DSGE Models with Dynare++. A Tutorial.

Ondra Kameník

July 2005

Contents

1 Installation

Dynare++ installation procedure is pretty straigthforward. Take the following steps:

1. Grab Dynare++ package from Dynare site either for Windows or for Linux according to your operating system.[1]
2. Unzip or untar the package to a directory of your choice.
3. Set operating system path to point to `dynare++-*.*`[2] subdirectory of the directory you have chosen. In Windows, this step ensures that libraries distributed along Dynare++ are on the path. This step is not really necessary in Linux.
4. If you have Matlab and want to run custom simulations (see 4.3), then set the Matlab path to the path from the previous step.

[1] If unsure, download the one for Windows.
[2] Stars stand for a version number.

Dynare++ uninstall procedure is even more simple. Just remove the `dynare++-*.*` directory.

If you want (or need) to compile Dynare++ from sources, grab Dynare++ source package and do your best.[3]

2 Sample Session

As an example, let us take a simple DSGE model whose dynamic equilibrium is described by the following first order conditions:

$$
\begin{aligned}
& c_t \theta h_t^{1+\psi} = (1-\alpha) y_t \\
& \beta E_t \left[\frac{\exp(b_t) c_t}{\exp(b_{t+1}) c_{t+1}} \left(\exp(b_{t+1}) \alpha \frac{y_{t+1}}{k_{t+1}} + 1 - \delta \right) \right] = 1 \\
& y_t = \exp(a_t) k_t^{\alpha} h_t^{1-\alpha} \\
& k_t = \exp(b_{t-1})(y_{t-1} - c_{t-1}) + (1-\delta) k_{t-1} \\
& a_t = \rho a_{t-1} + \tau b_{t-1} + \epsilon_t \\
& b_t = \tau a_{t-1} + \rho b_{t-1} + \nu_t
\end{aligned}
$$

The timing of this model is that the exogenous shocks ϵ_t, and qν_t are observed by agents in the beginning of period t and before the end of period t all endogenous variables with index t are decided. The expectation operator E_t works over the information accumulated just before the end of the period t (this includes ϵ_t, ν_t and all endogenous variables with index t).

The exogenous shocks ϵ_t and ν_t are supposed to be serially uncorrelated with zero means and time-invariant variance-covariance matrix. In Dynare++, these variables are called exogenous; all other variables are endogenous. Now we are prepared to start writing a model file for Dynare++, which is an ordinary text file and could be created with any text editor.

The model file starts with a preamble declaring endogenous and exogenous variables, parameters, and setting values of the parameters. The preamble follows:

```
var Y, C, K, A, H, B;
varexo EPS, NU;

parameters beta, rho, beta, alpha, delta, theta, psi, tau;
alpha = 0.36;
rho   = 0.95;
tau   = 0.025;
beta  = 0.99;
delta = 0.025;
psi   = 0;
theta = 2.95;
```

[3] Feel free to contact me if setting up the sources takes more time than solving your model by hand.

The section setting values of the parameters is terminated by a beginning of the `model` section, which states all the dynamic equations. A timing convention of a Dynare++ model is the same as the timing of our example model, so we may proceed with writing the model equations. The time indexes of c_{t-1}, c_t, and c_{t+1} are written as `C(-1)`, `C`, and `C(1)` resp. The `model` section looks as follows:

```
model;
C*theta*H^(1+psi) = (1-alpha)*Y;
beta*exp(B)*C/exp(B(1))/C(1)*
  (exp(B(1))*alpha*Y(1)/K(1)+1-delta) = 1;
Y = exp(A)*K^alpha*H^(1-alpha);
K = exp(B(-1))*(Y(-1)-C(-1)) + (1-delta)*K(-1);
A = rho*A(-1) + tau*B(-1) + EPS;
B = tau*A(-1) + rho*B(-1) + NU;
end;
```

At this point, almost all information that Dynare++ needs has been provided. Only three things remain to be specified: initial values of endogenous variables for non-linear solver, variance-covariance matrix of the exogenous shocks and order of the Taylor approximation. The remaining portion of the model file looks as follows:

```
initval;
Y = 1;
C = 0.8;
H = 1;
K = 5;
A = 0;
B = 0;
end;

vcov = [
  0.0002  0.00005;
  0.00005 0.0001
];

order = 7;
```

Note that the order of rows/columns of the variance-covariance matrix corresponds to the ordering of exogenous variables in the `varexo` declaration. Since the `EPS` was declared first, its variance is 0.0002, and the variance of `NU` is 0.0001.

Let the model file be saved as `example1.mod`. Now we are prepared to solve the model. At the operating system command prompt[4] we issue a command:

```
dynare++ example1.mod
```

When the program is finished, it produces two files: a journal file named `example1.jnl` and an output file Matlab MAT-4 `example1.mat`. The journal file contains information about time, memory and processor resources needed for all steps of solution. The

[4] Under Windows it is a `cmd` program, under Unix it is any shell

output file is more interesting. It contains various simulation results. It can be loaded into Matlab or Scilab and examined.[5] The following examples are done in Matlab, everything would be very similar in Scilab.

Let us first examine the contents of the MAT file:

```
>> load example1.mat
>> who

Your variables are:

dyn_g_1            dyn_i_Y            dyn_npred
dyn_g_2            dyn_irfm_EPS_mean  dyn_nstat
dyn_g_3            dyn_irfm_EPS_var   dyn_shocks
dyn_g_4            dyn_irfm_NU_mean   dyn_ss
dyn_g_5            dyn_irfm_NU_var    dyn_state_vars
dyn_i_A            dyn_irfp_EPS_mean  dyn_steady_states
dyn_i_B            dyn_irfp_EPS_var   dyn_vars
dyn_i_C            dyn_irfp_NU_mean   dyn_vcov
dyn_i_EPS          dyn_irfp_NU_var    dyn_vcov_exo
dyn_i_H            dyn_mean
dyn_i_K            dyn_nboth
dyn_i_NU           dyn_nforw
```

All the variables coming from one MAT file have a common prefix. In this case it is `dyn`, which is Dynare++ default. The prefix can be changed, so that the multiple results could be loaded into one Matlab session.

In the default setup, Dynare++ solves the Taylor approximation to the decision rule and calculates unconditional mean and covariance of the endogenous variables, and generates impulse response functions. The mean and covariance are stored in `dyn_mean` and `dyn_vcov`. The ordering of the endogenous variables is given by `dyn_vars`.

In our example, the ordering is

```
>> dyn_vars
dyn_vars =
H
A
Y
C
K
B
```

and unconditional mean and covariance are

```
>> dyn_mean
dyn_mean =
    0.2924
    0.0019
```

[5] For Matlab `load example1.mat`, for Scilab `mtlb_load example1.mat`

```
    1.0930
    0.8095
   11.2549
    0.0011
>> dyn_vcov
dyn_vcov =
    0.0003    0.0006    0.0016    0.0004    0.0060    0.0004
    0.0006    0.0024    0.0059    0.0026    0.0504    0.0012
    0.0016    0.0059    0.0155    0.0069    0.1438    0.0037
    0.0004    0.0026    0.0069    0.0040    0.0896    0.0016
    0.0060    0.0504    0.1438    0.0896    2.1209    0.0405
    0.0004    0.0012    0.0037    0.0016    0.0405    0.0014
```

The ordering of the variables is also given by indexes starting with `dyn_i_`. Thus the mean of capital can be retrieved as

```
>> dyn_mean(dyn_i_K)
ans =
   11.2549
```

and covariance of labor and capital by

```
>> dyn_vcov(dyn_i_K,dyn_i_H)
ans =
    0.0060
```

The impulse response functions are stored in matrices as follows

matrix	response to
`dyn_irfp_EPS_mean`	positive impulse to `EPS`
`dyn_irfm_EPS_mean`	negative impulse to `EPS`
`dyn_irfp_NU_mean`	positive impulse to `NU`
`dyn_irfp_NU_mean`	negative impulse to `NU`

All shocks sizes are one standard error. Rows of the matrices correspond to endogenous variables, columns correspond to periods. Thus capital response to a positive shock to `EPS` can be plotted as

```
plot(dyn_irfp_EPS_mean(dyn_i_K,:));
```

The data is in units of the respective variables, so in order to plot the capital response in percentage changes from the decision rule's fix point (which is a vector `dyn_ss`), one has to issue the commands:

```
Kss=dyn_ss(dyn_i_K);
plot(100*dyn_irfp_EPS_mean(dyn_i_K,:)/Kss);
```

The plotted impulse response shows that the model is pretty persistent and that the Dynare++ default for a number of simulated periods is not sufficient. In addition, the model persistence puts in doubt also a number of simulations. The Dynare++ defaults can be changed when calling Dynare++, in operating system's command prompt, we issue a command:

```
dynare++ --per 300 --sim 150 example1.mod
```

This sets the number of periods to 300 and the number of simulations to 150.

3 Running Dynare++

This section deals with Dynare++ calculations and its input. The first subsection 3.1 describes what Dynare++ solves and calculates. Subsection 3.2 provides a list of command line options. The last subsection 3.3 deals with a format of Dynare++ model file.

3.1 What Dynare++ Calculates

Dynare++ solves first order conditions of a DSGE model in the recursive form:

$$E_t[f(y^{**}_{t+1}, y_t, y^*_{t-1}, u_t)] = 0,$$

where y is a vector of endogenous variables, and u a vector of endogenous variables. Some of elements of y can occur at time $t+1$, these are y^{**}. Elements of y occurring at time $t-1$ are denoted y^*. The exogenous shocks are supposed to be serially independent and normally distributed $u_t \sim N(0, \Sigma)$.

The solution of this dynamic system is a decision rule

$$y_t = g(y^*_{t-1}, u_t)$$

Dynare++ calculates a Taylor approximation of this decision rule of a given order. The approximation takes into account deterministic effects of future volatility, so a point about which the Taylor approximation is done will be different from the fix point y of the rule yielding $y = g(y^*, 0)$.

The fix point of a rule corresponding to a model with $\Sigma = 0$ is called *deterministic steady state*. In contrast to deterministic steady state, there is no consensus in literature how to call a fix point of the rule corresponding to a model with non-zero Σ. I am tempted to call it *stochastic steady state*, however, it might be confused with unconditional mean or with steady distribution. So I will use a term *fix point* to avoid a confusion.

By default, Dynare++ solves the Taylor approximation about the deterministic steady state, calculates the fix point, and recalculates the Taylor approximation so that it would work with deviations from the fix point rather than from the deterministic steady state.

Alternatively, Dynare++ can split the uncertainty to a few steps and take smaller steps when calculating the fix points. For models with large uncertainty and/or with non-linearities along the uncertainty dimension, this method will provide more precise answers. Such problems include optimal portfolios or asset pricing.

3.1.1 Simulating the Decision Rule

Both methods end up with a decision rule centralized around its fix point y, this is (in Einstein notation)

$$y_t - y = \sum_{i=1}^{k} \frac{1}{i!} \left[g_{(y^* u)^i} \right]_{\alpha_1 \dots \alpha_i} \left[\begin{array}{c} y^*_{t-1} - y^* \\ u_t \end{array} \right]^{\alpha_1 \dots \alpha_i}$$

Note that there is no term for $i = 0$ since the rule is centralized around its fix point y.

Then, the decision rule is simulated to obtain draws from ergodic (unconditional) distribution of endogenous variables. The mean and the covariance are reported.

These simulations are then used for impulse response function calculations. For each shock, the realized shocks in these control simulations are taken and an impulse is added and the new realization of shocks is simulated.Then the control simulation is subtracted from the simulation with the impulse. This is done for all control simulations and the results are averaged. As the result, we get an expectation of difference between paths with impulse and without impulse. In addition, the sample variances are reported. They might be useful for confidence interval calculations.

For each shock, Dynare++ calculates IRF for two impulses, positive and negative. Size of an impulse is one standard error of a respective shock.

3.1.2 Numerical Approximation Checks

Optionally, Dynare++ can run three kinds of checks for Taylor approximation errors. All three methods numerically calculate the the residual of the DSGE equations

$$E[f(g^{**}(g^*(y^*, u), u'), g(y^*, u), y^*, u)|y^*, u]$$

which must be ideally zero for all y^* and u. This integral is evaluated by either product or Smolyak rule applied to one dimensional Gauss–Hermite quadrature. The user does not need to care about the decision. An algorithm yielding higher quadrature level and less number of evaluations less than a user given maximum is selected.

The three methods differ only by a set of y^* and u where the residuals are evaluated. These are:

- The first method calculates the residuals along the shocks for fixed y^* equal to the fix point. We let all elements of u be fixed at 0 but one element, which varies from $-\mu\sigma$ to $\mu\sigma$, where σ is a standard error of the element and μ is the user given multiplier. In this way we can see how the approximation error grows if the fix point is disturbed by a shock of varying size.

- The second method calculates the residuals along a simulation path. A random simulation is run, and at each point the residuals are reported.

- The third method calculates the errors on an ellipse of the state variables y^*. The shocks u are always zero. The ellipse is defined as

$$\{Ax| \ \|x\|_2 = \mu\},$$

where μ is a user given multiplier, and $AA^T = V$ for V being a covariance of endogenous variables based on the first order approximation. The method calculates the residuals at low discrepancy sequence of points on the ellipse. Both the residuals and the points are reported.

3.2 Command Line Options

The calling syntax of the Dynare++ is

```
dynare++ [--help] [--version] [options] <model file>
```

where the model file must be given as the last token and must include its extension.

The options are as follows:

`--help` This prints a help message and exits.

`--version` This prints a version information and exits.

`--per` *num* This sets a number of simulated periods to *num*. This number is used when calculating unconditional mean and covariance and for IRFs. Default is 100.

`--sim` *num* This sets a number of stochastic simulations. This number is used when calculating unconditional mean and covariance and for IRFs. The total sample size for unconditional mean and covariance is the number of periods times the number of successful simulations. Note that if a simulation results in `NaN` or `+-Inf`, then it is thrown away and is not considered for the mean nor the variance. The same is valid for IRF. Default is 80.

`--steps` *num* If the number *num* is greater than 0, this option invokes a multi-step algorithm (see section 3.1), which in the given number of steps calculates fix points and approximations of the decision rule for increasing uncertainty. Default is 0, which invokes a standard algorithm for approximation about deterministic steady state.

`--prefix` *string* This sets a common prefix of variables in the output MAT file. Default is `dyn`.

`--seed` *num* This sets an initial seed for the random generator. However, if a number of parallel threads is greater than 1, there is no guarantee that the identical runs with identical initial seed will yield identical results. This is because there is only one instance of the random generator shared by multiple threads, and the operating system's scheduler thus becomes another source of uncertainty. I hope this will be changed in some future release. Default is 934098.

`--threads` *num* This sets a number of parallel threads. Complex evaluations of Faa Di Bruno formulas, simulations and numerical integration can be pararellized, Dynare++ exploits this advantage. You have to have a hardware

support for this, otherwise there is no gain from the parallelization. As a rule of thumb, set the number of threads to the number of processors. An exception is a machine with Pentium 4 with Hyper Threading (abbreviated by HT). This processor can run two threads concurently. Since these processors are present in most new PC desktops, the default is 2.

`--check` *pPeEsS* This selects types of residual checking to be performed. See section 3.1.2 for details. The string consisting of the letters "pPeEsS" governs the selection. The upper-case letters switch a check on, the lower-case letters off. "P" stands for checking along a simulation path, "E" stands for checking on ellipse, and finally "S" stands for checking along the shocks. It is possible to choose more than one type of check. The default behavior is that no checking is performed.

`--check-evals` *num* This sets a maximum number of evaluations per one residual. The actual value depends on the selected algorithm for the integral evaluation. The algorithm can be either product or Smolyak quadrature and is chosen so that the actual number of evaluations would be minimal with maximal level of quadrature. Default is 1000.

`--check-num` *num* This sets a number of checked points in a residual check. One input value num is used for all three types of checks in the following way:

- For checks along the simulation, the number of simulated periods is $10 \cdot num$
- For checks on ellipse, the number of points on ellipse is $10 \cdot num$
- For checks along the shocks, the number of checked points corresponding to shocks from 0 to $\mu\sigma$ (see 3.1.2) is num.

Default is 10.

`--check-scale` *float* This sets the scaling factor μ for checking on ellipse to $0.5 \cdot float$ and scaling factor μ for checking along shocks to $float$. See section 3.1.2. Default is 2.0.

The following are a few examples:

```
dynare++ --sim 300 --per 50 blah.mod
dynare++ --check PE --check-num 15 --check-evals 500 blah.dyn
dynare++ --steps 5 --check S --check-scale 3 blahblah.mod
```

The first one sets the number of periods for IRF to 50, and sets a sample size for unconditional mean and covariance calculations to 6000. The second one checks the decision rule along a simulation path having 150 periods and on ellipse at 150 points performing at most 500 evaluations per one residual. The third one solves the model in five steps and checks the rule along all the shocks from -3σ to 3σ in $2 * 10 + 1$ steps (10 for negative, 10 for positive and 1 for at zero).

3.3 Dynare++ Model File

The model file must comply the following formulation of a DSGE model:

$$E_t[f(y^{**}_{t+1}, y_t, y^*_{t-1}, u_t)] = 0$$

This has the following implications:

- Variables declared as shocks must occur only at time t. This is not binding, since one can add an endogenous variable to bring a shock to period $t+1$, or $t-1$.
- Endogenous variables cannot occur at times before $t-1$. This is not binding, since one can add an endogenous variable to bring it to one period backward.
- Endogenous variables cannot occur at times after $t+1$. This is binding. Currently, one has to use a smarter software to solve models with more than one lead.
- There is no constraint on variables occurring at both times $t+1$ and $t-1$. Virtually, all variables can occur at $t-1$, t and $t+1$.
- Dynare++ applies the operator E_t to all occurrences at time $t+1$. The realization of u_t is included in the information set of E_t. See an explanation of Dynare++ timing on page 2.

The model equations are formulated in the same way as in Matlab Dynare. The time indexes different from t are put to round parenthesis in this way: `C(-1)`, `C`, `C(+1)`.

The model file can contain user comments. Their usage can be understood from the following piece of the model file:

```
P*C^(-gamma) = // line continues until semicolon
  beta*C(+1)^(-gamma)*(P(+1)+Y(+1)); // asset price
// choose dividend process: (un)comment what you want
Y/Y_SS = (Y(-1)/Y_SS)^rho*exp(EPS);
/*
Y-Y_SS = rho*(Y(-1)-Y_SS)+EPS;
*/
```

3.4 Incompatibilities with Matlab Dynare

This section provides a list of incompatibilities between a model file for Dynare++ and Matlab Dynare. These must be considered when a model file for Matlab Dynare is being migrated to Dynare++. The list is the following:

- There is no `periods` keyword.
- An r-value in `parameters` and `initval` sections must be a numerical literal, not a Matlab expression. See the following:

```
initval;
P = 1/(1-beta); // illegal in Dynare++
P = 20;         // perfectly OK
```

- There are no commands like `steady`, `check`, `simul`, `stoch_simul`, etc.
- There are no sections like `estimated_params`, `var_obs`, etc.
- The variance-covariance matrix of endogenous shocks is given by `vcov` matrix in Dynare++. An example follows. Note that there is no semicolon at the end of last row.

```
vcov = [
0.05 0 0 0;
0 0.025 0 0;
0 0 0.05 0;
0 0 0 0.025
];
```

4 Dynare++ Output

There are two output files; a data file of MAT-4 format containing the output data, and a journal text file containing an information about the Dynare++ run.

4.1 MAT File

The contents of the data file is depicted below. We suppose that the prefix is `dyn`.

`dyn_nstat`	Scalar. A number of static variables (those occurring only at time t).
`dyn_npred`	Scalar. A number of variables occurring at time $t-1$ and not at $t+1$.
`dyn_nboth`	Scalar. A number of variables occurring at $t+1$ and $t-1$.
`dyn_nforw`	Scalar. A number of variables occurring at $t+1$ and not at $t-1$.
`dyn_vars`	Column vector of endogenous variable names in Dynare++ ordering.
`dyn_i_`*endovar*	Scalar. Index of a variable named *endovar* in the `dyn_vars`.
`dyn_shocks`	Column vector of exogenous variable names.
`dyn_i_`*exovar*	Scalar. Index of a shock named *exovar* in the `dyn_shocks`.
`dyn_state_vars`	Column vector of state variables, these are stacked variables counted by `dyn_npred`, `dyn_nboth` and shocks.

`dyn_vcov_exo`	Matrix $nexo \times nexo$. The variance-covariance matrix of exogenous shocks as input in the model file. The ordering is given by `dyn_shocks`.
`dyn_mean`	Column vector $nendo \times 1$. The unconditional mean of endogenous variables. The ordering is given by `dyn_vars`.
`dyn_vcov`	Matrix $nendo \times nendo$. The unconditional covariance of endogenous variables. The ordering is given by `dyn_vars`.
`dyn_ss`	Column vector $nendo \times 1$. The fix point of the resulting approximation of the decision rule.
`dyn_g_`*order*	Matrix $nendo \times ?$. A derivative of the decision rule of the *order* multiplied by $1/order!$. The rows correspond to endogenous variables in the ordering of `dyn_vars`. The columns correspond to a multidimensional index going through `dyn_state_vars`. The data is folded (all symmetrical derivatives are stored only once).
`dyn_steady_states`	Matrix $nendo \times nsteps + 1$. A list of fix points at which the multi-step algorithm calculated approximations. The rows correspond to endogenous variables and are ordered by `dyn_vars`, the columns correspond to the steps. The first column is always the deterministic steady state.
`dyn_irfp_`*exovar*`_mean`	Matrix $nendo \times nper$. Positive impulse response to a shock named *exovar*. The row ordering is given by `dyn_vars`. The columns correspond to periods.
`dyn_irfp_`*exovar*`_var`	Matrix $nendo \times nper$. The variances of positive impulse response functions.
`dyn_irfm_`*exovar*`_mean`	Same as `dyn_irfp_`*exovar*`_mean` but for negative impulse.
`dyn_irfp_`*exovar*`_var`	Same as `dyn_irfp_`*exovar*`_var` but for negative impulse.
`dyn_simul_points`	A simulation path along which the check was done. Rows correspond to endogenous variables, columns to periods. Appears only if `--check P`.
`dyn_simul_errors`	Errors along `dyn_simul_points`. The rows correspond to equations as stated in the model file, the columns to the periods. Appears only if `--check P`

`dyn_ellipse_points`	A set of points on the ellipse at which the approximation was checked. Rows correspond to state endogenous variables (the upper part of `dyn_state_vars`, this means without shocks), and columns correspond to periods. Appears only if `--check E`.
`dyn_ellipse_errors`	Errors on the ellipse points `dyn_ellipse_points`. The rows correspond to the equations as stated in the model file, columns to periods. Appears only if `--check E`.
`dyn_shock_`*exovar*`_errors`	Errors along a shock named *exovar*. The rows correspond to the equations as stated in the model file. There are $2m + 1$ columns, the middle column is the error at zero shock. The columns to the left correspond to negative values, columns to the right to positive. Appears only if `--check S`.

4.2 Journal File

The journal file provides information on resources usage during the run and gives some informative messages. The journal file is a text file, it is organized in single line records. The format of records is documented in a header of the journal file.

The journal file should be consulted in the following circumstances:

- Something goes wrong. For example, if a model is not Blanchard–Kahn stable, then the eigenvalues are dumped to the journal file.

  If the unconditional covariance matrix `dyn_vcov` is NaN, then from the journal file you will know that all the simulations had to be thrown away due to occurrence of NaN or Inf. This is caused by non-stationarity of the resulting decision rule.

  If Dynare++ crashes, the journal file can be helpful for guessing a point where it crashed.

- You are impatient. You might be looking at the journal file during the run in order to have a better estimate about the time when the calculations are finished. In Unix, I use a command `tail -f blah.jnl`.[6]

- Heavy swapping. In case if the physical memory is not sufficient, an operating system starts swapping memory pages with a disk. If this is the case, the journal file can be consulted for information on memory consumption and swapping activity.

- Not sure what Dynare++ is doing. If so, read the journal file, which contains a detailed record on what was calculated, simulated etc.

[6] This helps to develop one of the three programmer's virtues: *impatience*. The other two are *laziness* and *hubris*; according to Larry Wall.

4.3 Custom Simulations

When Dynare++ run is finished it dumps the derivatives of the calculated decision rule to the MAT file. The derivatives can be used for a construction of the decision rule and custom simulations can be run. This is done by `dynare_simul.m` M-file in Matlab. It reads the derivatives and simulates the decision rule with provided realization of shocks.

All the necessary documentation can be viewed by command

```
help dynare_simul
```