Package Guide for SolveDSGE

Richard Dennis*
University of Glasgow

December 1, 2014

*Email: richard.dennis@glasgow.ac.uk

1 Introduction

SolveDSGE is a Julia package for solving Dynamic Stochastic General Equilibrium (DSGE) models. The package is aimed at macroeconomists interested in first-order-accurate or second-order-accurate solutions to their general equilibrium models. Although there is much that this package does not do, SolveDSGE offers a broad array of solution methods that can be applied provided the DSGE model can be expressed in one of several standard dynamic representations.

2 Installation

To install SolveDSGE you will need to type the following into the Julia REPL

Pkg.clone("git://github.com/RJDennis/SolveDSGE.git")

3 First-order-accurate solution methods

SolveDSGE provides a range of solution methods for computing first-order accurate solutions. Exploiting Julia's multiple dispatch all of these solution methods are called via the single command **solve_re**(). From this single command the particular solution method employed depends principally on the model type that enters the **solve_re()** function call. Models are represented in various forms that are summarized by types. The model types are

- **Blanchard_Kahn_Form**
- **Klein_Form**
- **Sims_Form**

- **Binder_Pesaran_Form**

3.1 Blanchard-Kahn Form

The Blanchard-Kahn Form is given by

$$\left[\begin{array}{c} x_{t+1} \\ \mathrm{E}_t y_{t+1} \end{array} \right] = A \left[\begin{array}{c} x_t \\ y_t \end{array} \right] + C \left[\epsilon_{t+1} \right],$$

$\epsilon_t \sim i.i.d.[0, \Sigma]$, where x_t is an $n_x \times 1$ vector of predetermined variables and y_t is an $n_y \times 1$ vector of non-predetermined variables. To solve models of this form we first create the **Blanchard_Kahn_Form** type for the model, then we use **solve_re()** to solve the model. The relevant lines of code would be something like

cutoff = 1.0
model = Blanchard_Kahn_Form(nx,ny,a,c,sigma)
soln = solve_re(model,cutoff)

Here **soln** contains the solution, which is of the form

$$\begin{aligned} x_{t+1} &= Px_t + K\epsilon_{t+1}, \\ y_t &= Fx_t, \end{aligned}$$

and information about the number of eigenvalues with modulus greater than cutoff and whether the "solution" returned is determinate, indeterminate, or explosive.

Models of the **Blanchard_Kahn_Form** type can also be solved using an iterative method to solve a non-symmetric, continuous, algebraic, Riccati equation. In this case the relevant lines of code might look like

tol = 1e-10
cutoff = 1.0
model = Blanchard_Kahn_Form(nx,ny,a,c,sigma)

soln = solve_re(model,cutoff,tol)

For this iterative method, the variable **cutoff** is still used to establish determinacy, but is not used to order eigenvalues.

3.2 Klein Form

The Klein Form for a model is given by

$$B \begin{bmatrix} x_{t+1} \\ \mathrm{E}_t y_{t+1} \end{bmatrix} = A \begin{bmatrix} x_t \\ y_t \end{bmatrix} + C \left[\epsilon_{t+1} \right],$$

$\epsilon_t \sim i.i.d.[0, \Sigma]$, where x_t is an $n_x \times 1$ vector of predetermined variables, y_t is an $n_y \times 1$ vector of non-predetermined variables, and B need not have full rank. We can solve models of this form using the code

cutoff = 1.0
model = Klein_Form(nx,ny,a,b,c,sigma)
soln = solve_re(model,cutoff)

The composite type **soln** contains the solution which is of the form

$$\begin{aligned} x_{t+1} &= Px_t + K\epsilon_{t+1}, \\ y_t &= Fx_t, \end{aligned}$$

as well as information about the number of eigenvalues that have modulus greater than cutoff and whether the "solution" returned is determinate, indeterminate, or explosive.

3.3 Sims Form

An alternative model form is used by Sims (2000) and is given by

$$\Gamma_0 z_t = \Gamma_1 z_{t-1} + C + \Psi v_t + \Pi \eta_t,$$

where v_t is a shock process, possibly serially correlated, with mean-zero innovations whose variance-covarince matrix is given by Σ. To solve models that are in this form we would do something like the following

cutoff = 1.0
model = Sims_Form(gamma0,gamma1,c,psi,pi,sigma)
soln = solve_re(model,cutoff)

Here the solution, summarized by **soln**, is of the form

$$z_t = G_1 z_{t-1} + c + impact \times v_t + ywt \times \left[I - fmat \times L^{-1}\right]^{-1} \times fwt \times v_{t+1}.$$

3.4 Binder-Pesaran Form

A model is in "structural form" if it is written as

$$Az_t = A_1 z_{t-1} + B\mathrm{E}_t z_{t+1} + C\epsilon_t,$$

where $\epsilon_t \sim i.i.d.[0, \Sigma]$. We have two ways of solving structural form models. The first recasts them in terms of the Klein form and here the relevant code would look something like

cutoff = 1.0
model = Binder_Pesaran_Form(a,a1,b,c,sigma)
soln = solve_re(model,cutoff)

The second method is iterative, implementing Binder and Pesaran's "brute force" method; here the code would be something like

tol = 1e-10
cutoff = 1.0
model = Binder_Pesaran_Form(a,a1,b,c,sigma)
soln = solve_re(model,cutoff,tol)

Regardless of which of the two methods is used, the solution, summarized in **soln**, has the form

$$z_t = Pz_{t-1} + K\epsilon_t.$$

As earlier, **soln** is a composite type that in addition to the solution itself also contains information about the number of eigenvalues with modulus greater than cutoff and whether the solution returned is determinate, indeterminate, or explosive.

4 Second-order-accurate solution methods

In addition to the first-order accurate solution methods documented above, SolveDSGE also contains two methods form obtaining second-order-accurate solutions to nonlinear DSGE models. As coded here, these two methods employ the same model form but differ in how the solution is computed. To employ either method the DSGE model is first expressed in the form

$$\mathrm{E}_t G\left(x_t, y_t, x_{t+1}, y_{t+1}\right) = 0.$$

With x_t containing n_x predetermined variables and y_t containing n_y non-predetermined variables, $G()$ is a vector containing n_x+n_y functions. Bundling x_t and y_t into a new vector $z_t = \left[\begin{array}{cc} x_t^{'} & y_t^{'} \end{array}\right]^{'}$ and bundling z_t and z_{t+1} into a new vector $p_t = \left[\begin{array}{cc} z_t^{'} & z_{t+1}^{'} \end{array}\right]^{'}$ we get

$$\mathrm{E}_t G\left(p_t\right) = 0.$$

We now approximate $G\left(p_t\right)$ around the deterministic steady state, $\overline{p}$, using a second-order Taylor expansion giving

$$G\left(p_t\right) \simeq G_p\left(p_t - \overline{p}\right) + \left[I \otimes \left(p_t - \overline{p}\right)\right]^{'} G_{pp}\left[I \otimes \left(p_t - \overline{p}\right)\right] = 0,$$

where G_p is a matrix of first-derivatives and G_{pp} is a matrix of stacked Hessians, one Hessian for each of the $n_x + n_y$ equations.

We now recognize that some elements of x_t (usually the first s elements) are shocks that have the form

$$s_{t+1} = \Lambda s_t + \eta \epsilon_{t+1},$$

where $\epsilon_t \sim i.i.d.[0, \Sigma]$. The essential components required for a second-order-accurate solution are now given by n_x, n_y, G_p, G_{pp}, η, and Σ.

The two model types that we consider for second-order-accurate solution methods are

- **Gomme_Klein_Form**
- **Lombardo_Sutherland_Form**

4.1 Gomme-Klein Form

To compute a second-order accurate solution using the Gomme and Klein (2011) method we summarize the model in the form of the **Gomme_Klein_Form** composite type. Once this model type is constructed the model can be solved. The code to compute the solution would be something like

```
cutoff = 1.0
model = Gomme_Klein_Form(nx,ny,Gp,Gpp,eta,sigma)
soln = solve_re(model,cutoff)
```

Here **soln** is a composite type that contains the solution, which is of the form

$$\begin{aligned} x_{t+1} - \overline{x} &= \frac{1}{2} ssh + hx\left(x_t - \overline{x}\right) + \frac{1}{2}\left[I \otimes \left(x_t - \overline{x}\right)\right]' hxx \left[I \otimes \left(x_t - \overline{x}\right)\right] + \eta \epsilon_{t+1}, \\ y_t - \overline{y} &= \frac{1}{2} ssg + gx\left(x_t - \overline{x}\right) + \frac{1}{2}\left[I \otimes \left(x_t - \overline{x}\right)\right]' gxx \left[I \otimes \left(x_t - \overline{x}\right)\right], \end{aligned}$$

where hxx and gxx are stacked matrices containing the second order coefficients for each of the n_x and n_y equations, respectively. **soln** also contains

information about the number of eigenvalues with modulus greater than cutoff and the solution's determinacy properties, where these properties are associated with the model first-order dynamics.

4.2 Lombardo-Sutherland Form

Implementing the Lombardo and Sutherland (2007) solution method is no different than for Gomme and Klein (2010). The key difference is the form in which the solution is presented. Ths code to implement the Lombardo-Sutherland method would look something like

cutoff = 1.0
model = Lombardo_Sutherland_Form(nx,ny,Gp,Gpp,eta,sigma)
soln = solve_re(model,cutoff)

where now the solution has the form

$$\begin{aligned} x_{t+1} - \overline{x} &= \frac{1}{2} ssh + hx\left(x_t - \overline{x}\right) + \frac{1}{2} hxx \times v_t + \eta \epsilon_{t+1}, \\ y_t - \overline{y} &= \frac{1}{2} ssg + gx\left(x_t - \overline{x}\right) + \frac{1}{2} gxx \times v_t, \\ v_t &= \Phi v_{t-1} + \Gamma vech\left(\epsilon_t \epsilon_t^{'}\right) + \Psi vec\left(x_t \epsilon_t^{'}\right), \end{aligned}$$

with v_t given by

$$v_t = vech\left(x_t x_t^{\prime}\right).$$

The solution form produced by the Lombardo-Sutherland method can be converted into that produced by the Gomme-Klein method by using the **convert_second_order** function as follows

new_soln = convert_second_order(soln)

Where **soln** is of type **Lombardo_Sutherland_Soln**, **new_soln** is of type **Gomme_Klein_Soln**.

References

[1] Binder, M., and H. Pesaran, (1995), "Multivariate Rational Expectations Models and Macroeconomic Modeling: A Review and Some New Results," in: Pesaran, H., M. Wickens, (Eds.), *Handbook of Applied Econometrics*, Basil Blackwell, Oxford, pp.139–187.

[2] Blanchard, O., and C. Kahn, (1980), "The Solution to Linear Difference Models under Rational Expectations," *Econometrica*, 48, pp.1305–1311.

[3] Gomme, P., and P. Klein, (2011), "Second-Order Approximation of Dynamic Models Without the Use of Tensors," *Journal of Economic Dynamics and Control*, 35, pp.604–615.

[4] Klein, P., (2000), "Using the Generalized Schur Form to Solve a Multivariate Linear Rational Expectations Model," *Journal of Economic Dynamics and Control*, 24, pp.1405–1423.

[5] Lombardo, G., and A. Sutherland, (2007), "Computing Second-Order-Accurate Solutions for Rational Expectations models Using Linear Solution Methods," *Journal of Economic Dynamics and Control*, 31, pp.515–530.

[6] Sims, C., (2001), "Solving Linear Rational Expectations Models," *Computational Economics*, 20, pp.1–20.