

Esta obra esta bajo una licencia reconocimiento-no comercial 2.5 Colombia de creativecommons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by/2.5/co/> o envíe una carta a creative commons, 171 second street, suite 30 San Francisco, California 94105, USA

DYNARE



Autores:

**ANDREA ELIANA BARRERA ARDILA
LAURA VANESSA HERNÁNDEZ CRUZ**

Director Unidad Informática: Henry Martínez Sarmiento

Tutor Investigación: Alejandro Bolívar

Coordinadores: Álvaro Schneider Guevara
Juan Felipe Reyes Rodríguez

Coordinador Servicios Web: Miguel Ibáñez

**Analista de Infraestructura
y Comunicaciones:** Alejandro Bolívar

**Analista de Sistemas de
Información:** Mesías Anacona Obando

**UNIVERSIDAD NACIONAL COLOMBIA
FACULTAD DE CIENCIAS ECONÓMICAS
UNIDAD DE INFORMÁTICA Y COMUNICACIONES
BOGOTÁ D.C.**

DYNARE

Director Unidad Informática: Henry Martínez Sarmiento

Tutor Investigación: Alejandro Bolívar

Auxiliares de Investigación:

ALEJANDRO NIETO RAMOS	JORGE ALBERTO TORRES VALLEJO
ANDREA ELIANA BARRERA ARDILA	JORGE LEONARDO LEMUS CASTIBLANCO
ÁNGEL LEONARDO JEREZ CARVAJAL	JORGE LUIS FANDIÑO GIRALDO
ÁNGELA PATRICIA VEGA CABRA	JOSÉ SANTIAGO APARICIO CASTRO
BENJAMÍN EDUARDO VENEGAS VENEGAS	JUAN CARLOS TARAPUEZ ROA
CAMILO ALBERTO ZAPATA MARTÍNEZ	JULIE ANDREA PADILLA GONZÁLEZ
CINDY LORENA PABÓN GÓMEZ	LAURA VANESSA HERNÁNDEZ CRUZ
DANIEL ALEXANDER LINARES PUERTO	LILIANA CAROLINA HERRERA PRIETO
DAVID CAMILO SÁNCHEZ ZAMBRANO	LUIS ALEJANDRO PICO SILVA
DAVID FELIPE BELTRÁN GÓMEZ	LUIS FERNANDO ALFONSO MUÑOZ
DIANA MARCELA ROJAS TÉLLEZ	MÓNICA YOLANDA MOGOLLÓN PLAZAS
DIEGO ARMANDO POVEDA ZAMORA	MYRIAM JASMIN GUERRA CÁRDENAS
EDGAR ANDRÉS GARCÍA HERNÁNDEZ	NUBIA ALEJANDRA SEGURA TENJICA
IVÁN ALBEIRO CABEZAS MARTÍNEZ	NURY BIBIAN BEJARANO CÁRDENAS
IVÁN DARÍO BARRETO BERNAL	RAÚL ANDRÉS CAMACHO CRUZ
JISSETH TATIANA ÁNGEL RODRÍGUEZ	SANDRA MIREYA AGUILAR MAYORGA

Este trabajo es resultado del esfuerzo de todo el equipo perteneciente a la Unidad de Informática.

Se prohíbe la reproducción parcial o total de este documento, por cualquier tipo de método fotomecánico y/o electrónico, sin previa autorización de la Universidad Nacional de Colombia.

**UNIVERSIDAD NACIONAL COLOMBIA
FACULTAD DE CIENCIAS ECONÓMICAS
UNIDAD DE INFORMÁTICA Y COMUNICACIONES
BOGOTÁ D.C.
DICIEMBRE 2009**

TABLA DE CONTENIDO

TABLA DE CONTENIDO.....	3
1. RESUMEN.....	6
2. ABSTRACT.....	7
3. INTRODUCCIÓN.....	8
4. ¿QUÉ ES DYNARE?.....	9
5. INSTALACIÓN DE DYNARE.....	11
5.1. Requerimientos de software	11
5.2. Instalación de Dynare en Windows	11
5.3. Instalación en Debian GNU/Linux y Ubuntu.....	15
6. SOLUCIÓN DE MODELOS BÁSICOS EGDE	19
6.1. UNA DISTINCIÓN FUNDAMENTAL.....	19
6.1.1. Modelos determinísticos vs estocásticos	20
6.2. Introduciendo un ejemplo	21
6.3. Estructura de los archivos .mod en Dynare.....	25
6.4. PREÁMBULO.....	26
6.4.1. Caso determinístico	27
6.4.2. Caso estocástico.....	27
6.5. Especificaciones del modelo	28
6.5.1. Declaración del modelo en Dynare	28

6.5.2.	Generalidades.....	29
6.5.3.	Convenciones en las notaciones.....	30
6.5.4.	Condiciones sobre la temporalidad de las variables.....	30
6.5.5.	Convenciones para las variables no predeterminadas.....	30
6.5.6.	Modelos lineales y logarítmicos linealizados	31
6.6.	Especificación de estados estacionarios y/o valores iniciales.....	32
6.6.1.	Modelos estocásticos y estado estacionario.....	33
6.6.2.	Modelos determinísticos y valores iniciales.....	34
6.6.3.	Encontrar el estado estacionario	34
6.6.4.	Verificar la estabilidad del sistema.....	35
6.7.	Añadir choques al sistema	35
6.7.1.	Modelos determinísticos: choques temporales.....	36
6.7.2.	Modelos determinísticos: choques permanente	36
6.7.3.	Modelos estocásticos	37
6.8.	Selección del cálculo.....	38
6.8.1.	Para modelos determinísticos	38
6.8.2.	Para modelos estocásticos	39
6.9.	EL ARCHIVO .MOD COMPLETO	41
6.9.1.	El modelo estocástico.....	41
6.9.2.	El modelo determinístico (caso de choques temporales).....	42
6.10.	Ejecución del archivo y resultados.....	43
6.10.1.	Resultados – Modelos estocásticos.....	43

6.10.2. Resultados – Modelos determinísticos.....	48
7. ESTIMACIÓN DE MODELOS BÁSICOS EGDE MEDIANTE TÉCNICAS BAYESIANAS.....	50
7.1. Introducción.....	50
7.2. Declaración de variables y parámetros	50
7.3. Declaración del modelo.....	51
7.4. Declaración de variables observables.....	51
7.5. Especificación del estado estacionario	52
7.6. Declaración de los valores apriori.....	52
7.7. Presentar la estimación.....	55
7.8. El archivo .mod completo.....	59
7.9. Interpretación de los resultados	60
7.9.1. Resultados tabulados.....	60
7.9.2. Resultados gráficos.....	61
8. CONCLUSIONES	63
9. BIBLIOGRAFIA.....	64

1. RESUMEN

Dynare es un preprocesador que colecciona rutinas de MATLAB o GNU Octave para resolver, simular y estimar modelos no lineales que pueden ser tanto determinísticos como estocásticos. Este Software fue desarrollado por el grupo CEPREMAP en París¹.

Este manual pretende ser una guía básica para las personas que empiezan a utilizar el Software. Por esta razón lo primero que se presenta es una breve descripción de la forma como trabaja el programa. Seguidamente se muestran los procesos de instalación para Windows y para Linux, tanto para Matlab como para GNU Octave. Esto con el fin de mostrar la versatilidad del programa y la capacidad de adaptarse a las necesidades y preferencias del usuario, quien puede decidir el entorno en el que desea trabajar.

Posteriormente, se describe de forma sencilla del modelo que será utilizado como ejemplo a lo largo del manual, para la presentación de las características, funciones y generalidades de Dynare. Para esto además, se resolverá para el caso determinístico y para el caso estocástico, haciendo uso de los dos métodos con los que trabaja Dynare: Máxima Verosimilitud y métodos Bayesianos.

¹ CEPREMAP es un organismo Francés, nacido en 1967, de investigación económica y administración pública.

2. ABSTRACT

Dynare is a preprocessor that collects routines MATLAB or GNU Octave to solve, simulate and estimate nonlinear models that can be both deterministic and stochastic. This software was developed by the group CEPREMAP in Paris².

This manual is intended as a basic guide for people starting to use the Software. For this reason the first thing that occurs is a brief description of how the program works. It then shows the installation procedures for Windows and Linux, for both Matlab and GNU Octave. This is to show the versatility of the program and the ability to adapt to the needs and preferences of the user, who may decide the environment in which you want to work.

Subsequently described in a simple model to be used as examples throughout the manual, for the presentation of the features, functions and generalizations of Dynare. For this also the case be resolved for deterministic and stochastic case, using the two methods are working with Dynare: Maximum likelihood and Bayesian methods.

² CEPREMAP is a French agency, born in 1967, of economic research and public administration

3. INTRODUCCIÓN

Dynare es un programa de Software Libre de Código abierto. Cuenta con una página principal en la cual se consigue toda la documentación, soporte y ayuda necesaria para su utilización. Así mismo cuenta con una Wiki y foros especializados. Sin embargo todo el material consignado está en inglés, razón por la cual el presente manual es parte de la traducción de uno de los manuales más completos que proporciona Dynare: “*Dynare User Guide. An Introduction to the Solution & Estimation of DSGE Models*” de Tommaso Mancini. Específicamente está focalizado en la resolución de ejercicios BÁSICOS por los métodos ya antes mencionados.

La clase de modelos que resuelve Dynare, cada vez son más utilizados en los Bancos Centrales para análisis de políticas y pronósticos de mediano plazo, es por esta razón que se hace necesario promover el conocimiento de las herramientas que proporciona Dynare, e involucrar a los estudiantes con estos instrumentos, para así darles herramientas prácticas para su vida profesional.

Es por esto que, como se mencionó con anterioridad, el propósito de este manual es dar una guía introductoria mediante la explicación del desarrollo y funcionamiento de Dynare, utilizando para este fin modelos básicos. Estas funcionalidades pueden extenderse posteriormente con modelos más complejos, situación que se deja planteada para futuras investigaciones.

4. ¿QUÉ ES DYNARE?

Dynare es un motor poderoso y sumamente personalizable, con una interfaz intuitiva, que resuelve, simula y estima modelos de equilibrio general dinámico estocástico (EGDE).

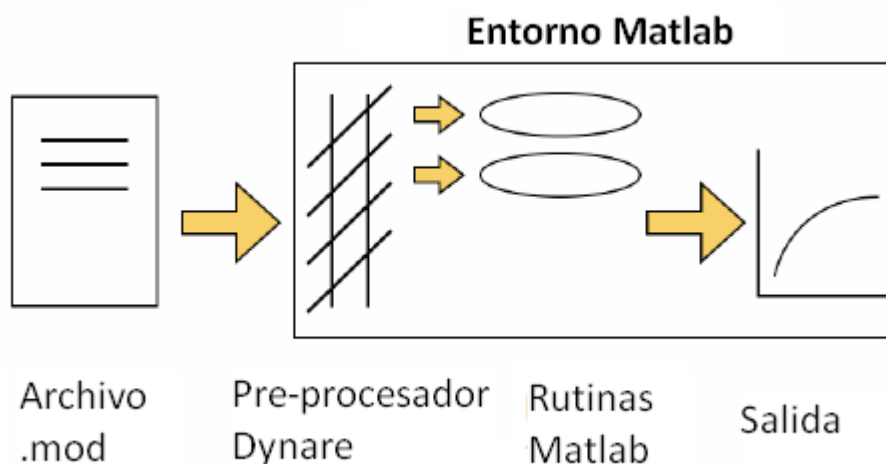


Figura 1.1: El archivo .mod es leído por el preprocesador Dynare, el cual llama las rutinas más relevantes de Matlab para llevar a cabo las operaciones deseadas y presentar los resultados.

En pocas palabras, es un pre-procesador y un coleccionador de rutinas de Matlab³ que tiene grandes ventajas puesto que lee las ecuaciones de modelos EGDE, escritas como en los documentos académicos. Esto no sólo facilita la introducción de un modelo, sino que también permite compartir fácilmente su código, ya que es sencillo de leer por cualquier persona.

La figura 1.1 da una idea de la forma en que trabaja Dynare. Básicamente, el modelo y los atributos relacionados con él, como una estructura de choque por ejemplo, son escritos ecuación por ecuación en un editor de su elección. El archivo que resulta será llamado el archivo .mod. Éste archivo es llamado entonces desde Matlab. Allí se inicia el pre-procesador Dynare, el cual traslada el archivo .mod en una entrada adecuada para las

³ También trabaja sobre GNU Octave. Sin embargo, por simplicidad este trabajo muestra el funcionamiento de Dynare principalmente sobre Matlab.

rutinas de Matlab (más precisamente, Dynare crea un intermediario Matlab o archivos C los cuales son usados por el código Matlab) utilizado ya sea para resolver o para estimar el modelo. Finalmente los resultados son presentados en Matlab.

Dynare es capaz de hacer lo siguiente:

Calcular el estado estacionario de un modelo.

Calcular la solución de modelos determinísticos.

Calcular la aproximación de primer y segundo orden de la solución de modelos estocásticos.

Estimar parámetros del modelo EGDE utilizando el método de máxima verosimilitud, o una aproximación bayesiana.

Calcular políticas óptimas en modelos lineales-cuadráticos.

5. INSTALACIÓN DE DYNARE

5.1. *Requerimientos de software*

Como se mencionó anteriormente, Dynare está disponible tanto para Windows como para Linux. Respecto a Windows, funciona con Windows® 98/NT/2000/XP/Vista; para Linux es compatible con Debian GNU/Linux y Ubuntu, y trabaja sobre arquitecturas que utilicen procesadores Intel o AMD.

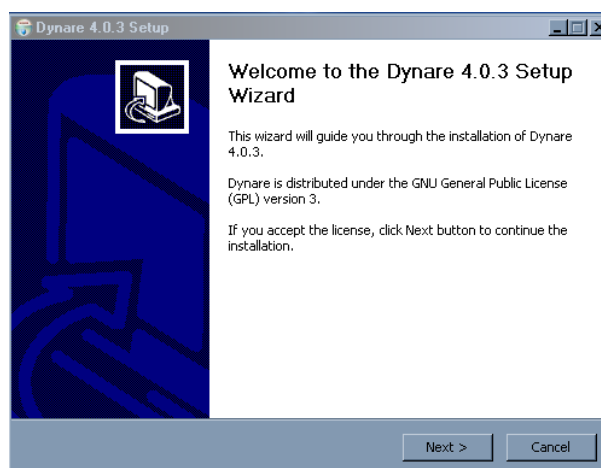
Debe tenerse en cuenta que el Software requiere que la versión de Matlab sea la 6.5 o una superior, mientras que para GNU Octave requiere que la versión se la 3.0.0 o superior.

El Software puede trabajar en otros sistemas operativos o en otras arquitecturas, pero para esto necesita procedimientos extra que no se van a trabajar en esta investigación.

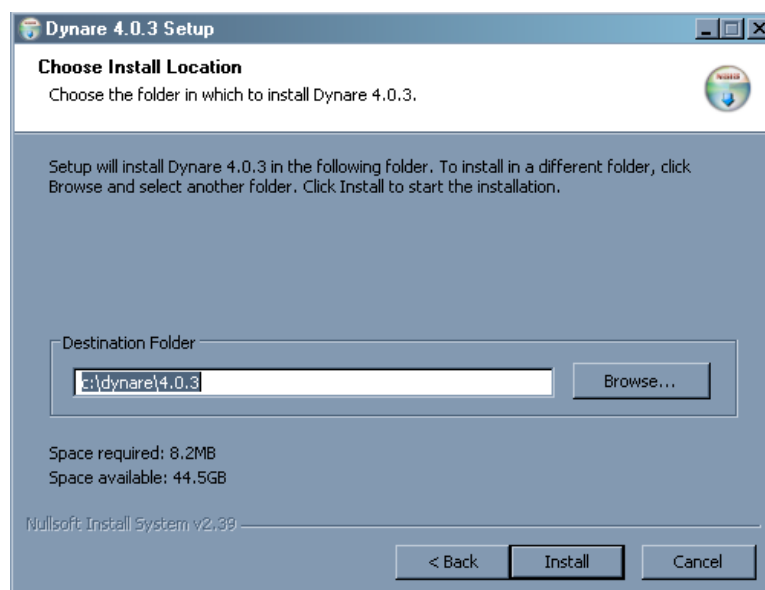
5.2. *Instalación de Dynare en Windows*

Basados en la documentación se procedió a realizar la instalación en Windows. En el directorio [\\serverecono2\\publicas\\$](http://serverecono2.publicas$) se encuentra el ejecutable para la instalación de Dynare, en la versión 4.0.3.

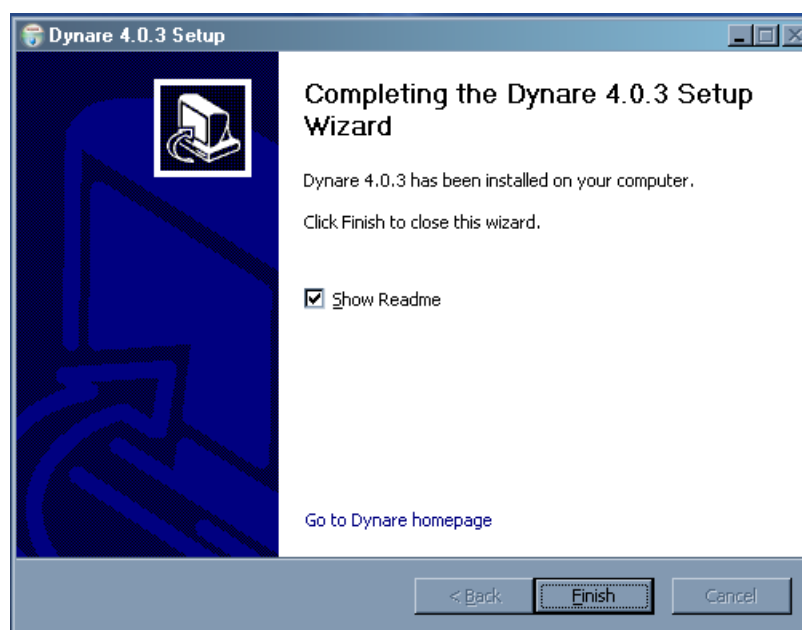
Luego de ejecutar el archivo dynare-4.0.3-win32.exe se abrirá una ventana como la siguiente donde se seleccionará la opción Next.



En la siguiente ventana aparecerá la ubicación donde se instalará el Software; por defecto se guardará en c:\dynare\4.0.3. Se selecciona la opción Install.

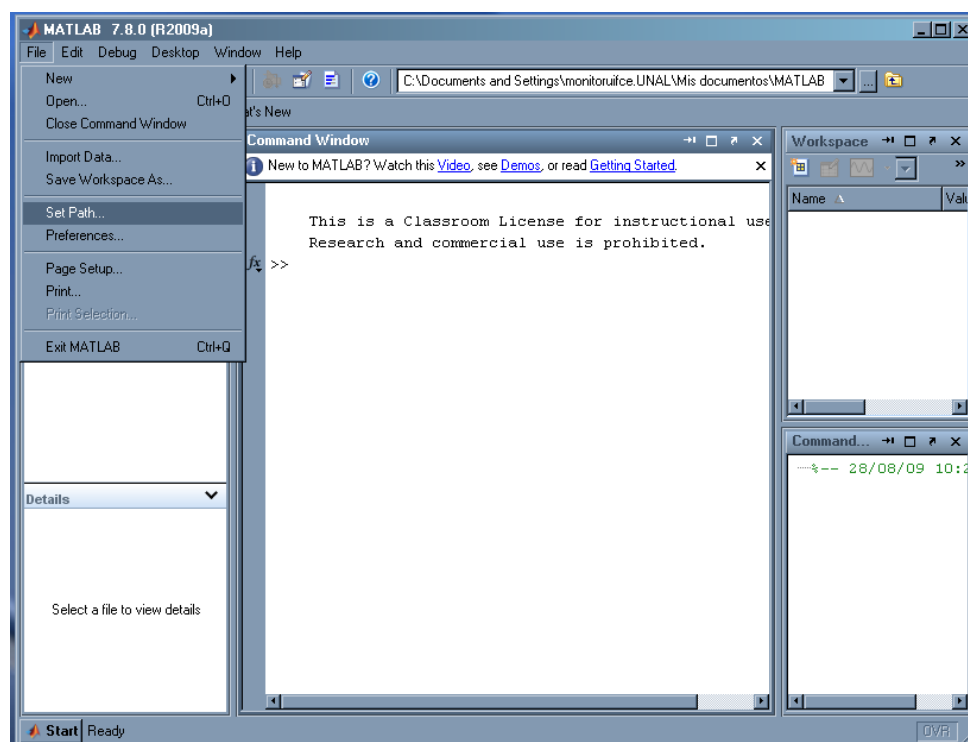


Se realiza la instalación del programa, se escoge la opción siguiente y se da click en finalizar.

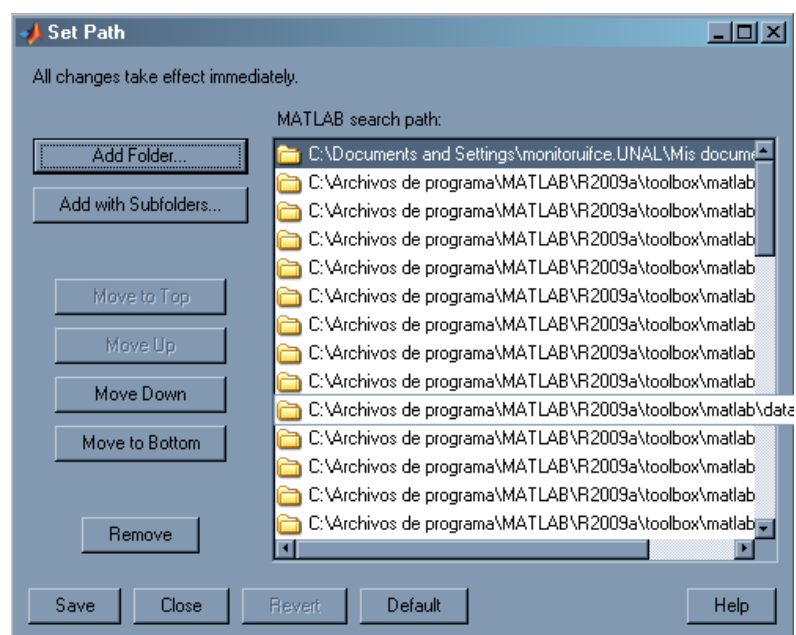


Ya Dynare está instalado en el equipo, sin embargo aún falta agregarlo a Matlab. Para esto se procede como sigue:

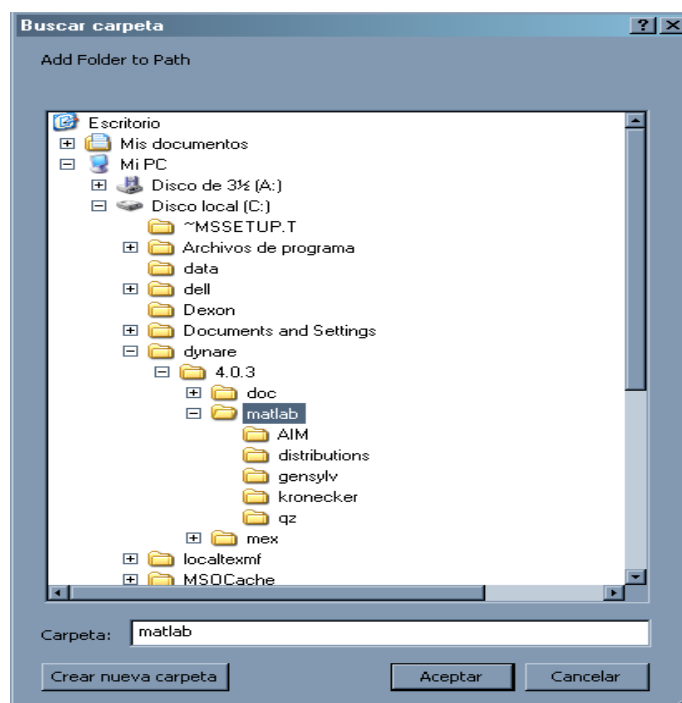
En primer lugar se abre Matlab. En el menú File se escoge la opción Set Path... como se muestra a continuación.



Se abre una ventana como la siguiente, en la cual se escoge la opción Add Folder



A continuación se busca la ubicación de dynare que se le dio en el proceso de instalación (en este caso como se especificó anteriormente es c:\dynare\4.0.3.). En la carpeta de dynare se selecciona la subcarpeta “matlab” y se selecciona la opción Aceptar.

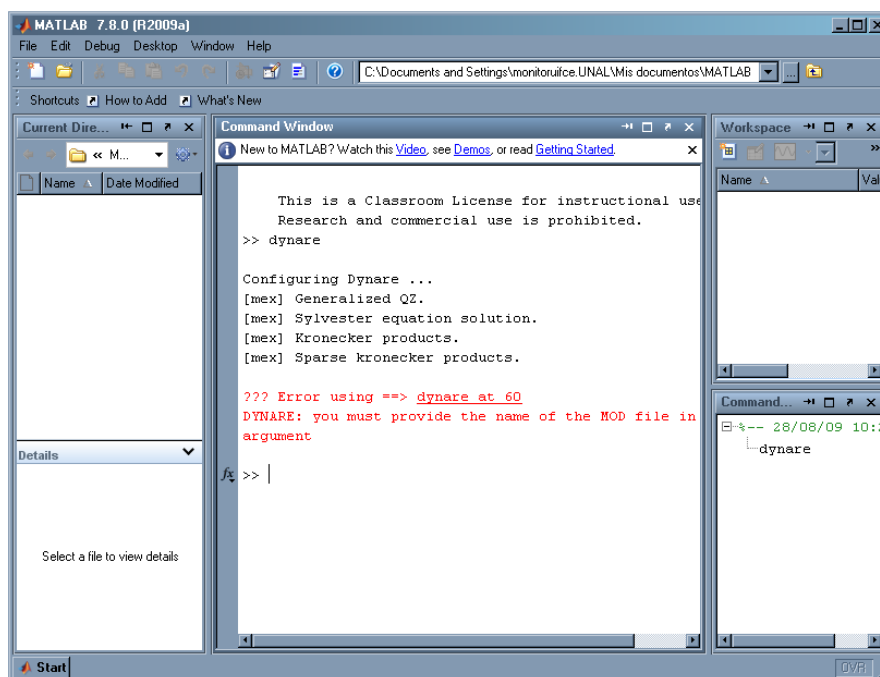


Seguidamente se da la opción “Save” en la ventana “Set Path”.

Para comprobar si Dynare se agregó correctamente a Matlab se escribe en la ventana de comandos “dynare”. Debe aparecer el siguiente error:

Error using ==> dynare at 60
 DYNARE: you must provide the name of the MOD file in
 Argument

Como se muestra en la siguiente ventana.

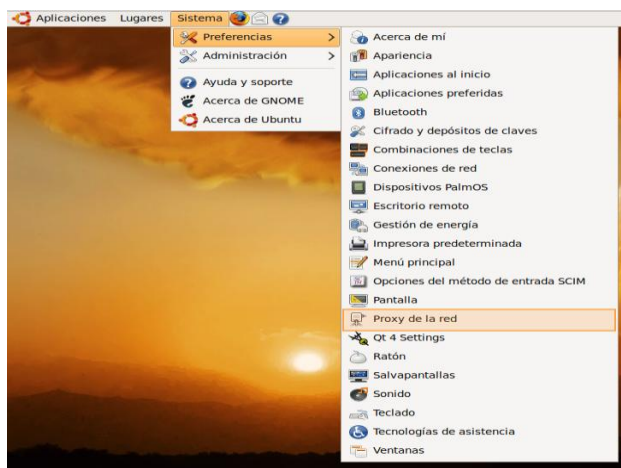


De esta forma Dynare está correctamente agregado a Matlab.

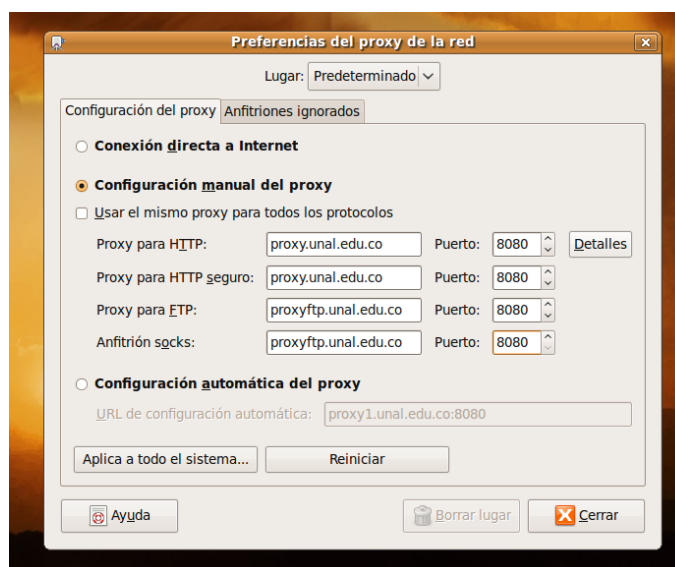
5.3. *Instalación en Debian GNU/Linux y Ubuntu*

A continuación, va a explicarse la instalación de Dynare en Ubuntu "Jaunty Jackalope" 9.04, versión con la que se cuenta en la Unidad de Informática de la Facultad de Ciencias Económicas UIFCE.

Lo primero que debe hacerse es un cambio en la configuración del proxy, para que permita la conexión a Internet. Este se realiza en el menú sistema, en el submenú preferencias se escoge la opción proxy de la red.



Se activa la Configuración manual del proxy. En proxy para HTTP y para HTTP seguro, utilizamos proxy.unal.edu.co; en proxy para FTP y en Anfitrión socks utilizamos proxyftp.unal.edu.co. En todos utilizamos el puerto 8080.



En detalles se activa la opción Usar autenticación y se activa el nombre de usuario y la contraseña del sia.



Luego de realizado este procedimiento, procede a abrirse la consola. Los

comandos que deben utilizarse para establecer la conexión con la página del programa son los siguientes:

```
sudo wget -O - http://www.dynare.org/dynare.public.key | sudo apt-key add -
```

Una vez establecida la conexión, agregamos los repositorios al sources.list.



```
administrador@administrador-desktop: ~
Archivo Editar Ver Terminal Ayuda
administrador@administrador-desktop:~$ sudo wget -O - http://www.dynare.org/dyna
re.public.key | sudo apt-key add -
[sudo] password for administrador: [sudo] password for administrador:
--2009-11-09 12:07:19-- http://www.dynare.org/dynare.public.key
Resolviendo proxy.unal.edu.co... 168.176.5.136, 168.176.55.15, 168.176.55.18, ..
.
Conectando a proxy.unal.edu.co|168.176.5.136|:8080... conectado.
Petición Proxy enviada, esperando respuesta... 200 OK
Longitud: 1722 (1,7K) [application/pgp-keys]
Guardando: «STDOUT»

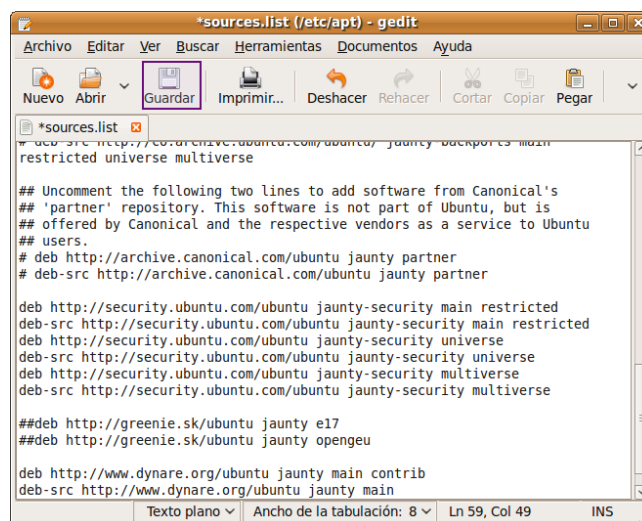
100%[=====] 1.722 ---K/s en 0s

2009-11-09 12:07:20 (125 MB/s) - '-' guardado [1722/1722]

administrador@administrador-desktop:~$ sudo gedit /etc/apt/sources.list
```

Los repositorios son:

```
deb http://www.dynare.org/ubuntu jaunty main contrib
deb-src http://www.dynare.org/ubuntu jaunty main
```



```
*sources.list (/etc/apt) - gedit
Archivo Editar Ver Buscar Herramientas Documentos Ayuda
Nuevo Abrir Guardar Imprimir... Deshacer Rehacer Cortar Copiar Pegar
*sources.list
deb http://security.ubuntu.com/ubuntu jaunty-security main restricted
deb-src http://security.ubuntu.com/ubuntu jaunty-security main restricted
deb http://security.ubuntu.com/ubuntu jaunty-security universe
deb-src http://security.ubuntu.com/ubuntu jaunty-security universe
deb http://security.ubuntu.com/ubuntu jaunty-security multiverse
deb-src http://security.ubuntu.com/ubuntu jaunty-security multiverse

##deb http://greenie.sk/ubuntu jaunty e17
##deb http://greenie.sk/ubuntu jaunty opengeu

deb http://www.dynare.org/ubuntu jaunty main contrib
deb-src http://www.dynare.org/ubuntu jaunty main
```

Luego de guardar los cambios y cerrar el sources.list, se actualizan los repositorios en la consola mediante el comando

```
sudo apt-get update
```

Finalmente se utiliza el comando de instalación:

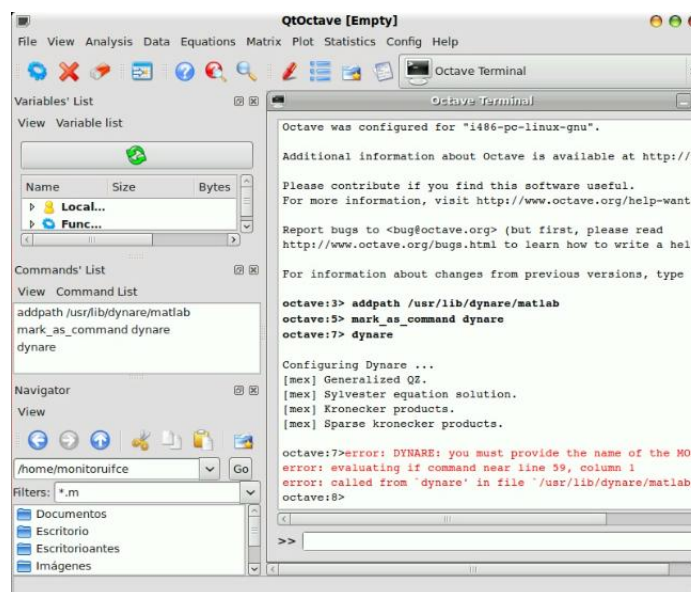
```
sudo apt-get install dynare
```

Similar a la instalación en Windows, debido a que Dynare trabaja bien sea sobre Matlab o sobre GNU Octave, luego de instalado debe configurarse en el respectivo programa. A continuación se explica cómo se configura Dynare para GNU Octave.

Procedemos a abrir QtOctave, que es la forma gráfica de Octave, y en el cuadro de comandos copiamos los siguientes:

```
addpath /usr/lib/dynare/matlab
mark_as_command dynare
```

Igual que en la instalación en Windows, comprobamos que el programa quedó correctamente agregado a Octave si al digitar “dynare”, el programa nos arroja un error que indica que no se reconoce el archivo .MOD con el que se está trabajando.



De esta forma Dynare está correctamente instalado y agregado a Octave.

6. SOLUCIÓN DE MODELOS BÁSICOS EGDE

6.1. UNA DISTINCIÓN FUNDAMENTAL

Es importante reconocer los distintos tipos de modelo, pues esta distinción es fundamental. Se trata de conocer si el modelo es estocástico o determinístico. La diferencia depende de si se conocen las crisis futuras. En modelos deterministas, la presencia de todas las crisis futuras se sabe exactamente en la hora de calcular la solución del modelo. En los modelos estocásticos, en cambio, sólo se conoce la distribución de las futuras crisis. Se va a considerar un choque a una innovación en el modelo sólo en el período I. En un contexto determinista, los agentes toman sus decisiones a sabiendas de que los valores futuros de las innovaciones serán cero en todos los períodos venideros. En un contexto estocástico, los agentes toman sus decisiones sabiendo que los valores futuros de las innovaciones serán al azar. Esto no es lo mismo debido a la desigualdad de Jensen. Por supuesto, si se considera sólo una aproximación de primer orden lineal del modelo estocástico o de un modelo lineal, en los dos casos a ser prácticamente la misma, debido a la certeza de la equivalencia. Una aproximación de segundo orden en cambio dará lugar a resultados muy diferentes, debido a la importancia de la varianza de los choques.

El método de solución para cada uno de estos tipos de modelo difiere de manera significativa. En modelos deterministas, una solución de alta precisión se puede encontrar mediante métodos numéricos. La solución no es más que una serie de números que coinciden con un conjunto determinado de ecuaciones. Intuitivamente, si un agente ha de tener una suposición perfecta, se puede especificar hoy - en el momento de tomar la decisión - con precisión cómo serán sus acciones en el futuro. En un entorno estocástico, en cambio, lo mejor que el agente puede hacer es especificar una decisión, política o regla de retroalimentación para el futuro, por lo que sus acciones están condicionadas a la realización de cada posible choque. En este caso, por lo tanto, se busca una función que satisfaga las condiciones de primer orden del modelo. Para complicar las cosas, esta función puede ser no-lineal y por lo tanto debe ser aproximada. En la teoría de control, las soluciones a modelos determinísticos se llaman soluciones de "circuito cerrado", y de modelos estocásticos se denominan de "ciclo abierto".

6.1.1. Modelos determinísticos vs estocásticos

Los modelos determinísticos tienen las siguientes características:

- Como la literatura del modelo EGDE (estocástico) se ha ganado la atención en la economía, los modelos determinísticos se han convertido en algo raro. Los ejemplos incluyen los modelos OLG (Modelos de Generaciones Traslapadas) sin incertidumbre agregada.
- Estos modelos suelen ser introducidos para estudiar el impacto de un cambio en el régimen, como la introducción de un nuevo impuesto, por ejemplo.
- Asume toda la información, hay suposición perfecta y no hay incertidumbre en torno a los choques.
- Los choques pueden afectar a la economía de hoy o la de cualquier momento en el futuro, dado el caso de previsión perfecta. También puede durar uno o varios períodos.
- Muy a menudo, sin embargo, los modelos introducen un choque positivo hoy y ningún choque a partir de entonces (con certeza).
- La solución no requiere de linealización, de hecho, ni siquiera realmente necesita de un estado estacionario. En su lugar, se trata la simulación numérica para encontrar las rutas exactas de las variables endógenas de primer orden que cumplan con las condiciones del modelo y la estructura del choque.
- Este método de solución por lo tanto puede ser útil cuando la economía está muy lejos del estado estacionario.

Los modelos estocásticos, en cambio, tienen las siguientes características:

- Estos tipos de modelos tienden a ser más populares en la literatura. Ejemplo: incluyen la mayoría de los modelos de RBC (Ciclo Real de Negocios), o nuevos modelos monetarios keynesianos.
- En estos modelos, los choques golpean el día de hoy (con una sorpresa), pero después su valor esperado es cero. Se esperan choques futuros, o cambios permanentes en las variables exógenas que no pueden ser manejados por el uso de aproximaciones de Taylor en torno a un estado estacionario.
- Hay que tener en cuenta que cuando estos modelos son linealizados al primer orden, los agentes se comportan como si los choques futuros fueran iguales a cero (ya que su expectativa es nula), que es la certeza de

la propiedad de equivalencia. Se trata de una frecuencia en un punto alto en la literatura que induce a un error de los lectores en el supuesto de que sus modelos puedan ser determinísticos.

6.2. *Introduciendo un ejemplo*

A continuación se introducirá un modelo RBC básico con competencia monopolística. El objetivo es mostrar cómo se construye dicho modelo en código Dynare, exponiendo el caso estocástico y el determinístico, haciendo las salvedades pertinentes para cada caso. Lo primero entonces es la explicación teórica del modelo. Debe tenerse en cuenta que hablar de expectativas sólo es relevante en un entorno estocástico, como se mencionó anteriormente. Por lo tanto, durante la explicación del caso determinístico tomando como base este modelo, debe tenerse un poco de imaginación para entender que se tratan de casos de perfecta previsión, y así omitir los signos de expectativas.

Pasando entonces a describir el modelo, los hogares maximizan su utilidad sobre el consumo c_t , y sobre el ocio $1 - l_t$, donde l_t es el trabajo, de acuerdo a la siguiente función de utilidad:

$$\mathbb{E}_t \sum_{t=0}^{\infty} \beta [\log c_t + \psi \log (1 - l_t)]$$

Sujeta a la siguiente restricción presupuestal

$$c_t + k_{t+1} = w_t l_t + r_t k_t + (1 - \delta) k_t, \quad \text{para todo } t > 0$$

Donde k_t es el stock de capital, w_t el salario real, r_t tasa de interés real o costo de capital, y δ es la tasa de depreciación.

La ecuación anterior puede verse como una identidad contable, con el gasto total en el lado izquierdo y los ingresos (incluyendo la depreciación del capital) en el lado derecho. Alternativamente, con un poco más de imaginación, la ecuación también puede interpretarse como una ecuación de acumulación de capital, situando c_t en el lado derecho, y notando que $w_t l_t + r_t k_t$ es el total de la remuneración de los factores, lo cual es igual a y_t , si se tiene en cuenta la condición de beneficios nulos. Como consecuencia, si se define la inversión como $i = y_t - c_t$, reemplazando se obtiene que $i = k_{t+1} - (1 - \delta) k_t$, lo cual expresa que la inversión repone el stock de capital, contrarrestando los efectos de la depreciación. En un periodo dado, por lo tanto el consumidor se enfrenta a una disyuntiva entre el consumo y la inversión con el fin de aumentar el stock de

capital y consumir más en los períodos siguientes (como veremos más adelante, la producción depende del capital).

La maximización del problema de los hogares con respecto al consumo, el ocio y el capital se someten a la ecuación de Euler en el consumo, capturando el

$$\frac{1}{c} = \beta \mathbb{E}_t \left[\left(\frac{1}{c+1} \right) (1 + (r+1) - \delta) \right]$$

equilibrio intertemporal, y a la ecuación de la oferta de trabajo que relaciona el trabajo positivamente con los salarios, y negativamente con el consumo (entre más ricos prefieren más ocio debido a la utilidad marginal decreciente del consumo). Estas ecuaciones son:

y

Por otro lado, el problema de la empresa es un poco más complicado debido a la existencia de la competencia monopolística; sin embargo, a continuación se ilustra

$$\psi \frac{c_t}{1 - l_t} = w$$

de la forma más simple posible.

Existen dos formas de introducir la competencia monopolística. Puede asumirse que las empresas venden variedades diferenciadas de un bien a los consumidores, quienes eligen dichas variedades de acuerdo a una función de utilidad de elasticidad de sustitución constante (CES). O puede postularse que existe un continuum de productores intermedios con poder de mercado, en el que cada uno vende una variedad diferente a unos productores de bienes finales competitivos, cuya función de producción es una CES agregada de variedades intermedias.

Si se sigue la segunda opción, el productor de bienes finales escoge su demanda óptima de cada variedad, obteniendo una curva de demanda a la Dixit-Stiglitz. Los productores de bienes intermedios, en cambio, se enfrentan a las siguientes decisiones, cuánto trabajo y capital emplear dados los precios de competencia perfecta de los factores, y cómo fijar el precio de la variedad que ellos producen.

La producción de bienes intermedios sigue una función de producción de Retornos Constantes a Escala (CRS), definida como:

$$y_{it} = k_{it}^{\alpha} (e^{z_t} l_{it})^{1-\alpha}$$

Donde el subíndice i indica que es la empresa i de un continuum de firmas entre 0

y 1, y donde α es la elasticidad del capital en la función de producción, con $0 < \alpha < 1$. Además, z_t captura la tecnología, la cual evoluciona de acuerdo a:

En donde ρ es un parámetro que captura la persistencia del progreso tecnológico, y donde $e_t \sim N(0, \sigma)$.

$$z_t = \rho z_{t-1} + e_t$$

La solución al problema de abastecimiento conduce a una relación capital-trabajo óptima, o a una relación óptima entre la remuneración de los factores:

$$k_{it} r_t = \frac{\alpha}{1 - \alpha} w_t l_{it}$$

La solución al problema de la fijación de precios, por el contrario, conduce a la bien conocida condición de precio constante del mercado de la competencia monopolística:

$$p_{it} = \frac{\epsilon}{\epsilon - 1} mc_t p_t$$

Donde p_{it} es el precio específico de la firma i , mc_t es el costo marginal real, p_t es el precio agregado CES o el precio medio, y el parámetro ϵ representa la elasticidad de sustitución de las diferentes variedades; cuanto más alto es ϵ , menor es el markup ya que hay menor poder monopolístico. Un paso adicional simplifica esta expresión: el supuesto de que las firmas son simétricas implica que todas las firmas asignan el mismo precio, entonces $p_{it} = p_t$. Entonces se tiene $mc_t = \frac{\epsilon - 1}{\epsilon}$.

Para saber a qué es igual el costo marginal se reemplaza la relación óptima de capital-trabajo en la función de producción, y se aprovechan sus propiedades al ser una función de CRS, para saber la cantidad de trabajo o de capital necesarios para producir una unidad de producto. El costo real de utilizar esa cantidad de cualquiera de los dos factores es dada por $w_t l_t + r_t k_t$ donde se sustituyen los pagos del otro factor usando de nuevo la relación óptima de capital-trabajo. Solucionando para el trabajo se tiene:

$$mc_t = \left(\frac{1}{1 - \alpha} \right)^{1 - \alpha} \left(\frac{1}{\alpha} \right)^{\alpha} \frac{1}{A_t} w_t^{1 - \alpha} r_t^{\alpha}$$

Éste no depende de i , es decir, es el mismo para todas las firmas.

Curiosamente, lo anterior se puede resolver usando la relación óptima entre capital-trabajo, para llegar a $w_t \left[(1 - \alpha) \frac{y_{it}}{l_{it}} \right]^{-1}$, o $w_t \frac{\partial l_{it}}{\partial y_{it}}$, que es la definición de costo marginal: el costo en términos de unidades de trabajo para producir una unidad adicional de producto. No debería ser una sorpresa, dado que la relación óptima entre capital-trabajo, sigue la maximización de la función de producción (menos los costos reales) con respecto a sus factores.

Combinando estos resultados con el costo marginal, así como su contraparte en términos de capital, con la condición de precios óptimos, conduce a las últimas dos ecuaciones importantes del modelo.

$$w_t = (1 - \alpha) \frac{y_{it} (\epsilon - 1)}{l_{it}^\epsilon}$$

y

$$r_t = \alpha \frac{y_{it} (\epsilon - 1)}{k_{it}^\epsilon}$$

Para terminar, se agrega la producción de cada firma para encontrar una función agregada de producción. Por el lado de la oferta, se factoriza la relación capital-trabajo, $\frac{k_t}{l_t}$, que es el mismo para todas las empresas y por lo tanto no depende de i . Por otro lado, se tiene una demanda tipo Dixit-Stiglitz para cada variedad. Al igualar las dos e integrando a ambos lados, y observando que la dispersión de precios es nula, o como se indicó anteriormente $p_{it} = p_t$, se obtiene la producción agregada:

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

Que puede ser mostrado como la suma de total de las variedades compradas por el productor del bien final (de acuerdo con su función CES) y, a su vez, igual a la producción agregada del bien final, que equivale al consumo de los hogares. Finalmente, tenga en cuenta que debido a que la proporción de cada factor en la producción es el mismo para cada empresa productora de bienes intermedios, y que cada firma, así como la producción, tienen retornos constantes a escala, pueden escribirse esas dos ecuaciones para w_t y para r_t , sin el subíndice i .

6.3. *Estructura de los archivos .mod en Dynare*

Para invocar a Dynare, se necesita un archivo .mod, el cual puede estar escrito en cualquier editor, externo o interno de Matlab. Este será luego leído por el programa base (Matlab); el archivo .mod se invoca escribiendo en el cuadro de comandos: Dynare nombredelarchivo.mod o simplemente escribiendo el nombre del archivo.

Sin embargo, antes de esto se mostrará la estructura de los archivos .mod.

Están divididos en cinco bloques como sigue:

- Preámbulo, el cual muestra la lista de las variables y los parámetros del modelo.
- Modelo, en el cual se detallan las ecuaciones del modelo.
- Estado estacionario o valor inicial, donde se dan las indicaciones para encontrar el estado estacionario de un modelo, o el punto de inicio de las simulaciones o funciones que darán solución al modelo.
- Choques, se definen los choques del sistema.
- Computación, encarga a Dynare a llevar a cabo operaciones específicas, por ejemplo, previsión, estimación de funciones de respuesta de impulso, etc.

Antes de empezar a explicar cada bloque, hay que tener en cuenta un par de observaciones:

- ✓ Cada instrucción y cada elemento del archivo .mod debe terminar con punto y coma (;) Una misma instrucción puede ocupar más de una línea, en este caso el punto y coma se pone al final de la instrucción, no al final de la línea.
- ✓ Al finalizar todos los bloques, se termina con end.
- ✓ Pueden hacerse comentarios escribiendo un doble slash al principio de la línea (//), o, si se hace en varias líneas, debe escribirse al principio: /*, y al final: */

6.4. **PREÁMBULO**

Generalmente involucra tres comandos que indican a Dynare cuáles son las variables del modelo, las variables endógenas y los parámetros.

Los comandos son:

- `var`: declara las variables endógenas.
- `varexo`: muestra la lista de las variables exógenas. Estas son necesarias para poder aplicar choques al modelo.
- `varexo_det`: es un comando opcional que declara las variables determinísticas en un modelo estocástico.

Es posible mezclar los choques determinísticos y estocásticos para construir modelos en los que los agentes saben desde el principio de la simulación sobre los cambios exógenos futuros. En ese caso, el comando `stoch_simul` calcula la solución de expectativas racionales agregando información futura al espacio de estado; mientras que el comando `forecast` calcula una simulación condicional sobre las condiciones iniciales y sobre la información futura

- `parameters`: indica la lista de parámetros. Seguidamente pueden indicarse los valores asociados a cada uno de éstos.

Cada comando puede repetirse varias veces en el archivo, pues Dynare lo concatenará.

A continuación, utilizando el ejemplo del modelo RBC básico con competencia monopolística, se mostrarán las diferencias entre un modelo determinístico, y uno estocástico.

6.4.1. Caso determinístico

El modelo es el mismo que el descrito anteriormente, excepto que ya no se necesita la variable et , pues estamos quitando la parte de las expectativas, pues ahora puede trabajarse con zt directamente como variable exógena. Entonces el preámbulo quedaría:

```
var y c k i l y l w r;  
varexo z;  
parameters beta psi delta alpha sigma epsilon;  
alpha = 0.33;  
beta = 0.99;  
delta = 0.023;  
psi = 1.75;  
sigma = (0.007/(1-alpha));  
epsilon = 10;
```

6.4.2. Caso estocástico

En este caso, vuelve a considerarse la evolución de la tecnología como se explicó teóricamente en el ejemplo, que consiste en un choque exógeno, et . Con respecto a la anterior, por lo tanto, se ajusta la lista de variables endógenas y exógenas y se agrega el parámetro ρ . En este caso, el preámbulo iría:

```
var y c k i l y l w r z;  
varexo e;  
parameters beta psi delta alpha rho sigma epsilon;  
alpha = 0.33;  
beta = 0.99;  
delta = 0.023;  
psi = 1.75;  
rho = 0.95;  
sigma = (0.007/(1-alpha));  
epsilon = 10;
```

6.5. Especificaciones del modelo

6.5.1. Declaración del modelo en Dynare

Una de las facilidades de Dynare es la forma en que permite escribir las ecuaciones del modelo, pues es casi igual a como se escribirían para un trabajo académico. Son necesarias sólo unas pocas convenciones. Continuando con el ejemplo del modelo RBC con competencia monopolística, a continuación se especifican las ecuaciones básicas para luego ser escritas en código Dynare.

Las ecuaciones son:

$$\frac{1}{c} = \beta \mathbb{E}_t \left[\left(\frac{1}{c+1} \right) (1 + (r+1) - \delta) \right]$$

Ecuación de Euler en el consumo.

$$\psi \frac{c_t}{1 - l_t} = w$$

Función de oferta de trabajo.

$$c + i = y$$

Identidad contable.

$$y_{it} = k_{it}^\alpha (e^{z_t} l_{it})^{1-\alpha}$$

Función de producción.

$$w_t = (1 - \alpha) \frac{y_{it}}{l_{it}} \frac{(\epsilon - 1)}{\epsilon}$$

Ecuaciones que igualan el costo marginal al poder de mercado.

$$r_t = \alpha \frac{y_{it}}{k_{it}} \frac{(\epsilon - 1)}{\epsilon}$$

$$i_t = k_{t+1} - (1 - \delta)k_t$$

Igualdad de la inversión

$$y_l = \frac{y}{l}$$

Identidad

$$z_t = \rho z_{t-1} + e_t$$

Ecuación del cambio de la tecnología

En el archivo .mod , este modelo se declara de la siguiente forma:

```
model;

(1/c) = beta*(1/c(+1))*(1+r(+1)-delta);
psi*c/(1-l) = w;
c+i = y;
y = (k(-1)^alpha)*(exp(z)*l)^(1-alpha);
w = y*((epsilon-1)/epsilon)*(1-alpha)/l;
r = y*((epsilon-1)/epsilon)*alpha/k(-1);
i = k-(1-delta)*k(-1);
y l = y/l;
z = rho*z(-1)+e;

end;
```

6.5.2. Generalidades

Este ejemplo ilustra el uso de algunos comandos y convenciones importantes para transformar un modelo en un archivo .mod de Dynare legible:

- Lo primero que hay que resaltar, es que el bloque del modelo del archivo .mod, comienza con el comando model y termina con el comando end.
- Segundo, en medio de estos comandos, cabe mencionar que deben tenerse tantas ecuaciones como variables endógenas se determinaron al principio. De hecho, esto es una de las primeras cosas que Dynare revisa; el programa hará saber inmediatamente si hay algún problema respecto a esto.
- Tercero, al igual que en el preámbulo y como en todo el archivo .mod, cada línea de instrucción debe terminar con punto y coma (;), excepto cuando la instrucción continúa en la siguiente línea, como se explicó anteriormente. Esta característica es diferente en Matlab, donde, si desea continuarse con la instrucción en la siguiente línea, debe continuarse con puntos suspensivos (...).
- Cuarto, las ecuaciones deben introducirse una tras otra (el programa no reconoce la representación matricial). Los nombres de las variables y de los parámetros utilizadas en el bloque del modelo deben ser los mismos que los utilizados en el bloque del preámbulo.

Recuerde que los nombres las variables y de los parámetros son sensibles a las mayúsculas.

6.5.3. Convenciones en las notaciones

Las variables que en el modelo están definidas en el tiempo t se determinan en el modelo sólo con el nombre de la variable. Por ejemplo, de última ecuación, z_t es escrita en el .mod como z .

Las variables que en el modelo están definidas en $t-n$ deben escribirse con un $(-n)$ en frente de éstas. Por ejemplo, la variable x_{t-2} debe ser escrita como $x(-2)$.

De igual forma, las variables definidas en $t+n$ deben escribirse con un $(+n)$ en frente.

6.5.4. Condiciones sobre la temporalidad de las variables

En Dynare, la temporalidad de cada variable indica cuándo las variables son decididas. En el modelo del ejemplo en el que el stock de capital no es algo que se decida el día de hoy sino que depende de las decisiones tomadas ayer (lo que se define como variable predeterminada), la variable descrita como: $k_{t+1} = i_t + (1-\delta)k_t$, en el lenguaje Dynare debe escribirse como $k=i+(1-\delta)*k(-1)$.

A manera de otro ejemplo, considerando algunos modelos de negociación de salarios en los que los salarios son fijados en los períodos anteriores. En este caso, la ecuación del salario puede escribirse en el período t (el período en el que se definen), pero en la ecuación de la demanda de trabajo, los salarios deben aparecer con un período de desfase.

6.5.5. Convenciones para las variables no predeterminadas

Cuando hay un $(+1)$ junto a la variable se le está indicando a Dynare que debe reconocer la realización de la variable en un salto en el futuro, es decir, la variable no está predeterminada.

Las condiciones Blanchard-Kahn solo se cumplen si el número de variables no predeterminadas es igual al número de valores propios mayores que uno, pues si es mayor, el modelo es explosivo y no tiene solución convergente; por el contrario, si es menor el modelo es indeterminado, es decir, existen infinitas soluciones convergentes. Si esta condición no se cumple, Dynare dará el respectivo aviso.

Una variable puede ocurrir de forma predeterminada como no predeterminada. Por ejemplo, el consumo puede aparecer de forma adelantada en la ecuación de Euler (forma no predeterminada), pero también puede aparecer con un rezago en una ecuación de la formación de hábitos, pues esta dependería de consumos anteriores (forma predeterminada). En este caso la diferenciación de segundo orden tendría dos valores propios, uno superior y otro inferior a uno para la estabilidad.

6.5.6. Modelos lineales y logarítmicos linealizados

Existen otras dos variantes del sistema de ecuaciones que Dynare puede trabajar. Estos son los modelos lineales y los no lineales (en exp y log). En el primer caso, todo lo que es necesario es escribir el termino (linear) en frente del comando model. En el modelo que se ha venido trabajando, por ejemplo, para la igualdad $y_l = y$ que correspondía en dynare a $y_l = y/l$; sería:

```
model (linear);
yy_l=yy - ll;
end;
```

Donde repetir la letra para la variable significa la diferencia del estado estacionario.

Si lo que se desea es transformar las variables y trabajarlos mediante exponenciales o funciones logarítmicas, lo que se hace es transformarlas de la siguiente manera:

Recordemos que en el modelo inicial se tenía:

```
model;
(l/c) = beta*(l/c(+1))*(1+r(+1)-delta);
psi*c/(1-l) = w;
...
end;
```

Entonces transformamos el modelo de la siguiente manera:

```
model;
(1/exp(cc)) = beta*(1/exp(cc(+1)))*(1+exp(rr(+1))-delta);
psi*exp(cc)/(1-exp(ll)) = exp(ww);
end;
```

Cabe mencionar que a manera de ejemplo se muestran solo las dos primeras ecuaciones del modelo. Sin embargo debe hacerse para todas las variables. Esta vez, repetir la letra de la variable indica que se está transformando la variable de forma logarítmica, donde el nivel de una variable está dado por $\exp(\text{la variable repetida})$.

6.6. Especificación de estados estacionarios y/o valores iniciales

Para evitar confusiones y entender de qué se trata esta sección dentro del archivo .mod , debemos tener en cuenta ciertos aspectos. Primero debe recordarse que los modelos estocásticos necesitan ser linealizados. Por lo tanto éstos necesitan tener un estado estacionario. En segundo lugar, independientemente de si se está trabajando con un modelo estocástico o determinístico, lo más probable es que se esté interesado en iniciar las simulaciones o las funciones de respuestas a impulsos a partir de un estado de equilibrio o a partir de otro punto.

En esta sección se busca proveer los valores de esos estados de equilibrio o aproximaciones a dichos valores.

Los comandos más relevantes en esta sección son: `initval`, `endval`, o de forma menos usual `histval`. A continuación se hará una breve descripción de esos comandos:

- `initval` : Especifica los valores numéricos iniciales para encontrar el estado estacionario y/o los valores iniciales para las simulaciones.
El block `initval` tiene dos propósitos: declarar las condiciones iniciales (y posiblemente las finales) en un ejercicio de simulación; y proveer valores aproximados para las soluciones no lineales.
- `endval` : Sólo se utiliza en modelos determinísticos para especificar los valores finales para dicho tipo de simulaciones. Tiene dos propósitos: primero, establece las condiciones finales para todos los periodos siguientes al último período de la simulación. Segundo, provee los valores iniciales aproximados de todas las simulaciones para las soluciones no lineales.
Así, Dynare calcula un primer estado estacionario con los valores dados en el `initval`, y calcula un estado estacionario final para los valores dados en el `endval`.
- `histval` : Especifica los valores históricos antes de comenzar la simulación; es un comando opcional que se utiliza para los modelos que tienen rezagos en más de un período.

Por convención, en Dynare se reconoce el período 1 como el primer período de la simulación. Por lo tanto, el período inmediatamente anterior al comienzo de la simulación es el período 0, antes estará el período -1 y así sucesivamente.

Para entender mejor estos comandos, es aconsejable remitirse al manual, donde se encontrarán ejemplos de cada uno de estos.

6.6.1. Modelos estocásticos y estado estacionario

Como se mencionó anteriormente, un modelo estocástico debe ser linealizado antes de ser resuelto. Para hacerlo, Dynare necesita conocer el estado estacionario. Se pueden ingresar los valores exactos del estado estacionario en el archivo .mod o se pueden ingresar sólo aproximaciones y dejar que Dynare encuentre el equilibrio exacto, el cual lo hará mediante métodos numéricos basados en las aproximaciones dadas. En cualquier caso, esos valores son ingresados en el block `initval` como se muestra a continuación.

```
initval;  
k = 9;  
c = 0.7;  
l = 0.3;  
w = 2.0;  
r = 0;  
z = 0;  
e = 0;  
end;
```

```
steady;
```

Usando el comando `steady` puede controlarse si se desea comenzar con la simulación a partir del estado estacionario, o a partir de los valores exactos proporcionados en el block `initval`. Agregando el comando `steady` luego de finalizar el block `initval` (es decir, luego de la instrucción `end;` como se muestra en el ejemplo anterior) se le pide a Dynare considerar los valores iniciales, proporcionados en el block, solamente como aproximaciones y empezar las simulaciones del estado estacionario exacto.

Por el contrario, si no se agrega dicho comando, las simulaciones empezarán a partir de los valores dados, incluso si antes el programa ya ha calculado el estado estacionario exacto del modelo en el momento de la linealización.

Al respecto, es conveniente tener en cuenta que si se trabaja con un modelo estocástico, su linealización es buena sólo alrededor del estado estacionario, por lo cual lo más recomendado es empezar la simulación desde el estado estacionario, es decir, utilizando el comando `steady`.

6.6.2. Modelos determinísticos y valores iniciales

Los modelos determinísticos no necesitan ser linealizados para ser resueltos. Entonces, técnicamente, no se necesita dar un estado estacionario para resolver el modelo. Sin embargo, en la práctica las investigaciones están enfocadas a analizar cómo reaccionan los modelos a los choques cuando inicialmente se encuentran en el equilibrio. En el caso determinístico el `block initval` cumple funciones muy similares a las descritas anteriormente, de esta manera, si se quiere introducir el choque partiendo del estado estacionario se utiliza el comando `steady`, y en caso contrario, si se quiere partir de un valor aleatorio no se utiliza ningún comando.

6.6.3. Encontrar el estado estacionario

Lo difícil de lo que se acaba de explicar es por supuesto encontrar los valores del estado estacionario. Para esto, Dynare puede ayudar invocando las funciones adecuadas de Matlab, pero por lo general sólo se tiene éxito si los valores iniciales proporcionados están cerca del estado estacionario real. Si se presenta algún problema para encontrar el equilibrio en el modelo, se puede empezar a jugar con las siguientes opciones del comando `steady` :

`solve algo = 0`: usa el Toolbox de optimización de Matlab, `FSOLVE`.

`solve algo = 1`: usa las propias soluciones de Dynare para ecuaciones no lineales.

`solve algo = 2`: divide el modelo en bloques recursivos y resuelve cada bloque por separado.

`solve algo = 3`: usa el método Sims. Ésta es la opción por defecto si no se especifica ninguno.

Sin embargo, puede que algunos modelos presenten problemas en el momento de hallar el estado estacionario, por lo que otra opción es introducir el modelo linealizado. En este caso, las variables estarán dadas como desviaciones porcentuales del estado estacionario, y los valores iniciales se tomarán como 0, por lo que es necesario calcular el estado estacionario en el modelo no linealizado.

Otra opción muy práctica es utilizar Matlab para encontrar el estado estacionario. En este caso, el m-file que se utiliza para hacer el cálculo de los valores de estado estacionario debe tener el mismo nombre del .mod, seguido por _steadystate. Así, si el .mod se llama ejemplo.mod el m-file debe llamarse ejemplo_steadystate.m y debe ser guardado en el mismo directorio del .mod. Dynare automáticamente revisará si existe algún archivo de Matlab en el directorio en el que se encuentra el .mod, y al guardarlo de esta manera, utilizará ese archivo para encontrar los valores de estado estacionario, a pesar de haber introducido otros valores iniciales en el .mod.

Hay que recordar que Matlab no trabaja con expresiones analíticas, por lo cual no sirve escribir el modelo de la forma en la que se escribió el .mod, es decir, deben escribirse las ecuaciones como si se resolviera la operación a mano.

6.6.4. Verificar la estabilidad del sistema

Como se menciono anteriormente, para que el modelo sea resuelto por el método de solución que utiliza Dynare y para que exista estabilidad alrededor del estado de equilibrio, debe cumplirse que existan tantos valores propios mayores a uno, como variables control. Para esto es útil el comando check, el cual computa y muestra los valores propios del sistema.

El uso de este comando es opcional, pues si la condición no se cumple, Dynare arrojará un mensaje advirtiéndolo que las condiciones de Blanchard-Kahn no se cumplen, úsese o no dicho comando.

6.7. *Añadir choques al sistema*

Cuando se trabajan modelos determinísticos se tiene la opción de introducir choques tanto temporales como permanentes. La diferencia es que bajo un choque temporal, el modelo va a retornar en algún momento al estado estacionario, mientras que bajo un choque permanente, el modelo va a permanecer en el nuevo estado estacionario. Como se vio anteriormente, en esta clase de modelos ambos choques son totalmente esperados.

Por otro lado, en los modelos estocásticos sólo son permitidos los choques temporales. Un choque permanente no puede ser apropiado por el modelo debido a la necesidad de volver estacionario el modelo alrededor del equilibrio. Además, los choques sólo pueden ser introducidos hoy debido a que el valor esperado de los futuros choques debe ser cero.

6.7.1. Modelos determinísticos: choques temporales

Al trabajar con un choque temporal, se está en la libertad de escoger la duración y el nivel del choque. Por ejemplo, para especificar un choque en la tecnología z_t que dure 9 períodos, debe escribirse:

```
shocks;  
var z;  
periods 1:9;  
values 0.1;  
end;
```

Con estas instrucciones Dynare reemplaza el valor de z_t dado en el block `initval` con el valor 0.1. Si las variables están dadas en términos de logaritmos, este corresponderá a un choque del 10%.

Para introducir varios choques puede hacerse de la siguiente manera:

```
shocks;  
var z;  
periods 4 5 6;  
values 0.1 0.3 0.2;  
end;
```

Por otro lado, el comando `mshocks` especifica choques multiplicativos sobre las variables exógenas.

Pueden especificarse choques futuros en el block `shocks` definiendo en los períodos, por ejemplo 3:7, para analizar el comportamiento de los agentes ante futuros choques.

6.7.2. Modelos determinísticos: choques permanente

Para estudiar choques permanentes que afectan el modelo hoy, simplemente se cambian los valores iniciales del block `initval`, y se modifican los nuevos valores en el block `endval`. Dynare se encargará de calcular la transición. Es decir, suponiendo el mismo cambio en la tecnología del ejemplo anterior se tendría:

```
initval;  
k = 9;  
c = 0.7;  
l = 0.3;  
w = 2.0;  
r = 0;
```

```
z = 0;
end;

steady;

endval;
k = 9;
c = 0.7;
l = 0.3;
w = 2.0;
r = 0;
z = 0.1;
end;

steady;
```

En este ejemplo, el cambio de la tecnología a 0.1 se hace en el período 1 (mañana) y permanece de esta manera en el sistema. Sin embargo, las demás variables (las variables endógenas) tardarán más en llegar de nuevo al estado estacionario.

Si lo que se quiere es analizar choques futuros pero permanentes, luego del block endval se agrega un block shocks para “deshacer” los primeros períodos del choque permanente. Por ejemplo, para introducir el choque en tecnología de 0.1 de forma permanente pero a partir del período 10, luego de los comandos escritos arriba debe agregarse:

```
shocks;
var z;
periods 1:9;
values 0;
end;
```

6.7.3. Modelos estocásticos

Como se especificó anteriormente, estos modelos sólo tienen choques temporales. Sin embargo, puede hacerse el efecto de un choque propagado lentamente en la economía introduciendo una “variable de choque latente” como es la variable et introducida en el ejemplo, que afecta la variable z_t (recordemos que z_t se comporta como un modelo $AR(1)$). En ese caso se declara la variable z_t como una variable endógena y la variable et como la variable exógena, como se hizo anteriormente en el preámbulo del .mod.

Suponiendo que quiere introducirse un choque con una varianza σ^2 (σ fue determinado en el preámbulo del .mod), debe escribirse:

```
shocks;  
var e = sigma & 2;  
end;
```

6.8. Selección del cálculo

Hasta ahora se han descrito instrucciones para el archive .mod, pero no se ha explicado acerca de lo que Dynare hace con los anteriores comandos. En la mayoría de los casos, lo que se hace es la función de impulso respuesta (FIR), debido a los choques externos. A continuación se van a explicar los comandos apropiados para dar a Dynare, para esto se va a hacer distinción nuevamente entre modelos determinísticos y estocásticos.

6.8.1. Para modelos determinísticos

En el caso determinístico, lo que se tiene que hacer es añadir el comando simul a la parte inferior del archive .mod. Dicho comando simula el modelo determinístico. La forma de digitarlo es simul [(períodos=INTEGER)]. El comando simul desencadena el cálculo y la simulación numérica de la trayectoria de la solución del modelo para el número de períodos colocado en la opción períodos. Para esto es usado un método Newtoniano que resuelve simultáneamente todas las ecuaciones para todos los períodos. Hay que tener en cuenta que a menos que se utilice el comando endval, el algoritmo hace que se suponga que el sistema está de vuelta al equilibrio después del número especificado de períodos. Por lo tanto se debe especificar un número suficientemente grande, de modo que si se presenta un incremento futuro no cambie la simulación para fines prácticos.

En el caso de un choque temporal, por ejemplo, la trayectoria describirá básicamente como el sistema vuelve al equilibrio después de ser perturbado por los choques que han sido introducidos.

6.8.2. Para modelos estocásticos

En los casos de modelos estocásticos más comunes, es apropiado el comando `stoch_simul`. Éste comando da instrucciones a Dynare para calcular una aproximación tipo Taylor de las funciones de decisión y transición para el modelo (la lista de ecuaciones de valores actuales de las variables endógenas del modelo como una función del estado previo del modelo y los choques actuales), dar la función de impulso de respuesta y varias estadísticas de tipo descriptivo (momentos, descomposición de varianza, correlación y coeficientes de autocorrelación).

Las funciones de impulso de respuesta son rutas futuras esperadas de las variables endógenas condicionales acerca de un choque en el período 1 con una desviación estándar. Si se linealiza el modelo en un primer orden, las funciones de impulso de respuesta son simplemente la iteración algebraica delante de la política del modelo o regla de decisión. Si en cambio se linealiza en un segundo orden, las funciones de impulso de respuesta serán el resultado de la actual simulación de Monte Carlo de futuros choques. Esto se da porque en el segundo orden de las ecuaciones lineales, se tendrán términos cruzados envolviendo los choques, entonces los efectos de los choques dependen del estado del sistema en el que ocurre el choque. Por esto, es imposible obtener valores algebraicos medios de todos los futuros choques y sus impactos. La técnica es, en cambio, halar futuros choques desde sus distribuciones y mirar cómo es el impacto en el sistema, y repetir este procedimiento numerosas veces con el fin de obtener un medio de respuesta. Se debe notar que los futuros choques no tendrán un impacto significativo en los resultados, pues son obtenidos mediante cada ensayo Monte Carlo y en el límite debe sumar cero, resultando el promedio cero. Cabe resaltar que en el caso de un segundo orden de aproximación, Dynare devolverá los momentos actuales de muestra de las simulaciones. Para linealización de primer orden, Dynare en cambio, reporta momentos teóricos. En ambos casos, el regreso al estado estacionario es asintótico. Debe asegurarse el especificar suficientes períodos en las funciones de impulso de respuesta de manera que actualmente se vean las gráficas de retorno al estado estacionario. A continuación se muestran las opciones del comando `stoch_simul`:

- `ar = INTEGER`: Ordena los coeficientes de autocorrelación para calcular y publicar (default = 5).
- `dr_algo = 0` ó `1`: Especifica el algoritmo usado para calcular la aproximación cuadrática de las reglas de decisión.
- `drop = INTEGER`: Número de puntos puestos al principio de la simulación antes de calcular el resumen de estadísticas.

- `hp_filter` = INTEGER: se usa con `lambda` = INTEGER antes de calcular momentos.
- `hp_ngrid` = INTEGER: Número de puntos en la matriz discreta de la Transformada Inversa de Fourier usada en el cálculo HP filter. Puede ser necesario incrementarlo para elevados procesos de autocorrelación.
- `irf` = INTEGER: Número de períodos en los cuales se calculan las funciones de impulso de respuesta.
- `relative_irf`: Requiere el cálculo de funciones de impulso de respuesta (IRF) normalizado en porcentaje del error estándar en cada choque.
- `nocorr`: No imprime la matriz de correlación.
- `nofunctions`: No imprime los coeficientes de solución aproximados.
- `nomoments`: No imprime momentos de variables endógenas.
- `noprint`: Cancela toda impresión.
- `order` = 1 ó 2: Ordena la aproximación de Taylor, a menos que se esté trabajando con un modelo lineal, en este caso el orden es automáticamente establecido para 1.
- `periods` = INTEGER: Especifica el número de períodos para utilizar en las simulaciones.
- `qz_criterium` = INTEGER ó DOUBLE: Valor usado para dividir valores propios estables de inestables reordenando la descomposición usada para resolver problemas de primer orden.
- `replic` = INTEGER: Número de series simuladas usando el cálculo de IRFs.
- `simul_seed` = INTEGER, DOUBLE ó EXPRESSION: Especifica una semilla para la generación de números aleatorios con el fin de obtener la misma muestra aleatoria en cada corrida del programa. Por otra parte, una muestra diferente es usada para cada corrida. Si se linealiza en segundo orden, Dynare se encarga de actualizar la simulación Monte Carlo para generar momentos en las variables. Debido a la simulación, los resultados son arrojados para ser ligeramente diferentes cada vez que se corra el programa, excepto si se fija la semilla para la generación de números aleatorios.

Volviendo al ejemplo, suponer que se está interesado en imprimir todas las diversas medidas de momentos de las variables, se quieren ver las funciones de impulso de respuesta de las variables, se está básicamente satisfecho con todas las opciones por defecto y se quieren llevar a cabo simulaciones encima de un buen número de períodos. Se terminaría el archivo `.mod` con el siguiente comando:

UNIVERSIDAD NACIONAL COLOMBIA
FACULTAD DE CIENCIAS ECONÓMICAS
UNIDAD DE INFORMÁTICA Y COMUNICACIONES


```
stoch_simul(periods=2100);
```

6.9. EL ARCHIVO .MOD COMPLETO

A continuación se presentan compiladas las dos estructuras de los archivos .mod para los ejemplos tratados: caso estocástico y determinístico del modelo RBC con competencia monopolística

6.9.1. El modelo estocástico

```
var y c k i l y l w r z;

varexo e;

parameters beta psi delta alpha rho sigma epsilon;
alpha = 0.33;
beta = 0.99;
delta = 0.023;
psi = 1.75;
rho = 0.95;
sigma = (0.007*(1-alpha));
epsilon = 10;

model;
(l/c) = beta*(l/c(+1))*(1+r(+1)-delta);
psi*c/(1-l) = w;
c+i = y;
y = (k(-1)^ alpha)*(exp(z)*l)^(1-alpha);
w = y*((epsilon-1)/epsilon)*(1-alpha)/l;
r = y*((epsilon-1)/epsilon)*alpha/k(-1);
i = k-(1-delta)*k(-1);
y l = y/l;
z = rho*z(-1)+e;
end;

initval;
k = 9;
c = 0.7;
l = 0.3;
w = 2.0;
r = 0;
z = 0;
```

```

e = 0;
end;

steady;

check;

shocks;
var e = sigma 2;
end;

stoch simul(periods=2100);

```

6.9.2. El modelo determinístico (caso de choques temporales)

```

var y c k i l y_l w r;

varexo z;

parameters beta psi delta alpha sigma epsilon;
alpha = 0.33;
beta = 0.99;
delta = 0.023;
psi = 1.75;
sigma = (0.007*(1-alpha));
epsilon = 10;

model;
(l/c) = beta*(l/c*(1))*(1+r*(1)-delta);
psi*c/(1-l) = w;
c+i = y;
y = (k*(1)^alpha)*(exp(z)*l)^(1-alpha);
w = y*((epsilon-1)/epsilon)*(1-alpha)/l;
r = y*((epsilon-1)/epsilon)*alpha/k*(1);
i = k*(1-delta)*k*(1);
y_l = y/l;
end;

initval;
k = 9;
c = 0.7;
l = 0.3;
w = 2.0;

```

```
r = 0;  
z = 0;  
end;  
  
steady;  
  
check;  
  
shocks;  
var z;  
periods 1:9;  
values 0.1;  
end;  
  
simul(periods=2100);
```

6.10. Ejecución del archivo y resultados

6.10.1. Resultados – Modelos estocásticos

Los resultados tabulados pueden resumirse en lo siguiente:

- I. Resumen del modelo: Un recuento de los distintos tipos de variables en el modelo. En el ejemplo que hemos trabajado se obtiene:

MODEL SUMMARY	
Number of variables:	9
Number of stochastic shocks:	1
Number of state variables:	2
Number of jumpers:	2
Number of static variables:	5

2. Valores propios, deben ser mostrados, y se debe ver una confirmación de las condiciones Blanchard-Kahn si se usó el comando check en el archivo .mod.

EIGENVALUES:		
Modulus	Real	Imaginary
0.95	0.95	0
0.9521	0.9521	0
1.074	1.074	0
4.636e+015	-4.636e+015	0

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

The rank condition is verified.

3. Matriz de covarianza de choques exógenos: Debe ser cuadrada, con los valores de las varianzas y covarianzas del choque estipuladas en el archivo .mod.

MATRIX OF COVARIANCE OF EXOGENOUS SHOCKS	
Variables	e
e	0.000109

4. Política y transición de las funciones: Resolver la expectativa racional del modelo $E_t [f(y_{t+1}, y_t, y_{t-1}, u_t)] = 0$, significa que se encontró una función desconocida, $y_t = g(y_{t-1}, u_t)$ que puede ser falsa en el modelo original y cumplió las restricciones implicadas (las condiciones de primer orden). Una aproximación de primer orden de esta función puede ser escrita como $y_t = g(y_{t-1}, u_t)$ con $y_t = y_t - y$ y y comienza en el estado estacionario tomando el valor de y , y donde g_x es la derivada parcial de la función g con respecto a la variable x . En otras palabras, la función g es representación recursiva del tiempo (aproximadamente) del modelo que puede generar series de tiempo que aproximarán satisfactoriamente la

UNIVERSIDAD NACIONAL COLOMBIA
FACULTAD DE CIENCIAS ECONÓMICAS
UNIDAD DE INFORMÁTICA Y COMUNICACIONES

expectativa racional de la hipótesis contenida en el modelo original. En Dynare, la tabla “Policy and Transition function” contiene los elementos de g_y y g_u .

POLICY AND TRANSITION FUNCTIONS									
	y	i	l	y_l	w	k	z	c	r
Constant	0.892094	0.184116	0.302738	2.946.754	1.776.893	8.004.271	0	0.707977	0.033101
(correction)	0.000010	0.000018	0.000005	-0.000016	-0.000009	0.000018	0	-0.000009	0
k(-1)	0.019712	-0.024926	-0.008644	0.149257	0.090002	0.952074	0	0.044638	-0.003206
z(-1)	0.834868	0.625969	0.135264	1.441.126	0.868999	0.625969	0.950000	0.208899	0.028389
e	0.878809	0.658915	0.142383	1.516.974	0.914736	0.658915	1.000.000	0.219894	0.029884
k(-1),k(-1)	-0.001590	-0.000551	0.000371	-0.004605	-0.002777	-0.000551	0	-0.001039	0.000325
z(-1),k(-1)	0.032731	0.026865	0.003372	0.049756	0.030003	0.026865	0	0.005866	-0.002023
z(-1),z(-1)	0.314076	0.257939	-0.008572	0.476995	0.287628	0.257939	0	0.056137	0.008367
e,e	0.348006	0.285804	-0.009498	0.528526	0.318701	0.285804	0	0.062202	0.009271
k(-1),e	0.034454	0.028279	0.003550	0.052375	0.031582	0.028279	0	0.006175	-0.002129
z(-1),e	0.661212	0.543029	-0.018046	1.004.200	0.605533	0.543029	0	0.118184	0.017615

5. Momentos de variables simuladas: hasta los cuartos momentos.

MOMENTS	OF	SIMULATED	VARIABLES		
VARIABLE	MEAN	STD. DEV.	VARIANCE	SKEWNESS	KURTOSIS
y	0.895727	0.037630	0.001416	0.039778	-0.279739
c	0.710566	0.022996	0.000529	-0.105594	-0.448709
k	8.050.239	0.370600	0.137344	-0.091861	-0.497439
i	0.185161	0.018458	0.000341	0.207286	-0.174450
l	0.302762	0.003600	0.000013	0.098137	-0.241114
y_l	2.957.900	0.102593	0.010525	-0.064468	-0.396939
w	1.783.614	0.061864	0.003827	-0.064468	-0.396939
r	0.033063	0.000843	0.000001	0.098433	-0.303134
z	0.002444	0.036312	0.001319	-0.021876	-0.213959

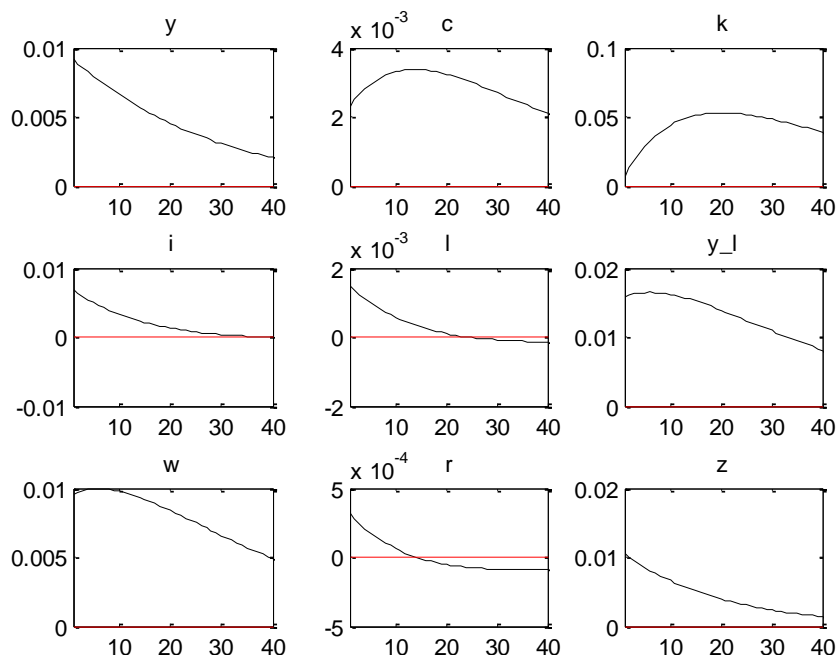
6. Correlación de variables simuladas: son las correlaciones contemporáneas presentadas en la tabla.

CORRELATION OF SIMULATED VARIABLES									
VARIABLE	y	c	k	i	l	y_l	w	r	z
y	10.000	0.9269	0.8353	0.8839	0.7052	0.9697	0.9697	0.1383	0.9910
c	0.9269	10.000	0.9805	0.6438	0.3879	0.9905	0.9905	-0.2431	0.8697
k	0.8353	0.9805	10.000	0.4813	0.1996	0.9442	0.9442	-0.4284	0.7560
i	0.8839	0.6438	0.4813	10.000	0.9544	0.7430	0.7430	0.5848	0.9369
l	0.7052	0.3879	0.1996	0.9544	10.000	0.5110	0.5110	0.7995	0.7921
y_l	0.9697	0.9905	0.9442	0.7430	0.5110	10.000	10.000	-0.1074	0.9292
w	0.9697	0.9905	0.9442	0.7430	0.5110	10.000	10.000	-0.1074	0.9292
r	0.1383	-0.2431	-0.4284	0.5848	0.7995	-0.1074	-0.1074	10.000	0.2669
z	0.9910	0.8697	0.7560	0.9369	0.7921	0.9292	0.9292	0.2669	10.000

7. Autocorrelación de variables simuladas: hasta el quinto retraso, como se especificó en las opciones de stoch_simul.

AUTOCORRELATION OF SIMULATED VARIABLES					
VARIABLE	1	2	3	4	5
y	0.9723	0.9426	0.9138	0.8850	0.8584
c	0.9969	0.9902	0.9822	0.9729	0.9628
k	10.012	0.9984	0.9939	0.9880	0.9807
i	0.9311	0.8629	0.7990	0.7377	0.6831
l	0.9145	0.8308	0.7528	0.6779	0.6121
y_l	0.9902	0.9772	0.9635	0.9489	0.9343
w	0.9902	0.9772	0.9635	0.9489	0.9343
r	0.9300	0.8606	0.7957	0.7330	0.6774
z	0.9608	0.9203	0.8817	0.8436	0.8091

Los resultados gráficos, en cambio, muestran las funciones de impulso de respuesta actuales para cada variable endógena, dando su movimiento actual. Esto puede ser especialmente usado en la visualización de la forma de la función de transición y la medida en que cada variable es afectada. Si algunas variables no retornan a su estado estacionario, se revisa el haber incluido suficientes períodos en las simulaciones, o se revisa si el modelo es estacionario: el estado estacionario actualmente existe y es estable. Si no, se debe revisar las variables y reescribir el modelo en términos de esas variables.



6.10.2. Resultados – Modelos determinísticos

Automáticamente se muestran resultados que son mucho más escasos en el caso de modelos determinísticos. Si se ingresó steady, se verá una lista de los resultados del estado estacionario. Si se ingresó check, los valores propios serán también mostrados y se debe recibir una declaración de que las condiciones del rango han sido satisfechas, si todo marchó bien. Finalmente, se verán algunas salidas intermedias: los errores de cada iteración del método de Newton usado para estimar la solución del modelo. Debe verse una disminución de errores sobre cada iteración; si no, el modelo probablemente no converge. Si esto pasa, se puede tratar de incrementar los períodos para que la transición llegue a un nuevo estado estacionario (el número de períodos de la simulación). Pero más a menudo, puede ser buena idea revisar

las ecuaciones. Claro está, aunque Dynare no muestra una gran serie de estadísticas y gráficas correspondientes a la salida de la simulación, no supone que no puedan ser creadas en Matlab.

STEADY-STATE	RESULTS:	
y	0.892084	
c	0.707986	
k	800.425	
i	0.184098	
l	0.302733	
y_l	294.677	
w	17.769	
r	0.033101	
EIGENVALUES:		
Modulus	Real	Imaginary
0.9521	0.9521	0
1.074	1.074	0
2,30E+18	-2,30E+18	0
There are 2 eigenvalues larger than 1 for 2 forward-looking variables		
The rank condition is verified		

MODEL SIMULATION :		
1 -	err = 0.71734	
	Time of iterat	:0.313
2 -	err = 0.023271	
	Time of iterat	:0.219
3 -	err = 3.8909e-005	
	Time of iterat	:0.219
4 -	err = 1.0306e-010	
	Time of iterat	:0.235

Total time of simulation :1.141

Convergency obtained.

UNIVERSIDAD NACIONAL COLOMBIA
FACULTAD DE CIENCIAS ECONÓMICAS
UNIDAD DE INFORMÁTICA Y COMUNICACIONES

7. ESTIMACIÓN DE MODELOS BÁSICOS EGDE MEDIANTE TÉCNICAS BAYESIANAS

7.1. *Introducción*

En esta sección se mostrarán la funcionalidad básica de Dynare para estimar modelos básicos DSGE, usando técnicas Bayesianas. Continuando con el trabajo realizado en las secciones anteriores, se seguirá utilizando el modelo RBC con competencia monopolística que se ha utilizado durante el manual.

Siempre que se tengan observaciones de alguna(s) de las variables endógenas, es posible utilizar Dynare para estimar todos o algunos de los parámetros. Aunque como se mencionó en esta sección se explicará el procedimiento con métodos Bayesianos, esto también es posible mediante las técnicas de máxima verosimilitud.

Acerca del ejemplo:

Continuando entonces con el modelo RBC con competencia monopolística, suponemos ahora que se tienen datos sobre las variaciones del ciclo económico de la producción (Y), y que el modelo que se ha venido trabajando explica la realidad. Es posible entonces utilizar los métodos Bayesianos para estimar los parámetros del modelo. Recordando, estos son: α , la proporción del capital en la producción; β , el factor de descuento; δ , la tasa de descuento; ψ , el peso o influencia del ocio en la función de utilidad; ρ , el grado de persistencia de la productividad; y ϵ , el parámetro que condiciona el precio de salida de la competencia monopolística.

Es importante tener en cuenta que la condición para la estimación Bayesiana es que hayan tantos choques como variables observables (es una condición menos estricta que la estimación para máxima verosimilitud). Puede darse el caso en que no se puedan identificar todos los parámetros, pero la estimación Bayesiana seguirá funcionando con éxito.

7.2. *Declaración de variables y parámetros*

Para ingresar el modelo en Dynare para la estimación, similar al método anterior, primero deben declararse las variables del modelo en el preámbulo del archivo .mod. Se hace de la misma forma en que se especificó en las secciones anteriores:

```

var y c k l y l w r z;
varexo e;
parameters beta psi delta alpha rho epsilon;

```

7.3. *Declaración del modelo*

Suponga que la ecuación del cambio de la tecnología es estacionario AR(1) con un parámetro autorregresivo, ρ , menor que uno. Las variables del modelo serían por lo tanto estacionarias y se podrá continuar sin complicaciones. El escenario alternativo con variables no estacionarias es más complicado y por lo tanto no se explicará en esta sección. Entonces, en el caso estacionario, el block del modelo es exactamente igual de cómo se explicó en las secciones anteriores:

```

model;
(l/c) = beta*(l/c(+1))*(1+r(+1)-delta);
psi*c/(1-l) = w;
c+i = y;
y = (k(-1)^alpha)*(exp(z)*l)^(1-alpha);
w = y*((epsilon-1)/epsilon)*(1-alpha)/l;
r = y*((epsilon-1)/epsilon)*alpha/k(-1);
i = k-(1-delta)*k(-1);
y l = y/l;
z = rho*z(-1)+e;
end;

```

7.4. *Declaración de variables observables*

Dynare es capaz de conocer cuáles variables son observables para el procedimiento de la estimación. Para esto vamos a utilizar el comando `varobs`, seguida del nombre de la(s) variable(s) observable(s).

Esas variables deben estar disponibles en el archivo de datos `data file`, lo cual se explicará posteriormente en esta misma sección. Por ahora según el ejemplo, debe escribirse

```

varobs Y;

```

Pues como se mencionó, el supuesto es que se tienen los valores del ciclo de la producción, entonces la única variable observable es la variable `Y`.

7.5. Especificación del estado estacionario

Antes de que Dynare estime el modelo, lo linealiza alrededor del estado estacionario. Por lo tanto debe existir un estado estacionario, y aunque Dynare puede calcularlo, deben proporcionársele los valores aproximados del mismo, de la misma forma en que se explicó en las secciones anteriores y de acuerdo a la misma sintaxis utilizada, mediante los comandos `initval`, `steady` y `check`. Por lo tanto el block `steady state` es exactamente igual al utilizado anteriormente:

```
initval;
k = 9;
c = 0.7;
l = 0.3;
w = 2.0;
r = 0;
z = 0;
end;

steady;

check;
```

Durante la estimación, en búsqueda de la moda posterior, Dynare recalcula el estado estacionario del modelo como cada iteración de la rutina de optimización, basado en la última ronda de parámetros disponibles. De esta manera, al proveer valores iniciales aproximados y basándose en el comando de Dynare para encontrar el estado estacionario podrá reducirse el tiempo para encontrar la moda posterior. El programa utilizará un 60% del tiempo recalculando estados estacionarios. Es mucho más eficiente escribir un archivo de estado estacionario externo para Matlab y dejar a Dynare utilizar ese archivo para encontrar el estado estacionario del modelo mediante un procedimiento algebraico.

7.6. Declaración de los valores *a priori*

Los valores *a priori* juegan un rol importante en la estimación Bayesiana, y consecuentemente juegan un papel central en la especificación del archivo `.mod`. Dichos valores, en la estimación Bayesiana, son declarados como una distribución. La sintaxis general para introducir los valores *a priori* en Dynare es la siguiente

estimated params;

PARAMETER NAME, PRIOR SHAPE, PRIOR MEAN, PRIOR STANDARD ERROR
[, PRIOR 3rd PARAMETER] [,PRIOR 4th PARAMETER] ;

end;

donde la siguiente tabla define cada término más claramente:

DISTRIBUCIÓN DE LOS VALORES	Distribución correspondiente	Rango
NORMAL_PDF	$N(\varphi, \sigma)$	\mathbb{R}
GAMMA_PDF	$G2(\varphi, \sigma, p_3)$	$[p_3, +\infty)$
BETA_PDF	$B(\varphi, \sigma, p_3, p_4)$	$[p_3, p_4]$
INV_GAMMA_PDF	$IG1(\varphi, \sigma)$	\mathbb{R}^+
UNIFORM_PDF	$U(p_3, p_4)$	$[p_3, p_4]$

donde φ es el PRIOR MEAN (la media de los valores a priori), σ el PRIOR STANDARD ERROR (el error estándar de los valores a priori), p_3 PRIOR 3rd PARAMETER (que por defecto toma valor 0), y p_4 el PRIOR 4th PARAMETER (cuyo valor por defecto es 1).

Debe tenerse en cuenta que cuando se especifica una distribución uniforme entre 0 y 1 como un valor a priori de un parámetro, primero deben dejarse dos espacios vacíos, los que corresponden a φ y a σ . Por ejemplo, si es para el parámetro α se escribe:

alpha, uniform pdf, , , 0, 1;

Si algunos parámetros en el modelo están calibrados y no deben ser estimados, se hace mediante el comando parameters como se explicó en las secciones anteriores.

Escoger los valores a priori para los parámetros es una labor difícil pero muy importante. Por esto vale la pena dedicar tiempo en elegirlos y en poner a prueba la solidez de sus resultados. Algunas consideraciones pueden ser de ayuda. Primero, piense acerca el dominio de los valores a priori para cada parámetro;

esto es si deben encontrarse en un conjunto acotado, o si puede ser abierto hacia alguno de sus lados, entre otras cosas. Recuerde también que si se especifica una probabilidad de cero sobre un dominio determinado en los valores a priori, necesariamente se encontrará una probabilidad de cero en las distribuciones posteriores. Luego, piense sobre la forma de la distribución de los valores a priori. Debe ser simétrica? Sesgada? Si es así, hacia qué lado? Puede luego construir una distribución para cada parámetro en la mente. Pregúntese cuál es la probabilidad de que el parámetro sea mayor que cierto valor, y repita el ejercicio disminuyendo dicho valor gradualmente. Puede luego tomar la distribución estándar que mejor encaje en la distribución que percibió.

En ocasiones puede ser deseable estimar transformaciones de los parámetros más que el parámetro en sí. En tal caso, es posible declarar el parámetro a ser estimado en la declaración de parámetros y definir la transformación en la parte superior de la sección “modelo”, como una expresión de Matlab, adicionando el signo “numeral” (#) al principio de la línea correspondiente. Por ejemplo, puede que le resulte más fácil definir un valor a priori sobre el factor de descuento β , que sobre su inversa, que a menudo aparece en las ecuaciones de Euler. Por lo tanto deberá escribirse:

```
model;
# sig = 1/bet;
c = sig*c(+1)*mpk;
end;
estimated params;
bet,normal_pdf,1,0.05;
end;
```

Finalmente, otro comando a usar es el comando `estimated params init` que declara los valores iniciales numéricos para la optimización cuando son diferentes de la media de los valores a priori. Este comando se utiliza especialmente cuando se vuelve a hacer la estimación (por ejemplo, si se bloquea o se frena el proceso en el momento en que está haciendo la estimación) y cuando se quiere conocer los valores posteriores de la moda de los valores a priori estimándolos con los valores iniciales de los parámetros, en lugar de con los valores inmediatamente anteriores.

Retomando el ejemplo básico que se viene manejando, debe escribirse:

```
estimated params;
alpha, beta_pdf, 0.35, 0.02;
beta, beta_pdf, 0.99, 0.002;
delta, beta_pdf, 0.025, 0.003;
psi, gamma_pdf, 1.75, 0.02;
rho, beta_pdf, 0.95, 0.05;
```

```
epsilon, gamma_pdf, 10, 0.003;  
stderr e, inv_gamma_pdf, 0.01, inf;  
end;
```

7.7. *Presentar la estimación*

Para pedirle a Dynare que estime un modelo, lo único que es necesario es adicionar el comando `estimation` al final del archivo `.mod`. Lo que es realmente complejo son las opciones del comando (que deben figurar en paréntesis, seguidas por comas y a continuación del comando). A continuación se mostrará una lista de las opciones más usuales.

1. `datafile = FILENAME` : el archivo de datos (un archivo `.m` , `.mat` , o `.xls`). Tenga en cuenta que las observaciones no necesitan aparecer en ningún orden, pero los vectores de las observaciones deben ser nombradas con los mismos nombres que tienen en el comando `var_obs`. En los archivos de Excel, por otro lado, las observaciones pueden ser ordenadas en columnas, y los nombres de las variables pueden aparecer en la primer celda de cada columna.
2. `nobs = INTEGER` : Número de observaciones a ser usadas (por defecto serán todas las observaciones del archivo)
3. `first_obs = INTEGER` : El número de la primera observación a ser usada (por defecto es = 1). Esta opción es usada cuando se ejecutan bucles, o para dividir las observaciones en sub-períodos.
4. `prefilter = 1` : El procedimiento de estimación baja los datos (por defecto = 0, es decir, no hace un prefiltro). Esta opción se utiliza si las variables del modelo están en desviación del estado estacionario, por ejemplo, y por lo tanto tienen media cero. Bajando las observaciones se impondría entonces una media de cero también en las variables observadas.
5. `nograph` : no presenta los gráficos
6. `conf_sig = {INTEGER — DOUBLE}` : nivel de intervalo de confianza de los resultados (por defecto es = 0.90)

7. `mh_replic` = INTEGER : número de replicaciones del algoritmo Metropolis Hasting. Por el momento, debe ser mayor de 1200 (por defecto es = 2000)
8. `mh_nblocks` = INTEGER : Número de cadenas paralelas del algoritmo Metropolis Hasting (por defecto = 2) . Lo más aconsejable es trabajar con un valor de 5 o mayor. Esto mejora el cálculo de la varianza y de la media de los parámetros, uno de los criterios principales para evaluar la eficiencia del Metropolis-Hasting para evaluar la distribución posterior.
9. `mh_drop` = DOUBLE : la fracción del vector de parámetros inicialmente generado a ser eliminado antes de usar las simulaciones posteriores (por defecto = 0.5; esto significa que la primera mitad de los sorteos del Metropolis-Hasting es descartado).
10. `mh_jscale` = DOUBLE : escala a ser usada para la distribución del salto en el algoritmo MH. El valor por defecto es extrañamente satisfactorio. Esta opción debe ser utilizada para obtener una tasa ideal de aceptación del 25% en el algoritmo Metropolis-Hasting (valor por defecto = 0.2). La idea es no rechazar o aceptar con frecuencia un parámetro candidato; la literatura ha establecido un valor entre 0.2 y 0.4. Si la tasa de aceptación es muy alta, las iteraciones Metropolis-Hastings nunca estarán siquiera en las colas de la distribución, mientras si es muy baja, las iteraciones se quedarán atascadas en un subespacio del rango de los parámetros. Tenga en cuenta que la tasa de aceptación disminuye si incrementa la escala usada en la distribución del salto y viceversa.
11. `mh_init_scale`=DOUBLE : Escala a ser usada para la elaboración del valor inicial de la cadena de Metropolis-Hastings (por defecto = $2 * mh_jscale$). La idea aquí es sacar los valores iniciales de un tendido de distribución con el fin de maximizar las posibilidades de que estos valores no estén demasiado juntos, lo cual no permitiría ejecutar varios bloques de cadenas MH.
12. `mode_file`=FILENAME : nombre del archivo que contiene los valores previos a la moda. Cuando se calcula la moda, Dynare guarda la moda (`xparaml`) y la hesiana (`hh`) en un archivo que llama `NOMBRE DEL MODELO_mode`. Esta opción es especialmente para el proceso de estimación si ya se han realizado las estimaciones iniciales y se tienen los valores posteriores de la moda.
13. `mode_compute`=INTEGER : Especifica el optimizador para el cálculo de la moda.

- 0 : La moda no es calculada. Debe ser especificado el archivo de la moda `mode_file`.
- 1: usa la opción `fmincon` de Matlab
- 2: usa la simulación Lester Ingber's Adaptive Simulated Annealing
- 3: usa la opción `fmincon` de Matlab
14. `mode_check` : Cuando se utiliza esta opción, Dynare muestra el negativo de las densidades posteriores de los valores alrededor del calculo de la moda para cada parámetro estimado. Esta opción es útil para diagnosticar problemas con el optimizador. Una clara indicación de un problema sería que la moda no esté en la sección mínima del negativo de la distribución posterior.
15. `load_mh_file` : con esta opción Dynare añade las simulaciones anteriores MH en vez de empezar desde cero. De nuevo, esta opción es útil para acelerar el proceso de estimación.
16. `nodiagnostic` : No calcula el diagnostico de convergencia de la simulación Metropolis-Hastings (por defecto los diagnósticos son calculados y mostrados). En realidad lo más recomendable es ver si los distintos bloques de valores de MH convergen, para evaluar la confianza del modelo.
17. `bayesian_irf` : desencadena el cálculo de la distribución posterior de la función impulso-respuesta (FIR). La longitud de la FIR es controlada por la opción `irf` (opción del comando `stoch_simul`). Para construir la distribución posterior de la FIR, Dynare golpea los parámetros y choca los valores de la correspondiente distribución estimada y, para cada serie de funciones genera una FIR.
18. Todas las opciones disponibles para `stoch_simul` pueden ser simplemente agregadas a las opciones citadas anteriormente, separadas por comas.
19. `moments_verendo`: desencadena el cálculo de la distribución posterior de los momentos teóricos de las variables endógenas como en `stoch_simul` (la distribución posterior de la descomposición de varianza está incluida también).
20. `filtered_vars`: desencadena el cálculo de la distribución posterior de variables endógenas y choques filtrados.

21. smoother: desencadena el cálculo de la distribución posterior de variables endógenas y choques suavizados. Los choques suavizados son una reconstrucción de los valores de choques desapercibidos sobre la muestra, usando toda la información contenida en las observaciones de la muestra. Choques filtrados, en cambio, son construcciones basadas solamente en conocimientos de información pasada. Para calcular los errores de predicción de un período adelante, por ejemplo, se deben usar variables filtradas, no suavizadas.
22. forecast = INTEGER: calcula la distribución posterior de un pronóstico en INTEGER períodos después de terminar la muestra usada en la estimación. La gráfica correspondiente incluye un intervalo de confianza describiendo incertidumbre debido a parámetros y un intervalo de confianza describiendo incertidumbre debido a parámetros y futuros choques. Hay que notar que Dynare no predice afuera del modo posterior. Se necesita ejecutar iteraciones Metropolis – Hastings antes de poder ejecutar pronósticos sobre un modelo estimado.

Finalmente, ejecutar un pronóstico es muy similar a una función de impulso de respuesta, como en `bayesian_irf`, excepto que el pronóstico no comienza en un estado estacionario, simplemente en el punto correspondiente a la última serie de observaciones. El objetivo de emprender un pronóstico es observar cómo el sistema retorna al estado estacionario desde su punto de inicio. Claro está que como la observación no existe para todas las variables, éstas necesariamente son reconstruidas por muestreos fuera de la distribución posterior de parámetros. Nuevamente, repitiendo este paso, frecuentemente basta para una distribución posterior de pronóstico.

Finalmente, volviendo al ejemplo, se escogería una opción estándar:

```
estimation(datafile=simuldataRBC,nobs=200,first obs=500,  
mh replic=2000,mh nblocks=2,mh drop=0.45,mh jscale=0.8);
```

Esto finaliza la descripción del archivo .mod.

7.8. El archivo .mod completo

A continuación se encuentra el archivo .mod completo para la estimación de un modelo muy básico.

```

var y c k i l y_l w r z;56

varexo e;

parameters beta psi delta alpha rho epsilon;

model;
(l/c) = beta*(l/c(+l))*(l+r(+l)-delta);
psi*c/(l-l) = w;
c+i = y;
y = (k(-l)^alpha)*(exp(z)*l)^(1-alpha);
w = y*((epsilon-l)/epsilon)*(1-alpha)/l;
r = y*((epsilon-l)/epsilon)*alpha/k(-l);
i = k-(l-delta)*k(-l);
y_l = y/l;
z = rho*z(-l)+e;
end;

varobs Y;

initval;
k = 9;
c = 0.7;
l = 0.3;
w = 2.0;
r = 0;
z = 0;
e = 0;
end;

steady;

check;

estimated params;
alpha, beta pdf, 0.35, 0.02;
beta, beta pdf, 0.99, 0.002;
delta, beta pdf, 0.025, 0.003;
psi, gamma pdf, 1.75, 0.02;

```

UNIVERSIDAD NACIONAL COLOMBIA
FACULTAD DE CIENCIAS ECONÓMICAS
UNIDAD DE INFORMÁTICA Y COMUNICACIONES

```
rho, beta pdf, 0.95, 0.05;  
epsilon, gamma pdf, 10, 0.003;  
stderr e, inv gamma pdf, 0.01, inf;  
end;  
  
estimation(datafile=simuldataRBC,nobs=200,first obs=500,  
mh replic=2000,mh nblocks=2,mh drop=0.45,mh jscale=0.8);
```

7.9. Interpretación de los resultados

Como en el caso de solución y simulación del modelo, Dynare retorna salidas en tabulación y gráficas. Teniendo las bases de las opciones entradas en el ejemplo del archivo .mod mostrado anteriormente, Dynare arrojará los siguientes resultados.

7.9.1. Resultados tabulados

Los primeros resultados en ser mostrados (y calculados desde el punto de vista cronológico) son los estados estacionarios. De observan los valores nulos de I para las variables no estacionarias Y_{obs} y P_{obs} . Estos resultados están seguidos por valores propios del sistema presentados en el orden en el cual las variables endógenas son declaradas al principio del archivo .mod. La tabla de los valores propios es completada con una declaración acerca de que se está cumpliendo la condición de Blanchard-Kahn.

La siguiente serie de resultados son para que las iteraciones numéricas necesariamente encuentren su modo posterior. La mejora desde una iteración hacia la siguiente llega a cero, Dynare da el valor de la función objetivo (el posterior Kernel) del modo y muestra dos tablas importantes resumiendo resultados de la maximización posterior.

La primera tabla resume los resultados para los valores de los parámetros. Ésta incluye: el medio anterior, el modo posterior, la desviación estándar y la t-estadística del modo (basado en el supuesto de una Normal, probablemente errónea cuando hay una estimación Bayesiana, ya que es opuesta a la máxima probabilidad estándar), tan bueno como la anterior distribución y la desviación estándar. Ésta va seguida por una segunda tabla que resume algunos resultados de los choques. Puede ser posible que se tenga un valor infinito para la desviación estándar, éste es simplemente, el caso límite de la distribución gamma inversa.

7.9.2. Resultados gráficos

La primera figura que surge después de un pequeño cálculo realizado por Dynare es necesario generarlo. La figura devuelve una representación gráfica de las prioridades para cada parámetro de interés.

La segunda serie de figuras muestra “Diagnósticos univariados MCMC”, donde MCMC son siglas en Inglés de Cadenas de Markov Monte Carlo. Éste es el principal origen de la realimentación para ganar confianza, o notar un problema, con resultados. Hay que recordar que Dynare completa varias corridas de simulaciones Metropolis-Hastings (tantas como se determine en la opción `mh_nblocks`, empezando cada vez desde una valor diferente). Si los resultados de una cadena son acertados, y el optimizador no se atascó en un área extraña del parámetro subespacio, dos cosas pueden suceder. Primero, los resultados dentro de cualquiera de las tantas iteraciones de la simulación Metropolis-Hastings, deben ser similares. Y segundo, los resultados entre las distintas cadenas deben ser cercanos. Esta es la idea de lo que diagnostican las MCMC.

Más específicamente, las líneas rojas y azules sobre las cartas representan específicamente medidas de los parámetros vectores entre las cadenas. Para que los resultados sean acertados, éstas deben ser relativamente constantes (aunque éstas siempre tendrán alguna variación) y deben converger. Dynare reporta tres medidas: “interval”, siendo construido desde un 80% de confianza el intervalo alrededor del parámetro medio, “m2”, siendo una medida de varianza y “m3” basado en terceros momentos. En cada caso, Dynare informa las medidas entre las cadenas. La figura titulada “multivariate diagnostic” presenta resultados de la misma naturaleza, excepto que éstos reflejan unas medidas adicionales basadas en los valores propios de la matriz de varianza-covarianza de cada parámetro.

En el ejemplo realizado anteriormente, se puede decir que en efecto se obtuvo convergencia y relativa estabilidad en todas las medidas de los parámetros momento. Hay que notar que el eje horizontal representa el número de iteraciones Metropolis-Hastings que han sido realizadas y el eje vertical las medidas de los parámetros momento, con el primero, correspondiente a la medida del valor inicial de las iteraciones Metropolis-Hastings.

Si los momentos trazados son sumamente inestables o no convergen, se tiene un problema de escasas prioridades. Para esto es aconsejable rehacer la estimación con diferentes prioridades. Si se tienen problemas ahora con la nueva prioridad, comenzar intentando con una prioridad uniforme y relativamente amplia y ver a dónde la conduce la información de la siguiente

distribución. Otro enfoque es realizar un gran número de simulaciones Metropolis-Hastings.

La primera parte de la última figura –para este ejemplo la figura 6- muestra la serie de resultados más interesante, pues son dirigidos hacia la mayoría de los cálculos realizados por Dynare: la distribución posterior. De hecho, la figura compara la posterior con la primera distribución (línea negra vs. gris). Adicionalmente, sobre la posterior distribución, Dynare traza una línea verde la cual representa el modo posterior. Esto permite hacer declaraciones acerca de la otra información que la simplemente concerniente al promedio y la varianza de los parámetros, se puede también discutir la probabilidad de que el parámetro sea más grande o más pequeño que un valor exacto.

Éstas gráficas son de proceso especialmente relevante y presentan resultados clave, pero pueden servir también como herramientas para detectar problemas o estructurar confianza adicional en los resultados. Primero, la primera y la posterior distribución pueden no ser excesivamente diferentes. Segundo, La posterior distribución debe acercarse a normal, o por lo menos no mostrar un aspecto que sea claramente no normal. Tercero, el modo verde (calculado desde la optimización numérica del núcleo posterior) no debe ser lejano al modo de la distribución posterior. Si no, es aconsejable realizar un gran número de simulaciones Metropolis-Hastings.

La última figura devuelve la estimación suavizada de los choques en una útil ilustración que muestra la plausibilidad del tamaño y frecuencia de los choques. El eje horizontal, en este caso representa el número de períodos del ejemplo. Una cosa para revisar es el hecho de que los choques deben centrarse alrededor de cero. Esto es evidente en el caso del ejemplo.

8. CONCLUSIONES

- Es conveniente la adquisición de Dynare puesto que es un software libre que se puede descargar directamente de la página principal (<http://www.cepremap.cnrs.fr/dynare/>) y cuya instalación es muy sencilla de realizar para el sistema operativo Windows.
- Dynare es una herramienta para el análisis económico que se especializa en la estimación de modelos de Equilibrio General Dinámico Estocástico. Puede encontrarse gran cantidad de información específica, el único inconveniente es que la gran mayoría de la información está en inglés.
- Al ser el software tan específico podría pensarse en su utilización y capacitación para los estudiantes de la Facultad de Ciencias Económicas que estén interesados en el tema que abarca Dynare, puesto que podría ser de gran ayuda en un futuro.
- El análisis que realiza Dynare es muy completo (dentro de su campo) y podría ser útil para quienes estudian este tipo de modelos.

9. BIBLIOGRAFIA

“Dynare Manual version 4.0.3”.

<http://www.cepremap.cnrs.fr/dynare/>

Mancini, Tommaso. *“Dynare. User Guide. An Introduction to the solution & estimation of DSGE models”*. 2007-2008.