MS Macroeconomics – Tech Slides 3
Introduction to local solution methods and Dynare

Camilo Granados
University of Texas at Dallas
Spring 2023

Outline

I. DSGE formulation and solution (via perturbation/linear local approximation)

II. Dynare

Model Setup (summarized)

We consider a simple model with no frictions here. The focus is on the solution methods rather than the features of the model. For a discussion of the model see the slides on the Topic 2 (RBC model).

Neoclassical Growth Model:

A household maximizes its lifetime utility subject to its budget constraint (BC).

The BC accounts for the fact that they are the owners of a firm with technology: $y_t = z_t k_{t-1}^{\alpha}$ (uses capital k to manufacture a product y). z_t is the total factor productivity (TFP) and is subject to shocks.

(Optimization) Problem to consider: The household in each period decides how much to allocate to consumption and investment ($i_t = k_t - (1-\delta)k_{t-1}$, replaced below in the BC)

$$\max_{\{c_t, k_t\}_{t=1}^{\infty}} \mathbb{E}_0 \sum_{t=1}^{\infty} \beta^{t-1} \frac{c_t^{1-\nu} - 1}{1-\nu}$$

$$s.t.$$

$$c_t + k_t = z_t k_{t-1}^{\alpha} + (1-\delta)k_{t-1}$$

$$z_t = (1-\rho) + \rho z_{t-1} + \epsilon_t$$

with k_0 given, $\mathbb{E}_t[\epsilon_{t+1}] = 0$, and $\mathbb{E}_t[\epsilon_{t+1}^2] = \sigma^2$

Characterizing the equilibrium:

The equilibrium allocation we are looking for is the one consistent with the optimality conditions of the agents. These First Order Conditions (F.O.C.) are:

$$
\begin{aligned}
c_t^{-\nu} &= \mathbb{E}_t \left[\beta c_{t+1}^{-\nu} (\alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta) \right] \\
c_t + k_t &= z_t k_{t-1}^{\alpha} + (1-\delta) k_{t-1} \\
z_t &= (1-\rho) + \rho z_{t-1} + \epsilon_t
\end{aligned}
$$

What are we looking for?

We want to use this system and solve for the endogenous variables (c_t, k_t) in terms of exogenous:

$$
\begin{aligned}
c_t &= c(z_t, k_{t-1}) \\
k_t &= k(z_t, k_{t-1})
\end{aligned}
$$

where the functions $c(\cdot)$ and $k(\cdot)$ are consistent with the optimality conditions above.

These functions are also called **policy rules** (given a state, you can use these rules to determine the optimal value of your variables of interest-i.e. your "policy" to follow if you're trying to implement an optimal allocation)

Solution method

Unfortunately, finding such solution is not easy.

We **cannot solve the model analytically**: The system of equations characterizing the equilibrium is non-linear, and has expectations terms that aren't always trivial to evaluate.

There are versions of the model solvable with pencil and paper, but would be oversimplified (e.g., complete depreciation case)

Fortunately, economists have come up with methods to solve these models:

- **Local methods: (Perturbation)** obtain policy rule from Taylor expansion of the optimality conditions around a given reference point.
  - this solution is locally valid around the reference point used $\rightarrow$ we use the **Steady State**.
  - **This is what Dynare does and we focus on today.**
- Global methods: Value Function Iteration, Policy Function Iteration, Parametrized Expectations Algorithm.
  - These solutions are valid globally, not only around a given point.
  - However, these methods are harder to implement (computationally costly). Hence, they are used for models with few state variables.

Local Methods: Perturbation

We take our system of equations (the one we want to solve) and use Taylor expansions to approximate the policy rules.

To refresh your memory, a Taylor expansion for a function $f(x)$ set around $\bar{x}$ looks like this:

$$f(x) \approx f(\bar{x}) + \frac{f'(\bar{x})}{1!}(x - \bar{x}) + \frac{f''(\bar{x})}{2!}(x - \bar{x})^2 + \frac{f'''(\bar{x})}{3!}(x - \bar{x})^3 + \cdots$$

↳ at 1st order $\Delta \ln x \approx$ log-linearization

[More details/If interested in the details] To apply this to the system above we just have to take derivatives of each equation with respect to the exogenous variables and evaluate the derivatives at our known reference point given a set of parameter values. After obtaining expressions for the derivatives of our policy functions with respect to the states (some steps involved) we can obtain linear functions of the form $c_t = c(z_t, k_{t-1})$

Since we use Taylor expansions two features are important:

- The point of reference for the expansion ($\bar{x}$ above). The close x is to $\bar{x}$ the better the approximation will be.
- The order of the approximation (how many derivative terms we are taking on the RHS)

Log-linearization

A first order approximation (using only the first derivative) is also equivalent to **log-linearization**

Log-linearization:

- we express our model in terms of the log-difference (or percentage deviations) of the variables with respect to the steady state (mean of the detrended model).
- That is, instead of x_t, our model is stated in terms of $\hat{x}_t = \ln(x_t) - \ln(\bar{x}) = \frac{x_t - \bar{x}}{\bar{x}}$.
- This can be convenient as the resulting system of variables is linear now
  (can be solved analytically, e.g. using the Method of Undetermined Coefficients)
- Route taken by many papers, interested in analyzing the structure of the solution further
  (and not only obtaining the IRF, or numerical values for the solutions)
- Of course, the problem is that sometimes we want to use a higher order approximation. For higher orders (higher than 2), the algebra becomes too cumbersome.

Log-linearization (cont.)

How to do it? (we already saw how in Topic 2 (RBC), but will review it again now)

it's not hard, just need to be careful when re-expressing the model in log-linear terms.

Let's look at a simple example: we will log-linearize the equation $y_t = z_t k_{t-1}^{\alpha}$

- Take logs of the original equation: $\ln(y_t) = \ln(z_t) + \alpha \ln(k_{t-1})$ (1)
- Take logs of the static (steady state) equation: $\ln(\bar{y}) = \ln(\bar{z}) + \alpha \ln(\bar{k})$ (2)
- Get (1)-(2) and re-arrange conveniently:

$$\ln(y_t) - \ln(\bar{y}) = \ln(z_t) - \ln(\bar{z}) + \alpha(\ln(k_{t-1}) - \ln(\bar{k}))$$

$$\hat{y}_t = \hat{z}_t + \alpha \hat{k}_{t-1}$$

There are other tricks depending on what type of function you have. The one above works easily with something like $y_t = z_t k_{t-1}^{\alpha}$, but it does not work well with sums like: $i_t = k_t - (1-\delta)k_{t-1}$

For non-multiplicative cases, we just use other steps/shortcuts (or take the Taylor expansion and rearrange)

A great (complete and easy to understand) handout on this is: econweb.rutgers.edu/rchang/linearization08.pdf

We do this with each equation of our model (optimality conditions).

The result is a linear system of n equations with n unknowns that we can solve for analytically.

Log-linearization vs. Dynare

This is the **same** that Dynare does ...

With the addition that we can make other changes easily (e.g. use higher order of approximations)

The trade-off is that many people use Dynare as a black box.

[So read carefully what each command and feature of Dynare you use actually does]

Having said that, Dynare is also useful in providing other quantities and results of interest to us:

- Impulse responses to shocks (remember autorregressive processes?)
- Moments of our variables (expected moments, variances, correlations)
- Autocorrelations
- Simulated data (and the same statistical quantities mentioned for theoretical counterparts)
- Estimation of Models (Maximum Likelihood, Bayesian methods if you want to educate your model with some real data)

The Steady State

Now let's go back to one important feature of this method (in general and in Dynare): the reference point for the approximation.

The reference point used will be the **Steady State**

- Then, the solution of the model will be valid around (close enough) that point
- Why we pick the steady state?

  It's convenient, because it's where the model fluctuates around. That is, in the presence of fluctuations the economy will converge to it after the effects of shocks die out.

  > After a shock, once we reach that point, that is: $c_t = \bar{c}$, the economy remains there: $c_{t+1} = c_{t+2} = c_{t+3} = \cdots = \bar{c}$.

  Then, most of the time our economy should be close or converging to this point and then the solution would remain valid.

  With that in mind, there is no other point that works better
  (in the sense of making sure the solution remains valid most of the time)

The Steady State (cont.)

By definition, the Steady State will be the given by the values of the variables characterizing the **static** version of our (F.O.C.) system of equations to solve:

$$\begin{aligned} c^{-\nu} &= \beta c^{-\nu}(\alpha z k^{\alpha-1} + 1 - \delta) \\ c + k &= z k^{\alpha} + (1-\delta)k \\ z &= (1-\rho) + \rho z + \epsilon \end{aligned}$$

Let's find it:

First, look at the last equation, the long run, or mean value of the shock is zero ($\epsilon = 0$). Then, $z = 1$.

Also, from the first eq. we can get k:

$$\cancel{c^{-\nu}} = \beta \cancel{c^{-\nu}}(\alpha \cancel{z}^{1} k^{\alpha-1} + 1 - \delta) \quad \Rightarrow k = \left[\tfrac{1}{\alpha}\left(\tfrac{1}{\beta} - 1 + \delta\right)\right]^{\frac{1}{\alpha-1}}$$

Given we already got k we can easily get the value for c from the second equation:

$$c + \cancel{k} = \cancel{z}^{1} k^{\alpha} + (\cancel{1} - \delta)k \quad \Rightarrow c = k^{\alpha} - \delta k$$

And we're done. Of course, this is a relatively simple model, that's why it was easy.

The Steady State and Dynare

Unfortunately, for many models finding the steady state is hard.

Actually, one of the most troublesome parts of Dynare is to find the the Steady State. Without a valid one, Dynare cannot solve your model.

It's troublesome because many times, after simplifying the model (removing trends or dynamics and $E[\cdot]$) we are still left with a non-linear system that we can't solve analytically.

An **alternative**, is to obtain a **numerical solution** for the Steady State and use those values instead.

We'll talk about the alternatives for providing the Steady State in Dynare in a moment.

The Solution of the model

After determining the steady state, we will be able to find a solution that looks like this:

$$c_t = \bar{c} + a_{c,k}(k_{t-1} - \bar{k}) + a_{c,z_{-1}}(z_{t-1} - \bar{z}) + a_{c,z}\epsilon_t$$
$$k_t = \bar{k} + a_{k,k}(k_{t-1} - \bar{k}) + a_{k,z_{-1}}(z_{t-1} - \bar{z}) + a_{k,z}\epsilon_t$$
$$z_t = \rho z_{t-1} + \epsilon_t$$

Is that the solution we look for? ... yes. Everything on the RHS of the eqs. is known or exogenous.

How it looks in Dynare: run the file I provided with the HW3 with this command

```
dynare SimpleDSGEexample
```

You'll find the solution in the output:

```
POLICY AND TRANSITION FUNCTIONS
                                k             z             c
Constant                28.348419      1.000000      2.306617
k(-1)                    0.962061             0      0.048040
z(-1)                    2.157104      0.950000      0.707457
eps                      2.270636      1.000000      0.744692
```

These are the constant and coefficients of the equations above.

About this Solution and What Else Dynare can Do

The solution takes the form a linear dynamic equation that depends on the shock.

Hence, a **crucial feature is that the variables are inheriting the time series properties of the shocks** as time goes on.

> This is what economists refer to when they talk about "shock-driven economic fluctuations"

The shock is a white-noise process ($\epsilon_t \sim N(0, \sigma^2)$) $\cdots$ but it feeds an **autorregressive** TFP process (z_t).

And we just saw that the solution is a linear function of the states (z_t included).

Thus: **all of our variables will be autorregressive processes** of some sort

As a result, we will be interested in the time series (and statistical) properties of our variables.

The output of Dynare

That is why Dynare gives as part of the output the expected values of the rest of the system, the correlations, and autocorrelations.

```
THEORETICAL MOMENTS
VARIABLE          MEAN  STD. DEV.   VARIANCE
k              28.3484     0.8800     0.7743
z               1.0000     0.0224     0.0005
c               2.3066     0.0545     0.0030


MATRIX OF CORRELATIONS
Variables          k       z       c
k             1.0000  0.6723  0.9813
z             0.6723  1.0000  0.8021
c             0.9813  0.8021  1.0000


COEFFICIENTS OF AUTOCORRELATION
Order           1       2       3       4       5
k          0.9990  0.9962  0.9918  0.9858  0.9785
z          0.9500  0.9025  0.8574  0.8145  0.7738
c          0.9951  0.9887  0.9811  0.9722  0.9622
```

I hope now its apparent why we've been studying this statistical parameters and quantities from the start of the course.

Impulse Responses

Additionally we will be able to apply shocks to our model and see how the rest of the variables react (and converge back to their means) in the **impulse responses**

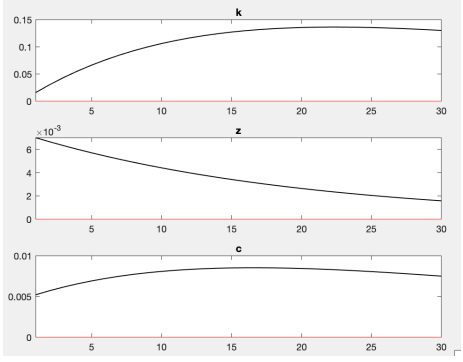


Figure: Impulse responses to a 1 SD productivity shock

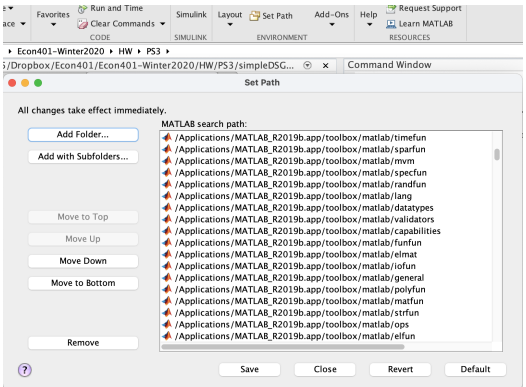
Installing Dynare

The installation is not hard:

Go here: https://www.dynare.org/download/

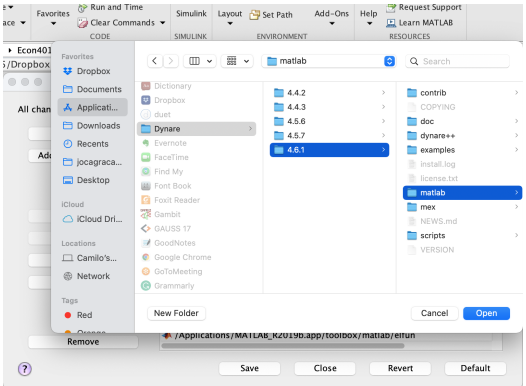
Download and Install in your Laptop (by following the usual instructions).

Then enter Matlab. Click in `Set Path`



Installing Dynare (cont.)

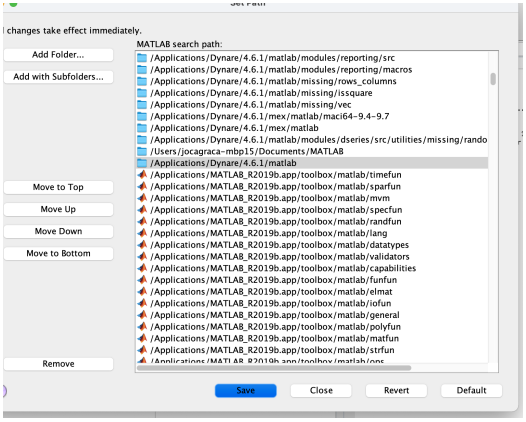
Then click in `Add Folder`



Go to the folder where Dynare was installed, then select the folder called **matlab**

(This is key: do not select/add the whole Dynare folder, only the "matlab" one.)

Installing Dynare (cont.)



Then you are done. Now Matlab will be able to run Dynare commands.

Structure of the Code

The file with a code for a model consists of several blocks:

1. Variables block
2. Parameters block
3. Model block
4. Steady State block
5. Shocks block
6. Solution (or simulation, or estimation) block

As with Matlab, we end an instruction (and a block) with semicolon (;).

Also, we can enter comments with double slash or percentage sign (// or %)

Let's review each of the blocks in our sample .mod file (the one attached with the HW3)

Variables and Parameters blocks

Variables block: We declare variables with `var` followed by the names. Similar for exogenous shocks with `varexo`

```
// Variables block
var c, k, z;
varexo eps;
```

Parameter block: Here we declare the list of parameters, and then the values they take.

```
// Parameters block
parameters betta, rho, alphha, nu, deltta, sig;

alphha  = 0.33;
rho     = 0.95;
betta   = 0.99;
nu      = 1;
deltta  = 0.025;
sig     = 0.007;
```

Note: my spelling of the greek letter names for the parameters may look weird to you. The reason (this is one of many good coding practices) is that I'm changing the typical spelling a bit to prevent overwriting built-in functions of Matlab with the same name.

Model block

We open the block with "`model;`" and end it with "`end;`".

In the middle we include the equations of our model (F.O.C.)

```
// Model Equations block
model;

c^(-nu) = betta*c(+1)^(-nu)*(alphha*z(+1)*k^(alphha-1) + 1 - deltta) ;
c + k = z*k(-1)^alphha + (1 - deltta)*k(-1) ;
z = (1 - rho) + rho*z(-1) + eps;

end;
```

Variables timing: How to enter something like c_t or $\mathbb{E}_t c_{t+1}$ to Matlab/Dynare?

Use `x` for a variable timed at t

Use `x(-1)` for predetermined variables (timed at $t-1$)

Use `x(+1)` for expectations about the future

Steady State Block [can skip if not interested in SS details]

In practice, this is the hardest part and the main input Dynare needs to find a solution.

The system evaluated at the Steady State is static (we dropped the time subscripts), **but it is still non-linear** and hard to solve for.

There are two options you can opt for:

1. Let Dynare calculate the Steady State

   In this case we still have to provide initial values for Dynare to start a numerical search. These values should be a good guess, i.e., they should be close enough to the actual steady state.

   Alternatively (but very rarely), we can provide algebraic expressions for the solution of the Steady State (as in the HW3 file). However, this is feasible only for very simple models.

2. Calculate the steady state yourself and give it to Dynare

   With this option, what we do is to provide an external file with the name "*filename*`_steadystate.m`" (where *filename* corresponds to the name of your mod file) where we calculate the Steady State (usually numerically but can provide algebraic expressions or actual values)

Note: If you don't provide anything Dynare will assume the Steady State values for all variables are zero.

If you find files online where that works, it's because they include a log-linearized model directly, and the steady state of such model is zero ($\frac{\bar{x}-\bar{x}}{\bar{x}}$)

If you want a specific video guide on how to obtain the steady state in Dynare follow this **[link]**

Steady State block (cont)

If using the **option 1**:

```
// Steady State block
initval;
k = (betta*alphha/(1-betta*(1-deltta)))^(1/(1-alphha));
c = k^alphha-deltta*k;
z = 1;
end;
```

If using the **option 2**: you don't write the block

Instead, we would provide an external file (e.g., "simpleDSGEexample_steadystate.m", for our HW file) with a particular predefined structure and commands for obtaining and declaring the Steady State (SS).

When we have to find the SS, it's usually done with a numerical solver, where we declare the static system of equations, provide initial values and look for it.

We will try to provide models that are already running (i.e. where we specify the Steady State), either directly, or with a "*_steadystate.m*" accompanying file.

If curious, see this mini-tutorial video by one of the Dynare developers: https://www.youtube.com/watch?v=Ei_z0HSfYNo&t=4s

Shocks block

You start it with `shocks;`, end it with `end;`

In the middle declare the name of the shock that will act up, and the size of its standard deviation.

```
// Shocks block
shocks;
var eps; stderr sig;
end;
```

Note: (trick) if you want to apply a negative shock, you should change the TFP formula rather than this block.

To do it, just change the sign accompanying the error (e.g. `z = (1 - rho) + rho*z(-1) - eps;`)

Solution block

Finally, you can use the `stoch_simul(options)` command for solving the model.

There are many options you can consider.

At the end you can specify which variables you want to be reflected in the output (and IRFs).

If you don't provide a list it will consider all the variables.

```
stoch_simul(order=1,IRF=30) k z c;
```

As for the options used here, I am asking Dynare to solve the model using a first order Taylor Approximation, and to compute the IRFs for 30 periods.

Conclusions

- We reviewed the very basics of Dynare. This is the simplest introduction you can find.
- Many, many more features and details that I abstracted from. But this is a good way to get started. Again, make sure you know what you are using Dynare for and not to use it as a black box.
- Reminder: this is just one of the possible solution methods for macroeconomic models. Depending on the features of each specific model, it may or not be the best (or appropriate).
- There is also a lot of documentation and references online (see next slide).

References

These are some references to Dynare tutorials, examples and materials. This is by no means exhaustive, there's a lot more online.

- Wouter Den Haan website: http://www.wouterdenhaan.com/numerical/slidesdynare.pdf
  The whole website has plenty of resources in quantitative macro.
- Johannes Pfeiffer website: https://sites.google.com/site/pfeiferecon/dynare
  Repository on the mod files for many models.
- Eric Sims website:

  A great handout on Dynare: https://www3.nd.edu/ esims1/using_dynare.pdf

  Another repository with files for replicating several models:
  https://www3.nd.edu/ esims1/matlab_codes.html
- Dynare official website: https://www.dynare.org/resources/
  Includes the links to tutorials, examples, website with more models' implementations.
- Macroeconomic Model Data Base: http://www.macromodelbase.com/
  Here you can make a online comparison of the output of many models, e.g., of IRFs!