Dynare v4.5 - User Guide for Advanced Topics

Wenli Xu
xuweny87@hotmail.com

This draft: August 2019

Other information

WECHAT: cimers_dsge

宏观经济研学会

微信扫描二维码，关注我的公众号

Contents

List of Figures

Work in Progress!

This is the draft version of the Dynare User Guide for Advanced Topics which is still work in progress! This means three things. First, please read this with a critical eye and send me comments! Are some areas unclear? Is anything plain wrong? Are some sections too wordy, are there enough examples, are these clear? On the contrary, are there certain parts that just click particularly well? How can others be improved? I'm very interested to get your feedback.

The second thing that a work in progress manuscript comes with is a few internal notes. These are mostly placeholders for future work, notes to myself or others of the Dynare development team, or at times notes to you - our readers - to highlight a feature not yet fully stable.

The third thing is regard to the references of my notes. When I had been writing the notes, I copy-paste some contents from many nice papers, so I thanks for the authors's contribution. The reference lists are still work in progress!

Especially, thanks Tommaso Mancini Griffoli for his Dynare User Guide. Meantime, thanks very much for your patience and good ideas. Please write either direclty to myself: xuweny87@hotmail.com, or **preferably on the Dynare Forum** available in the Dynare Forums.

Contacts and Credits

Dynare was originally developed by Michel Juillard in Paris, France. Currently, the development team of Dynare is composed of

- Stphane Adjemian `<stephane.adjemian@univ-lemans.fr>`
- Houtan Bastani `<houtan@dynare.org>`
- Michel Juillard `<michel.juillard@mjui.fr>`
- Frdric Karam `<frederic.karame@univ-lemans.fr>`
- Junior Maih `<junior.maih@gmail.com>`
- Ferhat Mihoubi `<ferhat.mihoubi@cepremap.org>`
- George Perendia `<george@perendia.orangehome.co.uk>`
- Johannes Pfeifer `<jpfeifer@gmx.de>`
- Marco Ratto `<marco.ratto@jrc.ec.europa.eu>`
- Sbastien Villemot `<sebastien@dynare.org>`

Several parts of Dynare use or have strongly benefited from publicly available programs by G. Anderson, F. Collard, L. Ingber, O. Kamenik, P. Klein, S. Sakata, F. Schorfheide, C. Sims, P. Soederlind and R. Wouters.

Finally, the development of Dynare could not have come such a long ways withough an active community of users who continually pose questions, report bugs and suggest new features. The help of this community is gratefully acknowledged.

The email addresses above are provided in case you wish to contact any one of the authors of Dynare directly. We nonetheless encourage you to first use the Dynare forums to ask your questions so that other users can benefit from them as well; remember, almost no question is specific enough to interest just one person, and yours is not the exception!

Chapter 1

Introduction

Welcome to Dynare!

1.1 About this Guide - approach and structure

This User Guide **aims to help you master Dynare**'s main functionalities, from getting started to implementing advanced features. To do so, this Guide is structured around examples and offers practical advice. To root this understanding more deeply, though, this Guide also gives some background on Dynare's algorithms, methodologies and underlying theory. Thus, a secondary function of this Guide is to **serve as a basic primer** on DSGE model solving and Bayesian estimation.

This Guide will focus on the most common or useful features of the program, thus emphasizing **depth over breadth**. The idea is to get you to use 90% of the program well and then tell you where else to look if you're interested in fine tuning or advanced customization.

This Guide is written mainly for an **advanced economist** - like a professor, graduate student or central banker - needing a powerful and flexible program to support and facilitate his or her research activities in a variety of fields. The sophisticated computer programmer, on the one hand, or the specialist of computational economics, on the other, may not find this Guide sufficiently detailed.

We recognize that the "advanced economist" may be either a beginning or intermediate user of Dynare. This Guide is written to accommodate both. If you're **new to Dynare**, we recommend starting with chapters 3 and 4, which introduce the program's basic features to solve (including running impulse response functions) and estimate DSGE models, respectively. To do so, these chapters lead you through a complete hands-on example, which we recommend following from A to Z, in order to "**learn by doing**". Once you have read these two chapters, you will know the crux of Dynare's functionality and (hopefully!) feel comfortable using Dynare for your own work. At that point, though, you will probably find yourself coming back to the User Guide to skim over some of the content in the advanced chapters to iron out details and potential complications you may run into.

If you're instead an **intermediate user** of Dynare, you will most likely find the advanced chapters, **??** and **??**, more appropriate. These chapters cover more advanced features of Dynare and more complicated usage scenarios. The presumption is that you would skip around these chapters to focus on the topics most applicable to your needs and curiosity. Examples are therefore more concise and specific to each feature; these chapters read a bit more like a reference manual.

We also recognize that you probably have had repeated if not active exposure to programming and are likely to have a strong economic background. Thus, a black box solution to your needs is inadequate. To hopefully address this issue, the User Guide goes into some depth in covering the **theoretical underpinnings and methodologies that Dynare follows** to solve and estimate DSGE models. These are available in the "behind the scenes of Dynare" chapters 8 and 5. These chapters can also serve as a **basic primer** if you are new to the practice of DSGE model solving and Bayesian estimation.

Finally, besides breaking up content into short chapters, we've introduced two different **markers** throughout the Guide to help streamline your reading.

- **TIP!** introduces advice to help you work more efficiently with Dynare or solve common problems.
- **NOTE!** is used to draw your attention to particularly important information you should keep in mind when using Dynare.

1.2 What are the advanced topics?

Before we dive into the thick of the "trees", let's have a look at the "forest" from the top ...just what is Dynare?

Dynare is a powerful and highly customizable engine, with an intuitive front-end interface, to solve, simulate and estimate DSGE models.

In slightly less flowery words, it is a pre-processor and a collection of Matlab routines that has the great advantages of reading DSGE model equations written almost as in an academic paper. This not only facilitates the inputting of a model, but also enables you to easily share your code as it is straightforward to read by anyone.

Figure 1.2 gives you an overview of the way Dynare works. Basically, the model and its related attributes, like a shock structure for instance, is written equation by equation in an editor of your choice. The resulting file will be called the .mod file. That file is then called from Matlab. This initiates the Dynare pre-processor which translates the .mod file into a suitable input for the Matlab routines (more precisely, it creates intermediary Matlab or C files which are then used by Matlab code) used to either solve or estimate the model. Finally, results are presented in Matlab. Some more details on the internal files generated by Dynare is given in section **??** in chapter **??**.

Each of these steps will become clear as you read through the User Guide, but for now it may be helpful to summarize **what Dynare is able to do**:

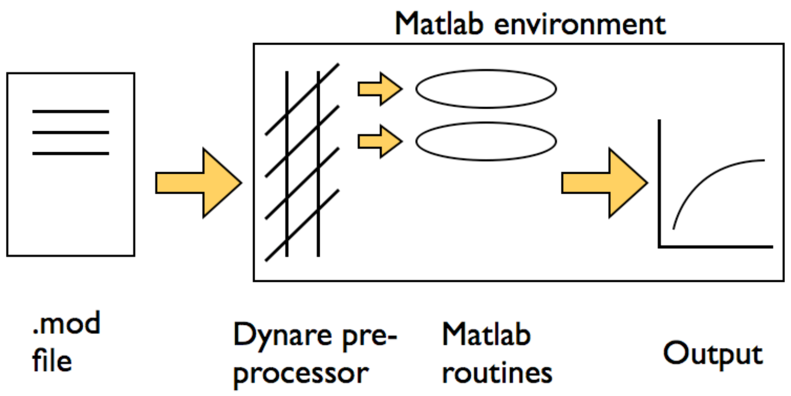


Figure 1.1: The .mod file being read by the Dynare pre-processor, which then calls the relevant Matlab routines to carry out the desired operations and display the results.

- compute the steady state of a model
- compute the solution of deterministic models
- compute the first and second order approximation to solutions of s-tochastic models
- estimate parameters of DSGE models using either a maximum likelihood or a Bayesian approach
- compute optimal policies in linear-quadratic models

1.3 Additional sources of help

While this User Guide tries to be as complete and thorough as possible, you will certainly want to browse other material for help, as you learn about new features, struggle with adapting examples to your own work, and yearn to ask that one question whose answer seems to exist no-where. At your disposal, you have the following additional sources of help:

- **Reference Manual**: this manual covers all Dynare commands, giving a clear definition and explanation of usage for each. The User Guide will often introduce you to a command in a rather loose manner (mainly through examples); so reading corresponding command descriptions in the Reference Manual is a good idea to cover all relevant details.

- **Official online examples**: the Dynare website includes other examples - usually well documented - of .mod files covering models and methodologies introduced in recent papers.

- **Dynare forums**: this lively online discussion forum allows you to ask your questions openly and read threads from others who might have run into similar difficulties.

- **Frequently Asked Questions** (FAQ): this section of the Dynare site emphasizes a few of the most popular questions in the forums.

- **DSGE.net**: this website, run my members of the Dynare team, is a resource for all scholars working in the field of DSGE modeling. Besides allowing you to stay up to date with the most recent papers and possibly make new contacts, it conveniently lists conferences, workshops and seminars that may be of interest.

1.4 Nomenclature

To end this introduction and avoid confusion in what follows, it is worthwhile to agree on a few **definitions of terms**. Many of these are shared with the Reference Manual.

- **Integer** indicates an integer number.

- **Double** indicates a double precision number. The following syntaxes are valid: 1.1e3, 1.1E3, 1.1E-3, 1.1d3, 1.1D3

- **Expression** indicates a mathematical expression valid in the underlying language (e.g. Matlab).
- **Variable name** indicates a variable name. **NOTE!** These must start with an alphabetical character and can only contain other alphabetical characters and digits, as well as underscores (_). All other characters, including accents, and **spaces**, are forbidden.
- **Parameter name** indicates a parameter name which must follow the same naming conventions as above.
- **Filename** indicates a file name valid in your operating system. Note that Matlab requires that names of files or functions start with alphabetical characters; this concerns your Dynare .mod files.
- **Command** is an instruction to Dynare or other program when specified.
- **Options** or optional arguments for a command are listed in square brackets [ ] unless otherwise noted. If, for instance, the option must be specified in parenthesis in Dynare, it will show up in the Guide as [(option)].
- `Typewritten text` indicates text as it should appear in Dynare code.

1.5 v4.5, what's new and backward compatibility

The current version of Dynare - for which this guide is written - is version 4.5. With respect to version 4.4, this new version introduces several important features, as well as improvements, optimizations of routines and bug fixes. The major new features are the following:

- Ramsey policy

  Added command ramsey_model that builds the expanded model with FOC conditions for the planner's problem but doesn't perform any computation. Useful to compute Ramsey policy in a perfect foresight model and for estimation, ramsey_policy accepts multipliers in its variable list and displays results for them.

- Perfect foresight models

  New commands perfect_foresight_setup (for preparing the simulation) and perfect_foresight_solver (for computing it). The old simul command still exist and is now an alias for perfect_foresight_setup + perfect_foresight_solver. It is no longer possible to manipulate by hand the contents of oo_.exo_simul when using simul. People who want to do it must first call perfect_foresight_setup, then do the manipulations, then call perfect_foresight_solver,

  By default, the perfect foresight solver will try a homotopy method if it fails to converge at the first try. The old behavior can be restored with the no_homotopy option,

  New option stack_solve_algo=7 that allows specifying a solve_algo solver for solving the model,

  New option solve_algo that allows specifying a solver for solving the model when using stack_solve_algo=7,

  New option lmmcp that solves the model via a Levenberg-Marquardt mixed complementarity problem (LMMCP) solver,

  New option robust_lin_solve that triggers the use of a robust linear solver for the default solve_algo=4,

  New options tolf and tolx to control termination criteria of solvers,

  New option endogenous_terminal_period to simul,

  Added the possibility to set the initial condition of the (stochastic) extended path simulations with the histval block.

- Optimal simple rules

  Saves the optimal value of parameters to oo_.osr.optim_params,

  New block osr_params_bounds allows specifying bounds for the estimated parameters, New option opt_algo allows selecting different optimizers while the new option optim allows specifying the optimizer options,

The osr command now saves the names, bounds, and indices for the estimated parameters as well as the indices and weights of the variables entering the objective function into M_.osr.

- Forecasts and Smoothing

  The smoother and forecasts take uncertainty about trends and means into account,

  Forecasts accounting for measurement error are now saved in fields of the form HPDinf_ME and HPDsup_ME,

  New fields oo_.Smoother.Trend and oo_.Smoother.Constant that save the trend and constant parts of the smoothed variables,

  new field oo_.Smoother.TrendCoeffs that stores the trend coefficients.

  Rolling window forecasts allowed in estimation command by passing a vector to first_obs,

  The calib_smoother command now accepts the loglinear, prefilter, first_obs and filter_decomposition options.

- Estimation

  New options: logdata, consider_all_endogenous,consider_only_observed, posterior_max_subsample_draws, mh_conf_sig, diffuse_kalman_tol, dirname, nodecomposition

  load_mh_file and mh_recover now try to load chain's proposal density

  New option load_results_after_load_mh that allows loading some posterior results from a previous run if no new MCMC draws areadded,

  New option posterior_nograph that suppresses the generation of graphs associated with Bayesian IRFs, posterior smoothed objects, and posterior forecasts,

  Saves the posterior density at the mode in oo_.posterior.optimization.log_density,

  The filter_covariance option now also works with posterior sampling like Metropolis-Hastings, New option no_posterior_kernel_density to suppress computation of kernel density of posterior objects,

Recursive estimation and forecasting now provides the individual oo_ structures for each sample in oo_recursive_,

The trace_plot command can now plot the posterior density,

New command generate_trace_plots allows generating all trace plots for one chain,

New commands prior_function and posterior_function that execute a user-defined function on parameter draws from the prior/posterior distribution ,

New option huge_number for replacement of infinite bounds with large number during mode_compute,

New option posterior_sampling_method allows selecting the new posterior sampling options: tailored_random_block_metropolis_hastings (Tailored randomized block (TaRB) Metropolis-Hastings), slice (Slice sampler), independent_metropolis_hastings (Independent Metropolis-Hastings),

New option posterior_sampler_options that allow controlling the options of the posterior_sampling_method; its scale_file-option pair allows loading the _mh_scale.mat-file storing the tuned scale factor from a previous run of mode_compute=6,

New option raftery_lewis_diagnostics that computes Raftery/Lewis (1992) convergence diagnostics,

New option fast_kalman_filter that provides fast Kalman filter using Chandrasekhar recursions as described in Ed Herbst (2015),

The dsge_var option now saves results at the posterior mode into oo_.dsge_var,

New option smoothed_state_uncertainty to provide the uncertainty estimate for the smoothed state estimate from the Kalman smoother,

New prior density: generalized Weibull distribution,

Option mh_recover now allows continuing a crashed chain at the last save mh-file, New option nonlinear_filter_initialization for the estimation command. Controls the initial covariance matrix of the state variables in nonlinear filters.

The conditional_variance_decomposition†option now displays output and stores it as a LaTeX-table when the TeX option is invoked,

The use_calibration to estimated_params_init now also works with ML,

Improved initial estimation checks.

- Steady state

  The default solver for finding the steady state is now a trust-region solver (can be triggered explicitly with option solve_algo=4),

  New options tolf and tolx to control termination criteria of solver,

  The debugging mode now provides the termination values in steady state finding.

- Stochastic simulations

  New options nodecomposition,

  New option bandpass_filter to compute bandpass-filtered theoretical and simulated moments,

  New option one_sided_hp_filter to compute one-sided HP-filtered simulated moments,

  stoch_simul displays a simulated variance decomposition when simulated moments are requested,

  stoch_simul saves skewness and kurtosis into respective fields of oo_ when simulated moments have been requested,

  stoch_simul saves the unconditional variance decomposition in oo_.variance_decomposition

  New option dr_display_tol that governs omission of small terms in display of decision rules,

  The stoch_simul command now prints the displayed tables as LaTeX code when the new TeX option is enabled, The loglinear option now works with lagged and leaded exogenous variables like news shocks,

New option spectral_density that allows displaying the spectral density of (filtered) endogenous variables,

New option contemporaneous_correlation that allows saving contemporaneous correlations in addition to the covariances.

- Identification

  New options diffuse_filter and prior_trunc,

  The identification command now supports correlations via simulated moments,

- Sensitivity analysis

  New blocks irf_calibration and moment_calibration,

  Outputs LaTeX tables if the new TeX option is used,

  New option relative_irf to irf_calibration block.

- Conditional forecast

  Command conditional_forecast now takes into account histval block if present.

- Shock decomposition

  New option colormap to shocks_decomposition for controlling the color map used in the shocks decomposition graphs,

  shocks_decomposition now accepts the nograph option,

  New command realtime_shock_decomposition that for each period T= [presample,...,nobs] allows computing the:

  realtime historical shock decomposition Y(t|T), i.e. without observing data in [T+1,...,nobs]

  forecast shock decomposition Y(T+k|T)

  realtime conditional shock decomposition Y(T+k|T+k)-Y(T+k|T)

New block shock_groups that allows grouping shocks for the shock_decomposition and realtime_shock_decomposition commands,

New command plot_shock_decomposition that allows plotting the results from shock_decomposition and realtime_shock_decomposition for different vintages and shock groupings.

- Macroprocessor

  Can now pass a macro-variable to the #include macro directive,

  New preprocessor flag -I, macro directive #includepath, and dynare config file block [paths] to pass a search path to the macroprocessor to be used for file inclusion via #include.

- Command line

  New option onlyclearglobals (do not clear JIT compiled functions with recent versions of Matlab),

  New option minimal_workspace to use fewer variables in the current workspace,

  New option params_derivs_order allows limiting the order of the derivatives with respect to the parameters that are calculated by the preprocessor,

  New command line option mingw to support the MinGW-w64 C/C++ Compiler from TDM-GCC for use_dll.

- dates/dseries/reporting classes

  New methods abs, cumprod and chain,

  New option tableRowIndent to addTable,

  Reporting system revamped and made more efficient, dependency on matlab2tikz has been dropped.

- Optimization algorithms

  mode_compute=2 Uses the simulated annealing as described by Corana et al. (1987), mode_compute=101 Uses SOLVEOPT as described by Kuntsevich and Kappel (1997),

mode_compute=102 Uses simulannealbnd from Matlab's Global Optimization Toolbox (if available),

New option silent_optimizer to shut off output from mode computing/optimization,

New options verbosity and SaveFiles to control output and saving of files during mode computing/optimization.

- LaTeX output

  New command write_latex_original_model,

  New option write_equation_tags to write_latex_dynamic_model that allows printing the specified equation tags to the generate LaTeX code,

  New command write_latex_parameter_table that writes the names and values of model parameters to a LaTeX table,

  New command write_latex_prior_table that writes the descriptive statistics about the prior distribution to a LaTeX table,

  New command collect_latex_files that creates one compilable LaTeX file containing all TeX-output.

- Misc.

  Provides 64bit preprocessor,

  Introduces new path management to avoid conflicts with other toolboxes,

  Full compatibility with Matlab 2014b's new graphic interface,

  When using model(linear), Dynare automatically checks whether the model is truly linear,

  usedll: the msvc option now supports normcdf, acosh, asinh, and atanh,

  New parallel option NumberOfThreadsPerJob for Windows nodes that sets the number of threads assigned to each remote MATLAB/Octave run, Improved numerical performance of schur_statespace_transformation for very large models,

The all_values_required option now also works with histval,

Add missing horizon option to ms_forecast,

BVAR now saves the marginal data density in oo_.bvar.log_marginal_data_density and stores prior and posterior information in oo_.bvar.prior and oo_.bvar.posterior.

Chapter 2

Installing Dynare

2.1 Dynare versions

Dynare runs on both **MATLAB** and **GNU Octave**.

There used to be versions of Dynare for **Scilab** and **Gauss**. Development of the Scilab version stopped after Dynare version 3.02 and that for Gauss after Dynare version 1.2.

This User Guide will exclusively **focus on Dynare version 4.0 and later**.

You may also be interested by another program, **Dynare++**, which is a standalone C++ program specialized in computing k-order approximations of dynamic stochastic general equilibrium models. Note that Dynare++ is distributed along with Dynare since version 4.1. See the Dynare++ webpage for more information.

2.2 System requirements

Dynare can run on Microsoft Windows, as well as Unix-like operating systems, in particular GNU/Linux and Mac OS X. If you have questions about the support of a particular platform, please ask your question on **Dynare forums**.

To run Dynare, it is recommended to allocate at least 256MB of RAM to the platform running Dynare, although 512MB is preferred. Depending on the type of computations required, like the very processor intensive Metropolis

Hastings algorithm, you may need up to 1GB of RAM to obtain acceptable computational times.

2.3 Installing Dynare

Please refer to the section entitled "Installation and configuration" in the Dynare reference manual.

2.4 MATLAB particularities

A question often comes up: what special MATLAB toolboxes are necessary to run Dynare? In fact, no additional toolbox is necessary for running most of Dynare, except maybe for optimal simple rules (see chapter 6), but even then remedies exist (see the Dynare forums for discussions on this, or to ask your particular question). But if you do have the 'optimization toolbox' installed, you will have additional options for solving for the steady state (solve_algo option) and for searching for the posterior mode (mode_compute option), both of which are defined later.

Chapter 3

Solving Steady State - basics

Basically, you have to remove all the time indices (t-1,t,t+1,...) on the endogenous and exogenous variables in all the equations, and express the endogenous variables as functions of the exogenous variables and parameters. This cannot be done analytically in all models, and often you will have to resort on a numerical solver to obtain the steady state (for given values of the parameters and exogenous variables). This is what Dynare does if you do not provide a closed form expression for the steady state. In small models (basic RBC or NK models), it is possible to obtain an analytical expression for the steady state.

众所周知，DSGE 模型的稳态值较为难解，尤其对于复杂的非线性模型。通常，我们利用 DSGE 模型来进行研究，80% 以上的时间可能都花费在了求解内生变量的稳态值上。Doing so borders on a form of art, and luck is unfortunately part of the equation. 因此，下面的方法/技巧/步骤可能对大家有一些帮助。

3.1 Analytical Solutions

首先，我们来看一个简单的模型。

模型的均衡系统为：

$$C_t^{\sigma}\theta N_t^{\phi} = w_t \tag{3.1}$$

$$E_t\frac{C_{t+1}^{\sigma}}{C_t^{\sigma}} = \beta E_t(R_{t+1} + (1-\delta)) \tag{3.2}$$

$$E_t\frac{C_{t+1}^{\sigma}}{C_t^{\sigma}} = \beta E_t(1 + r_t) \tag{3.3}$$

$$K_{t+1} = I_t + (1-\delta)K_t \tag{3.4}$$

$$\alpha A_t K_t^{\alpha-1} N_t^{1-\alpha} = R_t \tag{3.5}$$

$$(1-\alpha)A_t K_t^{\alpha} N_t^{-\alpha} = w_t \tag{3.6}$$

$$Y_t = A_t K_t^{\alpha} N_t^{1-\alpha} \tag{3.7}$$

$$Y_t = C_t + I_t \tag{3.8}$$

$$lnA_t = \rho lnA_{t-1} + \epsilon_t \tag{3.9}$$

（3.1）是劳动供给方程。（3.2）是投资 -储蓄均衡，也就是资本欧拉方程。（3.3）也是投资 -储蓄均衡，是债券欧拉方程。（3.4）是资本积累方程。（3.5）是资本需求方程。（3.6）是劳动需求方程。（3.7）是生产函数。（3.8）是产品市场出清。（3.9）是生产率冲击。其中，有一个前看变量 C_{t+1}，两个状态变量 K_t A_t。模型均衡系统包含 9 个内生变量 $Y_t, C_t, N_t, K_t, I_t, w_t, R_t, r_t, A_t$，9 个方程。

上文已经定义了模型的均衡，下面的内容我们来定义并解出模型的稳态。稳态即内生变量在每一期都相等，$E_t x_{t+1} = x_t = x_{t-1} = x_{ss}$。

首先，在稳态时，去掉所有变量的时间下标。$E_t\epsilon_t = 0$，因此，从生产率演化方程（3.9）可以得到 $A = 1$。且在稳态时，$K_{t+1} = K_t = K, C_{t+1} = C_t = C$。

我们从欧拉方程（3.2）和（3.3）开始，最容易解出稳态。

(1) 从（3.2）和（3.3）可知，

$$\begin{aligned}\beta(R+1-\delta)&=1\\ \Leftrightarrow R&=\frac{1}{\beta}+\delta-1\end{aligned}\tag{3.10}$$

$$\begin{aligned}\beta(1+r)&=1\\ \Leftrightarrow r&=\frac{1}{\beta}-1\end{aligned}\tag{3.11}$$

(2) 从资本需求函数（3.5）可以解出资本 -劳动比率的稳态：

$$\begin{aligned}R&=\alpha(\frac{K}{N})^{\alpha-1}\\ \Leftrightarrow \frac{K}{N}&=(\frac{R}{\alpha})^{\frac{1}{\alpha-1}}\\ \Leftrightarrow \frac{K}{N}&=(\frac{\alpha}{\frac{1}{\beta}+\delta-1})^{\frac{1}{1-\alpha}}\end{aligned}\tag{3.12}$$

注意，（3.12）的第三行是将（3.10）的结果代入第二行得到。

(3) 根据（3.12）的资本 -劳动稳态比率，代入（3.6）的劳动需求方程，得到

$$\begin{aligned}w&=(1-\alpha)(\frac{K}{N})^{\alpha}\\ \Leftrightarrow w&=(1-\alpha)(\frac{\alpha}{\frac{1}{\beta}+\delta-1})^{\frac{\alpha}{1-\alpha}}\end{aligned}\tag{3.13}$$

(4) 从资本积累方程（3.4）可知，$I=\delta K$，而从生产函数（3.7）可知 $Y=(\frac{K}{N})^{\alpha}N$。

(5) 将第四步中的两式代入产品市场出清方程（3.8），得到

$$\begin{aligned}(\frac{K}{N})^{\alpha}N&=C+\delta K\\ \Leftrightarrow \frac{C}{N}&=(\frac{K}{N})^{\alpha}-\delta\frac{K}{N}\end{aligned}\tag{3.14}$$

(6) 将上式（3.14）和劳动需求方程（3.13）联立起来，而 $\frac{K}{N}, w$ 是已知的，由此得到

$$\begin{cases} C^{\sigma}\theta N_t^{\phi} = w \\ \dfrac{C}{N} = (\dfrac{K}{N})^{\alpha} - \delta\dfrac{K}{N} \end{cases} \tag{3.15}$$

通过上式（3.15）可以解出消费和劳动的稳态值 C, N。一旦得到劳动的稳态 N，就可以将其代入资本 -劳动比率稳态（3.12）解出资本的稳态 K，进而得到产出和投资的稳态 Y, I。

对于小型 DSGE 模型来说，我们可以解出每个内生变量的解析解，因此，可以得到精确的稳态值。但是，dynare 对初值比较敏感，此时，还是有可能会报错。那么，我们就需要在 steady 命令后加一些最优化选项：

- $solve\_algo = 0$: 用 Matlab 最优化工具箱中的 fsolve 函数
- $solve\_algo = 1$: 用 Dynare 自带非线性方程算法
- $solve\_algo = 2$: 将模型分割成递归模块，然后逐一解出每个模块
- $solve\_algo = 3$: 用 Sims 算法。这是默认选项

3.2 Adding new variable one by one

解析解法一般针对小规模 DSGE 模型才适用。对于中等规模模型和大规模模型，很难解出解析解。幸运的是，DSGE 模型十分模块化，复杂模型基本都是基于基准模型（小规模）扩展而来，因此，我们可以逐步引入我们感兴趣的部门或者因素来扩展至新的模型，例如在 RBC 模型中，先引入企业的垄断竞争结构，然后再引入中间产品企业的价格粘性，这样就形成了 NK 模型。

这种一步一步增加因素的思想也为我们求解稳态提供了思路。

也就是说，我们再求解稳态时，也遵循 —— 1—— 2——···——的路径。

例如，我们在上一节的 RBC 模型中，引入垄断竞争因素，即均衡系统变为：

$$C_t^{\sigma}\theta N_t^{\phi} = w_t \tag{3.16}$$

$$E_t\frac{C_{t+1}^{\sigma}}{C_t^{\sigma}} = \beta E_t(R_{t+1} + (1-\delta)) \tag{3.17}$$

$$E_t\frac{C_{t+1}^{\sigma}}{C_t^{\sigma}} = \beta E_t(1+r_t) \tag{3.18}$$

$$K_{t+1} = I_t + (1-\delta)K_t \tag{3.19}$$

$$\alpha MC_tA_tK_t^{\alpha-1}N_t^{1-\alpha} = R_t \tag{3.20}$$

$$(1-\alpha)MC_tA_tK_t^{\alpha}N_t^{-\alpha} = w_t \tag{3.21}$$

$$Y_t = A_tK_t^{\alpha}N_t^{1-\alpha} \tag{3.22}$$

$$Y_t = C_t + I_t \tag{3.23}$$

$$MC_t = \frac{\xi_t - 1}{\xi_t} \tag{3.24}$$

$$lnA_t = \rho lnA_{t-1} + \epsilon_t \tag{3.25}$$

$$ln\xi_t = (1-\rho_\xi)ln\xi + \rho_\xi ln\xi_{t-1} + \epsilon_\xi \tag{3.26}$$

该均衡系统由 10 个内生变量 $Y_t, C_t, N_t, K_t, I_t, w_t, R_t, r_t, MC_t, A_t, \xi_t$，10 个方程。

稳态

上文已经定义了模型的均衡，下面的内容我们来定义并解出模型的稳态。稳态即内生变量在每一期都相等，$E_tx_{t+1} = x_t = x_{t-1} = x_{ss}$。首先，在稳态时，去掉所有变量的时间下标。$E_t\epsilon_t = 0$，因此，从生产率演化方程（3.25）可以得到 $A = 1$。且在稳态时，$K_{t+1} = K_t = K, C_{t+1} = C_t = C$。

我们从欧拉方程（3.17）和（3.18）开始，最容易解出稳态。

(1) 从（3.17）和（3.18）可知，

$$\begin{aligned} &\beta(R+1-\delta) = 1 \\ &\Leftrightarrow R = \frac{1}{\beta} + \delta - 1 \end{aligned} \tag{3.27}$$

$$\begin{aligned}\beta(1+r)&=1\\ \Leftrightarrow r&=\frac{1}{\beta}-1\end{aligned}\tag{3.28}$$

(2) 从资本需求函数（3.20）可以解出资本 -劳动比率的稳态：

$$\begin{aligned}R&=\alpha MC(\frac{K}{N})^{\alpha-1}\\ \Leftrightarrow \frac{K}{N}&=(\frac{R}{\alpha MC})^{\frac{1}{\alpha-1}}\\ \Leftrightarrow \frac{K}{N}&=(\frac{\alpha MC}{\frac{1}{\beta}+\delta-1})^{\frac{1}{1-\alpha}}\end{aligned}\tag{3.29}$$

注意，（3.27）的第三行是将（3.25）的结果代入第二行得到。

(3) 根据（3.27）的资本 -劳动稳态比率，代入（3.21）的劳动需求方程，得到

$$\begin{aligned}w&=(1-\alpha)(\frac{K}{N})^{\alpha}\\ \Leftrightarrow w&=(1-\alpha)(\frac{\alpha MC}{\frac{1}{\beta}+\delta-1})^{\frac{\alpha}{1-\alpha}}\end{aligned}\tag{3.30}$$

(4) 从资本积累方程（3.19）可知，$I=\delta K$，而从生产函数（3.22）可知 $Y=(\frac{K}{N})^{\alpha}N$。

(5) 将第四步中的两式代入产品市场出清方程（3.23），得到

$$\begin{aligned}(\frac{K}{N})^{\alpha}N&=C+\delta K\\ \Leftrightarrow \frac{C}{N}&=(\frac{K}{N})^{\alpha}-\delta\frac{K}{N}\end{aligned}\tag{3.31}$$

(6) 将上式（3.29）和劳动需求方程（3.16）联立起来，而 $\frac{K}{N},w$ 是已知的，由此得到

$$\begin{cases}C^{\sigma}\theta N_t^{\phi}=w\\ \frac{C}{N}=(\frac{K}{N})^{\alpha}-\delta\frac{K}{N}\end{cases}\tag{3.32}$$

通过上式（3.29）可以解出消费和劳动的稳态值 C, N。一旦得到劳动的稳态 N，就可以将其代入资本 -劳动比率稳态（3.27）解出资本的稳态 K，进而得到产出和投资的稳态 Y, I。

(7) 而边际成本稳态 MC 可以通过价格加成冲击 (3.26) 和 (3.24) 来求解出。

再例如，我们在上述扩展模型中引入价格粘性因素。我们得到如下均衡系统：

$$C_t^\sigma \theta N_t^\phi = w_t \tag{3.33}$$

$$E_t \frac{C_{t+1}^\sigma}{C_t^\sigma} = \beta E_t (R_{t+1} + (1-\delta)) \tag{3.34}$$

$$E_t \frac{C_{t+1}^\sigma}{C_t^\sigma} = \beta E_t (1 + r_t) \tag{3.35}$$

$$K_{t+1} = I_t + (1-\delta) K_t \tag{3.36}$$

$$\frac{\alpha}{1-\alpha} \frac{N_t}{K_t} = \frac{R_t}{w_t} \tag{3.37}$$

$$Y_t v_t^p = A_t K_t^\alpha N_t^{1-\alpha} \tag{3.38}$$

$$Y_t = C_t + I_t \tag{3.39}$$

$$MC_t = \frac{w_t}{(1-\alpha) A_t (\frac{K_t}{N_t})^\alpha} \frac{\xi - 1}{\xi} \tag{3.40}$$

$$lnA_t = \rho lnA_{t-1} + \epsilon_t \tag{3.41}$$

$$v_t^p = (1-\phi)(1+\pi_t^\star)^{-\xi}(1+\pi_t)^\xi + (1+\pi_t)^\xi \phi v_{t-1}^p \tag{3.42}$$

$$(1+\pi_t)^{1-\xi} = (1-\phi)(1+\pi_t^\star)^{1-\xi} + \phi \tag{3.43}$$

$$1+\pi_t^\star = \frac{\xi}{\xi-1}(1+\pi_t)\frac{x_{1t}}{x_{2t}} \tag{3.44}$$

$$x_{1t} = C_t^{-\sigma} MC_t Y_t + \phi\beta E_t(1+\pi_{t+1})^{\xi} x_{1,t+1} \tag{3.45}$$

$$x_{2t} = C_t^{-\sigma} Y_t + \phi\beta E_t(1+\pi_{t+1})^{\xi-1} x_{2,t+1} \tag{3.46}$$

$$r_t = (1-\rho_r)r + \rho_r r_{t-1} + (1-\rho_r)(\phi_\pi(\pi_t-\pi) + \phi_y(lnY_t - lnY)) + \epsilon_t \tag{3.47}$$

该均衡系统由 15 个内生变量 $Y_t, C_t, N_t, K_t, I_t, w_t, R_t, r_t, MC_t, A_t, v_t^p, \pi_t, \pi_t^\star, x_{1t}, x_{2t}$，15 个方程。

稳态

首先，我们可以由（3.47）解出 π；

然后，将 π 带入（3.43），解出 $\pi^\star$；

由（3.45）和（3.46）可知，

$$MC = \frac{x_1}{x_2}\frac{1-\phi\beta(1+\pi)^{\xi}}{1-\phi\beta(1+\pi)^{\xi-1}}$$

一旦知道 MC 的稳态，我们就可以返回（3.29）解出 $\frac{K}{N}$ 的稳态，然后依次解出其它内生变量的稳态值。

练习：引入财政政策的的模型。

3.3 Calibrating Steady State

校准稳态？我们似乎只听说过校准参数。但是其实两者是一样的。

参数校准貌似是最为人们“嫌弃”，但又不得不用的方法。但是，我想说的是，这种方法在计量经济学会创会时，那些计量经济学的祖师爷们就开始在用了，而且它是所有后续估计方法的基石。

Kydland and Prescott（1982）用了一种方法将他们所分析的 DSGE 模型转换成了一种经验分析。他们所使用的方法就是校准练习。DeJong and Dave（2007）将这种方法称为 MM/ML/BL 的基础。

我们知道，在 Lucas（1976）、Sims（1980）的批判下，概率论方法的估计与假设检验对结构模型不适用。因此，校准练习的目标是为了用参数化的

结构模型来探讨一些特定的定量问题。Kydland and Prescott（1991，SJE；1996，JEP）追溯了校准联系作为一种经验研究方法的历史起源，大家可以去看看。在 Econometrica 的创刊语中，Frisch（1933a）指出：

"……[计量经济学会] 要促进那些旨在统一理论定量方法与经验定量方法的经济研究，并鼓励那些具有建设性和严谨的思考……一切旨在进一步促进经济学理论与经验相统一的活动都属于计量经济学会感兴趣的范围。"

而且，Frisch（1933b）还利用微观数据来校准了生产技术参数，并用来研究经济周期冲击的传播机制。我们都承认这种方法存在一些缺陷，尤其是关于模型误设方面。但是，正如 Prescott（1986）指出：

"所有构建在理论框架之上的模型必然是一种高度抽象的东西。因此，它们肯定存在模型误设问题，统计假设检验会拒绝它们。但是，这并不意味者我们不能从这些理论定量分析中学到一些有用的信息。"

总之，校准练习肯定是一种经验工具（DeJong and Dave，2007）。Kydland and Prescott（1996，JEP）指出："一定要注意，校准并不是为了得到某些参数的大小，即它不是估计。"因此，在校准 DSGE 参数的时候，通常来源于两个方面：一是长期均值；二是微观证据的经验结果（例如，Cooley and Prescott （1995）就利用了 Ghez and Becker（1975）的微观证据来校准参数）。

后一种方式，我们经常用，一般在前人的研究结果中寻找。但是，金融危机之后，越来越多的学者，在文章中利用微观数据/微观计量模型来估计关键特征的相关参数。

The steady state is a mapping between endogenous variables and the parameters of the model. To calibrate a model, we somehow revert this mapping, expressing parameters as function of steady state ratios. For instance, in the Solow model, we have the law of motion for the physical capital stock:

$$K_{t+1} = (1-\delta)K_t + I_t$$

at the steady state we must have:

$$K^{\star} = (1-\delta)K^{\star} + I^{\star}$$

$$\delta K^{\star} = I^{\star}$$

hence we know that the depreciation rate must be such that:

$$\delta = \frac{I^\star}{K^\star}$$

If we have long time series on physical capital and investment we can then calibrate the depreciation rate with average of the investment/capital ratio. Not all the parameters can be deduced from steady state considerations, since not all parameters will affect the steady state (rigidity parameters for instance). In this case we will have to consider other moments to calibrate these parameters.

下面，我们来看看利用长期均值/数据矩如何得到模型内生变量的稳态值。

例如，DSGE 建模与编程入门（41）：金融中介

均衡系统为：

$$\frac{1}{c_t} = \frac{\gamma}{d_t} + \beta E_t \frac{1+r_t^d}{c_{t+1}} \tag{3.48}$$

$$h_t l_t^\phi = \frac{w_t}{c_t} \tag{3.49}$$

$$y_t = a_t k_{t-1}^\alpha l_t^{1-\alpha} \tag{3.50}$$

$$r_t^k = \alpha a_t k_{t-1}^{\alpha-1} l_t^{1-\alpha} \tag{3.51}$$

$$w_t = (1-\alpha) a_t k_{t-1}^\alpha l_t^{-\alpha} \tag{3.52}$$

$$k_t = m_t^c + m_t^s \tag{3.53}$$

$$m_t^s = (1-\kappa) ABS_t \tag{3.54}$$

$$(1-\kappa) E_t (1 + r_{t+1}^k - \delta) = 1 + r_t^a \tag{3.55}$$

$$m_t^c + ABS_t = d_t + n_t \tag{3.56}$$

$$n_t = \hat{\omega} m_t^c \tag{3.57}$$

$$\omega_t = \hat{\omega}^{1-\rho_\omega} \omega_{t-1}^{\rho_\omega} \left(\frac{i_t / i_{t-1}}{gdp_t / gpd_{t-1}} \right)^{\Phi(1-\rho_\omega)} e^{\epsilon_{\omega,t}} \tag{3.58}$$

$$C(g_t) = -p_1 ln(1 + p_2 g_t) \tag{3.59}$$

$$F(\frac{ABS_t}{m_t^c}) = \frac{\phi}{2} \left(\frac{ABS_t}{m_t^c} - \frac{ABS}{m^c} \right)^2 \tag{3.60}$$

$$1 + C^{'} = \beta E_t \frac{c_t}{c_{t+1}} (1 + r_t^d) \tag{3.61}$$

$$(1 + s_t) + (1 - \omega_t) C^{'} - F^{'} \frac{ABS_t}{(m_t^c)^2} = \beta E_t \frac{c_t}{c_{t+1}} (1 + r_{t+1}^k - \delta) \tag{3.62}$$

$$1 + C^{'} + F^{'} \frac{1}{m_t^c} = \beta E_t \frac{c_t}{c_{t+1}} (1 - P_{t+1})(1 + r_t^a) \tag{3.63}$$

$$s_t = \rho_s s_{t-1} + \epsilon_{s,t} \tag{3.64}$$

$$P_t = \rho_p P_{t-1} + \epsilon_{p,t} \tag{3.65}$$

$$lna_t = \rho_a lna_{t-1} + \epsilon_{a,t} \tag{3.66}$$

$$lnh_t = (1 - \rho_h) h + \rho_h h_{t-1} + \epsilon_{h,t} \tag{3.67}$$

$$y_t = c_t + i_t + C(g_t) + F(\frac{ABS_t}{m_t^c}) + s_t m_t^c + \kappa ABS_t \tag{3.68}$$

$$gdp_t = c_t + i_t \tag{3.69}$$

$$share_t = \frac{m_t^s}{k_t} \tag{3.70}$$

$$leverage_t = \frac{m_t^c + ABS_t}{n_t} \tag{3.71}$$

使用的宏观经济指标包括扣除名义 GDP、GDP 平减指数、名义净出口、名义消费总额、CPI、就业人数、影子银行贷款总额、全社会融资总额、国债回购利率、劳动收入份额、固定资产投资、商业银行资本充足率等。频率全部为季度数据。

劳动收入份额数据的时期为 1996 年一季度 -2017 年第四季度。固定资产投资数据的时期为 1992 年一季度 -2017 年四季度。GDP 平减指数、名义净出口、名义消费总额、CPI、就业人数、影子银行贷款总额、全社会融资总额的时期为 2002 年一季度 -2017 年四季度。以上数据全部来自源于 Chang et al.(2015)、Higgins and Tao Zha(2016) 的中国宏观经济数据库。名义 GDP 的时期为 1992 年一季度 -2017 年四季度，数据来源于中国统计局数据库。国债回购利率的时期为 2000 年一季度 -2017 年四季度，数据来源于中经数据库。商业银行资本充足率的时期为 2009 年一季度 -2017 年四季度，数据来源于中国银监会网站。

以上数据中，名义值全部利用相关价格指数进行平减处理得到实际值。由于本文模型经济并不包含国外部门，因此，从 GDP 中扣除净出口得到本文使用的 GDP 观测值。利用影子银行贷款总额除以全社会融资总额得到本文使用的影子银行份额观测值。除 GDP 观测值、影子银行份额之外，实际消费、就业人数、国债回购利率用于模型部分参数的贝叶斯估计。且其中，GDP、消费、就业人数、影子银行份额全部经过季节调整、对数差分和去均值处理，国债回购利率则经过季节调整和去均值处理。

其他宏观经济变量的数据则用于模型参数的校准。

3.4 The External Files of Solving Steady State

参见 DSGE 建模与编程：中等规模 DSGE 及其 code

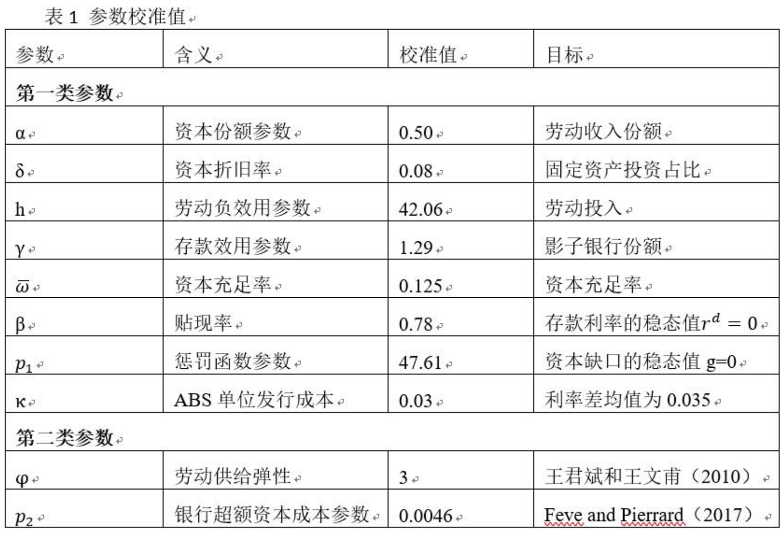


表 1 参数校准值

参数	含义	校准值	目标
第一类参数			
α	资本份额参数	0.50	劳动收入份额
δ	资本折旧率	0.08	固定资产投资占比
h	劳动负效用参数	42.06	劳动投入
γ	存款效用参数	1.29	影子银行份额
$\overline{\omega}$	资本充足率	0.125	资本充足率
β	贴现率	0.78	存款利率的稳态值$r^d = 0$
p_1	惩罚函数参数	47.61	资本缺口的稳态值 g=0
κ	ABS 单位发行成本	0.03	利率差均值为 0.035
第二类参数			
φ	劳动供给弹性	3	王君斌和王文甫（2010）
p_2	银行超额资本成本参数	0.0046	Feve and Pierrard（2017）

Chapter 4

Occasionally Binding Constraints

2008 年金融危机后，全球央行都降低利率来抵抗经济下滑，这也使得各国的名义利率下降/接近零下限。于此同时，由于资产价格的下降，家庭、企业、金融机构也受到其自身融资约束的限制。而 ZLB 和融资约束则是 OBCs 最著名的例子。

Occasionally binding constraints are part of the economic landscape: for instance recent experience with the global financial crisis has highlighted the gravity of the lower bound constraint on interest rates; emerging economies have been affected by sudden stops, or slowdowns in private capital inflows, that occur from time to time; mortgagors are subject to more stringent borrowing conditions when credit growth has been excessive or there is a downturn in the economy; workers have an aversion to taking pay cuts, a problem that is more acute during economic downturns.

The problem of modelling occasionally binding constraints is not a new one. The RBC and DSGE literatures have addressed these problems using a range of different methodologies. These include: eternally binding constraints, smooth approximation of occasionally binding constraints, the extended path algorithm, piecewise linear methods, anticipated shocks, global solution methods, and regime-switching methods.

4.1 OBCs in Macroeconomics

OBCs 意味着模型具有非线性性质，因此，这也为 DSGE 模拟和估计带来计算方法的挑战。我在过往的推文中多次阐述过 DSGE 中的 OBCs 特征，例如 DSGE 建模与编程入门 (44):ZLB 与 News Shocks、【新增 Dynare code】ZLB+OBC(Occasionally Binding Constraints)、DSGE 建模与编程入门 (36): 金融摩擦（三）等。

One of the simplest and most straightforward approaches to handling occasionally binding constraints is to assume that the constraint is binding at all times. Following Brzoza-Brzezina et al. (2015), we refer to this approach as eternally binding constraints (EBC). Eternally binding constraints are commonly used in the modelling of borrowing and collateral constraints as demonstrated by Iacoviello (2005) and Kiyotaki and Moore (1997). In order to pursue this strategy, the Blanchard-Kahn conditions need to hold when the constraint is binding. While this is a reasonable strategy to pursue for some problems, like the modelling of borrowing constraints, it is not readily applicable to problems like the zero lower bound, which does not in isolation, satisfy the Blanchard-Kahn conditions and is genuinely an occasionally binding constraint that would be poorly approximated as an eternally binding constraint.

Others have tried to solve occasionally binding constraints using smooth approximations. In particular Den Haan and De Wind (2012) propose using an exponential penalty function to prevent negative asset positions in a Deaton-type model. Brzoza-Brzezina et al. (2015) use the same type of penalty function to model occasionally binding collateral constraints.Kim and Ruge-Murcia (2011) use Linex adjustment costs to model downward nominal wage rigidities. While the use of smooth functions permits the use of derivatives, the effective application of these methods requires non-linear solution techniques in order to preserve the non-linearity and asymmetry introduced by the occasionally-binding constraint. When it comes to implementability, smooth approximations work well when the constraint is ”soft”, that is agents can violate the constraint but at some cost.

A range of non-linear solution techniques have been used to solve models with occasionally binding constraints. One of the oldest and most commonly used methods is the extended path algorithm due to Fair and Taylor (1983). The extended path algorithm is a certainty equivalent method that solves for the paths of all variables numerically, assuming perfect foresight at each point in time. It has been used by Coenen et al. (2007), Adjemian and Juillard (2010) and Braun and Kober (2011) to account for the zero lower bound on interest rates. A stochastic version by Adjemian and Juillard (2013) has been used to model irreversible investment in a DSGE model. While this method can be applied to a range of different occasionally binding constraints, certainty equivalence means agents' behavior does not change in the vicinity of the constraint binding. In fact the constraint only impacts agents' behavior when it binds.

Piecewise linear methods have been developed by Jung et al. (2005), Cagliarini and Kulish(2013) and Guerrieri and Iacoviello (2015b) to model occasionally binding constraints. Guerrieri and Iacoviello (2015b), Eggertsson and Woodford (2003) and Braun et al. (2015) have used these methods to enforce the lower bound constraint on interest rates. Guerrieri and Iacoviello (2015a) and Akinci and Queralt (2014) have used piecewise linear methods to account for occasionally binding collateral constraints in DSGE models. Amano and Gnocchi (2017) model downward nominal wage rigidities as a non-negativity constraint on wage inflation and solve the model using piecewise linear methods. While this method is simple and applicable to a range of different occasionally binding constraints, it suffers from some of the same problems as the extended path algorithm, namely that agents' behavior does not change in the vicinity of the constraint binding.

Occasionally binding constraints can also be imposed through the addition of shocks. More specifically Holden and Paetz (2012) have proposed the use of news shocks to impose borrowing constraints and the lower bound on interest rates, while Linde et al. (2016) have used anticipated shocks to enforce the lower bound on interest rates.[1] This method can suffer from sign

[1] 1Anticipated shocks differ from the alternative approach commonly referred to as news shocks in many ways. Solving the model with anticipated shocks does not require a modification of the equations of the original system, in contrast to news shocks that are typically implemented by augmenting the law of motion of a shock process with additional shocks. An anticipated shock is genuinely a particular structural shock in the original system, while in the news shocks, it is a different iid shock with no other interpretation than a news shock and is unrelated to any structural shock in the system. Because it is unrelated, it will have its

reversals and the forward guidance puzzle.[2] [3]Just like the extended path and piecewise linear methods, agents are unaware of the constraint until it is actually binding. Moreover the Kuhn-Tucker and complementary slackness conditions associated with many occasionally binding constraints are not easily incorporated into this methodology.[4]

Many practitioners have used global and projection methods to account for occasionally binding constraints. Fernandez-Villaverde et al. (2015) and Judd et al. (2012), among others, have used global methods to impose the lower bound on interest rates. Christiano and Fisher(2000) have used global methods to enforce the non-negativity constraint on investment. Mendoza and Smith (2004) use value function iteration to solve a model with an occasionally binding debt constraint. These methods are certainty non-equivalent which means agents' behavior will be affected in the neighborhood of the constraint binding even when the constraint is not binding. The computational cost of these methods can be heavy putting a low upper bound on the size of models that can be solved.[5]

Regime-switching can also be used to impose occasionally binding constraints. Bianchi and Melosi (2014), Chen (2014), Binning and Maih (2016b) and Binning and Maih (2016a) all use regime-switching to impose the lower bound constraint on interest rates. Benigno et al.(2015) use regime-switching to model an occasionally binding debt constraint in a small open economy

own distribution independently of other parts of the system. Under the anticipated shocks approach the policy functions are explicitly expressed in terms of leads of future shocks as opposed to lags in the news shocks approach (Maih, 2015).

[2]Binning and Maih (2016a) describe sign reversals as:... a phenomenon that occurs when anticipated monetary policy shocks are used to enforce the lower bound constraint. In the face of large negative demand shocks, the [effective lower bound (ELB)] is generally seen as a contractionary monetary policy since the interest rate is not able to decrease any further in order to give a boost to the economy. In that case, the anticipated shocks required to keep the interest rate from going below its lower bound are also expected to be positive. The positive monetary policy shocks then act as contractionary policy shocks. In some cases, however, if the ELB is expected to last for a very long time, some of the shocks in the sequence of shocks required to keep the interest rate at the ELB may be negative. Negative monetary policy shocks are expansionary, which leads to an improvement of economic conditions at the ELB.

[3]The forward guidance puzzle describes a situation where anticipated monetary policy shocks have a larger than reasonable impact on the current period due to the lack of discounting on the shocks themselves in the consumption Euler equation.

[4]Estimation using these methods requires specialized filters (see Juillard and Maih, 2010).

[5]Estimation requires non-linear filters which further restrict model size.

model. Many occasionally binding constraints involve a policy aspect: the entry and exit of the zero lower bound are functions of the monetary policy regime, debt and collateral constraints can be affected by macroprudential policies like variable loan-to-value ratios. A desirable feature of any solution technique in such an environment is that it be robust to the Lucas critique: regime-switching methods exhibit such a feature.

在 User Guide for Advanced Topics 中较为详细地论述了目前 DSGE 中的四种类型 OBCs（Occasionally Binding Constraints）：

- 家庭/企业借贷约束（包括 CIA 等）；
- 不可逆投资
- ZLB
- 其他金融约束

4.1.1 Borrow Constraints

代表性李嘉图家庭和非李嘉图家庭模型中，借贷约束总是成立，例如 Iacoviello and Neri（2010）。Guerrieri and Iacoviello（2017）则放松这个假设，并研究不等式借贷约束如何驱动宏观经济不对称动态。

机制：

- 房地产繁荣时期，借贷约束宽松，因为家庭的抵押价值上升；
- 房屋价格崩塌，家庭抵押品价值迅速贬值，借贷约束收紧；
- 这进一步导致经济下行，并形成螺旋式下降趋势。

模型：

比例为 $0 < \lambda < 1$ 的李嘉图家庭的最优化问题为

$$\underbrace{max}_{c_t^p, h_t^p, b_t^p} \quad E_0 \sum_{t=0}^{\infty} \beta^{p,t} u(c_t^p, h_t^p) \tag{4.1}$$

$$c_t^p + q_t h_t^p + b_t^p = y_t^p + (1-\delta) q_t h_{t-1}^p + r_{t-1} b_{t-1} \tag{4.2}$$

其中，q_t 表示房屋价格，h^p 表示房屋持有量。FOCs：

$$u'(h_t^p) = q_t u'(c_t^p) - \beta^p(1-\delta)E_t[q_{t+1}u'(c_{t+1}^p)] \tag{4.3}$$

$$1 = \beta^p E_t\left[\frac{u'(c_{t+1}^p)}{u'(c_t^p)}\right] r_t \tag{4.4}$$

$1-\lambda$ 比例的非李嘉图家庭最优化问题与上述李嘉图家庭类似，除了多了一个借贷约束：

$$b_t^i \leq mq_t h_t^i \tag{4.5}$$

此时，非李嘉图家庭的 FOCs：

$$u'(h_t^i) = q_t u'(c_t^i) - \beta^i(1-\delta)E_t[q_{t+1}u'(c_{t+1}^i)] - \rho_t q_t m \tag{4.6}$$

$$\rho_t = u'(c_t^i) - \beta^i E_t\left[u'(c_{t+1}^i)\right] r_t \tag{4.7}$$

其中，ρ_t 表示借贷约束的拉格朗日乘数：

$$\rho_t \geq 0 \tag{4.8}$$

$$\rho_t(mq_t h_t^i - b_t^i) = 0 \tag{4.9}$$

4.1.2 Irreversible Investment

在现实中，至少有部分投资是不可逆的，例如如果经济条件发生显著变化使得投资变成沉没成本。

那么，此时，标准模型中就不存在投资的这种机会成本，而这种机会成本会引起不确定性效应。

机制：

1. 当企业投资，它观测到未来的状态分布：
    - 在好的经济情形下，企业会投资更多；
    - 在坏的情形下，企业会减少投资；
2. 定义随着未来状态分布方差的上升，不确定性也上升：

- 更高不确定性，更高的概率减少投资；

3. 在不可逆投资下，

   - cause rms to insure against this be waiting to invest；
   - Increase the threshold value of positive investment, i.e. rms will need to see higher expected returns from a project in order to invest.

企业生产函数：

$$Y_t = A_t f(K_{t-1}) \tag{4.10}$$

资本演化方程：

$$K_t = I_t + (1-\delta)K_{t-1} \tag{4.11}$$

企业要么投资，要么支付红利来实现股东的跨期贴现效用最大化

$$\underbrace{max}_{I_t, D_t} \ E_t \sum_{s=0}^{\infty} \beta^s u(D_{t+s}) \tag{4.12}$$

$$D_t + I_t = A_t f(K_{t-1}) \tag{4.13}$$

$$K_t = I_t + (1-\delta)K_{t-1} \tag{4.14}$$

$$I_t \geq 0 \tag{4.15}$$

那么，拉格朗日算式为

$$\underbrace{max}_{K_t} \ E_t \sum_{s=0}^{\infty} [\beta^s u(A_{t+s} f(K_{t+s-1}) - K_{t+s} + (1-\delta)K_{t+s-1}) + \lambda_{t+s}(K_{t+s} - (1-\delta)K_{t+s-1})] \tag{4.16}$$

其中，$\lambda_t \geq 0$ 表示正投资的拉格朗日乘数。那么，FOCs：

$$E_t\left[\beta u'(D_{t+1})(A_{t+1} f'(K_t) + 1 - \delta)\right] + \lambda_t - E_t[\lambda_{t+1}](1-\delta) = u'(D_t) \tag{4.17}$$

注意：对于任何时候，只要 $\lambda_t = 0$，下式恒成立：

$$E_t\left[\beta\frac{u'(D_{t+1})}{u'(D_t)}(A_{t+1}f'(K_t)+1-\delta)\right]=1 \tag{4.18}$$

上式为标准 RBC 模型中的资本欧拉方程。

拉格朗日乘数 λ_t 可以定义为放松正投资约束的影子价格。

- 如果 I_t 为正，则 λ_t 严格等于 0；
- 如果 $I_t = 0$，那么，$\lambda_t \geq 0$；
- 如果经济的状态使得投资在 0 处受到约束，那么，$E_t[\lambda_{t+1}] > 0$。

假设 $I_t > 0$，则 FOCs 变为

$$E_t\left[\beta u'(D_{t+1})(A_{t+1}f'(K_t)+1-\delta)\right]-E_t[\lambda_{t+1}](1-\delta)=u'(D_t) \tag{4.19}$$

如果标准假设 $(f' > 0, f'' < 0)$ 成立, 那么，**非零 $E_t[\lambda_{t+1}]$ 就意味着，相对于无约束投资，有约束投资的边际产出更高，这也意味着相对投资更低**。

相关研究：

Caballero and Pindyck (1996) 、Bloom, Bond and Van Reenen (2007) 、Gilchrist, Sim and Zakrajsek (2014) 、Bernanke (1983), Caballero ad Pindyck (1996), Bloom,Bond and Van Reenen (2007), Gilchrist, Sim and Zakrajsek (2014)。

4.1.3 ZLB

在标准 NK 模型中（DSGE 建模与编程入门（29）：NK、程序及结果），货币政策通常表示为应对目标通胀和目标产出缺口的偏离（DSGE 建模与编程入门 (31)：最优货币政策（一））。且名义利率以 0 为下界，因为货币具有储藏职能，因此，持有货币的理性人不可能接受负名义利率。这就是所谓的零下界（Zero Lower Bound，ZLB）。

但是，全球金融危机后，世界上许多央行都在挑战这一命题，陆续将名义利率调整到 0 附近，甚至设置为负利率。Walsh（2017）在其“Monetary Theory and Policy”中指出，如果持有货币有一定的成本，那么，人们或许愿意接受一定程度的负利率。因此，我们不要按照字面意思将其理解成名义利率钉住 0，最本质的理解应该是名义利率对经济条件变化的反应不敏感。Walsh（2017）将这种现象称为“有效下界（Effective Lower Bound，ELB）”。

ELB 问题的重要性至少体现在两个方面：第一，以 ELB 为代表的非线性效应；第二，政策含义，货币政策操作空间有限，财政政策如何作出反应。

也就是说，基准 NK 模型中的货币政策变为：

$$r_t = \begin{cases} (1-\rho_r)r + \rho_r r_{t-1} + (1-\rho_r)(\phi_\pi(\pi_t - \pi) + \phi_y(lnY_t - lnY)) + \epsilon_t & if > 0 \\ 0, \ otherwise \end{cases} \tag{4.20}$$

1. Multiple equilibria

   Whilst the uniqueness of equilibria can be guaranteed in the standard model by choice of parameters, this is not obviously true when interest rates are constrained at zero.

   - Many methods used to solve models with ZLB choose equilibrium path arbitrarily: When the constraint binds, a return path to long-run steady state is looked for with no check whether solution is unique (we'll see more specifically how in a bit)
   - Some papers recognise and discuss the many equilibrium paths
   - Other papers discuss multiple steady states/regimes.

   Consider standard 3-equation NK model:

   implies two long-run equilibria that depend on which agents expect:

   1. Normal/inflationary: $r = \hat{r}, \pi = \pi^{star}$;

   2. Deflationary: $r = 0, \pi = -r$.

   Non-fundamental (i.e. not to do with technology, output etc) shifts in expectations can cause economy to jump between equilibria (sunspots).

2. Indeterminacy

   Being in the deflationary equilibrium implies that agents believe that the economy is converging to the bad steady-state:

   - Many papers rule this out explicitly both on theoretical and empirical grounds.

It turns out under many specifications, the model is still indeterminate when ZLB binds, even in good equilibrium.

- Holden (2017) provides necessary and sufficient conditions for determinacy.
- There is a unique equilibrium under a price-level targeting regime rather than inflation targeting regime.

3. Fiscal multipliers

   A number of papers claim that the fiscal multiplier is large when the ZLB binds.

   Other papers claim that the fiscal multiplier is only about 1—i.e. no multiplier!

4.1.4 Financial Constraints

Another set of OBCs in the literature is on either firm financing or financial intermediation.

For further reading see:

- Adverse selection in investment: Swarbrick (2018).
- Bank borrowing constraints: Holden, Levine and Swarbrick (2016).
- Equity constraints: Brunnermeier and Sannikov (2014), He and Krishnamurthy(2013).
- Cash in advance constraints: Dixon and Pourpourides (2016).

4.2 Numerical Solutions

4.2.1 Computational Challenges

例如，要解 DSGE 模型，就意味着解下列形式的方程

$$1 = \beta E_t \left[\frac{u'(c_{t+1})}{u'(c_t)} R_{t+1} \right]$$

其中，$c_t = f(s_t, k_{t-1}, k_t)$，也就是说

- agents 的决策依赖于当前经济状态 st 和过去状态 k_{t-1}，且满足上述欧拉方程；
- 我们最终想解出一个政策函数 $k_t = g(s_t, k_{t-1})$，这就意味着 $c_t = f(s_t, k_{t-1}, g(s_t, k_{t-1}))$；
- 找到政策函数 $g()$ 的解析解几乎是不可能的，因此，必须利用函数近似来得到它的近似形式。

那可以采用什么方法来近似 $g()$ 呢？

方法一：采用全局方法来近似政策函数 g，即在一个非常大的状态空间（s_t k_{t-1} 的所有可能值）来得到一个非线性近似函数 $g'() = g()$：

- 这种方法很容易刻画 OBCs 特征；
- 但是，计算耗时非常长，从而只限于较小的模型（状态空间较小）。

方法二：扰动法（局部近似法），即在一个不动点（通常为确定性稳态）附近来近似模型：

- 快速、较为精确；
- 在不动点出解模型时，如果约束成立，那么就一直成立，如果约束不成立，那么就一直不成立。（注：在 OccBin 方法中表现得尤为明显）

全局算法方面有投影法和惩罚函数法，这些方法均需要自己编写程序。

The main drawback of the global solution methods is that

- models solved using global techniques do not scale well to larger models;
- the majority of model solved using this method make restrictive assumptions to limit the state space, such as using exogenous endowment instead of an endogenous production sector;
- Global methods also cannot guarantee convergence for non-optimal problems such as the Taylor rule with a ZLB;
- require a high level of technical skill to set-up.

下面，我们只讲解 Dynare 可用的局部算法。

4.2.2 OccBin

下面介绍一下 Iacoviello 的 OccBin 算法，及其操作。据我了解，OccBin 方法是目前国际上最流行的 OBCs 解法之一。

一、基本思想

OccBin 算法的学名称为“分段线性”算法，顾名思义，它将 OBC 处的非线性性质划分成两个分段函数，每个分段函数称为一种“机制（regime）”（注：没错 OBCs 与 Markov switch-regime 关系非常紧密，我在将来也会给大家呈现二者之间的详细关系），然后，分别 run 两个 regime 下的模型。具体的算法为：

- 在“机制 regimes”附近，线性化模型：在一种机制下，约束成立；在另一种机制下，约束不成立；
- 其中一种机制为基准机制（reference regime），且基准机制下一定要满足 BK 条件。注：哪种机制为基准机制，这要看研究目的，例如，在 OBCs in macroeconomics 中呈现的 ZLB 情形中，基准机制为 ZLB 约束不成立的情形；而在金融约束情形下，基准机制为金融约束成立；
- 在金融约束 DSGE 中，从机制 1 开始，随着金融约束逐渐放宽，直至约束不成立，则转向机制 2；
- 上述算法可以良好运行，是假设代理人认为（a）未来没有冲击；（b）经济会在有限期内回到初始机制。

例子一：在 ZLB-DSGE 模型中，ZLB 为：

$$R_t = max(Z_t, 1)$$

而泰勒规则为：

$$Z_t = R_{t-1}^{\rho_R}\left[R\left(\frac{\Pi_t}{\Pi}\right)^{\phi_\pi}\left(\frac{Y_t}{Y}\right)^{\phi_y}\right]^{1-\rho_R}$$

其中，不带时间下标的字幕均表示稳态值。

也就是说，我们可以将上述模型划分为两种机制：

机制 1:

$$Z_t = R_{t-1}^{\rho_R}\left[R\left(\frac{\Pi_t}{\Pi}\right)^{\phi_\pi}\left(\frac{Y_t}{Y}\right)^{\phi_y}\right]^{1-\rho_R}$$

机制 2:

$$R_t = 1$$

例子二：

在借贷约束模型中，家庭的借贷决策为：

$$\rho_t = u'(c_t^i) - \beta^i E_t \left[u'(c_{t+1}^i)\right] r_t$$

$$\rho_t \left(m q_t h_t^i - b_t^i\right) = 0$$

ρ_t 为借贷约束的拉格朗日乘数。借贷约束成立时，还需要满足 Kuhn-Tucker 条件 $\rho_t \geq 0$，且 $b_t^i \leq m q_t h_t^i$。

那么，我们就可以将上述借贷约束划分为两个机制：

机制 1:

$$b_t^i = m q_t h_t^i$$

机制 2:

$$\rho_t = 0$$

或者

$$1 = \beta^i E_t \left[\frac{u'(c_{t+1}^i)}{u'(c_t^i)}\right] r_t$$

同理，我们也可以这样来划分其它金融约束。

那么，OccBin 算法为：模型从机制 1 开始运行，

1. 假设暂时性冲击使得 ρ_t 下降到 0 以下；

2. 猜测对于 ρ_t 来说，经过 T 期后，ρ_t 会返回为正值；

3. 对于 t>=T，就使用机制 1 的决策来解出 ρ_t，c 等等内生变量；

4. 而对于 t<T，则利用机制 2 的决策来向后解出 c，y 等内生变量，且在无冲击的情形下；

5. 升级 T，重复上述步骤。

相关论文可参见 Guerrieri and Iacoviello (2015)，或者【Seminar 预告】第十六期 CIMERS Seminar：OBC 所导致的宏观经济波动的不对称性

二、模型

我们以 ZLB 为例，来看看 OccBin 的操作。

首先，完整的模型为

（1）欧拉方程

$$\frac{1}{C_t} = \beta u_t E_t \left(\frac{R_t}{C_{t+1}\Pi_{t+1}} \right)$$

C 表示消费，u 表示贴现率冲击，R 表示名义利率，Π 表示通胀率；

（2）利润最大化问题 FOC1：

$$x_{2t} = \left(\frac{\epsilon}{\epsilon - 1} \right) \psi v_t^{\vartheta} \left(\frac{Y_t}{A_t} \right)^{1+\vartheta} + \theta\beta u_t E_t \Pi_{t+1}^{\epsilon} x_{2t+1}$$

（3）利润最大化问题 FOC2：

$$\frac{x_{2t}}{\Pi_t^{\star}} = \left(\frac{Y_t}{C_t} + \theta\beta u_t E_t \frac{\Pi_{t+1}^{\epsilon-1}}{\Pi_{t+1}^{\star}} x_{2t+1} \right)$$

（4-1）ZLB

$$R_t = max(Z_t, 1)$$

（4-2）泰勒规则

$$Z_t = R_{t-1}^{\rho_R} \left[R \left(\frac{\Pi_t}{\Pi} \right)^{\phi_\pi} \left(\frac{Y_t}{Y} \right)^{\phi_y} \right]^{1-\rho_R}$$

其中，稳态 $R = \frac{\Pi}{\beta}$

（5）通胀运动法则

$$1 = \theta \Pi_t^{\epsilon-1} + (1-\theta)(\Pi_t^{\star})^{1-\epsilon}$$

（6）价格扩散指数

$$v_t = \theta \Pi_t^{\epsilon} v_{t-1} + (1-\theta)(\Pi_t^{\star})^{-\epsilon}$$

（7）市场出清

$$Y_t = C_t + gY_t$$

（8）贴现率冲击

$$lnu_t = \rho_u lnu_{t-1} + \epsilon_t^u$$

（9）生产率冲击

$$lnA_t = \rho_A lnA_{t-1} + \epsilon_t^A$$

参数校准值

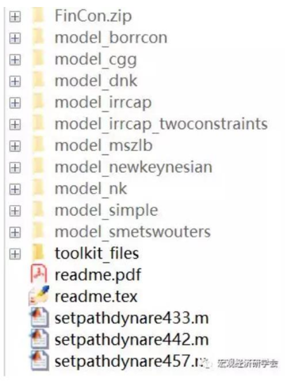
```
\beta=0.994;
\phi=1;
\epsilon=6;
\theta=0.90;
\phi_p=2.5;
\phi_y=0.25;
\rho_r=0;
g=0.20;
\Pi_{ss}=1.005;
\rho_u=0.8;
\rho_a=0;
\psi=1;
```

三、OccBin 操作

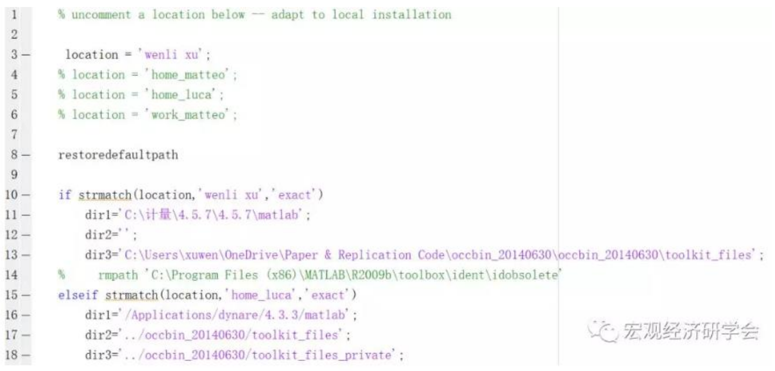
OccBin 是基于 Dynare 和 Matlab 的一个程序集，下载地址为：

https://www2.bc.edu/matteo-iacoviello/research_files/occbin_20140630.zip

下载之后，解压，我们会看到很多文件及文件夹：


```
FinCon.zip
model_borrcon
model_cgg
model_dnk
model_irrcap
model_irrcap_twoconstraints
model_mszlb
model_newkeynesian
model_nk
model_simple
model_smetswouters
toolkit_files
readme.pdf
readme.tex
setpathdynare433.m
setpathdynare442.m
setpathdynare457.m
```

- setpathdynareXXX.m 文件是设置 dynare 和 OccBin 路径的文件，一般来说，我们要将 Dynare-matlab 和 OccBin-toolkit_files 添加到 matlab 的默认路径中。该文件打开后，类似如下：



```
1      % uncomment a location below -- adapt to local installation
2
3 -     location = 'wenli xu';
4      % location = 'home_matteo';
5      % location = 'home_luca';
6      % location = 'work_matteo';
7
8 -    restoredefaultpath
9
10 -   if strmatch(location,'wenli xu','exact')
11 -       dir1='C:\计量\4.5.7\4.5.7\matlab';
12 -       dir2='';
13 -       dir3='C:\Users\xuwen\OneDrive\Paper & Replication Code\occbin_20140630\occbin_20140630\toolkit_files';
14     %     rmpath 'C:\Program Files (x86)\MATLAB\R2009b\toolbox\ident\idobsolete'
15 -   elseif strmatch(location,'home_luca','exact')
16 -       dir1='/Applications/dynare/4.3.3/matlab';
17 -       dir2='../occbin_20140630/toolkit_files';
18 -       dir3='../occbin_20140630/toolkit_files_private';
```

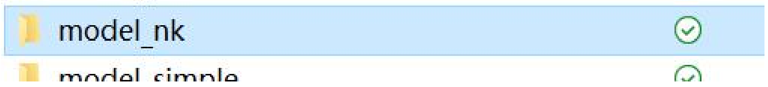
1. 我们需要更改电脑本地用户名，例如，我的电脑本地用户名为“wenli xu”，因此，我将第 3 行单引号改为 **location=** ‘wenli xu’；
2. 将第 10 行的字符匹配，也相应的更改为‘wenli xu’；
3. 将第 11 行路径 1 设置为我们安装的 Dynare 中的 matlab 子文件夹所在的位置，例如 dir1=’C:/计量/4.5.7/4.5.7/matlab’;
4. 将第 13 行的路径 3 设置为我们安装的 OccBin 中的 toolkit_files 子文件夹所在的位置，例如 dir3=’C:/Users/xuwen/OneDrive/Paper Replication Code/occbin_20140630/occbin_20140630/toolkit_files’;
5. 保存该文件，运行该文件，要么点击 run 按钮，要么在 matlab 的命令窗口输入该文件名：»setpathdynare457，回车。

但是，我强烈建议不使用这种方法来添加路径，因为这样每次重新启动 matlab 后，都要重新运行 setpathdynare457。

推荐：直接点击 matlab 菜单栏上的 setpath—Add with Subfolders—toolkit_files—save.

除了 setpathdynare 文件和 toolkit_files 文件夹之外，还有其他的文件夹，那些都是我们自定义的模型文件，里面包括 mod 文件、稳态文件和模拟文件。我们以上述 ZLB 模型为例：

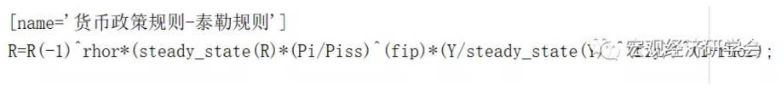
- 我们新建一个文件夹 model_nk



OccBin 的两个关键步骤：

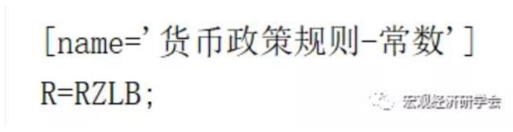
第一步，写两个.mod 文件，一个为无 ZLB 约束的模型（例如 nk.mod），一个为有 ZLB 约束的模型（例如 nk_zlb.mod）。

– 在无约束模型中，就如我们平时写 mod 文件一样。需要注意的是，此时 ZLB 并没有约束，因此，货币政策规则还是写泰勒规则：



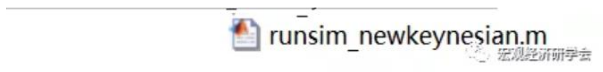
```
[name='货币政策规则-泰勒规则']
R=R(-1)^rhor*(steady_state(R)*(Pi/Piss)^(fip)*(Y/steady_state(Y[illegible];
```

– 在有约束模型 mod 文件中，只需要保留上述基准 mod 中的 var、varexo、parameters、model 模块。且货币政策规则现在是一个常数：



```
[name='货币政策规则-常数']
R=RZLB;
```

第二步，写一个单独的.m 文件，在这个文件中，我们需要声明约束（以线性化的形式），然后调用 OccBin 的 toolkit 函数来解、模拟约束和无约束模型。例如，我们写 runsim_newkeynesian.m 文件。



runsim_newkeynesian.m

我们可以看看这个文件的程序：

（1）声明全局变量：M_、oo_，如果对 Dynare 输出比较熟悉的话，M_ 和 oo_ 两个全局变量分别指代模型的内生/外生/参数变量名，以及模拟结果变量（其中，包含稳态、模拟值等等）

```
\textcolor{red}{M\_~oo\_}
```

声明全局变量，可以让备择模型的 mod 调用基准模型的参数，以及模拟结果等。大家回忆一下，在备择模型 mod 文件中，只包含了 var、varexo、parameters 以及 model 模块。因此，需要用这些全局变量来反馈到备择模型中。

（2）声明基准模型和备择模型的 mod 文件名：

```
modnam = 'nk';
modnamstar = 'nk\_zlb';
```

其中，modnam 表示基准模型的 mod 文件名，modnamstar 表示备择模型的 mod 文件名。由于在我们的 ZLB-DSGE 案例中，基准模型是无 ZLB 约束，备择模型是有 ZLB 约束。

（3）声明约束条件：

```
constraint = \textcolor{red}{'R<RZLB-Piss/betta'};
constraint_relax =\textcolor{red}{'R>RZLB-Piss/betta'};
```

其中，constraint 表示约束成立的条件；constraint_relax 表示约束不成立的条件。

在上例中，

– 原始 ZLB 约束为：

$$R_t < R^{ZLB} = 1$$

– 需要特别注意的是，在上述约束条件声明中，变量必须要重写成它们出现在 Dynare 解中的线性形式。一般来说，这依赖于我们在 mod 文件 model 模块如何声明模型形式（例如，levels，log-levels，等等）。在我们上述例子中，我们是以非线性的形式声明模型，因此，我们需要将约束条件转换为线性的形式，即 $\hat{X}_t = X_t - X$（$\hat{X}_t$ 表示线性化变量，X 表示稳态值）；

– 因此，我们需要将 ZLB 约束表达成

$$R_t = \hat{R}_t + R < R^{ZLB}$$

$$\hat{R}_t < R^{ZLB} - R$$

– 因此，我们可以看出，在声明约束条件时：

```
constraint = \textcolor{red}{'R<RZLB-Piss/betta'};
constraint_relax =\textcolor{red}{'R>RZLB-Piss/betta'};
```

（4）当程序判断 constraint 为 true 时，模型的解就从基准模型转换到备择模型，也就是 run nk_zlb；当 constraint_relax 为真时，解就 run nk。

（5）将.mod 文件中外生冲击变量赋予 irfshock：

```
irfshock =char('e_a','e_u');
```

由于上例中的外生冲击有两个 e_a 和 e_u，所以上面声明两个冲击。

（6）在 shockssequence 声明不可预期冲击序列，且在 nperiods 声明 IRF 期数：

```
SHOCKS = [ zeros(5,2)
    0   0.024
   zeros(20,2) ] ;



shockssequence = SHOCKS;
nperiods = 30;
```

（7）通过以下命令来解模型，并得到 IRFs：

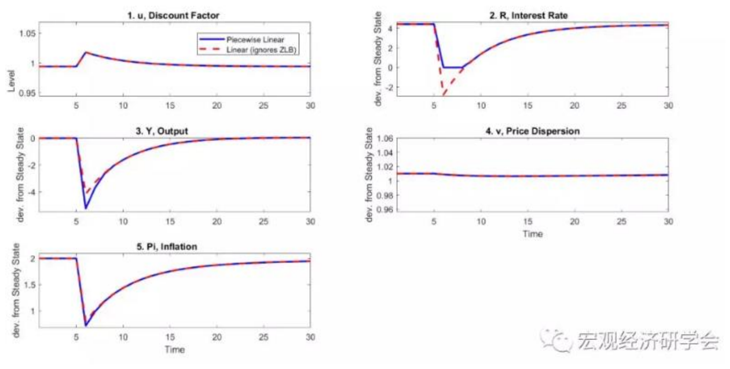
```
[zdatalinear zdatapiecewise zdatass oobase_ Mbase_
] = ...
   solve_one_constraint(modnam,modnamstar,...
   constraint, constraint_relax,...
   shockssequence,irfshock,nperiods);
```

其中，省略号表示换行。

- zdatalinear：忽略 OBCs 时，模拟内生变量的动态路径，以稳态偏离表示。每一列代表一个变量，顺序按照 mod 文件中声明的变量顺序；
- zdatapiecewise：OBCs 成立时，模拟内生变量的动态路径。顺序同上；
- zdatass：内生变量的稳态值；

– oobase_ 和 Mbase_ ：运行基准模型时，Dynare 产生的结构，参见手册。

（8）在 matlab 命令窗口输入：»runsim_newkeynesian，可以得到以下脉冲响应图



我们上上图 2 中可以明显看到，ZLB 约束成立时，名义利率的变化（蓝色线条）。

四、注意事项

上面，我们已经介绍了 OccBin 的基本原理和操作。但是仍然有一些事项需要特别注意。

1. 在声明约束条件时，OccBin 只允许使用线性化形式的约束条件；

2. constraint 和 constraint_relax 中只允许出现同期内生变量，例如，抵押约束 $b_t \leq \rho_t q_t h_t$，如果是 $b_t \leq \rho_t q_t h_{t+1}$，那么，这个条件不能直接输入 constraint 和 constraint_relax 中，我们应该先将 h_{t+1}，使用 lead_ 转换一下。如果是滞后期也是同理。领先滞后期转换请参见 dynare 手册。

3. 不需要声明备择模型的稳态。基准模型和备择模型都是在基准模型的稳态附近进行近似。基准模型需要满足 BK 条件，而备择模型则不需要满足 BK 条件。

4. 参数仅仅只需要在基准模型中赋值，而在备择模型中则不需要对参数赋值。如果参数只在备择模型中出现（例如上例中的 R^{ZLB}），我们也需要将该参数在基准模型中声明，并赋值。

此外，OccBin 与机制转移模型有非常重要的联系：

- 存在机制转换时，OccBin 并不 deliver 政策规则，它 100% 与模型一致。因为 OccBin 是假设代理人不可预期约束 binding，同时也不会形成预期什么时候进入或退出 OBCs。

- 但是，在某些宏观经济研究中，代理人关于机制转移的预期也非常有趣，且对宏观经济有非常重要的影响。因此，很多学者也对量化这种预期很感兴趣，这就是 MS-DSGE 模型。

Andrew Binning and Junior Maih（2017）就将 OBCs 当做一种特殊的 RS，并用 Regime-Switching 来建模 OBCs。后续，我们会介绍这种方法。

五、小结

When the expression $stoch\_simul()$ is used in the .mod file, Dynare computes a Taylor approximation of the decision and transition functions of the model. This is computed around the steady state and at first order will be of the form

$$y_t = y + A(y_{t-1} - y) + Bu_t$$

where y is the steady state value of y_t. It is worth pointing out here that if we wanted to include an inequality constraint in a model such as

$$y_t = \begin{cases} f(x_t), & when f(x_t) \geq 0 \\ 0, & otherwise \end{cases}$$

then although possible to write the function $y = max(fx, 0)$, if $stoch\_simul()$ is used, Dynare will perform a local approximation around the deterministic steady state. The equation will then either be a approximation of $y_t = f(x_t)$ or $y_t = 0$ depending on whether the constraint binds in steady state. The inequality constraint will be lost.

The approach generalises to higher orders and up to order three is available in Dynare. For further reading on the technical detail for the derivation of policy function and the higher order equivalents, we refer the reader to Collard and Juillard (2001).

Whilst the perturbation techniques employed by Dynare to solve stochastic models lose this type of constraint, the same is not true for perfect foresight or extended path solutions.

4.2.3 Extended_Path Approach

As write in the manual:

The use of the following functions and operators is strongly discouraged in a stochastic context: max, min, abs, sign, <, >, <=, >=, ==, !=. The reason is that the local approximation used by stoch_simul or estimation will by nature ignore the non-linearities introduced by these functions if the steady state is away from the kink. And, if the steady state is exactly at the kink, then the approximation will be bogus because the derivative of these functions at the kink is bogus (as explained in the respective documentations of these functions and operators).

Note that **extended_path** is not affected by this problem, because it does not rely on a local approximation of the model.

It is possible to compute a simulation to a stochastic model using the extended path method presented by Fair and Taylor (1983). This method is especially useful when there are strong nonlinearities or binding constraints. Such a solution is computed using the *extended_path* command.

一、基本思想

The extended path approach, introduced by Fair and Taylor (1983), relies on the perfect foresight model solvers to take full account of the deterministic nonlinearities (either related to the preferences or technology functional forms, or to the presence of occasionaly binding constraints).

- In each period of the sample, exogenous innovations are treated as surprise shocks in the first period of a deterministic simulation where shocks are set to their expected value in all future periods.

The advantages of this approach,compared to other global approximation methods, is that

- it allows to simulate very large models (like a multicountry model with thousands of equations).

- this approach is not model dependent, as other global approximation methods, it does not require more work than writting the equations of the model in a Dynare’s .mod file.

二、算法

Perfect Foresight Solutions

When *simul*() is used in the .mod file, Dynare uses a Newton-type algorithm which preserves these non-linearities. Details of the algorithm can be found at Juillard (1996). In such a model, the economy is in equilibrium up until period 1 at which point the agents learn about any shocks in the current or future periods. The simulation then describes the response of the agents before and following any shocks, until the economy returns to equilibrium. This equilibrium may or may not be the same as the initial one before period 1 and the path to equilibrium is imposed in finite time rather than asymptotically.

As using a deterministic set-up allows the analysis of the full implications of non-linearities, it is a useful starting point.

Suppose there were no past shocks and will be no future shocks so no uncertainty; we can write the borrowing constraints model from above:

$$\begin{pmatrix} ... \\ min\{\rho_t, (mq_t h_t^i - b_t^i)\} \end{pmatrix} = 0$$

or

$$f(x_{t-1}, x_t, x_{t+1}, u_t) = 0$$

where x_t is the endogenous state and u_t are exogenous shocks.

A perfect foresight solution given initial u_1, starting from x_0 is a path $x_0; x_1; x_2;$ ... converging to steady state and satisfying:

$$f(x_0, x_1, x_2, u_1) = 0$$

$$f(x_1, x_2, x_3, 0) = 0$$

$$f(x_2, x_3, x_4, 0) = 0$$

...

Dynare's perfect foresight solver can be used for calculating perfect foresight IRFs.

To do this, in the shocks block we specify the initial impulse, e.g.:

```
shocks;
    var epsilon; periods 1; values -10;
end;
```

where the number after values specifies the size of the initial impulse.

And then we perform the simulation with:

```
simul( periods=400 );
```

To solve, this is stacked over T periods

$$F(X) = \left\{ \begin{array}{c} f(x_0, x_1, x_2, u_1) \\ f(x_1, x_2, x_3, 0) \\ \vdots \\ f(x_{T-1}, x_T, x_{T+1}, 0) \end{array} \right\} = 0$$

Initial and end conditions given ($x_0 = x_{T+1} = \hat{x}$)

From an initial guess at n = 0, solve the following, updating x until suffcient accuracy is achieved.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Dynare uses steady state for initial guesses.

Dynare stores the output of perfect foresight simulations in: *oo_.endo_simul*

(Stochastic) Extended-Path Algorithm

Dynare (i) draws a random path for all exogenous variables then (ii) computes the corresponding path for the endogenous variables using the algorithm:

1. An integer k is chosen which is the number of periods for which expectations must be computed.

2. Beginning with expectations equal to the steady-state values $E_s x_{s+r} = x$ for r = 0,..., k, obtain a new set of expectations by solving the non-linear model dynamically. Every period the expectations are evaluated using a Gaussian quadrature.

3. Setting the new expectations as the starting values, this is repeated until convergence.

4. Repeat the above but increasing r by 1 and repeat this until convergence.

The solution method builds on the underlying perfect foresight algorithm of the last section, the difference being that $E_t\left[\{x_{t+s}\}_{s=1}^{k}\right]$ will not necessarily equal $\{x_{t+s}\}_{s=1}^{k}$.

This captures the non-linearities of the system and differs from the standard stochastic simulation which computes approximations to the transition and policy equations.

三、Dynare 命令

we use the New Keynesian model with a zero lower bound on nominal interest rates from Fern · ndez-Villaverde et al. (2015) as given in the provided code. The equilibrium conditions are outlined

$$\frac{1}{C_t} = E_t \frac{\beta}{C_{t+1}} \frac{R_t}{\Pi_{t+1}}$$

$$W_t = \psi L_t^{\vartheta} C_t$$

$$MC_t = \frac{W_t}{A_t}$$

$$\epsilon X_{1,t} = (\epsilon - 1) X_{2,t}$$

$$X_{1,t} = \frac{1}{C_t} MC_t Y_t + \theta E_t \beta \Pi_{t+1}^{\epsilon} X_{1,t+1}$$

$$X_{2,t} = \Pi_t^{\star} \left(\frac{Y_t}{C_t} + \theta E_t \beta \frac{\Pi_{t+1}^{\epsilon-1}}{\Pi_{t+1}^{\star}} X_{2,t+1} \right)$$

$$R_t = max(1, Z_t)$$

$$Z_t = R^{1-\rho_r} R_{t-1}^{\rho_r} \left[\left(\frac{\Pi_t}{\Pi}\right)^{\phi_\pi} \left(\frac{Y_t}{Y}\right)^{\phi_y} \right]^{1-\rho_r} M_t$$

$$G_t = S_{g,t} Y_t$$

$$1 = \theta \Pi_t^{\epsilon-1} + (1-\theta) \Pi_t^{\star\ 1-\epsilon}$$

$$v_t = \theta \Pi_t^{\epsilon} v_{t-1} + (1-\theta) \Pi_t^{\star\ -\epsilon}$$

$$Y_t = C_t + G_t$$

$$Y_t v_t = A_t L_t$$

In Dynare this is easy to implement. Z_t can be specified as a model-local variable1 by leaving it out of the variable declarations at the beginning of the mod file and entering '#' before the equation in the 'model' section. The Taylor-type rule can then be expressed in the .mod file

```
model;
.
.
.
#z=rho_r*r(-1) + (1 - rho_r)*( r_bar + phi_pi*...
( pi - pi_bar ) + phi_y*( y - y_bar ) + m;
r=max(0,z);
.
.
.
end;
```

确定性模型

- *simul*(*lmmcp*);

  Solves the perfect foresight model with a Levenberg-Marquardt mixed complementarity problem (LMMCP) solver (Kanzow and Petra 2004), which allows to consider inequality constraints on the endogenous variables (such as a ZLB on the nominal interest rate or a model with irreversible investment). This option is equivalent to stack_solve_algo=7 and solve_algo=10. Using the LMMCP solver requires a particular model setup as the goal is to get rid of any min/max operators and complementary slackness conditions that might introduce a singularity into the Jacobian. This is done by attaching an equation tag (see Section 4.5 [Model declaration], page 20) with the mcp keyword to affected equations. This tag states that the equation to which the tag is attached has to hold unless the expression within the tag is binding. For instance, a ZLB on the nominal interest rate would be specified as follows in the model block:

```
model;
...
[mcp='r>-1.94478']
r = rho*r(-1) + (1-rho)*(gpi*Infl+gy*YGap) + e;
...
end;
```

  where 1.94478 is the steady state level of the nominal interest rate and r is the nominal interest rate in deviation from the steady state. This construct implies that the Taylor rule is operative, unless the implied interest rate r<=-1.94478, in which case the r is fixed at -1.94478 (thereby being equivalent to a complementary slackness condition). By restricting the value of r coming out of this equation, the mcp-tag also avoids using max(r,-1.94478) for other occurrences of r in the rest of the model. It is important to keep in mind that, because the mcp-tag effectively replaces a complementary slackness condition, it cannot be simply attached to any equation. Rather, it must be attached to the correct affected equation as otherwise the solver will solve a different problem than originally intended.

Note that in the current implementation, the content of the mcp equation tag is not parsed by the preprocessor. The inequalities must therefore be as simple as possible: an endogenous variable, followed by a relational operator, followed by a number (not a variable, parameter or expression).

例子

We focus first on the government spending shock. The model is in deterministic steady state until period one at which point the government cuts spending G_t as a proportion of GDP by nearly half, sustaining the cuts for five periods before returning spending level to the original steady state value. The shock specification is detailed

```

shocks;
var epsilon_Sg;
periods 1:5;
values 100;
end;
```

Note that we have also altered the persistence ρ_g of the shock to zero, and set the variance S_g equal to 1. This is equivalent to ensuring the government spending is determined by the equation

$$S_{g,t} = S_g + \epsilon_{g,t}$$

where $S_{g,t}$ is the size of government spending relative to output and Sg = 20% is the steady state value. During this time, the shadow interest rate becomes negative and so the nominal interest rate is constrained at zero. This can be seen in figure 4.1.

Disadvantages:

- Captures full non-linearities including OBCs;

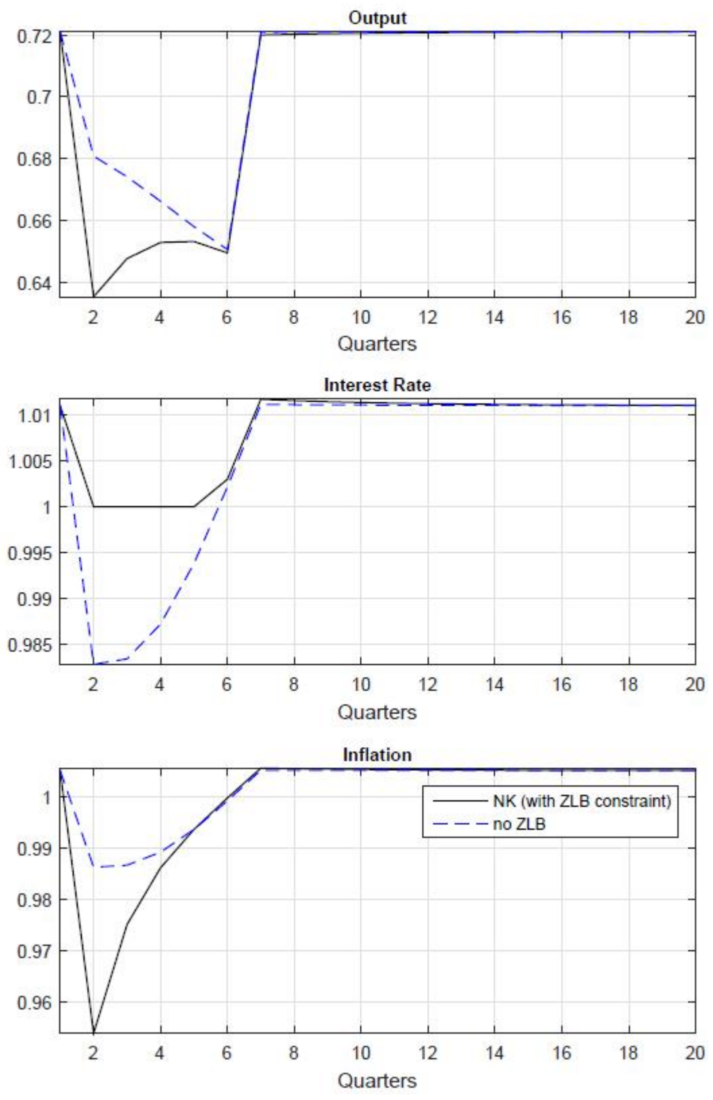


Figure 4.1: Perfect foresight impulse response function following negative demand shock

- However, heavy computational demand so can be slow and unstable. Particularly if equations not differentiable everywhere-as with OBCs

  The model is solved by stacking the system over T periods, where T is large enough to return the system to equilibrium. The Newton approach solves all the equations over every period simultaneously. This approach leads to a very large Jacobian; for a simulation over T periods, a model with n endogenous variables will require a Jacobian matrix of order n * T leading to high computational demands.

- the role of uncertainty in the behaviour of agents is lost. Fails to capture risk and precautionary behaviour.

- Also, The choice of solution can be a problem; it is unclear if there are multiple solutions to the model and why the one returned is chosen. It may not converge to a solution and(i)if it doesn't, we don't know it is because there isn't one,(ii)if it does, we don't know if it is unique.

随机模型

Shocks are specified as when using stoch_simul, but instead use:

- *extended_path*;

- *extended_path*(*OPTIONS*...);

  Description

  extended_path solves a stochastic (i.e. rational expectations) model, using the extended path method presented by Fair and Taylor (1983). Time series for the endogenous variables are generated by assuming that the agents believe that there will no more shocks in the following periods.

  This function first computes a random path for the exogenous variables (stored in oo_.exo_simul, page 46) and then computes the corresponding path for endogenous variables, taking the steady state as starting point. The result of the simulation is stored in oo_.endo_simul, page 46). Note that this simulation approach does not solve for the policy and transition equations but for paths for the endogenous variables.

  OPTIONS

- $periods = INTEGER$

  The number of periods for which the simulation is to be computed. No default value, mandatory option.

- $solver\_periods = INTEGER$

  The number of periods used to compute the solution of the perfect foresight at every iteration of the algorithm. Default: 200.

- $order = INTEGER$

  If order is greater than 0 Dynare uses a gaussian quadrature to take into account the effects of future uncertainty. If order=S then the time series for the endogenous variables are generated by assuming that the agents believe that there will no more shocks after period t+S. This is an experimental feature and can be quite slow. Default: 0.

- *hybrid*

  Use the constant of the second order perturbation reduced form to correct the paths generated by the (stochastic) extended path algorithm.

When the command $extended\_path(order = k, periods = T)$ is used, Dynare first draws a random path for the exogenous variables over T periods before computing the corresponding path for endogenous variables, taking the steady state as starting point. The order k specifies the number of periods ahead for which agents believe shocks may occur.

At order 0, this creates a stochastic simulation as if only the shocks of the current period were random, meaning that this is just a stochastic version of the standard perfect foresight solution method. As this neglects Jensen’s inequality, uncertainty plays no role in the model.

In practice, the extended-path method in dynare doesn't solve the full rational expectations model. The modeller specifies the maximum value of k for which to solve using the order option (k-level thinking).

By increasing the order, the number of periods of uncertainty considered are increased. For example at order one, the algorithm allows for shocks next period, but assumes no shocks will arrive in two periods’ time, at order 2, two periods of future shocks are allowed for, and so on. The approximation is solved by integrating over the shocks up to the specified horizon. Beyond this time horizon, the assumption is that all shocks will for evermore be zero and so consequently the model will ignore the long-run risk of hitting the bound.

For example, if order =2 were entered, dynare would only compute expectations as if shocks could be non-zero for 2 more periods. After this horizon, the agents would believe that there would be no more future shocks.

例子

To solve the NK model, we set the order k = 0 and periods T = 1000. Figure 4.2 shows the paths of the interest rate, output and inflation over 100 periods

At order zero there is neither future uncertainty nor precautionary behaviour. Up to the period the constraint binds and immediately after, the paths of the variables are identical as the agents ignore any risk of the constraint binding in future. To deal with this, it would be necessary to increase the order and allow the agents to evaluate future uncertainty. We find that by doing so, the model becomes prohibitively slow to solve.

- For accuracy, the order should be set to be the same magnitude as the decay of the model.

- dynare uses Gaussian quadrature to evaluate the expectations which scales exponentially in the number of shocks and the order (although not the number of states).

- In practice, solving with sufficient accuracy is infeasible except for very small models.

4.2.4 News Shocks Approach

As described in the above, there are a number of methods for solving larger models with OBCs including fast ones (such as Guerrieri and Iacoviello (2015b)) and accurate ones (such as Maliar and Maliar(2015)).

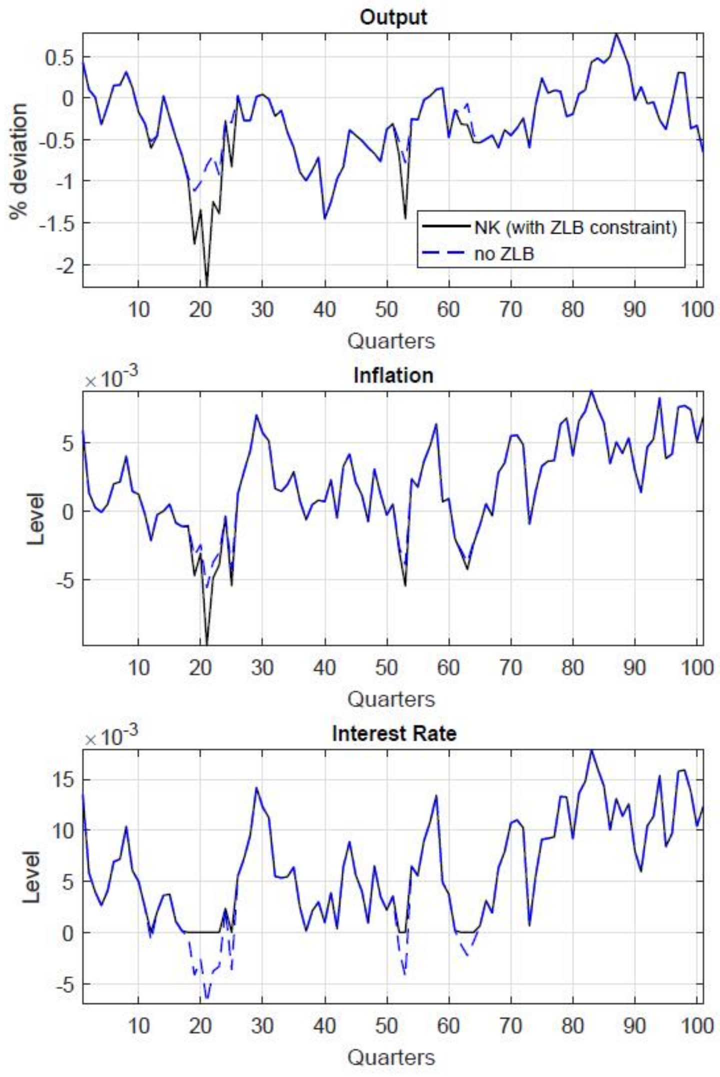


Figure 4.2: Extended-Path simulations (order = 0)

In this section, we will introduce the News shocks method[6] proposed in Holden (2016a,b) which provides a balance between accuracy and speed in simulation. For detail of the method, we direct the reader to a theoretical paper on the existence and uniqueness condition in Holden (2016b), and a paper detailing the computational algorithm in Holden (2016a).

Idea: use news shocks to impose the bounds, and an endogenous source of information about the constraint which can be predicted.

- If an OBC would otherwise be violated, anticipated news shocks push the constrained variable(s) back to their bound.
- A shock that drives a variable to its bound for an episode can be thought of as a combination of the original shock and a sequence of news shock stating that the variable will be higher than expected for some duration.

As showed in Figure 4.3

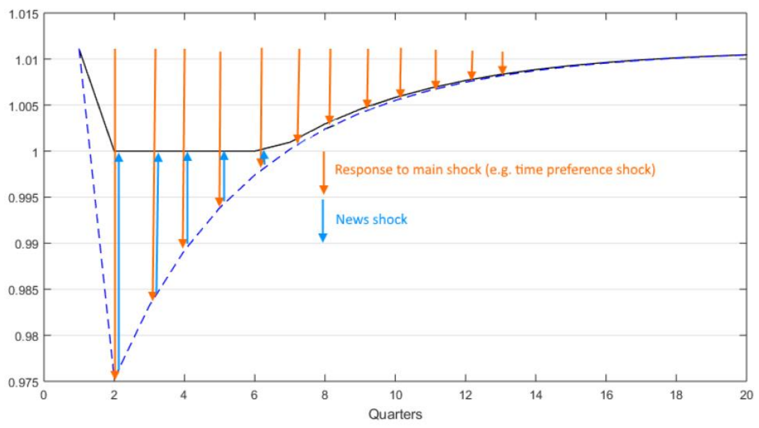


Figure 4.3: Holden(2016) news shocks method

The algorithms are designed to be easily implemented into Dynare. The occasionally binding constraint tool-kit DynareOBC needs to be downloaded and added to the Matlab path. The tool-kit is available from

https://github.com/tholden/dynareOBC

[6]参见 DSGE 建模与编程入门 (44):ZLB 与 News Shocks。News Shocks 更加详细的信息可以参考 Barsky and Sims（2011，JME）、Born et al. （2013，JEDC）和 Barsky et al.（2015，Whither News Shocks?）等。

DynareOBC works best if you have admin rights on your own machine, so it may install dependencies automatically. If you dont have admin rights, the ReadMe file contains details of what your IT administrator must install for you.

Once downloaded, the root DynareOBC folder, containing dynareOBC.m, needs to be added to the your Matlab path.

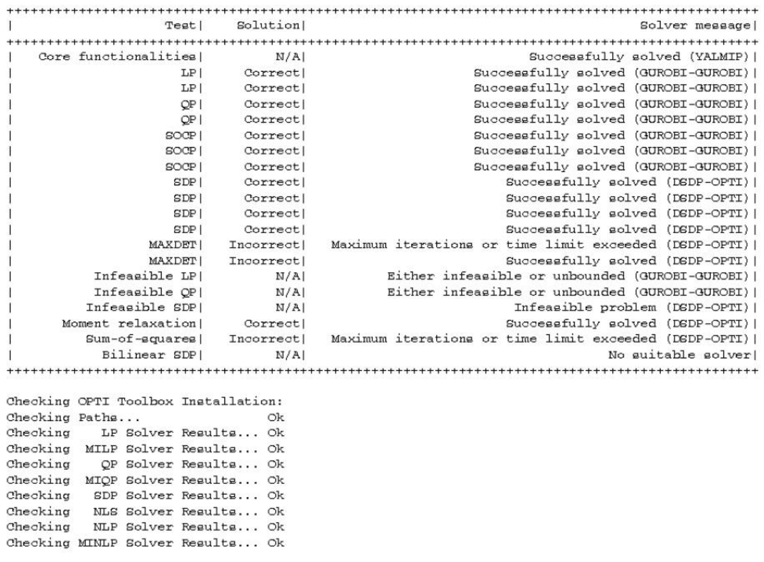
You can read information on the tool-kit by typing dynareOBC into the MATLAB command window. The first step is to test the Dynare installation by typing:

```
>> dynareOBC testsolvers
```

which will attempt to solve a number of computational problems and check whether the required solvers are working correctly.

After running TestSolvers, it is a good idea to clean up the path using:

```
>> dynareOBC rmpath
```



```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|                    Test|   Solution|                                         Solver message|
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
|    Core functionalities|        N/A|                           Successfully solved (YALMIP)|
|                      LP|    Correct|                    Successfully solved (GUROBI-GUROBI)|
|                      LP|    Correct|                    Successfully solved (GUROBI-GUROBI)|
|                      QP|    Correct|                    Successfully solved (GUROBI-GUROBI)|
|                      QP|    Correct|                    Successfully solved (GUROBI-GUROBI)|
|                    SOCP|    Correct|                    Successfully solved (GUROBI-GUROBI)|
|                    SOCP|    Correct|                    Successfully solved (GUROBI-GUROBI)|
|                    SOCP|    Correct|                    Successfully solved (GUROBI-GUROBI)|
|                     SDP|    Correct|                        Successfully solved (DSDP-OPTI)|
|                     SDP|    Correct|                        Successfully solved (DSDP-OPTI)|
|                     SDP|    Correct|                        Successfully solved (DSDP-OPTI)|
|                     SDP|    Correct|                        Successfully solved (DSDP-OPTI)|
|                  MAXDET|  Incorrect|  Maximum iterations or time limit exceeded (DSDP-OPTI)|
|                  MAXDET|  Incorrect|                        Successfully solved (DSDP-OPTI)|
|           Infeasible LP|        N/A|         Either infeasible or unbounded (GUROBI-GUROBI)|
|           Infeasible QP|        N/A|         Either infeasible or unbounded (GUROBI-GUROBI)|
|          Infeasible SDP|        N/A|                         Infeasible problem (DSDP-OPTI)|
|       Moment relaxation|    Correct|                        Successfully solved (DSDP-OPTI)|
|          Sum-of-squares|  Incorrect|  Maximum iterations or time limit exceeded (DSDP-OPTI)|
|            Bilinear SDP|        N/A|                                     No suitable solver|
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Checking OPTI Toolbox Installation:
Checking Paths...                Ok
Checking    LP Solver Results... Ok
Checking  MILP Solver Results... Ok
Checking    QP Solver Results... Ok
Checking  MIQP Solver Results... Ok
Checking   SDP Solver Results... Ok
Checking   NLS Solver Results... Ok
Checking   NLP Solver Results... Ok
Checking MINLP Solver Results... Ok
```

the basics of dynareOBC usage

The bounded variable simply has to be written in the .mod file as we have above. For example, to have interest rates set by a Taylor rule but bounded at zero, dropping the model-local variable z_t, we include the equation as shown in the following listing

```
r = max( rho_r*r(-1) + (1 - rho_r)*( r_steady + phi_pi*(pi -
pi_steady) + phi_y*(y - y_steady) ) + m,0);
```

The model is then run from the MATLAB command window with the command *dynareOBC*[*FILENAME*].*mod*[*OPTIONS*]. It should be noted that not all stoch_simul options are supported by dynareOBC, and currently no warning will be generated if unsupported options are used.

Suppose you have the NK ZLB model defined in the file NK_ZLB.mod. Typing

```
>> dynareOBC NK_ZLB.mod shockscale=10
```

will simulate the model without using cubature. If irf=40 or some other positive integer, IRFs will computed, and if, for example, periods=1000 then the model will use the extended-path type method to simulate the model over 1000 periods. The option shockscale=10 scales up the IRF shock by a factor of 10. Note that when you list options after the model file name that have '=' you must make sure to not include spaces around the equals sign. The options are not case sensitive.

If IRFs are computed, unless otherwise specified in the *stoch_simul*() command, you will see the IRF plots which have the paths with (solid line) and without (dotted line) the bounds. Figure 4.4 shows the plots again for output, inflation and the interest rate. As usual, the IRFs are stored in *oo_.irfs*.

In addition to the *oo_*. and *M_*. structures, dynareOBC stores results and data in a structure *dynareOBC_*.. For instance, the IRFs without the bounds are stored in *dynareOBC_.IRFsWithoutBounds*, and the M matrix is stored in *dynareOBC_.MMatrix*.

Estimation of Models with OBCs

Traditional linear DSGE models are estimated using the Kalman filter. This gives a very computationally efficient way to evaluate the exact likelihood of a linear-Gaussian model. I.e., given guesses for the parameters of the model, the Kalman filter returns the probability density of observing what we did,

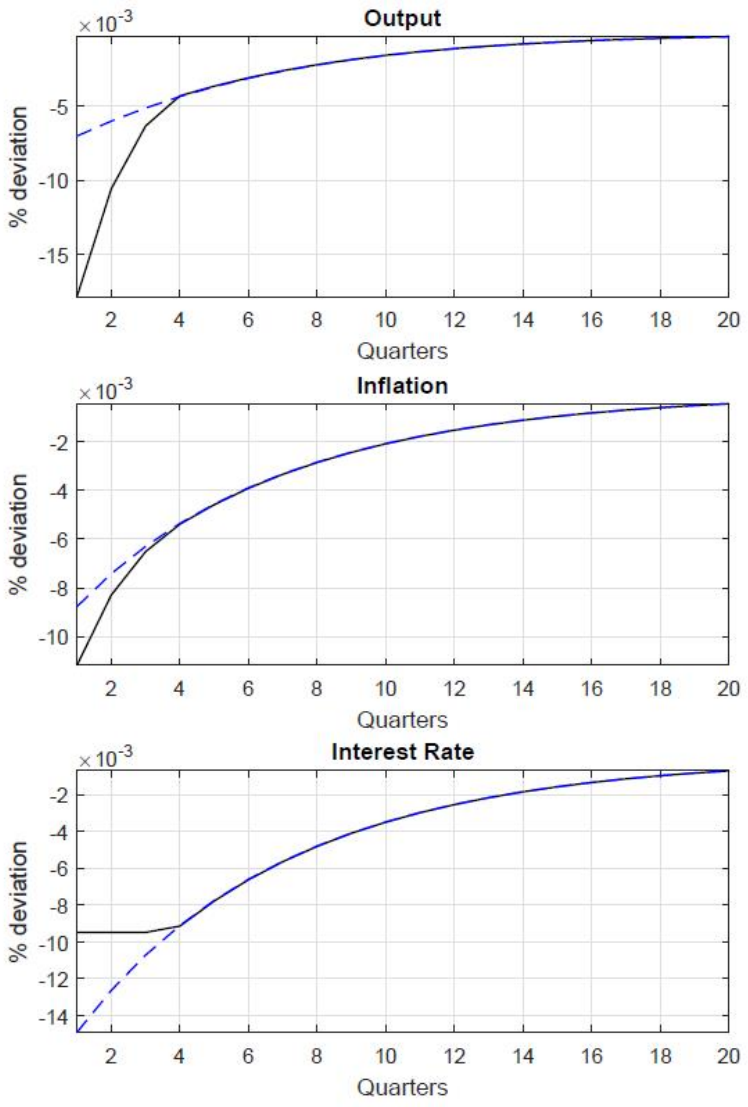


Figure 4.4: dynareOBC IRF simulation without cubature

given those parameters. Unfortunately, the derivation of the Kalman filter relies on the special properties of linear models with Gaussian shocks, and once any non-linearity is introduced, the Kalman filter is no longer applicable.

In a non-linear world then, an another approach must be taken. Here, we propose one which returns an approximation to the likelihood that requires relatively few simulation steps of the non-linear model, without overly distorting the shape of the likelihood function. Given how computationally expensive it is to simulate a DSGE model with an occasionally binding constraint, minimising the amount required is crucial for our application.

The central trick of the cubature Kalman filter approach is that for many non-linear models, even after having made several observations, the distribution of the state variables conditional on the parameters and on our observations is still approximately Gaussian. Thus although the distribution of the state will never be exactly Gaussian in a non-linear world, perhaps we will not lose to much if we nonetheless assume that it is.

DynareOBC includes built in code implementing the cubature Kalman filter. This enables the estimation of models with or without occasionally binding constraints, at up to a third order approximation. We will test this facility on a simple RBC model with irreversible investment, inelastic labour supply, and productivity known one period in advance.

We simulate some data from the model by changing to the Estimation folder. IrrevInv.mod is the file shown in the following listing:

```
var a M k r;

varexo epsilon;

parameters alpha beta delta mu sigma;

alpha = 0.3;
beta = 0.99;
delta = 0.0025;
mu = 0;
sigma = 0.075;
```

```

model;
#A = exp( a );
#LAG_A = exp( a(-1) );
#LEAD_K = exp( k(+1) );
#K = exp( k );
#LAG_K = exp( k(-1) );
#I = K - ( 1 - delta ) * LAG_K;
#LEAD_I = LEAD_K - ( 1 - delta ) * K;
#C = LAG_A ^ ( 1 - alpha ) * LAG_K ^ alpha - I;
#LEAD_C = A ^ ( 1 - alpha ) * K ^ alpha- LEAD_I;
#R = exp( r );
a = mu + sigma * epsilon;
I - C * ( C + I ) * M = C * ( C + I ) * beta * ( 1 / LEAD_C * (alpha * A ^ ( 1 - alpha ) * K ^ ( alpha - 1 ) + 1 - delta ) - ( 1 - delta ) * M(+1) ) - C;
I = max( 0, C * ( C + I ) * beta * ( 1 / LEAD_C * ( alpha * A ^( 1 - alpha ) * K ^ ( alpha - 1 ) + 1 - delta ) - ( 1 - delta ) * M(+1) ) - C );
1 / C = beta * R / LEAD_C;
end;

steady_state_model;
a = mu;
M = 0;
k = mu + 1 / ( alpha - 1 ) * log( ( 1 / beta - ( 1 - delta ) ) / alpha );
r = -log( beta );
end;

shocks;
var epsilon = 1;
end;
```

```

steady;
check;

stoch_simul( pruning, order = 3, irf = 0, periods = 500 ) C I R;
```

Typing in Matlab command:

```
>> dynareOBC IrrevInv.mod mlvsimulationmode=1
compilesimulationcode nocubature
```

After dynareOBC completes, we extract and save the resulting simulations (discarding the first 100 periods) using the code:

```
mlv = dynareOBC_.MLVSimulationWithBounds;
data = [ mlv.C', mlv.I', mlv.R' ];
data = data( 101:500, : );
xlswrite( 'IrrevInvTemp.xlsx', data );
```

While native Dynare relies on the estimated_params and estimation sections of the .mod file for specifying what exactly needs to be estimated, dynareOBC gets all of this information from the Excel spreadsheet containing the data. DynareOBC expects the Excel spreadsheet to contain two worksheets within it, where the first one has the names of the endogenous variables in the first row, and data in subsequent rows, and where the second contains the names of the parameters to be estimated in the first row, and their lower and upper bounds in the second and third rows respectively (with empty cells being interpreted as unbounded). Thus, we open IrrevInvTemp.xlsx in Excel and insert a new row into the top of the first sheet containing:

C	I	R

We then insert a second worksheet, containing the data:

alpha	beta	delta	mu	sigma
0	0	0		0
1	1	1		

This specifies all parameters except σ to be bounded between 0 and 1, while σ is just constrained to be positive. We then save the result as *IrrevInv.xlsx*. By default, when dynareOBC’ s *estimation* command line option is specified, dynareOBC automatically looks for a file with the same name as the .mod file, but with an .xlsx extension. Hence, with this file name, the data will be loaded automatically by dynareOBC. Alternatively, an explicit filename may be provided to dynareOBC with the *EstimationDataFile* = *FILE_NAME.xlsx* command line option. To run the estimation, we invoke dynareOBC with:

```
>> dynareOBC IrrevInv.mod estimation nocubature
```

4.2.5 Markov-Switching Approach

TBC

Chapter 5

Solving and Estimating Heterogenous Agent Model

5.1 Introduction

A cornerstone of thoughtful and lazy criticisms of mainstream macroeconomics alike is the idea that macroeconomists have spent 40 years just writing infinitely lived representative agent models (in which all agents behave the same way and are eternal), and isn' t it a ridiculous assumption? To which mainstream macroeconomics invariably respond: "heterogeneous agent models are all over the place," which is in turn invariably met with "yes, but this is a very recent development." I have myself long considered the development of heterogeneous agent macro as a response to the 2008 crisis. Then I began working on the history of economics at Minnesota, and I realized that by the mid-1980s, heterogeneous agent models were already all over the place. Much of the current discussions on the-state-of-macro are premised on a flawed historical narrative. And acknowledging that representative agent models have long coexisted with heterogeneous agent models of various stripes raises a host of new questions for critics and proponents of mainstream macro.

5.1.1 Heterogeneous agent models: the Minnesota genealogy

In 1977, Cowles economist Truman Bewley was looking to microfound the permanent income hypothesis (which basically state that agents try to smooth their consumption over time by saving and dis-saving). He came up with a model in which agents are heterogeneous in the income fluctuations they face (back then you didn' t call these "shocks" yet), and, crucially, who are not allowed to borrow. Though he subsequently moved from general equilibrium theory to survey-based investigation of why wage are sticky, his earlier work gave rise to a class of models named after him by Lars Ljundqvist and Tom Sargent. Their characteristic was that market incompleteness, in particular incomplete insurance, creates heterogeneity in the kind of shocks otherwise initially identical agents reacted to, so that their ex-post wealth follows a distribution that needs to be characterized. Major contributions to this literature included landmark works by former physics student Rao Aiyagari (1981 Minnesota PhD under Wallace) and Mark Huggett (1991 Minnesota PhD under Prescott) in the early 1990s.

Huggett wanted to understand why risk-free interest were on average lower than what calibrated representative-agent models predict. He constructed a model in which households face idiosyncratic income endowment shocks that they can' t fully insure against because they face a borrowing constraint, and showed that this results in higher precautionary savings (so that they can later smooth consumption in spite of uncertainty), thus a lower risk-free rate. "A low risk-free rate is needed to persuade agents not to accumulate large credit balances so that the credit market can clear," he explained. Aiyagari intended to study how an economy with a large number of agents behaved, hoping to bridge the gap between representative-agent models and observed patterns in individual behavior. He also wanted to study how individual risk affect aggregate saving (little, he found). He wrote a production model where agents differ in that they face uninsured idiosyncratic labor endowment shocks and trade assets between themselves. Since agent save more, the capital stock, the wage rate and the labor productivity are all higher, he showed. He also used a Bewley model to show that if market were incomplete, capital income could be taxed. This challenged the famous result independently reached by Kenneth Judd and Christophe Chamley (in a representative agent setting) that positive capital taxation is suboptimal.

Minnesota macroeconomists in the 1980s were fond of another type of model in which heterogeneity bred restricted participation to markets, leading to lack of insurance against extrinsic uncertainty (that is shocks affecting nonfundamental variables). The heterogeneity was in age, and it was generated by the coexistence of at least two generations of otherwise identical agents in each one of an infinite succession of periods. Since each agent is born then dies, this de facto restricts participation to past markets. What came to be known as overlapping generation models were originally engineered by Maurice Allais in 1947 and Paul Samuelson in 1958. Samuelson' s consumption-loan model allowed him to investigate the determination of interest rates (he identified an autarkic equilibrium in which money has no value because agents consume their endowment and another one in which money is used to delay consumption). His purpose was to 'point up a fundamental and intrinsic deficiency in a free price system …no tendency to get you to positions on the [Pareto efficiency frontier] that are ethically optimal." OLG has subsequently been wedded to the examination of the role of money in the economy – it was used in the famous 1972 paper by Lucas, the original model in which Cass and Shell framed their sunspots (they argued that OLG was the "only dynamic disaggregated macroeconomic model") and it was the core of a conference on microfounding the role of money organized by John Kareken and Neil Wallace at the Federal Bank of Minneapolis in 1978 (Bewley, Cass and Shell contributed to the resulting volume). Aiyagari, and many others (half of Wallace' s PhD students) at Minnesota, had spent years comparing the properties of infinitely-lived agent models and OLGs.

There were several reasons to bring heterogeneity into general equilibrium models. Some researchers wanted to study its consequences on the level and dynamics of price and quantities, others were interested in understanding the effects of business cycles on the welfare of various types of consumers (something that governments might want to offset by removing some risks agent faced, through social security for instance). It was the motive behind the dissertation Ayse Imrohoroglu completed at Minnesota in 1988 under Prescott. One of her papers pushed back against Lucas' 1987 conclusion that the business cycle did not really affect aggregate consumption. She wrote a model where variable probabilities to find a job create some idiosyncratic income uncertainty which agents cannot completely offset because they have borrowing constraints. She concluded that in specific settings and for some parameters, the welfare cost of business cycle was significant ($128 per person, 5 time larger than the one in an economy with perfect insurance).

Per Krusell (1992 Minnesota PhD under Wallace) and Carnegie economist Tony Smith were also concerned with the consequences of heterogeneity on the business cycle and its welfare implications. Their agenda was to check whether a heterogeneous agent model fared better than a representative agent one when it came to replicate the behavior of macro aggregates. They used a production model in which households face idiosyncratic employment shocks with borrowing restriction. These agents consequently hold different positions in the wealth distribution, with some of them ‘rich’ and some of them ‘poor.’ They also added an aggregate productivity shock, and, in a spinoff of the model, differences in agents’ random discount factor (their degree of patience).

They found out that when shocks are calibrated to mimic real GDP fluctuations, the resulting overall wealth distribution is close to the real-world one. They noted that the resulting level and dynamics of aggregates was not substantially different from what was obtained with a representative agent model, a result that was later attributed to their calibration choices. Furthermore, they explained that “the distribution of aggregate wealth was almost completely irrelevant for how the aggregates behave in the equilibrium” (because the value of the shocks they chose was not that big, agents ended up insured enough so that their marginal propensity to save was largely independent from their actual wealth and income, except for the poorest, who don’ t weigh a lot in aggregate wealth anyway. The borrowing constraint didn’ t play a big role in the end).

In a follow-up survey, Krusell and Smith made it clear that their purpose was not “to provide a detailed assessment of the extent to which inequality influences the macroeconomy …[or] how inequality is determined.” It seems to me that back then, studying inequality in wealth, income, wage was not the main motive for developing these models (Aiyagari excepted). The growing amount of micro data produced, in particular through the US Panel Study on Income Dynamics initiated by Michigan economist Jim Morgan in the wake of Johnson’ s War on Poverty, provided a new set of facts calibrators were challenged to replicate. These included a more disagregated picture of income and wealth distribution. If Bewley models featured prominently in Ljungqvist and Sargent's 2000 Recursive Macroeconomic Theory textbook, thus, it was because it was necessary to match the "ample evidence that individual households' positions within the distribution of wealth move over time." Macroeconomists' motives to use heterogeneous agent models gradually shifted, as they became more directly interested in the two-ways relations between inequalities and aggregate fluctuations. Other types of heterogeneity were introduced, in the demographic structure, the type of shock agent face (fiscal and productivity shocks among others) and their innate characteristics (in the liquidity of their asset portfolios, their marginal propensity to consume, their health, their preferences, etc.).

Innovations were much needed to solve these models, as aggregate equilibrium prices depended not just on exogenous variables, but also on the entire wealth and income distribution of agents that endogenously changes over time. This meant that solutions could not be analytically derived. If Krusell and Smith's work proved so influential, it wasn't merely because they proposed a new model, but also because they built on numerical methods to provide an iterative numerical solution algorithm (based on their idea that the only thing that was necessary for agents to make consumption decisions – thus for them to compute the solution – was the mean of the wealth distribution, which determined future interest rates). The development of heterogeneous agent macro models from the 1990s onward therefore paired theoretical and computational investigations.

In principle, Dynare can handle many state variables and, thus, should be able to solve models with multiple agents. The problem is that in macroeconomic models we would like to have millions of agents and such a large number of agents is problematic, even for Dynare.[1]The standard approach is to approximate these millions of agents as a continuum, but Dynare requires a finite number of agents.

[1]An interesting question is whether using a relatively small number, say 100, is enough to approximate a macroeconomy with millions of agents. It would be cumbersome to use Dynare to solve models with 100 agents, but this is likely to be feasible at least for some models and for lower-order perturbation solutions.

5.2 Gali(2018)'s TANK

If there are heterogeneous agents in the model, the degree of heterogeneity is limited, for example, to just having two representative agents, such as a patient and an impatient agent.

5.3 den Haan and Ocaktan(2011) Approach

In this section, we show how to use Dynare to solve models with (i) a continuum of heterogeneous agents and (ii) aggregate risk. To do this, two things are needed.

- First, a mother program is needed to solve for the laws of motion that describe the aggregate variables taking the solution of the individual policy rules as given.
- Second, a Dynare program is needed to solve for the individual policy rules taking the aggregate law of motion as given. This Dynare program, the *.mod file, needs to be able to read the coefficients of the aggregate law of motion.

We show that this can be accomplished with an easy procedure for a Dynare program and with a somewhat more cumbersome procedure for a Dynare++ program.

5.3.1 The Model

The model applied in this section describes a simple exchange economy with incomplete markets, aggregate uncertainty and an infinite number of agents. The source of heterogeneity comes from the assumption that there are idiosyncratic income shocks, which are partially uninsurable.

Individual Household The economy is represented by a stochastic growth model with a continuum of individuals indexed by $i \in [0, 1]$. The individual agents are characterized as facing an idiosyncratic unemployment risk. All agents are ex ante identical, however there is ex post heterogeneity due to incomplete insurance markets and borrowing constraints. Every quarter, individuals differ from each other through their asset holdings and employment opportunities. In order to transfer their ressources over time, agents can only control their capital holdings. We assume that an employed agent earns a wage rate of w, while an unemployed agent has no income. However, agents can insure themselves, at least partially, against employment risks by building up a capital stock. To insure satisfaction of intertemporal budget constraints, capital holdings are restricted by a borrowing limit $b \geq 0$, ensuring the repayment of loans and the absence of Ponzi schemes.

Agent i’ s maximization problem is given by

$$\underbrace{max}_{c_{i,t},a_{i,t+1}} \quad E_t \sum_{t=0}^{\infty} \beta^t \frac{c_{i,t}^{1-\gamma} - 1}{1-\gamma}$$

s.t.

$$c_{i,t} + a_{i,t+1} = r(k_t, l_t, z_t)a_{i,t} + w(k_t, l_t, z_t)e_{i,t}\hat{l} + (1-\delta)a_{i,t}$$

$$a_{i,t} \geq b$$

Each agent faces partially insurable labor market income risk and is endowed with one unit of time. This endowment is transformed into labor input according to $l_{it} = e_{it}\hat{l}$. The stochastic employment opportunity e_{it} follows an autoregressive process of the form

$$e_{i,t+1} = (1-\rho_e)\mu_e + \rho_e e_{i,t} + \epsilon^e_{i,t+1}$$

Firm Let k_t and l_t stand for per capita capital and the employment rate, respectively. Per capita aggregate output takes as inputs the aggregate capital stock and labor supply

$$y_t = z_t k_t^{\alpha} l_t^{1-\alpha}$$

the aggregate technology shock z_t, common to all households, satisfies

$$z_{t+1} = (1-\rho_z)\mu_z + \rho_z z_t + \epsilon^z_{t+1}$$

We assume that firms rent their factors of production from households in competitive spot markets. These aggregate inputs imply market interest and wage rates equal to

$$r(k_t, l_t, z_t) = \alpha z_t \left(\frac{t}{l_t}\right)^{alpha-1}$$

$$w(k_t, l_t, z_t) = (1-\alpha) z_t \left(\frac{t}{l_t}\right)^{alpha}$$

In order to solve the optimization program given agents must forecast future prices. Under the maintained assumptions (l_t, z_t) follow an exogeneous stochastic process. Therefore in order to forecast future wage and rental rates, agents must know the stochastic process that describes the evolution of the aggregate capital stock. However, the stochastic properties of the aggregate capital stock depend on the distribution of capital holdings in the population. As a consequence, the whole capital distribution becomes a state variable. In a setup with a continuum of agents, capital distribution is a function, which cannot be used as an argument of the individual policy rules. Krusell and Smith (1998) propose to summarize this distribution by a discrete and finite set of moments. For instance, if we take into consideration only the first order moments, the law of motion for aggregate capital, k_{t+1}, is given by

$$k_{t+1} = \zeta_0 + \sum_{i=1}^{I} \zeta_i M(i) + \zeta_{I+1} z_t$$

where M(i) is the cross-sectional average of a_i with $k = M(1)$, and s is a vector containing the aggregate state variables.

5.3.2 Implementing XPA with Dynare

The Dynare program itself that solves for the individual laws of motion would be almost a standard Dynare program. However, it differs from a standard Dynare program in two aspects.

- the program needs to be able to incorporate the values of the coefficients of the aggregate laws of motion.

- The second modification related to XPA is that auxiliary policy rules are needed in order to explicitly aggregate higher-order individual policy functions.

  For example, if the model has a policy rule for the capital choice, k, then one needs an auxiliary policy rule for k^2 is needed if a second-order perturbation solution is used. This would result in simply adding one variable and one equation to the standard Dynare file.

The two-part mother program would be very simple.

- The first part would contain a few steps of algebra to get the coefficients of the aggregate laws of motion from the individual laws of motion.

- The second part has to contain a procedure to make the laws of motion of the aggregate variables consistent with the laws of motion of the individual variables.

  For example, one might start with an aggregate policy rule, solve for the individual policy rule using perturbation techniques, obtain the aggregate law of motion by explicit aggregation of the individual policy rule, and iterate until the aggregate law of motion has converged.

5.4 The LR(2017) Approach

TBC

5.5 The Winberry Approach

TBC

Chapter 6

Optimal Policy and Welfare Analysis

Wenli Xu
Anhui University and CIMERS
xuweny87@hotmail.com

6.1 Introduction

During the ten years before the Globe Financial Crisis, the most common framework employed in Macroeconomics had incorporated price/wage rigidity into the DSGE models. We developed a basic NK model following Walsh(2015: Monetary theory and policy, Chapter 8) and Sims(2017, Course notes,A New Keyesian Model with price stickiness). The basic NK model lay out in the section has no investment or capital.This follows McCallum and Nelson(1999) and Walsh(2015), there is little relationship between the capital stock and output at business cycle frequencies. This simplifies the analysis and permits ones to get better intuition. The price stickiness used here is due to Calvo(1983).

6.2 The Model

6.2.1 Households

The representative household maximize the expected present discounted value of utility:

$$\underbrace{max}_{C_t,N_t} \; E_t \sum_{i=0}^{\infty} \beta^i \left(\frac{C_{t+i}^{1-\sigma}}{1-\sigma} - \chi \frac{N_{t+i}^{1+\eta}}{1+\eta} \right) \tag{6.1}$$

Household face an intertemporal problem, they find the value of consumption C_t, hours worked N_t and real bonds B_{t+1} which delivers the highest value of utility. The budget constrain of the household is

$$(1+\tau^c)P_tC_t + B_{t+1} = (1-\tau^n)_t w_t N_t + (1+r_{t-1})B_t + P_t\Pi_t - P_tT_t \tag{6.2}$$

where P_t is the general (aggregate) price level, w_t is the real wage and r_t is the real interest rate of Bond, Π_t are the real profits from intermediate firms and T_t is a lump sum tax paid to a government.

The FOCs of households are:

$$\chi N_t^{\eta} = \frac{1-\tau^n}{1+\tau^c} C_t^{-\sigma} w_t \tag{6.3}$$

$$P_t^{-1} C_t^{-\sigma} = \beta E_t P_{t+1}^{-1} C_{t+1}^{-\sigma} (1+r_t) \tag{6.4}$$

6.2.2 Firms

The production of goods is split into two types: final production and intermediate production. Intermediate firms (or wholesale goods products) produce different types of goods which are imperfect substitutes (at the origin of the monopolistic competition). Final firms (or retailers) produce an homogenous good by combining intermediate goods in a CES technology. Because of imperfect substitutes of the intermediates in final production process, these intermediate firms have the pricing power. These intermediate firms produce output only using labor and subject to an aggregate productive shock. They are no freely able to adjust prices each period, but in Calvo's pricing.

Final Good Firm

The final products are a CES form of a continuum of intermediate goods:

$$Y_t = \left(\int_0^1 Y_t(j)^{\frac{\epsilon-1}{\epsilon}} dj \right)^{\frac{\epsilon}{\epsilon-1}} \tag{6.5}$$

Following the profit maximization of the final good firm, we find the intermediates demand function for each variety j is:

$$Y_t(j) = \left(\frac{P_{j,t}}{P_t} \right)^{-\epsilon} Y_t \tag{6.6}$$

where $P_{j,t}$ is the price of variety j, P_t is the general (aggregate) price level:

$$P_t = \left(\int_0^1 P_{j,t}^{1-\epsilon} dj \right)^{\frac{1}{1-\epsilon}} \tag{6.7}$$

Intermediate Goods Firms

Intermediate producers have the following production technology:

$$Y_{j,t} = A_t N_{j,t} \tag{6.8}$$

where $Y_{j,t}$ is the intermediate goods, A_t is a common productive shock following AR(1) process:

$$lnA_t = \rho_a lnA_{t-1} + \epsilon_{a,t} \tag{6.9}$$

Intermediate products firms solve a two-stage problem. In the first stage, minimizing cost leads to the real marginal cost:

$$mc_{j,t} = \frac{w_t}{A_t}$$

Here We can drop the j index of marginal cost, because both w_t and A_t are common to all intermediate goods firms. This is :

$$mc_t = \frac{w_t}{A_t} \tag{6.10}$$

In the second stage, these intermediate producers decide their prices of intermediate goods in a Calvo(1983) pricing. Following the dynamic problem of an updating firms, we can write the express of the reset prices:

$$P_t^* = \frac{\epsilon}{\epsilon - 1}\frac{X_{1,t}}{X_{2,t}} \tag{6.11}$$

where the recursive forms of $X_{1,t}, X_{2,t}$ are

$$X_{1,t} = C_t^{-\sigma} mc_t P_t^{\epsilon} Y_t + \phi\beta E_t X_{1,t+1} \tag{6.12}$$

$$X_{2,t} = C_t^{-\sigma} P_t^{\epsilon-1} Y_t + \phi\beta E_t X_{2,t+1} \tag{6.13}$$

Here ϕ is the fraction of non-updated firms.

6.2.3 Aggregation

The government's budget constraint in nominal terms is there:

$$P_t G_t + (1 + r_{t-1})B_t = PtT_t + B_{t+1} + \tau^c P_t C_t + \tau^n P_t w_t N_t$$

where G_t is the government's spending. It is procyclical:

$$G_t = \omega Y_t \tag{6.14}$$

Where ω is a parameter which govern the sensitivity to fluctuation of output.

To close the model, let's assume the interest rule follows the Taylor rule:

$$r_t = (1 - \rho_r)r + \rho_r r_{t-1} + (1 - \rho_r)\left[\phi_\pi(\pi_t - \pi) + \phi_y\left(lnY_t - lnY\right)\right] + \epsilon_{r,t} \tag{6.15}$$

where π_t is the inflation rate, this is, $1 + \pi_t = \frac{P_t}{P_{t-1}}$. π_{target} is the inflation target.ϵ_r is a monetary shock.

The resource constraint is given by:

$$Y_t = C_t + G_t \tag{6.16}$$

6.2.4 Equilibrium

Define $v_t^p = \int_0^1 \left(\frac{P_{j,t}}{P_t}\right)^{-\epsilon} dj$, $1 + \pi_t^* = \frac{P_t^*}{P_{t-1}}$, $x_{1,t} = \frac{X_{1,t}}{P_t}, x_{2,t} = \frac{X_{2,t}}{P_t}$. So the full set of equilibrium conditions reads as follows:

$$\chi N_t^{\eta} = \frac{1 - \tau^n}{1 + \tau^c} C_t^{-\sigma} w_t \tag{6.17}$$

$$C_t^{-\sigma} = \beta E_t (1+\pi_{t+1})^{-1} C_{t+1}^{-\sigma}(1+r_t) \tag{6.18}$$

$$Y_t = \frac{A_t N_t}{v_t^p} \tag{6.19}$$

$$v_t^p = (1-\phi)(1+\pi_t^*)^{-\epsilon}(1+\pi_t)^{\epsilon} + (1+\pi_t)^{\epsilon}\phi v_{t-1}^p \tag{6.20}$$

$$(1+\pi_t)^{1-\epsilon} = (1-\phi)(1+\pi_t^*)^{1-\epsilon} + \phi \tag{6.21}$$

$$1+\pi_t^* = \frac{\epsilon}{\epsilon-1}(1+\pi_t)\frac{x_{1,t}}{x_{2,t}} \tag{6.22}$$

$$x_{1,t} = C_t^{-\sigma} mc_t Y_t + \phi\beta E_t (1+\pi_{t+1})^{\epsilon} x_{1,t+1} \tag{6.23}$$

$$x_{2,t} = C_t^{-\sigma} Y_t + \phi\beta E_t (1+\pi_{t+1})^{\epsilon-1} x_{2,t+1} \tag{6.24}$$

$$Y_t = C_t + G_t \tag{6.25}$$

$$G_t = \omega Y_t \tag{6.26}$$

$$mc_t = \frac{w_t}{A_t} \tag{6.27}$$

$$lnA_t = \rho_a lnA_{t-1} + \epsilon_{a,t} \tag{6.28}$$

$$r_t = (1-\rho_r)r + \rho_r r_{t-1} + (1-\rho_r)\left[\phi_\pi(\pi_t - \pi) + \phi_y\left(lnY_t - lnY\right)\right] + \epsilon_{r,t} \tag{6.29}$$

There are 13 endogenous variables in 13 equations:
$N_t, C_t, w_t, r_t \pi_t, Y_t, A_t, v_t^p, \pi_t^*, x_{1,t}, x_{2,t}, G_t, mc_t$.
There are 2 exogenous variables:ϵ_a, ϵ_r. The parameters of the model are:
$\chi, \eta, \sigma, \beta, \phi, \epsilon, \rho_a, \rho_r, \phi_\pi, \phi_y, \omega$.

6.3 What is the quadratic form for the objective function?

What are the appropriate objectives of the central bank? There is a long history in monetary policy analysis of assuming that the central bank is concerned with minimizing a quadratic loss function that depends on output and inflation(Walsh,2015). Although such an assumption is plausible, it is ultimately ad hoc. Woodford(2003) provides a path-breaking contribution regarding the microfoundation of optimal policies by deviating a loss function obtained making an approximation of the welfare criterion of households. We provide a same exercise as in Vermandel(2017)[1].

The method in the following analysis derives a quadratic loss function that represents a quadratic (second-order Taylor series) approximation to the level of expected utility of the representative household. We are not going to approximation up to second order all the equations, only focus on equations which really matters regarding the optimal conduct of optimal monetary policy[2].

6.3.1 A second order approximation

The Taylor series of a function $f(x)$ that is infinitely differentiable at a number a is the power series

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

Using the big O notation, the second order approximation of $f(x)$ can be written as

$$f(x) \simeq f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \parallel O_3 \parallel$$

where $\parallel O_3 \parallel$ is the higher order term. The unconditional expect of this function is given by:

[1] His website: http://vermandel.fr/

[2] The derivation by hand of a second order welfare criterion is most of the time arbitrary and may lead to fallacious results. Schmitt-Grohe and Uribe(2004) develops a accurate alternative by developping a second order approximation of all equilibrium conditions of the model (instead of ”most important” conditions).

$$\begin{aligned}
E[f(x)] &\simeq E\left[f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \parallel O_3 \parallel\right] \\
&\simeq f(a) + f'(a)E(x-a) + \frac{f''(a)}{2}E[(x-a)^2] + \parallel O_3 \parallel \\
&\simeq f(a) + \frac{f''(a)}{2}var(x)
\end{aligned}$$

Above, on average the linear deviations of x from its mean are asymptotically 0: $E(x-a) = 0$.

6.3.2 A derivation of quadratic welfare function

The representative household's utility function:

$$E_t \sum_{i=0}^{\infty} \beta^i \left(\frac{C_{t+i}^{1-\sigma}}{1-\sigma} - \chi \frac{N_{t+i}^{1+\eta}}{1+\eta} \right) \tag{6.30}$$

In general, assuming that the government sector cares about output gap and inflation. Why? If you call back to the CES aggregator over intermediate goods, you will note that the CES function is concave, meaning that households (or the final goods firms) would like to smooth over intermediate inputs. In a flexible price world, all intermediate producers would choose the same price (e.g. they all desire a relative price of 1). If aggregate inflation is different from zero, with price stickiness relative prices at the intermediate firm level get distorted (e.g. there is price dispersion). This leads to a non-smooth allocation of intermediates, which results in a welfare loss.

In the section, the goal at this point is to express the above welfare into output and inflation terms. Substituting (9.26) into (9.25): $C_t = (1-\omega)Y_t$, then substituting the consumption by above and labors by the production function (9.19), the welfare function reads as:

$$\begin{aligned}
Wel_t &= E_t \sum_{i=0}^{\infty} \beta^i \left\{ \frac{(1-\omega)^{1-\sigma}}{1-\sigma} Y_{t+i}^{1-\sigma} - \frac{\chi}{1+\eta} \left[\frac{Y_{t+i} v_{t+i}^p}{A_t} \right]^{1+\eta} \right\} \\
&= E_t \sum_{i=0}^{\infty} \beta^i \left\{ \frac{(1-\omega)^{1-\sigma}}{1-\sigma} f(Y_{t+i}) - \frac{\chi}{1+\eta} g(Y_{t+i}, \pi_{t+i}) \right\}
\end{aligned}$$

First of all, defining the steady state of endogenous variables: Y, C, N, π.

Then, using a second order expression:

$$f(Y_t) \simeq f(Y) + f'(Y)(Y_t - Y) + \frac{f''(Y)}{2}(Y_t - Y)^2 + \parallel O_3 \parallel$$

where $f'(Y)$ is the first order derivation of $f(Y_t)$ at the steady state. $f''(Y)$ is the second order derivation of $f(Y_t)$ at the steady state.

Then, the first term of the welfare can be approximation by:

$$E_t \sum_{i=0}^{\infty} \beta^i \left\{ \frac{(1-\omega)^{1-\sigma}}{1-\sigma} f(Y_{t+i}) \right\} \simeq \frac{1}{1-\beta} \frac{(1-\omega)^{1-\sigma}}{1-\sigma} \left(f(Y) + \frac{f''(Y)}{2} var(Y_t) \right) + \parallel O_3 \parallel + t.i.p.$$

where $t.i.p.$ indicates terms independent of the policy.

In the same vein, the second term reads as:

$$E_t \sum_{i=0}^{\infty} \beta^i \left\{ \frac{\chi}{1+\eta} g(Y_{t+i}, \pi_{t+i} \right\} \simeq \frac{1}{1-\beta} \frac{\chi}{1+\eta} \left(g(Y,\pi) + \frac{g''_{YY}(Y,\pi)}{2} var(Y_t) + \frac{g''_{\pi\pi}(Y,\pi)}{2} var(\pi_t) \right)$$
$$+ \parallel O_3 \parallel + t.i.p.$$

To summarize, the welfare follows

$$Wel_t \simeq \frac{1}{1-\beta} \left(\overline{U} + \frac{1}{2} \left(\gamma_1 var(Y_t) + \gamma_2 var(\pi_t) \right) \right) + \parallel O_3 \parallel + t.i.p.$$

The $\frac{1}{2}$ is just a scaling term that doesn't affect the optimum but simplifies things a lots. Where

$$\overline{U} = \frac{(1-\omega)^{1-\sigma}}{1-\sigma} f(Y) - \frac{\chi}{1+\eta} g(Y,\pi)$$

$$\gamma_1 = \frac{(1-\omega)^{1-\sigma}}{1-\sigma} f''(Y) - \frac{\chi}{1+\eta} g''_{YY}(Y,\pi)$$

$$\gamma_2 = -\frac{\chi}{1+\eta} g''_{\pi\pi}(Y,\pi)$$

Dynare has tools to compute optimal policies for various types of objectives. *ramsey_model* computes automatically the First Order Conditions (FOC) of a model, given the *planner_objective*. You can then use other standard commands to solve, estimate or simulate this new, expanded model.

In Dynare, for simulation, there are three types of optimal policy:

1. optimal simple rule (*osr*);

2. ramsey policy under commitment (*ramsey_policy*);

3. ramsey policy under discretion (*discretionary_policy*).

6.4 Optimal Policy

6.4.1 Optimal Simple Rule

Before Woodford(2003) provided micro foundation of the welfare function, the objectives as defined by an ad hoc loss function[3] don't depend on the utility of the representative households.

For instant, in optimal simple rule, the loss function follows as

$$\lambda_1 var(Y_t) + \lambda_2 var(\pi_t)$$

This defines the loss function as a quadratic loss function. It is worth note that the loss function does not need to be in terms of output and inflation. This is often just a convenient and intuitive representation. In generally, the objective is expressed as minimizing a weighted sum of variances and covariances of any endogenous variables.

First half of the .mod file

To input the above model into Dynare for optimal policy, we must declare the model' s variables in the preamble of the .mod file, declare the model in the model-block of the .mod file, declare the steady state or initial value in initval-block of the .mod file and declare shocks in shocks-block. This is done similarly as described in the previous chapter on solving/estimating DSGE models. We thus begin the .mod file with:

[3]What is a loss function in decision theory?
Defining a loss function as the "cost" incurred when the true value of θ is estimated by $\hat{\theta}$. A loss function is a mathematical representation of anything bad or at least undesirable: the point is that it is therefore something you want to minimise. Simple examples of a loss function arise when we consider the difference between some true or correct value θ and a estimation $\hat{\theta}$, which you would like to be as small as possible. Possible ways of taking that further are to work with $(\theta - \hat{\theta})^2$ or $|\theta - \hat{\theta}|$, where are both loss functions. In either case there is a minimum loss of 0 when $\hat{\theta} = 0$.

```
var C r pi N w mc A Y v pi_star x1 x2 G;

varexo e_a e_r;

parameters omegga betta fi siggma etta xi epsion thetta rho_a
rho_r fi_pi fi_y tau_n tau_c;

omegga=0.2;
betta=0.99;
fi=0.75;
siggma=1;
etta=1;
xi=1;
epsion=10;
thetta=1;
rho_a=0.95;
rho_r=0.9;
fi_pi=1.5;
fi_y=0.7;
tau_n=0.05;
tau_c=0.15;

model;

C^(-siggma)=betta*C(+1)^(-siggma)*(1+r)*(1+pi(1))^(-1);
xi*N^etta=(1-tau_n)/(1+tau_c)*C^(-siggma)*w;
mc=w/A;
C+G=Y;
G=omegga*Y;
Y=A*N/v;
v=(1-fi)*(1+pi_star)^(-epsion)*(1+pi)^epsion+(1+pi)^epsion*fi*v(-1);
(1+pi)^(1-epsion)=(1-fi)*(1+pi_star)^(1-epsion)+fi;
1+pi_star=epsion/(epsion-1)*(1+pi)*x1/x2;
```

```
x1=C^(-siggma)*mc*Y+fi*betta*(1+pi(1))^epsion*x1(1);
x2=C(-siggma)*Y+fi*betta*(1+pi(1))(epsion-1)*x2(1);
A=A(-1)^rho_a*exp(e_a);
r=(1-rho_r)*steady_state(r)+rho_r*r(-1)+(1-rho_r)*(fi_pi*(pi-steady_state(pi))+
fi_y*log(Y/steady_state(Y)))+e_r;

end;

initval;
A=1;
pi=0;
pi_star=0;
r=1/betta-1;
v=1;
mc=(epsion-1)/epsion;
w=mc;
N=0.9640;
Y=A*N/v;
C=(1-omegga)*Y;
x1=C^(-siggma)*mc*Y/(1-fi*betta);
x2=Ce xtasciicircum(-siggma)*Y/(1-fi*betta);
end;

steady;
check;
model_info;
model_diagnostics;

shocks;
var e_a;
stderr 0.01;
var e_r;
stderr 0.01;
end;
```

Computation of Optimal Simple Rule

The computation of optimal simple rules is controlled by main command *osr* and auxiliary commands *optim_weights*, *osr_params* and *osr_params_bounds*:

- *optim_weights* enumerates the non-zero elements of the weighting matrix used for the quadratic objective function;
- *osr_params* lists the subset of the model parameters over which the optimal simple rule is to be optimized;
- *osr* controls the computation of the optimal simple rule and displays results.
  The osr command will subsequently run stoch_simul and accepts the same options, including restricting the endogenous variables by listing them after the command, as stoch_simul plus
  - `opt_algo = INTEGER` Specifies the optimizer for minimizing the objective function. The same solvers as for mode_compute are available, except for 5,6, and 10.
  - `optim = (NAME, VALUE, ...)` A list of NAME and VALUE pairs. Can be used to set options for the optimization routines.
  - `silent_optimizer`
  - `huge_number = DOUBLE` Value for replacing the infinite bounds on parameters by finite numbers.

Thus, we end the .mod file with:

```
optim_weights;
pi 1.5;
Y 1;
end;
```

```
osr_params fi_pi fi_y;

osr Y;
```

This defines the loss function as being $var(Y_t) + 1.5var(\pi_t)$. The *osr* procedure must find the optimal value of parameters ϕ_π and ϕ_y.

In addition, we may use *osr_params_bounds* to declare lower and upper bounds for parameters in the optimal simple rule. Note that the use of this block requires the use of a constrained optimizer, i.e. setting [opt_algo], to 1,2,5, or 9. If not specified the optimization is unconstrained.

e.g.

```
osr_params fi_pi fi_y;

osr_params_bounds;
fi_pi,1,3;
end;

osr(opt_algo=9) Y;
```

This indicates that the parameter ϕ_π is limited to the range of [1,3].

Remark!

- *osr* uses numerical optimization, but numerical optimizers only find local optima. It is the responsibility of the user to insure that the optimum return by *osr* is a global optimum. At minimum, the user should at minimum try different initial values for the parameters of the policy rule.

- A better practice is to explore the space of parameters before calling *osr* and use this exploration to set the initial values for the parameters of the optimal rule.

Note!

The linear quadratic problem consists of choosing a subset of model parameters(e.g. ϕ_π, ϕ_y) to minimize the weighted (co)-variance of a specified subset of endogenous variables(e.g. Y_t, π_t), subject to a linear law of motion implied by the first order conditions of the model. A few things are worth mentioning.

1. the selected endogenous variables' deviations from their steady state, i.e. in case, they are not already mean 0, the variables entering the loss function are automatically demeaned so that the centered second moments are minimized.

2. *osr* only solves linear quadratic problems of the type resulting from combining the specified quadratic loss function with a first order approximation to the model' s equilibrium conditions. The reason is that the first order state-space representation is used to compute the unconditional (co)-variances. Hence, osr will automatically select $order = 1$.

3. because the objective involves minimizing a weighted sum of unconditional second moments, those second moments must be finite. In particular, unit roots in the selected endogenous variables are not allowed.

Output

Command osr produces the same output as *stoch_simul* and, in addition,

- the optimal value of the parameters;
- the value of the loss function at the optimum.

After *osr*, the value of the parameters is set to their optimal values.

6.4.2 Ramsey Policy under Commitment

Ramsey policy solves the following optimal control problem:

$$\underbrace{max}_{\{y_\tau\}_{\tau=0}^{\infty}} E_t \sum_{\tau=t}^{\infty} \beta^{\tau-t} U(y_\tau)$$

s.t.

$$E_\tau f(y_{\tau+1}, y_\tau, y_{\tau-1}, \epsilon_\tau) = 0$$

where y_t is a vector of n endogenous variables, U(y) is the objective function of the policy maker, β is discount factor.

$E_\tau f(y_{\tau+1}, y_\tau, y_{\tau-1}, \epsilon_\tau) = 0$ represents the set of constraints imposed to the policy maker by the behavior of private agents. This is, it is the FOC of intial model.

Let’s consider a new-Keynesian model with price adjustment cost

The utility function is

$$U_t = lnC_t - \phi\frac{N_t^{1+\gamma}}{1+\gamma}$$

The recursive equilibrium of the private economy is given by

$$\frac{1}{C_t} = \beta E_t \frac{1}{C_{t+1}}\frac{R_t}{\pi_{t+1}}$$

$$\frac{\pi_t}{C_t}(\pi_t - 1) = \beta E_t \frac{\pi_{t+1}}{C_{t+1}}(\pi_{t+1} - 1) + \epsilon\frac{A_t}{\omega}\frac{N_t}{C_t}\left(\frac{\phi C_t N_t^\gamma}{A_t} - \frac{\epsilon - 1}{\epsilon}\right)$$

$$A_t N_t = C_t + \frac{\omega}{2}(\pi_t - 1)^2$$

$$lnA_t = \rho lnA_{t-1} + \epsilon_t$$

Note that the equation determining the nominal interest rate(monetary policy) is missing.

mod 文件中 var、varexo、model 模块与随机模拟相同：

```
var pai, c, n, r, a;
varexo u;
parameters beta, rho, epsilon, omega, phi, gamma;

beta=0.99;
gamma=3;
omega=17;
```

```
epsilon=8;
phi=1;
rho=0.95;

model;
a = rho*a(−1)+u;
1/c = beta*r/(c(+1)*pai(+1));
pai*(pai−1)/c = beta*pai(+1)*(pai(+1)−1)/c(+1)
+epsilon*phi*^n(gamma+1)/omega
−exp(a)*n*(epsilon −1)/(omega*c);
exp(a)*n = c+(omega/2)*(pai^−1)2;
end;
```

但是，在初值或者稳态值模型，最优政策下的 code 有一些不同。

Computing the steady state under optimal policy may be difficult. Furthermore even a model for which there exists a steady state for constant value of the policy instruments may not have a finite steady state under optimal policy.

关于政策工具的问题：

大家回忆一下 DSGE 建模与编程入门 (31)：最优货币政策（一）。在无时间视角承诺 regime 下，我们可以得到最优政策规则为：

$$\pi_{t+i} = -\frac{\lambda}{\kappa}(x_{t+i} - x_{t+i-1})$$

其中，π 和 x 分别表示 t+i 期的通胀率和产出缺口。其经济学含义参见往期相关推文。

问题来了，很多人可能不理解，我们的政策工具不是名义利率 r_t 吗？可是，上式看起来与 r_t 无关呀。而且，很多书本和论文中，都将通胀率 π 和产出缺口 x 作为央行的工具呢？甚至 Bodenstein et al.（2019，JME）在文章中写道，“理论上来讲，模型中的所有内生变量都可以作为工具。”

从本质上讲，将模型中的所有内生变量（包括通胀率 π 和产出缺口 x）作为央行的工具与将名义利率 i 作为央行的工具是一样的。

我们以三方程 NK 模型为例。

央行的目标函数是下列二次型损失函数：

$$L_t = \frac{1}{2} E_t \sum_{i=0}^{\infty} \beta^i (\pi_{t+i}^2 + \lambda x_{t+i}^2)$$

Dynamic IS curve

$$x_t = E_t x_{t+1} - \frac{1}{\sigma}(r_t - E_t \pi_{t+1} - \hat{r}_t)$$

NKPC:

$$\pi_t = \beta E_t \pi_{t+1} + \kappa x_t + \epsilon_t$$

Note that the monetary policy is missing.

央行在 DIS 和 NKPC 的约束下，选择名义利率 r，通胀率 π 和产出缺口 x 来实现上述损失函数最小化。

那么，我们构造拉格朗日算式：

$$E_t \sum_{i=0}^{\infty} \beta^i \left[\frac{1}{2}(\pi_{t+i}^2 + \lambda x_{t+i}^2) + \theta_{t+i} \left(x_{t+i} - x_{t+i-1} + \frac{1}{\sigma}(r_{t+i} - \pi_{t+i-1} - \hat{r}_{t+i}) \right) + \psi_{t+i}(\pi_{t+i} + \beta E_t \pi_{t+i+1} + \kappa x$$

其中，θ 和 ψ 分别为 DIS 约束和 NKPC 约束的拉格朗日乘数。

首先，我们对名义利率 r 求 FOC：

$$\frac{1}{\sigma} E_t \theta_{t+i} = 0$$

so

$$\theta_t = E_t \theta_{t+i} = 0$$

上式也就意味着，只要对时变的名义利率 r_t 不施加任何约束（即货币政策规则去掉就意味着对名义利率没有施加约束），那么，DIS 曲线则没有对央行施加任何实际的约束。（注意：这一点非常重要，非常重要，非常重要）

换句话说，给定央行对产通胀率 π 和产出缺口 x 的最优选择，上述 DIS 曲线仅仅对于设定名义利率 r_t 来使得产出缺口 x_t 得到最合意（最优）的值起作用。

从数学上来理解，产出缺口 x 是名义利率的一个函数：

$$x_t = f(r_t)$$

那么，央行选择最优的产出缺口 x，实际上就是通过设定最优的名义利率 r 来实现的。同理，央行选择最优的通胀率 π，也需要通过设定最优的名义利率 r 来实现。

这也就是说，央行的最优政策工具 π 和 x 与最优货币政策工具 r 本质上是一样的。

上述问题与最优政策的稳态求解极其相关。有些人疑惑，既然我们在求解最优货币政策时，将货币政策规则去掉了，那么，应该少了一个方程。此时，均衡系统方程数量与内生变量数量不一致，怎么求解？

实际上，道理还是同上。因为原模型中的 N 个内生变量，由于最优政策下，其它 N-1 个内生变量都可以表达成名义利率 i 的函数，所以每个内生变量的稳态值实际也是政策工具的函数。

因此，我们在求解 Ramsey 问题下的稳态时，实际上是在求解内生变量条件依赖于政策工具变量的稳态值。即

$$x = f(r)$$

而内生变量 x 的稳态值实际依赖于 r 的值。所以对于中等规模以上模型，我们最好使用数值方法来求解稳态，且还给定政策工具变量的初值。

Dynare offers three different strategies to compute the steady state under optimal policy:

1. **provide an analytical, recursive solution for the steady state**. In most cases, it is not possible or necessary to provide exact steady state values for the endogenous variables and the Lagrange multipliers. The steady state of Lagrange mutlipliers can be easily derived from the steady state value of the endogenous variables and Dynare always computes them for you.

   Sometimes, it is possible to derive by hand optimal values for the steady state of all endogenous variables. These values or recursive relations permitting to compute them as a function of the parameters can be entered in a *steady_state_model* block.

   More often, it is only possible to derive the steady state values of the other endogenous variables given the value of the policy instruments or variables closely related to them (for example, in a monetary model, the inflation rate instead of the nominal interest rate). In this case as well, the analytical, recursive steady state can be entered in the *steady_state_model* block.

It is then necessary to declare the name of the endogenous variables that are considered as given in the **instruments** option and to provide a guess value for the instruments in the **initval** block.

Using this approach, Dynare has to solve a non–linear problem with only as many unkowns as there are instruments.

2. to provide numerical guess values for all endogenous variables (except Lagrange multipliers) in a **initval** block. In this case, Dynare solves a non-linear problem with as many unknowns as there are endogenous variables. More numerical difficulties can arise than with the first strategy.

3. write your own *filename_steadystate.m* Matlab function. This is only for people with good knowledge of Matlab and Dynare internals. It is rarely necessary given the existence of the first strategy above.

上述例子中的初值和稳态模块为：

```
initval;
r=1;
end;

steady_state_model;
a = 0;
pai = beta*r;
c = find_c(0.96,pai,beta,epsilon,phi,gamma,omega);
n = c+(omega/2)*(pai^-1)2;
end;
```

In the above code, given a guess value for r in initval-block, it is not possible to find c or n, analytically. It is necessary to solve for c (or for n) numerically. We use the following auxiliary Matlab function **find c.m**:

```

function c = find_c(c0,pai,beta,epsilon ,phi,gamma,omega)
c = csolve(@nk_ss,c0,[],1e-8,100,pai,beta,epsilon ,phi,gamma,omega);

function r = nk_ss(c,pai,beta,epsilon ,phi,gamma,omega)
r = pai*(pai-1)/c - beta*pai*(pai-1)/c ...
-epsilon*phi*(c+(omega/2)*(pai^^-1)2)(gamma+1)/omega ...
+(c+(omega/2)*(pai^-1)2)*(epsilon-1)/(omega*c

```

Note that for $\pi = 1$, the steady state can be solved for analytically, but this requires to know the optimal value of inflation in this model. In more complicated models, it may not be the case. Also, still for the sake of example, we use a guess value for r that is away from its optimal value.

The shock-block

```

shocks;
var u; stderr 0.008;
end;

```

In Ramsey policy, Command **planner_objective** declares the policy maker objective function. The syntax:

planner_objective MODEL EXPRESSION;

MODEL EXPRESSION is the one-period objective, not the discounted lifetime objective.The objective function can only contain current endogenous variables and no exogenous ones. This limitation is easily circumvented by defining an appropriate auxiliary variable in the model.

With **ramsey_policy**, you are not limited to quadratic objectives: you can give any arbitrary nonlinear expression.

```

```

```
planner_objective ln(c)-phi*^n(1+gamma)/(1+gamma);

```

Command **ramsey_policy** controls the computation of optimal Ramsey policy under commitment. The syntax:

ramsey_policy [VARIABLE NAME...]; ramsey_policy (OPTION. . .) [VARIABLE NAME.. .];

OPTIONS:

- **planner_discount = DOUBLE** the planner discount factor (β in the model used in introduction)
- **order = INTEGER** the order of approximation used for the set of first order conditions of the problem of the policy maker and of the corresponding solution
- **instruments = ([VARIABLE NAME, . . .)]** name of the policy instruments.

  Naming policy instruments is not necessary to find the solution. It is however useful, when providing a analytical, recursive solution for the steady state under optimal policy. Note that in that case, the term instrument is used somewhat abusively.
- all other options accepted by stoch_simul

This command computes the first order approximation of the policy that maximizes the policy maker' s objective function subject to the constraints provided by the equilibrium path of the private economy and under commitment to this optimal policy. The Ramsey policy is computed by approximating the equilibrium system around the perturbation point where the Lagrange multipliers are at their steady state, i.e. where the Ramsey planner acts as if the initial multipliers had been set to 0 in the distant past, giving them time to converge to their steady state value. Consequently, the optimal decision rules are computed around this steady state of the endogenous variables and the Lagrange multipliers.

This first order approximation to the optimal policy conducted by Dynare is not to be confused with a naive linear quadratic approach to optimal policy that can lead to spurious welfare rankings (see Kim and Kim (2003)). In the latter, the optimal policy would be computed subject to the first order approximated FOCs of the private economy. In contrast, Dynare first computes the FOCs of the Ramsey planner' s problem subject to the nonlinear constraints that are the FOCs of the private economy and only then approximates these FOCs of planner' s problem to first order. Thereby, the second order terms that are required for a second-order correct welfare evaluation are preserved.

```

ramsey_policy(planner_discount=0.99,order=1,instruments=(r));

```

6.4.3 Ramsey Policy under discretion

Command **discretionary_policy** controls the computation of optimal Ramsey policy under discretion. The syntax:

discretionary_policy [VARIABLE NAME...]; discretionary_policy (OPTION. . .) [VARIABLE NAME.. .];

You should ensure that your model is linear and your objective is quadratic. Also, you should set the linear option of the model block.

This command accepts the same options than ramsey_policy, plus:

- **discretionary_tol = NON-NEGATIVE DOUBLE**

  Sets the tolerance level used to assess convergence of the solution algorithm. Default:1e-7.

- **maxit = INTEGER** Maximum number of iterations. Default: 3000.

With **discretionary_policy**, the objective function must be quadratic.

6.5 Welfare Analysis

In many contexts, monetary policy being one, we can't use straight up linearization about a non-stochastic steady state to think about optimal policy.[4] Why is this? In a linearization, expected values of variables are equal to their steady state values, and in many contexts policy doesn't affect the steady state, and hence doesn't affect means. Concretely, in a linearization the expected value of utility (or welfare, taken to mean the present discounted value of flow utility) is independent of policy parameters like what show up in a Taylor rule. Therefore, the natural objective function for a policy-maker (expected welfare) isn't impacted by the policy parameters, and the problem is degenerate.

6.5.1 Policy Regime Comparison

Define welfare as the present discounted value of the flow utility of a representative agent. This can be written recursively as:

$$V_t = U(U_t, N_t) + \beta E_t V_{t+1}$$

Note that this looks like a Bellman equation, but there is no max operator, because I assume that C_t and N_t have been chosen optimally. The objective of a policy maker will be to pick policies to maximize the unconditional expectation of welfare, e.g. $E(V_t)$. In a first order approximation, $E(V_t) = V^\star = \frac{1}{1-\beta}U(C^\star, N^\star)$, which in this case is independent of the stance of monetary policy.

To think about policy, we therefore need to use a higher order approximation to welfare. There are two approaches one can take here.

- In the one described above, you linearize the equilibrium conditions of the model, but take a second order approximation to the recursive representation of welfare.

  Doing that and simplifying yields the quadratic loss function I showed above.

[4]Note that in different contexts where policy choices affect the steady state this is not the case. For example, the level of trend inflation will affect the non-stochastic steady state in the basic New Keynesian model. Similarly, tax instruments will affect the steady state of a model with fiscal policy.

- In the other approach, you can take a second order approximation to all the equilibrium conditions, including the recursive representation of welfare.

  This isn't as neat in the sense that it's not possible to derive simple quadratic loss functions that are clean, but one can numerically calculate the expected values of welfare for different policy specifications and compare those to one another.

The mod code in welfare analysis using higher order approximation is the same as common Dynare mod file, except for the fact that we used a second order approximation and that we included a recursive representation of welfare as an equilibrium condition.

```

var ... V;

...
...
model;
...
% (18) Welfare
V = C - psi*exp(N^)(1+eta)/(1+eta) + beta*V;
...
end;
...
...
stoch_simul(order=2,irf=20,ar=0,nocorr,nograph);

```

Below is the non-stochastic steady state of the model in Figure 6.1:

Below are the moments, including the expected values of the variables in Figure 6.2:

We observe here that the mean / expected value of welfare is lower (-50.5002) than steady state/welfare (-50.268). The reason why average welfare is lower than steady state welfare is because agents don't like volatility because

```
STEADY-STATE RESULTS:

Y       -0.0526803
C       -0.0526803
int     0.010101
infl    0
inflr   0
N       -0.0526803
w       -0.105361
mc      -0.105361
A       0
vp      0
x1      2.11105
x2      2.21641
m       4.55249
dm      0
r       0.010101
Yf      -0.0526793
gap     0
V       -50.268
```

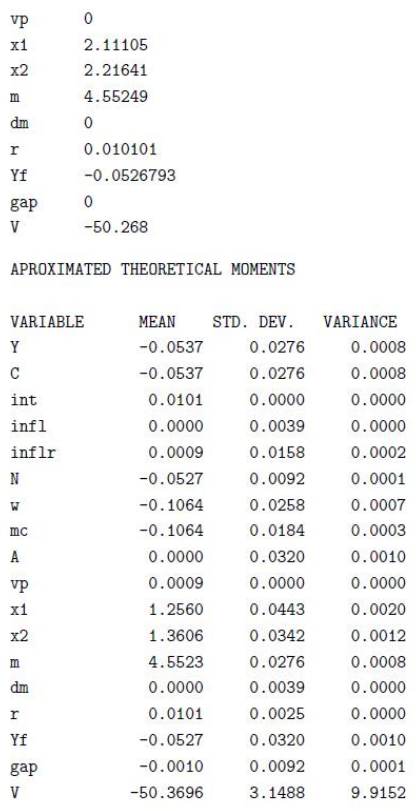
Figure 6.1: Steady State

```
VARIABLE       MEAN    STD. DEV.   VARIANCE
Y            -0.0549     0.0290     0.0008
C            -0.0549     0.0290     0.0008
int           0.0100     0.0000     0.0000
infl          0.0000     0.0060     0.0000
inflr         0.0022     0.0240     0.0006
N            -0.0527     0.0129     0.0002
w            -0.1077     0.0314     0.0010
mc           -0.1077     0.0257     0.0007
A            -0.0000     0.0320     0.0010
vp            0.0022     0.0000     0.0000
x1            1.2631     0.0693     0.0048
x2            1.3666     0.0525     0.0028
m             4.5571     0.0290     0.0008
dm            0.0000     0.0082     0.0001
r             0.0100     0.0039     0.0000
Yf           -0.0527     0.0320     0.0010
gap          -0.0022     0.0129     0.0002
V           -50.5002     3.1501     9.9233
```

Figure 6.2: Moments

of concavity in preferences. In a first order approximation this wouldn't show up, but in the second order approximation it does.

We can then solve the model under an alternative monetary policy rule. For example, suppose that we have a money growth rule, but we set the standard deviations of monetary shocks to 0. This in effect means that the money supply would be constant. Below are the steady state values and expected values in Figure 6.3:



```
vp      0
x1      2.11105
x2      2.21641
m       4.55249
dm      0
r       0.010101
Yf      -0.0526793
gap     0
V       -50.268

APROXIMATED THEORETICAL MOMENTS

VARIABLE       MEAN    STD. DEV.   VARIANCE
Y            -0.0537     0.0276     0.0008
C            -0.0537     0.0276     0.0008
int           0.0101     0.0000     0.0000
infl          0.0000     0.0039     0.0000
inflr         0.0009     0.0158     0.0002
N            -0.0527     0.0092     0.0001
w            -0.1064     0.0258     0.0007
mc           -0.1064     0.0184     0.0003
A             0.0000     0.0320     0.0010
vp            0.0009     0.0000     0.0000
x1            1.2560     0.0443     0.0020
x2            1.3606     0.0342     0.0012
m             4.5523     0.0276     0.0008
dm            0.0000     0.0039     0.0000
r             0.0101     0.0025     0.0000
Yf           -0.0527     0.0320     0.0010
gap          -0.0010     0.0092     0.0001
V           -50.3696     3.1488     9.9152
```

Figure 6.3: Steady State

Getting rid of the shock to the money growth rate doesn't affect the steady state value of welfare, but it does increase the mean value of welfare (-50.3696 vs. -50.5002).

6.5.2 Consumption Equivalent

The units of welfare are not particularly interpretable, so we often want to express the differences in "consumption equivalent" units. The thought experiment is to say "How much consumption would one be willing to give up (each period) under one policy to have the same welfare as under a different policy." It is arbitrary which policy one takes to be the "baseline"— here I'm going to assume it's the policy with no monetary shock.

Let λ denote the fraction of consumption one would be willing to give up in each period in one economy, call this economy 0. With log utility over consumption, we have:

$$E(V_t^0(\lambda)) = E\left[ln(C_t(1+\lambda)) - \theta\frac{N_t^{1+\xi}}{1+\xi} + \beta E_t V_{t+1}^0(\lambda)\right]$$

Since the λ shows up every period and is deterministic, this reduces to:

$$E(V_t^0(\lambda)) = \frac{1}{1-\beta}ln(1+\lambda) + E\left[ln(C_t) - \theta\frac{N_t^{1+\xi}}{1+\xi} + \beta E_t V_{t+1}^0(\lambda)\right]$$

The term in brackets is just:

$$E(V_t^0(\lambda)) = \frac{1}{1-\beta}ln(1+\lambda) + E(V_t^0)$$

Then we want to find the λ that equates this with expected welfare in the alternative economy, call it economy 1. We have:

$$E(V_t^0(\lambda)) = E(V_t^1)$$

or

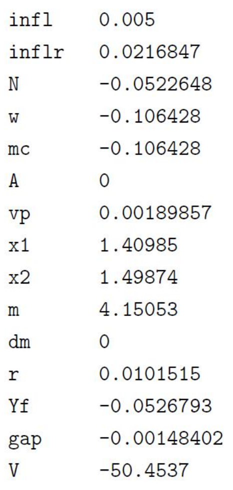
$$\frac{1}{1-\beta}ln(1+\lambda) + E(V_t^0) = E(V_t^1)$$

Solving for λ:

$$\lambda = e^{(1-\beta)(E(V_t^1)-E(V_t^0))} - 1$$

If I take the baseline economy 0 to have higher welfare, then the term inside the exp is negative, which means $\lambda < 0$. This makes sense you would be willing to give up consumption in the high welfare economy to have the same welfare as another economy governed by a policy resulting in lower welfare. Using the differences in expected welfare for the two economies in question here, and a value of $\beta = 0.99$, I get a value of $\lambda = -0.0013$. This means that an agent would be willing to forfeit about 0.1 percent of consumption each period in the economy with no monetary shocks to avoid going to the economy with monetary shocks. This number may not seem large but we typically find pretty small welfare differences under different macro policies, so it's not abnormally low.

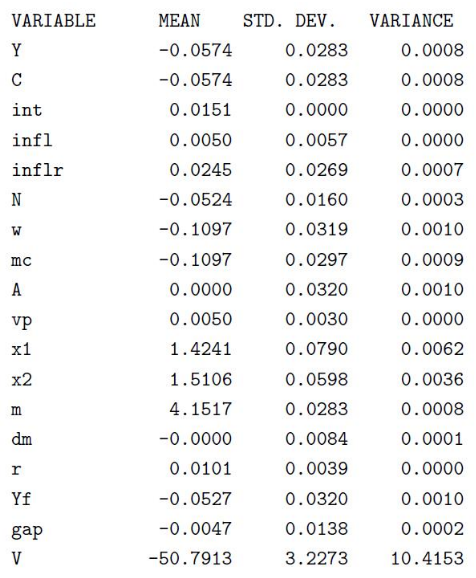
I solved the model assuming zero trend inflation. Suppose I solve the model where steady state inflation is instead set to 0.005 (approximately 2 percent at an annualized frequency). Below are the steady states and the means in Figure 6.4 and 6.5:



```
infl     0.005
inflr    0.0216847
N        -0.0522648
w        -0.106428
mc       -0.106428
A        0
vp       0.00189857
x1       1.40985
x2       1.49874
m        4.15053
dm       0
r        0.0101515
Yf       -0.0526793
gap      -0.00148402
V        -50.4537
```

Figure 6.4: SS

Unlike the standard deviation of the monetary policy shock, the level of trend inflation does affect the steady state we can see that steady state welfare is lower with higher trend inflation (-50.4537 vs. -50.268). We also see that expected welfare is lower, at -50.7913 (compared to -50.5002). We could calculate consumption equivalent welfare differences based both on steady state welfare or expected welfare. Based on steady states, we would get $\lambda = -0.0019$; based on the means, we would get $\lambda = -0.0029$.



```
VARIABLE        MEAN    STD. DEV.   VARIANCE
Y             -0.0574     0.0283     0.0008
C             -0.0574     0.0283     0.0008
int            0.0151     0.0000     0.0000
infl           0.0050     0.0057     0.0000
inflr          0.0245     0.0269     0.0007
N             -0.0524     0.0160     0.0003
w             -0.1097     0.0319     0.0010
mc            -0.1097     0.0297     0.0009
A              0.0000     0.0320     0.0010
vp             0.0050     0.0030     0.0000
x1             1.4241     0.0790     0.0062
x2             1.5106     0.0598     0.0036
m              4.1517     0.0283     0.0008
dm            -0.0000     0.0084     0.0001
r              0.0101     0.0039     0.0000
Yf            -0.0527     0.0320     0.0010
gap           -0.0047     0.0138     0.0002
V            -50.7913     3.2273    10.4153
```

Figure 6.5: Moments

Hence, we observe that the welfare loss based on means is higher than the welfare loss based on the steady state. This is because trend inflation does two things: first, it distorts the steady state by effectively increasing the steady state markup and increasing steady state price dispersion. But it also interacts with the shocks to have a bigger effect on means: with positive trend inflation, price dispersion is first order, which makes stochastic shocks more costly.

Chapter 7

Troubleshooting

Wenli Xu
Anhui University and CIMERS
xuweny87@hotmail.com

To make sure this section is as user friendly as possible, the best is to compile what users have to say! Please let me know what your most common problem is with Dynare, how Dynare tells you about it and how you solve it. Thanks for your precious help!Main sources: Dynare Forum(`forum.dynare.org`).

7.1 Model Comparison

1. same data for models, same priors for models, but additional parameters for one, not for another. Then does that bias the posterior odds ratio/model comparison?
   Reply: For bayesian model comparison, models don't need to be nested and there is a natural degrees of freedom correction. Hence, as long as you use the same data, having different parameters does not at all.

2. Based on the formula of posterior odds ratio, since same data and prior, it is enough to compare marginal densities?
   Reply: That is sufficient. The prior over parameters does not matter,

but the prior odds ratio over the models(see Koop(2003):Bayesian Econometrics). If you a prior assign equal probability to all models(0.5 for two models), you simply compare the marginal data densities.

3. From the estimation results:"Log data density" was compared using Modified Harmonic Mean and "Log data density [Laplace approximation]" via Laplace, right?
   Reply: Both the Laplace approximation and the modified harmonic mean estimation are ways to compute the marginal data density. As computing the marginal data density involves solving a complicated integral, these two methods for trackling the issue have been proposed. In theory, they should yield identical results as they measure the same thing. In practice, both involve approximations and may yield differing results.
   It is hard to tell which one to prefer. Footnote of SW(2007) state:" As discussed in John Greweke(1998), the MH-based sample of the posterior distribution can be used to evaluate the marginal likelihood of the model. Following Geweke(1998), we calculate the modified harmonic mean to evaluate the integral over the posterior sample. An alternative approximation is the Laplace approximation around the posterior mode, which is based on a normal distribution. In our experience, the results of both approximations are very close in the case of our estimated DSGE model. This is not too surprising, given the generally close correspondence between the histograms of the posterior sample and the normal distribution around the estimated mode for the individual parameters. Given the large advantage of the Laplace approximation in terms of computational costs, we will use this approximation for comparing alternative model specifications in the next section."

4. The decision rule for model comparison using marginal densities is: higher = better? e.g. model A(-950), model B(-1000), then choose model A.
   Reply: The marginal data density to be as high as possible. The logarithm is monotonic transformation, so we also want the log marginal densities to be as high as possible.

5. Do one need to make any transformation on these values first?
   Reply: It is often easier to compare models using posterior mode probabilities, see Koop(2003,p4).

6. Additional shocks matter? Reply: No matter. What matters is that you estimate those shocks' standard deviation. But this just another parameter. Note that: you do not need the same prior.

7. In the model_comparison command,

   - Bayes_Ratio refers to the ratio of Marginal Data Densities often called Bayes Factor[1].

[1]Bayes Factor,Wikipedia,
In statistics, the use of Bayes factors is a Bayesian alternative to classical hypothesis testing. Bayesian model comparison is a method of model selection based on Bayes factors. The aim of the Bayes factor is to quantify the support for a model over another, regardless of whether these models are correct.
Definition
The Bayes factor is a ratio of the likelihood probability of two competing hypothesis, usually a null and an alternative.
The posterior probability $Pr(M|D)$ of a model M given data D is given by Bayesi theorem:

$$Pr(M|D) = \frac{Pr(D|M)Pr(M)}{Pr(D)}$$

The key data-dependent term $Pr(D|M)$ is a likelihood, and represents the probability that some data are produced under the assumption of this model, M; evaluating it correctly is the key to Bayesian model comparison.
Given a model selection problem in which we have to choose between two models on the basis of observed data D, the probability of the two different models M_1 and M_2, parameterised by model parameter vectors θ_1 and θ_2 is assessed by the Bayes factor K given by

$$K = \frac{Pr(D|M_1)}{Pr(D|M_2)} = \frac{\int Pr(\theta_1|M_1)Pr(D|\theta_1, M_1)d\theta_1}{\int Pr(\theta_2|M_2)Pr(D|\theta_2, M_2)d\theta_2} = \frac{Pr(M_1|D_1)}{Pr(M_2|D_2)}\frac{Pr(M_2)}{Pr(M_1)}$$

When the two models are equally probable a priori, so that $Pr(M_1) = Pr(M_2)$, the Bayes factor is equal to the ratio of posterior probabilities of M_1 and M_2. If instend of the Bayes factor integral, the likelihood corresponding to the maximum likelihood estimate of the parameter for each model is used, then the test becomes a classical likelihood-ratio test. Unlike a likelihood-ratio test, this Bayesian model comparison does not depend on any single set of that is automatically, and quite naturally, includes a penalty for including too much model structure. It thus guards against overfitting. For models where an explicit version of the likelihood is not available or too costly to evaluate numerically, approximate Bayesian computation can be used for model selection in a other approaches are:

- to treat comparison as a decision problem, computing the expected value or cost of each model choice;
- to use minimum message length(MML).

- Posterior_Model_Probability gives you the posterior probability of the models. It is not identical to the Posterior Odds Ratio, because the Odds Ratio gives you the relative instead of the absolute probabilities. But given that the probability must sum to 1, you can compute them.
  For example, if you compare two models and the Model 1 has a Posterior_Model_Probability of 2/3, then the other model has to have one of 1/3 and Posterior Odds Ratio is (2/3)/(1/3)=2:1.
- Marginal Data Density(MDD) measure fit relative to other models on the same data. Note that it is pretty meaningless on its own.
- The acceptance rate only tells you about efficiency of the MCMC. Note that it does not tell you about convergence.
- The Geweke diagnostics for convergence.

7.2 Parameter Estimation

1. Whether it is possible to perform the parameter estimation using a very small sample with annual data?
   Reply: That is not ideal, but doable. The prior distribution will most probably play an important role in the case, i.e. the data will not be very informative.
2. The model is stochastically singular.
   Reply: Either that the number of observation variables is greater than the number of shocks, or observation imply an exact linear combination(more likely). Either that or measurement errors.
3. Based on bayesian method, we have the formula:
   a) log(posterior density)=log(likelihood)+log(prior density)
   b) combining with Dynare, then log-post(blue line) is log(posterior density) and log-lik kernel(green line) is log(likelihood).
   c) right?

   Reply: Right

4. (1)If the mode_check plot show that mode of log-post and log-lik kernel are identical, then we can consider this mode as the mean of proposal density for the MH algorithm. Otherwise, we have to find the mode again as long as mode of log-post and log-lik kernel are identical. Right? (2)If we can not find the truly global mode, then the MH algorithm can not converge. If such, how to find the truly global mode?
Reply: From the Bayes theorem, we know that the posterior density is equal to the likelihood times the prior density divided by the marginal density of the data:
$$p(\theta|y_T) = \frac{p(y_T|\theta)p(\theta)}{p(y_T)}$$
The numerator is posterior kernel (it's not a density, because it does not integrate to 1). If you are only concerned by the inference about the parameters(θ), you do not need to worry the denominator, all the information about the parameters is embodied in the posterior kernel. The mode_check option will return plots of the log of the posterior kernel and of the log likelihood as a function of one parameter, keeping the other parameters constant (equal to the estimated posterior mode). So if you has two parameters θ_1 and θ_2, and if your estimation of posterior mode is $(\hat{\theta}_1, \hat{\theta}_2)$, you will have the plots for:
$$f(\theta_1) == logp(y_T|\theta_1, \hat{\theta}_2) + logp(\theta_1, \hat{\theta}_2)$$
and
$$g(\theta_1) == logp(y_T|\theta_1, \hat{\theta}_2)$$
with θ_1 taking values around $\hat{\theta}_1$. and
$$f(\theta_2) == logp(y_T|\hat{\theta}_1, \theta_2) + logp(\hat{\theta}_1, \theta_2)$$
and
$$g(\theta_2) == logp(y_T|\hat{\theta}_1, \theta_2)$$
with θ_2 taking values around $\hat{\theta}_2$.
So these curves are not the full multi-dimensional log likelihood and posterior kernel, but only two-dimensional cuts through these objects. Note also that because the prior density is the difference between the two

objects, the log likelihood is typically smaller equal than posterior kernel (if the log prior density is positive), i.e. the graph of the likelihood would be below the posterior density and potentially not in the picture. For that reason, the likelihood is shifted upwards to have the same maximum as the posterior kernel in order to better compare them.

There is absolutely no reason to seek for an estimation where these cuts through the objects are identical. Indeed, if the prior brings some information to the inference about parameters, the curves have to be different, and in general the mode of the posterior kernel (or density) does not match the mode of the likelihood.

The last statement about the importance of finding the posterior mode is wrong. The MCMC will converge even if it starts from an initial state different from the (global) posterior mode, as long as for the initial state the posterior density is strickly positive, the jumping proposal density covariance is positive definite, and that all the usual assumptions on the posterior density are sutisfied. You will typically need a lot more iterations if the initial state is in a low posterior density region, but the MCMC will eventually converge to the posterior distribution. Actually, as long as mh_nblocks is greater than one (the default being two). Dynare does not start the MCMC from the estimated posterior mode. Rather, the initial stae of each chain is chosen randomly to be overdispersed around the estimated posterior mode (the distance to the estimated posterior mode is controlled by option mh_init_scale).

The form of the proposal (or jumping) distribution matters more than the initial state of the chain(s) for determining the number of iterations needs to ensure convergence. It is essentially in this respect that the estimation of the posterior mode is important, because we use the inverse Hessian at the estimated mode as an approximation of the posterior covariance matrix.

5. In An and Schorfheide(2007) "Bayesian Analysis of DSGE model", for random-walk metropolis algorithm[2], they chose two jump parameter-

[2]As the name suggests, the random walk MH algorithm specifies the candidate generating density as a random walk:

$$\Phi^{G+1} = \Phi^{G} + e$$

s:mh_init_scale=1 and mh_jscale=0.3 (the rejection rate is about 45%). So is there any general rule to choose mh_init_scale and mh_jscale? In Dynare, the default value for mh_init_scale=2*mh_jscale, this jump factor should be adjusted, rihgt?
Reply:

- mh_jscale should be adjusted to give an acceptance rate of 23%. For a multivariate normal posterior, this would be the most efficient choice to get quick convergence. For mh_init_scale, there is less good guidance. It should be bigger than mh_jscale to be overdispersed, but typically not too big, 2*mh_jscale is usually a good compromise.
- The target 23% acceptance rate is at best a rule of thumb. To our knowledge, there is no proof that this would be optimal for a particular DSGE model. So We do not think that we should be obsessed by this target.
- The choice for the parameter mh_init_scale does not affect the convergence of the MCMC. It is here to ensure that the initial conditions of the chains(if mh_b=nbloks>1) are different enough. This parameter only affects the convergence diagnostics.
- It works quite well in practice. Anything in the range of 20 to 30 percent should be ok!

6. How to adjust the jump scale to have a "reasonable" acceptance ratio of between 0.2 to 0.4 suggested by the literature?
Reply: The tuning the scale parameter so that the acceptance ratio is between 0.2 to 0.4 is only a heuristic.From very simple models, we know

where Σ the variance of e_t is set by the researcher. $\Sigma = \hat{\Omega} \times \lambda$, where $\hat{\Omega}$ is the estimated variance, λ is the scaling factor. A higher value for Σ could mean a lower rate of acceptances across the MH iterations (i.e. the acceptance rate is defined as the number of accepted MH draws divided by the total number of MH draws) but would mean that the algorithm explores a larger parameter space. In contrast, a lower value for Σ would mean a larger acceptance rate with the algorithm considering a smaller number of possible parameter values. The general recommendation is to choose Σ such that the acceptance rate is between 20% to 40%. We consider the choice of Σ in detail in the examples described below.
See Chib and Ramamurthy (2010), Blake and Mumtaz(2017) for a more efficient version of the basic algorithm described above.

that the acceptance rate should depend on dimension of the problem (the number of estimated parameters), and should be in this region. You can find an introduction on this in the book by Robert and Casella "Monte Carlo Statistical Methods"(Chapter 7, in particular section 7.8.4).
We only know for sure that acceptance ratios close ro 0 or 1 are bad. I usually target one third. But we cannot be sure, controlling only one parameter, that the acceptance rates will be the same across chains. Normally in the end, if the chains are long enough the acceptance ratios should be similar across chains. A small acceptance rate does not mean that the MCMC is trapped in a low density region. Imagine that the current state of the MCMC rate is the posterior mode. If the jumps provided by the proposal distribution are large it is very likely that all the proposals will be rejected (resulting in a low acceptance rate).

7. (1)Dynare compute the RW Metropolis-Hasting acceptance ratio as R =number of accepted draws/number of proposals. Right?(2)Normally, we prefer that the acceptance ratio should not be very close to 0 or 1, an ideal ratio is around 1/3. To reach the ideal ratio, we should adjust the jump scale (in Dynare, we adjust by using mh_jscale). Based on that, one set my jump scale with 0.3. Moreover, one set number of the MH chain with 10 and number of iteration with 200,000. So then lauching Dynare, I found that: for the first MH chain, Dynare report the RW Metropolis-Hasting(1/10) Current acceptance ratio of around 0.256. However, the RW MH(2/10) Current acceptance ratio reduces a lot to 0.129 in the second MH Chain. The acceptance ratio of 0.129 is very far from the ideal ratio of 1/3. Even though I do not change the jump scale. The jump scale is still the same as 0.3 in the first MH Chain. Why the acceptance ratio is not identical for all MH Chain with the same jump scale?
Reply:
   - The understanding of the acceptance ratio is correct.
   - This is simply impossible. You cannot tune a single parameter so that all the chains have the same acceptance ratios. The fact that the acceptance rate is significantly lower in one of the chains is

a matter of chance(the initial condition of this chain may be far from the initial conditions of the other chains, and the proposals are obviously different).
Frankly I would not bother about this. As long as all the acceptance ratios are strickly positive, there is no trouble.

- I trend to somewhat disagree with the above answer. If your acceptance rates very different, it suggests that you are not sampling from the same distribution, i.e. at least one chain does not sample from the ergodic distribution we are interested in. As the above, the difference may just be due to the initial convergence to the ergodic distribution. But this then suggests that the burnin is very large relative to the actual draw, i.e. your chain are too short. For that reason, I would be extremely careful in checking convergence (and the initial mode).
  Also, you might want to consider fewer, but longer chains in this case.
- This is most probably the problem, 200,000 is not enough. At this stage, i.e. as long as the chains did not converge to the (common) ergodic distribution, the total number of draw of 200,000 is not very pertinent, only the number of iteration per chain matters. As the above suggested that reduce the number of chains (I never use more than 5), or run the estimation with the parallel capabilities of Dynare.

8. A common problem that plogues the estimation of DSGE models is the computation of the inverse of the Hession at the posterior mode, that is not positive definite. I am aware that the option mode_compute=5 offers a different way of computing the Hessian, optim={'Hessian',2}. This seems to work well, from what I have seen. Do other optimisers also offer similar options?
   Reply:

   - I believe the answer is no. However, mode_compute=6, which is based on MCMC, provides an estimate of the posterior covariance matrix not based on the inverse of the Hessian matrix(we use M-CMC draws, so it works as long as the acceptance ratio is not too close to 0). Also, it is not mandatory to estimate the posterior mode(with Hessian at the estimated mode) for running a metropolis. You can optionally use another covariance matrix for the jumping distribution.

- mode_compute =5 with optim={'Hessian',2} relies on the outer product of gradients and requires the use of a univariate Kalman fiter. Therefore, something similar is not avaiable for other optimizers. As the above indicated, mode_compute=6 also differs.
  The reason for using the inverse Hessian at the mode is that this is the most efficient choice if the posterior would be normal. But any positive definite matrix for the proposal density will do. You could provide any arbitrary matrix via command.

- We use the draws of the Metropolis (run in mode_compute=6) to estimate the posterior covariance matrix. We need to have enough variability in these draws.If all the draws were identical (or more generally if the number of parameters) the sample covariance matrix would obviously not be full rank. That's why we target, by default, an acceptance rate of 1/3 (which is the value commonly considered in the literature.
  The optimization routine has several steps:

  1 we run a Metropolis, where we tune the scale parameter so that the acceptance rate is close to the target. In this first step the draws are not used to estimate the covariance matrix, we only continuously keep record of the vector of parameters which gives the best value for the objective. In this first step the covariance matrix if the jumping distribution is an identity matrix.

  2 we run a Metropolis to estimate the covariance matrix. Again we continuously keep record of the vector of parameters which gives the best value for the objective.

  3 We return to 1, with the new estimate of the posterior covariance

matrix.
By default, we iterate on these steps two or three times(I do not remeber, look at the reference manual). And in the last step we run a last metropolis, where we only update our estimate of the posterior mode. In this last round we decrease slowly the size of the jumps.

9. Why the Hessian found by the numerical optimisers is not positive definite?
Reply: In Dynare we do not use the Hessian matrices returned by the optimizers. For instance, mode_compute=4, the default algorithm derived from Chris Sims code, returns a crude estimate of the Hessian that is not used by Dynare (see any presentation of the BFGS, there is nice page on wikipedia, algorithm which is close to what is done here). Instead we compute the Hessian with finite difference (expect mode_compute=6 and mode_compute=5 which uses a gradient the outer product approach), by calling hessian.m function.
The main culprit is that the optimization routine failed in finding a (local) minimum of minus the likelihood (or posterior kernel). That's why you need to play with other optimization routine and/or the initial guesses. Another culprit, may be the noise in the objective function. In this case you have to change the length of the steps in the finite difference routine (controlled by options_.gstep).
Two additional issue are:

   - the Hessian only needs to be positive definite at an interior solution. If you have a corner solution, i.e. you are at the bound of your prior parameter space, there will be a problem.
   - if a parameter is not identified or if there is collinearity in the jacobian of the likelihood, the Hessian will also be non-positive definite.
     That is why a look at the mode_check plots is often revealing to see many pathological issues simply due to the finite difference approximation to the Hessian.

10. some parameters are at the prior bounds.
    Reply: some potential solution are:
    - Check your model for mistakes;
    - Check whether model and data are consistent (correct observation equations);
    - Shut off prior_trunc;
    - Change the optimization bounds;
    - Use a different mode_compute like 6 or 9;
    - Check whether the parameters estimated are identified;
    - Check prior shape (e.g. Inf density at bounds);
    - Increase the informativeness of the prior.

11. Log Data Density[Laplace approximation] is NaN. Error using chol: Matrix must be positive definite.
    Reply:
    - First of all, take a look at the mode_check plots if there is anything strange (in particular if there are any horizontal lines indicating non-identifiability);
    - Second, wrong observation equations;
    - It may also be that the computed posterior mode is not a true local maximum; in that case you may want to try other initial values, or another optimizer.

12. In historical and smoothed variables plot, two lines are different, if no measurement error?
    Reply:
    - It is up to you to decide if your model has measurement errors. If you decide to add measurement errors, then you will observe gaps between historical data and smoothed variables. The smaller are these differences, the better is the fit. Obviously this comparison does not make sense if you do not have measurement errors. It

may happen that, even without measurement errors, you observe differences between historical and smoothed variables (typically different means). This may reflect a problem in the specification of your measurement equations (missing constant) or a bug in Dynare.

- You can also get a gap between two lines in case of stochastic singularity. This could particularly occur before Dynare 4.5 as the use_univariate_filters_if_singularity_is_detected option was enabled by default.

13. After estimation, the absolute value of standard deviation of observation are quite high which are much higher than the data standard deviation. However, the relative standard deviation are similar. I take log-linearization of the model by hand, so every variable in the code represents percentage deviation from steady state(which are all 0). The measurement equation for demeaned growth rate data:

$$y\_obs = y - y(-1) + y\_ME$$

$$c\_obs = c - c(-1) + y\_ME$$

$$i\_obs = i - i(-1)$$

$$h\_obs = h$$

The measurement equation for one-side HP filtered data(cyclical component):

$$y\_obs = y + y\_ME$$

$$c\_obs = c$$

$$i\_obs = i$$

$$h\_obs = h$$

both model second moments are much larger than corresponding data second0 moments.

Reply: Unfortunately, this is not entirely unheard of. In Born, Peter and Pfeifer(2013) ”Fiscal news and macroeconomic volatility”, they use the endogenous_prior option for this reason. You should try whether this helps. If yes, the actual estimation might be fine.

14. After using the option, the problem has been solved. Could you explain a little bit more about endogenous_prior, what is the mechanism to decrease the second moments and is there any disadvantage using this option? I find that after using this option, the mode_check plots are a bit different, especially for the persistence of technology shock and preference shock.
Reply: See the manual 4.5.6(page78) on the option and the reference therein. The Christiano, Traband and Walentin's endogenous prior is based on "Bayesian learning". It updates a given prior using the first two sample moments.

15. Signs of the marginal likelihood(log data density)
Reply: There is no reason why the marginal likelihood should be less than one. The log data density may be positive. The reason is that the marginal likelihood may be larger than 1 so that its log is positive(Think of a normal distribution with infinitesimally small variance. In this case the density around the mean will go to infinity. Only for disrete distributions is the marginal density always negative).

16. When I start to find mode, set mh_replc=0, sometimes no matter I use mode_compute=9, the result always show (minus) the Hessian matrix at the "mode" is not positive definite. In this case ,I will run mode_compute=6 to get a "nice" mode check plot. Then I use the mode file to run mode_compute=9 again, if Hessian matrix is positive definite, then I run MCMC, like 200,000 draws with mode_compute=9. If Hessian matrix is not positive definite, I just run MCMC with mode_compute=6 to get final results.(9 is better than 6 to run MCMC if Hessian matrix is positive definite) Are these steps reasonable? or could you give me some advices? By the way, is mode_compute=9 always better than 4 since the former one is globally finding mode, while the later one is locally?
Reply:

- The issue in the end always is to find the global mode. What you describe is a iterative procedure to find it and it sounds reasonable. However, you should keep track of whether there is actual improvement in the posterior density across runs(and particularly the MCMC). If you are unsure where the mode is and you have rather diffuse priors that do not smooth out the likelihood by much, mode_compute=4 is not advisable, because it tends to get stuck at local modes.

- Sequentially running different mode_compute makes a lot of sense. In case you did not find the true mode, you increase the likelihood of finding it. In case you already found it, running mode_compute a second time will just cost your time, but will not do any other harm as the optimizer will stay at the same mode you found before. Using the calibrated value as starting values is recommented in case your mode_compute fails because of invalid starting values.

17. initial_estimation_check
    Reply: There have been stochastic singularity, i.e. the model implies a perfect linear combination of observables. In this case, having as many shocks as observables is not sufficient. Typical examples are observing all components of the budget constraint. You need to find out where comes from. Start by dropping one observables at a time.

18. Error using chol
    Reply: Prior is two tight. Change the prior to something more sensible or at least use prior_trunc=0.

7.3 Simulation

1. One of the eigenvalue is close to 0/0.
   Reply: A generalized eigenvalue of 0/0 is related to Blanchard and Kahn condition, because it means that any complex number can be a generalized eigenvalues. So we don't have a unique solution in this case (the uniqueness problem here is different from the one treated by Blanchard and Kahn condition which assume that the eigenvalues are uniquely determined and discuss the uniqueness and solutions of the

paths for endogenous variables). Obviously, in practice, we need to decide what is zero. By default, Dynare issue the 0/0 error message if the denominator and numerator are less than 1e-6. Adding an option for the threshold level (called qz_zero_threshold). The option can be used in check, stoch_simul and estimation commands. An example on the gist.github.com/stepan_a.

2. Collinear equations
   Reply: Unless there are unit roots involved in your model, there should not be collinearity in the first place. It would imply that some variables cannot be determined in your model.

3. Oscillating or Alternating IRFs
   Reply:
   - It means that there is a complex eigenvalue leading to oscillations. Usually, when people have this problem in the forum, it comes from them ignoring the Dynare timing convention and then torturing the timing until the BK conditions are satisfied. When doing so, the correct economic timing is ignored and oscillating IRFs results.
   - Most of time, this is timing problem. Wrong timing in feedback equations(wrong equations)
   - Something maybe non-convergence.
   - There is anything usually in the parametrization of policy rules, i.e. a fiscal rule with too much debt feedback.(very strange parameter value)

4. What does it mean the imaginary part of the eigenvalue is not equal to zero? Is that a problem? 5 looking-forward variables, 4 eigenvalues larger than 1, then changing the timing, 4 looking-forward variables, 4 eigenvalues larger than 1, BK condition is verified. Appropriate?
   Reply:
   - It is generalized eigenvalues. This is not a problem, unless you get oscillating IRFs.

- No, this is not appropriate. There is one unique correct timing and you cannot arbitarily shift timing.

5. MS-DSGE
   Reply: Dynare does not support Markov-Switching DSGE models yet. Dynare only supports perturbation, thus requiring that all functions are sufficiently smooth. It would not work with discrete Markov Chains.
   Confusing:

   - Markov process for an exogenous variable: essentially a discretized version a continuous distribution used as a numerical trick—Markove process for shock.
   - True Markov Switching process (Junior Maih's RISE toolbox(https://github.com/stepan-a/rise))

7.4 Steady State

7.5 General Issues

1. The Jacobian contains Inf. or NaN.
   Reply:

   - Just remove the check command before the model_diagnostics command and you will obtain more information.
   - Mathematically, that is not a problem, but numerically, the computer does not know how to handle the limit where you have a variable with $(s^0)^\infty$.
   - If really need to consider the limit case, you will have to do it analytically and change the equations accordingly(i.e. change a CES like production function for a CD form). Unfortunately, Dynare cannot do this for you.

2. Positive Monetary policy shock means that Nominal Rate decrease, and expansive policy?
   Reply: In standard NK models, nominal interest rate may change in

either way.
When Positive monetary policy shock with nominal interest rate falling，The mechanism is the following: Assume the initial monetary policy shock increases the nominal and thus the real interest rate. The central bank then reacts to lower output and inflation endogenously by lowering the nominal interest rate. If the response is strong enough, it overcompensates the initial increase due to the shock. The only thing you know is that the real interest rate increases, so that the shock is still contractionary, although the nominal interest rate goes down. This is one of the reasons why you cannot look at interest rates to determine whether monetary policy is expansionary.
Due to general equilibrium effects, the nominal interest rate can go any way. You only know that the real interest rate must increase after a positive shock to the policy rate:

$$r_real = r - pi(+1);$$

That is actually the case. The “erratic” behavior is actually not that erratic, it simply shows there is not much persistence. If you increase the persistence of the shocks and the interest smoothing, things look more “normal”. What stays is that reactions in the first period are quite differently from the rest.
In Gali’s textbook, the sign of the response depends on various parameters.

3. The BK condition and the rank condition
   Reply: TBC

4. How to solve a model with heterigenous agents?
   Reply:

   - Individual’s state variable includes their wealth, like asset or capital holdings. And every agent faces a probability of dying. So their assets evolve over time. And their decision rules depend on assets, maybe in a nonlinear way.
   - Taking a look at the JEDC special issue(2010,34(1)):”Computational Suite of Models with Heterogenous Agents: Incomplete Markets

and Aggregate Uncertainty". The Kim, Kollman and Kim approach is easily implementable in Dynare.

5. How is heterogeneity defined?
   Reply: It is about whether aggregation is possible or not. If you cannot easily aggregate up, you need to keep track of the distribution(typically the wealth distribution). Distribution are infinite dimensional objects (you need all moments from 1 to infinity to fully characterize a distribution). Dealing with this infinite dimensional object is what makes it so hard.
   In many RA models, there is a continuum of agents, but their consumption and thus wealth is perfectly insured via Arrow-Debreu securities. In this case, every agent will always have the same wealth at the end of the period, even if there are idiosyncratic shocks that only hit some agents. Take away complete markets and some agents will have positive shocks while other will have negative ones. In this case, they will choose different assets at the end of the period. You would need to keep track of this. If you only assume two types, this is doable. If there are infinitely many, it quickly becomes intractable.

6. Sate-dependent IRFs/GIRFs. Suppose one wants to investigate the effect of a government spending shock: the size of fiscal multiplier should vary, if the economy is in a state where output is below or higher than its steady state value.
   Reply:

   - In linear model, IRFs are state independent.
   - For higher order approximation, Dynare generates a generalized IRFs that is state dependent. However, the GIRFs is computed at the ergodic mean. i.e. `https://site.google.com/site/pfeiferecon/RBC_state_dependent_GIRF.mod?attredirects=0`.
   - In RBC model, only the state variables matter for the dynamics, a control variable is a function of the state variables. If one set states to a particular value, controls will be consistent with this value. Only states matter.

Chapter 8

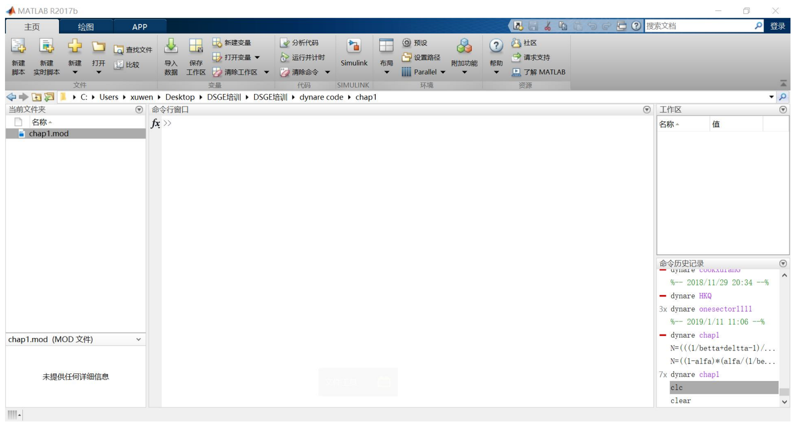
A Matlab Primer

Matlab 是 Matrix Laboratory 的缩写。它是一款非常流行，且功能强大的编程语言。在 Matlab 环境下，我们可以进行矩阵运算，函数和数据图形，算法运行，用户界面制作，以及提供其它编程语言编写的程序接口。Matlab 是一款公认的功能十分强大的数学与工程软件。但是，遗憾的是，它是一款收费软件，而且价格不菲（即使这样，也难不倒千千万万聪明的中国人）。

Matlab 功能虽然十分强大，但是它也有擅长和不擅长的事情：（1）它的优势在于矩阵、矩阵运算和一些线性代数运算；（2）除第一条外，其它都被擅长（当然，这是与其它编程语言相比）。学习 Matlab 的目的就是要充分发挥它的优势，且尽可能避免其劣势。

8.1 基本运算

成功安装 Matlab 之后，双击 Matlab 图标，我们会看到下列窗口：



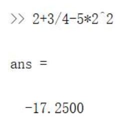
```
C: ▸ Users ▸ xuwen ▸ Desktop ▸ DSGE培训 ▸ DSGE培训 ▸ dynare code ▸ chap1
```

```
chap1.mod
chap1.mod (MOD 文件)
未提供任何详细信息
```

```
  dynare cookxuranG
  %-- 2018/11/29 20:34 --%
  dynare HKQ
3x dynare onesector1111
  %-- 2019/1/11 11:06 --%
  dynare chap1
  N=(((1/betta+deltta-1)/...
  N=((1-alfa)*(alfa/(1/be...
7x dynare chap1
  clc
  clear
```

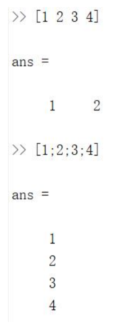
在命令行窗口，我们可以看到光标“»”，这意味着 matlab 准备好了，我们可以在此处输入指令。在光标后，我们可以输入：加“+”、减“-”、乘“*”、除“/”、指数“” 和小括号”()” 等等我们常用的数学和计算符号，函数等命令。

例如：



```
>> 2+3/4-5*2^2

ans =

  -17.2500
```

在 matlab 中，用“[]”来创建一个向量。每个元素之间用空格或者逗号“,”来隔开。每一行用分号来区分。例如



```
>> [1 2 3 4]

ans =

     1     2     3     4

>> [1;2;3;4]

ans =

     1
     2
     3
     4
```

注意：（1）ans 是默认的结果变量。我们也可以自定义结果变量。（2）Matlab 中遵循四则运算法则。

8.2 矩阵运算

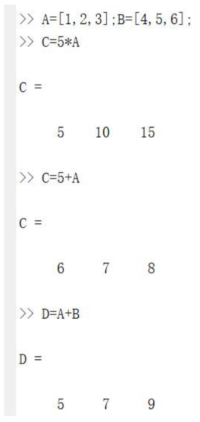
矩阵或数组是 Matlab 运算的基本元素。一个 1*1 矩阵就是一个标量或者单个数，而只有一行或一列的矩阵就是行向量或者列向量。例如下列两个矩阵的基本运算：

Matlab 还可以执行矩阵的内积、点乘、点除和幂运算。唯一的要求是向量的长度必须相同。例如

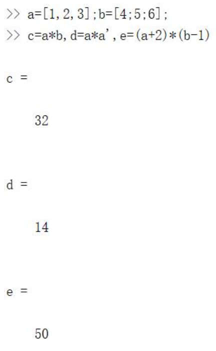
点乘如下

8.3 冒号的用法

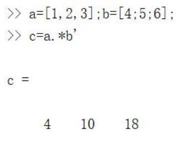
冒号“：”创建一个行向量的快捷方式。而且，冒号还可以用来查看或提取向量的某些元素。例如



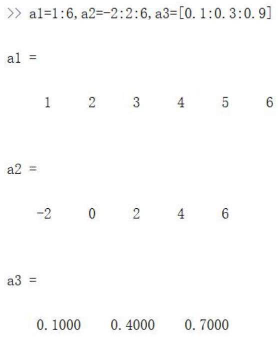
```
>> A=[1,2,3];B=[4,5,6];
>> C=5*A

C =

     5    10    15

>> C=5+A

C =

     6     7     8

>> D=A+B

D =

     5     7     9
```



```
>> a=[1,2,3];b=[4;5;6];
>> c=a*b,d=a*a',e=(a+2)*(b-1)

c =

    32


d =

    14


e =

    50
```



```
>> a=[1,2,3];b=[4;5;6];
>> c=a.*b'

c =

     4    10    18
```



```
>> a1=1:6,a2=-2:2:6,a3=[0.1:0.3:0.9]

a1 =

     1     2     3     4     5     6


a2 =

    -2     0     2     4     6


a3 =

    0.1000    0.4000    0.7000
```

注意：冒号的用法是：起始值：步长：终值。

从 a1 中提出第 3-5 个元素：

```
>> a11=a1(3:5)

a11 =

     3     4     5
```

从 a1 中提取第 2,4,6 个元素：

```
>> a12=a1(2:2:6)

a12 =

     2     4     6
```

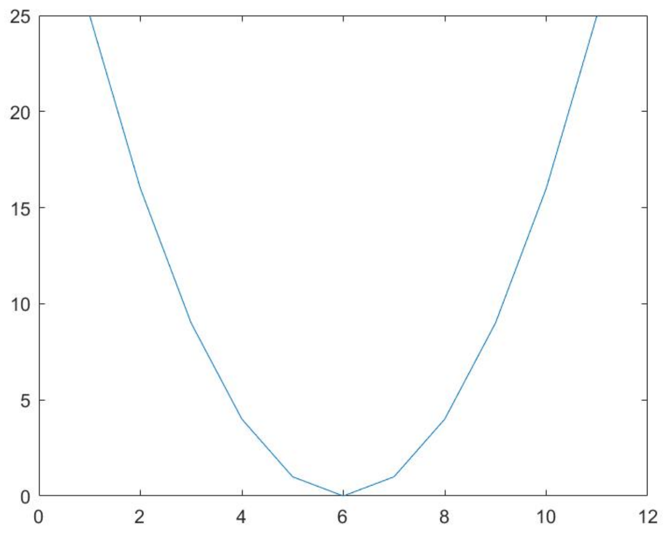
8.4 作图

Matlab 还有强大的作图功能。

在 Matlab 里创建一个 2D 图的基本函数是 **plot(.)**。例如，作出抛物线 $Y = X^2$ 的图

```
>> X=-5:5;
>> Y=X.^2;
>> plot(Y)
```

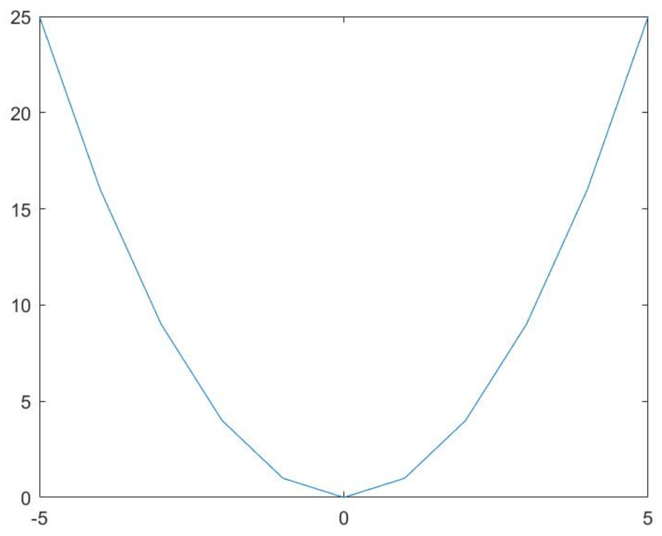
得到如下图



如果想显示 X 的范围：

```
>> plot(X,Y)
```

得到如下图



还可以在上图中加入图标题，X 轴和 Y 轴的标题：

```
>> title('抛物线');
>> xlabel('X')
>> ylabel('Y')
```

得到如下图

而作 3D 图形的函数为 **plot3(.)**。其基本用法与 plot 相似，只是多了一个元素。例如，作出 $Y=(XZ)^2$ 的图：

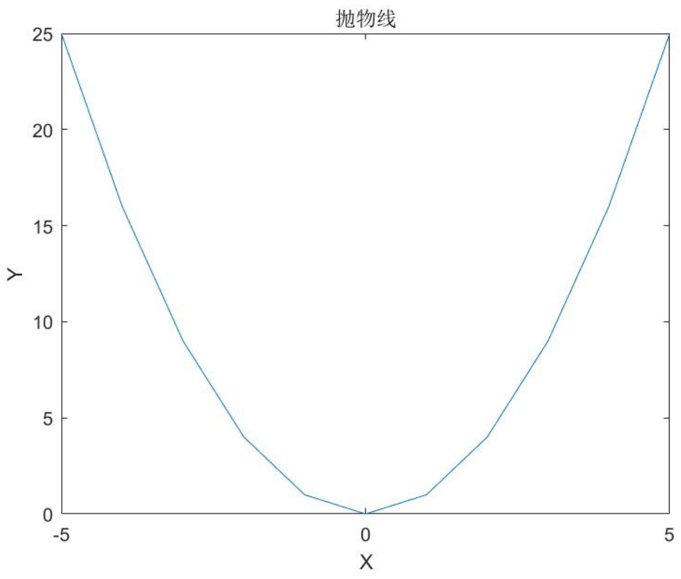
得到如下图

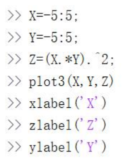
还可以用 **meshgrid(.)**函数作出两个变量的面：

得到如下图

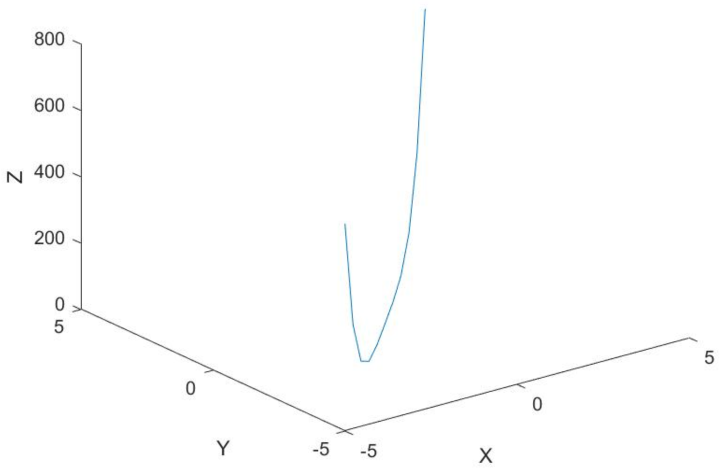
还可以画出它们的等高线：

得到如下图

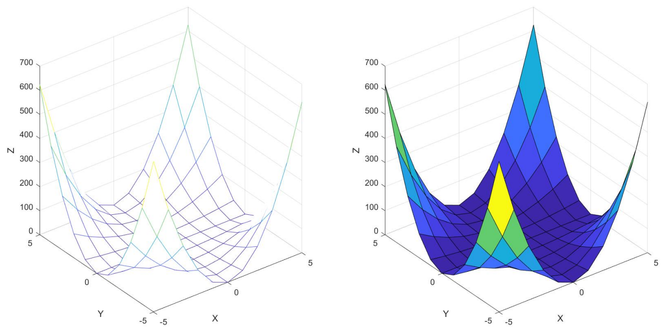




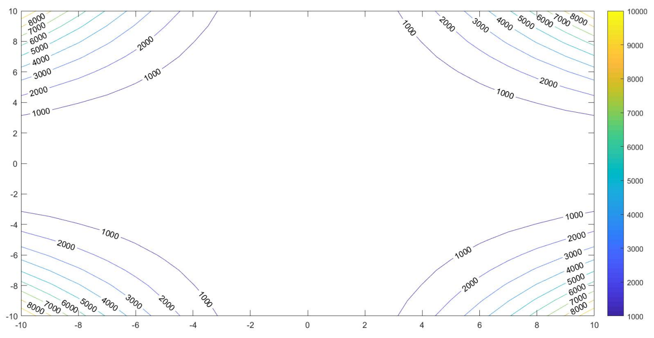
```
>> X=-5:5;
>> Y=-5:5;
>> Z=(X.*Y).^2;
>> plot3(X,Y,Z)
>> xlabel('X')
>> zlabel('Z')
>> ylabel('Y')
```



```
>> X=-5:5;
>> Y=-5:5;
>> [x,y]=meshgrid(X,Y);
>> Z=(x.*y).^2;
>> subplot(1,2,1)
>> mesh(X,Y,Z)
>> xlabel('X');ylabel('Y');zlabel('Z');
>> subplot(1,2,2)
>> surf(X,Y,Z)
>> xlabel('X');ylabel('Y');zlabel('Z');
```



```
>> X=-10:10;
>> Y=-10:10;
>> [x,y]=meshgrid(X,Y);
>> Z=(x.*y).^2;
>> [c,h] = contour(x,y,Z); clabel(c,h), colorbar; hold
```

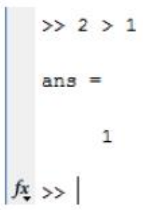


8.5 Boolean 表达式和循环

8.5.1 Boolean 表达式

当我们试图在某些条件下才执行程序时，我们首先需要判断这些条件 (或陈述) 是否为“真”，这些条件被称为“布林表达式”。

当我们在命令窗口输入“2>1”时，Matlab 会输出一个整数“0”或“1”，即告诉我们输入的表达式是否为“真”（1 表示“真”，0 表示“假”）。

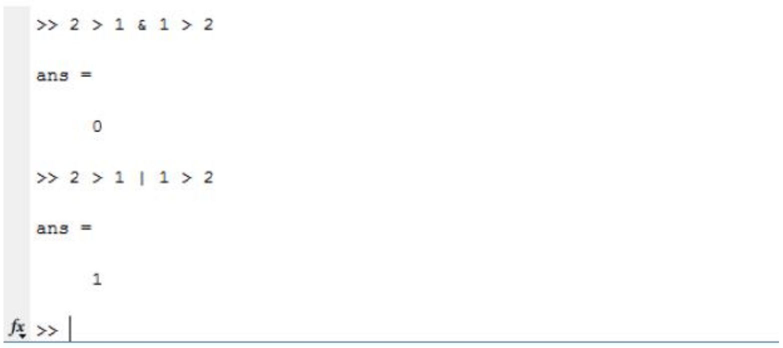


```
>> 2 > 1

ans =

     1

>> 
```

除此之外，还有其他一些关系符号：

- >= 表示大于或等于；
- < 表示严格小于；
- <= 表示小于或等于；
- == 表示等于（注意：两个等号）；
- 表示“非”。

我们也可以用“&(和)”或者“||(或)”来连接几个布林表达式，例如



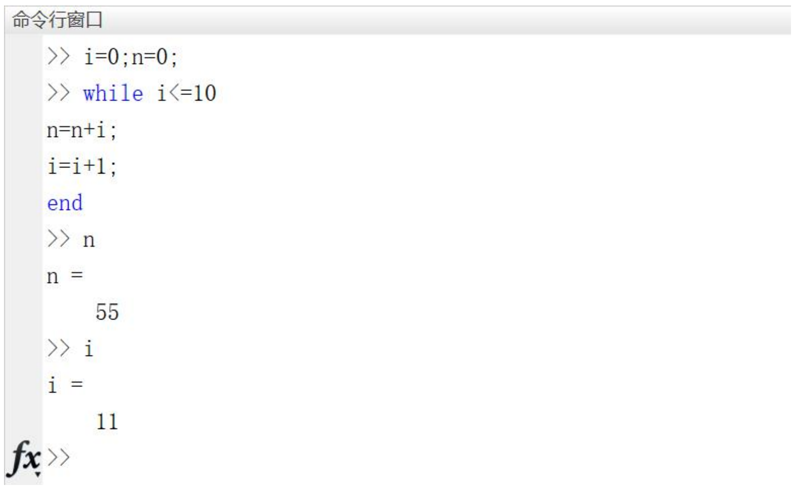
```
>> 2 > 1 & 1 > 2

ans =

     0

>> 2 > 1 | 1 > 2

ans =

     1

>> 
```

8.5.2 循环

对于 Matlab 来说，循环（loop）是指只要给定的布林陈述为“真”就执行某些指定的命令集。循环包括三种类型：while 循环、for 循环和 if 循环。它们各有优势和劣势。注意：我们要尽可能的避免使用循环语句，因为循环并不是矩阵运算，它们会使得 Matlab 代码运行得极其缓慢。

1. while 循环执行的是**当（while）**一个布林陈述为“真”时的命令集。

   我们首先要输入“while”来陈述条件，然后增加需要执行的命令。当我们想要命令集停止执行时，要输入“end”。例如，求 0 到 10 的整数之和：

```
>> i=0;n=0;
>> while i<=10
n=n+i;
i=i+1;
end
>> n
n =
    55
>> i
i =
    11
fx >>
```

   在上例中

   i 表示一个整数变量，指代我们使用的整数；n 也表示一个整数变量，指代我们求和的结果。在求和的每一步，n 的值都有表达式“$n = n+i$”来升级。那么，Matlab 如何读取 while 循环命令呢？

   第一步，读取布林陈述的右边，$n + i$，然后将其值暂时储存到工作区，例如，i=0 时，n=0，那么 n+i=0，0 这个值就会暂时储存在工作区；

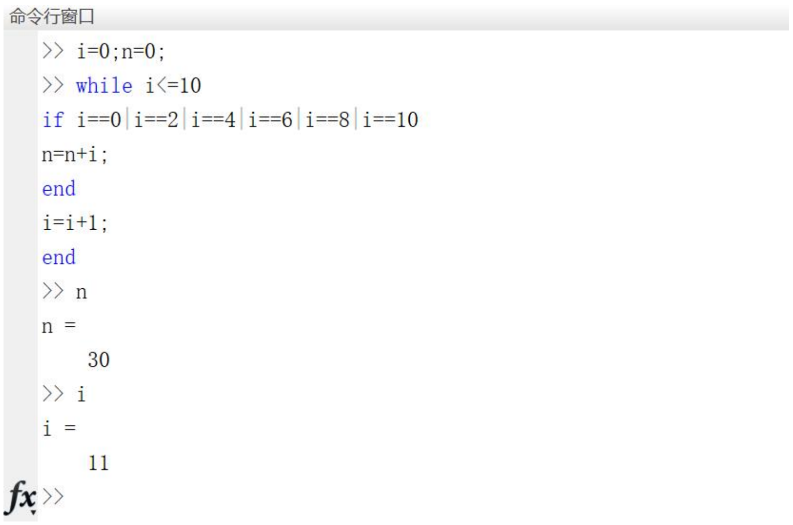
   第二步，创建一个新的变量，n，让新的 n 等于储存在工作区的临时值，例如，新创建的 n 要等于工作区的值，即 n=0；

   第三步，当 while 循环中的布林陈述为“真”时，重复前两步，并不断升级 n 的值，直到求出最终的结果，例如，当 i=1 时，因为第二步中

的 n=0，所以 n+i=0+1=1,1 这个值就储存在工作区，然后新创建的 n 就等于 1，重复这些步骤，最终得到 n=55。

2. if 循环执行的是“**如果 (if)**”给定布林陈述为“真”的命令集。

   其语法结构与 while 循环类似：以 if 开始，end 结束。例如，我们在上述整数求和的基础上，继续求解偶数之和。



```
>> i=0;n=0;
>> while i<=10
if i==0|i==2|i==4|i==6|i==8|i==10
n=n+i;
end
i=i+1;
end
>> n
n =
    30
>> i
i =
    11
fx >>
```

   在上述偶数求和的例子中，我们在 while 循环中加入了 if 表达式。对于每一个 i 的值，matlab 都要检验这个 if 表达式，如果 i 是偶数，那么，执行 n=n+1 运算。**注意：每一个布林表达式都需要在命令集结尾处加上”end”。**

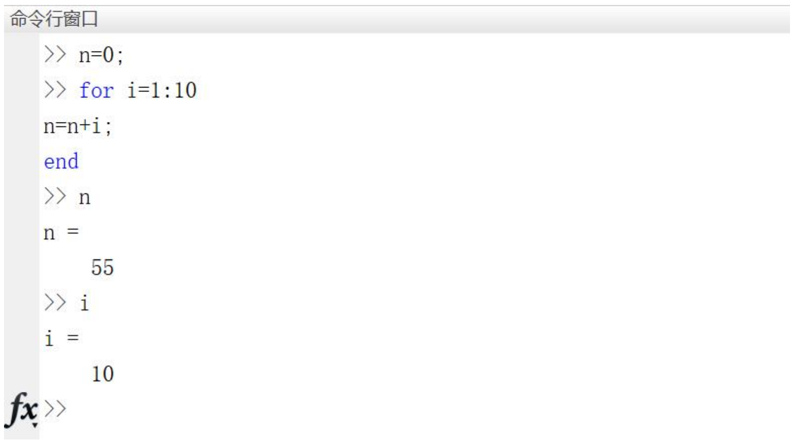
3. for 循环，严格来说，它并不像 while 和 if 循环一样，只要给定的布林表达式为“真”就执行命令集。它是对于一个“指示变量”值执行给定命令集。例如，while 和 if 循环中的整数求和，使用 for 循环可以使用一种更为简洁的方式来执行，即创建一个 n*1 阶向量。

   首先，计算整数的和

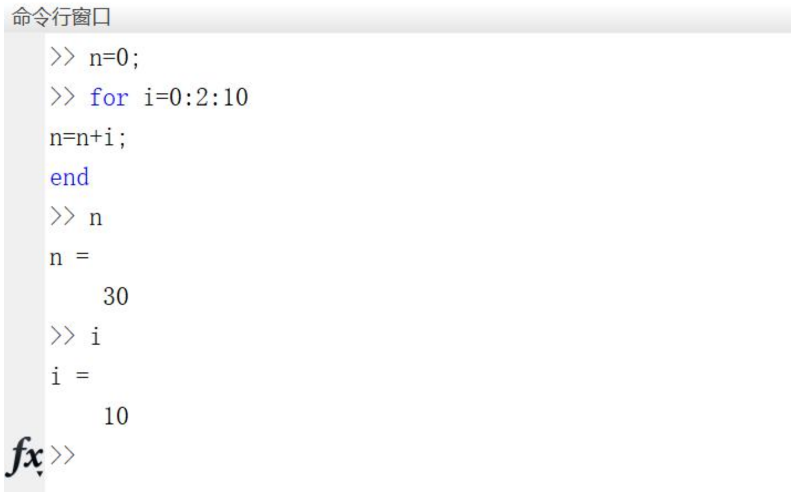
然后，计算偶数的和

这里使用了冒号，我们可以回忆一下前文对冒号“:”用法的阐述。即“0:2:10”告诉 Matlab 创建一个从 0 到 10 的向量，步长为 2，即 0,2,4,6,8,10。

注：Matlab 的优势在于矩阵处理，而布林表达式也可以用于矩阵。



```
>> n=0;
>> for i=1:10
n=n+i;
end
>> n
n =
    55
>> i
i =
    10
fx >>
```



```
>> n=0;
>> for i=0:2:10
n=n+i;
end
>> n
n =
    30
>> i
i =
    10
fx >>
```

8.6 一些有用的内置函数

8.6.1 fzero

fzero是求解非线性函数 $f(x) = 0$ 根的一个 Matlab 内置算法命令。
它具有以下几种命令格式：

- $x = fzero(fun, x0)$，其中，fun表示函数名，x0表示初值。

  该命令尝试找到一个点 x，使得 $fun(x) = 0$。在根 x 处，函数 $fun(x)$ 改变符号。注意：$fzero$ 不能求解诸如 x^2 这类函数的根。

  例子 1：某一点附近的根——计算 3 附近的正选函数 $sin(x) = 0$ 的根。

```
fun = @sin; % 函数名
x0 = 3; % 初值
```

```
x = fzero(fun,x0)
```

$x = 3.1416$

例子 2：区间内的根——求解 [0,1] 上的余弦函数的根。

```
fun = @cos; % 函数名
x0 = [1 2]; % 初始区间
x = fzero(fun,x0)
```

$x = 1.5708$

注意：$cos(1)$ 和 $cos(2)$ 的符号不同。

例子 3：自定义函数求根

求解函数 $f(x) = x^3 - 2x - 5$ 的根。

首先，我们要写一个自定义函数的 m 文件，命名为 f.m

```
function y = f(x)
y = x.^3-2*x-5;
```

将上述 f.m 文件保存到当前 Matlab 路径中。然后用 $fzero$ 命令求解 2 附近的根。

```
fun = @f; % function
x0 = 2; % initial point
z = fzero(fun,x0)
```

$x = 2.0946$

例子 4：带有参数的函数求根

```
myfun = @(x,c) cos(c*x);  % parameterized function
c = 2;                    % parameter
fun = @(x) myfun(x,c);    % function of x alone
x = fzero(fun,0.1)
```

$x = 0.7854$

- $x = fzero(fun, x0, options)$，其中，options 表示可选项。

  例子 5：设定一些图像功能来显示解的过程

  定义函数和初值

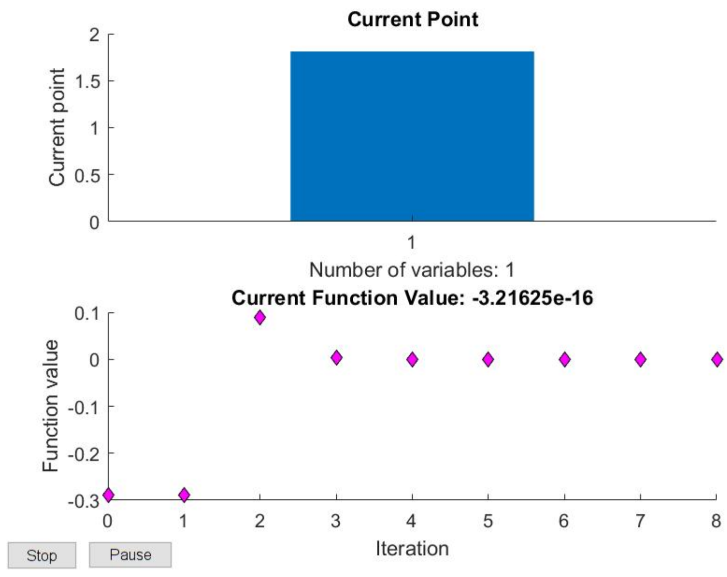
```
fun = @(x)sin(cosh(x));
x0 = 1;
```

  设定图像选项来显示解的过程

```
options = optimset('PlotFcns',{@optimplotx,@optimplotfval});
```

  运行 $fzero$：

```
x = fzero(fun,x0,options)
```



  $x = 1.8115$

- $[x, fval, exitflag, output] = fzero()$，该命令结构在 $fval$ 中显示 $fun(x)$ 的值，$exitflag$ 显示 $fzero$ 停止的原因，而 $output$ 中则包含了解的过程的信息。

  例子 6：

```
fun = @(x) exp(-exp(-x)) - x; % function
```

```
x0 = [0 1]; % initial interval
options = optimset('Display','iter'); % show iterations
[x fval exitflag output] = fzero(fun,x0,options)
```

8.6.2 fsolve

$fsolve$ 是解非线性方程组的 Matlab 内置解法。解下列问题

$$F(x) = 0$$

对于 x 来说，$F(x)$ 是函数值，以向量形式表示；x 是向量或者矩阵。
命令格式及描述为：

- $x = fsolve(fun, x0)$ starts at x0 and tries to solve the equations fun(x) = 0, an array of zeros.
- $x = fsolve(fun, x0, options)$ solves the equations with the optimization options specified in options. Use *optimoptions* to set these options.
- $x = fsolve(problem)$ solves problem, where problem is a structure described in Input Arguments. Create the problem structure by exporting a problem from Optimization app, as described in Exporting Your Work.
- $[x, fval] = fsolve()$, for any syntax, returns the value of the objective function fun at the solution x.
- $[x, fval, exitflag, output] = fsolve()$ additionally returns a value exitflag that describes the exit condition of fsolve, and a structure output with information about the optimization process.
- $[x, fval, exitflag, output, jacobian] = fsolve()$ returns the Jacobian of fun at the solution x.

例子 1：方程组为

$$e^{-e^{-(x_1+x_2)}} = x_2(1+x_1^2)$$
$$x_1cos(x_2) + x_2sin(x_1) = \frac{1}{2}$$

首先，将上式转化为 $F(x)=0$ 的形式

$$e^{-e^{-(x_1+x_2)}}-x_2(1+x_1^2)=0$$

$$x_1cos(x_2)+x_2sin(x_1)-\frac{1}{2}=0$$

然后，写一个 m 文件

```
function F = root2d(x)


F(1) = exp(-exp(-(x(1)+x(2)))) - x(2)*(1+x其中，(1)^2);%表
示向量，包含两个元素x$x_1,，分别对用着x_2$x和(1)x(2)
F(2) = x(1)*cos(x(2)) + x(2)*sin(x(1)) - 0.5;
```

将上述自定义函数保存为 root2d.m，存于 Matlab 当前文件夹。

从点 [0,0] 处开始解上述方程组：

```


fun = @root2d;
x0 = [0,0];
x = fsolve(fun,x0)
```

结果为

Equation solved.

fsolve completed because the vector of function values is near zero as measured by the value of the function tolerance, and the problem appears regular as measured by the gradient.

x =

0.3532 0.6061

8.6.3 参数传递

Sometimes objective or constraint functions have parameters in addition to the independent variable. The extra parameters can be data, or can represent variables that do not change during the optimization. There are three methods of passing these parameters:

- Anonymous Functions
- Nested Functions

- Global Variables

Global variables are troublesome because they do not allow names to be reused among functions. It is better to use one of the other two methods.

For example, suppose you want to minimize the function

$$f(x) = (a - bx_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-c + cx_2^2)x_2^2$$

for different values of $a, b, andc$. Solvers accept objective functions that depend only on a single variable (x in this case). The following sections show how to provide the additional parameters $a, b, andc$. The solutions are for parameter values $a = 4, b = 2.1, andc = 4$ near $x_0 = [0.50.5]$ using $fminunc$.

Anonymous Functions

To pass parameters using anonymous functions:

1. Write a file containing the following code:

```
function y = parameterfun(x,a,b,c)
y = (a - b*x(1)^2 + x(1)^4/3)*x(1)^2 + x(1)*x(2) + ...
    (-c + c*x(2)^2)*x(2)^2;
```

2. Assign values to the parameters and define a function handle f to an anonymous function by entering the following commands at the MATLAB prompt:

```
    a = 4; b = 2.1; c = 4; % Assign parameter values
    x0 = [0.5,0.5];
    f = @(x)parameterfun(x,a,b,c);
```

3. Call the solver $fminunc$ with the anonymous function:

```
    [x,fval] = fminunc(f,x0)
```

The following output is displayed in the command window:

Local minimum found.

Optimization completed because the size of the gradient is less than the default value of the function tolerance.

x = -0.0898 0.7127

fval = -1.0316

Nested Functions

To pass the parameters for Equation 1 via a nested function, write a single file that

- Accepts a, b, c, and x0 as inputs
- Contains the objective function as a nested function
- Calls *fminunc*

Here is the code for the function file for this example:

```
function [x,fval] =  runnested(a,b,c,x0)
[x,fval] = fminunc(@nestedfun,x0);

% Nested function that computes the objective function
    function y = nestedfun(x)
        y = (a − b*x(1)^2 + x(1)^4/3)*x(1)^2 + x(1)*x(2) +...
            (−c + c*x(2)^2)*x(2)^2;
    end
end
```

The objective function is the nested function nestedfun, which has access to the variables a, b, and c.

To run the optimization, enter:

```
a = 4; b = 2.1; c = 4;% Assign parameter values
x0 = [0.5,0.5];
[x,fval] = runnested(a,b,c,x0)
```

8.6.4 csolve

help csolve

$function[x, rc] = csolve(FUN, x, gradfun, crit, itmax, varargin)$

FUN should be written so that any parametric arguments are packed in to xp, and so that if presented with a matrix x, it produces a return value of same dimension of x. The number of rows in x and FUN(x) are always the same. The number of columns is the number of different input arguments at which FUN is to be evaluated.

gradfun: string naming the function called to evaluate the gradient matrix. If this is null (i.e. just "[]"), a numerical gradient is used instead.

crit: if the sum of absolute values that FUN returns is less than this, the equation is solved.

itmax: the solver stops when this number of iterations is reached, with rc=4

varargin: in this position the user can place any number of additional arguments, all of which are passed on to FUN and gradfun (when it is non-empty) as a list of arguments following x.

rc: 0 means normal solution, 1 and 3 mean no solution despite extremely fine adjustments in step length (very likely a numerical problem, or a discontinuity). 4 means itmax termination.

8.6.5 broyden

help broyden

broyden

Computes root of function from R^n to R^n using Broyden's inverse update method with backstepping.

Usage

$[x, fval, fjacinv] = broyden(f, x, varargin)$

Input

f : function of form fval=f(x,varargin)

x : n.1 initial guess for root

varargin : optional parameters passed to f

Output

x : n.1 root of f

fval : n.1 function value estimate

fjacinv : n.n inverse Jacobian estimate

Options

maxit : maximum number of iterations (100)

tol : convergence tolerance (1e-10)

maxsteps : maximum number of backsteps (25)

showiters : display results of each iteration (1)

initb : an initial inverse Jacobian approximation matrix ([])

initi : if initb empty, and initi is 1, use identity matrix to initialize if initi is 0, use numerical Jacobian (0)

8.7 自定义函数

在命令窗口直接输入命令会遇到很多问题，尤其是程序较长时。Matlab 也为我们提供了另一种代码编辑窗口——脚本窗口。脚本窗口中可以输入一系列命令，我们将这些命令保存为 matlab 默认文件——m 文件（之所以称为 m 文件是因为文件的后缀为.m）。

如果 m 文件不需要外部输入 (input)，这些命令就可以运行，那么，这个文件称为脚本（scripts）文件。

另一种 m 文件就是函数 (function)。函数文件的命令需要外部输入才能运行。

注：m 文件的名字应该与函数的名字相同。我们可能想要用自定义函数来求解 DSGE 模型的稳态。

例如，定义一个简单的抛物线函数 $f(x) = x^2$。其中，x 是外部输入，函数 f 返回到 x^2 的值。

首先要新建脚本，也就是 m 文件：

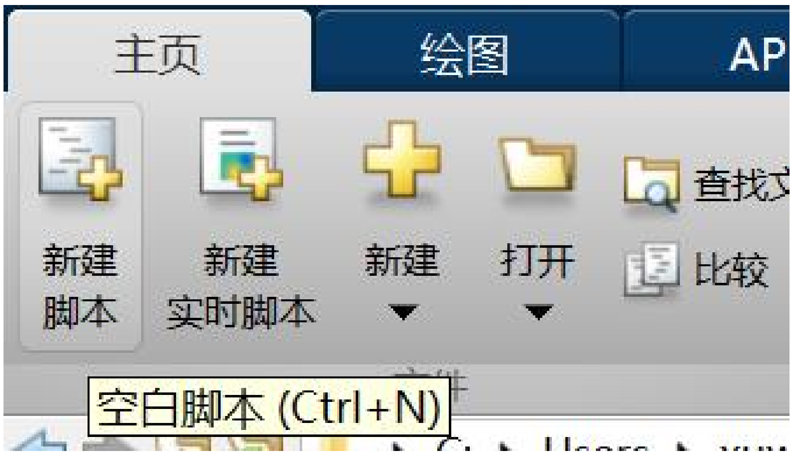
定义函数：

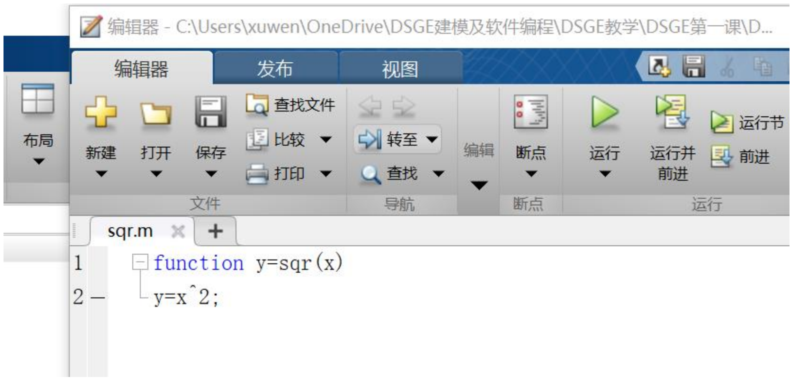
要创建一个函数文件：

第一步，我们需要在第一行命令以“**function**”开始，然后紧接着是该函数的输出（在上例中，y 是输出），等于函数名（例子中是“sqr”）和输入（x）。

第二步，编写出我们的相关命令。例子中就是给输出 y 赋值，赋值为输入 x 的平方。

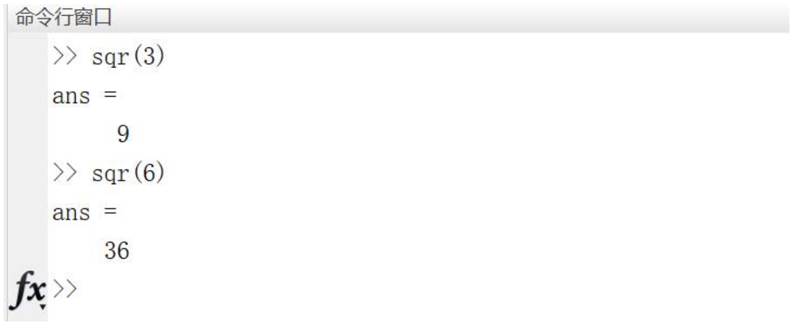
更加复杂的函数可能有多个输入和多个输出，或者不同类型的输入和输出（例如，矩阵，字符或其它函数）。





```
function y=sqr(x)
y=x^2;
```

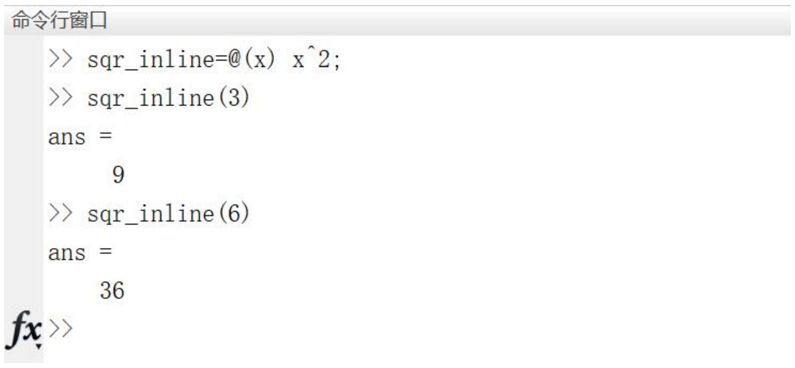
为了调用 sqr.m 文件，我们只需要在 matlab 命令窗口输入：



```
>> sqr(3)
ans =
     9
>> sqr(6)
ans =
    36
fx >>
```

此外，我们还可以利用所谓的“in-line”函数在 matlab 的命令窗口中直接编写简单的函数：

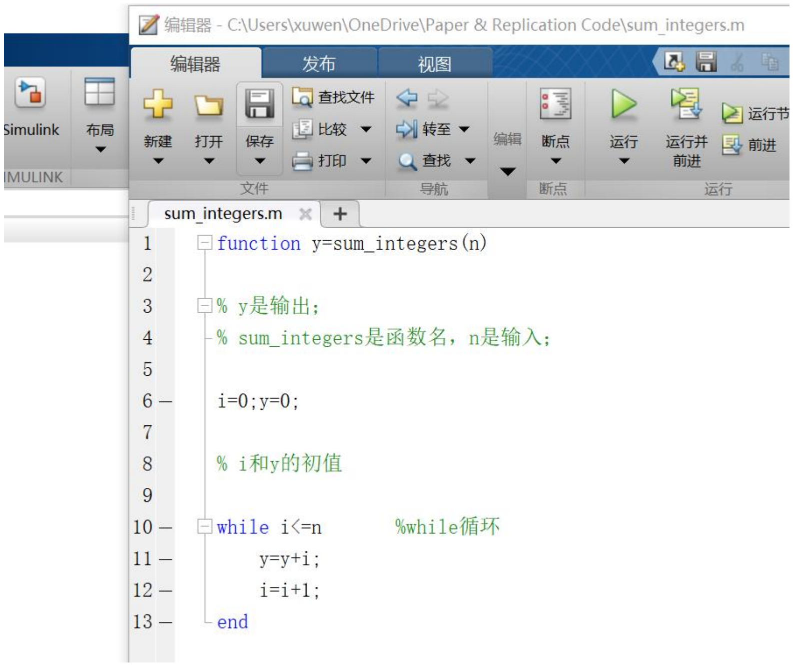
第一行命令就是定义函数，等号左边是函数名“*sqr_inline*”，等号右边”@(x)”告诉 Matlab，我们定义一个输入为 x 的函数，紧接着就是告诉 Matlab 该函数的返回值（输出值）x^2。例如，我们调用函数“*sqr_inline*”，输入为 3，返回值为 9。



命令行窗口

```
>> sqr_inline=@(x) x^2;
>> sqr_inline(3)
ans =
     9
>> sqr_inline(6)
ans =
    36
fx >>
```

下面，我们来编写一个更复杂的函数文件：计算前 n 个整数的和，整数 n 是函数的输入。



```
function y=sum_integers(n)

% y是输出;
% sum_integers是函数名，n是输入;

i=0;y=0;

% i和y的初值

while i<=n        %while循环
    y=y+i;
    i=i+1;
end
```

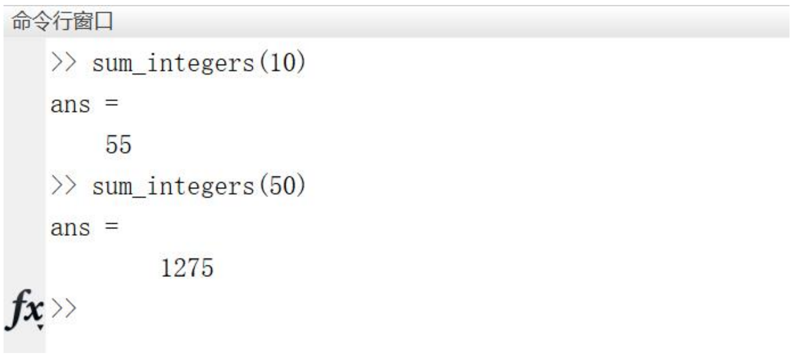
保存上述文件，文件名为函数名：*sum_integers.m*。现在，我们来调用这个函数，计算整数求和。

Matlab 中也内置了一些常用的函数。

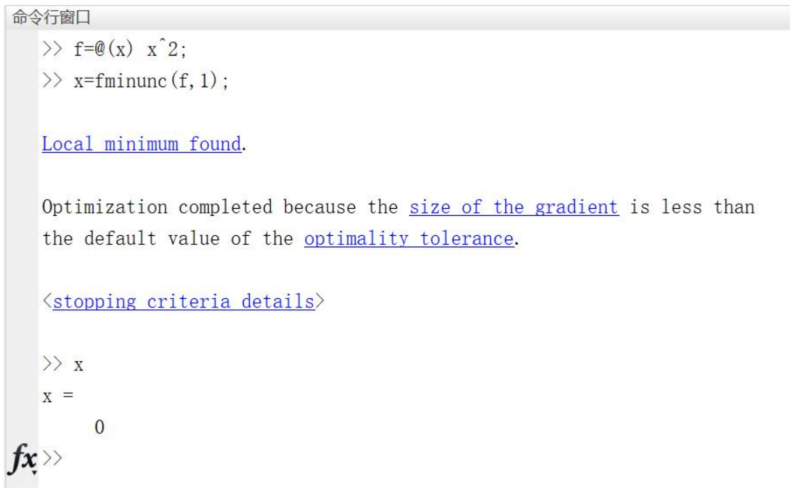
1、函数最小值

例如，我们想要解出使得抛物线 $f(x) = x^2$ 最小的 x。此时，我们可以使用 Matlab 内置的最小化函数“**fminunc**”。

上例中，第一行先定义了 in-line 函数 f。第二行命令，用函数“fminunc”来解出使函数 f 最小的 x 值。函数“fminunc”有两个输入元素：**最小化目标**



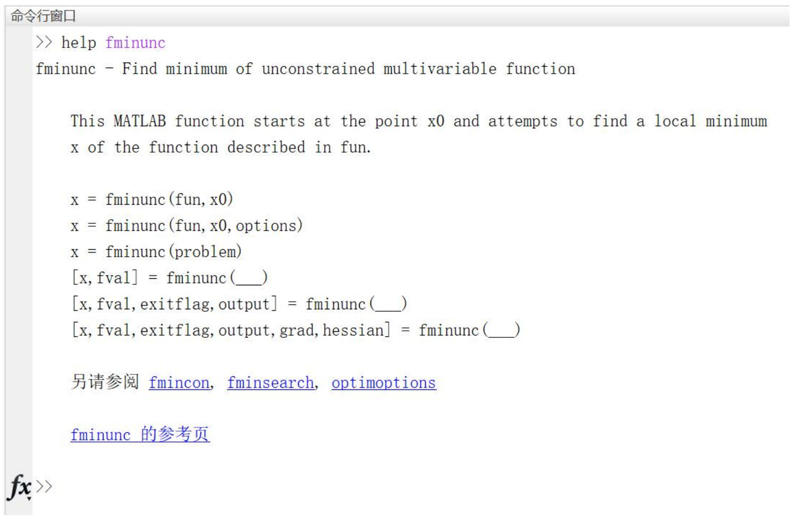
```
>> sum_integers(10)
ans =
    55
>> sum_integers(50)
ans =
        1275
fx >>
```



```
>> f=@(x) x^2;
>> x=fminunc(f,1);

Local minimum found.

Optimization completed because the size of the gradient is less than
the default value of the optimality tolerance.

<stopping criteria details>

>> x
x =
     0
fx >>
```

函数 f 和 x 的初始猜测值 1。

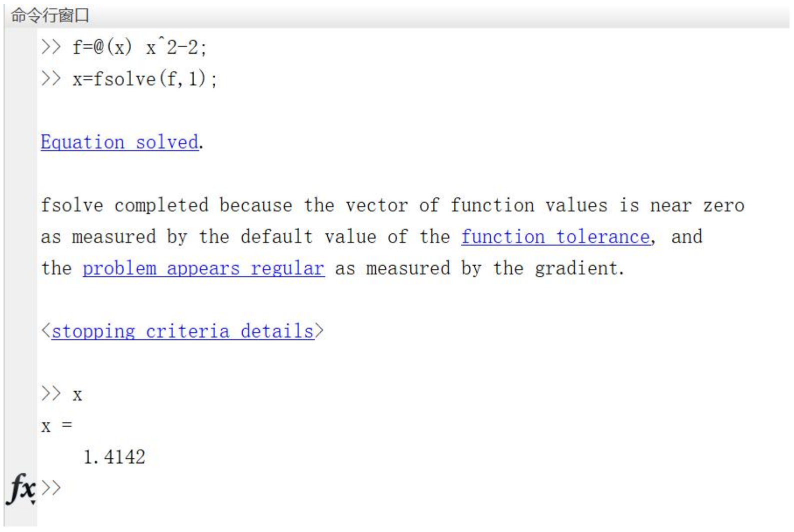
更多关于函数“fminunc”的信息，请查看帮助文件或在线资源。Matlab 中更多的最优化函数请参见 Matlab 的帮助文档或在线资源。



```
>> help fminunc
fminunc - Find minimum of unconstrained multivariable function

    This MATLAB function starts at the point x0 and attempts to find a local minimum
    x of the function described in fun.

    x = fminunc(fun,x0)
    x = fminunc(fun,x0,options)
    x = fminunc(problem)
    [x,fval] = fminunc(___)
    [x,fval,exitflag,output] = fminunc(___)
    [x,fval,exitflag,output,grad,hessian] = fminunc(___)

    另请参阅 fmincon, fminsearch, optimoptions

    fminunc 的参考页

fx >>
```

2、求根问题

另一个常用的函数就是求根问题，即解方程的根——找到 x 的值使得函数 $f(x) = 0$。常用的求根函数是“fsolve”：例如，求解函数 $f(x) = x^2 - 2$ 的根



```
>> f=@(x) x^2-2;
>> x=fsolve(f,1);

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>

>> x
x =
    1.4142
>>
```

需要注意的是，通用目的的函数运算并不是矩阵运算，因此，Matlab 并不善于此。但是，Matlab 比较善于做线性运算，因此，如果我们能将非线性方程转化为线性方程，那么，matlab 还是可以很好的胜任。例如，上例中，虽然我们要求解 $f(x) = x^2 - 2$ 的根，但是 matlab 求解 $x - \sqrt{2} = 0$ 的根，因此，我们看到最后的结果输出只有 1.4142。

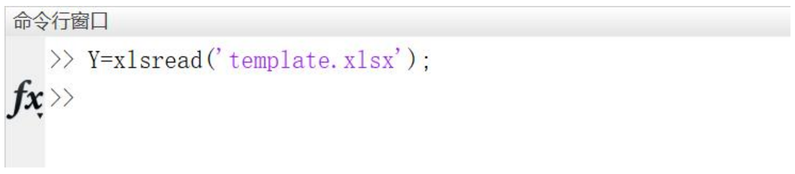
当然，为了求解线性方程的根，Matlab 还内置了许多其它的求根函数，请参见 Matlab 的帮助文档或在线资源。

8.8 读取数据

上面，我们已经尝试了一些创建数据文件的命令。但是，在实践中，我们通常需要直接读取外部数据文件，尤其是非 m 文件和 mat 文件。例如，我们要读取最常见的 excel 文件。幸运的是，Matlab 可以直接读取 excel 文件。

例如，我们的 excel 文件名为“template.xlsx”，用 matlab 读取该文件

其中，Y 表示读入文件所保存的变量名称，xlsread 表示 excel 文件读取函数，template.xlsx （或者 filename.xls）表示 excel 数据文件。需要注意的是，数据文件要用英文字符下的单引号括起来。



```
>> Y=xlsread('template.xlsx');
>>
```