Course 6 — Short Macroeconomics Course Using Julia

Petre Caraiani

Institute for Economic Forecasting; Bucharest University of Economic Studies

November 28, 2025

Outline

Perturbation Method

- I discuss one most used approaches in solving DSGE model.
- Based on the idea of approximation.
- Mathematical background: implicit function theorem and Taylor approximation.

DSGE packages in Julia

- MacroModelling: class 5.
- I cover two more packages: SolveDSGE and Dynare Julia.
- Both implement the perturbation method.

Course 6 Workflow

- Perturbation method.
- SolveDSGE package.
- Dynare Julia package.
- Practice session.

Goal of Perturbation

- Compute a local (Taylor) approximation to the policy functions of a DSGE model around a nonstochastic steady state.
- Output: decision rules for endogenous variables as functions of states and shocks.
- Advantages: fast, general, handles medium/large models, integrates with likelihood/Kalman filtering.

Generic DSGE Representation

The equilibrium conditions of many DSGE models can be written using the following representation:

$$E_t f(y_{t+1}, y_t, x_{t+1}, x_t) = 0 \tag{1}$$

Here E_t is the expectations operator which is conditioned on the information that the agents have at their disposition at time t. x_t is the vector predetermined variables of dimension $n_x \times 1$, while y_t is the vector of non-predetermined variables having a dimension $n_y \times 1$. Furthermore, n is defined as $n = n_x + n_y$.
Since this is a stochastic model, the state variable vector can be further split into endogenous and exogenous predetermined variables. We write that $x = [x_t^1; x_t^2]$. The exogenous state variables follow an autoregressive law of motion:

$$^{2}_{t+1} = \Lambda x_t^2 + \tilde{\eta}\sigma\epsilon_{t+1} \tag{2}$$

Solution

The solution to the model in equation (5.11) can be written as:

$$y_t = g(x_t, \sigma) \tag{3}$$

$$x_{t+1} = h(x_t, \sigma) + \eta\sigma\epsilon_{t+1} \tag{4}$$

Objective

We are interested in deriving first and second-order solutions around the non-stochastic steady state. This is defined by the vectors $(\bar{x}, \bar{y})$ such that the function f verifies the relationship:

$$f(\bar{y}, \bar{y}, \bar{x}, \bar{x}) = 0 \tag{5}$$

Approximate Solution

Using the prime for $t+1$ and dropping the time subscripts, we get:

$$F(x,\sigma) = E_t f(g(h(x,\sigma)+\eta\sigma\epsilon',\sigma), g(x,\sigma), h(x,\sigma)+\eta\sigma\epsilon', x) = 0 \tag{6}$$

Since $F(x,\sigma)$ is equal to zero for any values of the variable x and parameter σ, it also follows that the derivatives of F (for any order) are zero too. Then:

$$F_{x^k,\sigma^j}(x,\sigma) = 0, \forall x, \sigma, j, k \tag{7}$$

First-Order Solution

A first order approximation implies finding approximate solutions to the functions h, g around the point $(x, \sigma) = (\bar{x}, 0)$ such that the following relationships hold:

$$g(x, \sigma) = g(\bar{x}, 0) + g_x(\bar{x}, 0)(x - \bar{x}) + g_\sigma(\bar{x}, 0)\sigma \tag{8}$$

$$h(x, \sigma) = h(\bar{x}, 0) + h_x(\bar{x}, 0)(x - \bar{x}) + h_\sigma(\bar{x}, 0)\sigma \tag{9}$$

2nd order Solution I

We perform a second-order approximation for the functions g, h by considering the same steady state value given by $(x, \sigma) = (\bar{x}, 0)$.
The second order approximations for g and h can be written as follows:

$$\begin{aligned}
[g(x,\sigma)]^i = [g(\bar{x},0)]^i + [g_x(\bar{x},0)]^i_\alpha [(x-\bar{x})]_\alpha + [g_\sigma(\bar{x},0)]^i [\sigma] \\
+\frac{1}{2}[g_{xx}(\bar{x},0)]^i_{ab}[(x-\bar{x})]_\alpha [(x-\bar{x})]_b \\
+\frac{1}{2}[g_{x\sigma}(\bar{x},0)]^i_a [(x-\bar{x})]_\alpha [\sigma] \\
+\frac{1}{2}[g_{\sigma x}(\bar{x},0)]^i_a [(x-\bar{x})]_\alpha [\sigma] \\
+\frac{1}{2}[g_{\sigma\sigma}(\bar{x},0)]^i_a [\sigma][\sigma]
\end{aligned} \tag{10}$$

2nd order Solution II

$$
\begin{aligned}
[h(x,\sigma)]^i = [h(\bar{x},0)]^j + [h_x(\bar{x},0)]^j_\alpha [(x-\bar{x})]_\alpha + [h_\sigma(\bar{x},0)]^j [\sigma] \\
+\frac{1}{2}[h_{xx}(\bar{x},0)]^j_{ab}[(x-\bar{x})]_\alpha [(x-\bar{x})]_b \\
+\frac{1}{2}[h_{x\sigma}(\bar{x},0)]^j_a [(x-\bar{x})]_\alpha [\sigma] \\
+\frac{1}{2}[h_{\sigma x}(\bar{x},0)]^j_a [(x-\bar{x})]_\alpha [\sigma] \\
+\frac{1}{2}[h_{\sigma\sigma}(\bar{x},0)]^j_a [\sigma][\sigma]
\end{aligned}
\tag{11}
$$

What is SolveDSGE.jl?

- Julia package for specifying, solving, and simulating (linearized) DSGE models via text model files.
- Pipeline:
  1. Write a plain-text model file (variables, shocks, parameters, equations).
  2. Solution methods based on perturbation (up to 4th order) and projection methods.
  3. Process & solve to obtain a solution.
  4. Generate IRFs, simulate data.

Core functionality (at a glance)

- **Model parsing**: reads a simple domain-specific language from `.txt`.
- **Solution**: computes decision rules.
- **IRFs**: impulse responses to named shocks over chosen horizons.
- **Simulation**: draws shocks and simulates observables/states.

Basic Julia workflow

```
using SolveDSGE
# 1) Point to your model file
modelfile = joinpath(@__DIR__, "growth_model.txt")
# 2) Parse/validate and solve
process_model(modelfile)          # syntax + steady-state checks
sol = solve_model(modelfile)      # solve and return a solution object
# 3) IRFs and simulation
irf = impulses(sol, n_periods, size, mc)
```

Declare an Optimal Growth (Cass–Koopmans) model I

File: `dsge_sgm.txt` (minimal example)

```
states:
k, z
end
jumps:
c, ce
end
shocks:
eps
end
```

Declare an Optimal Growth (Cass–Koopmans) model II

File: `dsge_sgm.txt` (minimal example)

```
# PARAMETERS
parameters beta alpha delta rho_a sigma_a;
# CALIBRATION
beta    = 0.99;
alpha   = 0.33;
delta   = 0.025;
rho     = 0.95;
sd = 0.007;
```

Declare an Optimal Growth (Cass–Koopmans) model III

File: `dsge_sgm.txt` (minimal example)

```
# MODEL
k(+1) = (1.0 - delta)*k + exp(z)*k^alpha - c
c^(-sigma) =beta*ce(+1)
ce = c^(-sigma)*(1.0 - delta + alpha*exp(z)*k^(alpha - 1.0))
z(+1) = rho*z + sd*eps
```

Dynare Julia

- The user guide aims to cover the main functionalities in DynareJulia.
- Current list (Dynare version: 0.10.4): calib_smoother; deterministic_trends; endval; estimated_params; estimated_params_init; estimation; histval; homotopy; initval; initval_file; perfect_foresight_setup (only some options); perfect_foresight_solver (only some options, including lmmcp); planner_objective; ramsey_constraints; ramsey_model; shocks; steady (including numerical solution); stoch_simul (only order = 1).

Running Files

- Big advantage: you can run the same `.mod` files as in Dynare/Matlab.
- How to run a model (sketch): `context = @dynare "FILENAME.mod" [OPTIONS...]`
- Details on options:
  https://dynarejulia.github.io/Dynare.jl/dev/running-dynare/

Solving and Simulating a Model

- Focus on a well-known model: Iacoviello [2005].
- Monetary business cycle model with nominal loans and collateral constraints tied to housing values.

The Syntax I

1. The `.mod` file structure is (generally) similar to Dynare/Matlab.
2. Declaring endogenous and exogenous variables:
3. Endogenous:

```
var VAR_NAME [TEX_NAME] (long_name=QUOTED_STRING | NAME=QUOTED_STRING) ...
```

4. Exogenous:

```
varexo VAR_NAME [TEX_NAME] (long_name=QUOTED_STRING | NAME=QUOTED_STRING)
```

The Syntax II

1. Declaring parameters is also similar:

```
parameters PARAM_NAME [TEX_NAME] (long_name=QUOTED_STRING | NAME=QUOTED_ST
```

2. Initializing parameters:

```
PARAM_NAME = EXPRESSION;
```

Obtaining the Steady State I

1. Use `steady_state_model` block to define steady state.
2. Assign values to variables:

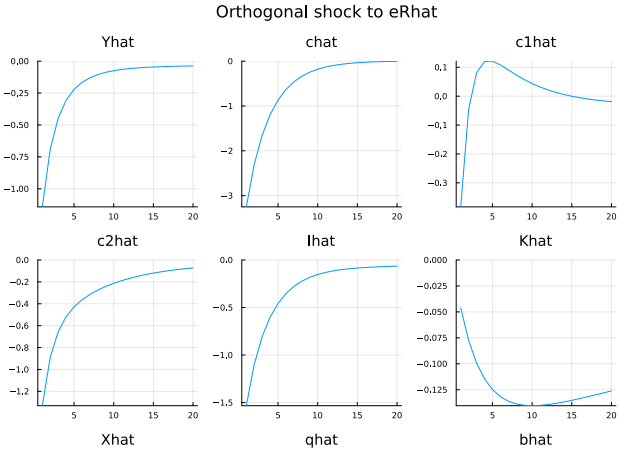
```
VARIABLE_NAME = EXPRESSION;
```

Obtaining a Local Approximation

1. Use `stoch_simul (OPTIONS...);` to get a local approximation.
2. Options include (examples):
   - `ar` = order of autocorrelation coefficients (default: 5).
   - `irf` = integer horizon for impulse responses.
   - `periods` = number of simulated periods.

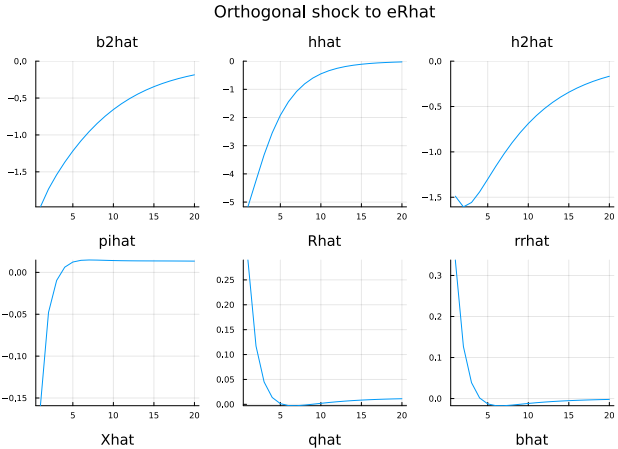
Results: IRFs I

Figure: IRFs to a Monetary Policy Shock



Results: IRFs II

Figure: IRFs to a Monetary Policy Shock



Deterministic Simulation

- Focus on a simple New Keynesian model: Gali [2015].
- We look at a perfect foresight simulation.

Perfect Foresight Syntax

1. To perform a perfect foresight simulation, use (schematically):

```
perfect_foresight!(; periods, context = context,
display = true,
linear_solve_algo = ilu, maxit = 50, mcp = false,
tolf = 1e-5, tolx = 1e-5)
```

Estimating a Model

- Simple and well-studied New Keynesian model: Herbst and Schorfheide [2015].
- Log-linearized simple New Keynesian structure.

Estimation Syntax I

1. Syntax is close to Dynare/Matlab, but not all options are available yet.
2. Setting priors (schematic form):

```
prior!(s::Symbol; shape::<:Distributions, initialvalue, mean, stdev, domain, variance, context)
```

3. Example:

```
prior!(:rhoz, shape=Uniform, mean=0.5, variance=1/12, domain=[0,1])

prior!(:sigr, shape=InverseGamma1, mean=0.5013, variance=0.0687)
```

Estimation Syntax II

1. Running the estimation (Dynare-style call):

```
estimation(datafile='../fsdat_simul.csv', ...);
```

2. In DynareJulia, the interface follows the same logic, with adaptations for Julia.

Estimation Results I

Table 1. Summary Statistics

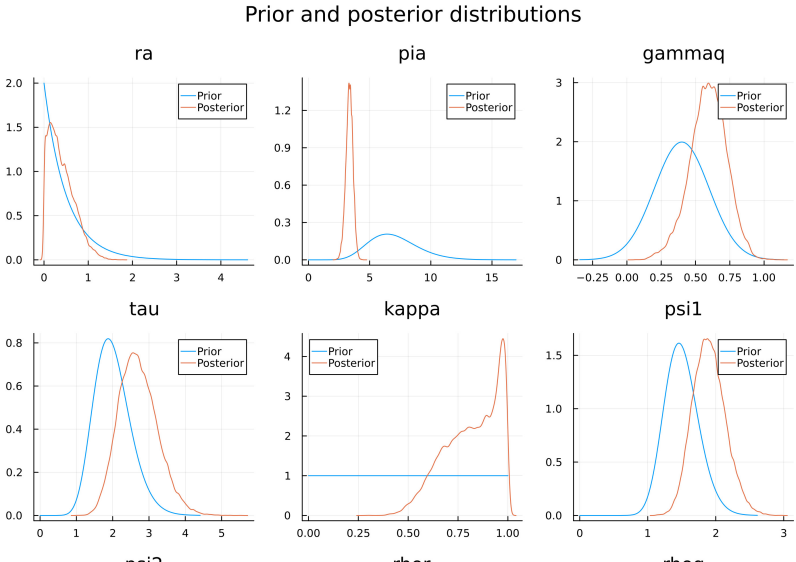
parameters	mean	std	mcse	ess
ra	0.3959	0.2873	0.0121	503.0669
gammaq	0.5922	0.1346	0.0044	955.7836
tau	2.7045	0.5337	0.0087	3737.7201
kappa	0.8171	0.1345	0.0063	444.7632
psi1	1.9036	0.2404	0.0078	963.4403
psi2	0.6851	0.3165	0.0094	1031.3881
rhor	0.7848	0.0287	0.0014	433.1071
rhog	0.9824	0.0065	0.0003	407.5423
rhoz	0.8876	0.0198	0.0010	415.5974
sigr	0.4995	0.2233	0.0094	592.4565
sigg	1.2582	0.6147	0.0158	1733.1290
sigz	0.6402	0.3876	0.0166	664.0536

Estimation Results II

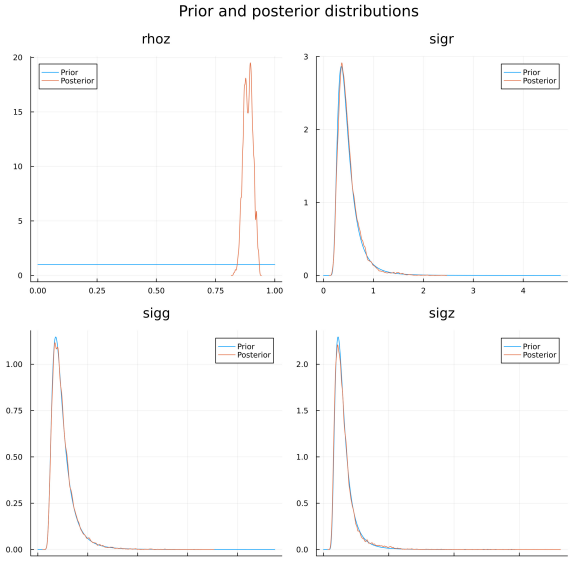
Table 2. Quantile Parameters

parameter	2.5%	25%	50%	75%	97.5%
ra	0.0129	0.1637	0.3398	0.5775	1.0654
pia	2.7432	3.1567	3.3495	3.5396	3.8887
gammaq	0.3098	0.5057	0.5963	0.6843	0.8452
tau	1.7638	2.3225	2.6678	3.0474	3.8503
kappa	0.5358	0.7170	0.8359	0.9390	0.9933
psi1	1.4667	1.7344	1.8925	2.0599	2.4052
psi2	0.1929	0.4567	0.6443	0.8680	1.4201
rhor	0.7188	0.7679	0.7862	0.8049	0.8365
rhog	0.9670	0.9781	0.9849	0.9872	0.9902
rhoz	0.8505	0.8729	0.8880	0.9019	0.9251
sigr	0.2437	0.3538	0.4447	0.5819	1.0792
sigg	0.6070	0.8682	1.1032	1.4550	2.8670
sigz	0.2959	0.4256	0.5446	0.7214	1.6528

Estimation Results III



Estimation Results IV



Performance

1. Compare estimation speed in DynareJulia.
2. Use `BenchmarkTools` in Julia.
3. Estimation runs in a few seconds (two MH chains of 100,000 draws each).

Jordi Gali. *Monetary Policy, Inflation, and the Business Cycle: An Introduction to the New Keynesian Framework and Its Applications*. Princeton University Press, 2 edition, 2015.

Edward P. Herbst and Frank Schorfheide. *Bayesian Estimation of DSGE Models*. Princeton University Press, 2015.

Matteo Iacoviello. House prices, borrowing constraints, and monetary policy in the business cycle. *American Economic Review*, 95(3):739–764, June 2005.