MONFISPOL FP7 project SSH-225149

Deliverable 1.1.2

User manual for optimal policy package

Michel Juillard[1]

September 2011

[1]CEPREMAP

Dynare has tools to compute optimal policies for various types of objectives. You can either solve for optimal policy under commitment with 'ramsey_policy' or for optimal simple rule with 'osr'.

This document provides a user manual for the related commands in Dynare. The last chapter presents a complete example.

Chapter 1

Optimal simple rule

The optimal simple rule problem is to minimize the a quadratic loss function of the form

$$\min_{\gamma} E(y_t' W y_t)$$

s.t

$$A_1 E_t y_{t+1} + A_2 y_t + A_3 y_{t-1} + C e_t = 0$$

where γ is a subset of parameters appearing in A_1, A_2, A_3 and pertaining to the policy rule; W is a semi–positive definite matrix; y_t is the vector of endogenous variables; e_t is a vector of shocks; A_1, A_2, A_3 and C are coefficient matrices.

The objective can equivalently be expressed as minimizing a weighted sum of variances and covariances of endogenous variables.

The computation of optimal simple rules is controlled by main command **osr** and auxiliary commands **osr_params** and **optim_weights**:

osr_params lists the name of the parameters belonging to γ;

optim_weights enumerates the non-zero elements of the matrix of the quadratic form W;

osr controls the computation of the optimal simple rule and displays results.

1.1 Command: osr_params

osr_params is used to declare the parameters over which optimization takes place.

1.1.1 Syntax

osr_params [PARAMETER_NAME...];

PARAMETER_NAME name of a parameter that must be computed optimally

These parameters must have been declared previously with the **parameters** command and must be used in the model.

1.1.2 Example

```
osr_params gamma1 gamma2;
```

The **osr** procedure must find the optimal value of parameters **gamma_1** and **gamma_2**.

1.2 Block: optimal_weights

The **optimal_weights** block is used to declare the non–zero elements of quadratic form matrix W.

1.2.1 Syntax

optimal_weights;
[VARIABLE_NAME EXPRESSION];
[VARIABLE_NAME, VARIABLE_NAME EXPRESSION];
end;

VARIABLE_NAME name of an endogenous variable;

EXPRESSION an expression returning a numerical scalar.

The weight on the variance of an endogenous variable is declared using

```
[VARIABLE_NAME EXPRESSION];
```

syntax, while the weight of the covariance between two endogenous variables is specified with the following syntax.

```
[VARIABLE_NAME, VARIABLE EXPRESSION];
```

1.2.2 Example

```
optim_weights;
pie lambda;
y 1;
end;
```

This defines the loss function as being

$$\lambda \text{var}(pie) + \text{var}(y).$$

1.3 Command: osr

The **osr** command triggers the computation of optimal simple rule and displays the results. Its output is very similar to the output of command **stoch_simul**.

1.3.1 Syntax

osr [VARIABLE_NAME...];
osr (OPTION...) [VARIABLE_NAME...];

VARIABLE_NAME name of an endogenous variable. This permits to limit output to those variables listed here.

OPTION options. **osr** accepts all commands of **stoch_simul**.

1.3.2 Example

```
osr(irf=10) pie, y, r;
```

triggers the computation of an optimal simple rule, requesting IRF over 10 periods and limiting reports to π, y and r.

1.3.3 Output

Command **osr** produces the same output as **stoch_simul** and, in addition,

- the optimal value of the parameters;
- the value of the loss function at the optimum

After **osr**, the value of the parameters is set to their optimal values.

1.3.4 Algorithm

osr uses numerical optimization as provided in **csminwel.m**, written by Chris Sims.

Initial values for the parameters of the optimal rule must be indicated before calling **osr**.

1.3.5 Remarks

Numerical optimizers only find local optima. It is the responsibility of the user to insure that the optimum return by **osr** is a global optimum. At minimum, the user should at minimum try different initial values for the parameters of the policy rule.

A better practice is to explore the space of parameters before calling **osr** and use this exploration to set the initial values for the parameters of the optimal rule.

1.4 A complete example

Consider the following 4-equation economy, where y_t is the output gap, π_t, inflation, r_t, the short term nominal interest rate, and, dr_t, the change in interest rate:

$$\begin{aligned} y_t &= \delta y_{t-1} + (1-\delta)E_t y_{t+1} + \sigma(r_t - E_t\pi_{t+1}) + e_{y_t} \\ \pi_t &= \alpha\pi_{-1} + (1-\alpha)E_t\pi_{t+1} + \kappa y_t + e_{\pi_t} \\ dr_t &= r_t - r_{t-1} \\ r_t &= \gamma_1\pi_t + \gamma_2 y_t \end{aligned}$$

The problem is to find optimal value for γ_1 and γ_2 so has to minimize a weighted sum of the variances of inflation, output gap and the change in interest rate

Objectif

$$\min_{\gamma_1,\gamma_2} \text{var}(y) + \lambda_1 \text{var}(\pi) + \lambda_2 \text{var}(dr)$$

The following code permits to solve the above problem using Dynare:

```
var y pie r dr;
varexo e_y e_pie;

parameters delta sigma alpha kappa gamma1 gamma2 lambda1 lambda2;

delta =  0.44;
kappa =  0.18;
alpha =  0.48;
sigma = -0.06;

model(linear);
y  = delta*y(-1)+(1-delta)*y(+1)+sigma *(r-pie(+1))+e_y;
pie = alpha*pie(-1)+(1-alpha)*pie(+1)+kappa*y+e_pie;
dr = r - r(-1);
r = gamma1*pie+gamma2*y;
end;

shocks;
var e_y;
stderr 0.63;
var e_pie;
stderr 0.4;
end;

lambda1 = 2;
lambda2 = 0.1;

optim_weights;
pie lambda1;
```

```
y 1;
dr lambda2;
end;

gamma1 = 1.1;
gamma2 = 0;

osr_params gamma1 gamma2;

osr;
```

Chapter 2

Ramsey policy

Ramsey policy solves the following optimal control problem:

$$\max_{\{y_\tau\}_{\tau=0}^{\infty}} E_t \sum_{\tau=t}^{\infty} \beta^{\tau-t} U(y_\tau)$$

s.t.

$$E_\tau f(y_{\tau+1}, y_\tau, y_{\tau-1}, \varepsilon_\tau) = 0$$

where y_t is a vector of n endogenous variables, $U(y_)$ is the objective function of the policy maker, β her discount factor.

$$E_\tau f(y_{\tau+1}, y_\tau, y_{\tau-1}, \varepsilon_\tau) = 0$$

represents the set of constraints imposed to the policy maker by the behavior of private agents.

The computation of Ramsey policy is controlled by main command **ramsey_policy** and auxiliary commands **planner_objective** and **mult_elimination**.

planner_objective specifies the objective of the policy planner;

ramsey_policy controls the computation of Ramsey policy;

mult_elimination eliminates the Lagrange multipliers from the solution.

2.1 planner_objective

Command **planner_objective** declares the policy maker objective function.

2.1.1 Syntax

planner_objective MODEL EXPRESSION;

MODEL EXPRESSION is the one-period objective, not the discounted lifetime objective. The discount factor is given by the **planner_discount** option of **ramsey_policy**.

2.1.2 Example

```
planner_objective ln(c)-phi*n^(1+gamma)/(1+gamma);
```

2.2 ramsey_policy

Command **ramsey_policy** controls the computation of optimal Ramsey policy.

2.2.1 Syntax

ramsey_policy [VARIABLE_NAME...];
ramsey_policy (OPTION...) [VARIABLE_NAME...];
where

VARIABLE_NAME name of an endogenous variable. This permits to limit output to those variables listed here.

OPTION any of

planner_discount = DOUBLE the planner discount factor (β in the model used in introduction)

order = INTEGER the order of approximation used for the set of first order conditions of the problem of the policy maker and of the corresponding solution

instruments = ([VARIABLE_NAME, ...)] name of the policy instruments. Naming policy instruments is not necessary to find the solution. It is however useful, when providing a analytical, recursive solution for the steady state under optimal policy. Note that in that case, the term *instrument* is used somewhat abusively.

all other options accepted by stoch_simul

2.2.2 Outputs

Command **ramsey_policy** generates the same output as **stoch_simul**. In addition, it provides two approximate evaluation of the current value of the objective under optimal policy. These evaluations are computed on the basis of a second order approximation of the objective function and of the first order conditions. The first evaluation is computed while setting the initial value of the Lagrange multipliers to 0. The second evaluation, on the basis of setting them to their steady state value.

2.2.3 Computing the steady state

Computing the steady state under optimal policy may be difficult. Furthermore even a model for which there exists a steady state for constant value of the policy instruments may not have a finite steady state under optimal policy (consider, for example, a model with unsatiable demand for money and flexible prices).

Dynare offers three different strategies to compute the steady state under optimal policy:

1. **provide an analytical, recursive solution for the steady state**. In most cases, it is not possible or necessary to provide exact steady state values for the endogenous variables and the Lagrange multipliers. The steady state of Lagrange mutlipliers can be easily derived from the steady state value of the endogenous variables and Dynare always computes them for you.

   Sometimes, it is possible to derive by hand optimal values for the steady state of all endogenous variables. These values or recursive relations permitting to compute them as a function of the parameters can be entered in a **steady_state_model** block.

   More often, it is only possible to derive the steady state values of the other endogenous variables given the value of the policy instruments or variables closely related to them (for example, in a monetary model, the inflation rate instead of the nominal interest rate). In this case as well, the analytical, recursive steady state can be entered in the **steady_state_model** block.

   It is then necessary to declare the name of the endogenous variables that are considered as given in the **instruments** option and to provide a guess value for the instruments in the **initval** block.

   Using this approach, Dynare has to solve a non–linear problem with only as many unkowns as there are instruments.

2. to provide numerical guess values for all endogenous variables (except Lagrange multipliers) in a **initval** block. In this case, Dynare solves a non-linear problem with as many unknowns as there are endogenous variables. More numerical difficulties can arise than with the first strategy.

3. write your own **¡fname¿_steadystate.m** Matlab function. This is only for people with good knowledge of Matlab and Dynare internals. It is rarely necessary given the existence of the first strategy above.

Examples with the first two strategies are provided in the **complete examples** chapter, below.

2.2.4 Example

```
ramsey_policy(planner\_discount=0.99,order=1,instruments=(r));
```

2.3 eliminate_lagrange_multipliers

When computing optimal policy in a timeless perspective, it is useful to replace the lagged value of the Lagrange multipliers in the solution by additional lagged values of endogenous variables and shocks. This substitution is operated by command **eliminate_lagrange_multipliers**

2.3.1 Syntax

eliminate_lagrange_multipliers;
This command must be placed after **ramsey_policy**. Note that the solution is not necessarily unique.

2.3.2 Example

```
ramsey_policy(planner_discount=0.99,order=1,instruments=(r));
eliminare_lagrange_multipliers;
```

Chapter 3

Complete examples

Let's consider a new-Keynesian model with price adjustment cost.0

The utility function and its derivatives are

$$\begin{aligned} U_t &= \ln C_t - \phi \frac{N_t^{1+\gamma}}{1+\gamma} \\ U_{c,t} &= \frac{1}{C_t} \\ U_{N,t} &= -\phi N_t^{\gamma} \end{aligned}$$

The recursive equilibrium of the private economy is given by

$$\begin{aligned} \frac{1}{C_t} =& \beta E_t \left\{ \frac{1}{C_{t+1}} \frac{R_t}{\pi_{t+1}} \right\} \\ \frac{\pi_t}{C_t} \left(\pi_t - 1\right) =& \beta E_t \left\{ \frac{\pi_{t+1}}{C_{t+1}} \left(\pi_{t+1} - 1\right) \right\} \\ &+ \varepsilon \frac{A_t}{\omega} \frac{N_t}{C_t} \left(\frac{\phi C_t N_t^{\gamma}}{A_t} - \frac{\varepsilon - 1}{\varepsilon} \right) \\ A_t N_t =& C_t + \frac{\omega}{2} \left(\pi_t - 1\right)^2 \\ \ln A_t =& \rho \ln A_{t-1} + \epsilon_t \end{aligned}$$

Note that the equation determining the nominal interest rate is missing.

3.1 Example 1: providing the analytical, recursive steady state

```
var pai, c, n, r, a;
varexo u;
parameters beta, rho, epsilon, omega, phi, gamma;
```

```
beta=0.99;
gamma=3;
omega=17;
epsilon=8;
phi=1;
rho=0.95;

model;
a = rho*a(-1)+u;
1/c = beta*r/(c(+1)*pai(+1));
pai*(pai-1)/c = beta*pai(+1)*(pai(+1)-1)/c(+1)
                +epsilon*phi*n^(gamma+1)/omega
                -exp(a)*n*(epsilon-1)/(omega*c);
exp(a)*n = c+(omega/2)*(pai-1)^2;
end;

initval;
r=1;
end;


steady_state_model;
a = 0;
pai = beta*r;
c = find_c(0.96,pai,beta,epsilon,phi,gamma,omega);
n = c+(omega/2)*(pai-1)^2;
end;

shocks;
var u; stderr 0.008;
end;

planner_objective(ln(c)
             -phi*((n^(1+gamma))/(1+gamma)));

ramsey_policy(planner_discount=0.99,order=1,instruments=(r));
```

In this model, given a guess value for r, it is not possible to find c or n, analytically. It is necessary to solve for c (or for n) numerically. We use the following auxiliary Matlab function **find_c.m**

```
function c = find_c(c0,pai,beta,epsilon,phi,gamma,omega)
    c = csolve(@nk_ss,c0,[],1e-8,100,pai,beta,epsilon,phi,gamma,omega);

function r = nk_ss(c,pai,beta,epsilon,phi,gamma,omega)
    r = pai*(pai-1)/c - beta*pai*(pai-1)/c ...
```

```
                -epsilon*phi*(c+(omega/2)*(pai-1)^2)^(gamma+1)/omega ...
                +(c+(omega/2)*(pai-1)^2)*(epsilon-1)/(omega*c);
```

Note that for $pai = 1$, the steady state can be solved for analytically, but this requires to know the optimal value of inflation in this model. In more complicated models, it may not be the case. Also, still for the sake of example, we use a guess value for r that is away from its optimal value.

3.2 Example 2: providing guess values for the steady state

```
var pai, c, n, r, a;
varexo u;
parameters beta, rho, epsilon, omega, phi, gamma;

beta=0.99;
gamma=3;
omega=17;
epsilon=8;
phi=1;
rho=0.95;

model;
a = rho*a(-1)+u;
1/c = beta*r/(c(+1)*pai(+1));
pai*(pai-1)/c = beta*pai(+1)*(pai(+1)-1)/c(+1)
                +epsilon*phi*n^(gamma+1)/omega
                -exp(a)*n*(epsilon-1)/(omega*c);
exp(a)*n = c+(omega/2)*(pai-1)^2;
end;

initval;
pai=1;
r=1;
c=0.9;
n=0.9;
a=0;
end;

shocks;
var u; stderr 0.008;
end;

planner_objective(ln(c)
             -phi*((n^(1+gamma))/(1+gamma)));
```

```
ramsey_policy(planner_discount=0.99,order=1);
```

In this small example, the second approach is simpler. However, this approach may encounter problems in larger models.