cfg**

Like G, Vam requires a configuration file, vam.cfg, which contains some very vital information -- namely, the starting date and ending dates for all the vam files, and for each vector or matrix in the model:
the name of the vector or matrix,
its number of rows,
its number of columns,
the number of lagged values with which it is used in the model, or a p if it is a sparse matrix which is to be "packed", so that only non-zero elements are stored.
the file name of the titles for the rows of a vector or matrix,
the file name of the titles for the columns of a matrix.
a # followed by a comment usually explaining the economic nature of the variable and possibly the name of the file with historical data.

The format is free. Here is part of the vam.cfg for the DPM model:

```
1950 2050 # Starting and ending years of vam.bin
# VAM file for DPM Version 2.0
#Name |Number of  |Files of titles of|   Description
#     |row col lag|  rows   cols     |
#
# Matrices (There are none yet)
# Vectors
scrm    110 1 1 ages.stb      # Male Scratch Vector
scrf    110 1 1 ages.stb      # Female Scratch Vector
male    110 1 1 ages.stb      # Male population by age, thousands
```

```
female  110 1 1 ages.stb      # Female population by age, thousands
mafo    110 1 1 ages.stb      # Male Armed Forces Overseas, thousands
fafo    110 1 1 ages.stb      # Female Armed Forces Overseas, thousands
fert    110 1 0 ages.stb      # Age-specific fertility rates
drtm    110 1 0 ages.stb      # Probability of death given reaching age I,
drtf    110 1 0 ages.stb      # Probability of death given reaching age I,
srtm    110 1 0 ages.stb      # Probability of surviving to age I, male
srtf    110 1 0 ages.stb      # Probability of surviving to age I, female
```

Note that the starting and ending dates do not control when a particular run of the model starts or stops, but define the range of the vam files. The model in the example cannot start earlier than 1950 or run past 2050, but it certainly does not have to run over the whole span on each simulation. In this example, the model will have a vector named "srtm" which will have 110 rows and 1 column, as indicated by the first two numbers on the line; only current year values of this vector are needed in computing the results of the model in the present year, so the number of lags used, the third number on the line, is 0. The next item on the line is the name of the file with the sector names to be used for the spreadsheet-like displays of the vector. Everything following the # is a comment.

**Vam Commands**

Like G, when Vam starts it gives a prompt to indicate that it is ready to take commands. Instead of G's ":" prompt, however, Vam uses a "|" to remind the us that we are in Vam, not G. From the "|" prompt we can give Vam any command in its repertoire.

We will explain the Vam commands in three groups, (1) commands for introducing data, (2) commands for displaying data, modifying it element-by-element, and getting help, and (3) commands for "sweeping" modifications of data. Before even turning to these, we begin with the one command, borrowed from G, absolutely essential for every Vam user to know:

quit
q

Exit from Vam.

The full command is shown on the first line; beneath it is shown its minimal abbreviation. Anything in between will work. In this case, q, qu, qui, and quit all accomplish the same thing.

**Commands for taking commands from files**

Like G, Vam expects that most data will be introduced from files. Those files will usually begin with the specific command for introducing the following data. The command which the Vam user types directly to Vam is, therefore, the command to take commands and data from a specific file. As in G, this command is "add". The form is as follows.

add <filename>
ad

Take input from the named file. As in G, there can be arguments to add files and there can be add files within add files. When an argument consists of more than one word, as a sector title, it should be enclosed in quotation marks, "like this".

There is also a command combining an "add" command with a file from which to take arguments. It is:

fadd <CommandFile> <ArgumentFile>

The named CommandFile is first executed with the arguments from the first line of named ArgumentFile, then the CommandFile is executed again with the arguments from the second line of the ArgumentFile, and so on until the ArgumentFile is exhausted. The ArgumentFile is often simple a list of numbers followed by names, like this:

1 "Age zero"
2 "Age one"
etc.

The CommandFile is then a set of actions to be performed on these arguments. In this example, the CommandFile

ti Population share of %2
f popshr%1 = (male%1 +female%1)/poptot
gr popshr%1

would graph (gr) the population shares of the successive ages.

It is also possible to introduce data from G banks. Like G, Vam always has a G-bank workspace file (composed of the files vamws.ind and vamws.bnk) and may have an assigned bank. The assigned bank may be of any of the three types used by G: a standard bank (assigned with the "bank" command), a compressed bank, (assigned with the "cbk" command) or a hashed (or huge) bank (assigned by the "hbk" command). Unlike G, no default bank is automatically assigned. The normal order of searching for a variable in Vam is to look first in the workspace, then in the assigned bank, then in the .vam file. We can override this order by putting a "a." or "v." in front of the variable name. Thus

type x

will look for x in the normal order, while

type a.x

will look for x only in the assigned bank and

type v.x

will look for x only in the .vam file.

There are three commands for changing data in these banks.

f <name> = <expression>

Evaluate the expression on the right and place it in the workspace bank with the "name". The expression on the right may be any expression legal in G and involving only variables in the workspace, the assigned bank, or .vam. The program will look for the variables in the expression first in the workspace, then in the assigned bank, then in .vam. To specify that a variable is to come from the assigned bank, not the workspace, precede its name with an "a." Similarly, to specify that a variable is to come from .vam, put an "v." in front of the variable. Examples:

```
f control = male1 + male2 + male3
f sum = v.male1 + v.male2 + vmale3
f scale = control/sum
```

(If the variables appear only in the .vam file, the first two examples do the same thing.)

All of the functions available in G and documented in *The Craft of Economic Modeling* are available in Vam f and vf commands. The csum function is a new function that is documented in the section of this manual on Compare. The csum function is used to sum up a specified group of industries for a given data concept. If, for example, you wanted the variable x to equal the sum of exports for sectors 1 through 10, you would use the command

```
f x = @csum(exp, 1-10)
```

See the section on Compare for further documentation on csum.

```
vf <name> = <expression>
```

Exactly like the previous command, but the variable calculated is placed into the .vam file. Example:

```
vf male1 = scale*v.male1
```

The <name> must be a valid name for a vector in the .vam file. Note that this command provides the route for getting data from a G bank into the .vam file.

```
vc <vector_name> = <expression>
```

The vc, or vector calculation, command evaluates the expression on the right and puts the results into the named vector. The only allowed operation for a matrix is multiplication. The expression is evaluated in plain sequence in which no parentheses are supported. Neither are scalars. Example:

```
vc out = am*out + out
```

The <vector_name> must be a valid vector name in the .vam file, as must all names in the expression.

```
zap
```

Clear the workspace bank.

**Commands for the display and minor modification of data**

type <name> [tdate1] [tdate2]
ty

As in G, display the named variable between the given dates. The tdates are remembered. The order of search for the variable is as explained for the f command. The "name" of the third element of the vector x is x3. The name of cell 7,11 of the am matrix is am7.11. The vector name "l" is reserved for the currently loaded vector. Thus, if we have just done "show VectorWithLongName", the "ty l8" will show its eighth element.

graph <name> [gdate1] [gdate2] [gdate3]
gr

This command works exactly as it does in G, except that it will also graph variables from the vam vile. It searches for variables in the way described above. The vector name "l" always refers to the currently loaded vector. Thus

gr srtm5

will graph the fifth element of the srtm vector, while

gr l5

will graph the fifteenth element of the loaded vector, whatever it is.

As mentioned above, Vam has all of G's graphics commands. These include gr, mgr, lgr, sgr, gdates, title, subtitle, vaxtitle, vaxlabel, legend, line, annline, anntext, annsave, autoprint, printer. Interactive annotation and printing work exactly as in G. These features are not further documented here, but are covered by the on-line help available by typing "help" in Vam.

look
Exactly like the G "look" command for looking at a data bank.

help
Display on-line help. The screen works just like the G on-line help, but the screen includes Vam specific information and excludes G-specific information.

load <vector_name>

Pull the whole time series for the vector into a workspace. Vam can work on series individually, but doing so requires frequent, time consuming, access to the disk. If a number of items within a single vector are being worked on, it is much better to load all of that vector into memory, work on

it there, and then store it all together back to the .vam file. If in create mode, load simply zeroes out the workspace and records the name of the loaded vector. Commands which work on all of a vector or groups of elements in it can function only if that vector is loaded. Any command which accesses a vector does an implicit "load" of that vector. Example: "load lifexpf".

store

Store the currently loaded vector back to the .vam file with the modifications that have been made. Store is automatic when a new "load" or implicit "load" or a "quit" is encountered when reading from an add file. If these are encountered when taking input from the keyboard and an un-stored, modified vector is loaded, one is asked "Store? (y/n)".

show <vector_name> [sector] [date]

This command "shows" the named vector. More precisely, it displays the vector like a spread-sheet matrix with dates across the top and sector numbers down the side. The optional sector and date determine the sector and date displayed in the upper left corner of the screen. The default values are sector 1 and the beginning of the vam file. Examples:

show lifexpf
show lifexpf 5 1995

Once the display is on the screen, you can scroll through the matrix with the arrow keys. PageUp, PageDown, and Home work as expected. 'Ctrl' left arrow and 'Ctrl' right arrow work like PageLeft and PageRight would work if there were such keys.

Data may be entered in the "show" mode. Just put the cursor in the desired cell, tap the 'End' key, and enter the data. The 'Insert' key toggles between the insert mode and the typeover mode. When entering data, the left and right arrow keys function to move over digits already displayed. The 'Backspace' and 'Delete' keys work in the usual ways. Do an 'enter' to end the input to a cell. Entering data in insert mode may result in messing up the display to the right of the cell you are working on. To restore the screen, just scroll until the present cell is off the screen, and then come back to it.

show <matrix_name> <view> <line> [a] [b]

Note that the format of the show command with a matrix name is different from the format of show with a vector name. The "view" argument must be one of the following:

r for row
c for column
y for year.

If view is r, then <line> is the number of the row to be displayed and the optional a and b are the first year and first column number to be displayed. (Rows are displayed as if they were a column.) If view is c, then <line> is the number of the column to be displayed and the optional a and b are the

first year and first row number to be displayed. If view is y, then line is the year number, and the optional a and b are the row and column numbers of the upper left element of the initial display. Examples:

show srtm r 5
show srtm c 7
show srtm y 1987
show srtm y 1987 25 25
show srtm r 5 1987 25

format <d>
form

Specify the number of decimal places displayed <d> by subsequent "show" commands. Examples: "format 0", "format 2", or "format g". The last uses the "general" format, which is the default.

punchvec <filename> <vecname> <startyear> <endyear> [<width> <decs>]
pv

Print out a vector to a file for all sectors, for the years specified. The optional width and decs arguments again specify the field width, and number of decimal places. The output file starts with a number of comment lines, telling the name of the vector, starting and ending years, and format information. Then one comment line gives the years as column headings. Each following line of the file contains the time series for one sector, with the name of the series, followed by the time series of data.

**Other commands**

commands

Should you have forgotten any of the preceding commands, the command "commands" will show you a list of them.

listvecs
lv

This command shows a list of the vectors and matrices in the model. An even better way to see them is to do "ed vam.cfg".

vtitle <title>

This command allows you to give the Vam file a title. This title is displayed when using Compare.

**Step 4. Put the DPM variables in the proper files for LIFT**

Using a program editing software, edit the macrofix.dat file located in the lift\adata directory.

Find the section that includes the DPM variables. Import the dpmmac.fix file into this section. Next, edit the fdv.dat file, also in the lift\adata directory. Find the section of DMG variables and replace the old forecast values with the new ones in the file dpmfdv.fix. Next, be sure to run the Macrofix and Fdvred programs to read the new data before actually running LIFT. Now LIFT is ready to go.

## DATA SOURCES

Data from the Bureau of the Census can be obtained through internet at the following site:
http://gopher.census.gov:70/0/Bureau/Population/Projection/Nation

Specifically, data for the armed forces and institutionalized population are from:

Day, Jennifer Cheeseman, *Population Projections of the United States, by Age, Sex, Race, and Hispanic Origin: 1995 to 2050*, U.S. Bureau of the Census, Current Population Reports, P25-1130, U.S. Government Printing Office, Washington, DC, 1996.

All population projection data was downloaded from the Census Bureau Bulletin Board. The following is an excerpt from the documentation that accompanied the data:

POPULATION PROJECTIONS OF THE UNITED STATES

BY AGE, SEX, RACE, AND HISPANIC ORIGIN:

1995 TO 2050

Population Projections of the United States by Age, Sex, Race, and Hispanic Origin: 1995 to 2050 [machine-readable data file] / prepared by the Bureau of the Census, Population Division, 1996.

Users who purchase these diskettes should also purchase the publication, "Population Projections of the United States by Age, Sex, Race, and Hispanic Origin: 1995 to 2050." (Current Population Reports P25-1130) This report includes an analysis of the results of these projections as well as detailed information on the assumptions and methodology used in generating the series. To order this publication, contact the Statistical Information Staff at 301/457-2422. Questions concerning the methodology or analysis of the data should be addressed to Population Projections Branch at 301/457-2428.

SUBJECT-MATTER DESCRIPTION:

Projections of the resident population of the United States by age, sex, race, and Hispanic origin, from 1995 to 2050.

There are seven data files for each series: six include data on the projected population and one file shows the projected components of change. The middle series diskette also includes a file of the Armed Forces overseas population as of July 1, 1994.

For July 1 of each year, 1995 to 2050, the population file presents projections classified by age, sex, race, and Hispanic origin. These projections are based on July 1, 1994 estimates, consistent with the 1990 census, as enumerated and projected forward using the inflation-deflation variant of the cohort-component method. Projections are provided for 102 age categories (yearly total, individual years 0 to 99, 100 and over), four race categories (White; Black; American Indian, Eskimo, and Aleut; and Asian and Pacific Islander), Hispanic origin, and the four races without the Hispanic origin population (non-Hispanic race groups). All data are unrounded.

The component file presents data from 1995 through 2050. This file includes the annual July 1 and January 1 populations in addition to the annual number and rates of birth, death, net immigration, natural increase, and net change.

UNIVERSE DESCRIPTION:

Resident Population of the United States

TECHNICAL DESCRIPTION:

FILE TYPE: ASCII

FILE SIZE:

| File name* | Number of bytes | Records | Columns |
|---|---|---|---|
| P1995_00.A | 135,864 | 612 | 220 |
| P2001_10.A | 226,440 | 1020 | 220 |
| P2011_20.A | 226,440 | 1020 | 220 |
| P2021_30.A | 226,440 | 1020 | 220 |
| P2031_40.A | 226,440 | 1020 | 220 |
| P2041_50.A | 226,440 | 1020 | 220 |
| COMPONEN.A | 58,240 | 560 | 91 |
| AFO94 | 10,584 | 49 | 213 |

Methodology for National Population Projections
(Used for the 1995 to 2050 projections)

Extracted from report P25-1130.

The Cohort-Component Framework

Six sets of data are required to generate these population projections using the cohort-component model. These are a base-year population, projected fertility rates, projected survival rates, future net immigration statistics, 1990 inflation/deflation rates, and an estimated Armed Forces overseas population. The most difficult aspect of producing these population projections involves deriving the base-year starting points and rates. Each data set is organized into 16 different race/ethnic/sex matrices with a cell for each year of age 0 to 100 and over. The sixteen matrices are White; Black; American Indian, Eskimo, and Aleut; Asian and Pacific Islander -- by Hispanic origin (Hispanic and Not Hispanic) and by sex. The sum of all the cells in all 16 matrices equals the total population. Starting with a July 1, 1994 modified population estimate based on the 1990 census, each cell is inflated by Demographic Analysis to correct for persons not included in the population count in 1990. Then each age/race/ethnic/sex cell is survived forward to July 1, 1995 by applying the appropriate survival rate.

The population under 1 is created by first calculating the population of women exposed to the risk of childbearing. Generally this involves averaging the July 1, 1994 and July 1, 1995 inflated female population, of each race/ethnic group by single years of age between the ages 14 through 49. Then, the corresponding age/race/ethnic specific fertility rate is applied to this averaged female population to produce, after aggregation, the total number of births by race/ethnicity for that 12-month interval. The assumed sex ratio for each group is used to divide the births into males and females. Then factors from a 1990 census file showing the race and/or origin reported for children in families with parents of differing race and/or origin were applied to the births. This resulted in the shift of some births from the mother's race/origin to that of the father. Finally, the number of births by sex and race are survived forward to July 1, 1995.

After the births are calculated, net immigration by age/sex/race/ethnicity are added. Then the movement of the population of Armed Forces overseas are applied to the population by detailed group. Next, the population is deflated to be consistent with the 1990 census count. A small pro rata adjustment is made to the deflated age estimates in each sex/race/ethnic group to bring them in exact agreement with an independent estimate of the total population of each group.

Finally, the 16 groups are summed, creating the groups most frequently requested, and are displayed in this report. This includes adding the Hispanic and Not Hispanic groups for each race to make a total population by age/sex for each race, and adding the eight Hispanic origin matrices to get the total age/sex Hispanic-origin population. As these groups are added, new total rates are derived for these new groups. The same set of procedures when applied to the July 1, 1995 population would generate the July 1, 1996 population. This process is continued through the year 2050.

## The Base Population

The beginning population of these projections is the July 1, 1994 estimate. These estimates are consistent with the 1990 census count, but cannot be directly compared to the published results by age and race because modifications were made to the data to adjust for age misreporting and the

reporting of an unspecified race in the 1990 census.

These results have been modified by use of the inflation- deflation variant of the cohort-component method. This method does not correct for the net undercount in the 1990 census. This procedure holds constant the size and age-sex-race-Hispanic origin composition (estimated by Demographic Analysis) for those persons not enumerated by the 1990 census. The inflation- deflation variant yields a population distribution in each projected year which is similar to that which would result if a census with the 1990 pattern of undercount (as estimated by Demographic Analysis) were conducted in that year. It therefore preserves the actual pattern of population change by age group rather than by cohort. If the cohort-component method alone were used, the effects of underenumeration would age as the population grew older, i.e., the unenumerated 1990 population would remain with the cohorts as they aged.

Similar to previous projections, the population in the Armed Forces overseas was held constant in size and age-sex-race- Hispanic origin composition throughout the projection period. In this case, however, the projected population is resident population only, excluding the population of Armed Forces overseas. Therefore, to accurately move the resident population forward in time, members in the Armed Forces overseas need to be subtracted as they leave the resident population going overseas and added when they return. This procedure assumes no deaths or births to the Armed Forces overseas. Without this movement of the Armed Forces population in and out of the resident population, the starting cohorts of ages 18, 19 and 20 in 1994 would always appear relatively smaller than younger cohorts when they reach those ages.

Fertility

Assumptions. Three different future fertility levels are used. These are derived from analysis of natality statistics for five groups of women by race group and Hispanic origin. Assuming that the pattern of fertility would be the same for all Hispanic women, across all races, we derived levels for Hispanic-origin women and the remaining four non-Hispanic race groups: non- Hispanic White; non-Hispanic Black; non-Hispanic American Indian; and non-Hispanic Asian. The levels for each total race group would then be the combination of the Hispanic and non-Hispanic proportions of that race group. As these proportions change, so would the combined fertility levels.

In 1989, the National Center for Health Statistics (NCHS) changed their method of tabulating race of births to reflect the race of the mother. The projected fertility levels and rates in this report use this change. Several fertility assumptions in this report are founded on past trends. First, this projection does not assume race/ethnic fertility convergence. Historical data shows that the major differences between White and Black fertility is timing, that is, Blacks tend to have their children at earlier ages than Whites. After age 25, however, White and Black fertility has been about the same. Yet, there is no compelling evidence of overall convergence of Black-White or any race-ethnic fertility. Second, in the last decade, many women delayed the start of childbearing until their late 20's or 30's. This recent shift to a new age pattern of childbearing is assumed to remain constant. Third, since the end of the Baby Boom, completed cohort fertility has remained about the same.

Therefore, there appears to be no reason to assume a change from current fertility levels.

In the low series, the fertility rates are assumed to fall for all races and Hispanic origin, decreasing 15 percent by the year 2010. The reverse is assumed for the high series.

Creation of 1994 birth rates. The birth rates for these projections are based on 1990 to 1992 fertility rates and raked by race and Hispanic origin to the fiscal number of births forthe period July 1, 1993 to June 30, 1994. The beginning rates were created using NCHS natality data divided by the July 1, 1991, population estimates for five race/ethnic groups: non- Hispanic White, non-Hispanic Black, non-Hispanic American Indian, non-Hispanic Asian and Pacific Islander, and the Hispanic origin.

Life Expectancy

Assumptions. As in the last Census projections, three basic mortality assumptions are used. The middle life expectancy assumptions reflect a slow improvement in life expectancy. The last 10-year trend of mortality improvement, from 1980 to 1990, is replicated, and some additional impact of AIDS is included. The incidence of AIDS is projected to increase linearly until 2005. After 2005, mortality from AIDS is assumed to slowly decrease, returning to the current level of AIDS mortality by 2050. The low life-expectancy series assumes that current mortality rates will continue, with an increase over the next 15 years in deaths due to AIDS. This uses a FY1994 base life table with AIDS projected to increase linearly up to the year 2010, then remain constant. The high life-expectancy assumption, or rapid improvement series, replicates the pattern of mortality between 1970 to 1980, thus ignoring the impact of AIDS.

Construction of the 2050 mortality rates in the middle assumption involved more than simply projecting the continuing trend of the 1980 to 1990 improvement. Although this may appear reasonable, variations in the trends by age and sex produced some questionable results. A few general conditions were imposed on the 2050 rates. These conditions included the following:

1. No 2050 death rate was allowed to be higher than it was in 1994.
2. No male rate was allowed to ever be lower than the equivalent female rate.
3. Within a given race-sex group, the death rates must steadily rise from age 25-29 to 100+.
4. No death rate was permitted to improve more than 3 percent per year during the 1994 to 2050 period.

Life tables. The beginning life table was based on NCHS death data (final 1991, provisional 1992, and sample year ending June 30, 1994) for age, sex, race, and Hispanic origin. For the oldest age groups, where frequently age specific death data are often misreported, the rates were corrected using data from the Social Security Administration. The total death rates were then adjusted to agree with the estimated 2,279,000 deaths in FY 1994. Calendar year life tables adjusted to be consistent with the 1990 census enumeration were created for 1995, 2005, and 2050.

The death rates used for every projected life table were based on the assumed change from the death

rates used in the beginning- year life table. In each assumption, sex and race differentials were not assumed to converge, but instead were determined by the rate of change applied to each individual group. The rates of change were computed based on adjusted level death rates for 1969-71, 1979-81, and 1989-91. Some additional information came from an analysis of Medicare data for the 1968 to 1979 period. In the middle series, life tables were computed for 2005---a turning point of the series, and 2050---the end point. For the alternative series, a 2010 life table was created for the low life-expectancy assumption, holding the rest of the projected period constant at 2010 levels (i.e., AIDS gets no worse or better), and for the high life expectancy series a 2050 life table was made. Survival rates were extrapolated for each year between these points.

Net Immigration

The net immigration component used in these projections is composed of six types of migration, five which increase the population (immigration) and one that decreases the population (emigration). In the low, middle, and high net immigration series, the same age/sex/race/ethnic ratios are used for each type of migration, but raked to the alternative levels for each type.

In the middle assumption, a total annual net immigration of 820,000 is held constant every year between 1994 and 2050. In the low and high assumptions, the annual figures of 300,000 and 1,370,000 are held constant beginning in 2000. This supposes that changes in the number of net immigration may take a few years to take effect.

The assumed future level of legal immigration in each series was not changed from the previous projections. The middle assumption is close to the average of the data for the July 1991 to June 1994 period. The age, sex, race, and Hispanic-origin distributions were based upon these same recent data which were derived from information provided by the Immigration and Naturalization Service. The low alternative represents the legal immigration experience of the 1980's; whereas the high assumption reflects the possibility of piercing the legal cap on this type of immigration through modifications to the current law.

The assumed future levels of refugee immigration in each series also were not changed from the previous projections. The middle assumption is based on current levels and interpretation of current laws. The low alternative represents the experience of the 1980's while the high assumption again reflects the possibility of piercing the legal cap on this type of immigration through modifications to the current law. The age, sex, race, and Hispanic-origin distributions for each alternative were based on the July 1993 to June 1994 data derived from Office of Refugee Resettlement statistics.

Undocumented net immigration is difficult to measure. Currently, the U.S. Census Bureau's best estimate adds about 225,000 net undocumented immigration to the United States each year. The wide range between the low and high alternatives reflects some of the uncertainty of the actual number. The age, sex, race, and Hispanic-origin distributions for each alternative were based on

data derived from the 1990 census.

Future Puerto Rican immigration is assumed to be 5000 per year in the middle series, equally bounded by the low and high alternatives. The age, sex, race, and Hispanic-origin distributions for each alternative were based on data derived from the 1990 Censuses of Puerto Rico and the United States.

Civilian citizen immigration is based on the stock of Armed Forces overseas and their returning children and spouses. Since the size of the military population is assumed to be constant in all the projection series, this flow is held constant across series. The age, sex, race, and Hispanic-origin distribution was based on available data for the Armed Forces overseas as of July 1, 1994.

Emigration, similar to the undocumented migration, contains some unknown qualities. Similar to previous Census Bureau projections, emigration is considered a constant flow, instead of a proportion of the total of all in-migration. Similar to the previous set of projections, the low series reflects a greater population exiting. Logically, if conditions exist for low in-migration (for example, an economic downturn), the same conditions or reasons would also drive more people out of the country. The reverse is assumed for the high series.

RACE AND ETHNIC DEFINITIONS AND CONCEPTS

The racial classification used by the Census Bureau generally adheres to the guidelines in Federal Statistical Directive No. 15, issued by the Office of Management and Budget, which provides standards on race and Hispanic origin categories for statistical reporting to be used by all Federal agencies. The race and Hispanic origin categories are defined as the following:

American Indian, Eskimo, and Aleut. A person having origins in any of the original peoples of North America, who maintains cultural identification through tribal affiliation or community recognition.

Asian and Pacific Islander. A person having origins in any of the original peoples if the Far East, Southeast Asian, the Indian subcontinent, or the Pacific Islands. This area includes, for example, China, India, Japan, Korea, the Philippine Islands, and Samoa.

Black. A person having origins in any of the black racial groups of Africa.

Hispanic. A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin,regardless of race.

White. A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.

## Appendix A: VAM

### Commands for introducing data

Vam has a number of ways of reading ASCII data. To begin with, there are seven commands similar to G commands for introducing data from text files, so as to provide various combinations of:

- data format (row-wise as for a G data command or column-wise as for a G matdata command)
- destination (the G-like workspace file of Vam or the vam file)
- action (initial introduction or update)

The commands are

| | |
|---|---|
| data | vdata |
| update | vupdate |
| matdata | vmatdata |
| matupdate | |

All of the commands in the left column work exactly like the corresponding command in G to introduce data into the G workspace that is used by Vam. The "vdata" and "vupdate" work exactly like "data" and "update" except that the series read goes into the vam file instead of into the G workspace.

The vmatdata command is unique to Vam, and needs a word of explanation. It can be used to bring in data in a variety of formats commonly used by statistical agencies in releasing input-output tables. It can read a file which shows any one of the following:

- one vector in columns for several years
- one vector in rows for several years
- several vectors in columns for one year
- several vectors in rows for one year

The form of the command is

```
vmatdat <r|c> <nv> <nyrs> <first> <last> [skip]
    <year>     <vec1> <vec2> ... <vecnv>
OR
    <vec>      <year1> <year2> ... <yearnyrs>
[form]
```

where

- r|c is r or c according to whether the vectors are rows or columns.
- nv is the number of vectors.

nyrs is the number of years.
first is the position in the vector in .vam of the first item in the data which follows.
last is the position in the vector in .vam of the last item in the data which follows.
skip is the number of spaces on each line which should be skipped before beginning to read data. *If no value for skip is provided, then a form line (explained below) will be expected as the third line. If no spaces are to be skipped and no form line provided, then give skip a value of 0.*

year is the year of the vectors or the first year if nyrs > 1.
vec1 is the name of the first vector, vec2 is the name of the second vector, etc.
form If no value for skip is given, then this form line must be provided. It should have the letter r in every position which is to be read in the following table. This device allows direct reading of tables that have columns of data separated by | or other symbols or which have subtotals which are not to be read. A typical form line and a matching data line are:

```
form:                rrrrrrrrrrrrrrrrrrr         rrrrr      rrrr
data:Agriculture    | 3450  3718  249   0  | 6780|  8102 |**| 1598
```

The first version of the second line is used if there are several vectors for one year; the second is used when giving data for one vector in several years. In both cases, the item of which there is only one is specified first and then the items of which there are several.

Any line beginning with a # is skipped completely.

For example, the lines

```
vmatdat c 6 1 1 18 3
1986   minst      pgm      pgf      finst      mmil      fmil
1      21         0        2        3          6         7
...    ...        ...      ...      ...        ...       ...
18     38         401      0        0          17        15
```

would read the 6 columns of data into positions 1 through 18 of the named vectors (minst,pgm, ...) for the year 1986. Positions 1 - 3 of each line, which contain the sector number for easy visual reference, are skipped.

Here is another example which reads the vectors as rows, as may often be the case when reading printed tables of primary inputs.

```
vmatdat  r 6 1 1 6  16
1986 labinc propinc deprec inter profit tax
#                              1            2            3            4
5        6
labor income          7303     21     368    1203    1302     820
proprietor inc  1215    9      100     107     303      92
depreciation          950      3      82     104     121      73
interest income       845      7      83      54      95      52
profits               50       3      25     217     337      38
indirect taxes  121   1        32     178     294      29
```

Note the use of the line beginning with a # to provide labels for the columns. The flexibility of vmatdat often makes it possible to read in final demand columns or primary inputs as they appear in printed tables or on the diskettes provided by statistical offices.

The second form of the second line is illustrated by the following example, which reads one vector, cons, for five years.

```
vmatdat c 1 5  1 57 2
cons 1985 1986 1987 1988 1989 1990
 1   31.8 37.2 32.5 38.7 41.2 43.1
....
57     1.2  1.7  1.8        2.0  2.1  2.2
```

It must be emphasized that, precisely because it can read in free format, the vmatdat command cannot interpret blanks as zero entries. There must be a numerical value for all cells in the tables, especially the 0's.

For large data sets, it is convenient to be able to read "folded" vectors. This is the function

of the fvread command. The usage is:

fvread <vector_name> <year> <skip>

Example:
fvread fd 1987 10
finaldem 1  23 36 83 92 32
finaldem 6  8 23 73 80 75

The <skip> is the number of columns to skip at the beginning of each line. There must be present as many elements as are in the vector, and they must be in order.

There are three commands to introduce a matrix from an ASCII file. The one comparable to vmatdat is "matin". It has the form

matin <matrix_name> <year> <firstrow> <lastrow> <firstcol> <lastcol> [skip] [form]

There should then follow a rectangular array of numbers matching the "firstrow", "lastrow", "firstcol", and "lastcol" parameters of the command. As in "vmatdata", the optional "skip" parameter is the number of spaces to be skipped in reading each line. The skip parameter and the form line work together exactly as described above for vmatdat: the absence of a skip indicates the presence of a form line. A # in the first space of a line means to skip the whole line. Use of "matin" does not affect entries outside the specified area, so it can be used to update a matrix as well as to introduce it originally.

The second command for reading matrices is solely for conversion of models previously built with the Slimforp program. Its form is

matin5 <matrix_name> <year>

and is followed by a matrix in the "punch5" format used in Slimforp. The end of the matrix is signaled either by a ';' in the first space on a line or by the end of the file. For those not familiar with the "punch5" format, it is a fixed length format, with 5 matrix entries to a line, each entry of the form <row> <col> <number>. The format in Fortran would be (5X,5(2I3,F9.0)). Here is a sample from the beginning of an A-matrix:

```
matin5  am 1977
# A-matrix for 1977. 83 rows and 83 columns.
AM     1  1 0.245790  1  4 0.000220  1  8 0.000410  1  9 0.281300  1 10 0.058360
AM     1 11 0.002070  1 12 0.005680  1 13 0.000380  1 15 0.001700  1 16 0.003060
AM     1 22 0.089770  1 24 0.000070  1 48 0.001210  1 49 0.000130  1 50 0.000030
```

This completes the different ways of introducing data from ASCII files.

**Commands for sweeping modification of data**

The remaining commands deal with extending or modifying data which has already been introduced. In these commands, it is convenient to be able to specify a group of sectors. For example, we may wish to refer to the machinery sectors or the food sectors. There are two ways to specify the sectoral composition of these groups. One is through the Fixer program which is described below and is also used for establishing groups for use with the simulation program. We will refer to these groups as "static" groups since they cannot be changed in the course of the program which uses them. There is also one "dynamic" group whose composition we can change as we go along. It is defined by the command:

group <sector numbers>

To make the dynamic group consist of sectors 19, 21, 23, 24, and 25, we can write:

group 19 21 23 24 25

or, equivalently:

group 19-25 (20 22)

"19-25" is a consecutive inclusion and means to include all the sectors from 19 to 25, inclusive, while numbers in parenthesis are excluded if they have been previously included in the group. (Think of the accountants' practice of putting negative numbers in parenthesis.) A user-defined group from the groups.bin file (described below under the Groups program) may be part of the definition. Thus, if group "mfg" has been defined by the Fixer program, ":mfg" will reference those sectors. The group definition can have any number of individual sectors, consecutive inclusions, or exclusions. Consecutive inclusions may appear within parenthesis, so that they become consecutive exclusions. Example:

group 1-78 (27-35 40 :mfg) 30

The inclusion or exclusion of sectors is done dynamically from left to right. In the above example, the group starts off being all sectors from 1 to 78, then excludes 27 to 35, 40 and all those in the "mfg" group from the groups.bin file, and then adds in sector 30.

The dynamic group definition remains in force until changed. It may be redefined as frequently as necessary.

There is here a slight circularity; Vam uses the static groups defined in the groups.bin file made by Fixer, but Fixer must have a .vam for some of its operations. Vam, however, will function without the groups.bin file. It then begins with the report "0 groups defined." One can then do everything except use static groups. Thus, one can run Vam to build a basic .vam file, then run Fixer, producing

groups.bin, and Vam run again for commands using the static groups.

glist <name>

List the sectors in a static group. The <name> specifies the group name.

listgroups

Lists all the groups currently active in the GROUPS.BIN file.

diagextract <matrix_name> <vector_name>

Extract diagonal elements from the matrix and put them into the named vector. This seemingly strange process is useful because across-the-row coefficient changes generally should not be applied to the diagonal elements of Leontief A matrices. They can be removed by this command, the across-the-row change done, and the diagonals then restored by the following command. The command works for both regular and packed matrices.

diaginsert <matrix_name> <vector_name>

Insert elements in the named vector into the diagonal of the matrix. The diagonal element will remain zero for a packed matrix which has zero coefficient in that position for all years.

lint <names>

Replace missing values in the named series by linearly interpolated values. Zeroes before the first or after the last observation are not replaced. This command applies only to series in vam.bin. Groups may be used in the names. Ex.:

lint pce1

load def
lint :2

The second example does the interpolation on the sectors in group 2 of the def vector.

Entire matrices may be interpolated with one command. For example
lint am

will interpolate the entire am matrix. This interpolation also works for packed matrices, though of course it is only the elements non-zero in at least one year which are interpolated.

fdates <fdate1> [fdate2]

Specify dates which will be used as the range of action of the "index" and "ctrl" commands. The default values of the fdates are the beginning and ending dates of the vam file, respectively.

index <base date> <guide> <name1> [name2] ...

Move the named elements of the currently loaded vector by the index of the <guide> series over the range specified by the current fdates. (The vector will typically have been loaded by the "load" or "show" commands. Rows and columns of matrices count as vectors here.) The value of the named series must be known in "base date" period. Normally the guide series is a single series that must exist in the workspace or assigned banks. (Sometimes, however, the guide can be a vector, as explained below.) The value of the named series in the base period will not be changed; other values in the series, however, possibly both before and after the base date observation, will follow the given index. The use of groups is allowed in the names of series to be affected. For example, if pce is the loaded vector, the names "1 7 9" will move sectors 1 7 and 9 of pce by the given index. The name "0" refers to the whole of the currently "loaded" vector; the name ":" means use the current dynamic group in the currently loaded vector, and names like ":1" or ":mfg", mean to use the static groups 1 or "mfg" of the currently loaded vector.

A specific name like "exp2" means "store the current loaded vector, load the exp vector, and move the series exp2 by the guide index."

A vector or matrix name not followed by a sector number means to move the whole vector or matrix by the index. A matrix name followed without space by a sector number means to move that row of the matrix by the guide. Thus, "am2" means move the second row of the am matrix by the guide.

Note that the values in the base-date period of all items moved by an index must have been previously specified.

If <guide> is the name of a vector and name1 is a matrix, then the rows of the matrix are indexed by the corresponding elements of the vector. If an element of the vector is zero in the base year, the corresponding rows of the matrix are unchanged. In this usage, there are no names following name1. This feature works for both standard and packed matrices.

ctrl [basedate] <guide> <name>

Impose the values of the <guide> series as a control total on the named sectors of the currently loaded vector. The control is imposed over the period specified by the current fdates. If no basedate is present, the absolute values of the guide series are the control total. If the basedate is present, the guide series will be scaled to the total of the <names> series in that period before being used as a control total. In naming the sectors, the allowable names are illustrated by:

| | |
|---|---|
| 0 | for all the sectors in the loaded vector. |
| : | for the sectors in the present dynamic group |
| :1 :mfg | for the sectors in static groups 1 and mfg |
| 1 7 9 | for the group composed of sectors 1, 7, 9 |

ras <matrix> <row> <col> [year]

Balance the named coefficient matrix by the rAs method. It is assumed that the columns of the matrix should add to 1.0 and that it is desired that <row> = <matrix>*<col>. That is <row> is vector of row controls and <col> is the vector of column controls. Strictly speaking, <row> should be a column vector and <col> should be a row vector. To Vam, however, it does not matter whether <row> and <col> are columns or rows. If the sum of the elements of <col> do not equal the sum of the elements of <row>, the user is given the opportunity to choose which should govern. The other vector will be scaled to have the right total. The rows will then be scaled to get the correct row totals, then the columns scaled to get the correct column totals. Every five iterations, Vam reports the row and column in which the maximum scaling occurs. When the maximum scale factor differs from 1.0 by less than .00002, convergence is declared to have been reached. The last scaling will be of the rows. If convergence does not occur after 30 iterations, the user can select to abandon the balancing, or to use the result, or to try 30 more iterations.

When balance is achieved, each column is divided by its column total. This division was done because most matrices are distribution matrices with columns summing to 1.0. For others, the normalized matrix obtained from the ras command can be converted to flows by the "flow" command described below, and the flows can then be converted to coefficients by dividing by any desired vector with the "coef" command.

If year is missing in ras, coef, and flow commands, the default value is the years specified by a previous fdates command.

coef <matrix> <vec> [year]

A flow matrix in the named matrix in the specified year is converted to a coefficient matrix by dividing each column of the matrix by the corresponding element of the named vector.

flow <matrix> <vec> [year]

This command is the opposite of coef command. It multiplies each column of the matrix by the corresponding element of the named vector. If no value of "year" is specified, the process is performed for each year in the range previously specified by the "fdates" command.

getsum <matrix> <r|c> <vector>

Gets the sum of the rows or columns of a given matrix or vector, for all the years in the Vam file. The first argument must be the matrix or vector for which we want to obtain the sums; the second argument must be 'r' or 'c' (to obtain the sum by rows or by columns); the third argument is the vector where the result is to be stored.

The sum is calculated and stored for all the years, so that we cannot have mixed information in the

target vector.

**Vam2vam -- Selective copying from one vam file to another**

Vam2vam is a utility program, run at the DOS prompt, to copy selected matrices and vectors from one vam file to another. It often happens that one discovers that one has left out of a vam file some essential matrix or vector. It is easy enough to modify the vam.cfg file and create a new, all-zero vam file with a place for the new matrix or vector. But how can data be copied from the old vam file to the new? Vam2vam is the answer. Or one discovers that a matrix or vector has to be enlarged. How can the data in the old vam file be copied into the new? Vam2vam is the answer. Or a vam file has been used in the preparation of data which has in it various matrices and vectors which were essential for preparing the data but which are not needed in the final model. How can we extract just the final product matrices and vectors into a new vam file for the model? Use Vam2vam.

The program is invoked by the command:

vam2vam <source> <target> <list> <startdate> <stopdate>

where

<source> is the filename, without the .vam extension, of the source vam file;

<target> is the filename, without the .vam extension, of the target vam file;

<list> is the name of a file with a list of the names of vectors or matrices to be copied. For example, to copy abc in the source to def in the target and xyz in the source to xyz in the target, the contents of the <list> file would be

abc def
xyz

Note that the default value of the name in the target is the name in the source, so we did not have to repeat xyz on the last line.

<startdate> is the first year whose data is to be copied

<stopdate> is the last year whose data is to be copied.

## Appendix B: Fixer

FIXES AND THE FIXER PROGRAMS

Fixes, as used here, are ways to make a model work the way we want it to, not necessarily the way that emerges from its equations. The power that fixes give over a model can certainly be, and often has been, abused. Nonetheless, they have a legitimate role. Suppose, for example, we wish to consider the impacts of some event which the equations never dreamed of, like a natural disaster or a massive overhaul of the health care system. Then a fix is the natural way to convey to the model that the equations are not to be entirely trusted.

Interdyme has three types of fixes, those for macro variables, those for vectors, and a special type for industry outputs.

### Macro Variable Fixes

The macro variable fixes work very like those of models built with the G-Build combination, but also have much in common with the vector fixes described in the next section. The program that handles the macro variable fixes is called Macfixer. The input to Macfixer is a file prepared by the user in a text editor, whose format we will explain in a moment. We recommend using the extension .mfx for these files, although this is not necessary. Once this file has been created, the program Macfixer is run, and the results of this program are written to a "macro fix bank", which is essentially a G bank, which can be read with the G regression program. The rootname of the macro fix G bank is passed to the simulation program at run time via the dyme.cfg file.

Macfixer also requires a configuration file, called macfixer.cfg, which specifies the root name for the fix index and bank files, the name of the text input file, the root name of the G bank file used for base values for the index and growth-rate fixes (this would normally be the G bank created for use with the simulation program), and the name of the output check file. This last file shows the values of each fix in each year, and serves as a check on the results in the binary files. The default values in the fixer.cfg file are as follows:

```
Name of input fix file;        macrofix.mfx
Name of fixes bank;            macfixes
Name of GBank file;            dyme
Name of check file;                  macfixer.chk
```

When invoked, Macfixer displays these options and allows you to change them for that run. They are not thereby changed in macfixer.cfg; should you want to change them there, just use a text editor. While it is up to the user to name files, it makes good sense to give files for the same simulation the same "root" name. A simulation that involves low defense expenditures, for example, could have a G bank file called lowdef.bnk, and a .mfx file called lowdef.mfx. If you know that you

will want to accept the defaults in the macfixer.cfg file, call macfixer with a -z option, thus "macfixer -z". This invocation will skip the opening display of options. This option is especially useful if you are running the programs in a batch file.

There are several varieties of macrofixes that may be given, and they are described in the list below:

skip

is the simplest type of fix. It simply skips the equation and uses the values in the model G bank. For example:

```
skip invn$35
```

would skip the equation for the macro variable invn$35, and use the value already in the model G bank.

ovr

overrides the result of the equation with the value of the time series given. Values between given years are linearly interpolated. In the example below, the macro fix program would calculate and override a fix series that starts in 1992, ends in 2000, and moves in a straight line between the two points. For example,

```
ovr uincome$
  92  154.1
  2000  182.3;
```

would override the value of the forecast of uincome$ with the values shown for the years shown. Note that year can be either 2-digits or 4-digits (they are all converted to 4-digits in the program).

mul

multiplies the equation's forecast by a factor specified by the data series on the following line. For example,

```
mul ulfi$
  1992  1.0
  1995  1.05
  2000  1.10;
```

multiplies the forecast results for the macrovariable ulfi$ by the factors shown. Values of the multiplicative fix between the years shown are linearly interpolated.

cta

does a constant term adjustment. That is, it adds or subtracts the value of the time series to the result of the equation. The time series is provided by the fix definition. For example,

```
cta nonagincome
  1992 .0001
  1995  200
  2000  180;
```

is a constant term adjustment for nonagricultural income from 1992 to 2000. Intermediate values are of course linearly interpolated.

ind

is a variety of the override fix that specifies the time series as an index. There must be data in the .vam file for the item being fixed up until at least the first year of the index series specified. The value for the item in that year is then moved by the index of the time series given by the fix lines. For example,

```
ind wag01
 1982  1.0  1.03  1.08  1.12  1.15
 1997  1.21  1.29  1.31  1.34;
```

will move the value of wag01 in 1982 forward by the rate of change of the series given, and will replace the calculated value of wag01 by this value when the model is run.

gro

is a type of override fix that specifies the time series by growth rates. For the growth rate fix to be legal, there must be data in the .vam file up until at least the year before the first year of the growth rate fix. Missing values of the growth rates are linearly interpolated.

```
gro wag01
 1983  3.1
 2000  3.4;
```

stp

is a step-growth fix. It is like gro except that a growth rate continues until a new one is provided. A value for the final period is necessary.

```
stp wag01
 83  4.1
 95  4.5
 2000 5.0;
```

rho

is a rho-adjustment fix. This type of fix finds the error made by an equation in the last year for which there is data; in the next year, it multiplies this error by the given rho and adds to the value forecast by the equation; the next year it multiplies what it added in the first year by rho again and adds the result to the equation's forecast, and so on. For rho-adjustment fixes, the format is:

```
rho <depvar> <rho_value> <rho_set_date>
```

where

```
rho_value is the value of rho.
rho_set_date is the year in which the rho-adjustment error is
    to be calculated.  If none is provided, it is set in the
    first year of the run.
```

for example:

rho invn$38 .40 1991

tells the model to apply a rho-adjustment to the variable invn$38 using the value .40 for rho, and starting the rho-adjustment in 1991.

A rho fix with a rho_set_date works like a "skip" in years before the rho_set_date. A variable can have a "rho" fix in conjunction with and a "cta","mul","ind" or other type fix. The rho adjustment is applied before the other fix.

When the input file as described above is ready and the macfixer.cfg file calls for its use, type "macfixer" at the DOS prompt to invoke the program Macfixer. When the model is running, calls to the "modify" function will apply the fixes, using the information in the macro fix G bank specified in the dyme.cfg for that run. Note that to view the fixes in the macro fix databank, specify the series name as the name of the macro variable, followed by a colon (':'), followed by a one-letter code signifying the type of fix. These codes are as follows:
skip ('k'), ovr ('o'), cta ('c'), ind ('i'), gro ('g'), stp ('s'), and rho ('r'). Therefore, to view a cta fix on the variable invn$38, do the following command in g:

ty invn$38:c

Macrofixes provide an alternative way to supply values of exogenous variables. Exogenous variables, to review, should be put into the "hist" bank in the process of running Idbuild. If the variable appears in no .sav file for a macro equation, then it is included in the "pseudo.sav" file. The "standard" way of providing the values of the exogenous variables is then through "update" or other commands in Vam. Another possibility for providing exogenous values is to have a special run of G with the "hist" or other bank as the workspace bank. Finally, one can provide the exogenous values as macrofixes. For example, if we want disinc to be an exogenous variable, then -- however we are going to provide the values -- we need the statement

f disinc = disinc

in the "pseudo.sav" file. To use the macrofix method of assigning values, we need in the code of the model the statements

```
depend=disinc[t];
disinc[t] = disinc.modify(depend);
```

We could then provide the values with "ovr", "ind", "gro", or "stp" commands to the macrofix program, for example, by

```
gro disinc
    1995 3.0
    2000 3.5
    2005 4.0;
```

This method has the advantage of keeping all the fixes which constitute a scenario in one place. It also allows the use of the "gro" and "stp" fixes, which may be convenient. It has the disadvantage of adding an additional series to the banks which constitute the model and an additional statement within the model.

**Output Fixes**

The output fixes allow the values of output specified in the vam file to override values computed by the input-output equations. There is then the question of what to do with the difference. Interdyme offers two possibilities: add any excess demand to imports or simply ignore the difference. The options are specified in column 18 of the "sectors.ttl" file, which is where the names of the input-output sectors are. The options for this column are:

- e use the equation
- i add the difference to imports
- d put the difference in a vam vector named "dump", where nothing is done with it, but it can be displayed.

**Vector Fixes**

The vector fixes are more complicated because they can apply to individual elements of a vector, to the sum of a group of elements, or to the sum of all elements in the vector. However, the format of the vector fixes is very similar to that of the macro variable fixes, described above. The preparation of the vector fixes is the work of the Fixer program.

Vam, it should be noted, prepares vectors of exogenous variables; fixes apply to vectors of endogenous variables.

The input to Fixer is a file prepared by the user in a text editor, whose format we will explain in a moment. We recommend using the extension .vfx for these files, though the program does not rely on this convention. Fixer also reads the definitions of static groups of sectors and writes them into the "groups.bin" file which can be used both by the simulation program and by Vam. To use the Fixer program, *it is essential that the model's vam.cfg file should have a vector called "fix"* with enough rows to allow one for each fix. As Fixer reads the fixes from the input file, it stores the numerical values of the fixes into this "fix" vector in the .vam file. It also creates a "fix index" file, which will have the extension .fin and tells the simulation what to do with each fix. Finally, it produces a binary file with the definitions of groups, called groups.bin. Fixer requires a configuration file, called fixer.cfg, which specifies the "root" filename for the fix index file, the name of the text input file, the root name of the .vam file used for base values for the index and growth-rate fixes, and the name of the output check file. This last file shows the sectors in each group and the values of each fix in each year. It is used only for manual checking of the program. The default values in the fixer.cfg file are as follows:

```
Name of text input file;   vectors.vfx
Name of fix index file;          vecfixes
Name of .vam reference; dyme
Name of text check file;  fixer.chk
```

When invoked, Fixer displays these options and allows you to change them for that run. They are not thereby changed in fixer.cfg; should you want to change them there, just use a text editor. While it is up to the user to name files, it makes good sense to give files for the same simulation the same "root" name. A simulation that involves low defense expenditures, for example, could have a .vam file called lowdef.vam, and a .vfx file called lowdef.vfx. As with the Macfixer program, if you know that you will want to accept the defaults in the fixer.cfg file, call fixer with a -z option

Fixes may apply to a single element of a vector or to a group of elements. The concept of a "group", already touched upon under Vam, is central to the working of Fixer. Basically, a group is simply a set of integers, usually representing sectors in the model. Defining groups is useful because we often want to impose a fix on a group of elements in a vector. For example, we may want to control the total exports of the chemical manufacturing sectors. We might then create a group named "chem", which would contain the sector numbers of all the sectors in question. The command for defining a group is "grp <groupname>", where the groupname can be a number or a name. The sectors defining the group are then entered on the next line. For example,

```
grp 1
    7 10 12
```

creates a group called 1 of the sectors 7, 10, and 12. The "-" sign means consecutive inclusion. Thus

```
group zwanzig
    1 - 20
```

consists of the first twenty integers. Parentheses mean exclusion. Thus

```
group duo
    :zwanzig (2 - 19)
```

makes the group "duo" consist of the integers 1 and 20.

When a group is referenced after it is defined, its name must be preceded by a colon, as shown when "zwanzig" was used in the definition of "duo" above. Names of groups are case sensitive; commands for Fixer must be lower case. Groups do not have to be kept in numerical order and can be defined anywhere in the input file before the first time you used them. If you try to redefine an existing group, the program will complain, unless the new group has less than or equal to the number of elements in the old group. References to other groups can be used in new group definitions only if the groups referenced have already been defined.

Interdyme provides a number of ways for a fix to work. In all of them, a time series is specified by the fix. The forms of the fix differ in how they obtain and in how they apply this time series. The basic format of the input file is

```
<command> <vectorname> <GroupOrSector>
```

followed on the next line by the year and value of the fix. Definitions of the 6 legal commands and examples follow.

ovr

overrides the result of the equations with the value of the time series given. Again, intermediate values are linearly interpolated. In the example below, the fix program would calculate and override fix series that starts in 1992, ends in 2000, and moves in a straight line between the two points. For example,

```
ovr ex 10
  92  154.1
  2000  182.3;
```

would override the value of the forecast of element 10 of the "ex" vector (probably exports) with the values shown for the years shown. Note that year can be either 2-digits or 4-digits (they are all converted to 4-digits in the program).

mul

multiplies the equation forecast by a factor specified by the data series on the following line. For example,

```
mul im 44
  1992  1.0
  1995  1.05
  2000  1.10;
```

multiplies the forecast results for imports of sector 44 by the factors shown. Values of the multiplicative fix on imports between the years shown are linearly interpolated.

cta

does a constant term adjustment. That is, it adds or subtracts the value of the time series to the result of the equations. The time series is provided by the fix definition. For example,

```
cta def :Alice
  1992 .0001
  1995  200
  2000  180;
```

is a constant term adjustment for defense expenditures of all sectors in the Alice group. Intermediate values are linearly interpolated.

ind

is a variety of the override fix that specifies the time series as an index. There must be data in the .vam file for the item being fixed up until at least the first year of the index series specified. The value for the item in that year is then moved by the index of the time series given by the fix lines. For example,

```
ind pceio :zwanzig
  1982  1.0  1.03  1.08  1.12  1.15
  1997  1.21  1.29  1.31  1.34;
```

will calculate the sum of the elements of the pceio vector included in the group "zwanzig" in 1982, will move that sum forward by the index of the series given, and will impose that control total on the those elements when the model is run.

gro

is a type of override fix that specifies the time series by growth rates. For the growth rate fix to be legal, there must be data in the .vam file up until at least the year before the first year of the growth rate fix. Missing values of the growth rates are linearly interpolated.

```
gro out 10
  1983  3.1
  2000  3.4;
```

stp

is a step-growth fix. It is like gro except that a growth rate continues until a new one is provided. A value for the final period is necessary.

```
stp out 1
  83  4.1
  95  4.5
  2000 5.0;
```

When the input file as described above is ready and the fixer.cfg file calls for its use, type "fixer" at the DOS prompt to invoke the program Fixer.