Grocer 1.0, an Econometric Toolbox for Scilab: an Econometrician Point of View.

Éric Dubois
Direction Générale du Trésor et de la Politique Économique
Télédoc 679
139, rue de Bercy
75012 PARIS
e-mail: Grocer.toolbox@free.fr

Abstract

Grocer is an econometric toolbox for Scilab, a free opensource matrix-oriented toolbox similar to Matlab and Gauss. It contains more than 50 econometric different methods, with many variants and extensions. Most standard econometric tools are available. Grocer contains also two original econometric tools: a function allowing the "automatic" estimation of the "true model" starting from a more general and bigger one, a method that provides the most thorough expression of the so-called LSE econometric methodology; a function calculating the contributions of exogenous variables to an endogenous one.

JEL: c870, c130, c220, c320

keywords: econometric software, estimation, general to specific, contributions.

1 Introduction

Grocer, available at http://dubois.ensae.net/grocer.html, is an econometric toolbox for Scilab, a free opensource matrix-oriented software very similar to Matlab and Gauss and available at http://scilabsoft.inria.fr. It is built both upon the translation of existing programs in Matlab or Gauss and upon native programs. Grocer has some properties that can make it appealing to an applied econometrician. First, it is free and opensource, and therefore portable and very flexible. Second, it contains most standard econometric procedures. And third, it has a few more or less common features that can make its use fruitful: it implements the so-called LSE methodology as regards univariate econometric estimation; in particular Grocer implements the more recent advances of this methodology, namely a *pc-gets* like program called automatic in grocer; it contains a host of multivariate methods from standard "old-fashioned" simultaneous

equations methods to many up to date VAR methods, such as VARMA, Vector Error Correction Models, Bayesian Vector Autoregressions; it proposes a calculation of the contributions of exogenous variables to an endogenous one for any dynamic equation.

The aim of this paper is to provide the reader with an insight into Grocer most interesting features from an econometrician point of view. The interested reader can find a detailed description of all econometric functions available in Grocer in Dubois (2004a) and explanations about their implementation in Dubois (2004b).

Part 2 of this paper presents Grocer basic features. Part 3 discusses how the LSE methodology has been implemented. Part 4 presents the multivariate methods. Part 5 explains the contributions method. Part 6 presents quickly the other functions available in Grocer. Part 7 concludes.

2 Grocer basic features

Grocer is basically a toolbox for Scilab. So a few words about Scilab are necessary (the user can find a sligthly more detailed description in Dubois (2004a) and a more detailed presentation in Scilab (1993) for an old but still useful tutorial or on Scilab web site - the on-line help at http://scilabsoft.inria.fr/product/man/html/eng/index.htm or).

2.1 A few words about Scilab

Scilab, developed by INRIA in France, is a matrix-oriented software and it is therefore very convenient for the programming of econometric functions. It looks much like Matlab: in fact, Scilab contains a translator from Matlab, that works well, even if it is not perfect. Commands can be run directly on the command window; they can also be gathered in scripts or in functions. Scilab functions can be compiled, notably to be added to the Scilab distribution: Grocer is in fact built upon more than 300 Scilab functions, that are compiled the first time the user runs Scilab after Grocer has been installed (which is made very simply by unzipping the Grocer distribution in the Scilab directory, see Dubois (2004a) for details).

As Gauss and Matlab, Scilab contains a robust numerical optimisation program, and therefore allows to implement any likelihood maximisation.

Scilab contains numerous data types: real and complex matrices, booleans, strings, lists, functions, ... It contains also a specific type, the typed list, which is at the basis of Grocer time series type.

Once you have installed Scilab on your computer, and thereafter Grocer, then running the software will lead the user to the following window:

```
-------------------------------------------
```

```
                        scilab-3.0

                Copyright (c) 1989-2004
            Consortium Scilab (INRIA, ENPC)
        -----------------------------------------


Startup execution:
  loading initial environment
  loading Grocer 1.0 / Copyright Eric Dubois et al. 2004

-->
```

Scilab commands are executed at the prompt `-->`. In what follows, every time an example will be presented, all the necessary commands will be exposed as they are run in Scilab. The results will be shown as they appear in Scilab window, with the font: `verbatim`. For example, showing how to add the numbers 4 and 5 in Scilab should be done as follows:

```
-->4+5
```

And the result should be presented as follows:

```
 ans  =

    9.
```

2.2 The time series type

Grocer can deal with matrices or vectors of data, but also with time series. Time series can be created through the importation of an Excel database (by Grocer function `impecx2bd` that can also be used to import vectors or matrices), by any operation on other time series or with the Grocer command `reshape`, which transforms a vector into a time series. Any frequency is allowed, but annual, quarterly and monthly time series have a specific, more user friendly representation than other frequencies.

Standard operations (addition, subtraction, multiplication and division) and functions (logarithm, exponential, sin, cosin,...) can be performed on time series as if they were vectors. For instance, ts1+ts2 denotes the addition of the 2 time series ts1 and ts2. NA (Non Available) values are authorized and the 2 time series ts1 and ts2 need not cover the same period (but need to have the same frequency): in that case the new time series is defined over the time period common to both time series. It should be noted that the time period is an attribute of the time series itself, not of the database where it possibly comes from: provided that they have the same frequency, time series from different databases can be added, subtracted,... without any

difficulty.

Time series allow the user to apply all econometric procedures to data with different lengths without having to transform them manually. The user can choose the time period over which to perform the estimation: this is done by setting the time bounds with the Grocer command `bounds`. For instance setting the bounds from the 3rd month of 1990 to the 10th month of 2003 involves the following Grocer command: `bounds('1990m3','2003m10')`. Bounds can also be discontinuous, such as: `bounds('1990m3','1995m12','2003m10')`. Lastly, the user can choose not to provide her own time bounds: in that case, the econometric procedure will choose the greatest possible time bounds starting from the first non NA values.

A few specific functions have been programmed to lag (function `lagts`), differentiate (function `delts`) time series, compute the growth rate (`growthr`) of a time series, aggregate monthly time series to quarterly (`m2q`) or quarterly to annual (`m2a`), extrapolate a time series by the value (`overlay`) or the growth rate (`extrap`) of another one...

3 The application of the LSE methodology: ordinary least squares, specification tests and the pc-gets like function automatic

Grocer is oriented towards the implementation of the LSE methodology. Grocer contains many specification tests (part 3.2) and implements the pc-gets methodology that represents the most thorough expression of this methodology (part 3.3). Before that, a short presentation of `ols`, the function performing ordinary least squares in Grocer, is required (part 3.1).

3.1 Ordinary least squares

There are several functions performing ordinary least squares in Grocer, that are more or less user-friendly and complete. The most user-friendly and complete one is `ols`. As any high-level Grocer econometric function, `ols` can be applied to various data types: matrices, vectors, time series as well as strings. Two kinds of strings are allowed: first, the strings 'const' or 'cte'[1], that avoid the user to create the constant vector or time series relevant to her problem[2]; second, the name of an existing variable between quotes. The variable `lm1` which is the log of the monetary aggregate m1 can for instance be entered as such (e.g: `lm1`) or between quotes (`'lm1'`). In the last case Grocer function is able to display the names of its inputs.

For instance estimating Hendry and Erisson (1991) equation (6) can be performed as follows.

[1]The string 'cte', that is an abbreviation of the French world 'constante', was first implemented; to maintain compatibility with earlier versions of Grocer, used by some forerunners, both abbreviations are available in the 1.0 version.

[2]Some other usual variables, such as trends, could also be treated this way; this is left for further releases.

First, load the corresponding database that is given with Grocer package, that contains the time series lm1, ly, lp and rnet:

```
-->load('SCI/macros/grocer/db/bdhenderic.dat') ;
```

Then set the time bounds:

```
-->bounds('1964q3','1989q2')
```

And lastly perform the estimation:

```
-->rhe=ols('delts(lm1-lp)','delts(lp)','delts(lagts(1,lm1-lp-ly))',
'rnet','lagts(1,lm1-lp-ly)','const');
```

The result is then:

```
ols estimation results for dependent variable: delts(lm1-lp)
estimation period: 1964q3-1989q2
number of observations: 100
number of variables: 5
R^2 = 0.7616185      adjusted R^2 = 0.7515814
Overall F test: F(4,95) = 75.880204       p-value = 0
standard error of the regression: 0.0131293
sum of squared residuals: 0.0163761
DW(0) =2.1774376
Belsley, Kuh, Welsch Condition index: 114

variable                  coeff      t-statistic p value
delts(lp)                 -0.6870384 -5.4783422  3.509D-07
delts(lagts(1,lm1-lp-ly)) -0.1746071 -3.0101342  0.0033444
rnet                      -0.6296264 -10.46405   0
lagts(1,lm1-lp-ly)        -0.0928556 -10.873398  0
const                     0.0234367  5.818553    7.987D-08
```

3.2 Specification tests

In the spirit of the LSE methodology Grocer can compute many *specification* tests. Available specification tests include: the Lagrange multiplier autocorrelation test (Grocer function `arlm`); the ARCH test (Grocer function `archz`); White heteroskedascity test (Grocer function `white`) as well as a function allowing any heteroskedascity test of Breusch and Pagan type (Grocer function `bpagan`) ; in-sample and out-of-sample Chow stability tests (Grocer functions `chowtest` and `predfail`) ; Brown, Durbin and Evans cusum backward and forward stability tests (Grocer

function `cusumb` and `cusumf`); Ramsey RESET test of linearity (Grocer function `reset`); Jarque and Bera and Doornik and Hansen normality tests (Grocer functions `jbnorm` and `doornhans`). Grocer provides also a function performing Wald restriction tests (Grocer function `waldf`).

As an example, performing the Lagrange multiplier autocorrelation test on the equation estimated above is done that way:

```
--> arlm(rhe,4);

Lagrange multiplier 1-4 autocorrelation test:
chi2(4)=7.1563181
(p -value                   = 0.1278545)


Lagrange multiplier 1-4 autocorrelation test:
F(4,91)=1.941783
(p -value                   = 0.1102067)
```

The reader can check that it is exactly the result presented by Hendry and Ericsson (1991) themselves.

3.3 The function `automatic`

Grocer contains the most recent developments of the LSE methodology, that is a function `automatic` that mimics Hendry and Krolzig package pc-gets [3]. This package aims at replicating automatically the approach followed by Hendry and his fellows when estimating a model. As far as I know, Grocer is with pc-gets the only econometric package today to provide such a program.

3.3.1 Theoretical background

Suppose that a researcher wants to recover a data generating process (DGP) starting from a data set with some variables that are relevant and other not.

Let the DGP be:
$E(y_t/x_t) = x_t \times b$
where x_t is a $(1 \times p)$ vector of variables $x_1^t, \ldots, x_k^t$ and b a $(p \times 1)$ vector of parameters.

The approach proposed by Hendry and Krozlig (2000), based upon Hoover and Perez (1999), is the following one (see also the presentation in figure 1).

Let $Z = \{z_1, \ldots, z_n\}$ be a set of variables which are potentially relevant, with $x_1, \ldots, x_k$ belonging to Z and let $E(y_t/z_t) = z_t \times \beta$ be the corresponding postulated general model.

[3] Informations about pc-gets are available at http://www.doornik.com/pcgive/pcgets/index.html.

Let $\alpha, \eta, \varphi, \gamma$ be 4 real numbers between 0 and 1: these are significance levels; usually, they should be equal to 0.05 or 0.01 (except for η). Let Ψ be a real number greater than 1.

The algorithm consists in the following potential 5 steps, the algorithm stopping when it has found the final model:

First step: Estimation of the general model

i) The general model is estimated, which leads to a set of estimated coefficients. If all variables are significant at the α level, then the general model is the final model.

ii) If at least one coefficient is non significant at the pre-specified α level, then perform k significance tests. If these specification tests do not all pass the pre-specified level [4], then adjust the significance level of the corresponding tests by a factor Ψ (the fact that at least one of the specification test is rejected should lead the user to add other variables to the information set, but this is clearly a task that the computer is not able to do, so the process goes on, but the significance level is adjusted to avoid rejecting too many subsequent models, if not all).

Let then $s^* = [s_1^*; \ldots; s_k^*]$ be the corresponding vector of significance levels:

$s_i^* = \eta$ if the corresponding test has been successfully passed,

$s_i^* = \frac{\eta}{\Psi}$ if the corresponding test has not been successfully passed.

Second step: Elimination of globally insignificant variables (programmed in function `auto_stage0`)

i) Range all from the lowest absolute value of Student's t to the greatest. Set (for initialisation sake):

$j = 1; F = 0$ and $s = s^*$

ii) while $F \leq \Psi$ and $s \leq s^*$ then:

estimate the model with the j least significant variables withdrawn; calculate the Fisher test, F, of this model against the general model; calculate the vector s of p-values of the k specification tests.

iii) call the second step model the last model that verifies $F \leq \Psi$ and $s \leq s^*$

Third step: Multiple reduction paths (programmed in function `auto_stage1`)

For all insignificant coefficients at level α of the second step model:

i) Drop the corresponding variable from the list of exogenous variables and estimate the corresponding model M1.

ii) Calculate the vector s of p-values of the k specification tests.

iii) if it exists $s_i \leq s_i^*$ then:

stop the process for this variable, do not store the corresponding model and go to the following insignificant variable of the second step model, if any.

else:

[4] In practice, parameter η can be differentiated with respect to the specification tests in that case, it is better to interpret η as a vector; this is indeed the way η is treated in Grocer implementation

a) in the estimation of M1, search an insignificant variable at level α, starting from the variable with the lowest Student's t to the one with the greatest Student's t, until the corresponding model verifies $s_i \geq s_i^*$ for all i (that is a model which passes all specification tests).

b) if there is no such variable, then stop the process for this variable, store model M1 in the list of third step models and do again all step 3 process with the following insignificant variable of the third step model, if any.

c) if there is such a variable and if the corresponding model has already been encountered on a preceding exploration in step 3, then stop the process for this variable and restart all step 3 process with the following insignificant variable of the second step model, if any.

d) if there is such a variable and if the corresponding model has not already been encountered on a preceding exploration in step 3, then call M1 this new model and restart the process at step 3 ii) a).

At the end of step 3, the list of third step models contains zero, one or a few models.

Fourth step: Encompassing

i) if the list of third step models is empty, then the final model is the general model.

ii) if the list of third step models contains only one model, then this model is the final model.

iii) if the list of third step models contains more than one model, then build the union of all these models and tests, at significance level γ, all third step models against this union model.

a) if all third step models are rejected against the union model, then the union model is the final model.

b) if only one third step models is not rejected against the union model, then this third step model is the final model.

Fifth step: redoing the process with the union model

Fifth step occurs only when there are more than one third step model not rejected against the union model.

In that case call the union model the new general model and restart the process with this new general model at the beginning of third step.

If, at the end of the process, there are still more than one third step model not rejected against the new union model, then select the final model on the basis of an information criterion.

*
* *

The exploration of several paths and the use of specifications tests cover against the risk of eliminating a relevant variable, but only the more relevant paths are indeed explored. Hendry

and Krozlig Monte Carlo simulations show that this method leads indeed to very satisfactory results: the average inclusion rate of a non relevant variable can be set at a low level, while retaining significant power.

3.3.2 An example: Hendry and Ericsson (1991) revisited

To show the power of this method, we plug Hendry and Ericsson (1991) equation in a much bigger set of potential explanatory variables : we add 2 to 4 lags of the variable $\Delta(lm1-ly-lp1)$ (variables called here 'delts(lagts(2,lm1-lp-ly))', 'delts(lagts(3,lm1-lp-ly))', 'delts(lagts(4,lm1-lp-ly))'), 0 to 4 lags of variable $\Delta(ly)$ (variables called here ('delts(ly)' to 'delts(lagts(4,ly))'), 0 to 4 lags of the variable $\Delta(\Delta(lp))$ and 1 to 4 lags of the variable $\Delta(rnet)$ (variables called here 'delts(rnet)' to 'delts(lagts(4,rnet))'): this seems a sensible parametrisation of a set containing 0 to 4 lags of every exogenous variable (the log of GDP, ly, the inflation rate, $\Delta(lp)$, the interest rate, rnet and lags of the log of m1, lm1).
Load the database again:

```
-->load('SCI/macros/grocer/db/bdhenderic.dat') ;
```

Set the same time bounds:

```
-->bounds('1964q3','1989q2')
```

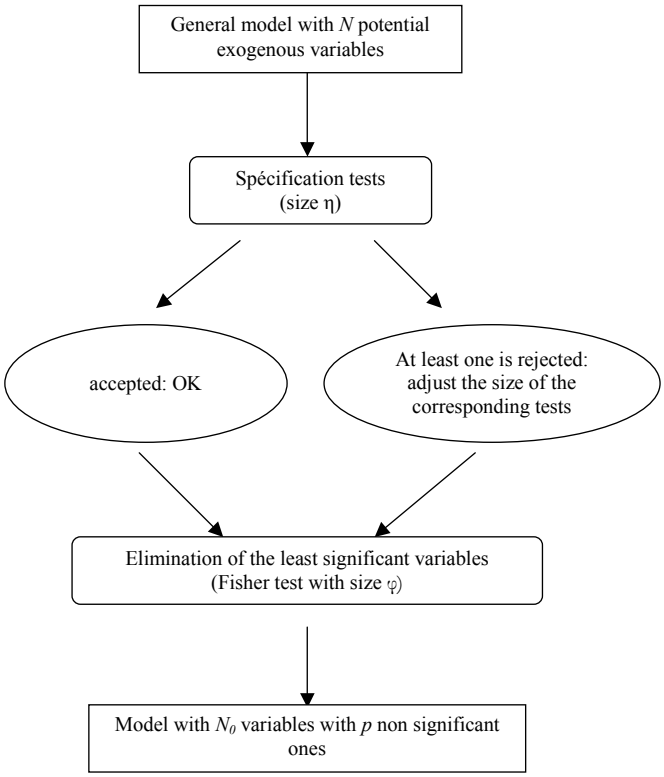
And use the function `automatic` with until 4 lags of each regressor:

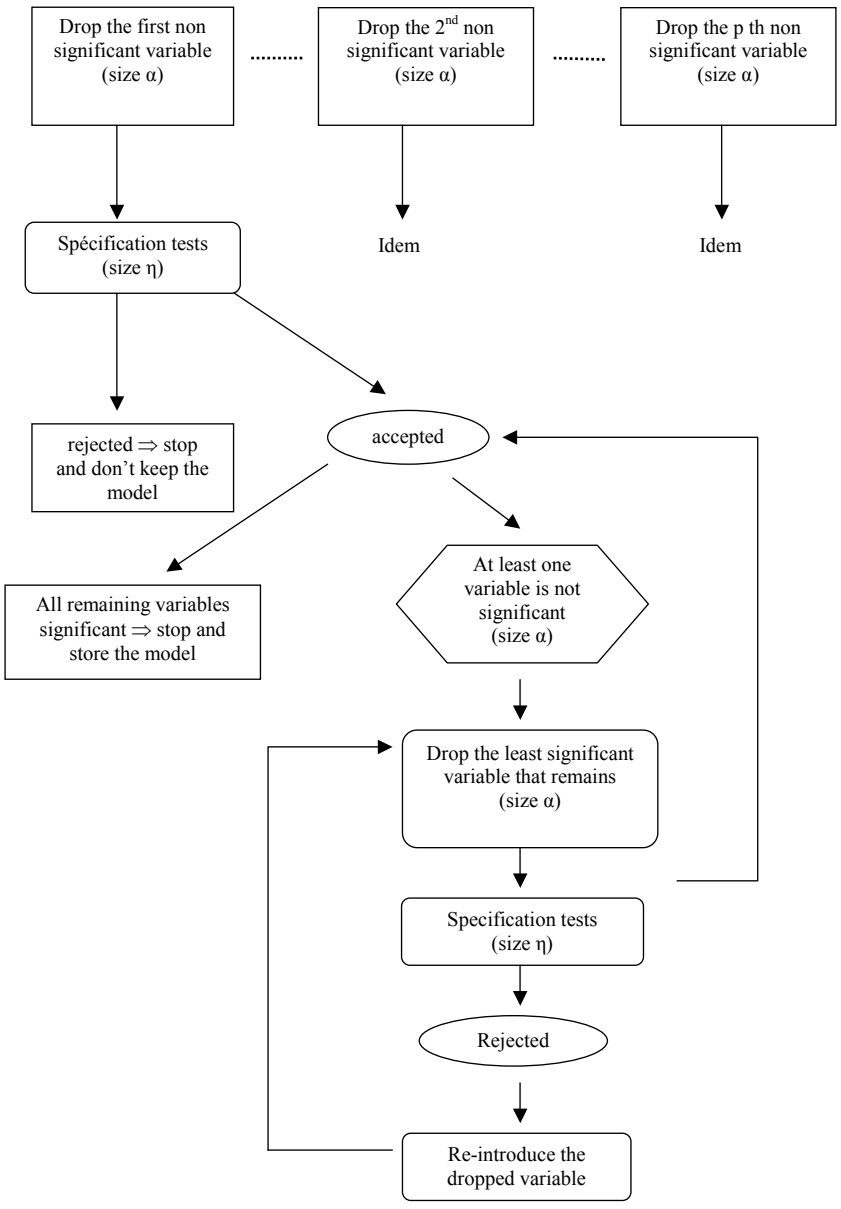
```
[r1]=automatic('delts(lm1-lp)','lagts(lm1-lp-ly)',...
'delts(lp)','rnet','delts(lagts(lm1-lp-ly))','delts(lagts(2,lm1-lp-ly,...
,'delts(lagts(3,lm1-lp-ly))','delts(lagts(4,lm1-lp-ly))',...
'delts(ly)','delts(lagts(1,ly))','delts(lagts(2,ly))',...
'delts(lagts(3,ly))','delts(lagts(4,ly))',...
'delts(delts(lp))','delts(delts(lagts(1,lp)))','delts(delts(lagts(2,lp)))',...
'delts(delts(lagts(3,lp)))','delts(delts(lagts(4,lp)))',...
'delts(rnet)','delts(lagts(1,rnet))','delts(lagts(2,rnet))',...
'delts(lagts(3,rnet))','delts(lagts(4,rnet))',...
'cte')
```

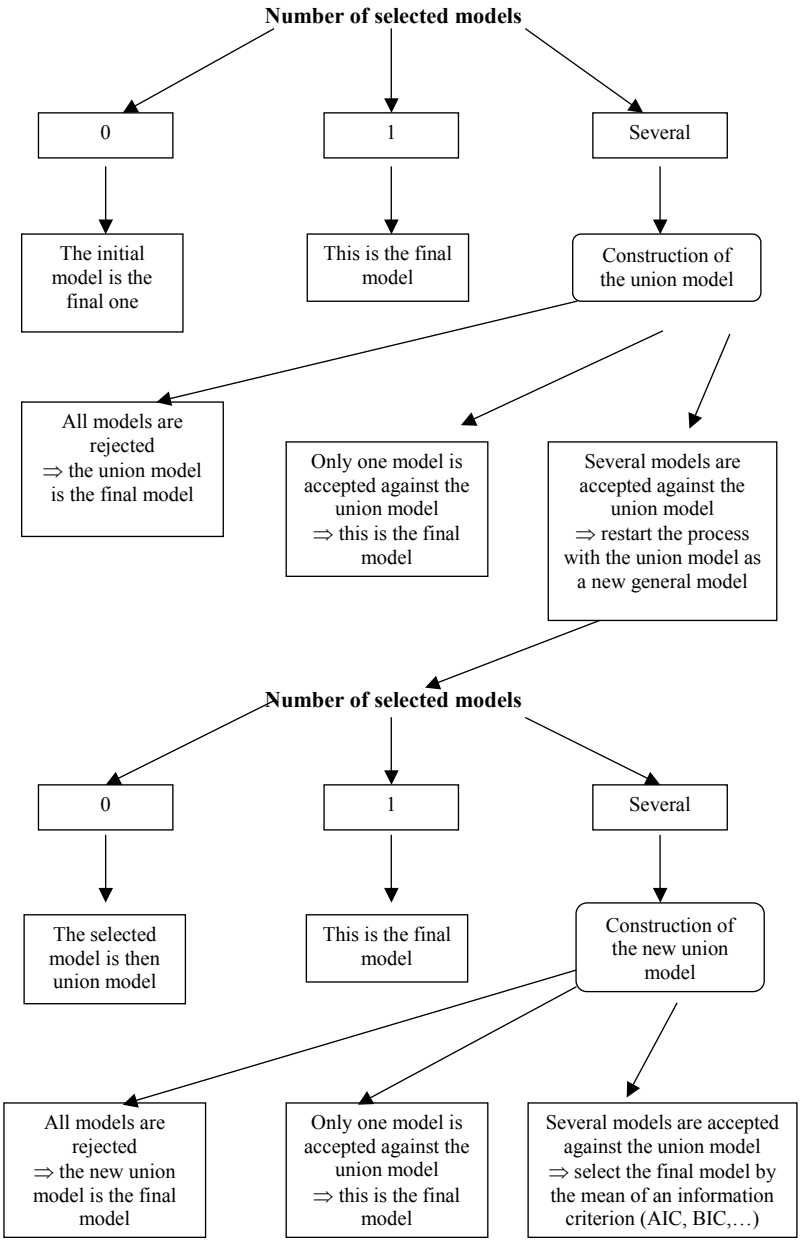
Grocer results are then:

```
 ------------------------------------------------
| results of the automatic regression package    |
 ------------------------------------------------

final model

ending reason: only one model selected by stage 1
```

Figure 1: estimation and reduction of the initial model




Drop the first non significant variable (size α)
.....
Drop the 2nd non significant variable (size α)
.....
Drop the p th non significant variable (size α)
Spécification tests (size η)
Idem
Idem
rejected ⇒ stop and don't keep the model
accepted
All remaining variables significant ⇒ stop and store the model
At least one variable is not significant (size α)
Drop the least significant variable that remains (size α)
Specification tests (size η)
Rejected
Re-introduce the dropped variable


Number of selected models
0
1
Several
The initial model is the final one
This is the final model
Construction of the union model
All models are rejected ⇒ the union model is the final model
Only one model is accepted against the union model ⇒ this is the final model
Several models are accepted against the union model ⇒ restart the process with the union model as a new general model
Number of selected models
0
1
Several
The selected model is then union model
This is the final model
Construction of the new union model
All models are rejected ⇒ the new union model is the final model
Only one model is accepted against the union model ⇒ this is the final model
Several models are accepted against the union model ⇒ select the final model by the mean of an information criterion (AIC, BIC,…)

```
ols estimation results for dependent variable: delts(lm1-lp)
estimation period: 1964q3-1989q2
number of observations: 100
number of variables: 9
R^2 = 0.7941498      adjusted R^2 =0.7760531
Overall F test: F(8,91) = 43.883636       p-value = 0
standard error of the regression: 0.0124659
sum of squared residuals: 0.0141413
DW(0) =2.1391056
Belsley, Kuh, Welsch Condition index: 166

variable                   coeff      t-statistic p value
delts(delts(lagts(2,lp))) 0.3821030  2.4121215   0.0178669
delts(lagts(3,lm1-lp-ly)) -0.1884416 -2.8830509  0.0049142
delts(lagts(lm1-lp-ly))   -0.2149036 -3.6961388  0.0003737
delts(rnet)               0.3198695  2.680324    0.0087312
delts(lagts(3,ly))        -0.3393305 -3.0482243  0.0030136
delts(lp)                 -0.7653748 -6.2359835  1.386D-08
cte                       0.0300893  6.9787739   4.690D-10
lagts(lm1-lp-ly)          -0.1104120 -10.974     0
rnet                      -0.7746108 -10.886597  0


                        *
                      *     *


tests results:
**************
test                    test value p-value
Chow pred. fail. (50%) 0.4727966  0.9939980
Chow pred. fail. (90%) 0.8364555  0.5951258
Doornik & Hansen       2.248449   0.3249043
AR(1-4)                0.6915387  0.5997762
hetero x_squared       1.1282701  0.3459758


                        *
                      *     *
```

One can note that the program finds new significant variables besides the ones found by Hendry and Ericsson (variables delts(delts(lagts(2,lp))), delts(lagts(3,lm1-lp-ly)), delts(delts(lagts(2,lp)))

and delts(lagts(3,ly))), and that the standard error of the regression is lower, which proves the usefulness of the program (whether these variables are economically meaningful is another question, beyond the scope of this paper).

4 Multivariate methods

Grocer contains the old simultaneous equations estimation methods (Seemingly Unrelated Regressions, two and three stages least squares) as well as the more recent VAR ones[5]. As regards the VAR methodology, the following procedures are currently available: VAR estimation, impulse response calculation, ecm estimation, bayesian VAR and its ecm form, VAR forecasting. Lastly, Grocer contains a program performing ARMA and VARMA estimations.

4.1 "Old-fashionned" simultaneous equation methods

Grocer Function `sur` is rather general: it can be applied to any system of equations, provided it is linear in its coefficients. So, you can impose constraints to the coefficients in the system, provided they remain linear.

Take as an example the sur estimation on Grunfeld's investment data presented in Greene's (2003) textbook (in chapter 14):

```
eq1='igm=a1*fgm+a2*cgm+a3'
eq2='ich=a4*fch+a5*cch+a6'
eq3='ige=a7*fge+a8*cge+a9'
eq4='iwest=a10*fwest+a11*cwest+a12'
eq5='iuss=a13*fuss+a14*cuss+a15'
```

Then the sur estimation is performed by Grocer instruction:

```
-->r=sur(eq1,eq2,eq3,eq4,eq5);
```

If you want to impose for instance the constraint that all constants are equal (for sure, a meaningless constraint on economic grounds,...), then you should write:

```
eq1='igm=a1*fgm+a2*cgm+a3'

eq2='ich=a4*fch+a5*cch+a3'

eq3='ige=a6*fge+a7*cge+a3'
```

[5]These programs, as well as most univariate econometric methods, all multicollienarity tests and outlier diagnostics, the Kalman filter, some unit root and cointegration tests rest mainly on the translation and adaptation of parts of LeSage Matlab toolbox -see LeSage (1999)

```
eq4='iwest=a8*fwest+a9*cwest+a3'

eq5='iuss=a10*fuss+a11*cuss+a3'

r=sur(eq1,eq2,eq3,eq4,eq5)
```

Functions performing two stage (function `twosls`) and three stage (function `threesls`) least squares have the same syntax. Note that the function `twosls` is here reserved to two stage least squares in a system of equations: two stage least squares applied on a single equation is performed through function `iv`. As with `sur`, the only compulsory arguments are the texts of the equations of the system. In that case, the program determines what are the coefficients and what are, in each equation, the endogenous variables. This imposes however constraints on the way equations are written: as with `sur`, the coefficients must be named $a1, a2, \ldots, an$ without any discontinuity; and the names of the endogenous variables must be exactly equal to the left hand sides of the equations.

The command below illustrates the simplest way of performing a Two Stage Least Squares estimation (see Dubois(2004a) for the results of estimation). It is contained in function `twosls_d`, which belongs to the library macros/grocer/multi in the library where the user has chosen to install Scilab:

```
--> rt=twosls('y1=a+b*x1','y2=d+e*(y1-x2)+f*x2','coef=a;b;d;e;f')
```

4.2 VAR methods

Grocer provides several tools to perform VAR estimations:

- VAR estimation itself: function `var` (and its low level counterpart `var1`);
- Impulse response functions (function `irf`) calculated either with an asymptotic formula or with a Monte-Carlo simulation;
- Bayesian VARs (function `bvar`), the user being able to impose her own prior;
- Error Correction estimation: `ecm` and a bayesian version, `becm`; in each case, the cointegration relationships are estimated by a Johansen estimation (see section 6); the user can either impose her own number of cointegration relationships or let the program determine the number of cointegration relationships at a prespecified level (1%, 5% or 10%).
- lastly, the user can forecast any of these VAR with the function `varf`.

As an example, here is the result of the estimation of the bayesian error correction model presented in LeSage (1999) (chapter 5 pp. 128-129).

First, load the database:

```
-->load('SCI/macros/grocer/db/datajpl.dat')
```

Then perform the estimation:

```
-->results = becm(2,0.1,0.5,1,'endo=[illinos;indiana;kentucky;michigan;...
-->ohio;pennsyvlania;tennesse;westvirginia]');

Johansen estimation results for variables:
illinos, indiana, kentucky, michigan, ohio, pennsyvlania, tennesse, westvirginia
time order: 0
# of lags: 2

NULL:    Trace Statistic Crit 90% Crit 95%  Crit 99%
r <= 0   214.38998       153.6341 159.529   171.0905
r <= 1   141.48161       120.3673 125.6185  135.9825
r <= 2   90.363179       91.109   95.7542   104.9637
r <= 3   61.554975       65.8202  69.8189   77.8202
r <= 4   37.103415       44.4929  47.8545   54.6815
r <= 5   21.069871       27.0669  29.7961   35.4628
r <= 6   10.605086       13.4294  15.4943   19.9349
r <= 7   3.1924592       2.7055   3.8415    6.6349

conclusions from the trace statistics:
at a 10% level, there are 8 cointegration relation(s)
at a 5% level, there are 2 cointegration relation(s)
at a 1% level, there are 2 cointegration relation(s)

NULL:    Max Eigenvalues Statistic  Crit 90%  Crit 95%  Crit 99%
l <= 0   72.908372                  49.2855   52.3622   58.6634
l <= 1   51.118434                  43.2947   46.2299   52.3069
l <= 2   28.808204                  37.2786   40.0763   45.8662
l <= 3   24.45156                   31.2379   33.8777   39.3693
l <= 4   16.033544                  25.1236   27.5858   32.7172
l <= 5   10.464785                  18.8928   21.1314   25.865
l <= 6   7.4126267                  12.2971   14.2639   18.52
l <= 7   3.1924592                  2.7055    3.8415    6.6349

conclusions from the maximal eigenvalues statistics:
at a 10% level, there are 8 cointegration relation(s)
at a 5% level, there are 2 cointegration relation(s)
at a 1% level, there are 1 cointegration relation(s)
```

```
*** cointegrating vectors from johansen estimation ***

variable     vector # 1  vector # 2
illinos      0.0131635   -0.0099366
indiana      -0.3050443  0.0399855
kentucky     0.0525625   -0.0273951
michigan     -0.0417678  0.1871467
ohio         0.0176776   0.0199591
pennsyvlania -0.0526335  0.0250484
tennesse     0.0823386   -0.2737890
westvirginia 0.0201478   0.0216022


                        *
                    *       *


becm estimation results for variables
del(illinos), del(indiana), del(kentucky), del(michigan), del(ohio),
del(pennsyvlania), del(tennesse) and del(westvirginia)

PRIOR hyperparameters
tightness = 0.1
decay = 1
symetric weights based on 0.5

estimation results for dependent variable del(illinos)
number of observations: 170
number of variables: 19
R^2 = 0.3004672      adjusted R^2 =0.2170792
standard error of the regression: 3.6390646
sum of squared residuals: 14.821402
DW(0) =2.0273897

variable                coeff      t de student p value
del(illinos)(-1)        0.1439182  1.3753802    0.1708340
del(indiana)(-1)        0.4953484  2.3979857    0.0175751
del(kentucky)(-1)       0.0380825  0.7010216    0.4842536
del(michigan)(-1)       0.0176187  0.2418778    0.8091681
del(ohio)(-1)           -0.1991313 -1.8615471   0.0644036
del(pennsyvlania)(-1)   0.1826746  3.48336      0.0006305
del(tennesse)(-1)       0.1300340  0.6176380    0.5376453
del(westvirginia)(-1)   -0.0349045 -1.4624539   0.1454740
```

```
del(illinos)(-2)         -0.0556921 -0.5565481   0.5785724
del(indiana)(-2)         0.0201131  0.0987109    0.9214848
del(kentucky)(-2)        0.0399308  0.7617436    0.4472748
del(michigan)(-2)        0.0290325  0.4069044    0.6845931
del(ohio)(-2)            -0.0995948 -1.0386543   0.3004493
del(pennsyvlania)(-2)    0.0557372  1.0787899    0.2822187
del(tennesse)(-2)        -0.0717670 -0.3583999   0.7204913
del(westvirginia)(-2)    -0.0045788 -0.2121682   0.8322315
lag 1 of coint. vec. #1 0.1081266  0.3911411    0.6961856
lag 1 of coint. vec. #2 -0.2929133 -1.0600059   0.2906545
cte                      3.4135785  0.9487682    0.3440932

                          *
                        *     *
```

[...] (results for the variables indiana, kentucky, michigan, ohio, pennsyvlania and tennesse are here omitted for the sake of brevity; they are of course given by the function `becm`)

```
estimation results for dependent variable del(westvirginia)
number of observations: 170
number of variables: 19
R^2 = 0.1970140      adjusted R^2 =0.1012939
standard error of the regression: 17.478053
sum of squared residuals: 341.89747
DW(0) =2.0311438

variable                 coeff      t de student p value
del(illinos)(-1)         0.1095202  0.2204470    0.8257891
del(indiana)(-1)         1.5753669  1.591901     0.1132757
del(kentucky)(-1)        0.3796700  1.4551827    0.1474736
del(michigan)(-1)        0.0409345  0.1170027    0.9069969
del(ohio)(-1)            -0.7621534 -1.4819112   0.1402261
del(pennsyvlania)(-1)    -0.5749257 -2.2692817   0.0245155
del(tennesse)(-1)        -0.0356746 -0.0352836   0.9718952
del(westvirginia)(-1)    -0.0166022 -0.1428371   0.8865890
del(illinos)(-2)         0.1730335  0.3750635    0.7080835
del(indiana)(-2)         -1.3777092 -1.420519    0.1572990
del(kentucky)(-2)        0.1541704  0.6121892    0.5412353
del(michigan)(-2)        0.2511154  0.7332133    0.4644442
del(ohio)(-2)            0.3895893  0.8455066    0.3990235
del(pennsyvlania)(-2)    -0.0116675 -0.0461857   0.9632167
del(tennesse)(-2)        0.6676482  0.6941234    0.4885578
del(westvirginia)(-2)    -0.1099663 -1.0124952   0.3127487
lag 1 of coint. vec. #1 -2.2584975 -1.7072336   0.0896151
```

```
lag 1 of coint. vec. #2 -2.3590674 -1.7774235   0.0772971
cte                     8.0747728  0.4674689    0.6407666

                          *
                        *     *
```

4.3 VARMA

Grocer also provides ARMA and VARMA: they are implemented by the same function `varma`[6]. As an example, here is how one can estimate the model proposed by Jenkins and Alavi (1981) to describe the relationship between the minks and the muskrats skins traded by the Hudson's Bay Company between 1948 and 1909.

First, load the (Matlab) data base:

```
--> mtlb_load('SCI/macros/grocer/db/mink.dat');
```

Then transform the data:

```
--> z2 = mink(:,2)-mean(mink(:,2));
--> z = [z1,z2(2:62)];

--> muskrat=z1
--> mink=z2(2:62)
```

One of the model estimated by Jenkins and Alavi takes the following form:

$$\begin{pmatrix} 1+\phi 1_{1,1}(L)+\phi 2_{1,1}(L^2)+\phi 3_{1,1}(L^3)+\phi 4_{1,1}(L^4) & 0 \\ 0 & 1+\phi 1_{2,2}(L)+\phi 2_{1,1}(L^2) \end{pmatrix} \begin{pmatrix} muskrat \\ mink \end{pmatrix} = \begin{pmatrix} 1+\theta 1_{1,1}(L) & \theta 1_{1,2}(L) \\ \theta 1_{2,1}(L) & 1+\theta 1_{2,2}(L) \end{pmatrix} \begin{pmatrix} \epsilon_{1t} \\ \epsilon_{2t} \end{pmatrix}$$

Its estimation in Grocer involves the definition of the matrices `phi1` to `phi4`, `theta` and `sigma`. This is done by the following instructions, where the sign '=' in the matrices indicates that the coefficients are constrained to the given value:

```
--> phi1 = ['0','=0';'=0','0'];
--> phi2 = ['0','=0';'=0','0'];
--> phi3 = ['0','=0';'=0','=0'];
--> phi4 = ['0','=0';'=0','=0'];
--> theta = [0,0;0,0];
--> sigma = [0,0;0,0];
```

[6]This function rests on a translation and adaptation of the Matlab package E4, developed by Terceiro et al. (2000)

```
--> result = varma(['muskrat' 'mink'],[phi1,phi2,phi3,phi4],[],theta,[],sigma,1);
```

And here is the result given by Grocer:

```
*************** VARMA estimation Results for model: ***************
Y = e
with:
      [ muskrat  ]
- Y = [ mink     ]
- V(e) = V


Log-likelihood: 9.0059411
Information criteria: AIC = 0.1309527, BIC = 0.5808111

Parameter Estimate   Std. Dev. t-test
AR1(1,1)  -0.6887091 0.1363550 -5.050854
AR1(2,2)  -1.2679083 0.1098543 -11.541726
AR2(1,1)  0.5941061  0.1402932 4.2347476
AR2(2,2)  0.5592538  0.0921302 6.0702529
AR3(1,1)  -0.0681945 0.1171420 -0.5821526
AR4(1,1)  0.2815736  0.0855952 3.2895965
MA1(1,1) -0.2965995 0.1606131 -1.8466709
MA1(2,1) 0.602702   0.0803646 7.4995981
MA1(1,2) -0.8641752 0.1387191 -6.2296768
MA1(2,2) -0.8351471 0.1529343 -5.4608236
V(1,1)    0.2492625  0.0225682 11.044879
V(2,1)    0.0424334  0.0147947 2.8681376
V(2,2)    0.2013021  0.0182819 11.011036
```

5 Contributions

Grocer proposes functions to calculate the contributions of the exogenous variables to an endogenous one, that is the part of the current evolution of the endogenous variable that can be explained by the evolutions of each variable, current *and* past. Part 5.1 exposes the theoretical background and 5.2 presents an example.

5.1 Theoretical background

Assume that an endogenous variable y_t is described by the following equation:
$A(L)y_t = B_1(L)x_{1t} + \ldots + B_k(L)x_{kt} + u_t$

With L the lag operator: $Ly_t = y_{t-1}$

Then one has:

$y_t = \frac{B_1(L)}{A(L)} x_{1t} + \ldots + \frac{B_1(L)}{A(L)} x_{1t} + \frac{1}{A(L)} u_t$

and:
$\Delta(y_t) = \frac{B_1(L)}{A(L)} \Delta(x_{1t}) + \ldots + \frac{B_1(L)}{A(L)} \Delta(x_{1t}) + \frac{1}{A(L)} \Delta(u_t)$

This formulation shows how the variation of the endogenous variable can be explained by the contribution of the exogenous variables $(x_{1t}, \ldots, x_{kt})$ and of the residuals, with contribution of variable j defined as:
$\frac{B_1(L)}{A(L)} x_{jt}$

The calculation of these contributions involves the infinite Moving Average decomposition of an ARMA model. It can be performed by standard methods. For instance, if p and q are the respective degrees of the polynoms $A(L)$ and $B_1(L)$ and $C(L)$ the corresponding infinite moving average representation, then one can calculate the coefficients of C with the corresponding algorithm:
$C(0) = B_1(0)$
for $i = 1$ to q:
$C(i) = B_1(i) - \sum_{j=1}^{min(p,i-1)} A(j) * C(i-j)$
for i = q+1 to N:
$C(i) = -\sum_{j=1}^{min(p,i-1)} A(j) * C(i-j)$

This algorithm has been written in Scilab: this is Grocer function `mainf`. Once the infinite moving average representation has been calculated, Grocer function `contrib` allows the calculation of the contribution of the corresponding variable.

However the decomposition above is not exact for 2 reasons:

- the algorithm can only provide a finite number of terms;
- the data do not go back to minus infinite; at best they can only go back to the big-bang!

In practice, you can ignore the discrepancy or distribute it one way or another on all the contributions. Grocer proposes a method, programmed in function `balance_identity`, that distributes the discrepancy proportionally to the absolute value of each contribution.

The method can be rigorously applied only to linear models. It can however be extended to any non-linear problem by a first order approximation. In Grocer, one such problem is dealt, through function `contrib_logq2gra`. This function calculates the annual contributions to the growth rate of an endogenous variable while the underlying equation is estimated on the quarterly logarithm of the variable. The corresponding first order approximation is the following one:

$$contrib_{i,A} = \sum_{k=2}^{4} w_{A-1,k} contrib_{i,A-1,k} + \sum_{k=1}^{4} w_{A,k} contrib_{i,A,k}$$

$$w_{A-1,i} = \frac{\partial \frac{YA_i}{YA_{i-1}} - 1}{\partial y_{A-1,i}} = \frac{\prod_{j=2}^{4} exp(y_{A-1,j}) \sum_{k=1}^{4} \prod_{j=1}^{i} exp(y_{A,j})}{1 + \sum_{k=1}^{4} \prod_{j=1}^{k} exp(y_{A-1,j})}$$

$$- \frac{\prod_{j=2}^{4} exp(y_{A-1,j}) \sum_{k=1}^{4} \prod_{j=1}^{i} exp(y_{A,j}) \sum_{k=i}^{4} \prod_{j=2}^{k} exp(y_{A-1,j})}{\left(1 + \sum_{k=1}^{4} \prod_{j=1}^{i} exp(y_{A-1,j})\right)^2}$$

$$w_{A,i} = \frac{\partial \frac{YA_i}{YA_{i-1}} - 1}{\partial y_{A,i}} = \frac{\prod_{j=2}^{4} exp(y_{A-1,j}) \sum_{k=i}^{4} \prod_{j=1}^{k} exp(y_{A,j})}{1 + \sum_{k=1}^{4} \prod_{j=1}^{k} exp(y_{A-1,j})}$$

with:
$y_{A,j}$ = contribution of the logarithm of variable Y in quarter j of year A
YA = level of variable A for year A
$contrib_{i,A,k}$ = contribution of variable i in quarter k of year A
$contrib_{i,A}$ = contribution of variable i in year A

5.2 An example

As an example, take again Hendry and Ericsson equation presented in section 3.1. This is a dynamic equation which makes the contributions procedure especially fruitful.

One can for instance estimate the impulse function from a one-off increase of 1 of ly. This can be done by the following instructions:

```
--> load('SCI/macros/grocer/db/bdhenderic.dat') ;bounds('1964q3','1989q2');

--> rhe=ols('delts(lm1-lp)','delts(lp)','delts(lagts(1,lm1-lp-ly))','rnet',...
--> 'lagts(1,lm1-lp-ly)','cte','noprint');

--> coef=rhe('beta');

--> mainf([1+coef(2)+coef(4) -coef(2)],[0 -coef(2)-coef(4) coef(2)],10)


 ans  =

!   0.        !
!   0.2674627 !
!   0.0213193 !
!   0.0623181 !
!   0.0493728 !
```

```
!   0.0470486 !
!   0.0430857 !
!   0.0397769 !
!   0.0366611 !
!   0.0338010 !
```

A one-off increase of ly is shown to increase lm1 by 0 the first period (you can easily check that it is in fact the case, since ly enters with a lag in the equation), 0.2674627 the second period (this is minus the sum of the coefficients of the variable 'delts(lagts(1,lm1-lp-ly))' and of the variable 'lagts(1,lm1-lp-ly)'), and so on.[7]

```
 ans  =

!   0.        !
!   0.2674627 !
!   0.0213193 !
!   0.0623181 !
!   0.0493728 !
!   0.0470486 !
!   0.0430857 !
!   0.0397769 !
!   0.0366611 !
!   0.0338010 !
```

If you want to calculate the contribution of the variable ly to the growth rate of lm1, then it can be done as follows:

```
--> ly_mainf=mainf([1+coef(2)+coef(4) -coef(2)],[0 -coef(2)-coef(4) coef(2)],200);

-->ly_contrib=contrib(delts(ly),ly_mainf)
```

And the result is:

```
1964q3 0
1964q4 -0.0002158
1965q1 0.0061336
1965q2 -0.0029902
1965q3 0.0029388
```

[7]Once again we have performed the estimation of the equation (but it can be made once and the results saved in a results 'typed list', as done here in rhe); note also that by the option 'noprint' in `ols`, we have chosen not to print the estimation results; we have recovered the coefficients in the vector coef and asked to calculate only the 10 first coefficients of the impulse function.

```
1965q4 0.0030687
1966q1 0.0040551
1966q2 0.0021385
1966q3 0.0023299
1966q4 0.0034330
1967q1 0.0016583
1967q2 0.0088530
1967q3 0.0024139
1967q4 0.0056231
1968q1 0.0029065
1968q2 0.0136596
1968q3 -0.0005435
1968q4 0.0113783
1969q1 0.0061560
1969q2 0.0021782
1969q3 0.0065130
1969q4 0.0061694
```

[...] (results for the dates 1970q1 to 1983q4 are here omitted for the sake of brevity; they are of course given by the function `contrib`)

```
1984q1 0.0050476
1984q2 0.0066269
1984q3 0.0045817
1984q4 0.0063316
1985q1 0.0083387
1985q2 0.0080658
1985q3 0.0060436
1985q4 0.0044149
1986q1 0.0066662
1986q2 0.0086946
1986q3 0.0078957
1986q4 0.0097245
1987q1 0.0080935
1987q2 0.0066234
1987q3 0.0117735
1987q4 0.0129108
1988q1 0.0096213
1988q2 0.0097046
1988q3 0.0117405
1988q4 0.0123862
1989q1 0.0100042
1989q2 0.0122195
```

One should note that the contribution of the variable is calculated to be 0 in the first period: this is because the variable ly has an impact on lm1 only after one period and because data before 1963q1 are not available in the data base. This means that the first calculated contributions will be sometimes very far from what should be their -unknown- true value. Indeed the discrepancy between lm1 and the sum of the contributions amounts to 0.0041 (around 0,4%) in 1964q1, but decreases quickly to less than 0.0001 (0.01 %) from 1975q3 on.

Lastly, from this equation, one can calculate the contributions of each variable to the *annual growth rate*. This entails the transformation from contributions of quarterly variations of logarithms into contributions in annual growth rates as explained in the previous section and is done through the function `contrib_logq2gra`. This is done as follows:

```
--> m1=exp(lm1);
--> he_lp_list=list('lp',[1+coef(2)+coef(4) ; -coef(2)],[1+coef(1) ;...
--> -1-coef(1)-coef(2)-coef(4) ; coef(2) ]);
--> he_ly_list=list('ly',[1+coef(2)+coef(4) ; -coef(2)],...
--> [0 ; -coef(2)-coef(4) ; coef(2) ]);
--> he_rnet_list=list('rnet',[1+coef(2)+coef(4) ; -coef(2)],coef(3));
--> he_resid_list=list('resid',[1+coef(2)+coef(4) ; -coef(2)],1);
--> [listcont_unbal,listcont_bal]=contrib_logq2gra('m1',he_lp_list,he_ly_list,...
--> he_rnet_list,he_resid_list);
```

The contribution of, for instance ly, can be recovered that way:

```
-->listcont_bal(2)
```

With the following result:

```
 ans  =

1965a 0.0225424
1966a 0.0213406
1967a 0.0231886
1968a 0.0296916
1969a 0.0280491
1970a 0.0269894
1971a 0.0306605
1972a 0.0308545
1973a 0.0492482
1974a 0.0335330
1975a 0.0192224
1976a 0.0213143
1977a 0.0237160
1978a 0.0269834
```

```
1979a 0.0286258
1980a 0.0214260
1981a 0.0015827
1982a 0.0112075
1983a 0.0181204
1984a 0.0251375
1985a 0.0294632
1986a 0.0312292
1987a 0.0396425
1988a 0.0468648
```

6 The list of other Grocer econometric tools

Besides the functions presented previously, Grocer contains some utilities (functions mimicking Gauss or Matlab ones, for the manipulation of strings and matrices, for the interface with Excel,...), not presented here and many standard functions, that are now listed.

6.1 Univariate econometric procedures

Besides `ols`, there are numerous functions applying econometric methods to a single equation model: instrumental variables (function `iv`), least absolute deviations regression (function `lad`), non linear least squares (function `nls`), logit (function `logit`), probit (function `probit`), tobit (function `tobit`), Cochrane-Orcutt ols (function `olsc`) and maximum likelihood ols regression for AR1 errors (function `olsar1`), ols with t-distributed errors (function `olst`), Hoerl-Kennard Ridge Regression (function `ridge`), Huber, Ramsay, Andrew or Tukey robust regression using iteratively reweighted least-squares (function `robust`), Theil-Goldberger mixed estimation (function `theil`).

A function performing estimation of a linear model with Garch(p,q) residuals is also available (`garch`).

6.2 The Kalman filter

A function performing a Kalman filter estimation (function `kalman`) is available. An application of the Kalman filter is also explicitly programmed, the time-varying parameters (function `tvp`).

6.3 Multicollinearity tests and outlier diagnostic

Grocer provides Belsey, Kuh and Welsch (1980) standard multicollineraity tests, through the function `bkw`. Outlier diagnostic tests (dfbeta, dffits, studentized residuals, hat-matrix diagonals) are implemented by the function `dfbeta`.

6.4 Unit roots and cointegration

Grocer provides four among the most popular unit root tests: the (augmented) Dickey-Fuller test (function `adf`); the Phillips-Perron test (function `phil_perr`); the Schmidt-Phillips test (function `schmiphi`) and the Kwiatkowski, Phillips, Schmidt, Shin stationarity test (function `kpss`).

Two cointegration tests are currently available: the corrected-Augmented Dickey Fuller test (with its Phillips-Perron variant) and the Johansen cointegration test. They are performed by the functions `cadf` and `johansen`.

6.5 Filters

Grocer provides three usual filters : the Hodrick-Prescott filter (`hpfilter`); the Baxter-King one (`bkfilter`) and the more recent Christiano-Fitzgerald one (`cffilter`).

7 Conclusion

This paper has proposed an insight into Grocer functions. Once again the interested reader can have a more precise view of Grocer with a look at the user manual, the on-line help embodied in Grocer package and even better by using it!

Grocer is due to evolve regularly and some new features are already underway. A few procedures existing in some commercial packages and some more rare but nevertheless useful methods (notably in the bayesian field) remain to be implemented. The matrix-oriented nature of Scilab will make it easy to stay at the forefront of the econometric science. And the great similarity between Scilab and Matlab or Gauss will help as well.

Acknowledgements

I wish to thank the Scilab team for their precious advices, James Le Sage for having provided the basis of so many grocer functions and Emmanuel Michaux for his faithful use and testing of Grocer. The usual disclaimer applies.

References

Dubois É. (2004a): "Grocer 1.0: An Econometric Toolbox For Scilab", user manual, available at http://dubois.ensae.net/grocer.html.

Dubois É. (2004b): "Grocer 1.0, an Econometric Toolbox For Scilab: a Scilab point of view", presented at the first Scilab International conference, 2&3 of november.

Greene W. (2003): *Econmetric Analysis*, fifth edition, Prentice Hall, New Jersey.

Hendry D.F and N.R Ericsson (1991): "Modelling the demand for narrow money in the United Kingdom and the United States"', *European Economic Review*, p833-886.

Hendry D.F. and H-M Krozlig (1999): "Improving on 'data mining reconsidered' by K.D. Hoover and S.J. Perez ", *Econometrics Journal*, n° 2, pp. 41-58.

Hendry D.F. and H-M. Krozlig (2001): "Computer Automation of General-to-Specific Model Selection Procedures", *Journal of Economic Dynamics and Control*, 25(6-7), pp. 831-866.

Hoover K.D. and S.J. Perez (1999): "Data mining reconsidered: a general to specific approach to specification search", *Econometrics Journal*, n°2, pp. 167-191.

Jenkins G.M. and A.S. Alavi (1981): "Some aspects of Modelling and Forecasting Multivariate Time Series", *Journal of Time Series Analysis*, vol. 2, n°1, pp. 1-47.

LeSage J. (1999): Applied Econometrics using MATLAB, University of Toledo, October, available at http://www.spatial-econometrics.com/.

Mrkaic M. (2001): "Scilab as an Econometric Programming System", *Journal of Applied Econometrics*, vol. 16, n°4, July-August, pp.553-559.

Scilab (1993): "Introduction to Scilab", INRIA, available at http://scilabsoft.inria.fr/product/index_product.php?page=

Terceiro J., J.M. Casals, M. Jerez, G.R. Serrano and S. Sotoca (2000): "Time Series Analysis using MATLAB, Including a complete MATLAB Toolbox", available at http://www.ucm.es/info/icae/e4/e4download.htm.