Solving LQ-Gaussian Robustness Problems in Dynare

July 2008

- Ellsberg's Paradox. Gambling Example.
- Knightian Uncertainty: there is a difference between Risk and Uncertainty.
- Robustness is a method that **Rationalizes** concerns about Knightian Uncertainty into certain preferences.
- What are these preferences? The General Case deals with entropy...I will show something simpler here.

- We attempt to show that Robustness Problems in LQ-Gaussian setups are easy to solve using **Dynare**.
- Linear Quadratic Gaussian Models are very general, they can approximate dynamic preferences, habit persistence, adjustment costs, durable goods, etc.
- We present a matlab subroutine that does the job of writing a mod file for you once you specify the LQ-setup.
- The following presentation reviews the LQ setup of Hansen and Sargent, the Robustness problem, shows how the code works and uses an example.

Problem (LQ-Regulator)

$$\max_{\{c_t, k_t\}} -(1/2)\sum_{t=0}^{\infty}\beta^t\left[(s_t - b_t)\cdot(s_t - b_t) + (g_t)\cdot(g_t)\right]$$

subject to the following set of restrictions:

$$\begin{aligned}
\Phi_c c_t + \Phi_g g_t + \Phi_i i_t &= \Gamma k_{t-1} + d_t \\
k_t &= \Delta_k k_{t-1} + \Theta_k i_t \\
h_t &= \Delta_h h_{t-1} + \Theta_h c_t \\
s_t &= \Lambda h_{t-1} + \Pi c_t
\end{aligned}$$

and exogenous variables evolve according to a VAR structure:

$$z_{t+1} = A_{22} z_t + C_2 w_{t+1},\ b_t = U_b z_t \text{ and } d_t = U_d z_t$$

w_{t+1} *is i.i.d Standard-Normal vector.*

- To solve this program we use a Lagrangian method:

$$
\begin{aligned}
&-(1/2)\,E\{\sum_{t=0}^{\infty}\beta^t\,[(s_t-b_t)\cdot(s_t-b_t)+(g_t)\cdot(g_t)]+\ldots\\
&+M_t^{d\prime}\,[\Phi_c c_t+\Phi_g g_t+\Phi_i i_t-\Gamma k_{t-1}-d_t]\ldots\\
&+M_t^{k\prime}\,[k_t-\Delta_k k_{t-1}-\Theta_k i_t]\ldots\\
&+M_t^{h\prime}\,[h_t-\Delta_h h_{t-1}-\Theta_h c_t]\ldots\\
&+M_t^{s\prime}\,[s_t-\Lambda h_{t-1}-\Pi c_t]\}
\end{aligned}
$$

The FONC for the problem, found in system 4.2.4 of HS-LQ book are:

$$
\begin{aligned}
&-(1/2)\, E\{\sum_{t=0}^{\infty} \beta^t \left[(s_t - b_t)\cdot(s_t - b_t) + (g_t)\cdot(g_t)\right] + \ldots \\
&+M_t^{d\prime}\left[\Phi_c c_t + \Phi_g g_t + \Phi_i i_t - \Gamma k_{t-1} - d_t\right]\ldots \\
&+M_t^{k\prime}\left[k_t - \Delta_k k_{t-1} - \Theta_k i_t\right]\ldots \\
&+M_t^{h\prime}\left[h_t - \Delta_h h_{t-1} - \Theta_h c_t\right]\ldots \\
&+M_t^{s\prime}\left[s_t - \Lambda h_{t-1} - \Pi c_t\right]\}
\end{aligned}
$$

and the restrictions:

$$
\begin{aligned}
\Phi_c c_t + \Phi_g g_t + \Phi_i i_t &= \Gamma k_{t-1} + d_t \\
k_t &= \Delta_k k_{t-1} + \Theta_k i_t \\
h_t &= \Delta_h h_{t-1} + \Theta_h c_t \\
s_t &= \Lambda h_{t-1} + \Pi c_t \\
b_t &= U_b z_t \\
d_t &= U_d z_t
\end{aligned}
$$

- We have a system of stochastic difference equations. Dynare is the perfect environment for solving for the optimal feedback policies.

The Robustness problem is the artificial *simultaneous* game:

$$\min_{\{w_{t+1}\}} \max_{\{c_t, k_t\}} -E\{\sum_{t=0}^{\infty} \beta^t \left(\frac{1}{2}\right) [(s_t - b_t) \cdot (s_t - b_t) + (g_t) \cdot (g_t)] - \beta\theta w_{t+1} \cdot w_{t+1}\}$$

subject to:

$$\begin{aligned}
\Phi_c c_t + \Phi_g g_t + \Phi_i i_t &= \Gamma k_{t-1} + d_t \\
k_t &= \Delta_k k_{t-1} + \Theta_k i_t \\
h_t &= \Delta_h h_{t-1} + \Theta_h c_t \\
s_t &= \Lambda h_{t-1} + \Pi c_t \\
b_t &= U_b z_t \\
d_t &= U_d z_t
\end{aligned}$$

and the exogenous process:

$$b_t = U_b z_t, d_t = U_d z_t.$$

The **difference** here is that minimizer chooses w_{t+1}:

$$z_{t+1} = A_{22} z_t + C_2 w_{t+1}$$

so w_{t+1} is not a Guassian vector any more.

The LQ-Robustness problem may also be written as a Lagrangian:

$$\min_{\{w_{t+1}\}} \max_{\{c_t, k_t\}} -\{\sum_{t=0}^{\infty} (1/2)\,\beta^t \left[(s_t - b_t)\cdot(s_t - b_t) + (g_t)\cdot(g_t)\right] - \beta\theta \frac{w_{t+1}\cdot w_{t+1}}{2}\}$$
$$+M_t^{d\prime}\left[\Phi_c c_t + \Phi_g g_t + \Phi_i i_t - \Gamma k_{t-1} - d_t\right]$$
$$+M_t^{k\prime}\left[k_t - \Delta_k k_{t-1} - \Theta_k i_t\right]$$
$$+M_t^{h\prime}\left[h_t - \Delta_h h_{t-1} - \Theta_h c_t\right]$$
$$+M_t^{s\prime}\left[s_t - \Lambda h_{t-1} - \Pi c_t\right]$$
$$+M_t^{z\prime}\left[-z_{t+1} + A_{22} z_t + C_2 w_{t+1}\right]\}$$

The NFOC's are:

$$
\begin{aligned}
(c_t) &\quad : \quad -M_t^{d\prime}\Phi_c + M_t^{h\prime}\Theta_h + M_t^{s\prime}\Pi = 0 \\
(g_t) &\quad : \quad g_t - \Phi_g M_t^d = 0 \\
(h_t) &\quad : \quad -M_t^{h\prime} + \beta E\left[M_{t+1}^{h\prime}\Delta_h + M_{t+1}^{s\prime}\Lambda\right] = 0 \\
(i_t) &\quad : \quad -M_t^{d\prime}\Phi_i + M_t^{k\prime}\Theta_i = 0 \\
(k_t) &\quad : \quad -M_t^{k\prime}\Delta_k + \beta E\left[M_{t+1}^{d\prime}\Gamma + M_{t+1}^{k\prime}\Delta_k\right] = 0 \\
(s_t) &\quad : \quad -(s_t - b_t) + M_t^s = 0 \\
(w_{t+1}) &\quad : \quad \beta\theta w_{t+1} + C_2^{\prime} M_t^z = 0 \\
(z_{t+1}) &\quad : \quad -\beta E\left[U_d^{\prime} M_{t+1}^d\right] + \beta E\left[A_{22}^{\prime} M_{t+1}^z\right] - \beta E\left[\left(s_{t+1} - b_{t+1}\right) U_b^{\prime} M_{t+1}^z\right] - M_t^z = 0
\end{aligned} \tag{1}
$$

This set of equations is then complemented by:

$$
\begin{aligned}
\Phi_c c_t + \Phi_g g_t + \Phi_i i_t &= \Gamma k_{t-1} + d_t \\
k_t &= \Delta_k k_{t-1} + \Theta_k i_t \\
h_t &= \Delta_h h_{t-1} + \Theta_h c_t \\
s_t &= \Lambda h_{t-1} + \Pi c_t
\end{aligned}
$$

and the minimizer or "evil" agent chooses w_{t+1}, that will affect:

$$z_{t+1} = A_{22} z_t + C_2\left(w_{t+1} + \epsilon_t\right)$$

where $\epsilon_t \sim N(0, I)$.

- The reason is that the solution to the minimizing problem is a choice of a mean + noise.
- This is a property of LQ-Gaussian Robust Problems.
- For these reasons, the solution is again a set of stochastic linear difference equations. Again, Dynare is the perfect environment for solving for the optimal feedback robust policies!

1. **Declaration of matrixes:** The first thing the user has to do is declare ALL of these matrixes: `Phi_c`, `Phi_g`, `Phi_i`, `Gamma`, `Delta_k`, `Theta_k`, `Delta_h`,.`Theta_h`, `Lambda`, `Pi`, `A22`,`C2`, `U_b`, `U_d`;
2. **Declaration of options for Dynare:** Variables that go in the standard set of options for dynare should be included. For example one can include statments like:
   - `periods=5000;` (for number of simulations in Dynare)
   - `irfperiods=15;` (for number of periods in irf's in Dynare)
3. **File Name:** A statemant assigning the variable `filename` a script should be included:(e.g. `filename ='Hall_Test.mod';`)
4. **Optional:** Assign a value for the optional dummy variable `run` if you decide whether or not you want to directly execute the generated dynare. Assign `run=1;`
5. **Calling the file:** Call the Hansen Sargent converter by simply writing the following statement `HS_Robust_dyn`;

How does it does the code do it? The code does something simple. It

reads the matrixes from HS-LQ setup and declares variables and equations in a mod.file using dynare syntax. The steps followed are:

1. Reading Dimensions of Matrixes
   - For example, for the matrix `Phi_c`, it uses the following statement to set in memory the number of rows and columns: [rr_Phi_c cc_Phi_c] =size(Phi_c) ;
2. Checking Consistency:
   - The matrixes must be conformable in HS-LQ setup. The code checks for consistency along columns and rows:
3. Creating an m-file:
   - This statement generates an object in memory that is used to record what we are writing:
     `fid = fopen(filename,'wt');`

1. Declaring Variables: Reads dimensions and declares a variable.
2. Declaring Parameters is done in a similar way:
   - ```
     fprintf(fid,'parameters '); fprintf(fid,'beta ') ;
     fprintf(fid,'theta ') ; for j=1:rr_Phi_c
         for i=1:cc_Phi_c
              fprintf(fid,'Phi_c_%g%g ',j,i);
         end
     end
     ```
     ```

1. Setting Parameter Values:
   - Using a similar statement we declare parameters. The variable %g in the statement is assign the value of the variable after the second comma. n means that the lines ends there:

```
fprintf(fid,'beta=%g ; n',beta) ;
fprintf(fid,'theta=%g ;n',theta) ;
for j=1:rr_Phi_c
    for i=1:cc_Phi_c
        fprintf(fid,'Phi_c_%g%g =%g ; n',j,i,Phi_c(j,i));
    end
end
```

1. Declaring Equations:

   - Let's use an example: For the first block of equations in the system, $M_t^{d\prime}\Phi_c + M_t^{h\prime}\Theta_h = 0$, we use the following commands:

```
for i=1:(rr_Phi_c)
    for j=1:(cc_Phi_c)
        fprintf(fid,'Phi_c_%g%g*c%g + ',i,j,j);
    end
    for j=1:(cc_Phi_g)
        fprintf(fid,'Phi_g_%g%g*g%g + ',i,j,j);
    end
    for j=1:(cc_Phi_i)
        fprintf(fid,'Phi_i_%g%g*i%g = ',i,j,j);
    end
    for j=1:(cc_Gamma)
        fprintf(fid,'Gamma_%g%g*k%g + ',i,j,j);
    end
    fprintf(fid,'d%g',i);
    fprintf(fid,'; n');
end
```

1. Running Dynare:
   - This step simply follows the dynare syntax to call dynare according to the previously defined preferences:

```
fprintf(fid,'shocks; n');
for i=1:cc_C2
    fprintf(fid,'var e%g; n',i);
    fprintf(fid,'stderr 1; n');
end
fprintf(fid,'end; n n');
fprintf(fid,' n');
fprintf(fid,'steady; n');
fprintf(fid,'stoch_simul(solve_algo=3, periods=%g);
n',irfperiods,periods);
fclose(fid);
if (run)
    dynare(filename)
end
```

The planner solves the following problem:

$$\min_{\{w_{t+1}\}} \max_{\{c_t, k_t\}} E\{\sum_{t=0}^{\infty} \beta^t - (1/2)\left[(s_t - b_t)^2 - \beta\Theta \frac{w_{t+1} \cdot w_{t+1}}{2}\right]$$

The input of the utility function are an indirect service for the household, s_t, and a preference shock b_t that satisfy the following:

$$s_t = \Lambda h_{t-1} + \Pi c_t$$

Resource constraints:

$$c_t + i_t = (1+r)\, k_{t-1} + y_t$$

Productivity Shifts:

$$y_t = U_y z_t$$

Again, the Robust planner problem is different from the standard model in that: a minimizer or "evil" agent chooses w_{t+1}, that will affect:

$$z_{t+1} = A_z z_t + C_Z w_{t+1}$$

and we parameterize this to have a stable solution:

$$\gamma := (1 + r) = \frac{1}{\beta}$$

For a pair of AR(2) process, we have the following values of A_z and C_Z are:

$$A_z = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \rho_{22} & \rho_{23} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho_{44} & \rho_{45} \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \text{ and } C_z = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \text{ and } U_y = [5\ 1\ 1].$$

The following code is declares the variables and runs dynare:

```
% Declaration of Paramteters
delta=0.05;
beta=1/1.05;
theta=10;


% Matrix Form
Phi_c=1 ;
Phi_g=0 ;
Phi_i=1 ;
Gamma=0.1 ;
Delta_k=1-delta ;
Theta_k=1 ;
Delta_h=0 ;
Theta_h=1 ;
Lambda=0 ;
Pi=1 ;
A22=[1 0 0; 0 0.0 0; 0 0 0.0] ;
C2=[0 0; 1 0; 0 1] ;
U_b=[30 0 1] ;
U_d=[5 1 0] ;


%% User Preferences for Dynare
periods=5000; %parameters for simulation
irfperiods=15; %parameters for irf's
filename='Hall_Test.mod'; %pick filename for mod file
%Optional:  decide whether or not you want to directly execute the generated dynare
%code (set run=1 to execute)


run=1;
%% Call the Hansen Sargent converter:

HS_Robust_dyn;
```

- LQ is a general setup. :)
- My guess is that we can extend this to Robust Filtering. :—)
- We cannot deal yet with Optimal Robust problems with expectations. :(
- The code may still present some errors. :(