Automatic identification of simultaneous equations models

Omtzigt, P.H.

UvA ECONOMETRICS

Discussion Paper: 2002/14

Automatic identification of simultaneous equations models

Pieter Omtzigt

www.fee.uva.nl/ke/UvA-Econometrics

Department of Quantitative Economics
Faculty of Economics and Econometrics
Universiteit van Amsterdam
Roetersstraat 11
1018 WB AMSTERDAM
The Netherlands

UvA UNIVERSITEIT VAN AMSTERDAM

Automatic identification of simultaneous equations models

Pieter Omtzigt *

January 2, 2003

Abstract

This paper considers within-equation restrictions in simultaneous equation models. It provides an algorithm, which renders them generically identifying. This algorithm works directly on the the restrictions and renders estimation by means of methods that require identification possible. Using this method it is possible to calculate the right number of degrees of freedom analytically and to provide standard errors on all the estimated parameters.

1 Introduction

Consider the simultaneous equations model:

$$\begin{aligned} \beta' z_t = A' y_t + B' x_t = u_t,\ t = 1, \ldots, T \\ u_t \backsim iid N(0, \Omega), \end{aligned} \tag{1}$$

where y_t is a vector of length r with endogenous variables, x_t a vector of length q with predetermined variables and $\beta = [A', B']'$ a $p \times r$ matrix of coefficients $(p = r + q)$. Assume that A and therefore β is of full rank and that x_t and u_t are independent.

Likelihood inference on (β, Ω) is possible, but it is readily verified that a parameter point (β^1, Ω^1) is not uniquely identified (which means that there is at least one other parameter point (β^2, Ω^2), with whom it shares the same probability measure). For any non-singular matrix C, $(\beta^1 C, C\Omega^1 C')$ has an identical probability measure. To uniquely identify a space we need to put restrictions on the parameter space. In this article we shall consider only within-equation restrictions on β without putting restrictions on Ω. More precisely, we consider

$$\beta = [H_1 \varphi_1, \ldots, H_r \varphi_r] \tag{2}$$

*Universiteit van Amsterdam, Roetersstraat 11, 1018WB Amsterdam. Email: P.H.Omtzigt@uva.nl

where H_i are $p \times s_i$ matrices of full column rank. Defining $R_i = (H_i)_\perp$ an equivalent expression of these restrictions is given by:

$$R_i' \beta_i = 0 \text{ for } i = 1, \ldots, r \tag{3}$$

We repeat the well known fact that the likelihood of the model is invariant under premultiplication by a full rank matrix S.

The Wald condition states a necessary and sufficient condition for identification:

Theorem 1 *The parameter value (β, Ω) is uniquely identified (up to a normalization of one of the elements in each vector of β) if and only if for any $i = 1, \ldots r$*

$$rank(R_i' \beta) = r - 1 \tag{4}$$

There are two problems in practice with this theorem: it depends on the a priori unknown parameters β and it does not give an indication as how to identify a model if (4) fails. The first problem was tackled by Johansen (1995) , who proved the following theorem:

Theorem 2 *If the only restrictions imposed on the parameters are (3) a set of necessary and sufficient conditions for the parameter value β and hence Ω to be uniquely identified (up to a normalization of one of the elements in each vector β) is:*

$$rank\left(R_j' \left[H_{k_1}, ..., H_{k_n}\right]\right) \geqq n \tag{5}$$

for $n = 1, \ldots r - 1$,
for all $j \in \{1, \ldots, r\}$,
and for every set $\{k_1, \ldots, k_n\}$ not containing j,

This theorem gives conditions that only depend on the restrictions, not on the parameters. If all the conditions (5) are satisfied, then there are $\sum_{i=1}^{r}(p - r - s_i + 1)$ degrees of freedom for testing the hypothesis (2).

However if one of the rank conditions (5) fails, serious problems arise not just in the interpretation of potential estimates, but in the maximization of the likelihood function and testing process itself. To my knowledge no analytical method exists to determine the number of restrictions imposed by (3) on the model.

This paper provides a simple algorithm to determine identifying restrictions, when (5) fails. There are four main applications of this algorithm.

1. A device for counting the number of restrictions in a particular model (if the restrictions are not identifying).

2. An instrument to be used for estimation algorithms, which require identification. For instance the algorithms of Johansen and Juselius (1994) and Johansen (1995) require identification. Without identification, they seem to work more than 99% of the time: for automated model selection, this is however not sufficient. Other algorithms for estimation based on general optimization methods, do not require identification (Doornik, 1995) .

3. Only in identified models can (asymptotic) standard errors be given for all estimated parameters. The algorithm finds an identification scheme, which is not necessarily unique. One can thus employ the algorithm to find standard errors of the estimated parameters. If there are multiple identification schemes, we can scan them all (usually there are only a few) and use the standard errors of all schemes to decide where to put additional restrictions.

4. In the cointegrated VAR model Davidson (1998) provides an algorithm to find all possible restricted cointegration vectors (using Wald testing). However he only considers one restricted vector at the time, but not a combination of them. Using the results in this paper and likelihood ratio tests, Omtzigt (2001) tests not only one restricted vector at the time, but also all possible combinations of them. The switching algorithm never breaks down, once the model has been generically identified and the automated model selection procedure results in one preferred restricted model only (with possibly equivalent formulations).

For further discussions on the (restricted) simultaneous equations models, we refer to Koopmans et al. (1950) , Fisher (1966) , Hsiao (1983) and Sargan (1988) and references therein. For potential applications to the I(1) model we refer to Johansen (1995) and for the I(2) model to Johansen (2000) .

The outline of the paper is as follows. Section 2 contains the main theoretical result. In section 3 the algorithm is presented and illustrated by means of an example. Section 4 provides an empirical illustration of its use in cointegrated VAR models. and section 5 concludes. The appendices contain all proofs and a Matlab program implementing the algorithm.

2 Results

In this paper we shall refer to (5) as rank conditions of order n. They can be given the following logical ordering (from order 1 to order r):

$$rank\left(R_j' H_{k_1}\right) \geqq 1, \quad j \neq k_1 \tag{6}$$

$$rank\left(R_j' \left[H_{k_1}, H_{k_2}\right]\right) \geqq 2, \quad j \neq k_1 \neq k_2 \tag{7}$$

$$\vdots$$

$$rank\left(R_j' \left[H_{k_1}, H_{k_2}, \ldots, H_{k_{m-1}}\right]\right) \geqq m-1, \quad j \neq k_1 \neq \ldots \neq k_{m-1} \tag{8}$$

$$\vdots$$

$$rank\left(R_j' \left[H_{k_1}, \ldots, H_{k_{r-1}}\right]\right) \geqq r-1, \quad j \neq k_1 \neq \ldots \neq k_{r-1} \tag{9}$$

There are r rank conditions of order 1, $r\left(r-1\right)/2$ rank conditions of order two and r rank conditions of order r. In case the rank conditions do not hold, many different ones may fail at the same time. Let m be the lowest order for which at least one rank condition

breaks down:

$$rank\left(R_j'\left[H_{k_1},..,H_{k_m}\right]\right)=m-1,\quad j\neq k_1\neq\ldots\neq k_m \tag{10}$$

The rank deficiency in (10) must be exactly one as all the lower rank conditions (6)-(8) hold and in particular

$$rank\left(R_j'\left[H_{k_1},..,H_{k_{m-1}}\right]\right)=m-1,\quad j\neq k_1\neq\ldots\neq k_{m-1}$$

Let the columns of H_j be $h_{j1},...,h_{js_j}$ and let $H_{j,-i}=[h_{11},\ldots h_{1i-1},h_{1i+1},\ldots h_{1s}]$, that is H_j without column H_{ji}. Furthermore let $k_{ji}=h_{ij}-H_{j,-i}\left(H_{j,-i}'H_{j,-i}\right)^{-1}H_{j,-i}'h_{ij}$.

The following theorem shows that we can always 'repair' this rank conditions by deleting one column from matrix H_j and adjusting R_j accordingly. Not any column can be deleted, but at least one of the columns repairs the rank condition.

Theorem 3 *If (6)-(8) hold and (10) then for at least one of the columns h_{ji} of H_j,*

$$rank\left(\left[R_j,k_{ji}\right]'\left[H_{k_1},..,H_{k_m}\right]\right)=m. \tag{11}$$

Without loss of generality, we shall assume that a condition involving R_1 is the first one for which the rank condition fails to hold and that h_{1d} is the column in Theorem 3.

The next theorem shows that we can rotate the columns of any matrix β which is restricted as in (2) to find a matrix β^* which obeys all the previous restrictions implied by (2) and the new restriction, caused by shifting h_{1d}' from H_1 to R_1.

Theorem 4 *If (6)-(8) hold and (10) and h_{1d}' satisfies (11) then for any $\beta=[H_1\varphi_1,...,H_r\varphi_r]$ there exists almost surely $\beta^*=[H_{1,-d}\varphi_1^*,H_2\varphi_2,...,H_r\varphi_r]$ such that $sp(\beta)=sp(\beta^*)$.*

The result has been split into two parts on purpose: theorem 3 only involves the restrictions, whereas theorem 4 shows that whatever the parameter value, the additional restriction can be satisfied. This means that we are only putting an extra identifying constraint on the model and do not put additional binding restrictions on it.

The idea of the proof is that if the rank condition of order m fails (and all the lower ones hold), then we can find exactly one linear combination of $(\beta_{k_1},..,\beta_{k_m})$, say γ which lies in the space of β_j. Let $\beta_j=H_j\varphi_j$ and $\gamma=H_j\psi$ To distinguish β_j from γ we put one additional restriction on the β_j.

3 Algorithm

Together these last two theorems give rise to an operational algorithm to identify the space, given by any set of restrictions. Each time the rank condition is not satisfied by $(H_1,...,H_r)$ we are able to take away a column of one of the $H's$ without imposing further restrictions. We repeat the operation until we have identifying restrictions (the algorithm is guaranteed to end as the number of columns of the matrices H is finite).

Formally we propose the following algorithm:

Algorithm 5

1. *Check the rank conditions (6)-(9), for identification, starting with the lowest one, (6).*
2. *If all rank conditions are satisfied, go to 4.*
3. *When the first rank condition is broken, as in (10), find a column h_{ij} such that (11) is satisfied. Cancel this column from H_i and then go to 1.*
4. *The space is generically identified*

An implementation of this algorithm in Matlab is available in the appendix of the paper. Note the loop structure in which all the rank conditions are checked, starting from the lowest one. If a rank condition does not hold, we see which of the columns h_{ij} we can eliminate from H_i to satisfy it. The checking of all the rank conditions then starts again.

3.1 An example

A detailed example of how the algorithm works in practice clarifies the exact functioning of the algorithm. Among other things it shows that if a rank condition of order m is repaired at step t, then at time $t+1$ it may be necessary to repair a lower order rank condition. It is thus absolutely vital that all conditions are checked in each round. The example is also just simple enough to be done by hand, but a computer will just do it quite a bit faster.

Consider the following matrix β with 5 rows and 3 columns, on which we impose within-equation restrictions (2) by means of the following matrices H_i: (Note that of each of the three matrices H_f the columns are mutually orthogonal, such that $h_{fi} = k_{fi}$.)

$$H_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, H_2 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, H_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{12}$$

$$H_1 = [h_{11}, h_{12}, h_{13}], H_2 = [h_{21}, h_{22}, h_{23}], H_3 = [h_{31}, h_{32}, h_{33}]$$

As bases of orthogonal complements to these matrices we choose:

$$R_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ -1 & 0 \end{bmatrix}, R_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, R_3 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ -1 & 0 \end{bmatrix}$$

$$R_1 = [r_{11}, r_{12}], R_2 = [r_{21}, r_{22}], R_3 = [r_{31}, r_{32}]$$

The algorithm now runs as follows:

3.1.1 First round

Check the first-order rank conditions

$$rank(R_1' H_2) = rank \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = 1$$

$$rank(R_1' H_3) = rank \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = 1$$

$$rank(R_2' H_1) = rank \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = 1$$

$$rank(R_2' H_3) = rank \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} = 1$$

$$rank(R_3' H_1) = rank \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = 1$$

$$rank(R_3' H_2) = rank \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = 1$$

Check the second-order rank conditions

As all first-order rank conditions are satisfied, we check the second-order rank conditions:

$$rank(R_1' [H_2, H_3]) = rank \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = 1$$

This rank condition fails, which means that we must apply step 3 of the algorithm.

Find a column of H_1 that satisfies (11)

We add one of the columns of H_1 to R_1 and see whether this particular rank condition is repaired. Try $H_1^* = [h_{12}, h_{13}]$ and $R_1^* = [r_{11}, r_{12}, h_{11}]$. The rank condition becomes:

$$rank(R_1^{*\prime} [H_2, H_3]) = rank \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \end{pmatrix} = 2$$

The rank condition is now satisfied and we take $H_1 = H_1^*$ and $R_1^* = R_1$ (leaving the other matrices as they were before) and start the algorithm at point 1:

3.1.2 Second round

Check the first order rank conditions

$$rank(R_1' H_2) = rank \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = 1$$

$$rank(R_1' H_3) = rank \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 0 \end{pmatrix} = 2$$

$$rank(R_2' H_1) = rank \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = 0$$

This rank condition fails.

Find a column of H_2 that satisfies (11)

When we move the first column of H_2 to R_2 we obtain the following candidates $H_2^* = [h_{22}, h_{23}]$ and $R_1^* = [r_{21}, r_{22}, h_{21}]$. The rank condition then reads:

$$rank(R_2^{*\prime} H_1) = rank \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} = 0$$

It is still not satisfied, so we try shifting the second column of H_2: $H_2^* = [h_{21}, h_{23}]$ and $R_2^* = [r_{21}, r_{22}, h_{22}]$. This results in the following rank condition:

$$rank(R_2^{*\prime} H_1) = rank \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} = 1$$

The rank condition now holds and we take $H_2 = H_2^*$ and $R_2 = R_2^*$ to go back to step 1 of the algorithm:

3.1.3 Third round

Check the first-order rank conditions

It is easily verified that of all the first-order rank conditions are satisfied, with the exception of

$$rank(R_3' H_2) = rank \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} = 0$$

Find a column of H_3 that satisfies (11)

Shifting the first column of H_3 to R_3 would clearly not work, as that would imply $sp(H_2) = sp(H_3)$. (In this case even $H_2 = H_3$). We therefore shift the second column of H_3 : $H_3^* = [h_{31}, h_{33}]$ and $R_3^* = [r_{31}, r_{32}, h_{32}]$. The rank condition is now satisfied:

$$rank(R_3^{*\prime} H_2) = rank \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} = 1$$

For the next round take $H_3 = H_3^*$ and $R_3 = R_3^*$.

3.1.4 Fourth round

Check the first and second-order rank conditions

All 6 first order and 3 second order rank conditions are satisfied, such that we conclude that the restrictions identify the model: The conditions of Theorem 3 in Johansen (1995) now hold for this example

For completeness we shall also give the matrices S from Theorem 4. If we have the

matrices (12), then we can write the matrix β as:

$$\beta = \begin{bmatrix} \varphi_{11} & 0 & \varphi_{31} \\ 0 & \varphi_{21} & \varphi_{32} \\ \varphi_{12} & \varphi_{22} & 0 \\ \varphi_{13} & \varphi_{23} & \varphi_{33} \\ \varphi_{11} & 0 & \varphi_{31} \end{bmatrix}$$

The combination $\varphi_{32}\beta_2 - \varphi_{21}\beta_3 \equiv \gamma \in sp(H_1)$. Post-multiplying β by the full-rank matrix

$$S_1 = \begin{bmatrix} 1 & 0 & 0 \\ \frac{\varphi_{32}}{\varphi_{31}} & 1 & 0 \\ -\frac{\varphi_{11}}{\varphi_{31}} & 0 & 1 \end{bmatrix}$$

gives way to

$$\beta^* = \begin{bmatrix} 0 & 0 & \varphi_{31} \\ 0 & \varphi_{21} & \varphi_{32} \\ \varphi_{12}^* & \varphi_{22} & 0 \\ \varphi_{13}^* & \varphi_{32} & \varphi_{33} \\ 0 & 0 & \varphi_{31} \end{bmatrix} \tag{13}$$

which satisfies the restrictions after the first round of the algorithm. Note that this transformation is not defined if $\varphi_{22} = 0$ or $\varphi_{31} = 0$.

Taking away the stars in the last expression, we can post-multiply again by

$$S_2 = \begin{bmatrix} 1 & \frac{\varphi_{22}}{\varphi_{12}} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

to obtain a matrix, satisfying the restrictions at the end of the second round. This step inserts a zero in place of φ_{22} in (13). In the last step, the matrix S_3 is given by:

$$S_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \frac{\varphi_{32}}{\varphi_{21}} \\ 0 & 0 & 1 \end{bmatrix}.$$

Post-multiplication leads to the following general matrix:

$$\beta = \begin{bmatrix} 0 & 0 & \varphi_{31} \\ 0 & \varphi_{21} & 0 \\ \varphi_{12} & 0 & 0 \\ \varphi_{13} & \varphi_{32} & \varphi_{33} \\ 0 & 0 & \varphi_{31} \end{bmatrix} \tag{14}$$

which satisfies all the rank conditions and is therefore generically identified.

3.2 Discussion

Making a change in a broken rank condition can cause a previously satisfied rank condition to fail. In the example above, all rank conditions of first order are satisfied in the first round, but the change made causes first-order rank conditions to fail subsequently. This demonstrates that in every round we have to start checking the lowest order rank conditions.

In the second round, we note that not any column can be eliminated from H, but we can still choose between deleting the second and the third column. This implies that the restrictions imposed by the algorithm are in general not unique. We thus find but only one of many ways to identify this space. It may be hard to attach an economic meaning to a particular identification in any one application. In some way this is the only weak point of the algorithm: in automatic search algorithms and other applications, the researcher may look for an different identification scheme to make economic sense of it. This however can easily be achieved by making available all equivalent identification schemes.

4 An application

As a practical application we consider the p-dimensional cointegrated VAR-model with k lags:

$$\Delta X_t = \alpha\beta' \begin{pmatrix} X_{t-1} \\ t \end{pmatrix} + \sum_{i=1}^{k-1} \Gamma_i \Delta X_{t-i} + \Psi d_t + \mu + \varepsilon_t \tag{15}$$

where β' are the cointegration vectors, t is a time trend and d_t are dummy variables.

We note that the likelihood function depends on $\Pi = \alpha\beta'$. This means that we can take $\alpha^* = \kappa\alpha$ and $\beta^* = \kappa^{-1}\beta$, where κ is any invertible matrix. The likelihood is unchanged after this transformation as $\Pi^* = \Pi$. We thus have exactly the same identification problem for β in this model as in the simultaneous equations model (1).

For tests of the kind (2), a number of computer packages have implemented the switching algorithm of Johansen and Juselius (1994) , henceforth JJ. CATS[1] by Hansen and Juselius (1994) performed better than PcFiml version 9.3[2] by Doornik and Hendry (1997) and my own implementation of the switching algorithm (which does not put identifying restrictions): The maximum in the likelihood function CATS found was the highest in all the examples considered below.

We compare the CATS implementation of the switching algorithm of JJ (which does not put identifying restrictions on the cointegration space) and the implementation of a Matlab program, which executes the switching algorithm after having imposed identifying restrictions by means of algorithm 5. We consider an Australian data set, first analyzed by JJ and also used by Doornik (1995) to illustrate his alternative numerical method. It consists of the log of nominal money (m), the log of real national income (y), the log of

[1] We use version 1 of this CATS in RATS 4.3

[2] Version 10.1 of PcGive does not give the user the option to use beta switching. We thus used the latest version of the program that did.

Hypothesis tested							CATS		New Algorithm	
	m	y	p	i3	i10	t	LR-test	# iterations	LR-test	# iterations
$\mathcal{H}_1$:	1	-1	-1	0	0	*	11.30	36	11.28	49
$\mathcal{H}_2 : \mathcal{H}_1 +$	0	0	0	*	*	*	12.63	93	12.61	80
$\mathcal{H}_3 : \mathcal{H}_1 +$	0	0	0	1	-1	*	15.88	200	13.44	53
$\mathcal{H}_4 : \mathcal{H}_1 +$	0	0	0	1	-1	0	16.62	2	16.10	158
$\mathcal{H}_5 : \mathcal{H}_1 +$	0	0	0	1	*	0	15.20	195	15.15	87
$\mathcal{H}_6 : \mathcal{H}_1 +$	0	0	0	0	1	*	17.28	200	17.13	200
$\mathcal{H}_7 : \mathcal{H}_1 +$	0	0	0	1	0	*	16.37	197	16.32	152
$\mathcal{H}_8 : \mathcal{H}_1 +$	1	0	*	0	*	0	11.40	89	11.62	200

Table 1: Comparison between optimization in CATS (without identification) and optimization with identification

the GDP deflator (p), a three month interest rate (i3) and the 10 year government bond rate (i10). d_t contains centered seasonal dummies and a dummy which takes value 0 until 1982, 2nd quarter and 1 afterwards.

JJ fit a VAR with 2 lags for the period 1976-1 until 1991-1 (61 effective observations). The trace and rank test point to a rank of at most one, but JJ choose three as their preferred rank.They test a a number of hypotheses, which we have tested in PcGive 9.3, PcFiml 10.1, CATS and our own program. All of them give identical answers. We then considered testing a number of hypothesis, where restrictions were put on only one or two of the cointegration vectors. By definition these restrictions are not identifying. In the table below we report the results of testing that the velocity of money (m-y-p) is trend stationary on its own and in combination with all possible cointegration relations between the interest rates and an unrelated hypothesis $\mathcal{H}_8$. Both algorithms have the same convergence criterium and number of maximum iterations (200).

For hypothesis $\mathcal{H}_1 - \mathcal{H}_7$ the new algorithm in Matlab does remarkably better. It needs less iterations and finds a higher maximum in the likelihood, resulting in a lower Likelihood Ratio test statistic. Unlike CATS it also reports the degrees of freedom of the likelihood ratio test. Just to show that there is no mathematical guarantee, we also report $\mathcal{H}_8$, where CATS does better than the new method.[3] We have checked these LR-tests against all other methods implemented in PcGive 9.3, namely linear switching (Boswijk, 1995) and the Broyden-Fletcher-Goldfarb-Shanno method (Doornik, 1995) . Neither of them found better maxima. (and the first one did notably worse in cases $\mathcal{H}_6 - \mathcal{H}_7$). Both these methods have the advantage that they are able to cope with more general restrictions on both α and β.

A partial explanation for the result is the difficulty of the the optimization problem at hand. We impose three cointegration relationship, where the evidence of the second and third ones are weak, such that those relationships are hard to find in the data. Prices and

[3]If the maximum number of iterations is lifted, it does find a maximum after 725 iterations for an LR-test statistic of 11.35

money are often modelled as I(2) and this would probably be better in the current data set as well. And the hypotheses tested are all soundly rejected at the 5% level by any method. The likelihood in the region we are searching is extremely flat. Yet this is the ideal situation to put algorithms to the test. In easy situations all of them find the same maximum, which in all likelihood is the global one.

5 Conclusions

We have presented a way of identifying an under-identified parameter space in simultaneous equations models and hence rendered estimation by the means of the switching methods of Johansen (1995) possible. In over 10^6 test executed so far in simulations Omtzigt (2002) , the method has not broken down once, such that it is ideal in automated model selection. It is reasonably fast, calculates the degree of freedoms for the likelihood ratio test automatically and analytically (no separate procedure is needed) and allows for calculating standard errors on all the estimated parameters. The procedure is very easy to implement and a Matlab version is attached to this paper.

References

Boswijk, H. (1995). Identifiability of cointegrated systems. Technical report, Department of Actuarial Sciences and Econometrics, University of Amsterdam.

Davidson, J. (1998). Structural relations, cointegration and identification: Some simple results and their applications. *Journal of Econometrics 87*, 87–113.

Doornik, J. (1995). Testing general restrictions on the cointegration space. Technical report, Nuffield College, Oxford.

Doornik, J. and D. Hendry (1997). *Modelling Dynamic Systems Using PcFiml 9 for Windows*. Timberlake, London.

Fisher, F. (1966). *The Identification Problem in Econometrics*. New York: McGraw-Hill.

Hansen, H. and K. Juselius (1994). *CATS for RATS4: Manual to Coinegration Analysis to Time Series*. Estima.

Hsiao, C. (1983). Identification. In Z. Grilliches and M. Intriligator (Eds.), *Handbook of Econometrics*, Amsterdam. North Holland.

Johansen, S. (1995). Identifying restrictions of linear equations: With applications to simultaneous equation and cointegration. *Journal of Econometrics 69*, 111–132.

Johansen, S. (2000). Testing hypotheses in the I(2) model. Technical report, Economics Department, European University Institute, Florence.

Johansen, S. and K. Juselius (1994). Identification of the long-run and short-run structure: An application to the IS-LM model. *Journal of Econometrics 63*, 7–36.

Koopmans, T., H. Rubin, and R. Leipnik (1950). *Statistical Inference in Dynamic Economic Models*. New York: Cowles Commission Monograph 10, J.Wiley.

Omtzigt, P. (2001). Selezione automatica dei modelli autoregressivi cointegrati. In P. Giudice, C. Tarantola, and F. Verrecchia (Eds.), *Modelli Statistici per le Applicazioni di Data Mining*, pp. 109–114. Università di Pavia.

Omtzigt, P. (2002). Automatic identification of simultaneous equations models. Technical Report 2002/01, University of Insubria, Varese, Italy.

Sargan, J. (1988). *Lecture Notes on Advanced Econometric Theory*. Oxford: Basil Blackwell.

Appendix: Proofs

The following lemma is needed for the proof of Theorem 3:

Lemma 6 *If the rank conditions (6)-(8) hold*

$$rank\left(\left[H_{k_1},\ldots,H_{k_j}\right]\right) \geqq j, j = 1,\ldots,m-1 \tag{16}$$

Proof. For $j = 1,\ldots,m-1$ the result follows directly from (6)-(8): for instance (8) implies that

$$m-1 \leq rank\left(R_j'\left[H_{k_1},H_{k_2},\ldots,H_{k_{m-1}}\right]\right) \leq rank(H_1,\ldots,H_{m-1})$$

such that

$$m-1 \leq rank(H_1,\ldots,H_{m-1})$$

For $j = m$ let us assume that the lemma does not hold, i.e. that $rank(H_1,\ldots,H_m) \leq m-1$. We find

$$m-1 \leq rank(H_1,\ldots,H_{m-1}) \leq rank(H_1,\ldots,H_m) \leq m-1 \tag{17}$$

such that equality holds throughout and $rank(H_1,\ldots,H_m) = m-1$ This leads to the existence of $h_1,\ldots,h_{m-1} \in sp(H_1,\ldots,H_m)$ so that $H_i = (h_1,\ldots,h_{m-1})M_i$ From (17) we see that

$$m-1 = rank(H_1,\ldots,H_{m-1}) = rank(h_1,\ldots,h_{m-1})$$

such that $(H_1,\ldots,H_{m-1})_\perp = sp(h_1,\ldots,h_{m-1})_\perp$, and hence $(H_1,\ldots,H_{m-1})_\perp' H_m = 0$, which contradicts (8), since H_m is a non-null matrix. ■

Proof of theorem 3. by lemma 6 we know that

$$rank\left([H_{k_1},..,H_{k_m}]\right) \geqq m \tag{18}$$

$$rank\left(\left[R_j, H_j\right]' \left[H_{k_1}, .., H_{k_m}\right]\right) \geqq m$$

as (R_j, H_j) is a matrix of full rank. As H_j is of full column rank, $[k_{j1}, \ldots, k_{ji}, R_j]$ is a square, full rank matrix, which together with (18) implies that

$$rank\left(\left[k_{j1}, \ldots, k_{ji}, R_j\right]' \left(H_{k_1}, ..H_{k_m}\right)\right) \geqq m$$

This combined with (10) means that

$$rank\left(\left[R_j, k_{ji}\right]' \left[H_{k_1}, .., H_{k_m}\right]\right) = m$$

for at least one column of H_j. ■

We note that

$$rank\left(k_{1d}' \left[H_{k_1}, ..H_{k_m}\right]\right) = 1 \tag{19}$$

Proof of theorem 4. $rank(R_1' \left[\beta_{k_1}, ..., \beta_{k_m}\right]) = m - 1$, (10), implies that there exists an $m \times 1$ vector $a_\perp$, such that $R_1' \left[\beta_{k_1}, ..., \beta_{k_m}\right] a_\perp = 0$. Without loss of generality we assume that $k_1 = 2, \ldots, k_m = m + 1$.

Therefore $\left[\beta_2, ..., \beta_{m+1}\right] a_\perp \equiv \gamma \in sp(H_1)$
As $rank([R_1, h_{1d}]' \left[\beta_2, ..., \beta_{m+1}\right]) = m$, $h_{1d}'\gamma \neq 0$. This implies that we can take $\beta_1^* = \beta_1 - \gamma \left(h_{1d}'\gamma\right)^{-1} h_{1d}'\beta_1$. This transformation is of the kind $\beta^* = \beta S$, where

$$S = \begin{bmatrix} 1 & 0 & 0 \\ -a_\perp \left(h_{1d}'\gamma\right)^{-1} h_{1d}'\beta_1 & I_m & 0 \\ 0 & 0 & I_{r-m-1} \end{bmatrix}.$$

This matrix does not exist if $\left(h_{1d}'\gamma\right) = 0$, but this only happens on a set of Lebesgue measure zero. When it exists it is clearly of full rank, which means that $sp\left(\beta\right) = sp\left(\beta^*\right)$ ■

Appendix: Matlab program

```
function [Hblockout,Rblock] = identify(Hblock)


% For a given set of linear restrictions of the kind
% beta =[Hblclock{1}*phi1,...,Hblock{r}*phir]
% (without normalizations), this function provides an equivalent
% identifying set of restrictions Hblockout


r = size(Hblock,2);
p =size(Hblock{1},1);
% Get the orthogonal complements (see equation (3))
for f=1:r
 Rblock{f}= null(Hblock{f}');
end
```

```
% The main loop of the program
identification = 0;
% As long as there is no identification run the following loop
while identification == 0
 [Hblock,Rblock,identification] = mainloop(Hblock,Rblock,r);
end



%*****************************************
% Internal function:
%*****************************************
function [Hblock,Rblock,identification] = mainloop(Hblock,Rblock,r)
identification = 1;
% Set identification flag to one.  If one the rank conditions fails
% we repair it and set it to zero (no identification)
% Start with rank condition of order 1 (for which k=2)
 M = nchoosek(1:r,k);
 % one of the indices,j, on the left (R) others (in C) on the right
(H's)
 for j=1:size(M,1)
  for m=1:k
   C = setdiff(M(j,:),M(j,m));
   right = zeros(size(Hblock{1},1),0);
   for m2=1:k-1
   right = [right,Hblock{C(m2)}];
  end
  % Check whether rank condition is satisfied.
  if rank(Rblock{M(j,m)}'*right, 0.00001)< k-1
   % if not, check which column of H can be shifted
   sizeH = size(Hblock{M(j,m)},2);
   H = Hblock{M(j,m)};
   for s2 = 1:sizeH
   H(:,1:s2-1);
   H(:,s2+1:sizeH);
   testblockH = [H(:,1:s2-1),H(:,s2+1:sizeH)];
   testblockR = null(testblockH');
   if rank(testblockR'*right, 0.00001) == k-1
    %this column can be shifted!
    Hblock{M(j,m)}=testblockH;
    Rblock{M(j,m)}=testblockR;
    identification = 0; % no identification
   %model has been changed, such that there is
   %no guarantee all rank conditions are satisfied
    break,end
   end
  end
 end
 end
end
```