Dynare

Michel Juillard, CEPREMAP and University Paris 8

Table of Contents

Preface

Dynare is a pre–processor and a collection of GAUSS, MATLAB or SCILAB routines which solve non–linear models with forward looking variables. It is the result of research carried at CEPREMAP by several people (see Laffargue, 1990, Boucekkine, 1995, and Juillard, 1996, Collard and Juillard 2001a and 2001b). Although it can be put to other use, Dynare has been built in order to study the transitory dynamics of non–linear models with consistent expectations.

When the framework is deterministic, Dynare can be used for models with the assumption of perfect foresight. Typically, the system is supposed to be in a state of equilibrium before a period "1" when the news of a contemporaneous or of a future shock is learned by the agents in the model. The purpose of the simulation is to describe the reaction in anticipation of, then in reaction to the shock, until the system returns to the old or to a new state of equilibrium. In most models, this return to equilibrium is only an asymptotic phenomenon, which one must approximate by an horizon of simulation far enough in the future. Another exercise for which Dynare is well suited is to study the transition path to a new equilibrium following a permanent shock.

For deterministic simulations, Dynare uses a Newton–type algorithm, first proposed by Laffargue (1990), instead of a first order technique like the one proposed by Fair and Taylor (1983), and used in earlier generation simulation programs. We believe this approach to be in general both faster and more robust. The details of the algorithm used in Dynare can be found in Juillard (1996).

In a stochastic context, Dynare computes one or several simulations corresponding to a random draw of the shocks. Starting with version 2.3 (not available for GAUSS), Dynare uses a second order Taylor approximation of the expectation functions (see Judd, 1996, Collard and Juillard, 2001a, 2001b, and Schmitt-Grohe and Uribe, 2002).

1 Introduction

In practice, the simulation process is done in two steps: in the first one, the model and the processing instructions written by the user in a *model file* are interpreted and the proper GAUSS, MATLAB or SCILAB instructions are generated; in the second step, the program actually computes the simulation. Both steps are triggered by a single keyword: `dynare`.

1.1 Software requirements

This version of Dynare works only under Windows 95/98/NT/2000. For a Unix version, please, write me.

The Gauss version of Dynare has been written with Gauss version 3.2. It most likely doesn't work with previous versions.

The Matlab version has been written with Matlab 6.5.

The Scilab version has been tested with Scilab 2.6 and 2.7 (CVS 7/30/2002).

1.2 Installation

In case of update from a previous version, it is a good idea to copy the old version in a backup directory so as to be able to revert to it in case of problems. None of the previous files are usefull anymore, so you are strongly encouraged to remove them from directory 'c:\dynare'.

Unpack the zip file in the directory 'c:\' (If you want to use another directory, see below). The Gauss version in automatically installed in 'c:\dynare\gauss', the Matlab version in 'c:\dynare\matlab' and the Scilab version in 'c:\dynare\scilab'.

1.2.1 Installing the GAUSS version

If you had any previous version of Dynare, use the Gauss editor or any text editor to remove all references to it from the library file 'user.lcg'.

After unpacking the archive, start the Gauss program and type the following:

```
library pgraph
lib user c:\dynare\gauss\dynare.src
lib user c:\dynare\gauss\dynare1.src
lib user c:\dynare\gauss\dynare2.src
lib user c:\dynare\gauss\dynare3.src
```

If you installed Dynare for Gauss in a directory different from 'c:\dynare\gauss', change the above instructions accordingly and edit the following line in 'dynare.src'

```
declare string PARSER = "c:\\dynare\\gauss\dynare_g ";
```

1.2.2 Installing the Matlab version

After unpacking the archive, start the Matlab program and use the menu `File/Set path` to add 'c:\dynare\matlab' to the list of your paths.

If you installed Dynare for Matlab in a directory different from 'c:\dynare\matlab', change the above instructions accordingly and edit the following line in 'dynare.m'

```
command = ['c:\dynare\matlab\dynare_m ' x];
```

1.2.3 Installing the Scilab version

In the top directory of your Scilab distribution (identified by the presence of the 'scilab.star' file), create a file 'scilab.ini' containing

```
load('c:/dynare/scilab/lib');
```

or add this lines to your 'scilab.ini' file if it already exists. Alternatively, the line can be added to the file 'scilab.star', after similar statements. This second solution has the advantage of keeping access to Dynare even after the command `clear`.

If you installed Dynare for Scilab in a directory different from 'c:\dynare\scilab', change the above instructions accordingly and edit the following line in 'dynare.sci'

```
command = 'c:\dynare\scilab\dynare_s '+fname;
```

Then, restart Scilab and run the command `uplib()`.

1.3 Executing Dynare

dynare 'filename'['.mod']   [Command]

dynare executes instruction included in 'filename.mod'. 'filename.mod' is the name of the model file containing the model and the processing instructions.

In Gauss, dynare creates an intermediary file <*filename*>.gau with the instructions for the simulations. The Gauss version still accepts the former '.mdl' extension, but it is now deprecated.

In Matlab, dynare creates three intermediary files:

- <filename>.m with the instructions for the simulations
- <filename>_ff.m with the dynamic model equations
- <filename>_fff.m with the long run static model equations

In Scilab, dynare creates three intermediary files:

- <filename>.sci with the instructions for the simulations
- <filename>_ff.sci with the dynamic model equations
- <filename>_fff.sci with the long run static model equations

These files may be looked at to understand errors reported at the simulation stage.

Examples

```
dynare ramst
```

or

```
dynare ramst.mod
```

2 Model file commands

2.1 General declarations

2.1.1 periods

periods *INTEGER;* [Command]

This command is now deprecated (but will still work for older model files). It is not necessary when no simulation is performed and is replaced by an option PERIODS in SIMUL and STOCH_SIMUL.

Set the number of periods in the simulation. The periods are numbered from 1 to INTEGER. In perfect foresight simulations, it is assumed that all future events are perfectly known at the beginning of period 1.

Example

```
periods 100;
```

2.1.2 var

var *VARIABLE_LIST;* [Command]

VARIABLE_LIST : VARIABLE_NAME [,] VARIABLE_NAME [,] ...

This required command declares the endogenous variables in the model. The variable names must start with a letter and can't contain the following characters : ()+-*/^=!;:@#. or accentuated characters.

In GAUSS, setting *_longname = 1* allows the use of more than 8 characters in the variable names and makes a distinction between lower and upper case letters.

Example

```
var c gnp q1 q2;
```

2.1.3 varexo

varexo *VARIABLE_LIST;* [Commands]

VARIABLE_LIST : VARIABLE_NAME [,] VARIABLE_NAME [,] ...

This optional command declares the exogenous variables in the model. See command `var` for the syntax of variable names.

Exogenous variables are required if the user wants to be able to apply shocks to her model.

Example

```
varexo m gov;
```

2.1.4 parameters

parameters *PARAMETER_LIST;*   [Commands]
    PARAMETER_LIST : PARAMETER_NAME [,] PARAMETER_NAME [,] ...

This optional command declares parameters used in the model, in variable initialization or in shock declarations. The parameters must then be assigned values using standard syntax of underlying matrix programming language. Be carefull not to use names reserved by DYNARE or the package.

Example

```
parameters alpha, bet;

alpha = 0.3;
bet = sqrt(2);
```

2.2 Model declaration

2.2.1 model

model; [(*LINEAR*)] *EQUATION_LIST* **end;**   [Commands]
    The equations of the model are written in a block delimited by `model;` and `end;`.

There must be as many equations as there are endogenous variables in the model. The lead and lag of the variables are written in parenthesis immediately after the variable name. Leads or lags of more than one period are allowed. All the functions available in GAUSS, MATLAB or SCILAB, respectively, are recognized. Each equation must be terminated by a semicolon (;).

When the equations are written in homogenous form, it is possible to omit the "= 0" part and write only the left hand side of the equation.

The option LINEAR declares the model as being linear. It avoids to have to declare initial values for computing the steady state and it sets automatically ORDER=1 in STOCH_SIMUL.

Example 1

```
model;
c =  - k + aa*x*k(-1)^alph + (1-delt)*k(-1);
c^(-gam) = (aa*alph*x(+1)*k^(alph-1) + 1 - delt)*c(+1)^(-gam)/(1+bet);
end;
```

Example 2

```
model;
c + k - aa*x*k(-1)^alph - (1-delt)*k(-1);
c^(-gam) - (aa*alph*x(+1)*k^(alph-1) + 1 - delt)*c(+1)^(-gam)/(1+bet);
end;
```

Example 3

```
model(linear);
x = a*x(-1)+b*y(+1)+e_x;
y = d*y(-1)+e_y;
end;
```

2.3 Initial and terminal conditions

Used in perfect foresight mode, the types of forward–loking models for which DYNARE was designed require both initial and terminal conditions. Most often these initial and terminal conditions are static equilibria, but not necessarily.

One typical application is to consider an economy at the equilibrium, trigger a shock in first period, and study the trajectory of return at the initial equilbrium. To do that, one needs initval and shock(see next section).

Another one is to study, how an economy, starting from arbitrary initial conditions converges toward equilibrium. To do that, one needs initval and endval;

For models with lags on more than one period, the command histval permits to specify different historical initial values in different periods.

2.3.1 initval

initval;   [Command]

INITIALIZATION_STATEMENT;
[INITIALIZATION_STATEMENT]
...
end;

INITIALIZATION_STATEMENT : VARIABLE_NAME = EXPRESSION;

EXPRESSION is any valid expression returning a numerical value and can contain already initialized variable names.

The required initval; ... end; block serves two purposes. It set the initial and, possibly, terminal conditions for the simulation and provides numerical initialization for various computation tasks (steady, simul, stoch_simul).

Theoreticaly, initial conditions are only necessary for lagged variables. However, as initval provides also numerical initialization, it is necessary to provide values for all variables in the model, except if the model is declared as linear.

For stochastic models, it isn't necessary to delcare 0 as initial values for exogneous stochastic variables as it is the only possible value.

When the initval block is followed by the command steady, it is not necessary to provide exact initialization values for the endogenous variables. steady will use the values provided in the initval block as initial guess in the non–linear equation solver and computes exact values for the endogenous variables at the steady state. The steady state is defined by keeping constant the value of the exogenous variables.

Example

```
initval;
c = 1.2;
k = 12;
x = 1;
end;

steady;
```

2.3.2 endval

endval;   [Command]

INITIALIZATION_STATEMENT;
[INITIALIZATION_STATEMENT]
...
end;

INITIALIZATION_STATEMENT : VARIABLE_NAME = EXPRESSION;

EXPRESSION is any valid expression returning a numerical value and can contain already initialized variable names.

The optional `endval; ... end;` block serves two purposes. It set the terminal conditions for the simulation with the LBJ alogrithm, when those differ from the initial conditions. When it is the case, the `endval` block also provides the numerical initialization for various computation tasks (`steady`, `simul`), starting in period 1.

Theoreticaly, terminal conditions are required in the LBJ algorithm only for forward variables. However, as `endval` provides also numerical initialization, it is necessary to provide values for all variables in the model.

When the `endval` block is followed by the command `steady`, it is not necessary to provide exact values for the endogenous variables. `steady` will use the values provided in the `endval` block as initial guess in the non–linear equation solver and computes exact values for the endogenous variables at the steady state. The steady state is defined by keeping constant the value of the exogenous variables.

Example

```
var c k;
varexo x;

...

initval;
c = 1.2;
k = 12;
x = 1;
end;

steady;

endval;
c = 2;
k = 20;
x = 2;
end;

steady;
```

The initial equilibrium is comptuted by `steady` for x=1, and the terminal one, for x=2.

2.3.3 endval

histval; [Command]

> INITIALIZATION_STATEMENT;
> [INITIALIZATION_STATEMENT]
> ...
> **end;**
>
> INITIALIZATION_STATEMENT : VARIABLE_NAME (integer) = EXPRESSION;
>
> EXPRESSION is any valid expression returning a numerical value and can contain already initialized variable names.
>
> In models with lags on more than one period, the optional `histval; ... end;` block permits to specify different historical initial values for different periods.
>
> By convention in Dynare, period 1 is the first period of the simulation. Going backward in time, the first period before the start of the simulation is period 0, then period -1, and so on.
>
> If your lagged variables are linked by identities, be careful to satisfy these identities when you set historical initial values.
>
> **Example**
>
> ```
> var x y;
> varexo e;
>
> model;
> x = y(-1)^alpha*y(-2)^(1-alpha)+e;
> ...
> end;
>
> initval;
> x = 1;
> y = 1;
> e = 0.5;
> end;
>
> steady;
>
> histval;
> y(0) = 1.1;
> y(-1) = 0.9;
> end;
> ```

2.4 Shocks on exogenous variables

In a deterministic context, when one wants to study the transition of one equilibrium position to another, it is equivalent to analyze the consequences of a permanent shock and this in done in DYNARE through the proper use of `initval` and `endval`.

Another typical experiment is to study the effects of a temporary shock after which the system goes back to the original equilibrium (if the model is stable ...). A temporary shock is a temporary change of value of one or several exogenous variables in the model. Temporary shocks are specified with the command `shocks`.

In a stochastic framework, the exogenous variables take random values in each period. In DYNARE, these random values follow a normal distribution with zero mean, but it belongs to the user to specify the variability of these shocks. The non–zero elements of the matrix of variance–covariance of the shocks can be entered with the `shocks` command. Or, the entire matrix can be direclty entered with `Sigma_e`. Note that, starting with version 2.5.2, the direct specification of the **internal** matrix *Sigma_e_*, prone to errors, is discouraged.

If the variance of an exogenous variable is set to zero, this variable will appear in the report on policy and transition functions, but isn't used in the computation of moments and of Impulse Response Functions. Setting a variance to zero is an easy way of removing an exogenous shock.

2.4.1 shocks

shocks;   [Command]

> SHOCK STATEMENT
> [SHOCK STATEMENT]
> ...
> **end;**
>
> **In deterministic context**:
> SHOCK STATEMENT:
> **var** VARIABLE NAME;
> **periods** PERIOD STATEMENT;
> **values** VALUE STATEMENT;
>
> PERIOD STATEMENT: NUMBER[:NUMBER] or LIST OF NUMBERS
> VALUE STATEMENT: EXPRESSION
>
> The VALUE STATEMENT must be a valid expression returning a scalar value or a vector when the shock applies to a range of periods and takes a different values in each of these periods.
>
> For deterministic simulations, the `shocks` block specifies temporary changes in the value of an exogenous variables. For permanent shocks, use an `endval` block.
>
> **Example**
>
> > shocks;

```
    var e;
    periods 1;
    values 0.5;
    var u;
    periods 4:5;
    values 0;
    end;
```

In stochastic context, syntax 1:
SHOCK STATEMENT:
var VARIABLE NAME[,VARIABLE NAME] = expression ;

For stochastic simulations (available only in the MATLAB or SCILAB versions), the
`shocks` block specifies the non zero elements of the covariance matrix of the shocks.

Example

```
    shocks;
    var e = 0.000081;
    var e,u = phi*0.009*0.009;
    var u = 0.000081;
    end;
```

In stochastic context, syntax 2:
SHOCK STATEMENT:
var VARIABLE NAME;
stderr expression;

Alternatively, when the shocks are uncorrelated, it is possible to specify only the stan-
dard errors with a slightly different syntax (for consistency with previous versions).

Example

```
    shocks;
    var e;
    stderr 0.009;
    var u;
    stderr 0.009;
    end;
```

2.4.2 Sigma_e

Sigma_e   [Command]
    SIGMA_E = [MATRIX ELEMENT [,] MATRIX ELEMENT [,] ... ; MATRIX
    ELEMENT ...];
    MATRIX ELEMENT = (expression) or NUMBER

The matrix of variance-covariance of the shocks can be directly specified as a upper (or
lower) triangular matrix. Dynare builds the corresponding symmetrix matrix. Each
row of the triangular matrix, except the last one, must be terminated by a semi-colon
';'. For a given element, an expression using predefined parameters is allowed but must

be placed between parentheses. THE ORDER OF THE COVARIANCES IN THE MATRIX IS THE SAME AS THE ONE USED IN THE VAREXO DECLARATION.

NOTE: In previous versions, it was possible to directly set Dynare's internal covariance matrix *Sigma_e_*. This is still possible for compatibility with older .mod files, but STRONGLY DISCOURAGED as too prone to error. When setting *Sigma_e_* directly, the order of the exogenous shocks is the ALPHABETICAL order of their names.

Example

```
varexo u, e;
...
Sigma_e = [ 0.81 (phi*0.9*0.009); 0.000081];
```

where the variance of *u* is 0.81, the variance of *e*, 0.000081, and the correlation between *e* and *u* is *phi*.

2.5 Computations and simulations

DYNARE has special commands for the computation of the static equilibrium of the model (`steady`, of the eigenvalues of the linearized model (`check`) for dynamics local analysis, of a deterministic simulation (`simul`) and for stochastic simulations (`stoch_simul`).

2.5.1 steady

steady;   [Command]

Computes the equilibrium value of the endogenous variables for the value of the exogenous variables specified in the previous `initval` or `endval` block.

`steady` uses an iterative procedure and takes as initial guess the value of the endogenous variables set in the previous `initval` or `endval` block.

For complicated models, finding good numerical initial values for the endogenous variables is the trickiest part of finding the equilibrium of that model. Often, it is better to start with a smaller model and add new variables one by one.

Examples

See `initval` and `endval`.

2.5.2 check

check;   [Command]

Computes the eigenvalues of the model linearized around the values specified by the last `initval`, `endval` or `steady` statement. Generally, the eigenvalues are only meaningfull if the linearization is done around a steady state of the model. It is a device for local analysis in the neighborhood of this steady state.

A necessary condition for the uniqueness of a stable equilibrium in the neighborhood of the steady state is that there are as many eigenvalues larger than one in modulus as there are forward looking variables in the system. An additional rank condition requires that the square submatrix of the right Schur vectors corresponding to the

forward looking variables (jumpers) and to the explosive eigenvalues must have full rank.

check returns the eigenvalues in the global variable *eigenvalues_*.

2.5.3 simul

simul[(PERIODS=Integer)];   [Command]

Triggers the computation of a deterministic simulation of the model for the number of periods set in the option periods=.

2.5.4 stoch_simul

stoch_simul[(OPTIONS)] [*LIST OF VARIABLES*];   [Command]

(available currently only in the Matlab and the Scilab versions)

OPTIONS:

AR = Integer

> Order of autocorrelation coefficients to compute and to print (default = 5)

DR_ALGO = [0,1]

> Specify the algorithm used for computing the quadratic approximation of the decision rules:
>
> 0: uses a "pure" perturbation approach as in Schmitt-Grohe and Uribe (2002) (default)
> 1: moves the point around which the Taylor expansion is computed toward the means of the distribution as in Collard and Juillard (2001)

DROP = Integer

> Number of points dropped at the beginning of simulation before computing the summary statistics (default = 100)

HP_FILTER = Integer

> Uses HP filter with lambda = Integer before computing moments (default: no filter)

HP_NGRID = Integer

> Number of points in the grid for the discreet Inverse Fast Fourier Transform used in the HP filter computation. It may be necessary to increase it for highly autocorrelated processes (default = 512)

IRF = Integer

> Number of periods on which to compute the IRFs (default = 40)

LINEAR   Indicates that the original model is linear (put it rather in the MODEL command).

NOCORR   Doesn't print the correlation matrix (default = PRINT)

NOFUNCTIONS

> Doesn't print the coefficients of the approximated solution

NOMOMENTS
> Doesn't print moments of the endogenous variables

PERIODS = Integer
> Specify the number of periods to use in simulations. At order=1, no simulation is necessary to compute theoretical moments and IRFs. A number of PERIODS larger than one triggers automatically SIMUL=1 (default = 0).

ORDER = [1,2]
> Order of Taylor approximation (default = 2)

REPLIC = Integer
> Number of simulated series used to compute the IRFs (default = 1, if order = 1, and 50 otherwise)

SIMUL
> Computes a stochastic simulation of the model for the number of periods specified in the `PERIODS` statement. Uses `INITVAL` values, possibly recomputed by `STEADY`, as initial values for the simulation. The simulated endogenous variables are made available to the user in a vector for each variable and in the global matrix `y_`. The variables are ordered alphabeticaly in the `y_` matrix (default: no simulation)

SIMUL_SEED = Integer
> Specify a seed for the random generator so as to obtain the same random sample at each run of the program. Otherwise a different sample is used for each run (default: seed not specified).

When a LIST OF VARIABLES is specified, results are displayed only for these variables.

`stoch_simul` computes a Taylor approximation of the decision and transition functions for the model, impulse response functions and various descriptive statistics (moments, variance decomposition, correlation and autocorrelation coefficients). For correlated shocks, the variance decomposition is computed as in the VAR literature through a Cholesky decomposition of the covariance matrix of the exogenous variables. When the shocks are correlated, the variance decomposition depends upon the order of the variables in the `varexo` command.

Variance decomposition, correlation, autocorrelation are only displayed for variables with positive variance. Impulse response functions are only ploted for variables with response larger than 1e-10.

The covariance matrix of the shocks is specified either with the `shocks` command or with the `Sigma_e` command.

Example 1

```
shocks;
var e;
stderr 0.0348;
end;

stoch_simul;
```

performs the simulation of the 2nd order approximation of a model with a single stochastic shock, e, with a standard error of 0.0348.

Example 2

        stoch_simul(linear,irf=60) y k;

performs the simulation of a linear model and displays impulse response functions on 60 periods for variables y and k.

2.6 Displaying and saving results

DYNARE has comments to plot the results of a simulation and to save the results.

2.6.1 rplot

RPLOT *VARIABLE*;   [command]
   Plots a variable

2.6.2 dynatype

DYNATYPE (*FNAME*) [*VARLIST*];   [command]

   or
   DYNATYPE FNAME;

   DYNATYPE prints the variables included in VARLIST in a text file named FNAME. If VARLIST is missing, all endogenous variables are printed.

2.6.3 dynasave

DYNASAVE (*FNAME*) [*VARLIST*];   [command]

   or
   DYNASAVE FNAME;

   DYNASAVE saves the variables included in VARLIST in a binary file named FNAME. If VARLIST is missing, all endogenous variables are saved.

   In Matlab, variables saved with the DYNASAVE command can be retrieved by the MATLAB command LOAD -MAT FNAME.

3 Examples

Fabrice Collard (GREMAQ, University of Toulouse) has written a guide to stochastic simulations with DYNARE entitled "Dynare in Practice" which is in '`guide.pdf`'.

4 Bibliography

Boucekkine, R. (1995) "An alternative methodology for solving nonlinear forward-looking models," *Journal of Economic Dynamics and Control*, 19, 711–734.

Collard, F. and M. Juillard (2001a) "Accuracy of stochastic perturbation methods: The case of asset pricing models," *Journal of Economic Dynamics and Control*, 25, 979–999.

Collard, F. and M. Juillard (2001a) "A Higher–Order Taylor Expansion Approach to Simulation of Stochastic Forward–Looking Models with an Application to a Non–Linear Phillips Curve," *Computational Economics*, 17, 125–139.

Fair, R.C and J. Taylor (1983) "Solution and Maximum Likelihood Estimation of Dynamic Nonlinear Rational Expectation Models." *Econometrica*, 51, 1169–85.

Judd, K.L. (1996) "Approximation, Perturbation, and Projection Methods in Economic Analysis," in H.M. Amman, D.A. Kendrick and J. Rust (eds.) *Handbook of Computational Economics.* Amsterdam: North Holland Press, pp. 511–585.

Juillard, M. (1996) DYNARE: A program for the resolution and simulation of dynamic models with forward variables through the use of a relaxation algorithm. Paris: CEPREMAP, Couverture Orange No. 9602.

Laffargue, J.–P. (1990) "Rsolution d'un modle macroconomique avec anticipations rationnelles," *Annales d'Economie et Statistique*, 17, 97–119.

Schmitt–Grohe, S. and M. Uribe (2002) Solving Dynamic General Equilibrium Models Using a Second-Order Approximation to the Policy Function. Working Paper, Rutgers Univsersity.