G7 Extensions: Matrix Operations and Quadratic Programming

Ronald Horst [1]

A variety of new features have been added to G7 in the past year. First, several tools have been added to allow algebraic manipulation of matrices. These include addition, multiplication, transposition, inversion, and assignment. Second, a handful of commands have been added to allow the user to interact with the operating system and to perform other tasks. Finally, "hard" linear equality constraints and inequality constraints now are available for single- and multiple-equation linear regressions models.

1. Introduction

G7 continues to undergo revisions and extensions. In the past year, a handful of new commands have been added to G7's significant arsenal. Also, continued efforts to fix bugs have left G7 more reliable. This paper documents most of the new features that have been added since the 2003 Inforum World Conference.

To clarify the material presented here a demonstration G7 script has been prepared. It should accompany the new version of G7 that you receive.

2. Matrix Operations

G7 now offers improved ability to manipulate matrices and vectors. Basic matrix operations now are possible, including addition, multiplication, transposition, inversion, and assignment. While it is not yet possible to perform these tasks in algebraic equations[2], the simple commands described here greatly extend the model builder's ability to construct and manipulate data.

---

[1] Paper presented at the Inforum World Conference in Ascea, Italy, in September 2004. Ronald Horst, Inforum, 1102B Morrill Hall, University of Maryland, College Park, MD 20742, USA, Tel: (301) 405-4608; E-mail: horst@econ.bsos.umd.edu.

[2] Recall that the `vc` command offers limited ability to evaluate equations with matrices on the right-hand-side. The `vf` command allows matrix elements on either side of the equation, and the `f` command allows matrix elements on the right-hand side.

mcopy [bank letter.]<destination> [=] [bank letter.]<source> [period]

The `mcopy` command copies the matrix (or vector) source to the matrix (or vector) destination. If no bank letters are specified, then both the source and destination objects are assumed to be in the default VAM bank. If a value is given for period, then only the values for that period are copied. Otherwise, the starting period and ending period are determined by the `fdates` command. An "=" may be included for clarity, but it is not required.

Both the source and destination objects must exist. Matrices and vectors are not created by `mcopy` or by any of the other commands described here. If the dimensions of the source and destination do not match, then the copying is performed on the smaller number of rows and the smaller number of columns.

For example, for matrices AM and FM that exist in the default VAM file, the following command copies FM to AM for the year 2000.

```
mcopy  AM = FM 2000
```

The following commands copy values for all period between 1990 and 2000.

```
fdates 1990 2000
mcopy AM FM
```

If instead the FM matrix is in the VAM bank assigned as bank `c` and the AM matrix is in the VAM bank assigned as bank `d`, the following command copies FM to AM for the year 1995.

```
mcopy d.AM  c.FM 1995
```

The following command copies vector fm from bank `a` to am in bank `b`:

```
mcopy b.am = a.fm
```

Note that the G7 `vc` command will copy vectors within the same bank, and the vc command also is capable of evaluating certain algebraic equations involving matrices.

madd [bank letter.]<A> = [bank letter.]< B> + [bank letter.]<C> [period]

madd [bank letter.]<A> = [bank letter.]<B> - [bank letter.]<C> [period]

The `madd` command performs matrix addition or subtraction of two matrices and stores the result in a third. If the optional period specification is absent, the operation is done over the `fdates` range. For example,

```
madd A = B + C
```

copies the sum of matrix A and matrix B to C, where all are found in the default VAM bank.

```
madd d.A = e.B – e.C
```

copies the difference of B and C to A, where A is in bank `d` and B and C are in bank `e`.

mmult [bank letter.]<A> = [bank letter.]<B> * [bank letter.]<C> [period]

mmult [bank letter.]<A> = [bank letter.]<B> ' [bank letter.]<C> [period]

mmult [bank letter.]<A> = [bank letter.]<B> / [bank letter.]<C> [period]

mmult [bank letter.]<A> = [bank letter.]<B> & [bank letter.]<C> [period]

Four matrix operations are available with the `mmult` command. When the asterisk (*) operator is employed, and if the matrices are conformable, then

```
mmult A = B * C 1990
```

multiplies matrix B by matrix C and stores the result in matrix A. Because a date is specified in this example, the operation is performed only for year 1990. If instead the apostrophe (') operator is employed, then the transpose of the first matrix is multiplied by the second matrix. For example,

```
mmult d.A = e.B ' e.C
```

multiplies the transpose of B by C and stores the result in A. No date is specified, so the operation is performed for all periods in the `fdates` range. Element-by-element division is performed with the / operator. For example,

```
mmult A = B / C
```

divides each element of matrix B by the corresponding element in C. Similarly,

```
mmult A = B & C
```

multiplies each element of B by the corresponding element of C. As before, specification of a single period limits the range of the operation.

mtrans [bank letter.]<A> [bank letter.]<B> [period]

Matrix transposition is performed with the `mtrans` command. The transpose of the second matrix (B) is stored in the first matrix (A). If period is not specified, then data is copied for all period in the fdate range. For example,

```
mtrans d.A e.B
```

sets A to the transpose of B.

minv [bank letter.]<A> [period]

Matrix inversion is performed with the `minv` command. For example,

```
minv d.A
```

inverts matrix A in bank d for all periods within the `fdates`.

linv [bank letter.]<A> [period]

The command `linv` computes the Leontief inverse of the specified matrix. The Leontief inverse is found by subtracting the given matrix from the identity matrix and inverting the result: $\left(I - A\right)^{-1}$. For example,

```
linv A 1995
```

replaces A with its Leontief inverse in 1995.

3. Miscellaneous Operations

A handful of commands have been added to allow the user to interact with the operating system and to perform other tasks. Some of these commands are new, and the others are improved versions of earlier commands.

dos

dos [options] <arguments>

dos [options] [batch file arguments] <{> ...

The `dos` command passes commands to the operating system. Three versions are offered, along with an important option.

To open a DOS window, simply enter `dos` with no arguments. G7 pauses until the window is closed. The window may be closed by clicking the button in the upper right-hand corner or by typing "exit".

To pass single commands to the system, type the desired system commands after `dos`. For example, to save the workspace bank as "newws", enter the following:

```
dos copy ws.* newws.*
```

To pass a group of commands to the operating system, use the `dos{}` command. The brackets tell G7 to create a temporary batch file containing the system commands between the brackets. The temporary file is destroyed automatically upon completion. Arguments may be passed to the batch file by listing them after `dos` and before the open bracket. Here is an example that again copies the workspace.

```
dos: ws newws{
   rem Copying bank %1 to %2.
   copy %1.ind %2.ind
   copy %1.bnk %2.bnk
   }
```

When the `dos` command is given with arguments (i.e. as in the second and third cases listed above), the output is captured that otherwise would appear in the DOS window. When the command window closes, this output is displayed in the G7 window. No output is displayed in the DOS window. While this often is acceptable

and desirable, it is not suitable for running programs or commands that require user input. For this reason, an option is given to execute the `dos` command in interactive mode. Specify the option by entering `-i` (for interactive) immediately after `dos`. For example,

```
dos -i idbuild.exe master
```

opens a command window and executes the program idbuild with `master` as a command-line argument. This command is displayed in the DOS window, together with output from idbuild. If idbuild requires user input (for example if no configuration file is available), the user will see the prompt in the DOS window and can enter a response. The same option is available in batch mode.

```
dos -i ws newws{
   rem Copying bank %1 to %2.  Type <ctrl> c to cancel.
   pause
   copy %1.ind %2.ind
   copy %1.bnk %2.bnk
   pause
   }
```

Multiple commands may be entered on a single line. The commands must be separated by a semicolon and enclosed with brackets. This capability works when there is only one line of G7 input; everything, including the brackets, must be entered on the same line. In particular, this is useful for entering multiple DOS commands in the G7 command box. The following command opens a DOS window, displays the contents of the directory one page at a time, and pauses at the end:

```
dos -i {dir /p; pause}
```

Note that the system also may offer the ability to enter more than one command per line. In Windows 2000, multiple commands may be separated with an ampersand (&). The above command may be entered as

```
dos -i dir /p & pause
```

dir [arguments]

The dir command prints the contents of the working directory. "arguments" are legal DOS arguments for the system `dir` command. Note, however, that any "#" characters are interpreted as the G7 comment flag, and so any text after # are ignored. For example, to display the listing of files beginning with 't', enter

```
dir t*.*  # This command displays file beginning with 't'.
```

and to display files beginning with 't' in wide format, enter

```
dir t*.* /w
```

Note that the `/p` option will not function properly, because at present there is no means for the G7 user to interact with the operating system in this case (where the user is prompted to hit a key after every page). Because the G7 window may be scrolled, the `/p` option probably is not needed. If necessary, the following G7 entries will allow use of the option, but the results will not appear in the G7 window:

```
dos –i {dir t*.* /p; pause}
dos –i dir t*.* & pause
```

cd [path]

The `cd` command allows users to change the working directory. By default, G7 looks to the current directory to read and write files.

pwd

The `pwd` command prints the path of the present working directory.

break

The `break` command terminates execution of an add file. If an add file calls a second add file, and the second contains the `break` command, then upon processing the `break` command G7 will return to the first add file. It currently is of limited use because G7 lacks other flow control commands (e.g. if-else commands).

seed(<integer>)

The `seed` command allows the user to reinitialize the random number generator. The random number generator is used with the `@rand()` and `@normal()` functions. The `@rand()` function provides draws from the uniform(0,1) distribution, and the `@normal()` function provides draws from the normal(0,1) distribution. If the `seed` command with identical arguments is employed between calls to `@rand()` or `@normal()`, then these random number generators should return identical series.

For example, the G7 runstream

```
seed(20707)
f uni1 = @rand()
seed(20707)
f uni2 = @rand()
```

will produce series uni1 and uni2 that contain identical elements.

close [bank letter] [bank letter] [...]

close all

The `close` command now possesses the ability to close multiple banks at once. For example,

```
close a c d
```

closes the banks assigned as `a`, `c`, and `d`. The `all` option closes all open banks.

4. Quadratic Programming

For many years, G7 offered the ability to "softly" constrain parameters of linear regression models. This convenient feature appears in few other regression packages. Still, it often is desirable to impose inequality and strict equality constraints on regression parameters. Such constraints now may be imposed on single- and multiple-equation linear regression models using the `qpcon` command.

Let us first examine the softly constrained regression model. It may be written as

$$\begin{bmatrix} y \\ r \end{bmatrix} = \begin{bmatrix} X \\ R \end{bmatrix} \beta + \begin{bmatrix} u \\ v \end{bmatrix}$$

where y is the dependent variable, X is the independent variable, and u is the error term. These are stacked above the soft constraints, with constraint values `r`, a linear combination of parameters R, and an assumed disturbance v. For example, for the following soft constraint command, when applied to a model with a constant and one independent variable,

```
con 5 1.0 = 0.5 a2
```

we see that $\mathrm{r} = 1.0$, $\mathrm{R} = \begin{bmatrix} 0 & 0.5 \end{bmatrix}$, **and** $\Psi^{-1} \equiv \left[\mathrm{cov}(v)\right]^{-1} = T * 5$, **where T is the number of observations.**

Parameters for this model may be found by minimizing the sum of squared errors. The optimization problem is specified as

$$\min_{\beta} \beta' \begin{bmatrix} X' & R' \end{bmatrix} \begin{bmatrix} \sigma^2 I & 0 \\ 0 & \psi \end{bmatrix} \begin{bmatrix} X \\ R \end{bmatrix} \beta - 2\beta' \begin{bmatrix} X' & R' \end{bmatrix} \begin{bmatrix} \sigma^2 I & 0 \\ 0 & \psi \end{bmatrix} \begin{bmatrix} y \\ r \end{bmatrix} - y'y$$

where σ^2 **is the error variance and is the assumed variance-covariance matrix for the soft constraints. Note that y'y does not affect the optimum, and thus may be ignored. If we add optional inequality and equality constraints, then the regression model is written as**

$$\min_{\beta} \frac{1}{2}\beta' \begin{bmatrix} X' & R' \end{bmatrix} \begin{bmatrix} \sigma^2 I & 0 \\ 0 & \psi \end{bmatrix} \begin{bmatrix} X \\ R \end{bmatrix} \beta - \beta' \begin{bmatrix} X' & R' \end{bmatrix} \begin{bmatrix} \sigma^2 I & 0 \\ 0 & \psi \end{bmatrix} \begin{bmatrix} y \\ r \end{bmatrix}$$

$$s.t.$$

$$A\beta \leq b$$

$$\beta \geq 0$$

where A specifies linear combinations of the parameters that is less than or equal to vector b. If these additional constraints are included, then the parameter vector automatically is constrained to be nonnegative.

Note that if the nonnegativity and inequality constraints are absent, then Equation can be solved by standard OLS methods. If such constraints are present, then Equation can be solved with quadratic programming methods.

In G7, constraints A and b are added with the `qpcon` command. Soft constraints also may be added with the `con` and `sma` commands. These constraints are imposed on the next linear regression, which is specified by the `r` command. If no constraints are imposed with the `qpcon` command, then G7 finds parameters using OLS. If constraints are imposed with `qpcon`, then G7 finds parameters using a quadratic programming algorithm.

The syntax for the `qpcon` command is as follows:

```
qpcon b <sign> [constant][*]ai [< > [constant][*] aj …]
```

where sign is <, =, or > (<= and >= also are accepted but recorded as strict inequality constraints). The scalars `constant` are optional scalar multiples for the respective parameters. An asterisk may be included to clarify that the parameters are multiplied by the corresponding constant. The parameters are denoted by `ai`, where `i` is the position in the regression equation of the corresponding variable. That is, the parameters are denoted as `a1, a2,…` Right-hand-side terms are separated by either a + or –.

Hopefully, the following example will clarify these ideas.

```
seed(20742)              # reset the random number generator
f x = @cum(x, 1.0, 0.0)  # construct a linear trend
f e = @normal()          # construct stochastic error term
f y = 10.0 + 3.0*x + e   # construct the dependent variable
con 2 3.0 = a2           # softly constrain the x parameter
qpcon 25.0 > a1          # constrain the constant
qpcon 13.0 = 1*a1 + 1*a2 # constrain the sum of parameters
r y = x                  # regress y on x
```

In this regression, the following variables corresponding to Equation are defined:

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ \dots & \dots \end{bmatrix}, R = \begin{bmatrix} 0 & 3 \end{bmatrix}, r = \begin{bmatrix} 3 \end{bmatrix}, y = \begin{bmatrix} 13 + e_1 \\ 14 + e_2 \\ \dots \end{bmatrix}, \Psi = \begin{bmatrix} 1 \\ T \times 2 \end{bmatrix}, \beta = \begin{bmatrix} a1 \\ a2 \end{bmatrix}.$$

The regression model also imposes one inequality and one equality constraint, along with implicit nonnegativity constraints on all parameters. The constraints corresponding to Equation are written as

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 25 \\ 13 \end{bmatrix}$$

where the first equation of $Ax \leq b$ is an inequality constraint and the second is an equality constraint. G7 rescales the objective function by 1/T to reduce the likelihood of overflow; the optimum is not affected by such scaling.

When constraints are imposed with the `qpcon` command, any parameter may be constrained, including the constant. The same is true for soft constraints when the quadratic programming algorithm is employed. Note that reported R^2 statistics may not be accurate if the parameter on the constant term is constrained.

With the exception of marginal explanatory values and normalized residual statistics, the quadratic programming routine and OLS routine report the same statistics. Mexvals and NorRes statistics are not computed. At this point, it is up to the user to verify the validity of remaining statistics. Future versions will offer appropriate statistics, including Lagrange multipliers.

If the `save` command is given before the regression is estimated, then the regression results are stored for inclusion in a larger model. Note that the quadratic programming routine does not support stochastic simulation. This feature is omitted to ensure that the parameters entering the model always satisfy the desired constraints.

Finally, note that the mandatory nonnegativity constraints on quadratic programming parameters need not be restrictive. To allow or to impose negative parameters, simply employ the additive inverse of the desired independent variable.

Constraints also may be employed to impose cross-equation restrictions. In this case, the constraints work with the G7 `sur` and `stack` commands. In the following example, parameters for a Diewert cost function are estimated.

The Diewert cost function is specified as

$$C(P,Q) = Q\sum_i \sum_j b_{i,j}\left(P_i P_j\right)^{\frac{1}{2}}$$

where P are factor prices, Q is output, and b are parameters. The parameters are restricted such that $b_{i,j} = b_{j,i}$, where *i* and *j* index capital and labor. Factor demand equations are derived by differentiating the cost function with respect to the prices. If we then divide by output, we get the following equations for optimal capital-output and labor-output ratios.

$$\left(\frac{K}{Q}\right)_t = \sum_{j=K,L} b_{K,j}\left(\frac{P_j}{P_K}\right)^{\frac{1}{2}}$$

$$\left(\frac{L}{Q}\right)_t = \sum_{j=K,L} b_{L,j}\left(\frac{P_j}{P_L}\right)^{\frac{1}{2}}$$

The parameters may be estimated by `sur` in G7 with the following code.

```
sur{
```

```
r KQ = sqrt_pLpK
r LQ = sqrt_pKpL
qpcon 0 = a2 - b2
}
```

Note that P_i / P_i = 1, and so $b_{K,K}$ and $b_{L,L}$ are estimated as the parameters on the constant terms. The symmetry restriction is imposed with the `qpcon` command. These parameter values may be employed in a second stage to estimate parameters for factor demand equations. Note finally that soft constraints also may be combined with equality and inequality constraints in SUR and stacked-regression models.

5. Conclusion

The latest version of G7 offers the model builder an extended array of powerful tools. In particular, these tools will greatly ease the task of constructing multi-sector databases, and they also should prove useful in the quest to find suitable regression parameters.

Nevertheless, a word of caution is in order. Many of the features described here are new and relatively untested. In particular, the equality and inequality constraint features are incomplete and somewhat inadequate. Very likely, some bugs remain undetected in this code. Still, I hope you find them useful. I hope to have an improved version of G7 ready to distribute within a few months. Please let me know of your questions and ideas. As always, please report bugs that you find.