

Symbsolve 1.0

Giovanni Lombardo

ECB

Frankfurt am Main

E-mail: giovanni.lombardo@ecb.int

28 December 2005

1 Introduction

This note describes the working and use of Symsolve: a MATLAB based code for the approximation and solution of forward-looking discrete-time dynamic models. This software is meant to be complementary to Dynare as it allows the symbolic extraction of second-order approximations of dynamic systems written in Dynare notation (extension `.mod`). The solution technique adopted for the second-order expanded system does also differ from the techniques currently available in Dynare, as Symsolve is based on the technique described in Lombardo and Sutherland (2005).

2 How to use Symsolve

Symsolve is a set of MATLAB function that reads a Dynare-compatible `.mod` file and constructs the symbolic first and second order canonical representation of a rational expectation model, as described in Lombardo and Sutherland (2005).

This symbolic second-order approximation is then used for numerical computation of first- and second-order state-space representation of the solution of the dynamic system.

For the first order solution a standard (real) QZ decomposition is used following Klein (2000) (the file `firstorder_stuff.m` does this). A state-space representation of the solution is returned in the form (where $s_{f,t}$ and $c_{f,t}$ stand for first-order accurate state variables and control variables respectively)

$$s_{f,t} = \Phi s_{f,t-1} + \varepsilon_t \quad (1)$$

$$c_{f,t} = \Pi s_{f,t} \quad (2)$$

IR functions are plotted if requested.

For the second order solution the method described in Lombardo and Sutherland (2005) is used (this is done in `secorder_stuff.m`).¹ It returns the state-space representation of the second-order solution as described below. See Lombardo and Sutherland (2005) for details.²

¹Notice that the QZ decomposition is computed only once as the second-order solution uses the same QZ-information used in the first order solution.

²Ask me for a set of `.mod` examples including the one used in Lombardo-Sutherland.

2.1 Canonical form and state-space representation

The reduced-form canonical representation of the second order system is

$$AA Y_{t+1} = BB Y_t + Ax \Lambda_t + C z_t \quad (3)$$

$$z_t = N z_t - 1 + \varepsilon_t \quad (4)$$

$$\Lambda_t \equiv \text{vech}(Y_t Y_t') \quad (5)$$

The state-space solution is

$$\tilde{Y}_t = \Pi s_t + \Pi_2 \vec{V}_t + C_{all} \vec{\Sigma} \quad (6)$$

$$s_t = \Phi s_{t-1} + \Phi_2 \vec{V}_t + M \varepsilon_t \quad (7)$$

$$\vec{V}_t \equiv \text{vech}(s_{f,t} s_{f,t}') = \hat{\Phi} \vec{V}_{t-1} + \Psi \vec{\xi}_t + M h \text{vech}(\varepsilon_t \varepsilon_t') \quad (8)$$

$$\vec{\xi}_t \equiv \text{vech}(s_{f,t-1} \varepsilon_t') \quad (9)$$

$$s_{f,t} = \Phi s_{f,t-1} + M \varepsilon_t \quad (10)$$

$$(11)$$

where \tilde{Y}_t is the reordered list of variables as contained in *selvare*, Σ is the covariance matrix of the innovations.

2.2 Some special requirements for the .mod used with Symbsolve and limitations of version 1.0

Symbsolve has been written in order to run .mod files. Nevertheless there are (still) few extra syntax rules that must be adopted in the .mod file.

1. all sentence, comments, groups etc. must be ended with a semicolon (;)
2. The parameter values (or the call to the file containing them) must be bracketed between these two sentences

```
%parameter values;
...
%end parameter values;
```

Notice that the assignment of parameter values is necessary only in order to produce quantitative results. It is not required if interested only in the expansion of the model and in storing the second-order canonical form.

3. The values reported between “initval;” and “end;” must be the true steady state as this is not recomputed by Symbsolve;
4. **Innovations** must be assigned to AR(p) processes and not directly to structural equations. This because the shock processes are assumed to be linear (in logs) and are not expanded.

Although Symbsolve should recognize the equations that govern the **exogenous stochastic process** it is recommended to list these equations last in the equation-block of the .mod file.

As the stochastic processes are taken to be log-linear, the size of the impulse in Symbsolve might need to be scaled by the steady-state value of the shock (to make it comparable to Dynare).³

5. There are a few **reserved names** that should not appear in the model. A possibly incomplete list is

[AA, BB, Xmx, Ax2, Ax1, Ax0, neq, nsh, jumping, numpre, jj, kk]

.

6. One important **limitation** of Symbsolve 1.0 is that leads and lags larger than one have not been tested and the code could crash. If these leads and lags are necessary, they should be replaced with dummy variables, each incrementing the lead and lag by one step at a time. E.g. if $c(+2)$ is needed, add the variable, say, ch and equation $ch = c(+1)$ and replace $c(+2)$ with $ch(+1)$. Likewise with lags.

2.3 Anatomy of Symbsolve

Symbsolve is composed of three main blocks

1. **The main file Symbsolve.m**

This reads the .mod file and extracts the name of the variables and shocks, the equations of the model, the parameter values and the steady-state values (those under “initval;”).

2. **The file firstorder_stuff.m**

This file solves for the first-order state-space representation and, if requested, plots the impulse responses.

³Obviously, if the steady-state value of the shock is 1, the scaling is irrelevant. This suggests to write shocks as to have a unitary steady-state value.

3. The file `secondorder_stuff.m`

This file solves for the second-order state-space representation of the model.

Notice that the last two files need parameter values to run.

2.4 Use of `Symsolve`

`Symsolve` should be used as follows:

1. The **symbolic extraction of the model** is the most time consuming. It should be executed only when the equations of the model have been changed.⁴

Notice that the steady-state values of the variables are represented as `< nameofvariable > _ss`. This is then assigned the steady state value internally. Nevertheless it is convenient that if steady-state values of variables appear in the non-linear model (e.g. in policy rules) these variables are called as `< nameofvariable > _ss`. This will facilitate the symbolic simplification of the expressions, avoiding “NaN” or “inf” returns.

The matrices have the row-order of the equations in the `.mod` file and the **column-order** of the variables as listed in `varyt`.

2. The **numeric first-order solution** of the model can be run at any time provided that the ASCII file with the entries of the matrices exists in the current folder.
3. The **numeric second-order solution** of the model can, likewise, be run at any time provided that the ASCII file with the entries of the second-order matrices exists in the current folder.

This modularization should make it easier to construct loops through some parameter values e.g. for policy evaluations, estimation, sensitivity analysis etc.

⁴As the symbolic canonical form is stored in ASCII files with matrix entries given in standard form, the user could amend those entries directly without the need to re-extract the symbolic equations.

2.5 Bottlenecks and suggestions for further improvements

Symsolve provides some information on the screen regarding the state of the model approximation and matrix extraction. Of these the most time consuming is the step named “PROCEEDING WITH RECONSTRUCTION OF X-PRODUCT TERMS”. This is one of the aspect of Symsolve that requires major improvements.

As for the numeric solution of the approximated system, the file `secorder_stuff.m` contains some operation with large matrices that require further improvement. In particular at present a set of vech-to-vec transformation matrices is produced quite expensively. It seems to me that this could be avoided, although I haven’t yet figured out how.

2.6 Troubleshooting

Some known problems that could emerge are

1. Symsolve does not solve correctly (though a solution exists e.g. using Dynare)
 - As Symsolve takes symbolic approximations, some expressions might yield infinite values or NaN if not assigned correctly. For example if a variable enters also as a parameter (e.g. steady state of a tax in a tax rule) but is given a different name than `< nameofvariable > _ss` (see earlier comment) the evaluation might be incorrect: e.g. consider the entry `tau_othertype/tau_ss` that could appear in the ASCII file for the model’s matrices. If `tau` is zero in the steady state, the evaluation of the ratio will produce NaN. Nevertheless, if the `< nameofvariable > _ss` was used instead of `tau_othertype`, the value of 1 would be returned.

References

- Klein, Paul (2000) ‘Using the generalized Schur form to solve a multivariate linear rational expectations model.’ *Journal of Economic Dynamics and Control* 24, 1405–1423
- Lombardo, Giovanni, and Alan Sutherland (2005) ‘Computing Second-Order-Accurate Solutions for Rational Expectation Models using Linear Solution Methods.’ European Central Bank, Working Paper No. 487, May (forthcoming JEDC)