Capitulo 2
Fundamentos de Dynare*

Hamilton Galindo
Arizona State University (ASU)
hamilton.galindo@asu.edu

Alexis Montecinos
Massachusetts Institute of Technology (MIT)
alexis.montecinos@sloan.mit.edu

Borrador: 8 de julio de 2017

Índice

*Disclaimer: cualquier error u omisión es responsabilidad de los autores.

1. Introducción

Los modelos DSGE se pueden resumir en un conjunto de ecuaciones en diferencias con no linealidades. La naturaleza de este sistema exige aplicar métodos numéricos para aproximarse a la solución. Realizar este trabajo sin un *software* es muy tedioso y hasta quizá ineficiente.

Matlab es un *software* que tiene implementado las herramientas de optimización y solución de ecuaciones en diferencias nolineales; además, trabaja bajo un enfoque matricial. Estas características convierten a este software en un candidato importante para la solución y simulación de los modelos DSGE. Sin embargo, contextualizar el modelo al lenguaje de Matlab requiere que el usuario tenga un nivel avanzado de programación en dicho lenguaje, lo cual complica la utilización de este *software*.

En este escenario diversos grupos de economistas, con formación en matemática e informática, han tratado de construir programas basados en Matlab que faciliten la solución de los modelos DSGE. En estos esfuerzos surge Dynare como un preprocesador que permite traducir el modelo en lenguaje de Matlab[1].

En línea con lo anterior, el objetivo de este capítulo es entender los principales comandos de Dynare que son utilizados para realizar cada uno de los pasos en la construcción y simulación de un modelo DSGE. Para ello este capítulo está dividido en tres partes.

En la primera parte se describe cada uno de los comandos necesarios para trasladar el modelo DSGE al ambiente de Dynare; además, se menciona los comandos necesarios para resolver y simular el modelo.

En la segunda parte se ilustra los códigos antes mencionados al aplicarlos en un modelo RBC básico (modelo de Long y Plosser, 1983). Finalmente, en la última sección se menciona los códigos utilizados en este capítulo.

2. ¿Qué es Dynare?

Dynare es un pre-procesador y colección de rutinas de Matlab. Es decir, Dynare es una colección de códigos de matlab que actua como un *toolbox*. El objetivo prncipal de Dynare es resolver, simular y estimar diferentes modelos no lineales con variables *forward looking*, entre los cuales se encuentran los modelos DSGE y OLG (generaciones traslapadas).

El principal insumo de este *toolbox* es un archivo con extensión ".mod", donde se escribe el modelo y las sentencias que se desea que Dynare ejecute (resolver, estimar, etc). Para crear este archivo se abre un block de notas y se guarda con extensión mod. En este contexto, ¿Cómo se invoca Dynare?: luego de crear el archivo .mod "`ejemplo.mod`", en el *prompt* de Matlab se coloca lo siguiente:

>> `dynare` ejemplo

[1]Este capítulo está basado en el manual de Dynare disponible en su página web "www.dynare.org". Asimismo, en esta página se encuentra este *toolbox* listo para descargarlo y diferentes ejemplos ilustrativos.

El comando **dynare** pone en marcha el pre-procesador (Dynare) sobre el archivo .mod y ejecuta las instrucciones incluidas en este archivo ("ejemplo.mod"). Considerando un nombre genérico para el archivo .mod "filename.mod", el pre-procesador crea 3 archivos intermedios:

Cuadro 1: Archivos creados por Dynare

3 archivos intermedios creados por Dynare		
filename.m	**filename_dynamic.m**	**filename_static.m**
Contiene [1]declaración de variables y [2]tareas de cálculo.	Contiene las ecuaciones del modelo dinámico; es decir, considera los adelantos y rezagos de las variables.	Contiene las ecuaciones del modelo estático de largo plazo; es decir, considera la ecuaciones sin temporalidad.

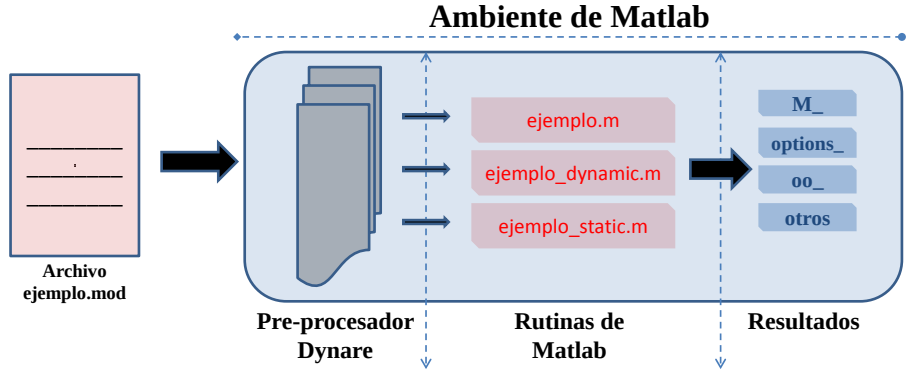
Dynare ejecutará las tareas de cálculo al ejecutar el archivo "**filename.m**". Entre los resultados de Dynare se encuentra 3 principales variables (se muestran en el *workspace de Matlab*):

Cuadro 2: Variables creadas por Dynare

3 principales variables (estructura) creados por Dynare		
M_	**options_**	**oo_**
Contiene información variada del modelo. Por ejemplo: el nombre del archivo mod y nombres de las variables.	Contiene los valores de varias opciones usadas por Dynare durante el cálculo	Contiene varios resultados del cálculo. Por ejemplo: la función impulso-respuesta y las simulaciones.

Dynare guarda estas tres variables en la carpeta de trabajo actual (*current folder*) con el nombre: "**filename_results.mat**". En la figura [1] se muestra cómo Dynare trabaja con Matlab.

Figura 1: Dynare como pre-procesador de Matlab



3. Estructura del archivo .mod

El archivo que contiene el modelo (.mod), el cual será utilizado por Dynare, tiene una estructura de seis bloques principales (figura [2]). El primer bloque es el preambulo, en el cual se especifica las variables endógeneas y exógenas, y los parámetros del modelo. El segundo bloque es el modelo propiamente dicho. Es este bloque se escribe las ecuaciones en sus versión no lineal o lineal (o log-lineal). El tercer bloque es la especificación de los valores iniciales, los cuales definen el punto de partida para que Dynare calcule el estado estacionario del sistema. El cuarto bloque es el cálculo del estado estacionario. El quinto bloque corresponde a la definición de la varianza o desviación estándar de los choques; finalmente, el sexto bloque contiene el cálculo de la solución del modelo, las simulaciones, el cálculo de los momentos y la construcción de las funciones impulso-respuesta.

Es importante mencionar que además de estos seis bloques se pueden agregar otros comandos dependiendo de las tareas que se requiera; por ejemplo, si se desea hacer análisis de sensibilidad de los parámetros se puede agregar comandos que realicen esta tarea. Los seis bloques antes mencionados son las partes fundamentales que todo archivo .mod debe de tener.

3.1. El preámbulo

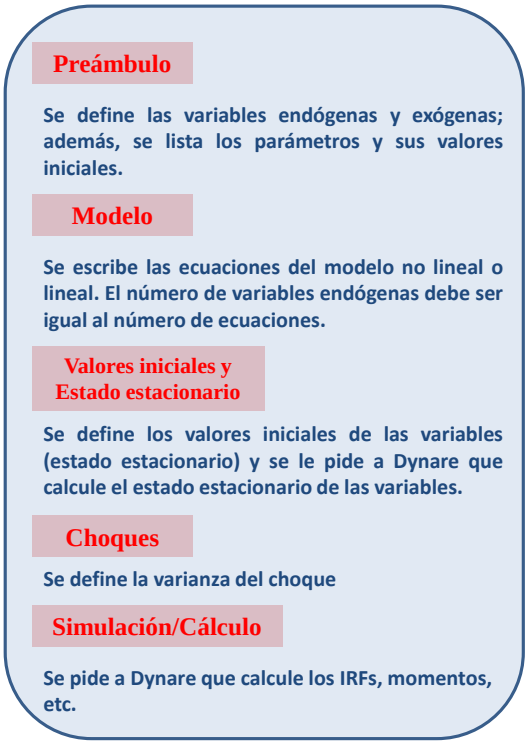
En el preámbulo se especifica las variables (endógenas y exógenas) y los parámetros (y sus valores). Tres comandos le indicarán a Dynare qué variables son del modelo y cuales son los parámetros: variables (*var* y *varexo*) y parámetros (*parameters*).

Código 1. Declaración de variables endógenas

```
var variable_name1 $latex_name1$ (long_name= 'nombre');
```

El comando **var** declara las variables endógenas y tiene tres componentes: el primero se refiere al nombre de la variable que será utilizado en todo el archivo .mod (`variable_name1`); el segundo indica el nombre que esta variable tomará en el archivo LaTeX (\$latex_name1\$); y el tercero es una opción (la cual tiene que estar entre paréntesis) que permite escribir el nombre largo de la variable (long_name= 'nombre'). El cuadro [3] ilustra la utilización de este código.

Figura 2: Estructura del archivo .mod



Cuadro 3: Ejemplos de declaración de variables endógenas

Ejemplo 1	**Ejemplo 2**	**Ejemplo 3**
var	var	var
y	y y_t	y y_t (long_name = 'Producto')
c	c c_t	c c_t (long_name = 'Consumo')
k;	k k_t;	k k_t (long_name = 'Capital');

Código 2. Declaración de variables exógenas

```
varexo variable_name1 $latex_name1$ (long_name= 'nombre');
```

El comando **varexo** declara las variables exógenas (choques) y, de la misma forma que las variables endógenas, tiene tres componentes: el primero se refiere al nombre de la variable que será utilizado en todo el archivo .mod (`variable_name1`); el segundo indica el nombre que esta variable tomará en el archivo LaTeX ($latex_name1$); y el tercero es una opción (la cual tiene que estar entre paréntesis) que permite escribir el nombre largo de la variable (long_name= 'nombre').

Cabe mencionar que en un modelo estocástico usualmente la productividad (a_t) tiene un comportamiento autorregresivo de la siguiente forma:

$$a_{t+1} = \rho a_t + \epsilon_t$$

Donde ϵ_t es el componente estocástico, con una distribución normal con media cero y varianza constate. Para Dynare, a_t es una variable endógena y debido a que ϵ_t es un ruido blanco, esta es considerada como una variable exógena. Por tanto, bajo los comandos declarados en el código 2, esta variable exógena se podría escribir en Dynare de tres formas (ver cuadro [4]).

Cuadro 4: Ejemplos de declaración de variables enxógenas

Ejemplo 1	**Ejemplo 2**	**Ejemplo 3**
varexo e	varexo e e_t	varexo e e_t (long_name = 'Choque de productividad')

Código 3. Parámetros

```
parameters parametro_name1 $latex_name1$ (long_name= 'nombre1')
parametro_name2 $latex_name2$ (long_name= 'nombre2');
```

El comando *parameters* declara los parámetros que se usarán en el modelo. No solo los parámetos de las ecuaciones (funciones) de comportamiento de los agentes (por ejemplo la función de utilidad); sino también, los valores iniciales, que usualmente son los valores de estado estacionario, y los parámetros asociados a los choques. Además, en este segmento se debe de asignar los valores que corresponden a cada parámetro (calibración). El cuadro [5] describe tres ejemplos de la declaración de los parámetros y de la asignación de sus valores.

Cuadro 5: Ejemplos de declaración de parámetros

Ejemplo 1	**Ejemplo 2**	**Ejemplo 3**
parameters	parameters	parameters
beta	beta $ \ beta$	beta $ \ beta$ (long_name = 'Elasticidad de Frisch')
delta;	delta $ \ delta$;	delta $ \ delta$ (long_name = 'Depreciación');
beta=0.99;	beta=0.99;	beta=0.99;
delta=0.22;	delta=0.99;	delta=0.99;

3.2. El modelo

El modelo (sistema de ecuaciones no lineales) es declarado en Dynare por medio del bloque *model;...end;*.

Código 4. Declaración del modelo

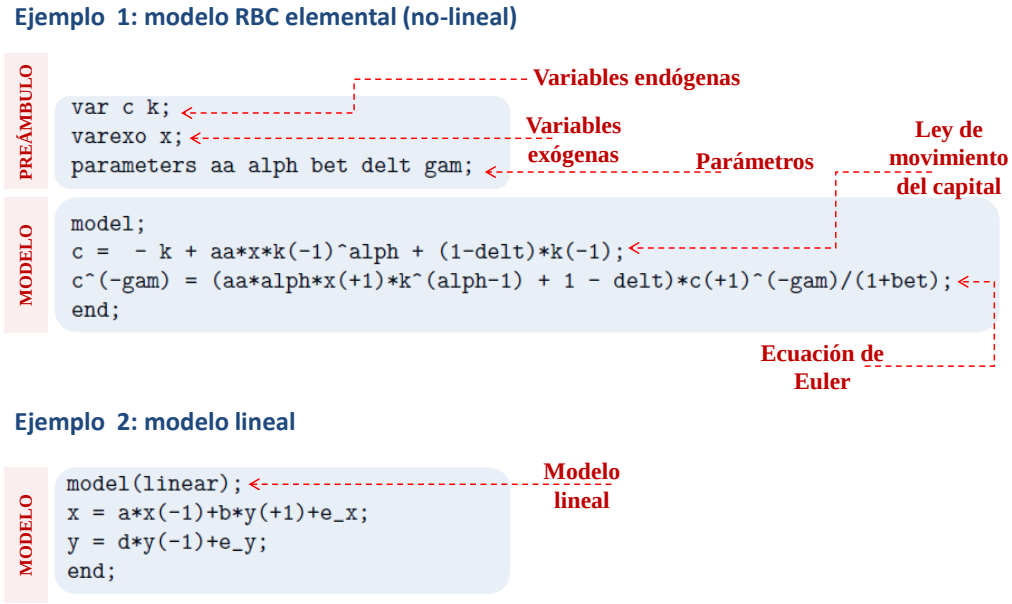
```
model(opciones); ecuación1; ecuación2;...;ecuaciónN; end;
```

Este bloque detalla las ecuaciones principales del modelo. Se puede escribir el modelo (no-lineal) en Dynare tal como se tiene en el *papel*, para ello se introduce las ecuaciones en el ambiente `model;` $---$ `end;`

Código
model;
ecuación1;
ecuación2;
...
ecuaciónN;
end;

Se tiene que tener en cuenta que el número de ecuaciones debe ser igual al número de variables endógenas. Si el modelo que se escribe en Dynare está linealizado (sea con variables en niveles o variables en logarítmo), entonces se escribe: **model(linear)**.

Figura 3: Modelo nolineal y lineal



3.3. Valores iniciales

Dentro de este bloque se coloca los valores inciales de cada una de las variables endógenas, los cuales generalmente son los valores de estado estacionario calculados por el usuario. Estos valores son utilizados por Dynare como punto de partida para el cálculo del estado estacionario. Es importante mencionar que cuando el modelo está en forma log-lineal, los valores de estado estacionario de las variables (en log-desviaciones) son iguales a cero; por tanto, los valores iniciales son iguales a cero.

Código 5. *Declaración de los valores iniciales*

```
initval; variable_name1 = valor1; variable_name2 = valor2;...;variable_nameN
= valorN; end;
```

El cuadro [6] describe dos ejemplos de valores iniciales.

Cuadro 6: Valores iniciales (modelo no-lineal y lineal)

Código	Ejemplo (no-lineal)	Ejemplo (log-lineal)
initval;	`initval;`	`initval;`
variable_name1 = valor1;	`c = 0.5;`	`ch = 0`
variable_name2 = valo2;	`k = 0.1;`	`kh = 0`
...	...	...
variable_nameN = valorN;	`y = 0.8;`	`yh = 0`
end;	`end;`	`end;`

Es preciso mencionar que la variable *ch* es igual a "$lnc_t - lnc_{ss}$"; es decir, es la desviación de la variable en logarítmo con respecto a su estado estacionario. La cual, por construcción, en estado estacionario es igual a cero.

Los valores iniciales son los que Dynare utilizará en el "filename_static.mod" para calcular el estado estacionario. Dynare necesita un punto de partida para dicho cálculo debido a que el método de solución es de apróximaciones sucesivas (método de Newton).

3.4. Estado estacionario

Dynare tiene dos formas de considerar el estado estacionario del modelo. La primera es que Dynare mismo calcule el estado estacionario. Para ello Dynare utiliza el método de Newton para resolver ecuaciones no lineales. La segunda es brindarle a Dynare un archivo de Matlab (m-file) que contenga el estado estacionario.

El método de Newton o conocido también como el método de Newton − Raphson es una técnica orientada a resolver ecuaciones no lineales. Esta técnica encuentra la solución por medio de iteraciones sucesivas desde un punto inicial. El objetivo de esta técnica consiste en encontrar los valores de la variable "x" que hacen que la función (o ecuación) sea cero; es decir, busca las raices de la función:

$$f(x) = 0$$

La técnica empieza con un punto inicial x_0 y aproxima el siguiente valor de "x" por medio:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \xrightarrow{\text{Generalizando}} x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Esto aplicado al cálculo de estado estacionario del modelo se tiene, por ejemplo, para la función de producción:

$$y_t = a_t k_t^{\alpha} h_t^{1-\alpha}$$

Definimos $f(x)$, donde $x = [y_t, a_t, k_t, h_t]$

$$f(x) = y_t - a_t k_t^{\alpha} h_t^{1-\alpha} = 0$$

El objetivo es encontrar la solución; es decir, el estado estacionario ($x_t = x_{t+1}$). Para ver mayor detalle del método revisar Kelley (2003).

Código 6. *(Método 1) Usando el método de Newton*

```
steady;
```

Para que Dynare calcule el estado estacionario por el método de Newton se coloca el comando **steady** después del bloque del modelo. Para ello considera como punto inicial los valores que se indiquen en la sección `initval`. Dynare aplica el método de Newton al modelo estático (el modelo del .mod sin rezagos ni adelantos) para encontrar el estado estacionario.

Código 7. *Guardando el estado estacionario*

```
oo_.steady_state;
```

El estado estacionario calculado por `steady` se guarda en "oo_.steady_state". Cabe mencionar que el orden en que aparecen los estados estacionarios es el mismo en cómo han sido declaradas las variables endógenas en la sección `var`.

Código 8. *(Método 2) Usando un archivo que contenga el estado estacionario*
En este segundo método existen dos opciones:

- **Bloque de "estado estacionario" en el .mod:** en este caso se define cada variable en función de los parámetros profundos. El código es:
  `steady_state_model; r = 1/beta; k = delta*beta;...;end;`

  El m-file en que guardará Dynare los estados estacionario es "NombreMod_steadystate2.m". Donde "NombreMod" es el nombre del archivo" .mod". Cabe mencionar que este bloque se coloca después del bloque de *parámetros*.

- **m-file que contenga el "estado estacionario":** en este caso se puede construir una función en Matlab que calcule los estados estacionarios. Esto requiere hacer un poco más de programación en un m-file.

3.5. Dynare y LaTeX

En dynare existe la posibilidad de trasladar las ecuaciones del modelo al formato de LaTeX. Para ello se usa dos códigos dependiendo de qué tipo de modelo se desea traducir en lenguaje LaTeX:

Código 9. *Modelo dinámico en LaTeX*

```
write_latex_dynamic_model;
```

Este código escribe cada ecuación del modelo en formato LaTeX y lo guarda en un archivo ".tex" dependiendo del nombre del archivo ".mod". Por ejemplo, si el archivo mod tiene el nombre de "campbell.mod", el archivo en LaTeX será "campbell_dynamic.tex", el cual contiene la lista de todas las ecuaciones dinámicas. Cabe mencionar que si en los parámetros y en las variables se han declarado nombres en versión LaTeX, el archivo ".tex" los usuará.

Código 10. *Modelo estático en LaTeX*

```
write_latex_static_model;
```

Este código permite guardar las ecuaciones del modelo, en su versión estática, en un archivo ".tex". La versión estática significa que en las ecuaciones se han eliminado los rezagos y atrasos. De la misma manera que el código para el modelo dinámico, la versión estática se guarda en "campbell_static.tex" (cuando el modelo en Dynare se llama "campbell.mod").

Cabe mencionar que estos códigos (ecuaciones dinámicas o estáticas) se escriben después del bloque *model*.

3.6. Definición de los choques

En este bloque se define los choques temporales del modelo (choque de productividad, de gasto público, etc.). En Dynare las variables exógenas (choque) toma valores aleatorios que siguen una distribución normal con media cero y varianza constante. En el archivo .mod se debe de especificar la varianza.

Código 11. *Definición de choques*

Para definir la varianza o desviación estándar el choque existe dos formas:

Alternativa 1	*Alternativa 2*
`shocks;`	`shocks;`
`var variable_name =` *valor_varianza;*	`var variable_name;`
`end;`	`stderr` *valor_desviación_estándar;*
	`end;`

En el cuadro [7] se describe un ejemplo de la aplicación del código del bloque del choque.

Cuadro 7: Definición del choque

Alternativa 1	**Alternativa 2**
`shocks;`	`shocks;`
`var e = 0.5;`	`var e;`
`end;`	`stderr` $0.5^{1/2}$`;`
	`end;`

3.7. Evaluación del modelo: condiciones de Blanchard y Kahn

Dynare tiene la capacidad de calcular los eigenvalores del modelo linealizado alrededor de los valores asignados en el bloque de `initval`, los cuales usualmente son los estados estacionarios. En particular, si el número de eigenvalores con módulo mayor a uno es igual al número de variable *forward looking*, entonces el sistema de ecuaciones linealizadas tiene solución única. En el capítulo 3 se detalla estos criterios y el método de solución de Blanchard y Kahn (1980).

Código 12. *Evaluación del modelo*

```
check;
```

Este comando calcula los eigenvalores y los guarda en la variable global "oo_.dr.eigval". Dynare también cuenta con otro comando que brinda varias pruebas de salud del modelo e imprime un mensaje si es que un problema es detectado: `model_diagnostics;`.

3.8. Cálculo de la solución estocástica

Dynare utiliza el código `stoch_simul` para obtener las funciones de política y de estado (transición) del modelo.

Código 13. *stoch_simul (opciones)*
Este comando resuelve el modelo estocástico (o modelo de expectativas racionales) usando el método de perturbación[2]*. Cabe mencionar que en el proceso de solución del sistema de ecuaciones, Dynare utiliza el método de descomposición generalizada de Schur (o descomposición QZ). Lo que hace este método es descomponer una matriz en tres matrices multiplicativas:* $A = QUQ^{-1}$*. La descomposición de Schur mantiene el mismo espíritu que la descomposición de Jordan (ver el capítulo 3 para mayor detalle).*

Dentro de las opciones de `stoch_simul` se le puede solicitar a Dynare que realice operaciones especificas como hallar los impulsos respuestas, estimar los parámetros, etc. Por ejemplo:

stoch_simul(order=1;irf=30)

Esta sentencia indica a Dynare que al sistema de ecuaciones no lineales (escrito en el bloque `model`) lo linealice por medio de un apróximación de Taylor de primer orden, la cual es calculada alrededor del estado estacionario (order=1). Luego, usa esas aproximaciones para calcular la función impuslo-respuesta (IRF por sus siglas en inglés) y diversos estadísticos descriptivos (momentos, descomposición de varianza, coeficientes de correlación y autocorrelación). Además, esta sentencia le indica a Dynare que calcule la función impulso-respuesta con 30 periodos (irf = 30). El IRF es calculado como la diferencia entre la trayectoria de la variable ante un choque (en $t = 1$) y su estado estacionario. Dynare grafica el IRF solo para 12 variables.

[2]Este método construye aproximaciones de series de Taylor para la solución del modelo DSGE alrededor de su estado estacionario determinístico. Este método ha sido utilizado en física y en otras ciencias naturales; en economía fue popularizado por Judd y Guu (1993). Este método ha ganado popularidad en economía en las últimas dos décadas debido a tres razones:

- Es adecuado. El método de perturbación encuentra una solución apróximada que es local; es decir, que es muy adecuada alrededor del punto donde se toma la expansión de Taylor.
- El resultado es intuitivo y facilmente interpretable.
- Gracias al desarrollo de *software* como Dynare y Dynare++, el método de perturbación para mayores grados de expansión son fáciles de calcular y no requiere estar familiarizado con métodos numéricos.

Cuadro 8: Opciones de stoch_simul

[1] Solución del modelo	
order = entero	Indica el orden de la apróximación de Taylor. Los valores disponibles son 1, 2 y 3 (*default = 2).* **Ejemplo:** $stoch\_simul(order = 1)$;
loglinear	Transforma todas las variables en log-lineales. Por tanto, nos tenemos que asegurar que los estados estacionarios son estrictamente positivos. Todos los resultados (FIR, momentos, función de política, etc) son considerando que las variables son log-lineal. **Ejemplo:** $stoch\_simul(loglinear)$;
[2] Función impulso respuesta (FIR)	
irf = entero	Número de periodos para el cálculo de la FIR (*default = 40). Las FIR se guardan en "oo_.irfs".* **Ejemplo:** $stoch\_simul(order = 1, irf = 30)$;
irf_shocks = (nombre de la variable exógena)	Calcula la FIR para la variable exógena solicitada. Se usa cuando en el modelo existen varios choques. **Ejemplo:** $stoch\_simul(order = 1, irf\_shocks = (e))$;

El código "stoch_simul" brinda las funciones de política y de estado (conocidas también como reglas de decisión), cuyos coeficientes se guardan en "oo_.dr" (**dr** es un diminutivo para *decision rules*). Cabe mencionar que la función de política, en Dynare, tiene la siguiente estructura:

$$y_t = y_{ss} + Ax_t + Bu_t$$

Donde: y_{ss} es el vector del estado estacionario de las variables.

- **oo_.dr.ys:** guarda los estados estacionarios y_{ss}, cuyo orden es similar a como han sido declaradas las variable en el bloque `var`.
- **oo_.dr.ghx:** guarda la matriz "A". Las filas corresponden a toda las endógenas (en orden como están listadas en "oo_.dr.order_var"); mientras que las columnas corresponden a las variables de estado.
- **oo_.dr.ghu:** guarda la matriz "B". Las filas corresponden a toda las endógenas (en orden como están listadas en "oo_.dr.order_var"); mientras que las columnas corresponden a las variables exógenas.

3.9. Simulación y filtro HP

De la solución del sistema de ecuaciones no lineales se puede obtener el comportamiento en series de tiempo de cada una de las variables endógenas; por ejemplo, la función de estado del capital podría sugerir que el capital se comporta como un AR(2) o de la función de política del producto se puede deducir que el producto se comporta como un

ARMA(2,1). El camino para obtener la serie de tiempo de cada variable se describirá en detalle en el capítulo 3 y 4. Al tener la representación de series de tiempo se puede realizar una simulación de la variable. Para ello dos insumos son importantes: el número de periodos (meses, trimestres o años) que deseamos simular la variable, y el número de veces de simulación; por ejemplo, se podría desear simular 30 veces la misma variable.

Para realizar esta simulación en Dynare se usa dos comandos en "stoch_simul": `periods` y `simul_replic`. Ambas están descritas en el cuadro [9]. De otro lado si se desea que Dynare grafique alguna variable simulada se puede escribir, después de "stoch_simul" el código "rplot nombre_variable" y Dynare mostrará la gráfica.

Cuadro 9: Opciones (continuación) de stoch_simul

[3] Simulación de las variables endógenas	
periods = entero	Indica el número de periodos que se usará en la simulación de cada variable endógena (solo realiza **una** simulación). Dicha simulación es guardada en la matriz global *oo_.endo_simul*. Bajo esta opción los momentos empíricos serán calculados en lugar de los teóricos. **Ejemplo:** $stoch\_simul(order = 1, periods = 300)$;
simul_replic = entero	Esta opción permite simular las variables el número de veces que se indica en el "entero". Esta opción siempre va acompañada de la opción "*periods*". **Ejemplo:** $stoch\_simul(order = 1, periods = 300, simul\_replic = 150)$;. Este ejemplo indica que se debe simular 150 veces las variables para 300 periodos cada una. Cabe mencionar que estas simulaciones no se consideran para calcular los momentos empíricos; además, se guardan en "NombreMod_simul". El valor por *default* es uno.
rplot nombre_variable	Grafica las variables simuladas, que estan guardadas en "oo_.endo_simul". Este comando se coloca después de "stoch_simul", en el cual es necesario colocar *periods*. **Ejemplo:** stoch_simul(order = 1, periods=150); rplot c;
[4] Filtro HP	
hp_filter = entero	Usa el filtro HP con λ = entero (mensual:14400 ;trimestral:1600; anual:100) para calcular los momentos. **Ejemplo:** $stoch\_simul(order = 1, hp\_filter = 1600)$;

Asimismo, si se desea evaluar la habilidad del modelo en capturar el comportamiento del ciclo económico es necesario calcular los momentos teórico del componente cíclico de cada variable proveniente del modelo. En ese sentido es necesario aplicar un filtro que permita separar el ciclo de la tendencia. Para esta tarea, Dynare cuenta con el filtro HP, cuyo código es: `hp_filter = entero`, el cual se coloca dentro de "stoch_simul". El "`entero`" refleja el parámetro de suavizamiento, el cual varía en valor dependiendo de la frecuen-

cia (mensual, trimestral o anual). La elección del parámetro radica en la frecuencia en que los parámetros del modelo han sido calibrados; por ejemplo, si todos los parámetros del modelo (tasa de depreciación, factor de descuento de la utilidad, etc) han sido calibrados trimestralmente, entonces el parámetro de suavizamiento en el filtro HP debe ser trimestral. El cuadro [9] se describe en mayor detalle lo mencionado líneas arriba.

3.10. Análisis de sensibilidad

Es usual realizar análisis de sensibilidad del modelo ante un cambio en el valor de los parámetros. Por ejemplo, es útil comparar los IRFs del modelo ante dos valores distintos del parámetro de persistencia del choque. Para realizar este tipo de tareas, Dynare brinda una opción por medio de comandos "macro". Este macro-lenguaje de Dynare provee un conjunto de macro-comandos, los cuales se pueden insertar en el archivo ".mod". Las principales tareas que realiza este macro-lenguaje son: incluir un archivo en el .mod, sustitución de expresiones, estructuras condicionales (if) y bucles (for).

En esta subsección se hace énfasis en los bucles (*loops*) debido a que estos ayudan a realizar análisis de sensibilidad. En el cuadro [10] se hace una comparación del macro-lenguaje de Dynare y el código de Matlab. Ambos producen lo mismo, la principal diferencia entre ellos es que el vector que contiene los valores del parámetro no aparecerá en el ambiente de Matlab cuando corremos el macro-lenguaje, mientras que el código de Matlab reemplazará sucesivamente los valores del parámetro, mostrando el último valor en el *workspace* de Matlab y guardará el vector de valores del parámetro. Cabe mencionar que estos códigos se escriben después del "stoch_simul".

Cuadro 10: Macro-lenguaje vs Matlab para análisis de sensibilidad

Macro-lenguaje	Matlab
rhos = [0.8, 0.9, 1];	rhos = [0.8, 0.9, 1];
@#for i in 1:3	for i = 1:length(rhos)
rho = rhos(@i);	rho = rhos(i);
stoch_simul(order=1);	stoch_simul(order=1);
save oo_ = oo_;	save oo_ = oo_;
@#endfor	end

3.11. Formas de escribir el modelo en Dynare

Una ventaja de Dynare es que se puede escribir el mismo modelo en diferentes formas. En primer lugar se puede escribir el modelo nolineal y esperar que Dynare lo linealice o se puede escribir directamente el modelo linealizado por el usuario. En segundo lugar, la variable se puede introducir en niveles o en logarítmo. Esto es importante porque cuando Dynare linealice el sistema o el usuario escriba el sistema linealizado los coeficientes de la función de política y de estado se leen como elasticidades. En la siguiente sección se utilizará el modelo de Long y Plosser (1983) para ilustrar los comandos antes descritos, para ello se considera cuatro formar de escribir este modelo en Dynare. El objetivo de esto último es ver las diferencias entre ellos en cuanto a la solución, la función impulso-respuesta y los momentos.

En el cuadro [11] se describe los cuatro archivos .mod que reflejan las cuatro formas distintas de ingresar o escribir un modelo en Dynare. Como se mencionó previamente, los cuatro archivos contienen el mismo modelo.

Cuadro 11: Cuatro formas de escribir un modelo en Dynare

Modelo (.mod)	Descripción
Long_Plosser_Dynare_lineal_log.mod (mod1)	Este .mod contiene las ecuaciones log-lineales. Cabe resaltar que el valor de estado estacionario de cada variable log-lineal es igual a cero.
Long_Plosser_Dynare_lineal_niv.mod (mod2)	Este .mod contiene las ecuaciones lineales pero con las variables en niveles.
Long_Plosser_Dynare_nolineal_log.mod (mod3)	Este .mod contiene las ecuaciones no-lineales y con las variables en logarítmo.
Long_Plosser_Dynare_nolineal_niv.mod (mod4)	Este .mod contiene las ecuaciones no-lineales y con las variables en niveles.

4. Modelo de Long y Plosser (1983): aplicación en Dynare

4.1. Modelo de Long y Plosser (1983)

Con el fin de aplicar los códigos de Dynare antes descritos en la solución y simulación de un modelo de equilibrio general, en esta sección se utilizará el modelo de Long y Plosser (1983), el cual es descrito en detalle en el capítulo 3. Además, es preciso mencionar que este modelo tiene dos supuestos importantes: el primero es que el capital se deprecia totalmente en cada periodo, y el segundo es que la utilidad es logarítmica en el consumo y el ocio. En el cuadro [12] se describe el problema de optimización de la familia y de la empresa.

Cuadro 12: Problema de optimización de los agentes

Familias	**Empresas**
$\underset{\{c_t,h_t,k_{t+1}\}_{t=0}^{\infty}}{\text{Max}} \; E_0 \sum_{t=0}^{\infty} \beta^t \left[ln(c_t) + \theta ln(1-h_t)\right]$ $c_t + i_t = w_t h_t + r_t k_t + \pi_t$ $k_{t+1} = i_t$	$\underset{\{k_t,l_t\}_{t=0}^{\infty}}{\text{Max}} \; \pi_t = y_t - [w_t h_t + r_t k_t]$ $y_t = a_t k_t^{1-\alpha} h_t^{\alpha}$

En cada problema de optimización se obtiene condiciones de primer orden que reflejan el comportamiento de cada agente. En conjunto estas reglas de comportamiento conforman un sistema de ecuaciones no lineales estocástica, las cuales se describen en el cuadro [13].

Estas nolinealidades hace difícil su solución. El camino usual para reducir la complejidad de este sistema de ecuaciones es obteniendo una apróximación de primer orden por medio de la expansión de Taylor, la cual es llamada linealización. Como se menciona en el capítulo 3, existe dos formas de linealizar el sistema de ecuaciones. La primera es considerando la variable en niveles y la segunda es considerando la variable en logarítmo. El

Cuadro 13: Sistema de ecuaciones no lineal del modelo (Long y Plosser, 1983)

Agente	Ecuaciones	Descripción
Familia	$\frac{1}{c_t} = \beta E_t\left[\frac{1}{c_{t+1}} r_{t+1}\right]$	Ecuación de Euler
	$k_{t+1} = i_t$	Ley de movimiento del capital
	$\frac{\theta}{1-h_t} = \frac{w_t}{c_t}$	Oferta de trabajo
Empresa	$y_t = a_t k_t^{1-\alpha} h_t^{\alpha}$	Función de producción
	$r_t = (1-\alpha)\frac{y_t}{k_t}$	Demanda del capital
	$w_t = \alpha \frac{y_t}{h_t}$	Demanda de trabajo
Equilibrio	$y_t = c_t + i_t$	Equilibrio mercado de bienes
Choque	$lna_t = \phi lna_{t-1} + \epsilon_t$	Choque de productividad

cuadro [14] muestra las ecuaciones luego de linealizar el modelo considerando las variables en niveles, donde para el caso del consumo se tiene: $\widetilde{c}_t = c_t - c_{ss}$.

De otro lado, el cuadro [15] muestra las ecuaciones linealizadas considerando las variables en logarítmo. En este enfoque el cambio de variable, por ejemplo para el consumo, sigue la siguiente forma: $\widehat{c}_t = lnc_t - lnc_{ss}$.

Cuadro 14: Sistema de ecuaciones lineal del modelo (Long y Plosser, 1983)

Agente	Ecuaciones	Descripción
Familia	$\widetilde{c}_t = \beta E_t(r_{ss}\widetilde{c}_{t+1} - c_{ss}\widetilde{r}_{t+1})$	Ecuación de Euler
	$\widetilde{k}_{t+1} = \widetilde{i}_t$	Ley de movimiento del capital
	$\widetilde{w}_t = \frac{w_{ss}}{(1-h_{ss})}\widetilde{h}_t + \frac{\theta}{1-h_{ss}}\widetilde{c}_t$	Oferta de trabajo
Empresa	$\widetilde{y}_t = \frac{y_{ss}}{a_{ss}}\widetilde{a}_t + (1-\alpha)\frac{y_{ss}}{k_{ss}}\widetilde{k}_t + \alpha\frac{y_{ss}}{a_{ss}}\widetilde{h}_t$	Función de producción
	$\frac{y_{ss}}{k_{ss}}\widetilde{k}_t = \widetilde{y}_t - \frac{k_{ss}}{1-\alpha}\widetilde{r}_t$	Demanda del capital
	$\widetilde{w}_t = \left(\frac{\alpha}{h_{ss}}\right)\widetilde{y}_t - \left(\frac{\alpha y_{ss}}{h_{ss}^2}\right)\widetilde{h}_t$	Demanda de trabajo
Equilibrio	$\widetilde{y}_t = \widetilde{c}_t + \widetilde{i}_t$	Equilibrio mercado de bienes
Choque	$\widetilde{a}_t = \phi\widetilde{a}_{t-1} + \epsilon_t$	Choque de productividad

En el cuadro [16] se muestra los valores que toman los parámetros del modelo, la cual está basada en King y Rebelo (2000). Cabe mencionar que estos parámetros se han obtenido considerando que los datos son trimestrales. Por tanto, cada periodo en el modelo sea en la simulación como en la función impulso-respuesta se entiende como un trimestre.

En el cuadro [17] se menciona el estado estacionario de cada variable. Para calcular este equilibrio de largo plazo se asume que la variable es la misma independientemente de la temporalidad; es decir, $x_t = x_{t+1}$. En ese sentido, todos los rezagos y adelantos presentes en el sistema de ecuaciones que reflejan el modelo desaparece. Es en este escenario donde se calcula el estado estacionario para cada variable, el cual finalmente depende de los

Cuadro 15: Sistema de ecuaciones log-lineal del modelo (Long y Plosser, 1983)

	Ecuaciones log-lineal	Descripción
[1]	$\widehat{c}_t = E_t\left[\widehat{c}_{t+1} - \widehat{r}_{t+1}\right]$	Ecuación de Euler
[2]	$\widehat{k}_{t+1} = \widehat{i}_t$	Ley de movimiento del capital
[3]	$\frac{h_{ss}}{1-h_{ss}}\widehat{h}_t = \widehat{w}_t - \widehat{c}_t$	Oferta de trabajo
[4]	$\widehat{y}_t = \widehat{a}_t + (1-\alpha)\widehat{k}_t + \alpha\widehat{h}_t$	Función de producción
[5]	$\widehat{r}_t = \widehat{y}_t - \widehat{k}_t$	Demanda de capital
[6]	$\widehat{w}_t = \widehat{y}_t - \widehat{h}_t$	Demanda de trabajo
[7]	$\widehat{y}_t = \frac{c_{ss}}{y_{ss}}\widehat{c}_t + \frac{i_{ss}}{y_{ss}}\widehat{i}_t$	Equilibrio en el mercado de bienes
[8]	$\widehat{a}_t = \phi\widehat{a}_{t-1} + \epsilon_t$	Choque de productividad

Nota: Para obtener directamente la solución del modelo con Dynare se puede utilizar el archivo "Long_Plosser_Dynare_nolineal_log.mod"

parámetros del modelo. El detalle de cómo se llegó a cada expresión se encuentra en el capítulo 3.

Cuadro 16: Calibración

Parámetro	**Observación**
$\alpha = 0.667$	Proporción de largo plazo del trabajo en el ingreso nacional
$\theta = 3.968$	Calibrado para que el trabajo en estado estacionario sea igual a 20 %
$\rho = 0.979$	Persistencia del choque
$\beta = 0.984$	Factor de descuento
$\sigma_e = 0.0072$	Desviación estándar del choque de productividad

4.2. Preambulo

Definición de variables endógenas: en el cuadro [18] se describe la declaración de las variables endógenas en cada uno de los archivos .mod. De este cuadro se desprende cuatro conclusiones. La primera es que en el [mod1] cada variable declarada es la variable que aparece en el modelo no-lineal. Por ejemplo "c" representa el consumo en periodo "t".

La segunda es que en el [mod2] cada variable declarada representa el logarítmo neperiano de la variable. Por ejemplo "cc" es igual al "lnc_t". Cabe mencionar que el [mod1] y [mod2] contienen al modelo no-lineal. La tercera es que en el [mod3] cada variable declarada representa la desviación de la variable con respecto a su estado estacionario. Por ejemplo "ct" es igual al "$c_t - c_{ss}$". Cabe mencionar que "ct" es una forma de representar $\widetilde{c}_t$, tal como aparece en el modelo linealizado en niveles (ver el cuadro [14]).

Además, una cuarta conclusión es que en el [mod4] cada variable declarada representa la desviación del logarítmo de la variable con respecto al logarítmo de su estado estacionario. Por ejemplo "ch" es igual al "$lnc_t - lnc_{ss}$". De igual forma que en el [mod3], "ch" es

Cuadro 17: Estado estacionario

Estado estacionario (forma recursiva)	Estado estacionario (forma paramétrica)
$r_{ss} = \frac{1}{\beta}$	$= \frac{1}{\beta}$
$h_{ss} = \frac{\alpha}{\theta(1-\beta(1-\alpha))+\alpha}$	$= \frac{\alpha}{\theta(1-\beta(1-\alpha))+\alpha}$
$a_{ss} = 1$	$= 1$
$k_{ss} = h_{ss}\left[\frac{1}{\beta(1-\alpha)}\right]^{-1/\alpha}$	$= \left[\frac{\alpha}{\theta(1-\beta(1-\alpha))+\alpha}\right]\left[\beta(1-\alpha)\right]^{1/\alpha}$
$i_{ss} = k_{ss}$	$= \left[\frac{\alpha}{\theta(1-\beta(1-\alpha))+\alpha}\right]\left[\beta(1-\alpha)\right]^{1/\alpha}$
$y_{ss} = k_{ss}\left[\frac{1}{\beta(1-\alpha)}\right]$	$= \left[\frac{\alpha}{\theta(1-\beta(1-\alpha))+\alpha}\right]\left[\beta(1-\alpha)\right]^{\frac{1}{\alpha}-1}$
$c_{ss} = k_{ss}\left[\frac{1}{\beta(1-\alpha)} - 1\right]$	$= \left[\frac{\alpha}{\theta(1-\beta(1-\alpha))+\alpha}\right]\left[\beta(1-\alpha)\right]^{1/\alpha}\left[\frac{1}{\beta(1-\alpha)} - 1\right]$
$w_{ss} = \alpha\frac{y_{ss}}{h_{ss}}$	$= \alpha\left[\beta(1-\alpha)\right]^{\frac{1}{\alpha}-1}$

Nota: El cálculo de los estados estacionarios se encuentran en Long_Plosser.m (sección 2) (ver el capítulo 3).

una forma de representar $\widehat{c}_t$, tal como aparece en el modelo log-linealizado (ver el cuadro [15]).

Asimismo, es importante mencionar que el número de variables declaradas es el mismo que el número de ecuaciones que se escribirá en el bloque `model`. Finalmente, la productividad en Dynare se declara como una variable endógena y es el choque ϵ_t el que se declara como exógena.

Cuadro 18: Declaración de variables endógenas

Modelo no-lineal		Modelo lineal	
Variable en niveles (mod1)	Variable en logarítmo (mod2)	Variable en niveles (mod3)	Variable en logarítmo (mod4)
var	var	var	var
c	cc	ct	ch
i	ii	it	ih
y	yy	yt	yh
k	kk	kt	kh
h	hh	ht	hh
r	rr	rt	rh
w	ww	wt	wh
a	aa	at	ah
;	;	;	;

Definición de variables exógenas: la única variable exógena es la perturbación (error) de la productividad ϵ_t. La forma de introducirlo en el archivo .mod es similar entre las cuatro versiones.

varexo e e_t (long_name = 'Choque de productividad');

Donde: e_t es el nombre que tomará a variable en formato LaTeX, y (long_name = 'Choque de productividad') es el nombre "largo" que se asigna a la variable.

Definición de parámetros: en el cuadro [19] se describe la definición de los parámetros, la cual es similar en los cuatro archivos .mod. Cabe mencionar que no solo se indica los parámetros asociados a las ecuaciones como la función de producción por ejemplo, sino que también se define como parámetro a los valores de estado estacionario.

Cuadro 19: Declaración de los parámetros

parameters	
theta	$ \theta$ (long_name = 'peso del ocio en la función de utilidad')
beta	$ \beta$ (long_name = 'factor de descuento')
alpha	$ \alpha$ (long_name = 'participación del trabajo en el ingreso nacional')
rho	$ \rho$ (long_name = 'persistencia del choque')
sigma_ee	$ \sigma_e$ (long_name = 'des. est. del choque')
y_ss	
c_ss	
i_ss	
w_ss	
r_ss	
k_ss	
h_ss	
a_ss	
;	

Nota: Esta declaración de parámetros pertenece al Mod1.

Luego de definir los parámetros es necesario indicarle a Dynare los valores de cada parámetro (calibración) incluyendo los valores del estado estacionario. Esto es importante porque en el modelo lineal usualmente aparece de forma multiplicativa o aditiva el valor del estado estacionar io de algunas variables y además porque estos estados estacionarios se colocan en el bloque de valores iniciales. En el cuadro [20] se describe la forma de introducir los valores de los parámetros en Dynare. Estos códigos se escriben después del bloque de parámetros y antes del bloque del modelo.

¿Donde guarda Dynare la información acerca de las variables y los parámetros? Dynare, luego de leer el archivo .mod, crea una variable en el *workspace* de Matlab: M_, en la cual guarda información del modelo. Bajo la categorización de las variables de Matlab, esta variable es una **estructura**; es decir, puede contener otras variables, como numéricas (matrices y vectores), lógicas y de cadena de caracteres (texto), inclusive puede contener otra estructura.

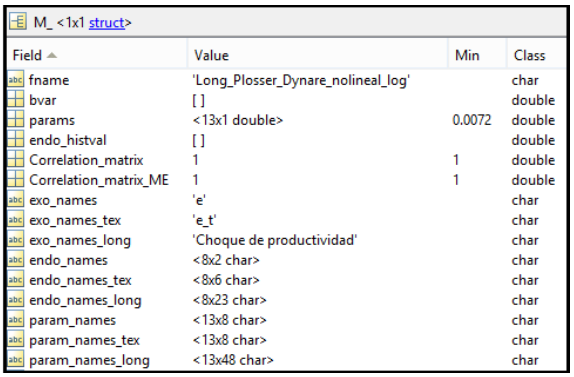
En la figura [4] se observa que la variable M_ contiene un conjunto amplio de otras variables. En este apartado se menciona aquellas en que Dynare guarda el nombre de las variables y de los parámetros del modelo. Cabe mencionar que dado que estas variables de Matlab guardan texto (nombres), entonces bajo la tipologia de Matlab son variables de "cadena de caracteres (char)".

- **Variable asociada al nombre del modelo:** "fname" es una variable la cual contiene el nombre del archivo .mod.
- **Variables asociadas a la variable exógena:** en este caso existen tres variables.

Cuadro 20: Declaración de los valores de los parámetros

Para (mod1) al (mod4)
h_ss=0.2;
beta=0.984;
alpha=0.667;
rho=0.979;
sigma_ee=0.0072;
theta=alpha*(1-h_ss)/(h_ss*(1-beta*(1-alpha)));
r_ss=1/beta;
a_ss=1;
k_ss=h_ss*(1/(beta*(1-alpha)))^(-1/alpha);
i_ss=k_ss;
y_ss=k_ss*(1/(beta*(1-alpha)));
c_ss=k_ss*(1/(beta*(1-alpha))-1);
w_ss=alpha*y_ss/h_ss;

Figura 4: Estructura M_



Nota: Esta estructura M_ se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

La primera es "exo_names", la cual contiene el nombre de la variable exógena. La segunda es "exo_names_tex", que contiene el nombre que la variable tomará en formato LaTeX; finalmente, "exo_names_long" que contiene los nombres extendidos de las variables exógenas.

- **Variables asociadas a la variable endógena:** al igual que en el caso de la variable exógena, también existen tres variables para la variable endógena. La primera es "endo_names", la cual contiene el nombre de las variables endógenas. La segunda es "endo_names_tex", que contiene el nombre que la variables tomará en formato LaTeX; finalmente, "endo_names_long" que contiene los nombres extendidos de las variables endógeas.

- **Variables asociadas a los parámetros:** en este caso existen cuatro variables. Una de ellas es "params", la cual contiene los valores de los parámetros (en el mismo orden con que se ha escrito en el bloque `parameters` en el .mod); las tres variables restantes

están asociadas a los nombres de los parámetros: la primera es "param_names", la cual contiene el nombre de los parámetros. La segunda es "param_names_tex", que contiene el nombre que los parámetros tomará en formato LaTeX; finalmente, "param_names_long" que contiene los nombres extendidos de los parámetros.

Cabe mencionar que para extraer alguna variable que se encuentra dentro de la estructura M_ basta escribir en el *Command Window* de Matlab: "M_.NombreVariable". Por ejemplo si se desea extraer el nombre del archivo .mod se escribe lo siguiente:

>> M_.fname

Esto mostrará:

ans =
Long_Plosser_Dynare_nolineal_log

Quiza se requiera extraer la versión larga de los nombres de las variables endógenas, entonces se escribe en el *Command Window* de Matlab lo siguiente:

>> M_.endo_names_long

Esto mostrará:

ans =
Ln Consumo
Ln Inversión
Ln Producto
Ln Capital
Ln Trabajo
Ln Tasa de interés real
Ln Salario real
Ln Productividad

Es importante mencionar que una de las variables que tiene la estructura M_ es Sigma_e. Esta es una variable especial de Dynare y no se puede usar este nombre para definir otra variable en el archivo .mod (por ejemplo la desviación estándar del choque). Sigma_e es la matriz de varianza-covarianza del choque estocástico y se escribe como una matriz triangular superior o inferior. Por ejemplo para el modelo no-lineal con variable en logarítmo (mod2), Sigma_e = 0.00005184. Lo cual corresponde a la varianza del choque. Cómo en el archivo .mod se ha definido que la desviación estándar del choque es sigma_ee = 0.0072, entonces la varianza es 0.00005184, lo cual Dynare calcula y guarda en Sigma_e (variable de la estructura M_).

4.3. Modelo

En el cuadro [21] se menciona la forma de escribir en Dynare el modelo no-lineal considerando las variables en niveles o variables en logarítmo. De este cuadro se desprende algunos comentarios generales que son transversales para cualquier modelo:

- **Definición del tiempo:** es importante mencionar cómo Dynare considera el "tiempo". En primer lugar cuando se escribe una variable en Dynare en "t" solo se coloca la variable. Por ejemplo "c" representa c_t. Si se desea escribir una variable adelantada en un periodo se escribe "c(+1)", la cual representa c_{t+1}.

- **Variables de control:** Dynare considera que las variable de control están escritas en "t" y que las variables acompañadas por (+1) son *forward looking*. En este caso no es necesario escribir las expectativas porque Dynare entiende que cualquier variable escrita en (+1) siempre lleva acompañado el operador de expectativas E_t. Por ejemplo, la ecuación de Euler descrita en el cuadro [21] no lleva el operador expectativas.

- **Variable de estado:** es importante mencionar que el capital, en este modelo, es una variable de estado; es decir, en "t" ya está determinada. Entonces el capital en "t+1" es k_{t+1}, la cual fue determinada en "t". Por tanto, cuando se escriba en Dynare se debe de colocar "k" para representar k_{t+1}, y cuando en una ecuación aparezca k_t se debe de escribir en Dynare como "k(-1)". Esto se puede observar en la demanda de capital y la función de producción en el cuadro [21]. Por ejemplo: la función de producción es $y_t = a_t k_t^{\alpha-1} h_t^{\alpha}$, la cual se escribe en Dynare como: y=a*((k(-1))^(1-alpha))*h^(alpha).

Cuadro 21: Declaración del modelo no-lineal

Variables en niveles (mod1)	Variables en logarítmo (mod2)
model;	model;
1/c=beta*(1/c(+1))*(r(+1));	1/exp(cc)=beta*(1/exp(cc(+1)))*(exp(rr(+1)));
k=i;	exp(kk)=exp(ii);
theta/(1-h)=w/c;	theta/(1-exp(hh))=exp(ww)/exp(cc);
y=a*((k(-1))^(1-alpha))*h^(alpha);	exp(yy)=exp(aa)*((exp(kk(-1)))^(1-alpha))*exp(hh)^(alpha);
r=(1-alpha)*y/k(-1);	exp(rr)=(1-alpha)*exp(yy)/exp(kk(-1));
w=(alpha)*y/h;	exp(ww)=(alpha)*exp(yy)/exp(hh);
y=c+i;	exp(yy)=exp(cc)+exp(ii);
ln(a)=rho*ln(a(-1))+e;	aa=rho*aa(-1)+e;
end;	end;

Además de lo anterior, del cuadro [21] se pueden desprender algunas conclusiones específicas:

- **Modelo no-lineal con variables en niveles (mod1):** en este caso la variable que representa el consumo es "c", lo cual es similar para las demás variables. Las ecuaciones están escritas de la misma forma que en el *papel*; es decir, son las relaciones de primer orden no lineales que surgen de la optimización. Cuando a Dynare se le pida que realice la linealización creará la variable $x = x - x_{ss}$.

- **Modelo no-lineal con variables en logarítmo (mod2):** en este caso en Dynare se define la variable "ln x" como "xx". Esto se realiza con el fin de considerar las variables en logarítmos. Entonces, en cada ecuación del sistema no-lineal en lugar de escribir "x" se reescribe dicha variable como "exp(ln x)", la cual brinda el mismo "x".

Pero se sabe que "xx = ln x", entonces "exp(ln x)" se convierte en "exp(xx)". Esta última expresión es lo que se escribe en Dynare para cada variable. Cuando a Dynare se le pida que realice la linealización creará la variable $xx = xx - xx_{ss} = lnx - lnx_{ss}$.

Con respecto a la versión lineal del modelo, el cuadro [22] muestra las dos alternativas de linealizar el modelo: en niveles (mod3) o en logarítmo (mod4). Cabe mencionar que esta linealización la realiza previamente el usuario y luego escribre el modelo linealizado en Dynare; en este caso Dynare ya no aplicará la apróximación de primer orden de Taylor sobre el modelo. Esto es diferente a los dos modelos previos donde el usuario escribía el modelo no-linal y solo cambiada la naturaleza de la variable (lineal o logarítmica). Del cuadro [22] se desprende algunas consideraciones:

- **Modelo lineal:** cuando se desea escribir un modelo linealizado en Dynare se debe de colocar la opción `linear` en el bloque `model`. Esto se hace de la siguiente manera: `model (linear); . . . end;`
- **Modelo lineal con variable en niveles (mod3):** en este caso se ha definido la variable $xt = x - x_{ss}$, esta variable es la que representa la variable $\widetilde{x}_t$ del cuadro [14].
- **Modelo lineal con variable en logarítmo (mod4):** en este caso se ha definido la variable $xh = lnx - lnx_{ss}$, esta variable es la que representa la variable $\widehat{x}_t$ del cuadro [15].

Una diferencia importante de las ecuaciones entre el modelo no-lineal y linealizado es que en este último los valores de estado estacionario están presentes en las ecuaciones.

Cuadro 22: Declaración del modelo lineal

Variables en niveles (mod3)	Variables en logarítmo (mod4)
model(linear);	model(linear);
ct=beta*(r_ss*ct(+1)-c_ss*rt(+1));	ch=ch(+1)-rh(+1);
kt=it;	(h_ss/(1-h_ss))*hh=wh-ch;
wt=(w_ss/(1-h_ss))*ht+(theta/(1-h_ss))*ct;	kh=ih;
yt=(y_ss/a_ss)*at+(1-alpha)*(y_ss/k_ss)*kt(-1) +alpha*(y_ss/h_ss)*ht;	yh=ah+(1-alpha)*kh(-1)+alpha*hh;
(y_ss/k_ss)*kt(-1)=yt-(k_ss/(1-alpha))*rt;	rh=yh-kh(-1);
wt=(alpha/h_ss)*yt-((alpha*y_ss)/(h_ss)^2)*ht;	wh=yh-hh;
yt=ct+it;	yh=(c_ss/y_ss)*ch+(i_ss/y_ss)*ih;
at=rho*at(-1)+e;	ah=rho*ah(-1)+e;
end;	end;

4.4. Valores iniciales

Los valores iniciales son importantes porque son el punto de partida que Dynare utiliza para calcular el estado estacionario por medio de aproximaciones sucesivas. Usualmente primero calculamos el estado estacionario manualmente para luego introducirlo en el bloque de valores iniciales. El cuadro [23] contiene lo forma de ingresar los valores iniciales en Dynare según el tipo de modelo que estamos utilizando. De este cuadro se desprende las siguientes conclusiones:

- **Modelo no-lineal con variables en niveles:** en esta forma de escribir el modelo, el valor inicial de cada variable es el estado estacionario que previamente se ha definido en el bloque de parámetros "`parameters`", y que luego se ha calculado (ver el cuadro [20]).
- **Modelo no-lineal con variables en logarítmo:** debido a que la variable que se ha definido es el logarítmo de ella misma (xx = ln x), entonces en los valores iniciales se coloca "$xx = lnx_{ss}$". Cabe mencionar que en Matlab el logarítmo neperiano (ln) se escribe como "log".
- **Modelo lineal con variables en niveles:** dado que la variable definida es "$xt = x - x_{ss}$", entonces en el bloque de valores iniciales se coloca: $xt = x_{ss} - x_{ss}$, entonces $xt = 0$.
- **Modelo lineal con variables en logarítmo:** de manera similar al caso anterior, dado que la variable definida es "$xh = lnx - lnx_{ss}$", entonces el estado estacionario: "$xh = lnx_{ss} - lnx_{ss}$", lo cual lleva a que $xh = 0$, siendo este valor lo que se coloca en el bloque de valores iniciales para todas las variables.

Cuadro 23: Declaración de los valores iniciales

Modelo no-lineal				Modelo lineal			
Variables en niveles (mod1)		Variables en logarítmo (mod2)		Variables en niveles (mod3)		Variables en logarítmo (mod4)	
initval;		initval;		initval;		initval;	
h	=h_ss;	hh	=log(h_ss);	ht	=0;	hh	=0;
k	=k_ss;	kk	=log(k_ss);	kt	=0;	kh	=0;
i	=i_ss;	ii	=log(i_ss);	it	=0;	ih	=0;
c	=c_ss;	cc	=log(c_ss);	ct	=0;	ch	=0;
w	=w_ss;	ww	=log(w_ss);	wt	=0;	wh	=0;
r	=r_ss;	rr	=log(r_ss);	rt	=0;	rh	=0;
y	=y_ss;	yy	=log(y_ss);	yt	=0;	yh	=0;
a	=a_ss;	aa	=log(a_ss);	at	=0;	ah	=0;
end;		end;		end;		end;	

Resid: este comando, que se coloca después del bloque de valores iniciales, calcula el residuo en cada ecuación cuando se reemplaza cada variable por su valor inicial; es decir, coloca los valores iniciales en cada ecuación y calcula el residuo entre la expresión de la derecha menos la expresión de la izquierda; por ejemplo para la función de producción:

$$F = y_t - a_t k_t^{1-\alpha} h_t^{\alpha}$$

En estado estacionario:

$$F = y - ak^{1-\alpha} h^{\alpha}$$

Lo que hace Dynare es colocar los valores iniciales en esta ecuación y calcular el residuo (F). Si este residuo es cero significa que los valores inciales que se ha introducido son exactamente los correctos; es decir, hemos calculado correctamente el estado estacionario.

Puede ser que el residuo sea diferente de cero, lo cual indica que nos hemos equivocado en el cálculo del estado estacionario, y aún así Dynare encuentre el verdadero valor del estado estacionario. Esto se debe a que Dynare necesita un punto de partida cercano al verdadero valor del estado estacionario y con ello empezar a iterar. Luego de calcular el estado estacionario, Dynare mostrará dichos valores en el *prompt* de Matlab, el cual se encuentra en cuadro [24].

Además, el cuadro [24], que corresponde al modelo no-lineal con variables en logarítmo, indica que los valores iniciales (estado estacionario calculado) que se ha considerado son exactamente los estados estacionarios correctos del modelo; por ello, el residuo de cada ecuación es igual a cero. Si no hemos colocado bien los valores iniciales Dynare mostrará que el residuo es diferente de cero en algunas ecuaciones, lo cual nos brinda información para detectar en qué variable no hemos calculado el estado estacionario correctamente. Si el residuo es muy grande significa que los valores iniciales son muy ditintos o muy lejanos del estado estacionario y Dynare podría parar el proceso debido a que no logrará encontrar el estado estacionario desde el punto inicial dado.

Cuadro 24: El comando Resid: resultados

Residuals of the static equations:
Equation number 1 : 0
Equation number 2 : 0
Equation number 3 : 0
Equation number 4 : 0
Equation number 5 : 0
Equation number 6 : 0
Equation number 7 : 0
Equation number 8 : 0

4.5. Estado estacionario

El cuadro [25] muestra los resultados de aplicar el comando `steady;`. De este cuadro se desprende algunas conclusiones:

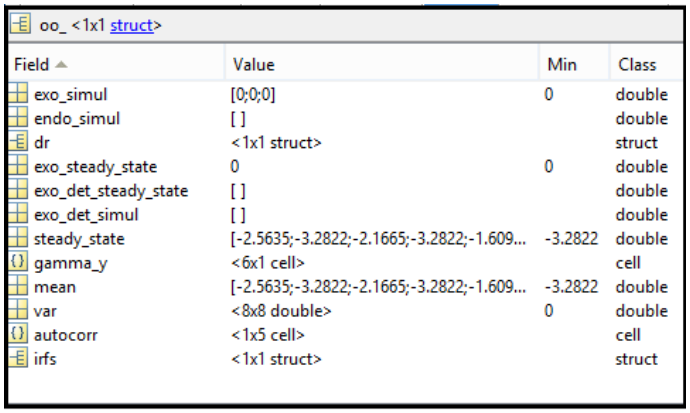
- **Modelo no-lineal con variables en logarítmo (mod2):** recordemos que en este modelo las variables están expresadas en logarítmo. Por ejemplo, el logarítmo del consumo está representado por "cc"; es decir, $cc = lnc$. Entonces en estado estacionario se tiene que $cc_{ss} = lnc_{ss}$. Considerando que $cc_{ss} = -2.56348$, entonces $c_{ss} = exp(-2.56348) = 0.0770361$. De la misma manera se hace con las demás variables.
- **Modelo lineal:** en caso del modelo lineal con variables en niveles (mod3), se sabe que cada variable está expresada como la diferencia entre su nivel y su estado estacionario. Por ejemplo, para el consumo se tiene $ct = c - c_{ss}$. Evaluando la variable en estado estacionario se tiene: $ct_{ss} = c_{ss} - c_{ss} = 0$. De manera similar se tiene para el modelo lineal con variables en logarítmo (mod4). Por ejemplo, para el consumo se tiene: $ch = lnc - lnc_{ss}$. Al evaluar esta variable en estado estacionario: $ch_{ss} = c_{ss} - c_{ss} = 0$.

Cuadro 25: Estado Estacionario

Modelo No-Lineal				Modelo Lineal			
Variables en niveles (mod1)		variables en logarítmo (mod2)		Variables en niveles (mod3)		Variables en logarítmo (mod4)	
c	0.0770361	cc	-2.56348	ct	0	ch	0
i	0.037545	ii	-3.28221	it	0	ih	0
y	0.114581	yy	-2.16647	yt	0	yh	0
k	0.037545	kk	-3.28221	kt	0	kh	0
h	0.2	hh	-1.60944	ht	0	hh	0
r	1.01626	rr	0.0161294	rt	0	rh	0
w	0.382128	ww	-0.962	wt	0	wh	0
a	1	aa	0	at	0	ah	0

¿Dónde guarda los estado estacionarios Dynare?
Dynare crea una variable estructura (similar a M_) llamada oo_, en la cual guarda las simulaciones, el estado estacionario, los momentos de las variables endógenas (media, varianza y autocorrelación) y la función impulso-respuesta de cada variable. En la figura [5] se muestra todas las variables que tiene la estructura oo_.

Figura 5: Estructura oo_



Nota: Esta estructura oo_ se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

En particular, la variable que contiene los estados estacionarios calculados por Dynare es "oo_.steady_state" (ver la figura [6]).

4.6. Definición del choque

Para los cuatro archivos .mod se define de manera similar el choque. El cuadro [26] muestra cómo se escribe en Dynare la varianza del choque de productividad.

Figura 6: oo_.steady_state

oo_.steady_state <8x1 double>

	1	2
1	-2.5635	
2	-3.2822	
3	-2.1665	
4	-3.2822	
5	-1.6094	
6	0.0161	
7	-0.9620	
8	0	

Nota: Esta variable oo_.steady_state se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

Cuadro 26: Definición del choque

```
shocks;
var e = (sigma_ee)^2;
end;
```

4.7. Evaluación del modelo

Para evaluar el modelo se coloca el código `check;`, el resultado de este comando es el vector de eigenvalores de la matriz F. Esta matriz se obtiene de escribir el modelo (sistema de ecuaciones lineales) en forma de estado-espacio. La ecuación (1) refleja la versión estado-espacio del modelo. En el capítulo 3 se describe cómo se obtiene esta ecuación en mayor detalle.

$$\begin{bmatrix} X_{t+1} \\ E_t Y_{t+1} \end{bmatrix} = F \begin{bmatrix} X_t \\ Y_t \end{bmatrix} + GV_{t+1} \tag{1}$$

En el cuadro [27] se escribe el vector de eigenvalores de cada modelo; además, Dynare muestra un mensaje que indica si el modelo cumple la condición de Blanchard y Kahn. Por ejemplo en el mod1 se observa que existen cuatro eigenvalores cuyos módulos son: 0.333, 0.979, 3.052 y ∞. De estos cuatro módulos, dos son mayores a uno. De otro lado, el modelo tiene dos variables *forward looking* c_{t+1} y r_{t+1}. Por tanto se cumple la condición de Blanchard y Kahn la cual indica que si el número de eigenvalores cuyo módulo es mayor a uno es igual al número de variables *forward looking*, entonces el sistema tiene solución única. Este es el mensaje que Dynare imprime en pantalla.

¿Dónde guarda Dynare el vector de eigenvalores?
Dentro de la estructura oo_ que Dynare crea cuando procesa el modelo se encuentra la estructura oo_.dr. Esta estructura guarda dos variables importantes: los eigenvalores y la regla de decisión; es decir, la solución del modelo. En la figura [7] se muestra las variables que contiene oo_.dr.

Cuadro 27: Eigenvalores

Modelo No-Lineal					
Variables en niveles (mod1)			Variables en logaritmos (mod2)		
Modulo	Real	Imaginario	Modulo	Real	Imaginario
0.333	0.333	0	0.333	0.333	0
0.979	0.979	0	0.979	0.979	0
3.052	3.052	0	3.052	3.052	0
Inf	-Inf	0	9.84E+15	-9.84E+15	0
Hay dos eigenvalores mayores que 1 en módulo para 2 variables forward-looking La condición de rango es verificada			Hay dos eigenvalores mayores que 1 en módulo para 2 variables forward-looking La condición de rango es verificada		
Modelo Lineal					
Variables en niveles (mod3)			Variables en logaritmos (mod4)		
Modulo	Real	Imaginario	Modulo	Real	Imaginario
0.333	0.333	0	0.333	0.333	0
0.979	0.979	0	0.979	0.979	0
3.052	3.052	0	3.052	3.052	0
4.59E+17	4.59E+17	0	Inf	-Inf	0
Hay dos eigenvalores mayores que 1 en módulo para 2 variables forward-looking La condición de rango es verificada			Hay dos eigenvalores mayores que 1 en módulo para 2 variables forward-looking La condición de rango es verificada		

La figura [8] muestra la variable oo_.dr.eigval, la cual contiene el vector de los eigenvalores.

4.8. Solución

En el cuadro [28] se muestra la solución del modelo (función de política y de estado) para los cuatro archivos .mod. Cabe mencionar que el código utilizado para obtener la solución es:

$$stoch\_simul(order = 1, irf = 40);$$

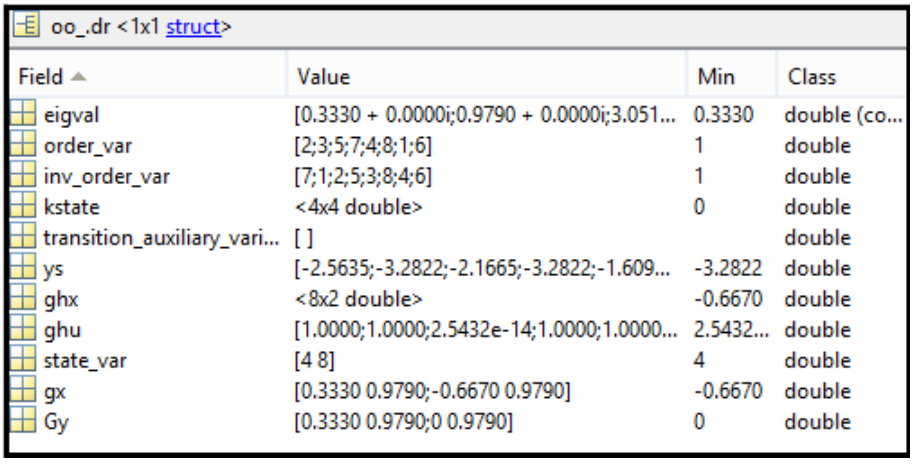
Donde: "order = 1" indica que el modelo debe ser aproximado por la expansión de Taylor de primer orden. Esta opción no opera cuando en el bloque del modelo se ha especificado que el modelo es lineal por medio de "`model(linear)`". Algunas observaciones emergen del cuadro [28]:

- **Modelo no-lineal con variable en niveles (mod1):** para el consumo "c" la solución es la siguiente:

$$c = 0.077 + 0.683k(-1) + 0.075a(-1) + 0.077\epsilon$$

Debido a que se le ha pedido a Dynare que linealice el sistema (`order=1`), lo cual permite obtener un sistema de ecuaciones como el cuadro [14], entonces la variable

Figura 7: Reglas de decisión (oo_.dr)

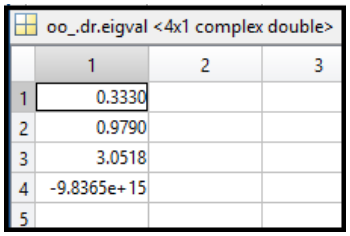


oo_.dr <1x1 struct>

Field	Value	Min	Class
eigval	[0.3330 + 0.0000i;0.9790 + 0.0000i;3.051...	0.3330	double (co...
order_var	[2;3;5;7;4;8;1;6]	1	double
inv_order_var	[7;1;2;5;3;8;4;6]	1	double
kstate	<4x4 double>	0	double
transition_auxiliary_vari...	[ ]		double
ys	[-2.5635;-3.2822;-2.1665;-3.2822;-1.609...	-3.2822	double
ghx	<8x2 double>	-0.6670	double
ghu	[1.0000;1.0000;2.5432e-14;1.0000;1.0000...	2.5432...	double
state_var	[4 8]	4	double
gx	[0.3330 0.9790;-0.6670 0.9790]	-0.6670	double
Gy	[0.3330 0.9790;0 0.9790]	0	double

Nota: Esta estructura oo_.dr se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

Figura 8: oo_.dr.eigval



oo_.dr.eigval <4x1 complex double>

	1	2	3
1	0.3330		
2	0.9790		
3	3.0518		
4	-9.8365e+15		
5			

Nota: Este vector oo_.dr.eigval se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

de estado k(-1) y exógena a(-1) están expresadas como desviaciones de su estado estacionario. Es decir: $k(-1) = k_t - k_{ss}$ y $a(-1) = a_{t-1} - a_{ss}$, entonces:

$$c_t = 0.077 + 0.683(k_t - k_{ss}) + 0.075(a_{t-1} - a_{ss}) + 0.077\epsilon_t$$

Cabe mencionar que la constante 0.077 es el valor de estado estacionario del consumo.

$$(c_t - 0.077) = 0.683(k_t - k_{ss}) + 0.075(a_{t-1} - a_{ss}) + 0.077\epsilon_t$$

Factorizando 0.077 de $(a_{t-1} - a_{ss})$ y e_t, se tiene:

$$(c_t - 0.077) = 0.683(k_t - k_{ss}) + 0.077(0.979(a_{t-1} - a_{ss}) + e_t)$$

Se sabe que en el modelo no-lineal la ecuación de la productividad es $lna_t = \rho lna_{t-1} + \epsilon_t$; sin embargo, al solicitar a Dynare que linealice el sistema, esta ecuación es transformada en: $\widetilde{a}_t = 0.979\widetilde{a}_{t-1} + e_t$, donde $\widetilde{a}_t = a_t - a_{ss}$. Por tanto reemplazando esta expresión en la ecuación anterior:

$$\underbrace{(c_t - 0.077)}_{\widetilde{c}_t} = 0.683\widetilde{k}_t + 0.077\widetilde{a}_t$$

$$\widetilde{c}_t = 0.683\widetilde{k}_t + 0.077\widetilde{a}_t$$

Esta es la función de política del consumo (expresada en desviaciones de su estado estacionario): $\widetilde{c}_t = F(\widetilde{k}_t, \widetilde{a}_t)$. Los coeficientes se leen de la siguiente manera: un incremento en una unidad en $\widetilde{k}_t$ (manteniendo lo demás constante) produce un incremento de $\widetilde{c}_t$ en 0.683 unidades; es decir, si el capital de hoy se aleja en una unidad con respecto a su estado estacionario, el consumo se aleja en 0.683 unidades con respecto a su estado estacionario. Estos coeficientes permiten calcular en unidades la desviación de las variables con respecto a su estado estacionario; sin embargo, una medida más apropiada sería considerar dicha desviación en términos porcentuales. Para ello se puede realizar lo siguiente:

$$\begin{aligned}
(c_t - 0.077) &= 0.683(k_t - k_{ss}) + 0.077(a_t - a_{ss}) + \epsilon_t \\
\frac{(c_t - 0.077)}{c_{ss}} c_{ss} &= 0.683\frac{(k_t - k_{ss})}{k_{ss}} k_{ss} + 0.077\frac{(a_t - a_{ss})}{a_{ss}} a_{ss} + \epsilon_t \\
\widehat{c}_t c_{ss} &= 0.683\widehat{k}_t k_{ss} + 0.077\widehat{a}_t a_{ss} + \epsilon_t \\
\widehat{c}_t c_{ss} &= 0.683 k_{ss}\widehat{k}_t + 0.077 a_{ss}\widehat{a}_t + \epsilon_t \\
\widehat{c}_t 0.077 &= 0.683 * 0.0375\widehat{k}_t + 0.077 * 1\widehat{a}_t + \epsilon_t \\
\widehat{c}_t 0.077 &= 0.0256\widehat{k}_t + 0.077\widehat{a}_t + \epsilon_t \qquad (2)
\end{aligned}$$

De la ecuación (2) la variable $\widehat{x}_t$ se lee como la desviación **porcentual** de la variable con respecto a su estado estacionario. Entonces: un incremento en 1 % en $\widehat{k}_t$; es decir, que el capital se esta incrementando en 1 % con respecto a su estado estacionario, produce un incremento de (0.0256/0.077)*1 % = 0.333 % de $\widehat{c}_t$; es decir, que el consumo se desvía por encima de su estado estacionario en un 0.333 %.

- **Modelo no-lineal con variable en logarítmos (mod2):** en este caso la solución, por ejemplo, para el consumo es:

$$cc = -2.563 + 0.333kk(-1) + 0.979aa(-1) + e$$

Considerando que Dynare ha realizado la linealización tomando en cuenta que cada variable está expresada en logarítmo, entonces $cc_t = lnc_t$, pero la variable de estado y la variable exógena están expresadas como: $kk(-1) = lnk_t - lnk_{ss}$ y $aa(-1) = lna_{t-1} - lna_{ss}$.

$$lnc_t = -2.563 + 0.333(lnk_t - lnk_{ss}) + 0.979ln(lna_{t-1} - lna_{ss}) + e_t$$

La constante de esta ecuación corresponde al estado estacionario de la variable; es decir, $lnc_{ss} = -2.563$, entonces: $c_{ss} = exp(-2.563) = 0.077$, lo cual coincide con lo calculado en el primer modelo (mod1). Introduciendo esto en la ecuación previa:

$$
\begin{aligned}
lnc_t &= -2.563 + 0.333(lnk_t - lnk_{ss}) + 0.979(lna_{t-1} - lna_{ss}) + e_t \\
lnc_t - lnc_{ss} &= 0.333(lnk_t - lnk_{ss}) + 0.979(lna_{t-1} - lna_{ss}) + e_t \\
\widehat{c}_t &= 0.333\widehat{k}_t + \underbrace{0.979\widehat{a}_{t-1} + e_t}_{\widehat{a}_t} \\
\widehat{c}_t &= 0.333\widehat{k}_t + \widehat{a}_t \qquad (3)
\end{aligned}
$$

En este caso los coeficientes son elasticidades; es decir, si el capital aumenta en 1 % con respecto a su estado estacionario (manteniendo lo demás constante), el consumo se incrementa en 0.333 % con respecto a su estado estacionario.

- **Modelo lineal:** con respecto al modelo lineal, se puede observar que tiene los mismos coeficientes que en el modelo no-lineal, el cual fue linealizado por Dynare. La principal diferencia es que cuando introducimos un modelo lineal en Dynare el estado estacionario es cero, por ello no aparece ningún intercepto en las ecuaciones del modelo lineal (mod3 y mod4).

Cuadro 28: Función de política y de estado

Modelo no-lineal: variables en niveles (mod1)								
	c	i	y	k	h	r	w	a
Constante	0.077	0.038	0.115	0.038	0.2	1.016	0.382	1
k(-1)	0.683	0.333	1.016	0.333	0	-18.054	3.389	0
a(-1)	0.075	0.037	0.112	0.037	0	0.995	0.374	0.979
e	0.077	0.038	0.115	0.038	0	1.016	0.382	1
Modelo no-lineal: variables en logarítmo (mod2)								
	cc	ii	yy	kk	hh	rr	ww	aa
Constante	-2.563	-3.282	-2.166	-3.282	-1.609	0.016	-0.962	0
kk(-1)	0.333	0.333	0.333	0.333	0	-0.667	0.333	0
aa(-1)	0.979	0.979	0.979	0.979	0	0.979	0.979	0.979
e	1	1	1	1	0	1	1	1
Modelo lineal: variables en niveles (mod3)								
	ct	it	yt	kt	ht	rt	wt	at
kt(-1)	0.683	0.333	1.016	0.333	0	-18.054	3.389	0
at(-1)	0.075	0.037	0.112	0.037	0	0.995	0.374	0.979
e	0.077	0.038	0.115	0.038	0	1.016	0.382	1
Modelo lineal: variables en logarítmo (mod4)								
	ch	ih	yh	kh	hh	rh	wh	ah
kh(-1)	0.333	0.333	0.333	0.333	0	-0.667	0.333	0
ah(-1)	0.979	0.979	0.979	0.979	0	0.979	0.979	0.979
e	1	1	1	1	0	1	1	1

Este cuadro ha sido construido en base a lo que muestra Dynare en el *prompt* de Matlab manteniendo el orden (inicial) de las variables que aparece en el .mod.

De lo anterior, dos conclusiones son importante mencionar: la primera es que es preferible que en la función de política cada variable esté expresada en desviaciones porcentuales con respecto al estado estacionario; es decir, se prefiere variables en logarítmo. Esto se debe a que los coeficientes de la solución se entienden como elasticidades y porque permite tener una lectura sencilla de la función impulso-respuesta. La segunda es que sea que coloquemos el modelo no-lineal en Dynare o que lo hayamos linealizado y luego lo coloquemos en Dynare, los coeficientes de la función de política y de estado serán los mismos. Por ejemplo para las variables en niveles: el modelo no-lineal (mod1) y el lineal (mod3) tienen los mismos coeficientes. De manera similar para las variables en logarítmo (mod2 y mod4).

¿Dónde guarda Dynare los coeficientes de la función de política y de estado? Dynare guarda la función de política y de estado en varias variables dentro de la estructura "oo_.dr". Existen algunas consideraciones para capturar de manera correcta los coeficientes de estas funciones.

[1] Orden de las variables: el orden inicial de las variables tal como se escribió en el archivo .mod es el siguiente: c, i, y, k, h, r, w, a. Entonces el consumo tiene el primer lugar, mientras que la inversión el segundo lugar y así sucesivamente para las demás variables. Sin embargo, cuando Dynare resuelve el sistema reordena estas variables, guardando el nuevo orden en "oo_.dr.oder_var". Esta variable muestra un vector de números que contiene la nueva posición de las variables: 2, 3, 5, 7, 4, 8, 1, 6. Esto significa que la variable que estaba en la posición inicial 2 (que es la inversión) ahora está en el primer lugar. Por ejemplo, el consumo estaba inicialmente en la primera posición pero ahora aparece en la posición 7. Esto es importante porque los coeficientes de la solución corresponden a este nuevo orden. Entonces el vector de variable reordenado es:

Posición inicial			Posición reordenada	
1	c		i	2
2	i		y	3
3	y		h	5
4	k	$\longrightarrow$	w	7
5	h		k	4
6	r		a	8
7	w		c	1
8	a		r	6

[2] Función de política y de estado como sistema de ecuaciones: sea el vector de variables endógenas reordenadas Y_t, el vector de los valores de estado estacionario Y_{ss}, la matriz que contiene los coeficientes de las variables de estado "ghx", el vector de las variables de estado X_t y el vector que contiene los coeficientes asociados al error "ghu". Entonces el sistema de ecuaciones que representan las funciones de política y de estado es:

$$Y_t = Y_{ss} + ghx * X_t + ghu * U_t$$

Escribiendo este sistema en su forma extensiva:

$$\begin{bmatrix} i \\ y \\ h \\ w \\ k \\ a \\ c \\ r \end{bmatrix} = \begin{bmatrix} i_{ss} \\ y_{ss} \\ h_{ss} \\ w_{ss} \\ k_{ss} \\ a_{ss} \\ c_{ss} \\ r_{ss} \end{bmatrix} + \begin{bmatrix} \eta_{ik} & \eta_{ia} \\ \eta_{yk} & \eta_{ya} \\ \eta_{hk} & \eta_{ha} \\ \eta_{wk} & \eta_{wa} \\ \eta_{kk} & \eta_{ka} \\ \eta_{ak} & \eta_{aa} \\ \eta_{ck} & \eta_{ca} \\ \eta_{rk} & \eta_{ra} \end{bmatrix} * \begin{bmatrix} k(-1) \\ a(-1) \end{bmatrix} + \begin{bmatrix} \eta_{iu} \\ \eta_{yu} \\ \eta_{hu} \\ \eta_{wu} \\ \eta_{ku} \\ \eta_{au} \\ \eta_{cu} \\ \eta_{ru} \end{bmatrix} * e$$

Por ejemplo, la ecuación de política de la inversión es:

$$i = i_{ss} + \eta_{ik} k(-1) + \eta_{ia} a(-1) + \eta_{iu} e$$

La cual para el mod2, en el cuadro [28], se tiene:

$$i = -3.282 + 0.333 k(-1) + 0.979 a(-1) + 1e$$

[3] Coeficientes de estas funciones en oo_.dr: el vector de estado estacionario se encuentra en "oo_.dr.ys", la cual mantiene el orden inicial de las variables. El nuevo orden solo aplica para la matriz asociada a las variables de estado y del error, las cuales están guardadas respectivamente en "oo_.dr.ghx" y "oo_.dr.ghu".

$$\text{oo\_.dr.ys} = \begin{bmatrix} -2.563 \\ -3.282 \\ -2.166 \\ -3.282 \\ -1.609 \\ 0.016 \\ -0.962 \\ 0 \end{bmatrix} \quad \text{oo\_.dr.ghx} = \begin{bmatrix} 0.333 & 0.979 \\ 0.333 & 0.979 \\ 3.09E-15 & 1.03E-14 \\ 0.333 & 0.979 \\ 0.333 & 0.979 \\ 0 & 0.979 \\ 0.333 & 0.979 \\ -0.667 & 0.979 \end{bmatrix} \quad \text{oo\_.dr.ghu} = \begin{bmatrix} 1 \\ 1 \\ 2.54E-14 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

4.9. Función Impulso Respuesta (IRF)

Como se menciono previamente, la estructura oo_ también contiene la función impulso-respuesta del modelo. Debido a que se requiere que todas las variables endógenas respondan al impulso, entonces Dynare crea otra estructura llamada "irfs", en la cual guarda la función impulso-respuesta de cada variable como se puede ver en la figura [9].

Si se requiere solo la función-impulso respuesta del consumo, por ejemplo, entonces basta con hace click en la variable "oo_.irfs.cc_e". Como se puede ver en la figura [10], la variable "oo_.irfs.cc_e" contiene un vector de 40 periodos, lo cual fue definifo en `stoch_simul` cuando se colocó "irf=40".

Dynare también grafica cada una de las variables que se encuentran en la estructura "oo_.irfs"; sin embargo, la gráfica es esencialmente básica en el sentido que no se modifican las líneas y además no considera los nombres de las variables en su versión extendida. Esto se observa en la figura [11].

La figura [11] se podría mejorar si se construye un código en Matlab que se alimente del archivo .mod y que grafique con los nombres extendidos de las variables y modificaciones

Figura 9: Función impulso-respuesta (oo_.irfs)

oo_.irfs <1x1 struct>

Field	Value	Min	Class
cc_e	<1x40 double>	0.0048	double
ii_e	<1x40 double>	0.0048	double
yy_e	<1x40 double>	0.0048	double
kk_e	<1x40 double>	0.0048	double
hh_e	<1x40 double>	0	double
rr_e	<1x40 double>	-1.983...	double
ww_e	<1x40 double>	0.0048	double
aa_e	<1x40 double>	0.0031	double

Nota: Esta función impulso-respuesta se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

Figura 10: Función impulso-respuesta del consumo (oo_.irfs.cc_e)

oo_.irfs.cc_e <1x40 double>

	1	2	3	4	5	6	7	8	9	10	11
1	0.0072	0.0094	0.0100	0.0101	0.0100	0.0098	0.0096	0.0094	0.0092	0.0090	0.0088

Nota: Esta función impulso-respuesta se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

estéticas en el gráfico. Esto se realiza en el código "irfs_nolineal_log.m" y se puede ver en el gráfico [12] el resultado.

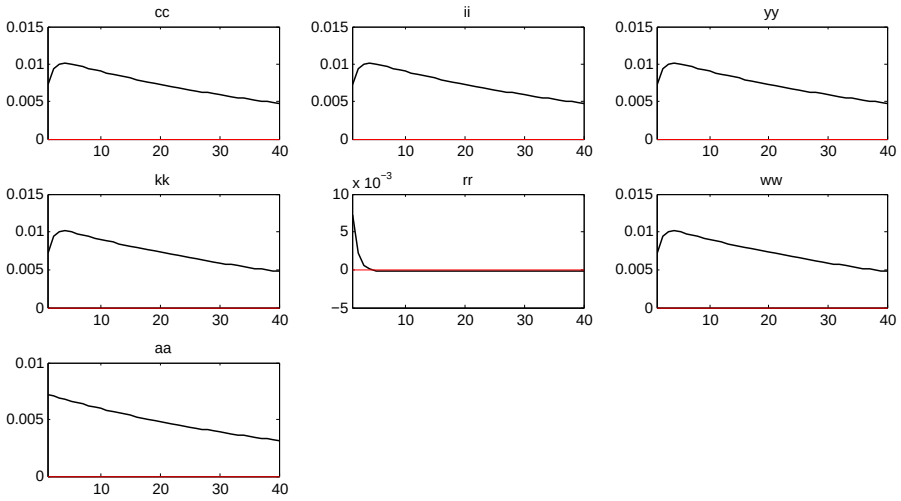
```
1  dynare Long_Plosser_Dynare_nolineal_log;
2  IRF = [oo_.irfs.cc_e', oo_.irfs.ii_e', oo_.irfs.yy_e', oo_.irfs.kk_e', ...
       oo_.irfs.hh_e',oo_.irfs.rr_e', oo_.irfs.ww_e', oo_.irfs.aa_e',];
3  names = {'Consumo', 'Inversión', 'Producto', 'Capital', 'Trabajo', ...
       'Tasa de interés', 'Salario real', 'Productividad'};
4  for i=1:size(IRF,2)
5      subplot(2,4,i)
6      plot(IRF(:,i),'LineWidth', 1.5);
7      title(names{i});
8      grid;
9  end
10 orient landscape
11 saveas(gcf,'ifrs_nolineal_log_matlab','pdf');
```

Descripción del código: Primero se corre el archivo .mod (línea 1 del código):

dynare Long_Plosser_Dynare_nolineal_log.mod;

Luego se define una matriz que contenga todas las funciones impulso-respuesta (IRF), línea 2 del código. En tercer lugar se define un vector de celda que contenga los nombres de las variables (names). Finalmente, se construye un bucle para graficar cada impulso-respuesta con el nombre de la variable apropiada y el tamaño de linea apropiado (línea 4 al 9 del código). Los dos últimas comandos de este segmento de código coloca la hoja, donde

Figura 11: Función impulso-respuesta (gráfica de Dynare)



Nota: Esta gráfica impulso-respuesta se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

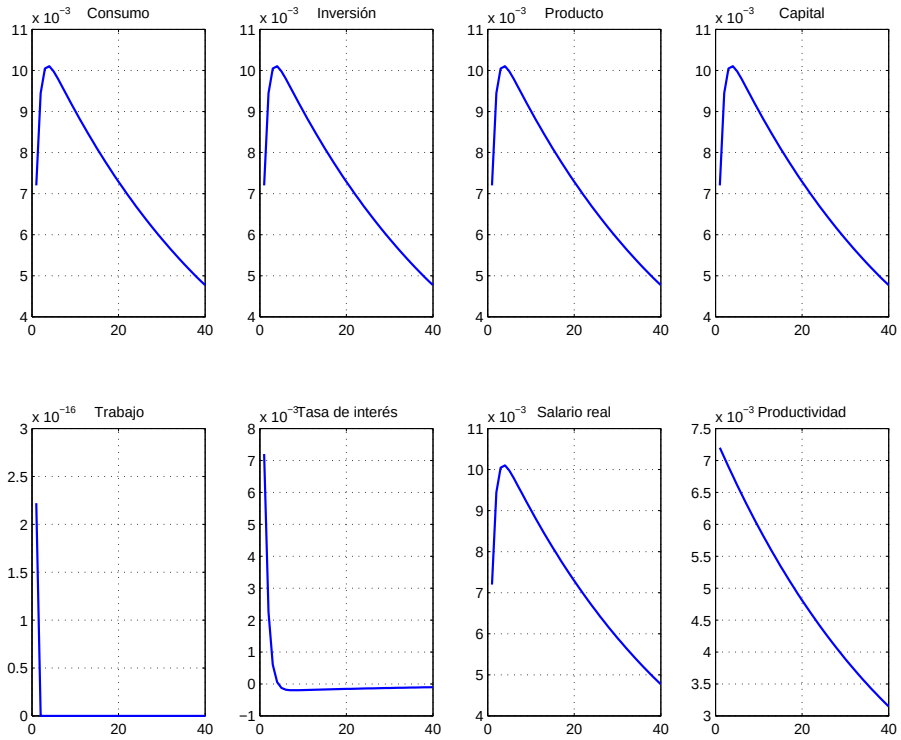
se guarda el gráfico, en orientación horizontal (`orient landscape`) y luego la guarda en extensión pdf (línea 11).

4.10. Análisis de sensibilidad

El cuadro [13] muestra la función impulso-respuesta ante tres valores de la persistencia del choque de productividad. Se puede observar que mientras mayor persistencia muestra el choque, mayor es la reacción de cada variable endógena (excepto la tasa de interés y el trabajo, el cual se mantiene en su valor de estado estacionario). Además, se puede observar que las variables se tomán más tiempo para volver a su estado estacionario. Este gráfico se obtiene al escribir los siguientes códigos en el archivo .mod después de escribir `stoch_simul`.

```
% Valores del parámetro
rhos = [0.5 0.7 0.9];
for j= 1:size(rhos,2)
rho = rhos(j);
stoch_simul(order=1, irf=40, nograph, nomoments,nofunctions);
oo_sen{j} = oo_;
end;
% Gráfica
name = {'Consumo', 'Inversión','Producción','Capital','Trabajo','Tasa ...
    de interés real', 'Salario', 'Productividad'};
field_name = fieldnames(oo_sen{1}.irfs); time = 1:40;
    for j=1:size(name,2)
        subplot(2,4,j)
        plot(time,oo_sen{1}.irfs.(field_name{j}),...
            time,oo_sen{2}.irfs.(field_name{j}),'--',...
            time,oo_sen{3}.irfs.(field_name{j}),'-.','LineWidth', 1.5)
        title(name{j});
```

Figura 12: Función impulso-respuesta (gráfica de Matlab)



Nota: Esta gráfica impulso-respuesta se obtiene del archivo "ifrs_nolineal_log.m"

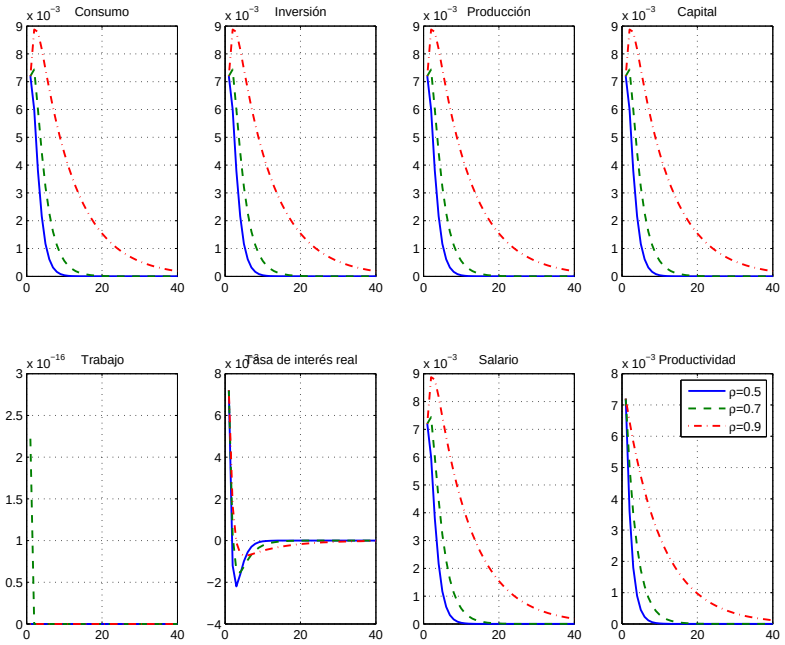
```
17          grid;
18      end;
19      legend('\rho=0.5', '\rho=0.7', '\rho=0.9');
20  orient landscape
21  saveas(gcf,'analisis_sensibilidad','pdf');
```

Descripción del código: este código tiene dos partes. La primera es guardar la simulación del modelo ante cada valor de la persistencia de la productividad. Para ello se define un vector de valores de la persistencia (línea 2), y luego se aplica `stoch_simul` para los tres valores y se guarda cada resultado (oo_) en un vector de celda (oo_sen).

La segunda parte del código es graficar la función impulso-respuesta bajo los tres valores del parámetro. Para obtener esta gráfica en primer lugar se define el nombre de las variables, cuyo orden es el que se describe al inicio del archivo .mod. En segundo lugar se extrae (como vector de celda) los nombres de la estructura "oo_sen{1}.irfs" (línea 9), la utilidad de esto es que servirá para hacer bucles con estructuras. En tercer lugar se construye un bucle para graficar el impulso-respuesta de cada variable ante los tres valores de la persistencia. La línea 13 es de especial importancia: el código "oo_sen{1}.irfs.(field_name{j})" para j=1 es "oo_sen{1}.irfs.cc_e". Para j=2 es "oo_sen{1}.irfs.ii_e" y así sucesivamente. En ello se puede ver la utilidad que tiene el "campo" de una variable estructura. Finalmente, el código `orient` indica la orientación de la hoja en la que se guardará el gráfico y el

código `saveas` indica el nombre y la extensión con que se guardará el gráfico (usualmente se guarda en pdf o eps por su utilidad en LaTeX).

Figura 13: Análisis de sensibilidad: persistencia del Choque de productividad ρ



Nota: Esta gráfica impulso-respuesta se obtiene del archivo "Long_Plosser_Dynare_nolineal_log.mod"

4.11. Simulación de las variables endógenas

Luego de que Dynare encuentra la solución del sistema linealizado se puede obtener la representación de series de tiempo ARMA(p,q) de las variables endógenas. Por ejemplo, del cuadro [28] para el capital del modelo no-lineal con variable en logarítmo se tiene:

$$\begin{aligned}
kk_{t+1} &= kk_{ss} + \eta_{kk} kk_t + \eta_{ka} aa_{t-1} + \epsilon_t \\
kk_{t+1} - kk_{ss} &= \eta_{kk} kk_t + \eta_{ka} aa_{t-1} + \epsilon_t \\
(lnk_{t+1} - lnk_{ss}) &= \eta_{kk}(lnk_t - lnk_{ss}) + \eta_{ka}(lna_{t-1} - lna_{ss}) + \epsilon_t \\
\widehat{k}_{t+1} &= \eta_{kk}\widehat{k}_t + \eta_{ka}\widehat{a}_{t-1} + \epsilon_t \\
\widehat{k}_{t+1} &= 0.333\widehat{k}_t + \underbrace{0.979\widehat{a}_{t-1} + \epsilon_t}_{\widehat{a}_t} \\
\widehat{k}_{t+1} &= 0.333\widehat{k}_t + \widehat{a}_t \\
(1 - 0.333L)\widehat{k}_{t+1} &= \widehat{a}_t
\end{aligned} \tag{4}$$

Considerando que la productividad, que se comporta como un AR(1), se puede expresar en su forma MA(∞):

$$\begin{aligned} aa_t &= \phi aa_{t-1} + \epsilon_t \\ (1-\phi L)aa_t &= \epsilon_t \\ aa_t &= \frac{\epsilon_t}{1-\phi L} \\ \text{Sabiendo} &: a_{ss} = 1 \\ lna_t - lna_{ss} &= \frac{\epsilon_t}{1-\phi L} \\ \widehat{a}_t &= \frac{\epsilon_t}{1-\phi L} \end{aligned} \tag{5}$$

Introduciendo la ecuación (5) en la ecuación (4) se tiene que el capital se comporta como un AR(2):

$$\begin{aligned} (1-0.333L)\widehat{k}_{t+1} &= \widehat{a}_t \\ (1-0.333L)\widehat{k}_{t+1} &= \frac{\epsilon_t}{1-\phi L} \\ \widehat{k}_{t+1} &= \frac{\epsilon_t}{(1-0.333L)(1-\phi L)} \end{aligned} \tag{6}$$

Dado que se tiene la expresión de series de tiempo de cada variable, Dynare podría simular el comportamiento de cada variable asumiendo un comportamiento aleatorio $N(0, \sigma_e^2)$ para el error ϵ. Para que Dynare realice esta tarea es suficiente indicarle en el `stoch_simul` el número de periodos (`periods`) que deseamos que la variable tenga. Por ejemplo:

```
stoch_simul(order=1, periods = 150)
```

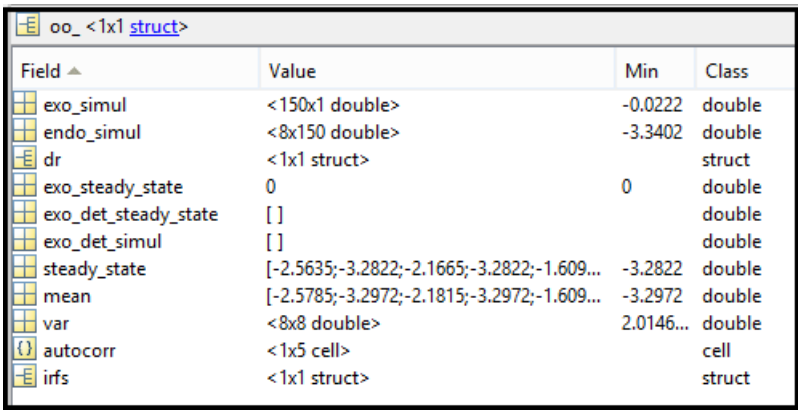
Tres resultados podemos enfatizar sobre este código: en primer lugar, las variables simuladas son guardadas en la estructura oo_, en particular en las variables "exo_simul" y "endo_simul" (ver figura [14]). La primera variable contiene la simulación de la variable exógena; es decir, el error ϵ_t que se distribuye como una normal con media cero y varianza constante σ_e^2 (ver figura [15]). La segunda variable (endo_simul) contiene la simulación de todas las variables endógenas (en este caso son ocho). Cada fila representa la simulación de una variable, el número de columnas es el número de periodos que se definio en stoch_simul (ver figura [16]).

Lo anterior muestra **una** simulación para cada una de las variables endógenas. Sin embargo, si se desea realizar por ejemplo 300 simulaciones para cada variable considerando 150 periodos es necesario no solo utilizar `periods = 150`, sino también `simul_replic = 300`. El resultado de estas opciones de `stoch_simul` es un archivo binario con el siguiente nombre: "NombreMod_simul". Una desventaja de este archivo es que no se puede abrir directamente en Matlab o algún otro programa. Para leer este archivo se usará una función creada por Johannes Pfeifer[3], el cual se puede descargar de la web. A continuación se describe los pasos para aplicar esta función:

- En primer lugar la función "get_simul_replications.m" tiene que estar presente en el directorio donde se encuentre el archivo .mod.

[3]https://github.com/JohannesPfeifer/DSGE_mod/blob/master/Hansen_1985/get_simul_replications.m

Figura 14: Estructura oo_ (simulación)



oo_ <1x1 struct>

Field	Value	Min	Class
exo_simul	<150x1 double>	-0.0222	double
endo_simul	<8x150 double>	-3.3402	double
dr	<1x1 struct>		struct
exo_steady_state	0	0	double
exo_det_steady_state	[ ]		double
exo_det_simul	[ ]		double
steady_state	[-2.5635;-3.2822;-2.1665;-3.2822;-1.609...	-3.2822	double
mean	[-2.5785;-3.2972;-2.1815;-3.2972;-1.609...	-3.2972	double
var	<8x8 double>	2.0146...	double
autocorr	<1x5 cell>		cell
irfs	<1x1 struct>		struct

Nota: Esta estructura se obtiene de "Long_Plosser_Dynare_nolineal_log.mod"

- En el archivo mod, después del comando `stoch_simul` se debe de colocar lo siguiente:

```
[sim_array]=get_simul_replications(M_,options_);
y_sim=squeeze(sim_array(strmatch('y',M_.endo_names,'exact'),:,:));
k_sim=squeeze(sim_array(strmatch('k',M_.endo_names,'exact'),:,:));
```

  La primera línea llama a la función "get_simul_replications.m" para convertir el archivo binario en una variable matricial de Matlab llamada "sim_array", la cual se guarda en el *workspace*. La segunda línea crea la variable "y_sim", la cual contiene la simulación del producto; es decir, contiene las 300 simulaciones (filas) de 100 periodos (columnas). Esta misma línea de código se puede aplicar a cada variable. El resultado de ello es una matriz, por variable, de 300 filas con 100 columnas. La tercera línea el código es la misma que la segunda solo que para el capital.

En la figura [17] se observa seis simulaciones del conjunto de 300 para el capital y el producto.

4.12. Cálculo de los momentos

El cuadro [29] muestra los momentos calculados por Dynare para los cuatro formas de escribir el modelo en Dynare. Una primera observación es que los momentos son similares cuando las variables tienen la misma naturaleza; es decir, si las variables están en niveles entonces no importa si el modelo que se escribió en Dynare fue no-lineal o linealizado, los momentos son similares. Lo mismo se concluye para las variables en logarítmo. Una segunda observación es que los momentos entre el modelo con las variables en niveles y el modelo con las variables en logarítmo son diferentes, lo cual es consistente con lo esperado.

¿Dónde guarda Dynare los momentos?
Dynare guarda la media, la matriz de varianza y covarianza y las autocorrelaciones dentro de la estructura oo_. Como se puede ver en la figura [14], el promedio se guarda en la

Figura 15: Simulación de la variable exógena

oo_.exo_simul <150x1 double>

	1	2
1	-2.8711e-04	
2	-0.0076	
3	-0.0019	
4	0.0058	
5	0.0044	
6	-0.0066	
7	-0.0126	
8	0.0018	
9	0.0039	
10	0.0027	
11	-0.0110	
12	-0.0162	

Nota: Esta estructura se obtiene de "Long_Plosser_Dynare_nolineal_log.mod"

Figura 16: Simulación de la variable endógena

oo_.endo_simul <8x150 double>

	1	2	3	4	5	6	7	8	9	10
1	-2.5638	-2.5715	-2.5752	-2.5697	-2.5632	-2.5679	-2.5816	-2.5827	-2.5778	-2.5727
2	-3.2825	-3.2902	-3.2939	-3.2884	-3.2820	-3.2866	-3.3003	-3.3014	-3.2966	-3.2914
3	-2.1668	-2.1745	-2.1782	-2.1727	-2.1662	-2.1709	-2.1846	-2.1857	-2.1808	-2.1757
4	-3.2825	-3.2902	-3.2939	-3.2884	-3.2820	-3.2866	-3.3003	-3.3014	-3.2966	-3.2914
5	-1.6094	-1.6094	-1.6094	-1.6094	-1.6094	-1.6094	-1.6094	-1.6094	-1.6094	-1.6094
6	0.0158	0.0084	0.0124	0.0216	0.0226	0.0115	0.0024	0.0151	0.0210	0.0213
7	-0.9623	-0.9700	-0.9737	-0.9682	-0.9618	-0.9664	-0.9801	-0.9812	-0.9763	-0.9712
8	-2.8711e-04	-0.0079	-0.0090	-0.0023	0.0023	-0.0045	-0.0166	-0.0132	-0.0080	-0.0044
9										

Nota: Esta estructura se obtiene de "Long_Plosser_Dynare_nolineal_log.mod"

variable "mean", la matriz de covarianza en la variable "var" y la matriz de autocorrelación en "autocorr".

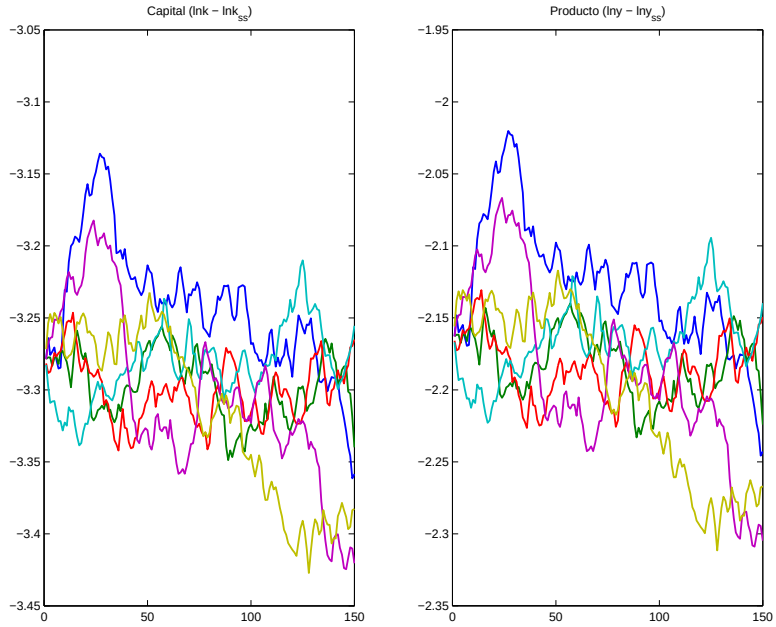
4.13. Filtro HP

El cuadro [29] muestra los momentos de las variables endógenas, más no muestra los momentos de su componente cíclico, el cual es necesario para compararlo con los hechos estilizados y evaluar el poder explicativo del modelo. Para obtener el componente cíclico es necesario aplicar un filtro; es decir, una técnica que descomponga la variable en sus dos componentes: tendencia y ciclo. Para realizar esta tarea usualmente se aplica el filtro HP, el cual Dynare tiene habilitado como una opción de stoch_simul. El código para usar el filtro HP es el siguiente:

stoch_simul(order=1, hp_filter= lambda)

Donde "lambda" es igual a 1600 para datos trimestrales (para mayor detalle ver el cuadro [9]). Este código brinda los momentos del componente cíclico en el *command window*

Figura 17: Seis simulaciones del capital y el producto



Nota: Esta estructura se obtiene de "Long_Plosser_Dynare_nolineal_log.mod"y del código "simulacion_filtrohp.m"

de Matlab, los cuales se muestran en el cuadro [30]. Cabe mencionar que Dynare no muestra el componente cíclico como una serie de tiempo.

Ante la desventaja que Dynare no calcula el componente cíclico de las series, Matlab dispone de una función llamada "hpfilter.m", la cual brinda el componente tendencial y cíclico de la serie. El uso de esta función se aprecia a continuación:

```
[trend_k,ciclo_k] =hpfilter(kk_sim(:,1),1600);
```

La función "hpfilter.m" requiere dos insumos. El primero es la serie o conjunto de series a la que se desea aplicar el filtro, en este caso es la primera simulación de capital "kk_sim(:,1)". La segunda es el parámetro de suavizamiento, el cual depende de la frecuencia de los datos (que se refleja en la calibración). Este parámetro toma el valor de 14400 para datos mensuales, 1600 para datos trimestrale y 100 para datos anuales; en este caso en particular estamos considerando datos trimestrales por ello se coloca 1600.

Asimismo, esta función entrega dos resultados: la primera es el componente tendencial de la serie (trend_k) y la segunda es el componente cíclico (ciclo_k). En la figura [18] se observa el componente cíclico y tendencial del capital derivado de la aplicación del filtro HP.

Cuadro 29: Momentos teóricos

Modelo No-Lineal							
Variables en niveles				Variables en logaritmos			
Variable	Media	Des. Est.	Varianza	Variable	Media	Des. Est.	Varianza
c	0.077	0.004	0	cc	-2.564	0.053	0.0028
i	0.038	0.002	0	ii	-3.282	0.053	0.0028
y	0.115	0.006	0	yy	-2.167	0.053	0.0028
k	0.038	0.002	0	kk	-3.282	0.053	0.0028
h	0.200	0	0	hh	-1.609	0	0
r	1.016	0.008	0.0001	rr	0.016	0.008	0.0001
w	0.382	0.020	0.0004	ww	-0.962	0.053	0.0028
a	1	0.035	0.0012	aa	0	0.035	0.0012
Modelo Lineal							
Variables en niveles				Variables en logaritmos			
Variable	Media	Des. Est.	Varianza	Variable	Media	Des. Est.	Varianza
ct	0.077	0.004	0	ch	-2.564	0.053	0.0028
it	0.038	0.002	0	ih	-3.282	0.053	0.0028
yt	0.115	0.006	0	yh	-2.167	0.053	0.0028
kt	0.038	0.002	0	kh	-3.282	0.053	0.0028
ht	0.200	0	0	hh	-1.609	0	0
rt	1.016	0.008	0.0001	rh	0.016	0.008	0.0001
wt	0.382	0.020	0.0004	wh	-0.962	0.053	0.0028
at	1	0.035	0.0012	ah	0	0.035	0.0012

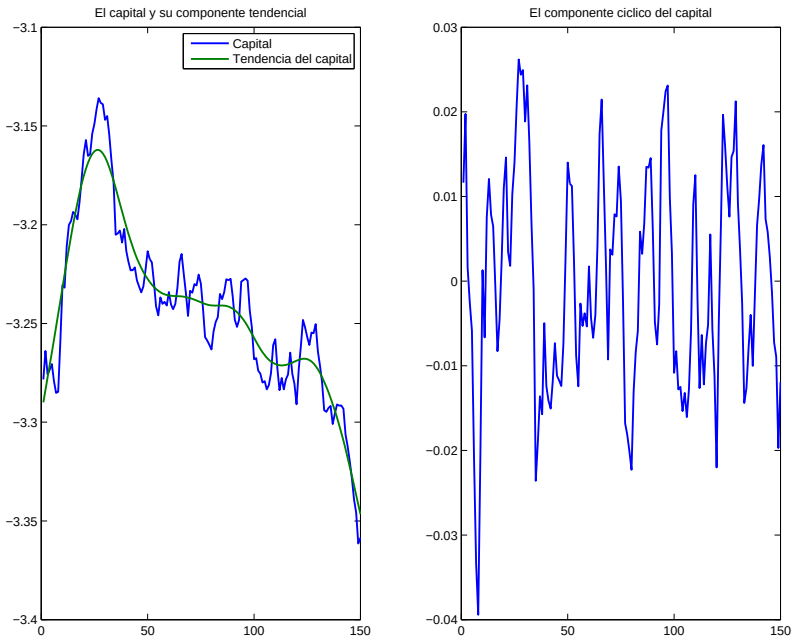
5. Códigos

En el cuadro [31] se indica los códigos utilizados en este capítulo.

Cuadro 30: Momentos teóricos (filtro HP)

Momentos teóricos (HP filter, lambda=1600)			
Variable	Media	Des. Est.	Varianza
cc	-2.5635	0.0126	0.0002
ii	-3.2822	0.0126	0.0002
yy	-2.1665	0.0126	0.0002
kk	-3.2822	0.0126	0.0002
hh	-1.6094	0	0
rr	0.0161	0.0072	0.0001
ww	-0.962	0.0126	0.0002
aa	0	0.0094	0.0001

Figura 18: Componente cíclico y tendencial del capital



Nota: Esta estructura se obtiene de "Long_Plosser_Dynare_nolineal_log.mod"y del código "simulacion_filtrohp.m"

Cuadro 31: Códigos en Matlab y Dynare

Códigos	Descripción
Matlab	
irfs_nolineal_log.m	Este m-file ilustra que los gráficos la función impulso-respuesta obtenidos de Dynare pueden ser mejorados mediante códigos de Matlab.
simulacion_filtrohp.m	Este m-file aplica el filtro HP a la series simuladas del modelo.
aux_irfs_nolineal_log.m	Este m-file grafica las funciones impulso-respuesta por medio de un bucle.
aux_analisis_sensibilidad.m	Este m-file decribe el código que se puede escribir al final de un archivo .mod para realizar análisis de sensibilidad; es decir, para obtener las funciones impulso-respuesta ante diferentes valores de los parámetros.
Dynare	
Long_Plosser_Dynare_nolineal_niv.mod	Este .mod contiene las ecuaciones no-lineales y con las variables en niveles del modelo de Long y Plosser (1983, 1989).
Long_Plosser_Dynare_nolineal_log.mod	Este .mod contiene las ecuaciones no-lineales y con las variables en logarítmo del modelo de Long y Plosser (1983, 1989). Este código es utilizado en el capítulo 2 y 3. En el capítulo 2, este código se usa para ejemplificar los comandos de Dynare. En el capítulo 3 se utiliza para obtener la solución del modelo y los IRF.
Long_Plosser_Dynare_lineal_niv.mod	Considera el modelo lineal con las variables en niveles del modelo de Long y Plosser (1983, 1989).
Long_Plosser_Dynare_lineal_log.mod	Considera el modelo lineal con las variables en logarítmo del modelo de Long y Plosser (1983, 1989).