A Practical Introduction to DSGE Modeling with Dynare

Callum Jones*
New York University

Mariano Kulish†
University of New South Wales

This document is a practical introduction to Dynare. It shows how to install Dynare and write a DSGE model in Dynare notation, and goes through the output from running a simple DSGE model, discussing where the output is stored in the Matlab workspace and common Dynare errors. We use Dynare to do some useful analysis. We briefly discuss estimation and forecasting using Dynare. The document closes with a research application.

Contents

*Corresponding author. Department of Economics, New York University. Email: `callum.jones@nyu.edu`. All remaining errors are mine. Date: January 26, 2016.

†School of Economics, University of New South Wales.

1 Dynare: Installing

Dynare is a collection of Matlab routines. It reads in a system of DSGE model equations, solves/estimates the model for a given set of parameter values and delivers useful output.

To get Dynare, go to `http://www.dynare.org/download` and download the latest stable version: the file should be `dynare-*-win.exe`, where `*` is the version number. Install it by running the `.exe` file. Let it install to its default folders, typically `C:\dynare\*`, where `*` is the version number. All the Matlab routines needed to run Dynare are in `C:\dynare\*\matlab`.

After installing Dynare, Matlab needs to be directed to recognise Dynare files. That is, the path of the Dynare Matlab files needs to be set in Matlab. In Matlab, click `File` → `SetPath`, then click `Add Folder...`, and choose as the folder, `C:\dynare\*\matlab`. All the dynare files should now appear in the `MATLAB search path`, as in Figure 1. Now click `Save` and `Close`.

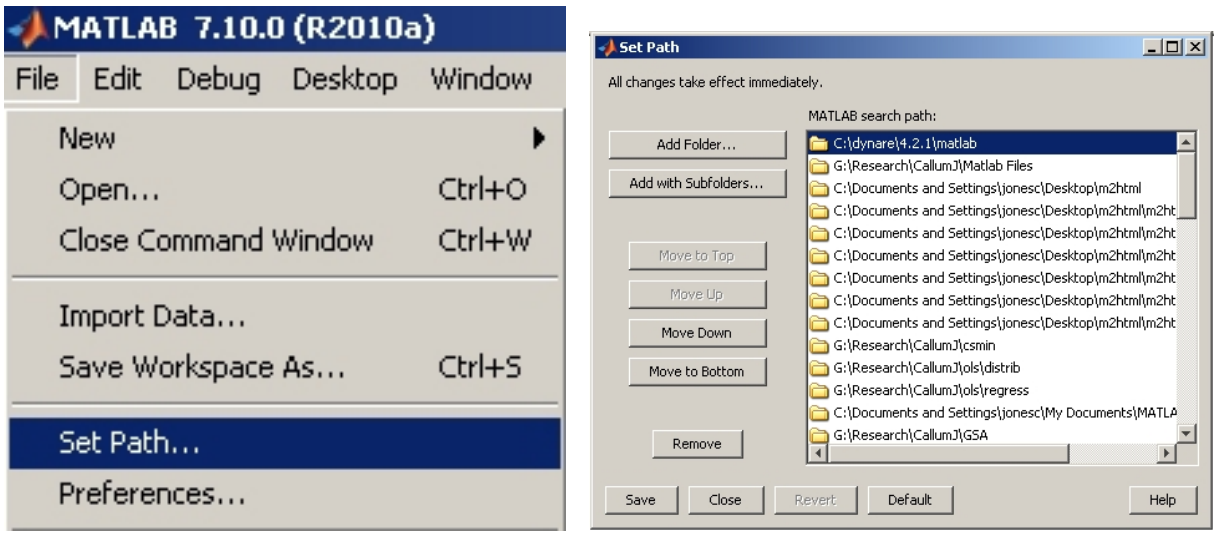
To test if Dynare has installed properly, type `dynare` into the Matlab command line. If it is installed correctly, you should get the error message `??? Input argument "fname" is undefined.` If not installed correctly, you should get the error message `??? Undefined function or variable 'dynare'.`

2 A model in Dynare

Dynare reads in `.mod` text files, parses these into Matlab files, which then call on the Dynare Matlab routines to do the model analysis. In this section, we present a model's linearized system of equations and show how to write the model in a Dynare consistent `.mod` file.

The model is Ireland (2004). Using his notation, the equations of the model are:

$$\hat{x}_t = E_t\hat{x}_{t+1} - (\hat{r}_t - E_t\hat{\pi}_{t+1}) + (1-\omega)(1-\rho_a)\hat{a}_t \tag{1}$$



(a) Set Path (b) After Setting

Figure 1: **Setting the Matlab Path to Dynare**

$$\hat{\pi}_t = \beta E_t \hat{\pi}_{t+1} + \psi \hat{x}_t - \hat{e}_t \tag{2}$$

$$\hat{r}_t = \hat{r}_{t-1} + \rho_\pi \hat{\pi}_t + \rho_g \hat{g}_t + \rho_x \hat{x}_t - \varepsilon_{r,t} \tag{3}$$

$$\hat{x}_t = \hat{y}_t - \omega \hat{a}_t \tag{4}$$

$$\hat{g}_t = \hat{y}_t - \hat{y}_{t-1} + \hat{z}_t \tag{5}$$

$$\hat{a}_t = \rho_a \hat{a}_{t-1} + \varepsilon_{a,t} \tag{6}$$

$$\hat{e}_t = \rho_e \hat{e}_{t-1} + \varepsilon_{e,t} \tag{7}$$

$$\hat{z}_t = \varepsilon_{z,t} \tag{8}$$

Equation (1) is the IS-curve, Equation (2) is the Phillips curve, Equation (3) is the Taylor Rule, Equation (4) defines the output gap, Equation (5) defines output growth, and (6), (7) and (8) define the demand, cost and technology shocks respectively. For a full derivation, see Ireland or his comprehensive notes.

We want to write the model into a form which Dynare can interpret. In the first part of the Dynare code, we need to define the endogenous variables which appear in the model. There are eight equations, so we have eight variables: $\hat{y}_t$, $\hat{x}_t$, $\hat{g}_t$, $\hat{r}_t$, $\hat{\pi}_t$, $\hat{a}_t$, $\hat{e}_t$, $\hat{z}_t$. The syntax for writing this in Dynare is `var [variable1, variable2, ... ];` with the line ended by a semicolumn. For Ireland's model we write:

```
var y, x, g, r, pi, a, e, z ;
```

Next we write the exogenous variables in the model. In Ireland's model, there are four shocks: $\varepsilon_{r,t}$, $\varepsilon_{a,t}$, $\varepsilon_{e,t}$ and $\varepsilon_{z,t}$. The syntax for writing this in Dynare is `varexo [variable1, variable2, ... ];` with the line ended by a semicolumn. For Ireland's model we write:

```
varexo eps_r, eps_a, eps_e, eps_z ;
```

Next we write the parameters of the model. In Ireland's model, there are eight structural parameters plus the four standard deviations for the four shocks, giving 12 parameters. The syntax for writing this in Dynare is `parameters [parameter1, parameter2, ... ];` with the line ended by a semicolumn. For Ireland's model we write:

```
parameters beta, omega, psi, rho_pi, rho_g, rho_x, rho_a, rho_e,
           sig_r, sig_z, sig_a, sig_e ;
```

Finally, we give the parameters values. Ireland estimated most of his parameters by maximum likelihood on quarterly US data from 1947. The syntax for giving parameters values is `[parameter] = [value];` which is similar to normal Matlab code. For Ireland's model we write:

```
beta   = 0.99 ;
psi    = 0.1 ;
omega  = 0.0617 ;
rho_pi = 0.3597 ;
rho_g  = 0.0000 ;
rho_x  = 0.0347 ;
rho_a  = 0.9470 ;
rho_e  = 0.9625 ;
sig_a  = 0.0405 ;
sig_e  = 0.0012 ;
sig_z  = 0.0109 ;
sig_r  = 0.0031 ;
```

We now write the equations of the linearized model. The equations are written in generally the same way as they appear in the paper. The equation section starts with `model (linear) ;` and ended by `end ;`. A few syntax considerations:

1. Variables which are multiplied (divided) with a parameter are separated by the parameter with $*$ ($/$).

2. Contemporaneous variables do not have a time subscript in the Dynare code. Lagged variables x_{t-i} are written as `x(-i)`, while expectational variables $E_t x_{t+i}$ are written as `x(+i)`.

3. Equations are ended by a semicolumn `;`.

The equations in Dynare syntax then are:

```
model (linear) ;
    x  = x(+1) - ( r - pi(+1) ) + ( 1 - omega ) * ( 1 - rho_a ) * a ;
    pi = beta * pi(+1) + psi * x - e ;
    r  = r(-1) + rho_pi * pi + rho_g * g + rho_x * x + eps_r ;
    x  = y - omega * a ;
    g  = y - y(-1) + z ;
    a  = rho_a * a(-1) + eps_a ;
    e  = rho_e * e(-1) + eps_e ;
    z  = eps_z ;
end ;
```

After specifying the equations, we need to tell Dynare where to begin simulations or impulse responses: the initial values of the variables. We do this in a similar way to specifying the parameter values, beginning the section with `initval ;` and ending the section with `end ;` and setting the variable to a value. Usually the initial values correspond to the steady-state of the model, particularly for linearized models. For Ireland's model, all the variables enter the Dynare routines as deviations from their steady-state values, and so the steady-state of the model variables is zeros. In this case, instead of setting each variable to a initial value, in Dynare, we can write `steady ;` which begins the simulation from the model's steady-state.

After specifying the equations, we define the shocks, and set the shocks' variances to the specified or estimated parameters. We begin the section with a `shocks ;` and end it with `end ;`. The syntax for specifying each shock in Dynare is `var [shock] = [shock variance] ;`. For Ireland's model, the section is:

```
shocks ;
    var eps_a = sig_a^2 ;
    var eps_e = sig_e^2 ;
    var eps_z = sig_z^2 ;
    var eps_r = sig_r^2 ;
end ;
```

Finally, we solve the model and obtain Dynare's default analytics. We use the `stoch_simul` command without specifying any further options. There are a lot of options which can be used with `stoch_simul`, which can be found in the Dynare manual. For now we'll consider the default output. The final line of the Ireland Dynare `.mod` file is then:

```
stoch_simul ;
```

This is sufficient to run Dynare, solve the model and do some analysis. The full `.mod` file is given in the Appendix.

3 Running Dynare and its output

To run the Ireland Dynare `.mod` file created, save the text file with a `.mod` extension in Matlab's current directory; for example, saving the file as `ireland.mod`. To initiate Dynare, in Matlab's command line, type

```
>> dynare ireland.mod
```

You will then see Dynare at work. What it's doing is reading in the `.mod` contents, recognising the model's details, and solves for the linear rational expectations solution. Dynare will print a lot of analytical information to the Matlab command window, save much of that to the workspace, and generate figures which plot impulse responses to the shocks for each variable. It also saves everything that it generates to the working directory.

The first bit of printed output Dynare generates is processing information. The `[mex]` lines say that it found `.mex` files – Matlab executables that do some computations more efficiently in non-Matlab languages. It is worth checking that the number of equations Dynare finds in the `.mod` file is the same as the number of endogenous variables in the model. For Ireland's model, there are 8 endogenous variables.

```
Configuring Dynare ...
[mex] Generalized QZ.
[mex] Sylvester equation solution.
[mex] Kronecker products.
[mex] Sparse kronecker products.
[mex] Bytecode evaluation.
[mex] k-order perturbation solver.
[mex] k-order solution simulation.

Starting Dynare (version 4.2.0).
Starting preprocessing of the model file ...
Found 8 equation(s).
Evaluating expressions...done
Computing static model derivatives:
 - order 1
Computing dynamic model derivatives:
 - order 1
 - order 2
Processing outputs ...done
Preprocessing completed.
Starting MATLAB/Octave computing.
```

Next, Dynare will print steady-state values for all the variables, and a summary of the model variables.

```
STEADY-STATE RESULTS:

y    0
x    0
g    0
r    0
pi   0
a    0
e    0
z    0

MODEL SUMMARY

  Number of variables:         8
  Number of stochastic shocks: 4
  Number of state variables:   4
  Number of jumpers:           2
  Number of static variables:  2
```

Next, Dynare generates the variance-covariance matrix of the shocks. The values in the diagonals correspond to the specified values of the `eps` values in the `.mod` file.

```
MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS

Variables    eps_r     eps_a     eps_e     eps_z
eps_r        0.000010  0.000000  0.000000  0.000000
eps_a        0.000000  0.001640  0.000000  0.000000
eps_e        0.000000  0.000000  0.000001  0.000000
eps_z        0.000000  0.000000  0.000000  0.000119
```

Dynare computes the solution of the model as:

$$\hat{y}_t = A\hat{y}_{t-1} + Bu_t$$

where $\hat{y}_t$ is a vector of endogenous variables as a deviation from steady-state, $\hat{y}_{t-1}$ is the deviation of y_{t-1} from steady-state, and u_t is a vector of the exogenous shocks. The first four lines of the below output correspond to matrix A^T, while the last four rows correspond to matrix B^T.

```
POLICY AND TRANSITION FUNCTIONS
        y           x           g           r           pi          ~
y(-1)   0           0           -1.000000   0           0           ~
r(-1)   -2.923646   -2.923646   -2.923646   0.623664    -0.764207   ~
a(-1)   0.182823    0.124393    0.182823    0.014178    0.027416    ~
e(-1)   5.939550    5.939550    5.939550    -0.384804   -1.642775   ~
eps_r   -2.923646   -2.923646   -2.923646   0.623664    -0.764207   ~
eps_a   0.193055    0.131355    0.193055    0.014972    0.028951    ~
eps_e   6.170961    6.170961    6.170961    -0.399796   -1.706780   ~
eps_z   0           0           1.000000    0           0           ~

                                ~ a           e             z
                                ~ 0           0             0
                                ~ 0           0             0
                                ~ 0.947000    0             0
                                ~ 0           0.962500      0
                                ~ 0           0             0
                                ~ 1.000000    0             0
                                ~ 0           1.000000      0
                                ~ 0           0             1.000000
```

The next section of Dynare output reports the moments of the model, using the model solution and the specified standard errors of the exogenous shocks.

```
THEORETICAL MOMENTS

VARIABLE    MEAN        STD. DEV.   VARIANCE
y              0.0000      0.0436      0.0019
x              0.0000      0.0422      0.0018
g              0.0000      0.0186      0.0003
r              0.0000      0.0070      0.0000
pi             0.0000      0.0056      0.0000
a              0.0000      0.1261      0.0159
e              0.0000      0.0044      0.0000
z              0.0000      0.0109      0.0001
```

Next, Dynare reports the variance decomposition of the endogenous variables. It gives the proportion of the variance in the endogenous variable which can be attributed to each of the exogenous shocks, ie the sum for each variable across the four shocks is 100%. For example, for Ireland's

model at his calibrated/estimated parameter values, 47% of output growth is due to shocks to permanent technology $\varepsilon_{z,t}$.

```
VARIANCE DECOMPOSITION (in percent)

       eps_r   eps_a   eps_e   eps_z
y       7.08    8.67   84.25    0.00
x       7.55    2.53   89.91    0.00
g      29.12   20.00   16.69   34.19
r      12.42   45.99   41.60    0.00
pi     29.43    7.03   63.54    0.00
a       0.00  100.00    0.00    0.00
e       0.00    0.00  100.00    0.00
z       0.00    0.00    0.00  100.00
```

Dynare then reports a matrix of theoretical correlations across the endogenous variables. These values are constructed from the variance-covariance matrix of the endogenous variables, which Dynare does not output to the command window.

```
MATRIX OF CORRELATIONS

Variables    y        x        g        r        pi       ~
y            1.0000   0.9841   0.1407  -0.5297  -0.4875   ~
x            0.9841   1.0000   0.1393  -0.6683  -0.5140   ~
g            0.1407   0.1393   1.0000  -0.1070   0.0608   ~
r           -0.5297  -0.6683  -0.1070   1.0000   0.2581   ~
pi          -0.4875  -0.5140   0.0608   0.2581   1.0000   ~
a            0.2656   0.0900   0.0329   0.6569   0.0561   ~
e            0.9121   0.9422   0.0800  -0.6303  -0.7689   ~
z            0.0000   0.0000   0.5847   0.0000   0.0000   ~

                                      ~ a        e        z
                                      ~ 0.2656   0.9121   0.0000
                                      ~ 0.0900   0.9422   0.0000
                                      ~ 0.0329   0.0800   0.5847
                                      ~ 0.6569  -0.6303   0.0000
                                      ~ 0.0561  -0.7689   0.0000
                                      ~ 1.0000   0.0000   0.0000
                                      ~ 0.0000   1.0000   0.0000
                                      ~ 0.0000   0.0000   1.0000
```

Next, Dynare shows coefficients of autocorrelation up to the fifth order, that is, the correlation of an endogenous value with p-order lags of itself, where $p = \{1, \ldots, 5\}$.

```
COEFFICIENTS OF AUTOCORRELATION

Order    1       2       3       4       5
y        0.9398  0.8902  0.8476  0.8099  0.7756
x        0.9454  0.8993  0.8589  0.8226  0.7891
g       -0.0578 -0.0386 -0.0265 -0.0189 -0.0140
r        0.9438  0.8942  0.8493  0.8081  0.7697
pi       0.7871  0.6483  0.5558  0.4925  0.4476
a        0.9470  0.8968  0.8493  0.8043  0.7616
e        0.9625  0.9264  0.8917  0.8582  0.8260
z        0.0000  0.0000  0.0000  0.0000  0.0000
Total computing time : 0h00m13s
```

Finally, Dynare will generate a number of impulse response graphics for the response of the endogenous variables to each of the exogenous shocks. Dynare filters out impulses which are essentially zero across the horizon. On each Matlab figure graphic, it will plot at most 9 impulse responses; if the model has more than 9 endogenous variables, it will plot the 10th+ variables on subsequent Matlab figure graphs. For Ireland's model, the impulse responses are given in Figure 6. Confirm that the number of impulse responses aligns with the number of zero entries in the B matrix of the policy and transition function output above.

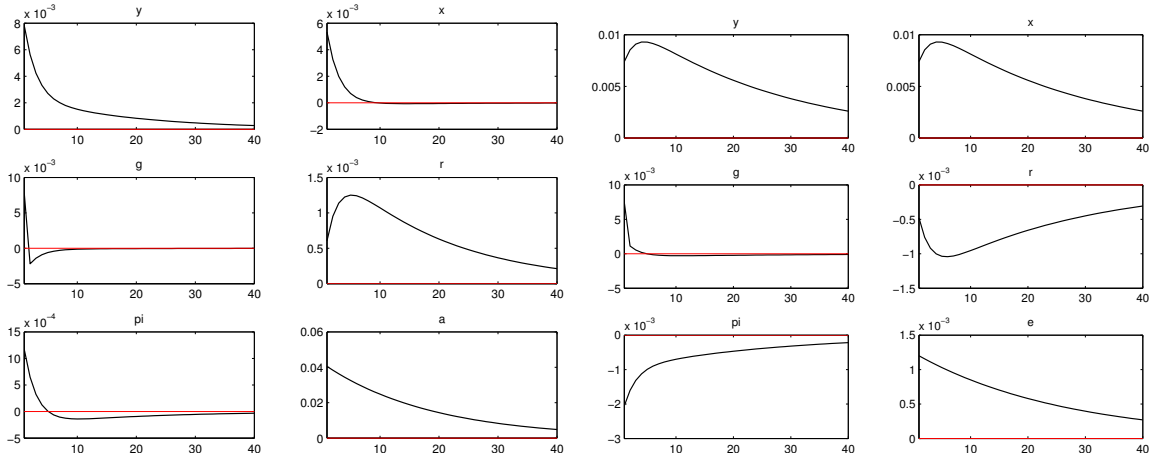
3.1 The Matlab workspace

Table 1 gives information on some of what is stored by Dynare to the Matlab workspace. More generally, Dynare stores to the workspace all the parameter values and impulse responses of all endogenous variables to shocks.[1]

3.2 More on the solution output

Here, it's worth understanding how Dynare orders the variables of the model in the computed solution matrices. Dynare distinguishes the endogenous variables by their type, that is, whether they fall into one of the following categories:

[1] A note on Matlab data structures. Matlab can store information in `struct` objects, which consist of fields, which can be any compatible Matlab item, like a matrix, string, another structure, etc. To call a field of a structure from the command line, type `struct.field`. Most of Dynare's important output is stored in structures.



(a) Shock to ε_a (b) Shock to ε_e

(c) Shock to ε_r (d) Shock to ε_z

Figure 2: **Dynare generated impulse responses**

- Predetermined variables: Those that appear in the model with only current or past period timing. In Ireland's model, there are 4 predetermined variables: y, r, a and e. The number of these appears in `oo_.dr.npred`.

- Purely forward looking variables: Those appearing at current and future periods. In Ireland's model, there are two purely forward looking variables: π, and x. The number of these appears in `oo_.dr.nfwrd`.

- Mixed variables: Those variables with both backward, current and forward looking components. There are none of these in Ireland's model. The number of these appears in `oo_.dr.nboth`.

- Static variables: Those who only appear at current timing. There are two in Ireland's model: g and z. The number of these appears in `oo_.dr.nstatic`.

The DR ordering is: static variables, then predetermined variables, then mixed variables, then

Table 1: **The Matlab Workspace**

Stored in	What
`oo_.exo_simul`	Simulated series of the exogenous variables (if simulating).
`oo_.endo_simul`	Simulated series of the endogenous variables (if simulating).
`oo_.dr`	Stored information about variables and policy function.
`oo_.dr.order_var`	The mapping from the declaration order to the decision rule order.
`oo_.dr.inv_order_var`	The mapping from the decision rule order to the declaration order.
`oo_.dr.ghx`	The A matrix of the transition function. More on this in Section 3.2.
`oo_.dr.ghu`	The B matrix of the transition function. More on this in Section 3.2.
`oo_.steady_state`	Values of the endogenous variables' steady-state.
`oo_.gamma_y{7,1}`	The variance decomposition outputted to the command window.
`oo_.mean`	The mean of endogenous variables.
`oo_.var`	The variance-covariance matrix.
`oo_.autocorr`	The autocorrelation matrices for endogenous variables; to 5 lags.
`oo_.irfs`	Impulse responses of endogenous variables to exogenous shocks.
`oo_.irfs.x_y`	Impulse response of `x` to an innovation in `y`.
`oo_.forecast`	Forecasts of endogenous variables.
`M_`	Structure with information about the model.
`M_.endo_names`	The names of endogenous variables in declaration order.
`M_.exo_names`	The names of exogenous variables in declaration order.
`M_.exo_nbr`	The number of exogenous variables.
`M_.endo_nbr`	The number of endogenous variables.
`options_`	Structure with all the options which Dynare recognises.

purely forward looking variables. The mapping from declaration order in the `var` section to DR order is given in `oo_.dr.order_var`. The mapping backwards from DR order to declaration order is `oo_.dr.inv_order_var`. That is, suppose your vector of endogenous variables is $\mathbf{y}_t$ and these variables are ordered in declaration order, then to rearrange the vector into DR order, write `y_dr = y(oo_.dr.order_var,:)`. To reverse the mapping from the DR order to declaration order, write `y_dec = y_dr(oo_.dr.inv_order_var,:)`. Check that `y_dr` is the same as `y_dec`.

To see why the ordering is relevant, consider the solution which was computed as:

$$\hat{\mathbf{y}}_t = A\hat{\mathbf{y}}_{t-1} + B\mathbf{u}_t$$

A appears in `oo_.dr.ghx` and B appears in `oo_.dr.ghu`. Dynare calculates the matrices so that the variables in each row are in DR order. The columns of A correspond to the state variables. The mapping of these state variables from the declaration order can be found in the following vector:

```
oo_.dr.order_var(oo.dr.nstatic+1 : oo_.dr.nstatic+oo_.dr.npred)
```

These variables are the same as the lagged variables which appear in the policy and transition

functions as part of Dynare output.

Putting this all together, suppose we want to manually calculate, using the solution matrices, the impulse response of all endogenous variables to a one standard deviation policy shock for 20 quarters, and have the output correspond to the declaration order of the variables. We can do this with the following code. The plot of the non-zero variables is in Figure 3. It should be the same as Figure 6.

```
horizon = 20 ;
shocks  = zeros(M_.exo_nbr,horizon) ;
irf     = zeros(M_.endo_nbr,horizon) ;

shocks(1,1) = sig_r; % Policy shock in period 1

irf(:,1) = oo_.dr.ghu(oo_.dr.inv_order_var,:)*shocks(:,1) ;

for t=2:horizon
  irf(:,t) = oo_.dr.ghx(oo_.dr.inv_order_var,:) * ...
    irf(oo_.dr.order_var( ...
    oo_.dr.nstatic+1:oo_.dr.nstatic+oo_.dr.npred),t-1) ...
    + oo_.dr.ghu(oo_.dr.inv_order_var,:)*shocks(:,t) ;
end
```

4 Some common Dynare error messages

Unfortunately, Dynare has a reputation for reporting ambiguous error messages. Here are some common errors in writing the `.mod` file, and the error messages that Dynare reports when it is executed. Note that Dynare will not pick up logical errors, such as errors in transcribing the linearized equations into the `.mod` file.

4.1 Parsing errors

In general, if Dynare comes across a parsing error, it will output the following message to the Matlab command window:

```
ERROR: ireland.mod:10.1

??? Error using ==> dynare at 126
DYNARE: preprocessing failed
```

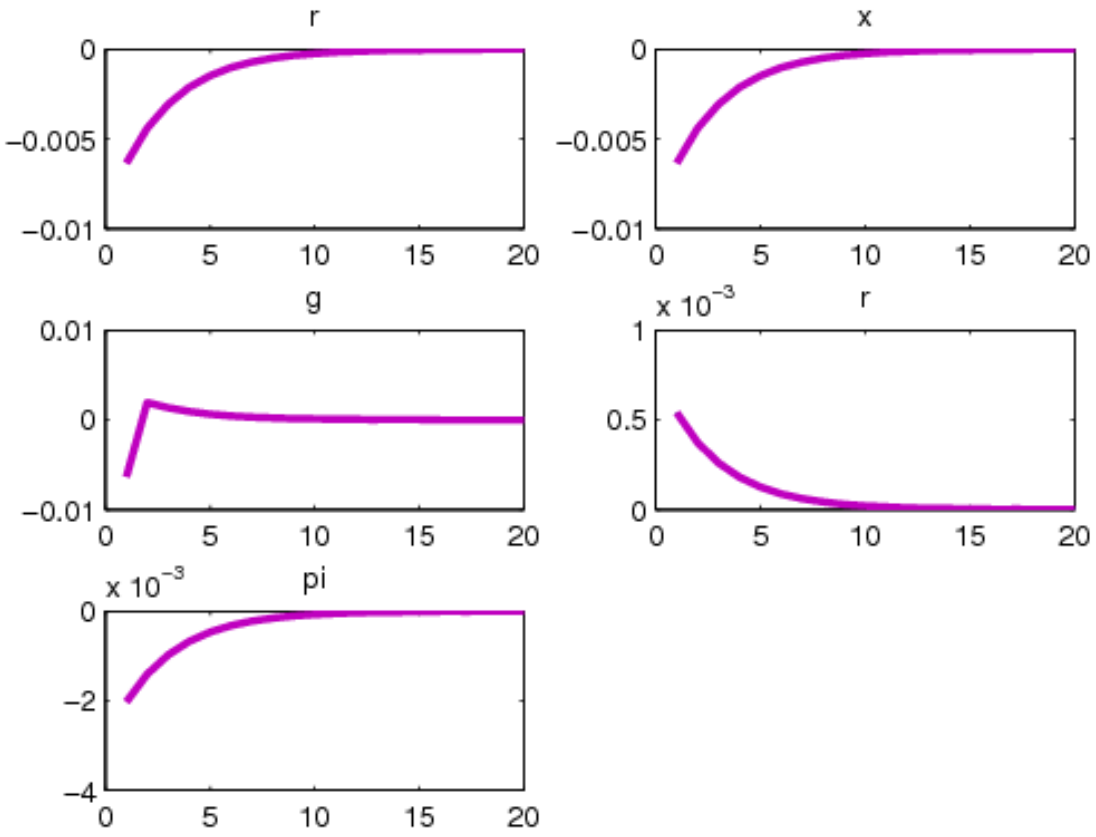


Figure 3: **Manual IRF to Policy Shock**

In the `ERROR: ireland.mod:10.1` part of the message, the number `10.1` indicates there was a parsing error on line 10, column 1 of the Dynare `.mod` file.

The following error may appear if a variable or parameter is omitted from the `var [.]` declaration, but it appears in the model equations. Here, I forgot to include `z`.

```
ERROR: ireland.mod:25.26: Unknown symbol: z
```

This error may also appear when an `end ;` line has been omitted after the model section. If `end ;` is omitted after the `shocks` section, you may get, because Dynare is trying to read in a line `stoch_simul` which does not accord with the `shocks ;` nomenclature:

```
ERROR: ireland.mod:40.1-11: syntax error, unexpected NAME,
    expecting CORR or END or VAR
```

The `unexpected NAME` error may also appear when Dynare encounters an option it does not understand, for example, by mistakenly having the option `nograp` instead of `nograph`. The `unexpected NAME` may also appear if, commonly, a mathematical operator is omitted in the declaration of the model equations, or if lines are not separated by `;`. For example, if a `*`, `-`, `+` or `/` operator is missing.

If less (or more) equations appear than declared endogenous variables, Dynare will recognise this and shriek back the identification error:

```
ERROR: There are 7 equations but 8 endogenous variables!
```

4.2 Declaration errors

If a parameter is not declared to have a value in the `.mod` file, you may see the following error message, which indicates the parsing was done successfully, but the model could not be solved, because the matrices which describe the model cannot be constructed. This could occur because a parameter is misspelled in the parameter setting section. Here, I did not give `beta` a value.

```
Warning: Some of the parameters have no value (beta) when using
stoch_simul. If these parameters are not initialized in a
steadystate file, Dynare may not be able to solve the model...


??? Error using ==> print_info at 36
MJDGGES returns the following error code: 6
```

If parameters are declared which are inconsistent with an unique equilibrium, Dynare reports that the Blanchard Kahn conditions are not met. For example, I declared the parameter values `rho_pi = 0`, `rho_g = 0` and `rho_x = 0` which I know gives a policy rule that does not satisfy the Taylor principle and implies that there may be multiple equilibrium paths. Dynare reports:

```
??? Error using ==> print_info at 42
Blanchard Kahn conditions are not satisfied: indeterminacy
```

The Blanchard Kahn conditions may not be met if parameters are specified such that there is no equilibrium. For example, I declared that the autoregressive parameter on a temporary technology shock `rho_a = 1.1` implying that every solution to the model will have an explosive path. Dynare reports:

```
??? Error using ==> print_info at 39
Blanchard Kahn conditions are not satisfied: no stable equilibrium
```

5 Using Dynare for basic analysis

5.1 Simulating series

Dynare can easily use the model to simulate series for the endogenous variables. To simulate 200 observations, replace `stoch_simul` with `stoch_simul(periods=200)`. Dynare will generate 200 shocks of the exogenous variables, and use the model solution to generate the endogenous

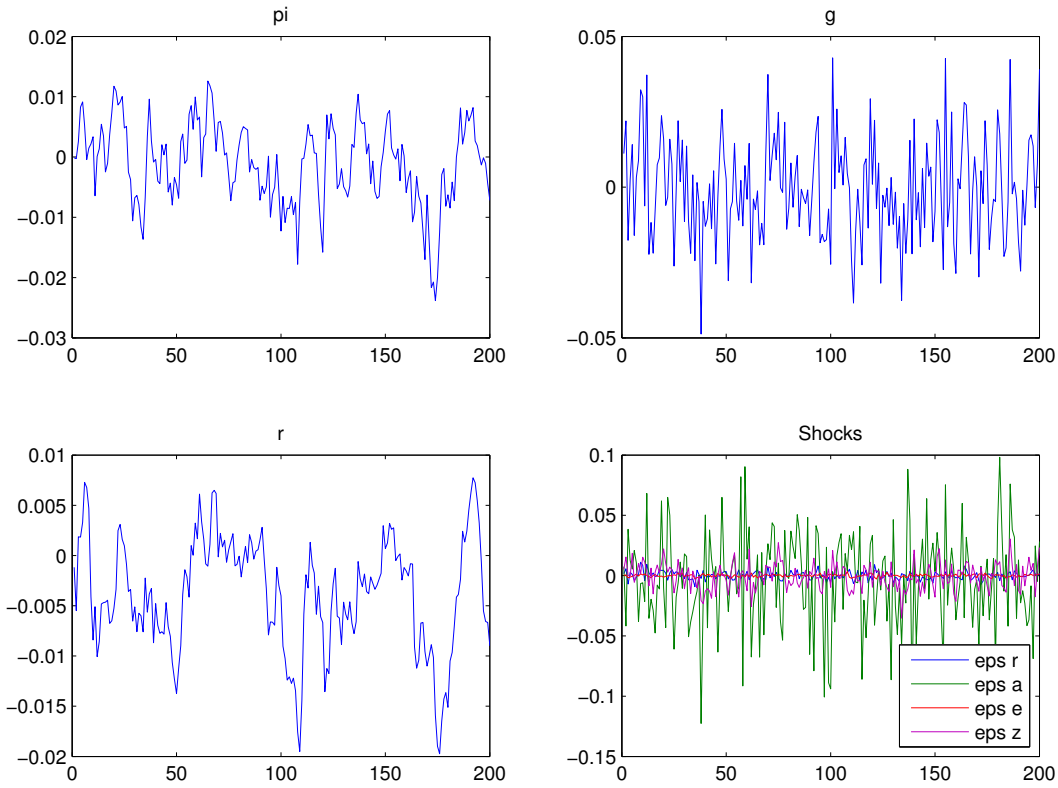


Figure 4: **Simulation**

variables. Figure 4 graphs the simulated shocks and simulated π_t, g_t and r_t. Compare the size of the shocks with the specified parameters.

The simulated endogenous variables are stored in the workspace under the name of the variable. For example, for Ireland's model, simulated inflation series will be stored under `pi`. The simulated endogenous variables are also stored under `endo_simul`, in the order of declaration in the `.mod` file (ie in the order in `M_.endo_names`). The generated shocks are stored in `exo_simul`, in the order of declaration (ie in the order in `M_.exo_names`).

The moments, correlations and autocorrelations are calculated from the simulated data. By default, Dynare drops the first 100 observations. This means that the number of periods needs to be larger than 100. Importantly, newer versions of Dynare set the random number generator seed by default, so the same simulated variables are generated with each run. A different seed can be manually set with the command `set_dynare_seed(x)` where `x` is an integer, declared before `stoch_simul` in the `.mod` file.

5.2 Looping over `stoch_simul`

Typically, we want to simulate many series from the same model, or to see how a model's analytics react to changes in parameter values. A natural way to do such analysis is with a loop control structure. Dynare allows us to combine Matlab's control structures with Dynare's functions directly in the Dynare `.mod` file, after we define the model.

For example, suppose we wanted to simulate 200 series with 200 observations of output growth $\hat{g}_t$ and inflation $\hat{\pi}_t$ from Ireland's model. That is, we would want to run `stoch_simul` 200 times, and save the simulated series in a matrix where the column number represents the simulation run. We use the following block of code:

```
for i=1:200
    set_dynare_seed(i);
    stoch_simul(periods=200, noprint, nograph);
    g_sim(:,i)  = g;
    pi_sim(:,i) = pi;
end
```

`i` here is a looping variable which we use to set different seeds and index the `g_sim` and `pi_sim` matrices. Beginning at `i=1`, solve the model and simulate the endogenous variables for 200 periods by `stoch_simul`, and store `g` in the first column of `g_sim` and correspondingly for `pi`. After doing this, it increases `i` by one and repeats, until `i=200`. For speed reasons, the `noprint` and `nograph` options for `stoch_simul` suppresses graphs of impulse responses and the printing of model information to the command window.

To loop over a range of possible parameter values - possible in that we get a unique solution to the model - use the same strategy as for simulating data. Suppose we want to see how the impulse response of inflation to a monetary policy shock changes across the degree of price adjustment costs. That is, we want to analyse `pi_eps_a` across different values of `psi`where $\psi = (\theta - 1)/\phi$ with ϕ governing price adjustment costs. We define a vector of possible `psi` parameter values and employ the looping strategy we used for simulation above:

```
psi_params = 0:0.05:1;

for i=1:length(psi_params)
    psi = psi_params(i);
    stoch_simul(noprint, nograph);
    pi_eps_a_mat(:,i) = pi_eps_a;
end
```

The possible `psi` parameters are stored in `psi_params`, and includes values from 0 to 1 in steps of 0.1, giving a 1×21 vector where `length(psi_params) = 21`. We use the looping variable to

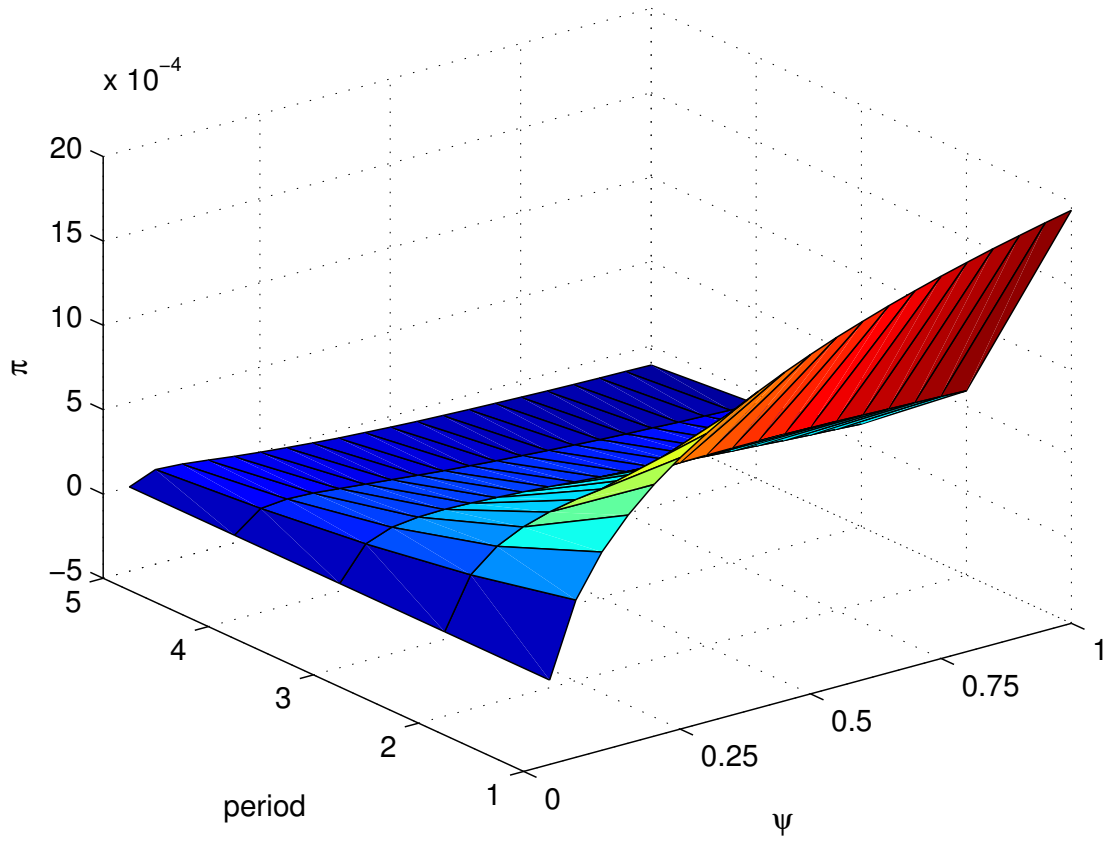


Figure 5: **Impulse Responses Over** `psi`

index the vector of possible `psi` parameters and to index to a corresponding column in a matrix which stores the impulse responses. The `psi` parameter is chosen from the vector, `stoch_simul` is called, and the impulse response `pi_eps_a` stored.

To strengthen the intuition behind the impulse responses, it is useful to graphically view the effect of changes in parameter values over an interesting horizon by drawing a 3D graph. In Matlab, we can use the following command:

```
>> surf(pi_eps_a_mat(1:5,:)); xlabel('\psi'); ylabel('t'); zlabel('\pi');
```

This will plot the magnitude of each value in the matrix of stored impulse responses across 5 time periods for every parameter value in `psi_params`. The resulting graph is shown in Figure 5. In terms of the model, it shows that firms have a greater incentive to raise prices in response to the demand shock when the cost of adjustment is low (when `psi` is high).

6 Estimation with Dynare

Dynare is very useful for estimating parameters and shocks with data. Here, we just show how to practically estimate the model. Note, to estimate a model, the number of data series cannot be greater than the number of shocks in the model.

6.1 Maximum likelihood

To estimate using maximum likelihood techniques, first declare the variables of the model which are observed. This is done using the `varobs VAR ;` command. For Ireland's model, it is sensible to use output growth, inflation and the policy interest rate as observed series:

```
varobs g pi r;
```

Next, we specify the parameters to be estimated in the `estimated_params` block. All variables that are not estimated need to be given values in the parameter setup section of the `.mod` file. When writing the `estimated_params` section, each parameter to be estimated needs to be given an initial value, and can be given bounds: the syntax to do this is `parameter, initial value [, lower bound, upper bound ] ;` where the parts in the square brackets are optional. For the standard errors of the shocks, we use the same syntax, preceding the exogenous variable with `stderr`, like for specifying the shocks. The full `estimated_params` section is:

```
estimated_params ;
    rho_a, 0.9, 0, 1 ;
    rho_e, 0.9, 0, 1 ;
    omega, 0.1, 0, 1 ;
    rho_pi, 0.4 ;
    rho_g, 0.3 ;
    rho_x, 0.05 ;
    stderr eps_a, 0.1, 0, 1 ;
    stderr eps_e, 0.1, 0, 1 ;
    stderr eps_z, 0.1, 0, 1 ;
    stderr eps_r, 0.1, 0, 1 ;
end ;
```

Finally, we need to call estimation and specify where Dynare can find the file containing the observed series. This is called with:

```
estimation(datafile=data) ;
```

6.2 Bayesian

When we are estimating with Bayesian techniques, we need to specify prior distributions for each parameter to be estimated. As for maximum likelihood, we first say which series are observed:

```
varobs g pi r;
```

In the `estimated_params` section, for each estimated parameter, we declare a distribution, and specify parameters relevant for that distribution. The syntax for this is `parameter, prior_shape, prior_`μ`, prior_`σ` [, p_3] [, p_4] ;`. Table 2 details the distributions, and to what parameters the Dynare declaration refer:

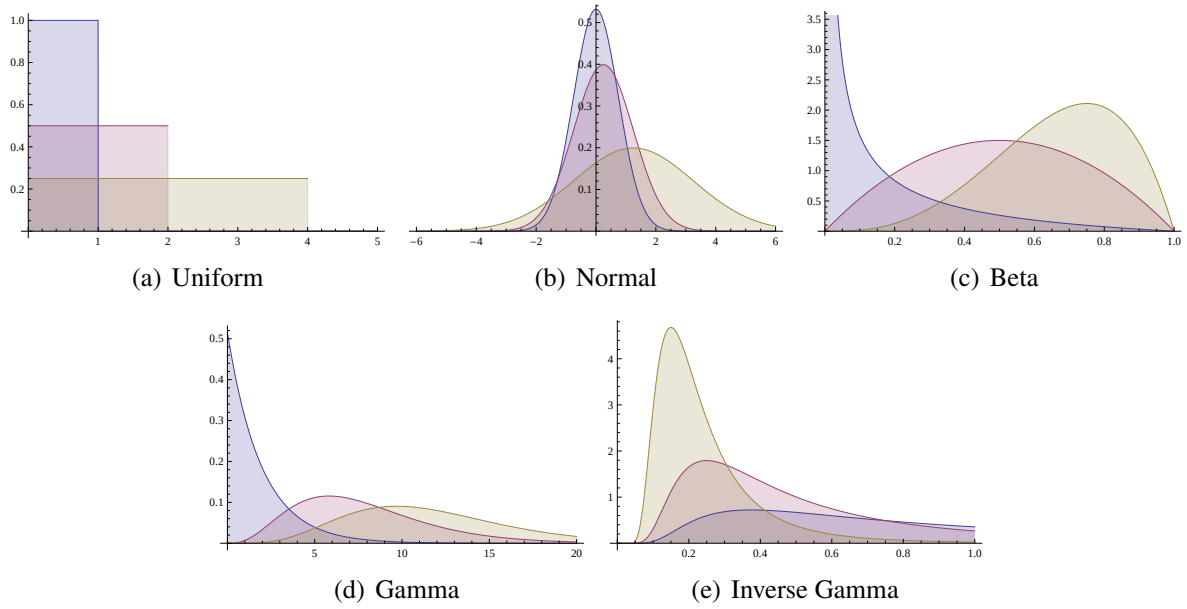
Table 2: **Bayesian Priors**

Shape	Distribution
`normal_pdf`	$N(\mu,\sigma)$
`gamma_pdf`	$G(\mu,\sigma,p_3)$
`beta_pdf`	$B(\mu,\sigma,p_3,p_4)$
`inv_gamma_pdf`	$IG(\mu,\sigma)$
`uniform_pdf`	$U(p_3,p_4)$

These distributions have very different shapes, and can depend a lot on what parameters are chosen. To illustrate this, Figure 6 shows the five distributions across different parameter choices. To play around with the shape of the distribution across parameters, try opening the Mathematica file here (if you have a Mathematica license). The point is, it is important to think about the distribution of the prior you want for your estimated parameters.

We illustrate the use of the declarations with Ireland's model below. Note the extra commas needed for the `uniform_pdf` to specify the third and fourth parameters.

```
estimated_params ;
    rho_a, beta_pdf, 0.7, 0.1 ;
    rho_e, beta_pdf, 0.7, 0.1 ;
    omega, normal_pdf, 0.1, 0.025 ;
    rho_pi, normal_pdf, 0.4, 0.1 ;
    rho_g, normal_pdf, 0.4, 0.1 ;
    rho_x, normal_pdf, 0.1, 0.05 ;
    stderr eps_a, uniform_pdf, , , 0, 0.1 ;
    stderr eps_e, uniform_pdf, , , 0, 0.1 ;
    stderr eps_z, uniform_pdf, , , 0, 0.1 ;
    stderr eps_r, uniform_pdf, , , 0, 0.1 ;
end ;
```



(a) Uniform (b) Normal (c) Beta

(d) Gamma (e) Inverse Gamma

Figure 6: **Distributions**

Finally, we declare the estimation command as we did for maximum likelihood estimation. There are lots of options for Bayesian estimation which can go here; it's worth seeing the Dynare Manual for an overview of these options, and resources on Bayesian estimation for what they mean.

```
estimation(datafile=data);
```

6.3 Output in workspace

The estimation results are saved in the `oo_` structure in the workspace. Some details are given in Table 3.

Table 3: The Matlab Workspace - Estimation Results

Stored in	What
`oo_.posterior_mode`	Parameters and shock standard errors at the posterior mode.
`oo_.posterior_std`	Standard error of the estimated values at the posterior mode.
`oo_.posterior_density`	Density of the posterior for each parameter (x,y values for each parameter).
`oo_.prior_density`	Density of the prior for each parameter (x,y values for each parameter).
`oo_.SmoothedVariables`	Estimated values of each endogenous variable for all time periods.
`oo_.SmoothedShocks`	Estimated values of each shock for all time periods.

6.4 Shock decomposition

Estimation produces a set of structural shocks, which are usually particularly interesting. One useful command to understand what these shocks mean at each time period is `shock_decomposition`,

which decomposes the movements in each endogenous variable in the model to the shocks. That is, the movement in every endogenous variable has to be described by a combination of movements in the model's shocks; `shock_decomposition` gives the contribution of each shock to the movement in each variable. It is called by:

```
shock_decomposition ;
```

6.5 Useful Resources

Here are some useful resources on estimation:

- Dynare User Guide

7 Forecasting with Dynare

Dynare can use the model solution to compute forecasts. Following the `stoch_simul` command, use the command:

```
forecast ;
```

Some relevant options: `periods = x` generates forecasts for `x` periods, and `nograph` suppresses the graph outputs. These options overlap with the `stoch_simul` options, so if no options are specified, the matching options specified in `stoch_simul` apply.

The output of the forecast is stored in `oo_.forecast.Mean`. If variables are not simulated, `forecast` just generates forecasts of the variables from steady-state – that is, they just stay at steady-state. If variables are simulated, as in Section 5, the initial period of the forecast is the first simulated period.

8 Other useful actions

8.1 Saving data to an `.m` file

If we want to save a vector or matrix to a Matlab script (`.m`) file, we can use Dynare's command `datatomfile()`. This might be helpful if we want to use simulated data to estimate a model, for instance. For example, in the following line, Dynare will save the three vectors of simulated data `g`, `pi`, and `r` to `data.m` in the working directory.

```
datatomfile('data',{'g';'pi';'r'})
```

8.2 Dynare macro language: including external text in `.mod` file

Dynare can parse commands written in the Dynare macro language, constructing a `.mod` file written with macro language commands to a `.mod` file written without commands, which is then pro-

cessed as a normal .mod file. This is useful for simplifying repetitive code in the .mod file or for automatically including blocks of text in the .mod file.

At any point in the Dynare code, you can use the @#include x.mod command. This adds the contents of x.mod to your Dynare file. x.mod should use the regular Dynare language conventions. One instance where we have found this useful is for changing the structure of the foreign sector in a small open economy model. We have an automated Matlab file that writes the equations of a .mod file describing a world VAR estimated over different samples and lag lengths. This .mod file is then added to the primary .mod file by using the include command; the remainder of the primary .mod file remains unchanged.

To write a VAR .mod file, first write the equations as a string. Then, specify a file name, and open the file with permissions wt, meaning the file can be written to in text mode. Next, print the string to the file with option %s, meaning write as a string. Finally, close the file.

```
VAR = ( 'y = rho11*y(-1) + rho12*x(-1) + e_y ; ', ...
        'x = rho21*y(-1) + rho22*x(-1) + e_x ; ' ) ;

filename    = 'WORLD.mod' ;           % Choose filename
fid1        = fopen(filename, 'wt') ; % Open the file
fprintf(fid1, '%s', VAR) ;            % Print to the new file
fclose(fid1) ;                        % Close the file
```

Next, we want to include the WORLD.mod file into the primary .mod file's set of equations. To do this, use the following block of code:

```
model (linear) ;

@#include "WORLD.mod"
```

Variables and parameters referred to in the included file need to be declared at the top of the primary .mod file as per usual. This can also be performed by the @#include command if you want to change variables/parameters/calibrations etc.

One time-saving use of @#include is to write estimated parameter values to a .mod file, and include these in the parameter declaration of the primary .mod file. This means we do not have to manually put in estimated parameter values into a new .mod file if we wanted to do analysis with the estimated parameters. For example, suppose we write a .m file which calls dynare ireland_estimation.mod. Then we can write a section of code which writes the parameters, as a string, to a .mod file called parameters.mod. We can then use @#include parameters.mod in the parameter setup section of ireland.mod and call it to Dynare. This routine becomes useful if we wanted to automate the analysis of many estimated models.

Finally, we can use the savemacro option in Dynare to save the .mod file with the macro language elements expanded–the saved .mod file of file.mod is called file-macroexp.mod.

8.3 Writing LaTeX from Dynare

Dynare has in-built functions to easily print the model equations and estimation results to a LaTeX file. For it to know what LaTeX variables to use, you need to define the LaTeX representation of each variable and parameter. This occurs in the declaration section, after each variable, in dollar signs. For example, for a selection of parameters, we use the block:

```
parameters beta $\beta$, omega $\omega$, psi $\psi$, ...
```

After declaring LaTeX equivalent variables and parameters, at the end of the Dynare model code, use the command:

```
write_latex_dynamic_model ;
```

to generate a `.tex` file with name `MODELNAME_latex_dynamic.tex` where `MODELNAME` is the name of the `.mod` file. This `.tex` file can be compiled, or the important lines copied into another LaTeX file.

9 An application: graphical DSGE

In this application, we will use Ireland's (2004) linearized equations to get aggregate demand and aggregate supply curves relating inflation to output growth. Expressing the model in this way helps us see the mechanisms of the DSGE in a familiar framework.[2]

9.1 Aggregate demand and aggregate supply curves

The linearized equations, with inflation and output growth expressed in levels terms, rather than deviations from steady-state, are:

$$\hat{x}_t = E_t\hat{x}_{t+1} - (\hat{r}_t - E_t\hat{\pi}_{t+1}) + (1-\omega)(1-\rho_a)\hat{a}_t \tag{9}$$

$$\pi_t = \pi + \beta E_t\hat{\pi}_{t+1} + \psi\hat{x}_t - \hat{e}_t \tag{10}$$

$$\hat{r}_t = \hat{r}_{t-1} + \rho_\pi\hat{\pi}_t + \rho_g\hat{g}_t + \rho_x\hat{x}_t + \varepsilon_{r,t} \tag{11}$$

$$\hat{x}_t = \hat{y}_t - \omega\hat{a}_t \tag{12}$$

$$g_t = g + \hat{y}_t - \hat{y}_{t-1} + \hat{z}_t \tag{13}$$

$$\hat{a}_t = \rho_a\hat{a}_{t-1} + \varepsilon_{a,t} \tag{14}$$

$$\hat{e}_t = \rho_e\hat{e}_{t-1} + \varepsilon_{e,t} \tag{15}$$

[2]This is based on Jones, C and M. Kulish (2016), 'A Graphical Representation of a DSGE Model', *Applied Economics*. 2016. All Dynare and Matlab files, data, results and documentation can be downloaded at `wp.nyu.edu/callum` or the RePEc page.

$$\hat{z}_t = \varepsilon_{z,t} \tag{16}$$

where, different to the above expression of the linearized equations, π is the log steady-state of inflation, and g is the log steady-state of output growth.

The equations of the model can be manipulated to form aggregate supply and aggregate demand schedules relating inflation to output growth. To find the aggregate supply schedule, substitute Equations (12) and (13) into (10) to get:

$$\pi_t = \psi g_t + \hat{s}_t + (\pi - \psi g) \tag{17}$$

where $\hat{s}_t = \beta E_t \hat{\pi}_{t+1} + \psi \hat{y}_{t-1} - \psi \hat{z}_t - \omega \psi \hat{a}_t - \hat{e}_t$. In the space of output growth and inflation (g_t, π_t), equation (17) expresses inflation as a linear function of output growth, with slope ψ and intercept $\hat{s}_t + (\pi - \psi g)$. Note that the time-varying intercept $\hat{s}_t$ is zero when the economy is on its balanced growth path. Also note that the slope of the curve depends on the degree of nominal price rigidities. That is, as $\psi \to \infty$ prices become fully flexible which implies a vertical aggregate supply curve. Conversely as the cost of price adjustment rises, $\psi \to 0$, implying that the aggregate supply curve flattens.

To obtain the aggregate demand schedule, substitute Equations (11), (12) and (13) into (9):

$$\pi_t = -\left(\tfrac{1+\rho_g+\rho_x}{\rho_\pi}\right) g_t + \hat{d}_t + \left(\pi + \tfrac{1+\rho_g+\rho_x}{\rho_\pi} g\right) \tag{18}$$

where $\hat{d}_t = -\frac{1}{\rho_\pi}\hat{r}_{t-1} + \frac{1}{\rho_\pi} E_t \hat{x}_{t+1} + \frac{1}{\rho_\pi} E_t \hat{\pi}_{t+1} - \left(\frac{1+\rho_x}{\rho_\pi}\right)\hat{y}_{t-1} + \left(\frac{1+\rho_x}{\rho_\pi}\right)\hat{z}_t + \frac{\omega(1+\rho_x)+(1-\omega)(1-\rho_a)}{\rho_\pi}\hat{a}_t - \frac{1}{\rho_\pi}\varepsilon_{r,t}$. Note that, as for the time-varying intercept in the aggregate supply curve, when the economy is on its balanced growth path, $\hat{d}_t$ is zero. The slope of the curve (18) depends on the parameters of the policy rule. A greater response to deviations of inflation from target, ρ_π, flattens the curve. Vice versa, stronger responses to output growth, ρ_g, and the output gap, ρ_x, steepen AD.

9.2 Adding the curves to Dynare

The model equations, can be written in a Dynare `.mod` file. It is helpful to add the aggregate demand $\hat{d}_t$ and supply curve $\hat{s}_t$ shifters to the equations. We can use Dynare's output of these shifters, together with the parameter and steady-state values to draw the curves. We add to the `ireland2004.mod` file we created earlier two endogenous variables, `adt` and `ast` where:

```
ast = beta * pi(+1) + psi * y(-1) - psi * z - omega * psi * a - e ;
adt = 1/rho_pi * x(+1) + 1/rho_pi * pi(+1) - (1+rho_x)/rho_pi * y(-1) ...
        - 1/rho_pi * r(-1) + (1+rho_x)/rho_pi * z ...
        + ((omega*(1+rho_x)+(1-omega)*(1-rho_a))/rho_pi) * a ...
        - (1/rho_pi) * eps_r ;
```

9.3 Estimating on US data

We need estimates of the model parameters to generate the estimated aggregate demand and aggregate supply curves. We follow Ireland's (2004) procedure with updated data. We calibrate β to 0.99, ψ to 0.1, and ω to 0.06. The remaining parameter estimates and standard errors are given in Table 4. Because of the strong estimated response of the short-rate to output growth and the output gap relative to the response to inflation, the slope of the aggregate demand curve is estimated to be steep: $-\frac{1+\rho_g+\rho_x}{\rho_\pi} = -4.4$. The aggregate supply curve is, owing to the low value of $\psi = 0.1$, relatively flat. This explains why the model attributes an important role to aggregate demand shocks.

Table 4: **Maximum Likelihood Estimates**

Parameter	Estimate	Standard Error
ρ_a	0.9489	0.0224
ρ_e	0.9470	0.0393
ρ_π	0.2885	0.0404
ρ_g	0.2090	0.0451
ρ_x	0.0679	0.0177
σ_a	0.0365	0.0122
σ_e	0.0008	0.0002
σ_z	0.0120	0.0015
σ_r	0.0026	0.0003

As noted, the evolution of the economy, in output growth-inflation space, can be described by the intersection of the aggregate supply and demand schedules at each point in time. A shock will shift the time-varying intercepts of the curves, $\hat{s}_t$ and $\hat{d}_t$. Over time, these curves move back toward the steady-state. So, for example, on the impact of a one standard deviation shock to $\varepsilon_{r,t}$, $\hat{s}_t = -0.001$ and $\hat{d}_t = -0.026$. Simultaneously, both curves move. This illustrates an important feature of the analysis, that in general equilibrium, shocks simultaneously act on the shifting components of aggregate demand and aggregate supply, $\hat{d}_t$ and $\hat{s}_t$, through the expectations channel. In the case of the monetary policy shock, in the period of the shock, the increase in $\hat{r}_t$ directly leads to a contraction in aggregate demand as agents adjust consumption. Simultaneously, agents expect the output gap to be negative in the next period and inflation to be below steady-state which, through the Euler condition, contracts aggregate demand further. For the supply side, a fall in inflation expectations acts to expand supply: because it is costly for firms to adjust prices, they have an incentive, for any given level of output, to reduce prices today.

We illustrate the curves and their dynamics following a monetary policy shock in output growth-inflation space. Figure 7 draws the aggregate demand and aggregate supply curves at steady-state, the period of the monetary policy shock $t = 1$, and four periods later at $t = 5$. It shows the contraction in aggregate demand and the expansion in aggregate supply from steady-

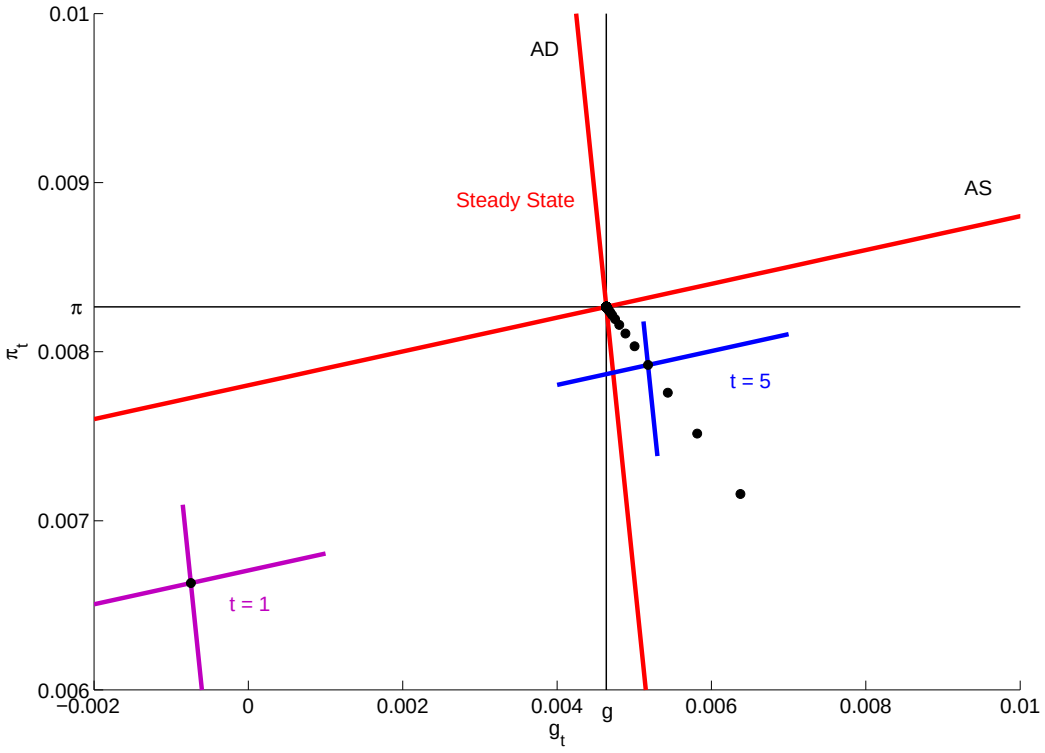


Figure 7: **Aggregate demand and aggregate supply curves: ε_r shock**

state following the shock, and the movements in the curves as the economy returns to steady-state.

A The full Dynare `.mod` file

```
var y, x, g, r, pi, a, e, z ;

varexo eps_r, eps_a, eps_e, eps_z ;

parameters beta, omega, psi, rho_pi, rho_g, rho_x, rho_a, rho_e,
           sig_r, sig_z, sig_a, sig_e ;

beta   = 0.99   ;  rho_x  = 0.0347 ;  sig_a  = 0.0405 ;
psi    = 0.1    ;  rho_g  = 0.0000 ;  sig_e  = 0.0012 ;
omega  = 0.0617 ;  rho_a  = 0.9470 ;  sig_z  = 0.0109 ;
rho_pi = 0.3597 ;  rho_e  = 0.9625 ;  sig_r  = 0.0031 ;

model (linear) ;
    x  = x(+1) - ( r - pi(+1) ) + ( 1 - omega ) * ( 1 - rho_a ) * a ;
    pi = beta * pi(+1) + psi * x - e ;
    r  = r(-1) + rho_pi * pi + rho_g * g + rho_x * x + eps_r ;
    x  = y - omega * a ;
    g  = y - y(-1) + z ;
    a  = rho_a * a(-1) + eps_a ;
    e  = rho_e * e(-1) + eps_e ;
    z  = eps_z ;
end ;

steady ;

shocks ;
    var eps_a = sig_a^2 ;
    var eps_e = sig_e^2 ;
    var eps_z = sig_z^2 ;
    var eps_r = sig_r^2 ;
end ;

stoch_simul ;
```