Introducción Matlab y Dynare:

Talleres Modelos RBC & NK

CAEP - Banco de la República

Juan Andrés Rincón Galvis

27 de junio de 2025

¿Cómo Resolver un Modelo DSGE?

- ¿Cómo resolver un modelo DSGE, cómo simularlo?

  A. Con papel y lápiz: usando a mano formas funcionales específicas para las $f(\cdot)$ de utilidad y producción

  B. Escribiendo su propio código en el computador

  C. Usando: Dynare + Matlab ó Dynare + Octave

Agenda

Lenguaje Básico de Matlab

Introducción a la Programación en Matlab

Usando Dynare

Taller Modelo RBC

Taller Modelo NK

Lenguaje Básico de Matlab

Consejos Iniciales

- Existen dos comandos que siempre serán útiles y que se escriben en la Ventana de Comando:
  - Obtener información y ejemplos de una función fun:

    help fun
  - Indagar sobre la sintaxis y todas las opciones de una función fun:

    doc fun
- En el Editor se escriben los códigos y las funciones. El " %" sirve para realizar comentarios.
- Es mejor utilizar "**;**" para evitar que una operación escrita en el Editor se haga en la Ventana de Comando, para mayor velocidad.
- Matlab es sensible a las mayúsculas y minúsculas, por lo que hay que ser cuidadoso con ello. En lugar de espacios, se usa "_"

Comandos Iniciales

Antes de empezar a trabajar en Matlab uno normalmente quiere:

- Limpiar su workspace (borrar variables y datos ya usados):
  - Para borrar todo:

    clear all
  - Para borrar la variable x:

    clear x
- Despejar la Ventana de Comando:

  clc
- Cerrar gráficas que se hayan creado antes:

  close all
- Definir un punto en el que un código en el Editor se va a detener:

  return;

Comandos Iniciales

Algunos comandos para actuar desde la Ventana de Comando:

- Ver los valores de una variable (matriz, vector, base):

  open x

- Abrir un determinado código o función "code.m":

  edit code.m

- Correr el código "code.m" desde la Ventana de Comando:

  code

- Matlab sobreescribe las variables, es decir, se queda siempre con el último valor asignado en una rutina:

$$x = 2; \qquad x = 5;$$

Comandos Iniciales

- Verificar qué variables y estructuras hay en el workspace:

  whos

- Comprobar cuáles son los archivos que hay en el directorio:
  - Con una determinada extensión (.m , .mod, etc.), :

    dir *.m

  - Con un mismo nombre, pero diferentes extensiones:

    dir nombre.*

- Saber cuáles son las dimensiones de una variable x:

  size(x)

- Indagar por el tamaño de una variable x en la dimensión **j**:

  size(x, **j**)

- Averiguar cuál es el tamaño de la dimensión más grande de una variable x:

  length(x)

Operaciones básicas

- Crear un escalar con un valor específico:

x=5; y=7;

Operaciones entre escalares:

- Suma y resta: z=x+y; t=x-y;
- Mutiplicación y División: u=x*y; v=x/y;
- Potencia: $u = x \wedge y$;
- Raíz Cuadrada: u=**sqrt**(x);
- Exponencial: v=**exp**(y);
- Logaritmo natural: v=**log**(x);
- Logaritmo en base jj: y=**logjj**(u);

Para jerarquizar operaciones sólo se utilizan los paréntesis:

w=(u-x+(z+x)*u)/y

Operaciones básicas

Distintas formas de crear un vector:

- Un vector de ceros de 3x1:

$$\text{x\_vec} = \textbf{zeros}(3,1);$$

- Un vector de unos de 10x1:

$$\text{y\_vec} = \textbf{ones}(10,1);$$

- Un vector horizontal con incrementos de 2:

$$w1 = 1 : 2 : 11;$$

- Un vector horizontal de 1 a 10 de 19 elementos equi-espaciados:

$$w2 = \textbf{linspace}(1, 10, 19)$$

- Un vector aleatorio de 100 elementos de normal estándar:

$$v1 = \textbf{randn}(100, 1)$$

- Un vector aleatorio de 100 elementos de una normal con media k y varianza g:

$$\text{v2} = \textbf{normrnd}(\text{k}, \text{g}, 100, 1)$$

Operaciones básicas

Distintas formas de crear una matriz:

- Matriz cuadrada de ceros de tamaño 5:

  X = **zeros**(5);

- Matriz de ceros de 3x4:

  X = **zeros**(3,4);

- Matriz identidad de tamaño 6:

  Y = **eye**(6);

- Matriz de números aleatorios:

  V = **randn**(40,60); V = **normrnd**(10,5,40,60);

- Matriz con valores puntuales, donde el ";"separa filas y la "," ó el espacio las columnas:

  A = [1 2 8 ; 9 3 4]; B = [2, 6, 7 ; 5, 2, 1];

Operaciones básicas

Operaciones clave entre matrices y vectores:

- Matriz transpuesta: E = A';
- Suma, resta y multiplicación: F =A + B; G =A - B; IH = E * B;
- Matriz inversa: F = **inv**(IH);
- 'División' entre matrices:
  En este caso se invertirá la matriz hacia donde está inclinado el *slash*:
  (Matlab recomienda usar esto, en lugar de la función inversa)

  $A = randn(3,3);$ $B = randn(3,3) + 40$ $L1 = A\text{“}B;$ $L2 = \mathbf{inv}(A) * B;$ $L3 = A/B;$
  $L4 = A * \mathbf{inv}(B);$
- Matriz al cuadrado: $AA = A \wedge 2;$
- Raíz cudrada: Q = **sqrtm**(A)

Operaciones básicas

Manipulando los elementos de una matriz o vector:

- Seleccionar un elemento (por columnas): N1=A(4)
- Indicar la coordenada exacta, fila 2 y columna 3: N2=A(2,3)
- Tomar un segmento de la matriz: $R3 = A(2 : end, 1 : 2)$
- Extraer toda una fila: $R1 = A(1, :)$
- Extraer toda una columna: $R2 = A(:, 2)$
- Cambiar un dato puntual: $R1(1, 1) = 9$
- Cambiar toda una fila: $R2(1, :) = R1(3, :)$
- Reemplazar el contenido de una posición puntual con cero:
$$A(1, 2) = 0$$
- Reemplazar el contenido de toda una columna con ceros:
$$A(:, 2) = \mathbf{zeros}(\mathbf{size}(A(:, 2)))$$
- Borrar toda una columna: $A(:, 2) = [\ ];$
- Otra función para eliminar datos sin poner ceros:
$$A(:, 2) = \mathbf{NaN}(\mathbf{size}(A(:, 2)))$$

Operaciones básicas

Operaciones elemento a elemento:

- Multiplicación: J = A .* B;
- "División": M = A ./ B;
- Potencia: P = $A. \wedge 2$;

Algunas operaciones adicionales:

- Repetir una matriz en la dimensión de las columnas:

  T = **repmat**(P, 1 ,4)

- Repetir una matriz en la dimensión de las filas:

  T = **repmat**(P, 4, 1)

- Cambiar las dimensiones de una matriz de 3x3 a un vector, 9x1:

  S = **reshape**(B, 9, 1)

- Crear una matriz a partir de otras matrices o vectores:

  G = [A B]; K=[1:10 31:40];

Cell Arrays

- Matlab permite guardar información en forma de cell arrays.
- En estas celdas se pueden albergar escalares, vectores, matrices y texto.
- Un array de celdas se crea de dos formas:
  - Creándolo enteramente vacío: C = { };
  - Formando un array vacío de un tamaño dado: C =**cell**(2,2);
- Un ejemplo de cómo se puede albergar información distinta en un mismo array de celdas:

$$C\hat{}1,1\text{‘} = A; \qquad C\hat{}1,2\text{‘} = 5;$$
$$C\hat{}2,1\text{‘} = \text{''}Hola\text{''}; \qquad C\hat{}2,2\text{‘} = 2:3:41;$$

- Con paréntesis se extrae lo que guardó en esa posición:

$$C(1,2)$$

- Pero con las llaves se obtiene el contenido de la celda como tal:

$$C\hat{}2,1\text{‘}$$

Data Structures

- Las estructuras permiten guardar todo tipo de datos y variables, y además asignarle un nombre a cada objeto guardado.
- Una estrucura se crea así:

data=**struct**();

- Cada que se desee agregar un objeto, se le asigna el nombre:

data.matrizA = A; data.matrizB = B;

- Para obtener un array con los nombres de los campos de la estructura:

names_data = **fieldnames**(data);

- Extraer los valores guardados en un campo dentro de la estructura:

mat_A = data.matrizA; mat_B = **getfield**(data, 'matrizB')

- Campos anidados dentro de la estructura:

data.cellC = C; hola= data.cellC{2,1}

new_data.data = data;

C_2 = new_data.data.C

Guardar y Cargar Archivos

- Vamos a especificar dos matrices, un vector y una estructura:

Mat_A = [2 54; 68 73]; Mat_B = [192 23; 18 NaN];

Vec_C = linspace(2,30,15); all_stru = **struct**();

all_stru.A = Mat_A; all_stru.B = Mat_B; all_stru.C = Vec_C

- Guardemos esta información en un **.mat** de nombre guarda que es un tipo de archivo en el que Matlab guarda resultados y datos:

save guarda Mat_A Mat_B Vec_C

Guardar y Cargar Archivos

- Otra alternativa es guardar directamente la estructura, ya que alberga a las otras tres en una sola variable.

save guarda_struct all_stru; **clear all**

- Como limpiamos el workspace, podremos recuperar la información si hacemos lo siguiente:

load guarda_struct

- Si sólo estuviéramos interesados en recuperar la matriz A, podríamos hacer:

load guarda Mat_A

Gráficas

- Para abrir una ventana propicia para graficar:

figure()

- La ventana se puede enumerar o nombrar:

figure(1) **figure**('Name','Gráfica')

- Luego se utiliza la función que grafica:

y = **randn**(80, 1)+12;

plot(y)

- Si se trata de un vector la función graficará los valores contra un vector de igual tamaño (ej. de 1 a 80).
- En el caso de una matriz, se graficará cada columna contra el índice de la fila de cada dato.

Gráficas

- Para graficar una fila particular de la matriz, se puede:

plot(A(1,:)')

- Si se van a extraer de matrices distintas dos series para graficarlas una sobre la otra:

plot([A(1,:)' B(3,:)'])

plot(A(1,:)') **hold on**
plot(B(3,:)')

- En el caso de que los datos estén en una estructura:

x = **normrnd**(-4, 7.5, 80,1); data.x= x; data.y= y;
plot(data.x, data.y)

- La información y propiedades del gráfico pueden guardarse para luego ser modificadas, así:

graf1 = **plot**(data.x, data.y)

Gráficas

- Agregarle nombre a los ejes:

  ylabel('eje y') **xlabel**('eje x')

- Incluir leyendas para las líneas graficadas:

  plot([data.x' data.y']) **legend**('x rand','y rand')

- Ponerle un título al gráfico:

  title('Aleatorios')

- Hacer visible la cuadrícula en el gráfico:

  grid on;

- Añadir una línea de referencia para la media de y:

  eje_x = **refline**(0,12);
  eje_x.**Color** = 'r+-';

  donde 0 es la pendiente y 12 el intercepto.

- Mostrar la última gráfica generada y que aún no se ha cerrado:

  shg

Introducción a la Programación en Matlab

Indexación

En Matlab es posible seleccionar cualquier sección de un arreglo.

- Elegir de un vector todos menos el último dato:

```
V = 1:2:20;      v_1 = V(1:end-1);
```

- Elegir de un vector todos menos el primer dato:

```
v_2 = V(2:end)
```

- Elegir de la columna de una matriz j filas menos desde el final:

```
M = normrnd(2, 1, 10, 10) ;      m_col = M(1:end-j, 5)
```

- Elegir de la fila de una matriz a partir de la columna k hasta el final:

```
m_row = M(8, k:end)
```

- Elegir de la fila de una matriz a partir de la columna k hasta el final:

```
m_row = M(8, k:end)
```

- Elegir con un vector las filas que se quieren de una matriz:

```
index = [1 2 7 9];       M_rows = M(index,:)
```

Función find

Una de las funciones más útiles en programación es **find**.

- Por default, la función encuentra las posiciones de los valores distintos de cero en un arreglo:

```
V = -3:1:3;        V_nzero = find(V)
M = [0 1 2; 3 0 1; 0 0 5];      M_nzero = find(M)
```

- Pero también puede usarse para condiciones simples como:

```
index_V = find(V<0);     V_neg = V(index_V)
```

- Encontrar posiciones con valores puntuales:

```
index_V = find(V==3);     V_3 = V(index_V)
```

- Encontrar posiciones donde haya NaN:

```
AA = [3 NaN 6 8 2 NaN 4 -1 6 -4 2 -9 NaN]';
index_AA = find(isnan(AA));     AA_NaN = AA(index_AA)
```

- También pueden excluirse dichos NaN:

```
index_AA = find(isfinite(AA));     AA_noNaN = AA(index_AA)
```

Condicionales

Especifiquemos primero estas matrices:

```
X=[1 2 5 7 8 9 10 25; 3 8 7 2 345 252 644 9];
Y=[1 3 5 7 7 9 9 25; 3 8 7 2 344 252 646 8];
```

- Ahora, queremos reemplazar el último valor de X por 11 si éste es menor o igual que diez:
  ```
  if X(end,end) <= 10
  X(end,end) = 11;
  end
  ```
- Queremos elevar al cuadrado los valores de X iguales a 5 solo si el valor anterior es 7:
  ```
  ind = find(X==5)
  if X(ind-1) == 2
  ```
  $X(ind) = X(ind) \wedge 2;$
  ```
  end
  ```

Loops

- Elevar al cuadrado todo valor en la primera fila de Y que coincida en esa posición con el de X:

```
for i = 1:size(Y,2)
  if Y(1,i) == X(1,i)
   Y(1,i) = Y(1,i)^ 2;
  end
end
```

- Si el valor no coincide reemplace con NaN:

```
for i = 1:size(Y,2)
  if Y(1,i) == X(1,i)
   Y(1,i) = Y(1,i)^ 2;
  else
   Y(1,i) = NaN;
  end
end
```

Loops

Recuperemos las matrices **X** y **Y** originales para hacer otro Loop:

```
for i = 1:size(X,1)
  for j = 1:size(X,2)
    if mod(X(i,j), 2) == 0
      c1(i,j) = X(i,j);
    end
    if X(i,j) == Y(i,j)
      c2(i,j) = 1;
    elseif X(i,j) > Y(i,j)
      c3(i,j) = 1;
    elseif X(i,j) ≅ Y(i,j)
      c4(i,j) = 1;
    else
      c4(i,j) = 2;
    end
  end
end
```

Loops

- Algunas otras funciones que resultan ser muy útiles para programar son:
  1. El statement **while** que sirve como un loop, pero este no itera a lo largo de un vector predeterminado, sino que repetirá una misma operación hasta que una condición dada de convergencia se alcance.

     help while

  2. El statement **switch** funciona a través de una expresión. Lo que hace es tomar una variable o vector y según el valor que ella tenga en ese momento se ubicará en un **case**, en el que habrá una o varias acciones que deberá ejecutar.

     help switch

- Al igual que con los statements **for** y el **if**, los dos anteriores también pueden combinarse.
- De hecho, es común encontrar que haya un statement de Loop y dentro de él haya un **switch** o un **if**.
- Este tipo de combinaciones entre Loops y Condicionales se denominan bloques de control de flujos ("Flow Control").

Función de Minimización

- Otra función muy importante es la de **fminsearch**, que busca alcanzar un mínimo global.
- Esta resuelve problemas de optimización no restringida, no lineal y multivariada.
- Primero, se debe definir el rango en el que se van a mover los valores de los inputs de la función objetivo:

$$\text{x = (3:10);}$$

- Segundo, se construye la función objetivo:

$$f = @(x)(x - 2).\wedge 2;$$

- Tercero, se utiliza la minimización:

$$[\text{x\_sol, min\_f, exitflag, output}] = \textbf{fminsearch}(\text{f,3})$$

- El primer argumento es el valor que minimiza la función objetivo y el segundo es el valor mínimo de la función.
- Finalmente, podemos realizar la gráfica de la función objetivo.

$$\textbf{fplot}(\text{f});$$

Usando Dynare

¿Qué es Dynare?

- Dynare es una plataforma de software gratis para manejar y trabajar en una amplia cantidad de modelos económicos, en particular modelos Dynamic Stochastic General Equilubrium Dynare (DSGEs) y modelos de generaciones traslapadas (OLG).
- Puntualmente, es un toolbox adicional que se puede correr en Matlab u Octave, pero que no es creado por Matlab.
- Toolbox externo: Desarrollado por un equipo liderado por Michel Julliard.
- Consiste en una serie de rutinas y funciones que están almacenadas en una carpeta que se descarga al computador.
- Ahora utilizado ampliamente en instituciones académicas

Inicializando Dynare

- Descargar Matlab u Octave.
- Descargar Dynare de http://www.dynare.org/
- Lo que se descarga es una carpeta con las rutinas y funciones que pueden correr Matlab u Octave.
- Esta carpeta debe agregarse al path de Matlab/Octave, para que el programa pueda encontrar los códigos de Dynare a medida que los va necesitando. Así:

  addpath C:\ruta\dynare\4.5.6 \matlab
- De este modo, Matlab ya tiene dentro de su acervo de funciones, aquellas que corresponden a Dynare y que van a permitir traducir los modelos del lenguaje "matemático/teórico" al de Matlab.
- El modelo a simular se escribe en un archivo ".mod ó .dyn". Por ejemplo: **my_mode.mod**

Orden usual del modelo

- Se enumeran las variables (mayúsculas y espacios siguen las mismas reglas de Matlab):

var y i k c z ;

- Luego se declaran las variables exógenas:

varexo epsilon ;

- Los parámetros y sus valores:

parameters betta deltta ;

$betta = 0.99; \quad deltta = 0.05;$

- Vienen luego las ecuaciones y hay que escribirlas respetando los nombres puestos: **¿Ven los tres errores?**

model;

$y = i + c$;

$K = (1 - delta)*k(-1) + i$;

$z = 0{,}9*z(-1) + epsilon$;

end;

Orden usual del modelo

- Se enumeran las variables (mayúsculas y espacios siguen las mismas reglas de Matlab):

initval;
k = 24;
c = 1.33;
y = 1;
end;

- Ahora, se asignan las varianzas de las innovaciones exógenas:

shocks;
var epsilon; stderr 0.1;
end;

- Finalmente, uno quiere solucionar y simular el modelo:

resid; → Hallar residuales del modelo estático
steady;→ Calcular el estado estacionario
check;→ Verificar estabilidad de solución
stoch_simul(order = 1, irf=40);→ Solucionar y simular IRFs

Dynare y su Output

- Dynare se puede correr directamente desde un código, así:

  dynare model.mod

- Usualmente, cuando uno lo corre por fuera de Dynare, uno sólo quiere la solución y otras estructuras que ofrece como resultado, por lo que uno agrega estas opciones:

  dynare model.mod nograph nolog noclearall

- Tras solucionar el modelo, Dynare crea varios archivos, pero dos son de especial interés:
  1. El modelo después del pre-procesamiento se contruye en el lenguaje de Matlab en el archivo:

     model.m

  2. Guarda información diversa sobre el modelo y sus resultados en el workspace y en:

     model_results.mat

Resultados del Modelo

- El problema con la información que se guarda en el workspace es que ante un **clear all** ya no estará disponible:

load model.mat

- Al cargar de nuevo estos resultados encontraremos tres estructuras fundamentales:
  1. Estructura que guarda información sobre el modelo: **M_**
  2. Contiene todas las opciones escogidas a lo largo del .mod: **options_**
  3. Contiene los resultados de la solución y otros cómputos hechos sobre el modelo por Dynare: **oo_**
- De este modo, si hemos corrido diversos modelos podremos información sobre ellos cargando los **.mat** correspondientes.
- Si corremos varias veces el mismo modelo con el mismo nombre, pero cada vez le hacemos algún cambio, los resultados y demás información se van a sobreescribir.

Resultados del Modelo

- En la estructura **M_** se puede encontrar:
  - Los nombres de los parámetros: **M_**.param_names
  - Los nombres de las variables endógenas: **M_**.endo_names
  - El número de variables forward-looking: **M_**.nfwrd
  - El número de variables predeterminadas: **M_**.npred
  - Los valores de los parámetros: **M_**.params
  - La matriz de varianzas y covarianzas: **M_**.Sigma_e
- El órden de esta información es el mismo orden de declaración, a no ser que se diga lo contrario en algún caso particular.
- Los nombres de los parámetros en **param_names**, por ejemplo, pueden usarse para encontrar la posición de un parámetro puntual y saber luego el valor que tiene en **params**.

Resultados del Modelo

- La otra estructura esencial es **oo_** que alberga:
  - La información sobre las reglas de decisión en: **oo_**.dr
  - La Policy F(.) del modelo: **oo_**.dr.ghx, donde las columnas son las variables de estado.
  - La matriz de coeficientes de los choques: **oo_**.dr.ghu, donde las columnas son las variables exógenas.
  - El orden de las variables en Policy F(.): **oo_**.dr.order_var
  - Los resultados de los impulso-respuesta: **oo_**.irfs
  - Los valores de estado estacionario: **oo_**.steady_state
- En las matrices que solucionan el modelo, las filas corresponden a todas las variables endógenas del modelo.
- Acá, por ejemplo, puede usarse la lista de variables endógenas obtenida en **M_** para obtener la posición de una variable e indagar sobre su estado estacionario.

Cambiar Parametrización

- Un ejercicio usual es alterar los parámetros del modelo y ver eso cómo modifica los impulso-respuesta o incluso, el estado estacionario.
- En este sentido, uno puede escribir un código en Matlab que utilice Loops para probar diferentes valores para un mismo parámetro.
- Luego usar cada uno de esos valores para modificar el que tiene el modelo en el **.mod**.
- Correr el modelo, extraer los resultados de interés y luego al final compararlos gráficamente.
- Un comando clave de Dynare que se escribe dentro del **.mod** para modificar estos parámetros es:

load new_params.mat;
set_param_value('*nombre*', nuevo_valor);

- Con él, cada vez que el Dynare corra, antes de solucionar y simular, buscará el número guardado como nuevo_valor en new_params.mat y se lo asignará al modelo.

Taller Modelo RBC

El modelo RBC

Suponga $\sigma = 1$

$$max_{\tilde{c}_t, \tilde{k}_{t+1}, n_t} \mathbb{E}_0 \left\{ \sum_{t=0}^{\infty} [\beta(1+\eta)]^t [(1-\alpha) log(\tilde{c}_t) + \alpha log(1-n_t)] \right\}$$

$$s.t$$

$$\tilde{c}_t + \tilde{i}_t = \tilde{y}_t$$

$$\tilde{y} = e^{z_t} \tilde{k}_t^{\theta} n_t^{1-\theta}$$

$$(1+\gamma)(1+\eta)\tilde{k}_{t+1} = (1-\delta)\tilde{k}_t + \tilde{i}_t$$

$$z_{t+1} = \rho z_t + \varepsilon_{t+1}, \varepsilon \sim \mathcal{N}(0, \sigma_{\varepsilon}^2)$$

Modelo RBC

- Re-escriba el modelo

$$max_{\tilde{c}_t, \tilde{k}_{t+1}, n_t} \mathbb{E}_0 \left\{ \sum_{t=0}^{\infty} [\beta(1+\eta)]^t [(1-\alpha) log(\tilde{c}_t) + \alpha log(1-n_t)] \right\}$$

$$s.t$$

$$\tilde{c}_t + (1+\gamma)(1+\eta)\tilde{k}_{t+1} - (1-\delta)\tilde{k}_t = e^{z_t} \tilde{k}_t^{\theta} n_t^{1-\theta}$$

$$z_{t+1} = \rho z_t + \varepsilon_{t+1}, \varepsilon \sim \mathcal{N}(0, \sigma_{\varepsilon}^2)$$

Condiciones de Equilibrio

$$\frac{\alpha}{1-\alpha}\left(\frac{\tilde{c}_t}{1-n_t}\right) = (1-\theta)e^{z_t}\tilde{k}_t^{\theta} n_t^{-\theta} \tag{1}$$

$$\left(\frac{1}{\tilde{c}_t}\right) = \frac{\beta}{1+\gamma}\mathbb{E}\left[\left(\frac{1}{\tilde{c}_t}+1\right)\left(\theta e^{z_{t+1}}\tilde{k}_{t+1}^{\theta-1} n_{t+1}^{1-\theta}+1-\delta\right)\right] \tag{2}$$

$$(1+\gamma)(1+\eta)\tilde{k}_{t+1} = (1-\delta)\tilde{k}_t + \tilde{i}_t \tag{3}$$

$$\tilde{c}_t + \tilde{i}_t = \tilde{y}_t \tag{4}$$

$$\tilde{y}_t = e^{z_t}\tilde{k}_t^{\theta} n_t^{1-\theta} \tag{5}$$

$$z_{t+1} = \rho z_t + \varepsilon_{t+1}, \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2) \tag{6}$$

Dadas las variables endógenas en estas expresiones, otras variables pueden ser calculadas, e.j., $w_t = (1-\theta)\frac{y_t}{n_t}$ y $\frac{y_t}{n_t}$

Calibración

Periodicidad del modelo = trimestral

Tecnología					**Preferencias**			
θ	δ	ρ	σ_ε	γ	β	σ	α	η
0.40	0.012	0.95	0.0007	0.0156	0.987	1	0.64	0.012

Solución

- Corra el código escribiendo la siguiente expresión y oprimiendo *enter*:
  dynare rbc_cooley.mod
- Dynare (log)linealiza el modelo y lo re-escribe de la siguiente forma

$$x_t = B\mathbb{E}_t x_{t+1} + Dx_{t-1} + N\varepsilon_t$$

- El modelo RBC, aún con ecuaciones adicionales, puede ser escrito de la siguiente forma:

$$\mathbb{E}_t x_{t+1} = \Lambda x_t + \Xi\varepsilon_t$$

B, D, N, Λ, and Ξ son matrices

Solución

El estado estacionario: cuando no hay choque ε y $x_t = x_{t-1} = x$

```
STEADY-STATE RESULTS:

z               0
k               24.694
c               1.33036
n               0.313068
i               0.466114
y_n             5.73827
y               1.79647
w               3.44296
```

Solución

La condición de Blanchard-Kahn Rank: El papel de los eigenvalues de Λ para la determinación del equilibrio $E_t x_{t+1} = \Lambda x_t + \Xi \varepsilon_t$

# of explosive eigenvalues = # of jump variables $\Longrightarrow$ Unique equilibrium

```
EIGENVALUES:
          Modulus              Real         Imaginary

             0.95              0.95                 0
           0.9663            0.9663                 0
            1.045             1.045                 0
        1.501e+17         1.501e+17                 0
        1.043e+35         1.043e+35                 0


There are 3 eigenvalue(s) larger than 1 in modulus
for 3 forward-looking variable(s)

The rank condition is verified.
```

Jump(forward-looking) variables: $\{\tilde{c}_t, \tilde{i}_t, n_t\}$ state variables: $\{\tilde{k}_t, z_t\}$

Solución

Dynare encuentra la solución

$$x_t^c = \Omega x_{t-1}^s + \Theta \varepsilon_t \text{ and } x_t^s = \Upsilon x_{t-1}^s + \Phi \varepsilon_t$$

donde x_t^c es un vector de variables de control (jump o no predeterminadas) y x_t^s es un vector de variables estado (predeterminadas).

Ω, Θ, Υ and Φ are matrices

POLICY AND TRANSITION FUNCTIONS

	z	k	c	n	i
Constant	0	24.694026	1.330358	0.313068	0.466114
k(-1)	0	0.966272	0.033176	-0.003197	-0.015085
z(-1)	0.950000	2.077582	0.416777	0.232938	2.091867
e	1.000000	2.186929	0.438713	0.245198	2.201965

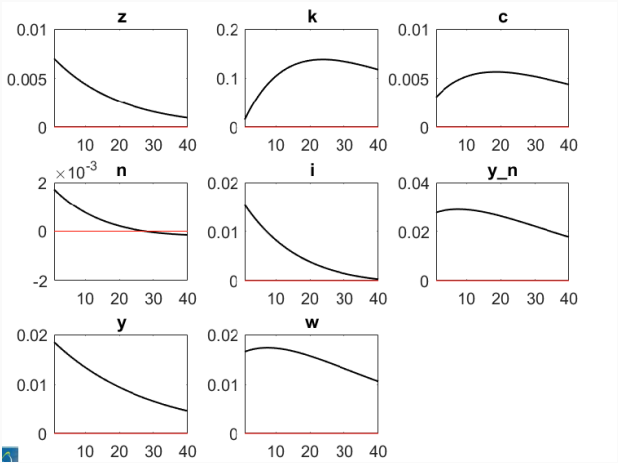
La constante es el nivel de estado estacionario de la variable

Modelo de ciclos reales vs Datos

	Model		Data	
Variable	SD (%)	Corr(var, y)	SD (%)	Corr(var, y)
y	1.34	1.00	1.72	1.00
c	0.35	0.88	1.27	0.83
i	4.32	0.99	8.24	0.91
n	0.72	0.99	1.69	0.92
y/n	0.64	0.98	0.73	0.34

Impulsos Respuesta

Analice la dinámica en respuesta a un choque de una sola vez en z, i.e., $\varepsilon_t \uparrow$ de una std 0.007 en $t = 0$ and después $\varepsilon_t = 0, \forall t \geqslant 1$

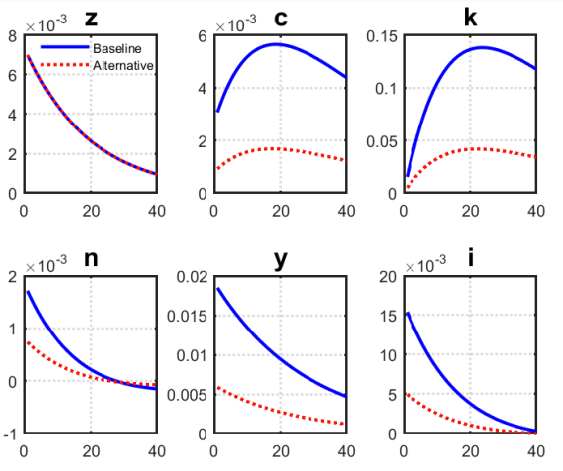


Análsis de sensibilidad

- Ejercicio 1: Cambie el parámetro de la participación del ocio en la función de utilidad α de su valor base de 0.64 a 0.9. ¿Qué ocurre?
- Ejercicio 2: Cambie la participación del capital θ de su valor base de 0.4 a 0.3. ¿Qué ocurre?
- Ejercicio 3: Cambie la persistencia del choque ρ de 0.95 a 0.2. ¿Qué ocurre?

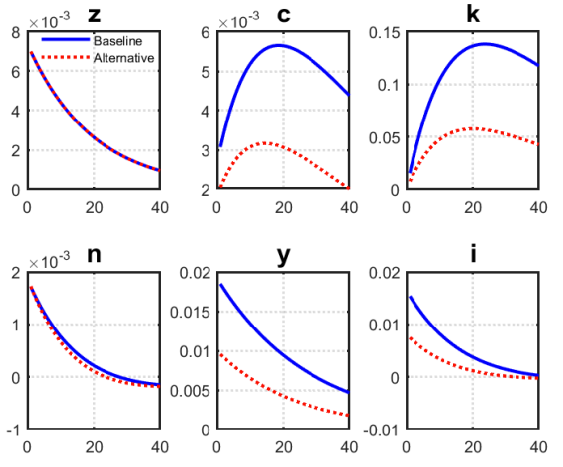
Análsis de sensibilidad

Ejercicio 1: Cambie el parámetro de la participación del ocio en la función de utilidad α de su valor base de 0.64 a 0.9. ¿Qué ocurre?



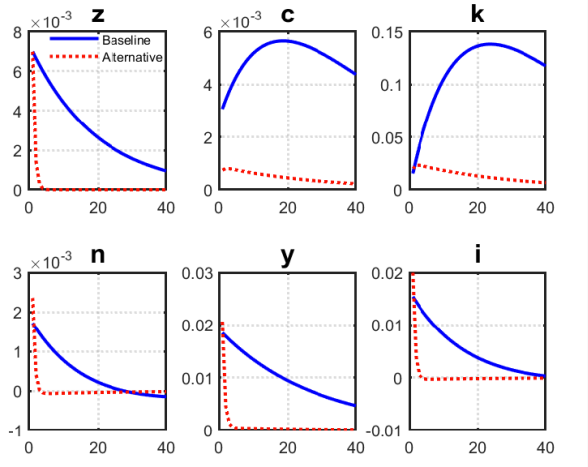
Análsis de sensibilidad

Ejercicio 2: Cambie la participación del capital θ de su valor base de 0.4 a 0.3. ¿Qué ocurre?



Análsis de sensibilidad: Persistencia del choque

Ejercicio 3: Cambie la persistencia del choque ρ de 0.95 a 0.2. ¿Qué ocurre?



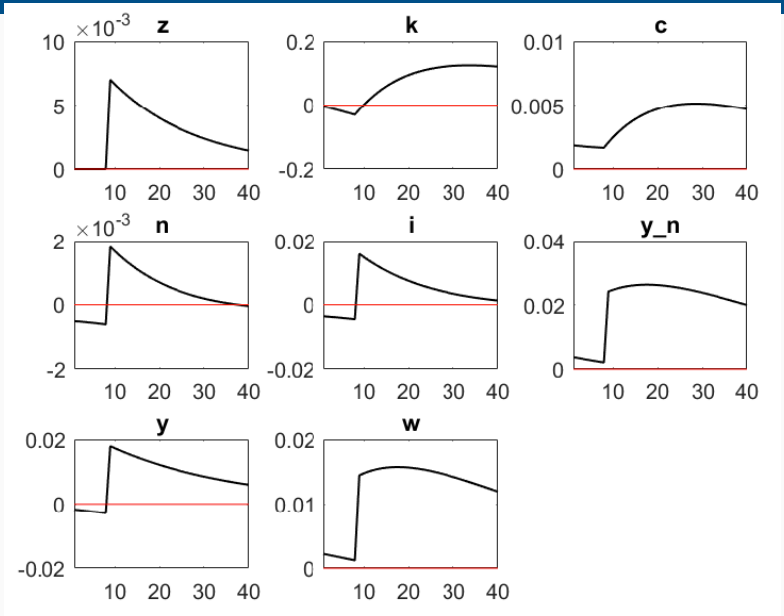
Choques anticipados - Choques de noticias

- Ejercicio 4: cambie el choque de productividad (TFP) en el código de la siguiente forma

$$z_{t+1} = \rho z_t + \varepsilon_{t-7}, \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$$

- El choque fue anticipado hace ocho periodos pero sólo se realiza hoy
- Lo llamamos “news shock” (Beaudry and Portier 2004, JME)
- Esto viene de Pigou: queremos saber si noticias positivas sobre el futuro crean un boom hoy

Impacto de choque de noticias



Mecanismo

- Buenas noticias (TFP ↑ en ocho trimestres) hacen que el producto y la inversión ↓ pero el consumo ↑
- ¿Por qué? “Efecto riqueza” hace sentir a los agentes más ricos, por tanto consumo ↑ and ocio ↑ (trabajo ↓)
- Las buenas noticias causan recesiones, lo cual es exactamente lo opuesto a la intuición y lo que muestran los datos

¿Cómo se puede arreglar el modelo?

- Jaimovich and Rebelo (2009 AER) proponen crear fluctuaciones reales que sean producidas por expectativas en un modelo RBC estándar introduciendo en el modelo características como:
  - Preferencias GHH (para apagar el efecto riqueza)
  - Costos de ajuste de la inversión
  - Utilización variable del capital

Taller Modelo NK

El modelo Neokeynesiano básico de 3 ecuaciones en Galí (2008)

- Curva de Phillips (NKPC)

$$\pi_t = \beta E_t[\pi_{t+1}] + \mathcal{K}\tilde{y}_t$$

$$\text{con } \mathcal{K} \equiv \lambda\left(\sigma + \frac{\phi+\alpha}{1-\alpha}\right), \lambda \equiv \frac{(1-\theta)(1-\beta\theta)}{\theta}\Theta, \text{ y } \Theta \equiv \frac{1-\alpha}{1-\alpha+\alpha\varepsilon}$$

- Curva IS (DISE)

$$\tilde{y}_t = E_t\tilde{y}_{t+1} - \frac{1}{\sigma}(i_t - E_t[\pi_{t+1}] - r_t^n)$$

$$\text{con } r_t^n = \rho + \sigma E_t(\triangle a_{t+1}) \quad \text{y} \quad \rho \equiv -log\beta$$

- Regla de Taylor

$$i_t = \rho + \Phi_\pi \pi_t + \Phi_y \tilde{y}_t + v_t$$

Otras ecuaciones útiles

- Dos choques

$$v_t = \rho_v v_{t-1} + \varepsilon_t^v, \rho_v \in [0,1), \varepsilon_t^v \sim \mathcal{N}(0, \sigma_v^2)$$
$$a_t = \rho_a a_{t-1} + \varepsilon_t^a, \rho_a \in [0,1), \varepsilon_t^a \sim \mathcal{N}(0, \sigma_a^2)$$

- Otras ecuaciones

$$\text{Brecha de producto} : \tilde{y}_t = y_t - y_t^n$$

$$\text{Producto natural} : y_t^n = \psi_{ya}^n a_t + \psi_y^n$$

$$\text{Producto real (}\textit{actual product}\text{)} : y_t = a_t + (1-\alpha) n_t$$

$$\text{Real interest rate} : r_t = i_t - E_t \hat{} \pi_{t+1} \text{‘}$$

$$\text{Money demand} : m_t - p_t = y_t - \eta i_t$$

donde $\psi_{ya}^n \equiv \frac{1+\varphi}{\sigma(1-\alpha)+\varphi+\alpha}$ y $\psi_y^n \equiv -\frac{(1-\alpha)(\mu - log(1-\alpha))}{\sigma(1-\alpha)+\varphi+\alpha}$

Resumen y Revisión - *Housekeeping*

- Variables endógenas: $\tilde{y}_t, y_t, y_t^n, \pi_t, n_t, i_t, r_t, r_t^n, a_t, \upsilon_t$
- Variables exógenas: ε_t^v y ε_t^a con varianzas σ_v^2 y σ_a^2
- Parámetros: $\beta, \sigma, \varphi; \alpha, \varepsilon, \theta; \eta; \phi_\pi, \phi_y; \rho_v, \rho_a; \rho, \mu$
- Parámetros compuestos: $\mathcal{K}, \lambda, \Theta, \psi_{ya}^n, \psi_y^n$

Resumen y Revisión - *Housekeeping*

- El modelo ya está log-linealizado, así que las variables ya están expresadas como desviaciones logarítmicas del estado estacionario. Por tanto, el estado estacionario de cada variable es cero.
- La frecuencia del modelo es trimestral
- La tasa de interés nominal i, la tasa de interés real r, y la inflación π también están en términos trimestrales
- Para expresar las tasas de inflación y de interés en términos anuales, multiplíquelas por 4.

Calibración

- Asigne valores a los parámetros
  - Consistentes con la evidencia empírica
  - Los momentos generados por el modelo sean consistentes con los datos

Calibración

Parámetro	Valor	Objetivo
β	0.99	Real annual financial asset return = 4 %
σ	1	Log utility
φ	1	Frisch Labor elasticity $(1/\varphi) = 1$
α	1/3	Data
ε	6	Literature
θ	2/3	Avg. price duration $\left(\frac{1}{1-\theta}\right)$ 3 quarters
ϕ_π	1.5	Mimic FFR during Greenspan era
η	4	Regress log(M2) on 3-month T-bill rate
ϕ_y	0.5/4	Mimic FFR during Greenspan era
ρ_v	0.5	
ρ_a	0.9	

Simulación

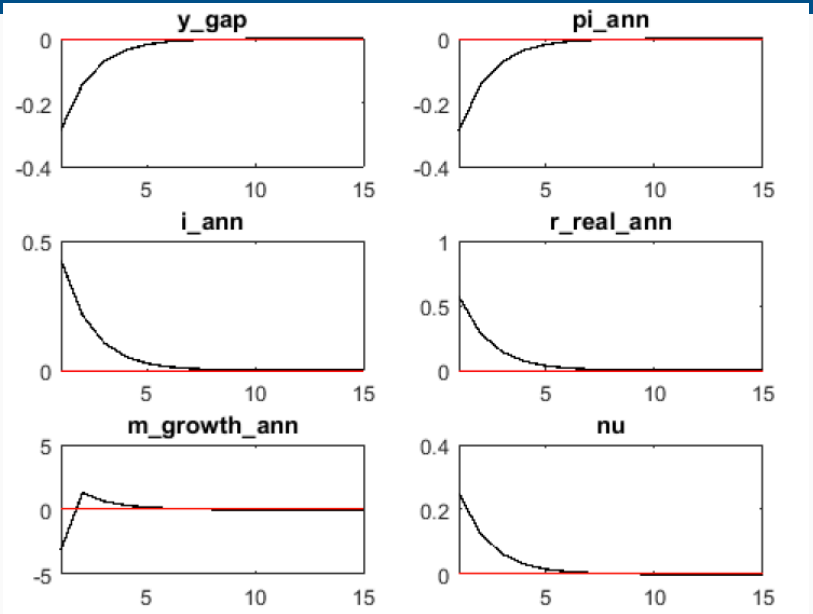
- El modelo es solucionado y simulado en Dynare utilizando el comando stoch_simul
- Opciones para stoch_simul

Opciones	Significado	Default
period	# de periodos usado en las simulaciones	0
nocorr	no imprima matriz de correlación	PRINT
drop	# de puntos ignorados al principio de la simulación	100
irf	# periodos para IRFs	15
order = [1,2,3]	Orden de aproximación de Taylor	1

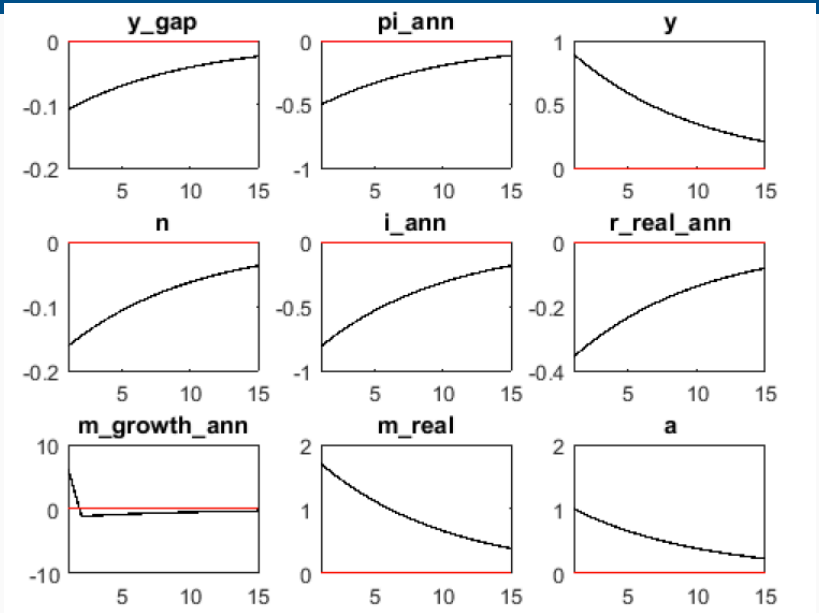
Ejecutando el Código de Dynare

- Corra el archivo de dynare Gali_2008_chapter_3.mod
- Escoja
  - money_growth_rule=0 para una regla de tasa de interés (regla de Taylor rule); or
  - money_growth_rule=1 para una regla de tasa de crecimiento del dinero (MGR)

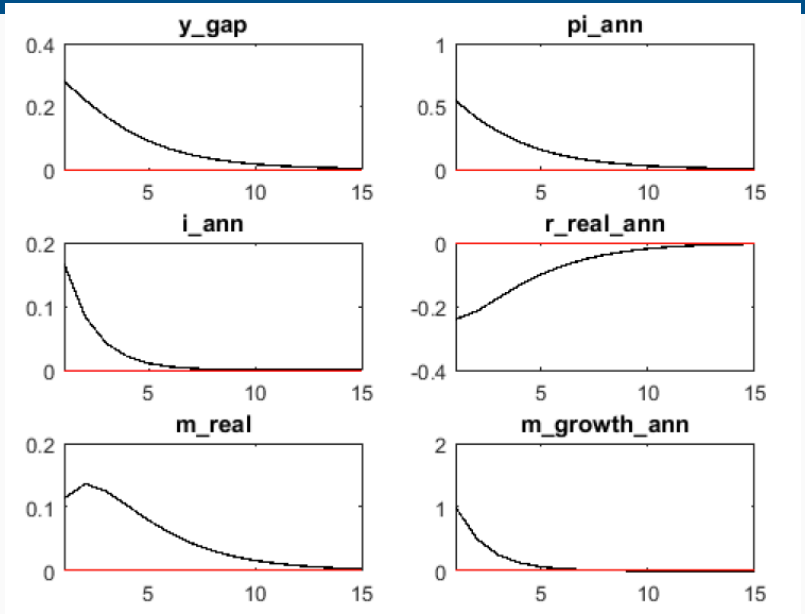
Efectos de un choque de política monetaria (Taylor Rule)



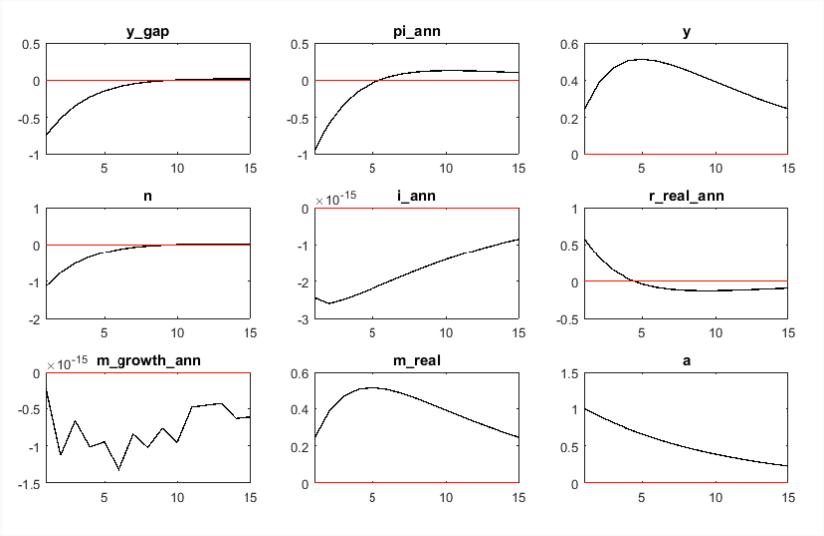
Efectos de un choque de tecnología (Taylor Rule)



Efectos de un choque de política monetaria (MGR)



Efectos de un choque de tecnología (MGR)

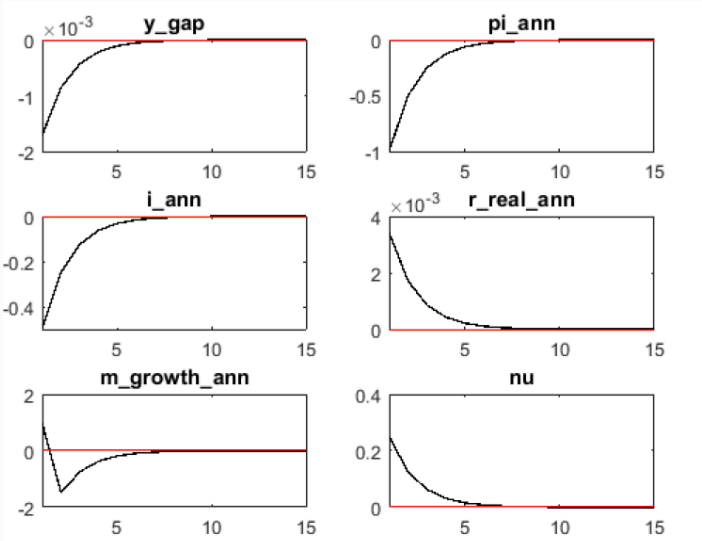


Sensibilidad

- Ejercicio 1: Bajo una regla de Taylor, cambie el parámetro de rigideces de precios θ de 2/3 (*benchmark*) a 0.01. ¿Qué pasa y por qué?
- Ejercicio 2: Bajo una regla de Taylor, cambie la elasticidad de la demanda ε de 6 a 1.01. ¿Qué pasa y por qué?

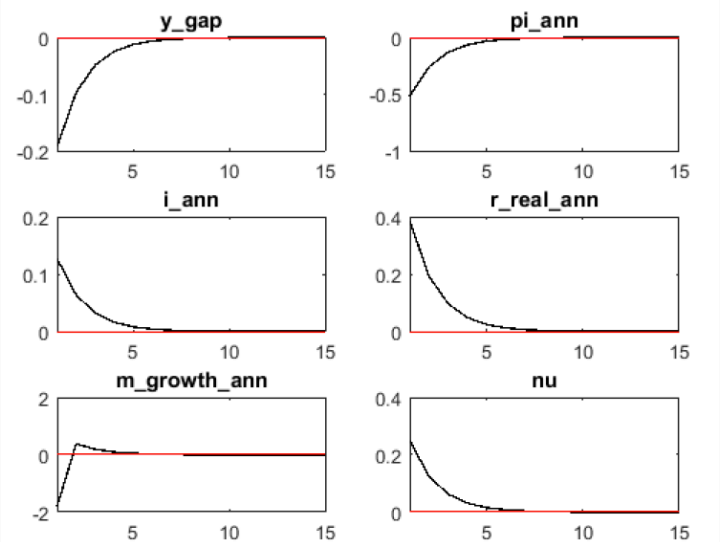
The Effects of a Monetary Policy Shock (Taylor Rule)

Ejercicio 1: Bajando el parámetro de rigideces de precios θ to 0.01



The Effects of a Monetary Policy Shock (Taylor Rule)

Ejercicio 2: Bajando la elasticidad de la demanda a ε to 1.01



Cambiando la regla de Taylor

- Ejercicio 3: Bajo una regla de Taylor, cambie el parámetro de respuesta ϕ_π de 1.5 a 0.8. ¿Qué ocurre?

Cambiando la regla de Taylor

- Ejercicio 3: Bajo una regla de Taylor, cambie el parámetro de respuesta ϕ_π de 1.5 a 0.8. ¿Qué ocurre?
- Se obtiene *indeterminacy*, equilibrio explosivo: ¡La condición de Blanchard-Kahn no se satisface!
- El principio de Taylor en el modelo NK canónico: un incremento de 1 % en inflación debería ser respondido con un incremento mayor a 1 % en la tasa de interés nominal.

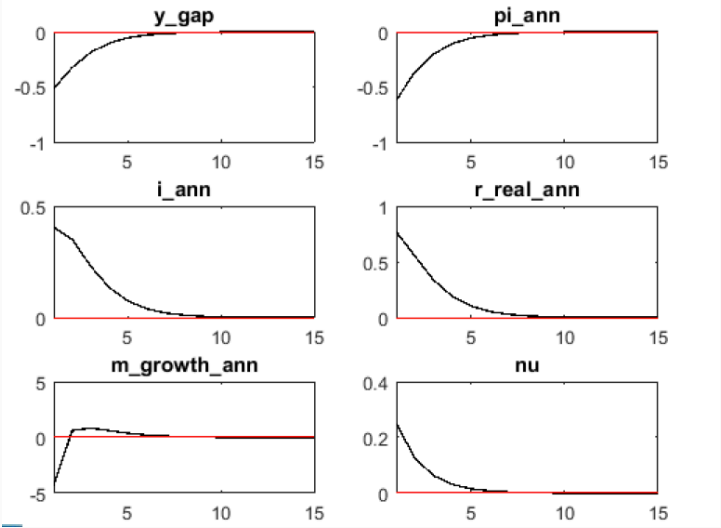
Suavizamiento de tasas de interés

Ejercicio 4: Cambie la regla de Taylor por

$$i_t = 0{,}5 \times i_{t-1} + 0{,}5 \times (\phi_\pi \pi + \phi_y \tilde{y}_t) + v_t$$

Efectos de un choque de política monetaria (Taylor Rule)

Ejercicio 4: Cambie la regla de Taylor por $i_t = 0{,}5 \times i_{t-1} + 0{,}5 \times (\phi_\pi \pi + \phi_y \tilde{y}_t) + \upsilon_t$



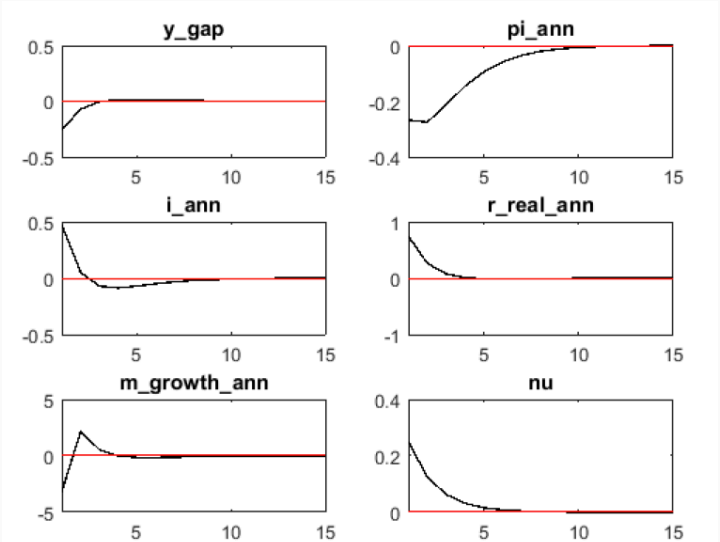
Phillips Curve Smoothing

Ejercicio 5: Agregue un término de indexación a la NKPC

$$\pi_t = 0{,}5 \times \pi_{t-1} + 0{,}5 \times \beta E_t \hat{} \pi_{t+1} ` + \kappa \tilde{y}_t$$

Efectos de un choque de política monetaria (Taylor Rule)

Ejercicio 5: Cambie la NKPC a $\pi_t = 0{,}5 \times \pi_{t-1} + 0{,}5 \times \beta E_t \hat{} \pi_{t+1} ' + \mathcal{K} \tilde{y}_t$



Por qué formación de hábitos?

- Note que en CEE (1999), el impacto de un choque monetario positivo (contractivo) sobre el producto es negativo pero tiene forma de joroba (*hump-shaped*)
- El modelo NK de la clase captura el impacto negativo, pero no la forma de joroba
- Agregamos formación de hábito "externa"(*"external" habit formation*) al modelo.
- La función de utilidad del periodo cambia a

$$u(C_t, N_t) = \frac{(C_t - hC_{t-1})^{1-\sigma}}{1-\sigma} - \frac{N_t^{1-\varphi}}{1+\varphi}$$

Donde h mide el grado de formación de hábitos

Condiciones de Primer Ordem

- Condición intratemporal

$$\frac{N_t^{\varphi}}{(C_t - hC_{t-1})^{\sigma}} = \frac{W_t}{P_t} \tag{7}$$

- Condición intertemporal

$$(C_t - hC_{t-1})^{-\sigma} = \lambda_t P_t \tag{8}$$

$$1 = \beta E_t \left(\frac{\lambda_{t+1}}{\lambda_t} \right) i_t \tag{9}$$

DISE Modificada

- Siguiendo pasos similares a los que se utilizaron en la derivación de la DISE en el modelo básico se obtiene

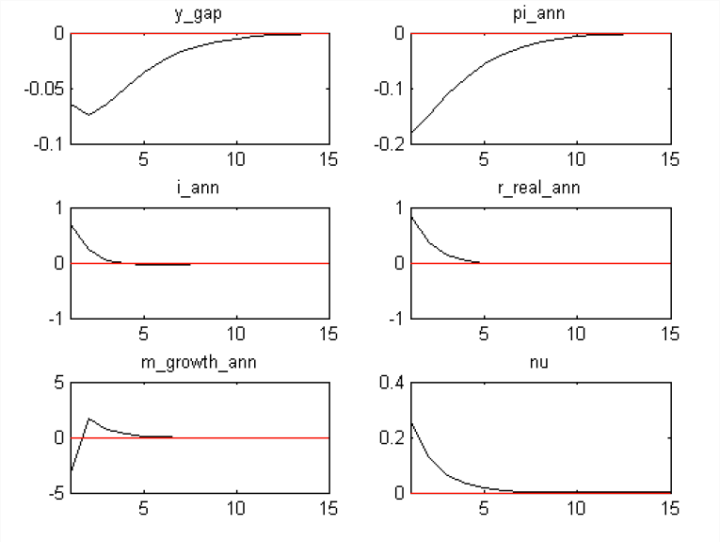
$$\tilde{y}_t = \frac{1}{1+h} E_t[\tilde{y}_{t+1}] + \frac{h}{1+h}\tilde{y}_{t-1} - \frac{1-h}{\sigma(1+h)}(i_t - E_t[\pi_{t+1}] - r_t^n)$$

- Note que cuando $h = 0$, esta se reduce a la DISE del modelo NK básico.

Ejercicio 6

- Modifique el código the Dynare **Gali_2008_chapter_3.mod** agregando formación de hábito
- Pista: sólo tiene que cambiar la equación IS.
- No olvide agregar un nuevo parámetro $h = 0{,}8$
- Cambie h y póngale diferentes valores.

Los efectos de un choque de política monetaria bajo formación de hábitos



Finalmente, toda la orquesta junta!!!

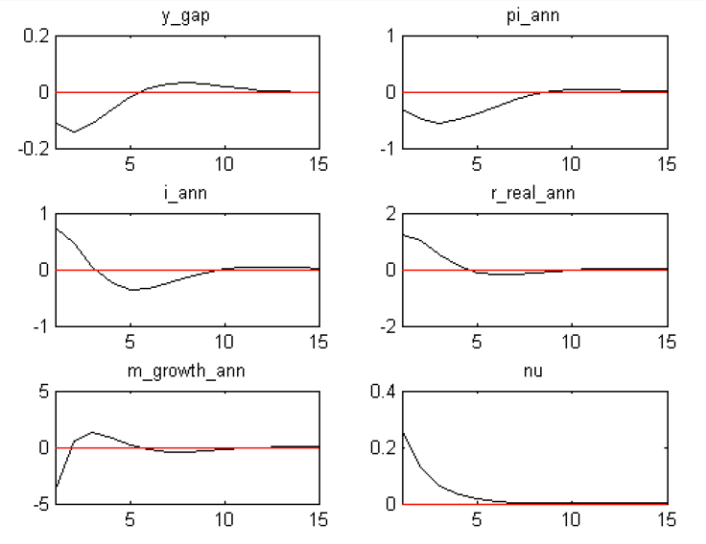
- Ejercicio 9: NK básico + formación de hábitos + suavizamiento de la tasa de interés+ suavizamiento de NKPC

$$\tilde{y}_t = \frac{1}{1+h}E_t[\tilde{y}_{t+1}] + \frac{h}{1+h}\tilde{y}_{t-1} - \frac{1-h}{\sigma(1+h)}(i_t - E_t[\pi_{t+1}] - r_t^n)$$

$$\pi_t = 0{,}5 \times \pi_{t-1} + 0{,}5 \times \beta E_t[\pi_{t+1}] + \kappa\tilde{y}_t$$

$$i = 0{,}5 \times i_{t-1} + 0{,}5 \times (\phi_\pi \pi + \phi_y \tilde{y}_t) + v_t$$

Los efectos de un choque de política monetaria bajo formación de hábitos e inercia



Recordemos CEE (1999)

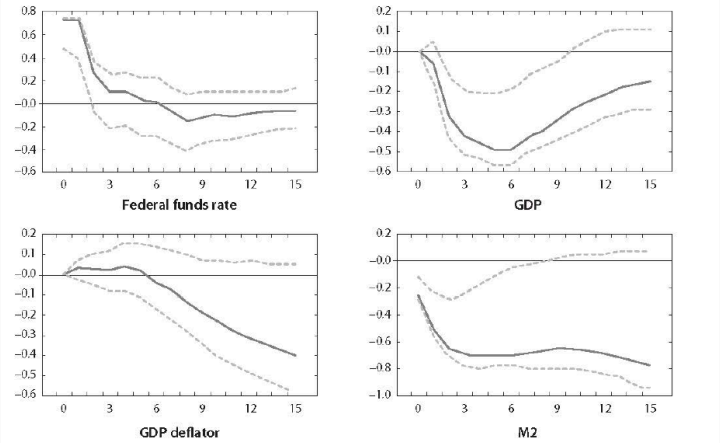


Figure 1.1. Estimated Dynamic Response to a Monetary Policy Shock
Source: Christiano, Eichenbaum, and Evans (1999).

¿Qué hemos aprendido?

- La rigidez de precios lleva a que la política monetaria tenga impactos reales
- Agregarle al modelo elementos que le den inercia ayuda a generar impulsos respuesta del producto a un choque de política monetaria que tengan forma de joroba
- Cómo modificar el modelo en Dynare para capturar elementos de la evidencia empírica