Dynare の使い方について

蓮見 亮 *

2025 年 9 月 Ver. 3.3

目次

* 本文中で説明に用いているプログラム類は、https://github.com/rhasumi/dynamicmodels/からダウンロードできる。

1 概要

2 節以下では、**Octave/MATLAB と Dynare がインストールされた PC が準備されていることを前提とする**が、Dynare と Octave/MATLAB の概要についてごく簡単に解説しておく。

1.1 Dynare とは何か、Dynare で何ができるのか

Dynare とは、DSGE モデルを解くためのツールである。パラメータを所与として、モデルの定常状態、パラメータを変化させた場合の新しい定常状態への移行過程、外生ショックに対するインパルス応答などが計算できる（3 節参照）。また、モデルに合ったデータを与えれば、それにフィットするようなパラメータを推定することもできる（4 節参照）。

Dynare は単体では利用できない。数値計算ソフトである Octave もしくは MATLAB と組み合わせて使用する。Dynare は以下のページから無償でダウンロードできる。基本的には最新の “stable release” を選べばよい。

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

1.2 Octave とは何か

Octave は GNU（フリーソフトの普及を目指す財団）によって提供されている数値計算用のフリーソフトである。Dynare の利用を含め、幅広い用途に用いることができる。他の数値計算用のソフトウェアと比較した場合の Octave の特徴は、MATLAB との高い互換性にある[*1]。

Octave は以下のページからリンクを辿っていくとダウンロードできる（もしくは、Dynare のダウンロードページにリンクが張られている）。

https://octave.org/

Windows の場合、2025 年 9 月時点では、Dynare の最新の stable である 6.4 を使うには、Octave 10.2.0（64-bit）をインストールする。GUI 版と CUI 版（コマンドプロンプト内

[*1] MATLAB とは米国の MathWorks 社が開発・販売している数値計算用のソフトウェアであり、この種のソフトウェアとしては最も古く、今でも広く使われている。ただし、一般の商用のソフトウェアであるため使い勝手は非常によいが、とても高価である。

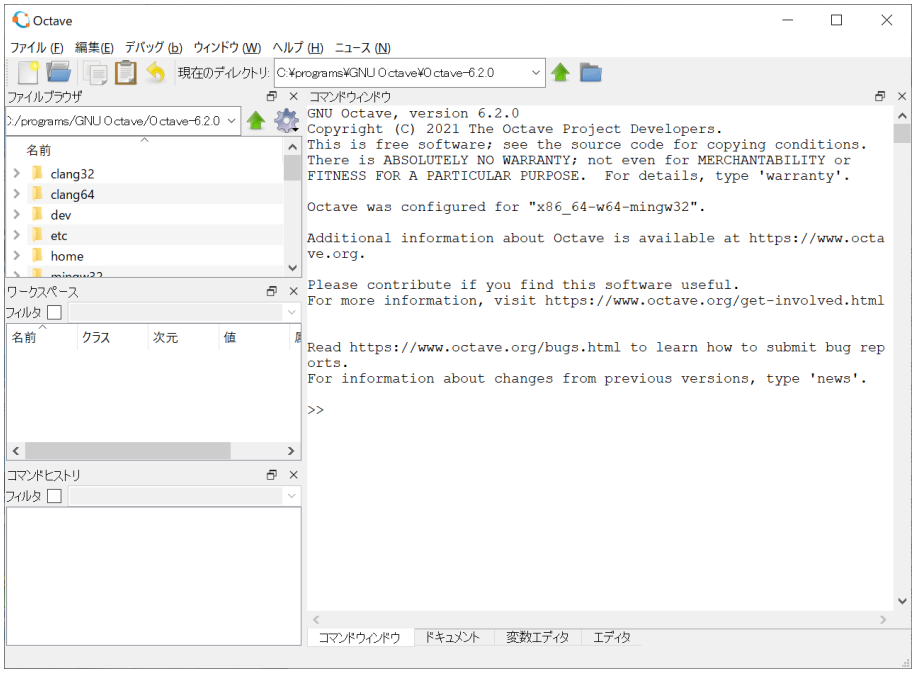

図 1　Octave のウインドウ（GUI 版）

で起動）がインストールされるが、**GUI 版でプログラムの実行時にエラーがでる場合は CUI 版を用いる**[*2]。

1.3　プログラムのファイル形式について

Dynare と Octave/MATLAB のプログラムファイル自体は**テキスト形式**だが、拡張子を適宜.txt から **Dynare プログラムの場合には拡張子.mod に、Octave/MATLAB プログラムの場合には拡張子.m に変更する**[*3]。これらのプログラムファイルは Octave(GUI 版)/MATLAB のテキストエディタで編集できる。

DSGE モデルを Dynare に解かせる場合には基本的に **mod ファイルのみを用意すればよい。**ただし、mod ファイルの Dynare プログラムには、Octave/MATLAB プログラムを部分的に書き込むことができる。したがって、Dynare を使う場合には Octave/MATLAB プログラムの書き方を知っているほうが望ましい[*4]。

[*2] コマンドプロンプト（CUI 版 Octave）では、「範囲選択→ enter キー」でコピー、「右クリック」でペーストである。

[*3] 設定で拡張子が表示されるようにしてほいたほうがよい（Windows 11 だと、エクスプローラーの表示タブの「ファイル名拡張子」にチェックを入れる）。

[*4] もちろん最初から全部を知っておく必要はなく、Octave/MATLAB はプログラム言語として扱いやすい部類に入るので徐々に慣れていけばよい。

2　プログラムの実行方法

Octave と Dynare がインストールされたパソコンと適切な mod ファイルが用意されていたとしよう。ここでは、3.1.2 節で説明する RBC モデルの mod ファイルの完成版が既に用意されていたものとする。

インストールされている Dynare のバージョンが 6.4 で、デフォルトの場所にインストールされているものとする。RBC モデルの mod ファイルの完成版である “RBC_det.mod” は、“C:\work\dsge” というフォルダに保存されているものとする[*5]。

この場合まず、テキストエディタを使って、

main.m

```
addpath C:\dynare\6.4\matlab
cd C:\work\dsge
dynare RBC_det
```

という内容の m ファイルを準備する。名前は何でもよいが、例えば “main.m” とし、（場所もどこでもよいが）分かりやすいように “RBC_det.mod” と同じフォルダに保存しておく。次に、Octave を起動後、この 3 行をコピーし、次いで Octave のウィンドウ内で右クリックしペーストする。これで、Octave/Dynare は “RBC_det.mod” を実行する。

なお、2 回目以降は、Octave で単に

```
main
```

と入力するか、あるいは

```
dynare RBC_det
```

と入力するだけでよい。

3　Dynare によるシミュレーション

3.1　ショック・シミュレーションの例（RBC_det.mod）

3.1.1　何を準備すればよいか

解きたい DSGE モデルに合わせて、以下を準備する。

[*5] 日本語エディタでは、多くの場合バックスラッシュ \ が Yen 記号 ¥ に置き換わっていることに注意する。

1. 内生変数一覧、外生変数一覧
2. パラメーター一覧とその値[a]
3. 方程式一覧
4. 定常状態の求め方
5. シミュレーションのシナリオ

[a] **データを与えれば Dynare でパラメータ推定することも可能だが（4 節参照）、3 節の例ではパラメータの値は予め与えられているものとする。**

以下のような RBC モデル

$$\frac{w_t}{C_t} = (\gamma + 1)\,\mu L_t^{\gamma} \tag{1}$$
$$\frac{C_{t+1}}{C_t} = \beta\,(r_{t+1} - \delta + 1) \tag{2}$$
$$Y_t = A_t K_t^{\alpha} L_t^{1-\alpha} \tag{3}$$
$$w_t = (1-\alpha)\,A_t K_t^{\alpha} L_t^{-\alpha} \tag{4}$$
$$r_t = \alpha A_t K_t^{\alpha-1} L_t^{1-\alpha} \tag{5}$$
$$K_{t+1} = Y_t + (1-\delta)\,K_t - C_t \tag{6}$$
$$\ln(A_{t+1}) = \rho \ln(A_t) + \varepsilon_{t+1} \tag{7}$$

を例として、具体的に説明する[*6]。

1. 内生変数一覧、外生変数一覧

まず、内生変数一覧、外生変数一覧は、それぞれ表 1、表 2 のように与えられる。一番右列のコードとはプログラムを記述する際の変数名であり、予め決めておく。プログラム内の変数名は大文字と小文字で区別されることに注意する。

2. パラメーター一覧とその値

パラメーター一覧とその値は、表 3 のように与える。

3. 方程式一覧

(1) 式〜(7) 式として既に与えた。内生変数の数と方程式数が等しいことを確認しておく（今回はどちらも 7)。

[*6] 『動学マクロ経済学へのいざない』（以下、拙著）4 章のモデルと全く同じモデルである。

変数	名称	コード
C_t	消費	C
L_t	労働供給	L
K_t	資本ストック	K
Y_t	生産量	Y
w_t	賃金	w
r_t	金利	r
A_t	技術水準	A

表 1　内生変数（RBC モデル）

変数	名称	コード
ε_t	技術ショック	e

表 2　外生変数（RBC モデル）

変数	名称	コード	値
α	資本分配率	alpha	0.3
β	割引因子	beta	0.99
δ	減耗率	delta	0.025
μ	労働の不効用の相対ウエイト	mu	1.0
γ	労働供給の弾力性の逆数	gamma	1.0
ρ	AR(1) 項の係数	rho	0.9

表 3　パラメータ（RBC モデル）

4. 定常状態の求め方

以下のように与えられる[*7]。

$$A^* = 1 \tag{8}$$

$$r^* = \beta^{-1} + \delta - 1 \tag{9}$$

$$\frac{K^*}{L^*} = \left(\frac{r^*}{A^* \alpha} \right)^{\frac{1}{\alpha - 1}} \tag{10}$$

$$\frac{Y^*}{L^*} = A^* \left(\frac{K^*}{L^*} \right)^{\alpha} \tag{11}$$

$$\frac{C^*}{L^*} = \frac{Y^*}{L^*} - \delta \frac{K^*}{L^*} \tag{12}$$

$$w^* = (1 - \alpha) A^* \left(\frac{K^*}{L^*} \right)^{\alpha} \tag{13}$$

$$L^* = \left\{ \frac{w^*}{(\gamma + 1)\mu} \right\}^{\frac{1}{\gamma + 1}} \left(\frac{C^*}{L^*} \right)^{-\frac{1}{\gamma + 1}} \tag{14}$$

[*7] 詳細については、拙著 4 章参照。

非線形の（つまり対数線型化していない）モデルを与える場合、この段階を省略して**適当な初期値を与えて Dynare に定常状態を求めさせることもできるが、モデルが大規模化すると失敗することが多くなる。**したがって、多くの場合は定常状態の求め方を予め準備しておく必要がある[*8]。

5. シミュレーションのシナリオ

初期（$t = 0$）に定常状態にあり、$t = 1$ に $e_1 = 0.01$ という 1% の技術ショックが加わったものとしよう。

3.1.2 プログラムの書き方

上記の 1〜5 を元に、以下の順序で mod ファイル “RBC_det.mod” を記述していく。

1. 内生変数、外生変数の宣言（var ブロック、varexo ブロック）
2. パラメータの宣言（parameters ブロック）と値の代入
3. 方程式の設定（model ブロック）
4. 定常状態の計算（手順のプログラムによる記述、initval ブロックと steady コマンド）、モデルのチェック（check コマンド）
5. シミュレーションの実行（perfect_foresight_setup, perfect_foresight_solver コマンド）とグラフ描写など

コメントアウト記号は m ファイルでは % 記号から改行まで、mod ファイルでは//記号から改行までである。その部分はコメントとして無視されるので、メモなどを自由に書き込める[*9]。

1. 内生変数、外生変数の宣言

まず、var ブロックで内生変数、varexo ブロックで外生変数の宣言を行う。セミコロンでそれぞれのブロックが閉じる。

```
// 1. 内生変数、外生変数の宣言
var C L K Y w r A;
varexo e;
```

[*8] このモデルでは定常値が解析的に解けているが、一般に必ずしもモデルの全ての変数の定常値がこのように表せるとは限らない。その場合には、連立方程式のソルバーの利用（Octave では fsolve 関数）を考慮に入れる。

[*9] mod ファイルには Octave/MATLAB プログラムを部分的に書き込むことができるので、実際には mod ファイルでは % 記号から改行までもコメントとして扱われ、無視される。

2. **パラメータの宣言と値の代入**

第 2 に、parameters ブロックでパラメータを宣言し、次いで値を代入する。

```
// 2．パラメータの宣言
parameters alpha beta delta mu gamma rho;

// パラメータ値の代入
alpha = 0.3;
beta = 0.99;
delta = 0.025;
mu = 1.0;
gamma = 1.0;
rho = 0.9;
```

3. **方程式の定義**

次に、model ブロックで方程式を定義する。model ブロックは “model;” で始まり “end;” までである。

このとき、$t+1$ 期の変数（フォワードルッキング変数）は例えば C(+1) のように、$t-1$ 期の変数は例えば A(−1) のように丸括弧を使って記述する。このとき、**状態変数（先決変数）は $t+1$ 期の変数が含まれない形に書き換える。**ここでの状態変数は K_t と A_t の 2 つであるが、(6) 式と (7) 式をそれぞれ

$$K_t = Y_{t-1} + (1-\delta) K_{t-1} - C_{t-1} \tag{15}$$

$$\ln(A_t) = \rho \ln(A_{t-1}) + \varepsilon_t \tag{16}$$

と書き換えておくと、K_{t+1} と A_{t+1} が含まれなくなる。したがって、model ブロックは以下のように記述する[*10]。

[*10] log や exp など基本的な関数も使用できる。

```
// 3．方程式の定義
model;
w/C = (gamma+1)*mu*L^gamma;
C(+1)/C = beta*(r(+1)-delta+1);
Y = A*K^alpha*L^(1-alpha);
w = (1-alpha)*A*K^alpha*L^(-alpha);
r = alpha*A*K^(alpha-1)*L^(1-alpha);
K = Y(-1)+(1-delta)*K(-1)-C(-1);
log(A) = rho*log(A(-1)) + e;
end;
```

4. 定常状態の計算、モデルのチェック

次に、定常状態の計算手順を記述し、initval ブロック内で Dynare が定常状態を計算する際の初期値として与える。定常値はここでは star 付きの変数に該当するが、特にどのような変数名にすべきか決まりはない。

```
// 4. 定常状態の計算
Astar = 1;
rstar = 1/beta + delta - 1;
K_L = (rstar/alpha/Astar)^(1/(alpha-1));
Y_L = Astar*K_L^alpha;
C_L = Y_L-delta*K_L;
wstar = (1-alpha)*Astar*K_L^alpha;
Lstar = (wstar/(gamma+1)/mu)^(1/(gamma+1))*C_L^(-1/(gamma+1));
Kstar = K_L*Lstar;
Ystar = Y_L*Lstar;
Cstar = C_L*Lstar;

// Dynare に定常状態を計算させる際の初期値
initval;
C = Cstar;
L = Lstar;
K = Kstar;
Y = Ystar;
w = wstar;
r = rstar;
A = Astar;
end;
```

steady コマンドにより、Dynare は与えられた初期値に基づき定常状態を計算する[*11]。ここでは予め Dynare に正解を与えているので、確かにそれが正しいことを確認する。

```
// Dynare に定常状態を計算させる
steady;

// 初期値と結果が変わらないことをチェック
[Cstar; Lstar; Kstar; Ystar; wstar; rstar; Astar]
```

[*11] initval ブロックは、steady コマンドが後ろにあれば Dynare に定常状態を計算させる際の初期値となるし、なければモデルを解く際の初期値そのものになる。プログラムの構成次第で役割が変わることに注意が必要。なお、steady コマンドより後に histval ブロック内で値を指定することで、モデルを解く際の初期値を任意に設定できる。

上記のプログラムの出力は以下のようになる。ans = の前までが Dynare が計算した定常値であり、ans = の後が自分で計算した定常値の出力である。両者が等しいことが確認できる。

出力

```
STEADY-STATE RESULTS:

C                1.31577
L                0.667162
K                14.3013
Y                1.6733
w                1.75566
r                0.035101
A                1
ans =

    1.315771
    0.667162
   14.301334
    1.673304
    1.755665
    0.035101
    1.000000
```

次いで、check コマンドでモデルとパラメータが最適成長モデルとして適切に定義されているかどうかをチェックする。

```
// モデルのチェック
check;
```

EIGENVALUES: 以下が check コマンドの出力結果である。ここでは、モデルを定常状態周りで一次近似した際にできる行列の固有値を計算している。ショックがなければモデルが充分時間が経過したあとに定常状態に戻るか（すなわちモデルが発散しないか）どうかが確認できる。計算された固有値のうち絶対値が 1 より大きいものの数と、フォワードルッキング変数の数（この場合 C(+1) と r(+1) で 2 つ）が等しいことが、階数（rank）の条件となる[*12]。この場合、“The rank condition is verified.（階数条件は満たされている）” という診断結果を得た。

[*12] Blanchard and Kahn [1980] の条件と意味合いは同じだが、計算しているものが異なる。

出力

```
EIGENVALUES:
         Modulus              Real         Imaginary

       1.528e-16         1.528e-16                 0
       1.842e-16         1.842e-16                 0
             0.9               0.9                 0
          0.9482            0.9482                 0
           1.065             1.065                 0
             Inf              -Inf                 0


There are 2 eigenvalue(s) larger than 1 in modulus
for 2 forward-looking variable(s)

The rank condition is verified.
```

5. シミュレーションの実行とグラフ描写など

初期 ($t = 0$) に定常状態にあり、$t = 1$ に $e_1 = 0.01$ という 1% の技術ショックを加えるというシナリオに対応するモデルの記述は以下のとおりである。perfect_foresight_solver コマンドの計算するシミュレーションは、完全予見解（perfect foresight solution）であり、この前に “perfect_foresight_setup(periods=150)” と指定すると、$t = 151$ に定常状態に到達するという仮定の元で方程式が解かれる。したがって、periods で指定する値は充分大きな値にする必要がある。

計算結果は、var ブロックで宣言した変数名の変数それ自体として、それぞれ保存される。**例えば、C_t のパスは、C という変数として保存される。長さは、$t = 0$ 〜 151 までの 152 である。**

```
// 5. シミュレーション
// シナリオの設定
shocks;
var e;
periods 1;
values 0.01;
end;

// シミュレーションの実行
perfect_foresight_setup(periods=150);
perfect_foresight_solver;
```

最後に、定常状態からの乖離率を計算し、グラフとして描写させるプログラムを書き込んでおこう。この部分は、完全に Dynare プログラムではなく部分的に埋め込まれた Octave/MATLAB プログラムにあたる。perfect_foresight_solver コマンドの計算結果は、各変数の水準として与えられるので、定常状態からの乖離率を計算している。

```
// 定常状態からの乖離率の計算
for i = 1:size(M_.endo_names, 1)
  assignin('base', string(M_.endo_names(i)), oo_.endo_simul(i,:)');
end
C1 = (C./Cstar-1)*100;
L1 = (L./Lstar-1)*100;
K1 = (K./Kstar-1)*100;
Y1 = (Y./Ystar-1)*100;
w1 = (w./wstar-1)*100;
r1 = (r-rstar)*100;
A1 = (A./Astar-1)*100;
I1 = ((Y-C)./(Ystar-Cstar)-1)*100;

// グラフ描写
figure(1)
subplot(2,2,1)
plot(0:50, A1(1:51)); title('A')
subplot(2,2,2)
plot(0:50, Y1(1:51)); title('Y')
subplot(2,2,3)
plot(0:50, C1(1:51)); title('C')
subplot(2,2,4)
plot(0:50, K1(1:51)); title('K')

figure(2)
subplot(2,2,1)
plot(0:50, L1(1:51)); title('L')
subplot(2,2,2)
plot(0:50, I1(1:51)); title('I')
subplot(2,2,3)
plot(0:50, w1(1:51)); title('w')
subplot(2,2,4)
plot(0:50, r1(1:51)); title('r')
```

プログラムの実行

あとは 2 節の要領で “RBC_det.mod” を実行すればよい。図 2 のようなインパルス応答の描写が得られる[*13]。

csv ファイルとして出力したければ、例えば

```
csvwrite('rbc_rslt.csv',[C, L, K, Y, w, r, A]);
```

とすればよい。このコマンドにより、全部の内生変数の水準の値が RBC_det.mod と同じフォルダに “rbc_rslt.csv” という名前で csv 形式で保存される。csv ファイルは、EXCEL で編集できる。

3.2 定常状態切り替えシミュレーションの例（Ramsey_tax2.mod）

次に、拙著 3 章の財政モデルのシミュレーションを mod ファイルとして記述してみよう。

3.2.1 準備するもの

3.1.1 節のボックス内に列挙したものと同じである。

1. 内生変数一覧、外生変数一覧

表 4、表 5 のようにそれぞれ与える。外生変数の初期値と終端値もここに掲げておく。

変数	名称	コード
C_t	消費	C
K_t	資本ストック	K

表 4　内生変数（財政モデル）

変数	名称	コード	初期値	終端値
$\tau_{c,t}$	消費税	tauc	0	0
$\tau_{k,t}$	資本所得税	tauk	0	0.1
g_t	政府支出	g	0.1	0.1

表 5　外生変数（財政モデル）

2. パラメーター一覧とその値

表 6 のように与える。

[*13] print('filename.eps','-depsc2')
というコマンドで、描写したグラフを保存できる。

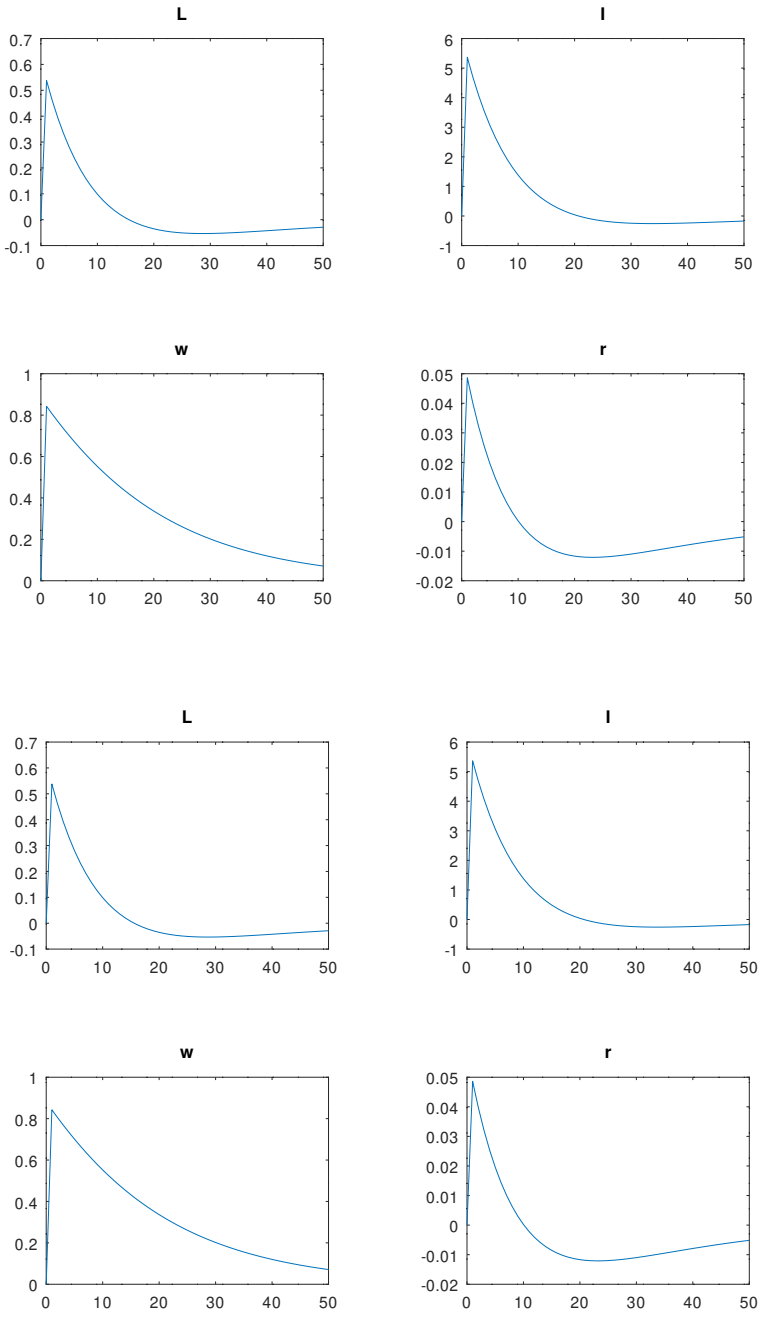

図 2　$e_1 = 0.01$ のショックに対するインパルス応答

変数	名称	コード	値
α	資本分配率	alpha	0.3
β	割引因子	beta	0.99
δ	減耗率	delta	0.25
A_t	生産性	At	1.0

表 6　パラメータ（財政モデル）

3. 方程式一覧

以下のように与えられている。

$$\frac{(1+\tau_{c,t+1})C_{t+1}}{(1+\tau_{c,t})C_t} = \beta\left[(1-\tau_{k,t+1})\alpha A_{t+1}K_{t+1}^{\alpha-1} - \delta + 1\right] \tag{17}$$

$$K_{t+1} = A_t K_t^{\alpha} + (1-\delta)K_t - C_t - g_t \tag{18}$$

状態変数（先決変数）K_t は $t+1$ 期の変数が含まれない形に書き換えなければならないが、ここでは K_{t+1} →K, K_t →K(−1), という置き換えで対処する。計算結果は、K→ K_{t+1} で読み替える。

4. 定常状態の求め方

モデルが小さいため Dynare に計算させる。したがって省略する。

5. シミュレーションのシナリオ

$t=0$ には定常状態にあったとし、$t=10$ から資本所得税率 $\tau_{k,t}$ が 0.1 になるというアナウンスが $t=1$ であったとする。消費税率 $\tau_{c,t}$ はゼロのまま据え置く。

3.2.2　プログラムの書き方（要点のみ）

上記の 1〜5 を元に、以下の順序で mod ファイル “Ramsey_tax2.mod” を記述していく。

1. 内生変数、外生変数の宣言（var ブロック、varexo ブロック）
2. パラメータの宣言（parameters ブロック）と値の代入
3. 方程式の設定（model ブロック）
4. 初期値の定義（initval ブロック）
5. 終端値（定常状態）の定義（endval ブロック、steady コマンド）

6. モデルのチェック（check コマンド）
7. シミュレーションの実行（perfect_foresight_setup, perfect_foresight_solver コマンド）とグラフ描写など

“3. 方程式の設定” までは 3.1 節の RBC モデルと全く同一なので省略する。

初期値の定義

この部分では、initval ブロックを使って初期値を定義する。シミュレーションのシナリオとして $t = 0$ では定常状態にあるものとするので、initval ブロックで直接パラメータ変化前の定常状態を与えてもいいし、直後に steady コマンドがあれば Dynare は initval ブロックで指定された初期値により定常状態を計算するので、そのように Dynare に計算させてもよい[*14] ここでは、Dynare に定常状態を計算させ、それを初期値とする。

初期値の定義として以下のように記述しておく。外生変数の初期値もここに記述する。

```
// 4. 初期値の定義
initval;
C = 1;
K = 1;
tauc = 0;
tauk = 0;
g = 0.1;
end;

steady;
```

終端値（定常状態）の定義

終端条件として、パラメータ変化後の定常値を与える。そのためには、改めて定常状態の計算過程を記述し、endval ブロックで Dynare に終端値を計算するための初期値を与え、steady コマンドで Dynare に定常状態を計算させる。外生変数の終端値もここに記述する。

[*14] ここでは「初期値」という言葉を二通りの意味で使っていることに注意。なお、3.1 節の例でもショックを与える前の $t = 0$ では必ずしも定常状態にある必要はない（脚注*11 も参照）。

```
// 5. 終端値（定常状態）の定義
endval;
C = 1;
K = 1;
tauc = 0;
tauk = 0.1;
g = 0.1;
end;

steady;
```

次いで、check コマンドでモデルをチェックするのが望ましい。方法は 3.1 節と全く同じであるため同様に省略する。

シミュレーションの実行とグラフ描写など

$t = 0$ には定常状態にあったとし、$t = 10$ から資本所得税率 $\tau_{k,t}$ が 0.1 になるというアナウンスが $t = 1$ であったというシナリオに対応するモデルの記述は以下のとおりである。

```
// 7. シミュレーションの実行
shocks;
var tauk; periods 1:9; values 0;
end;

perfect_foresight_setup(periods=31);
perfect_foresight_solver
```

3.3 対数線型化したモデルの例（NK_Linear.mod, NK_Linear_stoch.mod）

ここでは、拙著 5 章で紹介した対数線形化したニューケインジアン・モデルによるシミュレーションを Dynare で実行してみよう。

3.3.1 準備するもの

3.1.1 節のボックス内に列挙したものと同じだが、“4. 定常状態の求め方” は省略できる場合がある（今回は省略可能）。

1. 内生変数一覧、外生変数一覧

表 7、表 8 のようにそれぞれ与える。

変数	名称	コード
$\hat{x}_t$	GDP ギャップ	x
π_t	インフレ率	ppi
$\hat{a}_t$	技術水準	a
$\hat{i}_t$	名目金利	ii
v_t	金融政策ルールの誤差項	v

表 7　内生変数（ニューケインジアン・モデル）

変数	名称	コード
ε_t	技術ショック	e
z_t	金融政策ショック	z

表 8　外生変数（ニューケインジアン・モデル）

2. パラメーター一覧とその値

表 9 のように与える。

変数	名称	コード	値
β	割引因子	beta	0.99
γ	労働供給の弾力性の逆数	gamma	5
ϱ	価格改定できない確率	varrho	0.9
κ	$\kappa = \frac{(1-\varrho)(1-\varrho\beta)(\gamma+1)}{\varrho}$	kappa	–
ϕ_π	金融政策ルールの係数（インフレ率）	phi_pi	1.5
ϕ_y	金融政策ルールの係数（GDP ギャップ）	phi_y	0.5
ρ_A	AR(1) 項の係数（技術水準）	rho_A	0.9
ρ_v	AR(1) 項の係数（金融政策ルールの誤差項）	rho_v	0.7

表 9　パラメータ（ニューケインジアン・モデル）

3. 方程式一覧

以下のように与えられている（$\hat{r}_t^n$ は代入により削除）。

$$\pi_t = \beta\pi_{t+1} + \kappa\hat{x}_t \tag{19}$$

$$\hat{x}_t = \hat{x}_{t+1} - (\hat{i}_t - \pi_{t+1}) + (\rho_A - 1)\hat{a} \tag{20}$$

$$\hat{i}_t = \phi_\pi\pi_t + \phi_y\hat{x}_t + v_t \tag{21}$$

$$\hat{a}_{t+1} = \rho_A\hat{a}_t + \varepsilon_{t+1} \tag{22}$$

$$v_{t+1} = \rho_v v_t + z_{t+1} \tag{23}$$

4. **定常状態の求め方**

対数線形化後のモデル方程式に定常値が含まれないので、省略する[*15]。もちろん、対数線型化後の内生変数の定常状態での値は、全てゼロである。

5. **シミュレーションのシナリオ**

$t = 0$ で定常状態にあり、$t = 1$ で 1% の技術ショックが起こったものとする。

3.3.2 プログラムの書き方（要点のみ）

基本的には 3.1.2 節のボックス内に列挙した手順と同じだが、initval ブロックと steady コマンドは省略できる。また、シミュレーションする際、今までの perfect_foresight_solver コマンドに代えて stoch_simul コマンドも利用できる[*16]。それぞれ、上記のシナリオに対し、それぞれ以下のように記述する。

perfect_foresight_solver コマンドを使う場合

```
shocks;
var e; periods 1; values 0.01;
end;

perfect_foresight_setup(periods=150);
perfect_foresight_solver;
```

stoch_simul コマンドを使う場合（グラフ出力も同時に可能）

```
shocks;
var e = 1^2;
end;

stoch_simul(order=1, irf = 100) x ii ppi a v;
```

両者の違いは、今までのように完全予見解を求めるか、それとも行列を使っていわゆる合理的期待均衡解を求めるかにある（詳細については拙著 6 章を参照）[*17]。なお、**定常状態**

[*15] この場合はたまたま定常値が対数線形化の過程で消えてしまっただけであり、むしろ定常値が含まれていることのほうが普通である。

[*16] perfect_foresight_setup, perfect_foresight_solver コマンドは以前の simul コマンドに対応する。

[*17] 両者のコマンドは 1 つの mod ファイルで共存できない。したがって、必ずどちらかを選択する。perfect_foresight_solver コマンドでシミュレーションを行うのが NK_Linear.mod、stoch_simul コマンドでシミュレーションを行うのが NK_Linear_stoch.mod である。ただし、stoch_simul コマンドは

周りで対数線型化されたモデルでは、定常状態切り替え（パラメータ変更）シミュレーションが不可能であることに注意が必要である。

非線形のモデル（対数線型化する前のモデル）で stoch_simul コマンドを使ってシミュレーションを行う場合のプログラムの書き方については、RBC_stoch.mod を参考されたい[*18]。

3.4 Dynare のモデルの解き方のまとめ

上記で perfect_foresight_solver と stoch_simu という 2 つのシミュレーションのためのコマンドがあると述べた。前者 Perfect Foresight Simulation（完全予見シミュレーション）、後者は Stochastic Simulation（確率的シミュレーション）に対応する。その違いは以下の通りである。

本来誤差項に確率的なショックを与えるシミュレーションを行うためのものだが、ここでは 1 つのパスのみ計算させているので、完全予見で計算するのとほとんど違いがない。

[*18] モデルを記述する際、Dynare が実行する線型近似を意識して適宜変数変換する必要が生じる。

	Perfect Foresight Simulation	Stochastic Simulation	備考
コマンド	perfect_foresight_setup, perfect_foresight_solver	stoch_simul	両者は同じ mod ファイルで混在できない
モデルの対数線型化	どちらでもよい	要 *	* 非線形モデルを与えた場合、Dynare が自動的に線型化を行う
解き方	非線形の連立方程式を解く（ニュートン法など）	線型近似した方策関数によるモデルの再帰的表現（線型 VAR）*	*policy function の線型近似は Dynare 独自のアルゴリズムによる
終端条件	与える *	不要	* 終端条件は定常状態を与える
定常状態	初期条件の定常状態と終端条件の定常状態が異なっていてもよい	１つに固定	
パラメータ	任意に変更可能 *	固定	* 変更する場合は外生変数として扱う
シミュレーションの方法	外生変数（誤差項を含む）のパスを一意に与える	誤差項にショックを与える	
期待形成	完全予見	将来の誤差項のショックはゼロ	

4 Dynare によるパラメータ推定（NK_Linear_EST.mod）

対数線形近似した DSGE モデルとデータを用いることで、Dynare にパラメータをベイズ推定させることができる[*19]。具体的には、モデルとデータ、それと未知パラメータ

[*19] 非線形のモデル（対数線型化する前のモデル）を与え、Dynare に線型近似させることによってもパラメータ推定可能だが、予め対数線形近似したモデルを用意したほうが計算時間が短くてすむ。

の事前分布を与えることで、Dynare はデータにフィットするようなパラメータの事後分布を MCMC 法（M-H アルゴリズム）により求めてくれる。この節では、その一連の手順を説明する。

4.1 何を準備すればよいか

解きたい DSGE モデルに合わせて、以下を準備する。

1. モデル方程式、観測方程式
2. 内生変数一覧、誤差項一覧
3. 既知のパラメータの値
4. 未知のパラメータ一覧（誤差項の標準偏差を含む）と事前分布
5. データ
6. 事後分布のサンプリングの条件

1. モデル方程式、観測方程式

以下のような対数線型化したニューケインジアン・モデル

$$\hat{x}_t = \hat{x}_{t+1} - (\hat{i}_t - \pi_{t+1}) + (\rho_A - 1)\hat{a}_t \tag{24}$$
$$\pi_t = \beta\pi_{t+1} + \kappa\hat{x}_t + e_t \tag{25}$$
$$\hat{i}_t = (1 + \tilde{\phi}_\pi)\pi_t + \phi_y\hat{x}_t + v_t \tag{26}$$
$$v_{t+1} = \rho_v v_t + z_{t+1} \tag{27}$$
$$\hat{a}_{t+1} = \rho_A\hat{a}_t + \varepsilon_{t+1} \tag{28}$$

と、観測方程式

$$x_t^{obs} - \bar{x}^{obs} = \hat{x}_t \tag{29}$$
$$\pi_t^{obs} - \bar{\pi}^{obs} = \pi_t \tag{30}$$
$$(i_t^{obs} - \bar{i}^{obs})/4 = \hat{i}_t \tag{31}$$

を例として、具体的に説明する[*20]。バー付の変数は、期間平均を意味する。

2. 内生変数一覧、誤差項一覧

以下の通り。

[*20] 拙著 8 章の例と同一である。詳しくはそちらを参照。

変数	名称	コード
$\hat{x}_t$	GDP ギャップ	x
π_t	インフレ率	ppi
$\hat{a}_t$	技術水準	a
$\hat{i}_t$	名目金利	ii
v_t	金融政策ルールの誤差項	v

表 10　内生変数（推定用ニューケインジアン・モデル）

変数	名称	コード
ε_t	技術ショック	epsilon
z_t	金融政策ショック	z
e_t	コストプッシュ・ショック	e

表 11　誤差項（推定用ニューケインジアン・モデル）

3. 既知のパラメータの値

割引因子 β（beta）は 0.99 と予め設定しておく。

4. 未知のパラメーター覧（誤差項の標準偏差を含む）と事前分布

以下の通り。

パラメータ	確率分布の種類	平均	標準偏差	
γ	ガンマ分布	5.0	2.0	労働供給の弾力性の逆数
ϱ	ベータ分布	0.9	0.05	価格改定できない確率
$\tilde{\phi}_\pi$	ガンマ分布	0.5	0.25	金融政策ルールの係数（インフレ率）
ϕ_y	ガンマ分布	0.5	0.25	金融政策ルールの係数（GDP ギャップ）
ρ_A	ベータ分布	0.9	0.05	AR(1) 項の係数（技術水準）
ρ_v	ベータ分布	0.7	0.1	AR(1) 項の係数（金融政策ルールの誤差項）
σ_ε	逆ガンマ分布	5.0	1.0	技術ショック ε_t の標準偏差
σ_z	逆ガンマ分布	2.0	1.0	金融政策ショック z_t の標準偏差
σ_e	逆ガンマ分布	1.0	0.5	コストプッシュ・ショック e_t の標準偏差

表 12　パラメータの事前分布

5. データ

データとして、GDP ギャップ（x_t^{obs}）、GDP デフレーター（季節調整値、前期比; π_t^{obs}）、短期金利（i_t^{obs}）の 3 系列を用いる。

6. **事後分布のサンプリングの条件**

MCMC 法によるチェインごとのサンプリングの回数、チェインの数、定常状態に至っていないものとみなして捨てる割合を予め決めておく、ここでは、それぞれサンプリング 20,000 回、チェイン数 2、捨てる割合 0.25 とする[*21]。

4.2 データの前処理

観測方程式の左辺を

$$x_t^{act} = x_t^{obs} - \bar{x}^{obs} \tag{32}$$

$$\pi_t^{act} = \pi_t^{obs} - \bar{\pi}^{obs} \tag{33}$$

$$i_t^{act} = (i_t^{obs} - \bar{i}^{obs})/4 \tag{34}$$

のように別の変数 x_t^{act}, π_t^{act}, i_t^{act} で定義しなおしておくと、mod ファイルの記述が簡単になる。

データの前処理の具体的な方法については、script_dataset.m を参照。

[*21] ここでの “定常状態” とは、マルコフチェインのそれである。DSGE モデルの定常状態と混同しないこと。

script_dataset.m

```
FqReport8 = csvread('FqReport8.csv', 1, 1);

GAP0 = FqReport8(:,1);
PGDP0 = FqReport8(:,2);
IRS0 = FqReport8(:,3);

PC_PGDP0 = [NaN; (PGDP0(2:end)./PGDP0(1:end-1)-1)*100];

tt = 2:77;
MGAP = mean(GAP0(tt));
MPC_PGDP = mean(PC_PGDP0(tt));
MIRS = mean(IRS0(tt));

GAP = GAP0(tt)-MGAP;
PC_PGDP = PC_PGDP0(tt)-MPC_PGDP;
IRS4 = (IRS0(tt)-MIRS)*0.25;

x = GAP;
ppi = PC_PGDP;
ii = IRS4;

save dset.mat x ppi ii;
```

ここでは csv ファイル（FqReport8.csv）から数値のみ読み込み、適宜変数名を与えている。GDP デフレーターは元がレベルデータなので、最初にそれを前期比にしている。あとは、(32)〜(34) 式のような加工を行い、最後に x_t^{act}, π_t^{act}, i_t^{act} それぞれを x, ppi, ii という系列名で “dset.mat” という mat ファイル（MATLAB データ形式）に保存している。この系列名は、mod ファイルを作成する際に用いるので覚えておく。

4.3 プログラム（mod ファイル）の書き方

上記を元に、以下の順序で mod ファイル “NK_Linear_EST.mod” を記述していく。

1. 内生変数、外生変数の宣言（var ブロック、varexo ブロック）
2. パラメータの宣言（parameters ブロック）と値の既知パラメータの代入

3. 方程式の設定（model ブロック）
4. 事前分布の設定（estimated_params ブロック）
5. 観測データの指定（varobs ブロック）
6. 事後分布の推定（estimation コマンド）

1. 内生変数、外生変数の宣言

3 節のシミュレーションの場合と基本的に同じである。

```
var x ppi a ii v;
varexo epsilon z e;
```

ppiact については下記の 3. 方程式の設定を参照。

2. パラメータの宣言と値の既知パラメータの代入

同じく、3 節のシミュレーションの場合と同じだが、既知パラメータの値のみ代入すればよい。

```
// 2. パラメータの宣言
parameters beta gamma varrho phi_pi phi_y rho_A rho_v;

// パラメータ値の代入
beta = 0.99;
```

3. 方程式の設定

κ（kappa）は他のパラメータの関数になることから、#記号を使って model ブロック内に定義を書く。したがって、以下のように記述する。

```
model(linear);
# kappa = (1-varrho)*(1-varrho*beta)*(gamma+1)/varrho;
x = x(+1)-(ii-ppi(+1))+(rho_A-1)*a;
ppi = beta*ppi(+1) + kappa*x + e;
ii = (1+phi_pi)*ppi + phi_y*x + v;
v = rho_v*v(-1) + z;
a = rho_A*a(-1) + epsilon;
end;
```

モデル方程式の変数のうち $\hat{x}_t$, π_t, $\hat{i}_t$ がデータと一対一で対応する。これらの mod ファイル上での変数名 x, ppi, ii は、データを保存した “dset.mat” の中に同じ系列名で保存されていることに注意。

4. 事前分布の設定

表 12 を元に、以下のように記述する。

```
// 4. 事前分布の設定
estimated_params;
gamma, gamma_pdf, 5, 2;
varrho, beta_pdf, 0.9, 0.05;
phi_pi, gamma_pdf, 0.5, 0.25;
phi_y, gamma_pdf, 0.5, 0.25;
rho_A, beta_pdf, 0.9, 0.05;
rho_v, beta_pdf, 0.7, 0.1;
stderr epsilon, inv_gamma_pdf, 5, 1;
stderr z, inv_gamma_pdf, 2, 1;
stderr e, inv_gamma_pdf, 1, 0.5;
end;
```

5. 観測データの宣言

3. 方程式の設定で説明したように、$\hat{x}_t$, π_t, $\hat{i}_t$ がデータと一対一で対応するので、その変数名を宣言する。

```
// 5．観測データの宣言
varobs x ppi ii;
```

6. 事後分布の推定

推定には、estimation コマンドを用いる。それぞれ、datafile にデータを保存したファイルの（拡張子を除く）ファイル名 dset、mh_replic に MCMC 法によるチェインごとのサンプリングの回数 20,000、mh_nblocks にチェインの数 2、mh_drop に定常状態に至っていないものとみなして捨てる割合 0.25 を指定している。

最後の **mh_jscale に指定する値は最も重要であり、M-H アルゴリズム（酔歩連鎖）の採択率（acceptation rate）が 0.2〜0.5 の間に収まるように試行錯誤して適切に設定する。**一般に、この値を大きくすることで採択率が減少し、小さくすることで採択率が増加する。この例では、0.6 とすることで約 0.35 の採択率になる。

```
// 6．事後分布の推定
estimation(datafile='dset.mat', mh_replic=20000, mh_drop = 0.25,
mh_nblocks=2, mh_jscale=0.6, mode_compute = 4, mode_check);
```

4.4 推定の実行と推定結果

データを保存した mat ファイルさえ mod ファイルと同じフォルダに準備できていれば、シミュレーションの場合と同じく 2 節の要領で実行できる。乱数を用いるため実行ごとに結果が変わる（可能性がある）が、今回は以下のような推定結果を得た。

```
ESTIMATION RESULTS

Log data density is -152.657434.

parameters
          prior mean   post. mean        90% HPD interval    prior       pstdev

gamma          5.000       4.4640      1.6297      7.2371   gamm        2.0000
varrho         0.900       0.9403      0.9188      0.9619   beta        0.0500
phi_pi         0.500       0.0999      0.0240      0.1721   gamm        0.2500
phi_y          0.500       3.1554      2.4964      3.8466   gamm        0.2500
rho_A          0.900       0.9622      0.9488      0.9766   beta        0.0500
rho_v          0.700       0.8668      0.8283      0.9030   beta        0.1000

standard deviation of shocks
          prior mean   post. mean        90% HPD interval    prior       pstdev

epsilon        5.000       5.0107      3.5576      6.4576   invg        1.0000
z              2.000       2.3712      1.7262      2.9101   invg        1.0000
e              1.000       0.4928      0.4308      0.5585   invg        0.5000
```