Lecciones de Equilibrio General Computable con MatLab

Gonzalo Fernández de Córdoba Martos

Esta versión: 28 de julio de 2002

1 Lección sexta

En esta lección vamos a introducir una función de inversión por primera vez. Hasta este momento siempre hemos considerado que $k_{t+1} \leq i_t + (1-\delta)k_t$ y que la inversión era simplemente la parte no consumida del producto. Imaginemos ahora que las cosas no son tan sencillas y que es necesaria una ulterior manipulación de la inversión para que ésta pueda pasar a formar parte del stock de capital. Vamos a imaginar además que en esa transformación intervienen tanto los bienes transables como los no transables del capitulo anterior. Si denotamos por x_{Tt} y x_{Nt} a la inversión realizada en ambos bienes entonces podríamos escribir $k_{t+1} \leq I(x_{Tt}, x_{Nt}) + (1-\delta)k_t$. Si asumimos que la tecnología que transforma la inversión en nuevo capital es del tipo Coob-Douglas, nuestra nueva ecuación tendría el siguiente aspecto: $k_{t+1} \leq G x_{Tt}^{\gamma} x_{Nt}^{1-\gamma} + (1-\delta)k_t$.

Con esta especificación podemos estar seguros de que el proceso debe frenarse, y esto es así porque es necesario que concurra la participación de bienes no transables en la formación de nuevo capital, pero como sólo los bienes transables pueden ser importados sin límite, habrá que esperar a que la industria de construcción y servicios (no transables) tenga esas cantidades disponibles para invertir.

1.1 Las ecuaciones

El modelo que estamos planteando ahora sería:

$$\max \sum_{t=0}^{\infty} \beta^t \frac{1}{\rho} \left(\varepsilon c_{Tt}^{\rho} + (1-\varepsilon) c_{Nt}^{\rho}\right)^{\frac{\phi}{\rho}}$$

$$\begin{aligned} s.a \quad c_{Tt} + x_{Tt} + b_{t+1} &\leq A_T k_{Tt}^{\alpha_T} + (1+r) b_t \\ c_{Nt} + x_{Nt} &\leq A_N k_{Nt}^{\alpha_N} \\ k_{Tt+1} + k_{Nt+1} - (1-\delta)(k_{Tt} + k_{Nt}) &\leq G x_{Tt}^{\gamma} x_{Nt}^{1-\gamma} \\ & \quad k_0, b_0 \text{ dados} \end{aligned}$$

A la primera restricción le vamos a asociar el multiplicador de Langrange P_{Tt} que sabemos que será igual a 1. A la segunda restricción le asociaremos p_{Nt} y a la tercera le asociaremos el multiplicador q_{t+1}. Nótese que ahora el capital va a tener su propio precio, ahora va a ser un bien producido en la economía que será distinto de los otros bienes y que por tanto tendrá su propio precio.

Planteamos la ecuación auxiliar de Lagrange como:

$$L = \sum_{t=0}^{\infty} \beta^t \left[\frac{1}{\rho} \left(\varepsilon c_{Tt}^{\rho} + (1-\varepsilon) c_{Nt}^{\rho} \right)^{\frac{\phi}{\rho}} + ... \right.$$

$$... + p_{Tt} \left(A_T k_{Tt}^{\alpha_T} + (1+r) b_t - c_{Tt} - x_{Tt} - b_{t+1} \right) + ...$$

$$\left. ... + p_{Nt} \left(A_N k_{Nt}^{\alpha_N} - c_{Nt} - x_{Nt} \right) + q_{t+1} \left(G x_{Tt}^{\gamma} x_{Nt}^{1-\gamma} - k_{Tt+1} - k_{Nt+1} + (1-\delta)(k_{Tt} + k_{Nt}) \right) \right]$$

De donde extraemos las condiciones de primer orden y que formaran parte, junto con las restricciones de factibilidad, del sistema de ecuaciones que tenemos que resolver. Estas condiciones de primer orden son:

$$\begin{aligned}
u_{Tt} \varepsilon \rho c_{Tt}^{\rho-1} - p_{Tt} &= 0 \\
u_{Nt} (1-\varepsilon) \rho c_{Nt}^{\rho-1} - p_{Nt} &= 0 \\
-p_{Tt-1} + \beta(1+r) p_{Tt} &= 0 \\
-p_{Tt} - q_{t+1} G \gamma x_{Tt}^{\gamma-1} x_{Nt}^{1-\gamma} &= 0 \\
-p_{Nt} - q_{t+1} G (1-\gamma) x_{Tt}^{\gamma} x_{Nt}^{-\gamma} &= 0 \\
p_{Tt} A_T \alpha_T k_{Tt}^{\alpha_T - 1} + q_{t+1}(1-\delta) - q_t(1+r) &= 0 \\
p_{Nt} A_N \alpha_N k_{Nt}^{\alpha_N - 1} + q_{t+1}(1-\delta) - q_t(1+r) &= 0
\end{aligned}$$

y dos condiciones de transversalidad. Las ecuaciones de vaciado y de factibilidad que completan el sistema son:

$$\begin{aligned}
c_{Tt} + x_{Tt} + b_{t+1} &\leq A_T k_{Tt}^{\alpha_T} + (1+r) b_t \\
c_{Nt} + x_{Nt} &\leq A_N k_{Nt}^{\alpha_N} \\
k_{Tt+1} + k_{Nt+1} - (1-\delta)(k_{Tt} + k_{Nt}) &\leq G x_{Tt}^{\gamma} x_{Nt}^{1-\gamma} \\
& \quad k_0, b_0 \text{ dados}
\end{aligned}$$

1.2 El programa

El programa tendrá tres partes La primera parte contendrá la definición de parámetros del modelo, los parámetros del programa y la semilla inicial que consistirá en toda una trayectoria para cada variable. La semilla inicial será el estado estacionario, para lo cual necesitaremos un pequeño programa que nos compute el estado estacionario y que sirve de introducción al tercer programa en el que estarán todas las condiciones de primer orden y las

condiciones de factibilidad del problema. El primer programa efectuará una llamada a nuestro programa estándar **secant.m** que se encargará de iterar sobre nuestro segundo programa. Una vez que el estado estacionario esté computado volveremos a llamar a **secant.m** para que itere sobre el programa que contiene la dinámica. La lógica es por tanto la misma de siempre. Dado que este es el primer programa con cierta complejidad que vamos a realizar comentaré detalladamente qué hace cada línea de comando que tengamos en uno y otro programa. Veamos el programa principal **inver.m**

```
%programa inver.m
%Este programa resuelve el modelo con dos bienes e inversión que
%requiere la participación de los bienes transables y no transables
% inf
%max sum beta^t (c(t)^(1-eta)-1)/(1-eta)
% t=0
% s.a. b(t+1)+cT(t)+xT(t)<=aT*kT^alphaT+(1+r)*b(t)
% cN(t)+xN(t)<=aN*kN(t)^alphaN
% iT(t)+iN(t)<=GxT(t)^gamma*xN(t)^(1-gamma)
% kT(t+1)<=iT(t)+(1-delta)*kT(t)
% kN(t+1)<=iN(t)+(1-delta)*kN(t)
% k(0) dado, b(0) dado
clear
%Definición de los parámetros del modelo
aT = 0.96612775*34.3541;
aN = 0.90461754*35.4459;
alphaT = 0.3071665;
alphaN = 0.35370848;
gamma = 0.39681632;
G = 1.95756169;
rho = -1;
theta = -1;
epsilon = (0.852600116^(1/(1-rho)))/(1+0.852600116^(1/(1-rho)));
delta = 0.0576;
beta = 0.9463;
r = 1/beta-1;
bss = 0;
%Definición de parámetros del programa
maxit = 1000;
```

```
crit = 1e-8;
T = 40;
%Llamada a secant.m para computar el estado estacionario
x0 = [896; 981];
param = [aT aN alphaT alphaN gamma G rho theta epsilon delta
beta bss];
sol = secant('sscpo', x0, param, crit, maxit);
%Resultados del estado estacionario
kTss = sol(1);
kNss = sol(2);
rss = 1/beta-1;
kss = kTss+kNss;
mpkTss = aT*alphaT*kTss^(alphaT-1);
mpkNss = aN*alphaN*kNss^(alphaN-1);
pNss = mpkTss/mpkNss;
ratxss = gamma*pNss/(1-gamma);
xNss = delta*kss/(G*ratxss^gamma);
xTss = ratxss*xNss;
qss = ratxss^(1-gamma)/(G*gamma);
yTss = aT*kTss^alphaT;
yNss = aN*kNss^alphaN;
cNss = yNss-xNss;
cTss = ((1-epsilon)/(epsilon*pNss))^(1/(rho-1))*cNss;
%Estado estacionario computado
%Construcción de la semilla
kTguess = kTss*ones(size(1:T-1))';
kNguess = kNss*ones(size(1:T-1))';
bguess = bss*ones(size(1:T-1))';
Mguess = [kTguess kNguess bguess];
guess = [kTss kNss bss];
for t=2:T-1
c = Mguess(t,:);
guess = [guess c];
end
guess(3*t+1) = kTss;
guess(3*t+2) = kNss;
%LLamada a secant.m para computar la dinámica
kT0 = kTss*0.8;
```

```
kN0 = kNss*0.8;
param = [aT aN alphaT alphaN gamma G rho theta epsilon delta
beta bss T kT0 kN0];
sol = secant('dincpo', guess', param, crit, maxit);
    for t=1:T
    kT(t) = sol(3*t-2);
    kN(t) = sol(3*t-1);
    end
    for t=1:T-1
    b(t) = sol(3*t)
    end
%Reconstrucción de las trayectorias de equilibrio
mpkT(1) = aT*alphaT*kT(1)^(alphaT-1);
mpkN(1) = aN*alphaN*kN(1)^(alphaN-1);
pN(1) = mpkT(1)/mpkN(1);
ratx(1) = gamma*pN(1)/(1-gamma);
xN(1) = (kT(1)+kN(1)-(1-delta)*(kT0+kN0))/(G*ratx(1)^gamma);
xT(1) = ratx(1)*xN(1);
q(1) = ratx(1)^(1-gamma)/(G*gamma);
yT(1) = aT*kT(1)^alphaT;
yN(1) = aN*kN(1)^alphaN;
cN(1) = yN(1)-xN(1);
cT(1) = ((1-epsilon)/(epsilon*pN(1)))^(1/(rho-1))*cN(1);
u(1) = theta*(epsilon*cT(1)^rho+(1-epsilon)*cN(1)^rho)^(theta/rho-
1);
for t=2:T
mpkT(t) = aT*alphaT*kT(t)^(alphaT-1);
mpkN(t) = aN*alphaN*kN(t)^(alphaN-1);
pN(t) = mpkT(t)/mpkN(t);
ratx(t) = gamma*pN(t)/(1-gamma);
xN(t) = (kT(t)+kN(t)-(1-delta)*(kT(t-1)+kN(t-1)))/(G*ratx(t)^gamma);
xT(t) = ratx(t)*xN(t);
q(t) = ratx(t)^(1-gamma)/(G*gamma);
yT(t) = aT*kT(t)^alphaT;
yN(t) = aN*kN(t)^alphaN;
cN(t) = yN(t)-xN(t);
cT(t) = ((1-epsilon)/(epsilon*pN(t)))^(1/(rho-1))*cN(t);
```

```
u(t) = theta*(epsilon*cT(t)^rho+(1-epsilon)*cN(t)^rho)^(theta/rho-
1);
end
yT0 = aT*kT0^alphaT;
yT = [yT0 aT*kT(1:T).^alphaT];
yN0 = aN*kN0^alphaN;
yN = [yN0 aN*kN(1:T).^alphaN];
b0 = 0;
bT = [b0 b];
ca = [0 bT(2:T)-bT(1:T-1)];
td0 = 0;
td1 = aT*kT(1:T-1).^alphaT-cT(1:T-1)-xT(1:T-1);
td = [td0 td1];
figure(1)
plot(yT)
title('Producto')
figure(2)
plot(kT)
title('Capital')
figure(3)
plot(ca)
title('Cuenta corriente')
figure(4)
plot(td)
title('Déficit comercial')
```

El segundo programa llamado **sscpo.m** computa el estado estacionario del sistema.

```
function f=sscpo(z, p)
%sscpo.m Función que contiene las condiciones de primer orden
del estado estacionario
%Asignación de variables
kTss = z(1);
kNss = z(2);
%Asignación de parámetros
aT = p(1);
aN = p(2);
```

```
alphaT = p(3);
alphaN = p(4);
gamma = p(5);
G = p(6);
rho = p(7);
theta = p(8);
epsilon = p(9);
delta = p(10);
beta = p(11);
bss = p(12);
rss = 1/beta-1;
kss = kTss+kNss;
mpkTss = aT*alphaT*kTss^(alphaT-1);
mpkNss = aN*alphaN*kNss^(alphaN-1);
pNss = mpkTss/mpkNss;
ratxss = gamma*pNss/(1-gamma);
xNss = delta*kss/(G*ratxss^gamma);
xTss = ratxss*xNss;
qss = ratxss^(1-gamma)/(G*gamma);
yTss = aT*kTss^alphaT;
yNss = aN*kNss^alphaN;
cNss = yNss-xNss;
cTss = ((1-epsilon)/(epsilon*pNss))^(1/(rho-1))*cNss;
f(1) = aT*alphaT*kTss^(alphaT-1)-qss*(rss+delta);
f(2) = yTss-cTss-xTss+rss*bss;
f=f';
```

Y finalmente el programa **dincpo.m** computa la dinámica.

```
function f=dincpo(z, p)
%Asignación de parámetros
aT = p(1);
aN = p(2);
alphaT = p(3);
alphaN = p(4);
gamma = p(5);
G = p(6);
rho = p(7);
```

```
theta = p(8);
epsilon = p(9);
delta = p(10);
beta = p(11);
bss = p(12);
T = p(13);
kT0 = p(14);
kN0 = p(15);
%Asignación de variables
for t=1:T-1
kT(t) = z(3*t-2);
kN(t) = z(3*t-1);
b(t+1) = z(3*t);
end
kT(T) = z(3*t+1);
kN(T) = z(3*t+2);
r = 1/beta-1;
mpkT(1) = aT*alphaT*kT(1)^(alphaT-1);
mpkN(1) = aN*alphaN*kN(1)^(alphaN-1);
pN(1) = mpkT(1)/mpkN(1);
ratx(1) = gamma*pN(1)/(1-gamma);
xN(1) = (kT(1)+kN(1)-(1-delta)*(kT0+kN0))/(G*ratx(1)^gamma);
xT(1) = ratx(1)*xN(1);
q(1) = ratx(1)^(1-gamma)/(G*gamma);
yT(1) = aT*kT(1)^alphaT;
yN(1) = aN*kN(1)^alphaN;
cN(1) = yN(1)-xN(1);
cT(1) = ((1-epsilon)/(epsilon*pN(1)))^(1/(rho-1))*cN(1);
u(1) = theta*(epsilon*cT(1)^rho+(1-epsilon)*cN(1)^rho)^(theta/rho-
1);
    for t=2:T-1
    mpkT(t) = aT*alphaT*kT(t)^(alphaT-1);
    mpkN(t) = aN*alphaN*kN(t)^(alphaN-1);
    pN(t) = mpkT(t)/mpkN(t);
    ratx(t) = gamma*pN(t)/(1-gamma);
    xN(t) = (kT(t)+kN(t)-(1-delta)*(kT(t-1)+kN(t-1)))/(G*ratx(t)^gamma);
    xT(t) = ratx(t)*xN(t);
    q(t) = ratx(t)^(1-gamma)/(G*gamma);
```

```
    yT(t) = aT*kT(t)^alphaT;
    yN(t) = aN*kN(t)^alphaN;
    cN(t) = yN(t)-xN(t);
    cT(t) = ((1-epsilon)/(epsilon*pN(t)))^(1/(rho-1))*cN(t);
    u(t) = theta*(epsilon*cT(t)^rho+(1-epsilon)*cN(t)^rho)^(theta/rho-
1);
    end
kss = kT(T)+kN(T);
mpkTss = aT*alphaT*kT(T)^(alphaT-1);
mpkNss = aN*alphaN*kN(T)^(alphaN-1);
pNss = mpkTss/mpkNss;
ratxss = gamma*pNss/(1-gamma);
xNss = delta*kss/(G*ratxss^gamma);
xTss = ratxss*xNss;
yTss = aT*kT(T)^alphaT;
yNss = aN*kN(T)^alphaN;
cNss = yNss-xNss;
cTss = ((1-epsilon)/(epsilon*pNss))^(1/(rho-1))*cNss;
cT(T) = cTss;
cN(T) = cNss;
u(T) = theta*(epsilon*cT(T)^rho+(1-epsilon)*cN(T)^rho)^(theta/rho-
1);
q(T) = qss;
f(1) = aT*alphaT*kT(1)^(alphaT-1)+q(2)*(1-delta)-q(1)*(1+r);
f(2) = cT(1)+xT(1)+b(2)-aT*kT(1)^alphaT;
f(3) = u(1)*epsilon*rho*cT(1)^(rho-1)-u(2)*epsilon*rho*cT(2)^(rho-
1);
    for t=2:T-1
    f(3*t-2) = aT*alphaT*kT(t)^(alphaT-1)+q(t+1)*(1-delta)-q(t)*(1+r);
    f(3*t-1) = cT(t)+xT(t)+b(t+1)-aT*kT(t)^alphaT-(1+r)*b(t);
    f(3*t) = u(t)*epsilon*rho*cT(t)^(rho-1)-u(t+1)*epsilon*rho*cT(t+1)^(rho-
1);
    end
    f(3*t+1) = mpkTss-qss*(r+delta);
    f(3*t+2) = yTss+r*b(T)-cT(T)-xTss;
f=f';
```

En el programa **inver.m** nos encontramos, después de haber definido los parámetros del modelo y del programa, el conocido comando **sol = secant('sscpo', x0, param, crit, maxit);.** Este comando dispara a **secant.m** para operar sobre el sistema de ecuaciones introducido en **sscpo.m.** En este programa la primera línea **function f=sscpo(z, p)** dice que el programa es una función que depende de la variable **z** y del parámetro **p.** La variable **z** tiene dos componentes que son la semilla contenida en **x0 = [896; 981];** de modo que las líneas de comando **kTss = z(1);** y **kNss = z(2);** asignan como primer valor **kTss =896** y **kNss =981.** El listado que sigue a **%Asignación de parámetros** es la asignación apropiada a cada parámetro del valor que fue definido en **inver.m** en la línea de comando **param = [aT aN alphaT alphaN gamma G rho theta epsilon delta beta bss];** y cada uno de estos parámetros fue definido en la sección de definición de parámetros de modelo. Respetar el orden es absolutamente esencial. Una vez que el proceso haya terminado el resultado será guardado en la variable **sol** definida más arriba. Una vez que tenemos la **sol**ución al estado estacionario reconstruimos el valor de las variables. Esto tenemos que hacerlo en el programa **inver.m** porque las variables computadas dentro de **sscpo.m** no son variables globales, es decir, no son reconocidas en ningún lugar más que dentro de la función que las generó. Una vez que hemos computado el estado estacionario llegamos finalmente a la línea **%Estado estacionario computado**. Si detenemos el programa escribiendo un **pause** justo después de **sol = secant('sscpo', x0, param, crit, maxit)** y escribimos es la pantalla de comandos de MatLab el nombre de alguna variable que estuviera dentro de **sscpo.m** la respuesta será que la variable es desconocida. La razón es que las variables dentro de las funciones sólo son recordadas mientras son usadas, por eso tenemos que repetir en **inver.m** prácticamente todo el código que hay dentro de **sscpo.m**, que son básicamente las condiciones de primer orden y las condiciones de factibilidad en el estado estacionario.

1.2.1 Estado estacionario

El sistema de ecuaciones en el estado estacionario viene dado por:

$$\begin{aligned} c_{Tss} + x_{Tss} - A_T k_{Tss}^{\alpha_T} - r b_{ss} &= 0 \quad \mathbf{I} \\ c_{Nss} + x_{Nss} - A_N k_{Nss}^{\alpha_N} &= 0 \quad \mathbf{II} \end{aligned}$$

$$
\begin{aligned}
\delta(k_{Tss}+k_{Nss})-Gx_{Tss}^{\gamma}x_{Nss}^{1-\gamma} &= 0 \ \textbf{III}\\
u_{Tss}\varepsilon\rho c_{Tss}^{\rho-1}-1 &= 0 \ \textbf{IV}\\
u_{Nss}(1-\varepsilon)\rho c_{Nss}^{\rho-1}-p_{Nss} &= 0 \ \textbf{V}\\
-1-q_{ss}G\gamma x_{Tss}^{\gamma-1}x_{Nss}^{1-\gamma} &= 0 \ \textbf{VI}\\
-p_{Nss}-q_{ss}G(1-\gamma)x_{Tss}^{\gamma}x_{Nss}^{-\gamma} &= 0 \ \textbf{VII}\\
A_T\alpha_T k_{Tss}^{\alpha_T-1}-q_{ss}(\delta+r) &= 0 \ \textbf{VIII}\\
p_{Nss}A_N\alpha_N k_{Nss}^{\alpha_N-1}-q_t(\delta+r) &= 0 \ \textbf{IX}
\end{aligned}
$$

El sistema de ecuaciones que determina el estado estacionario tiene 9 ecuaciones y 9 incógnitas, que son $(c_{Tss}, x_{Tss}, k_{Tss}, c_{Nss}, x_{Nss}, k_{Nss}, b_{ss}, p_{Nss}, q_{ss})$. En principio nosotros podríamos introducir estas ecuaciones e incógnitas en un programa de MatLab, llamar a **secant.m** y terminar. No obstante es mejor proceder de otro modo. La dimensión del problema puede ser reducida muy fácilmente de un sistema de 9 ecuaciones en 9 incógnitas a uno de 2 ecuaciones en 2 incógnitas. Trabajar sobre un sistema más pequeño tiene la ventaja de que el algoritmo será más, rápido, más preciso y más estable. Para ver cómo podemos reducir la dimensión del problema empezamos imaginando que conocemos dos variables, como por ejemplo k_{Tss} y k_{Nss}. Si estas dos variables fueran conocidas, de las ecuaciones **VIII** y **IX** podríamos sacar el valor de p_{Nss}. Una vez conocido el valor de p_{Nss}, de las ecuaciones **VI** y **VII** podríamos extraer el valor de x_{Tss}/x_{Nss}. Conocido el valor de este ratio y nuestro supuesto conocimiento de k_{Tss} y k_{Nss}, podríamos extraer el valor de x_{Tss} de la ecuación **III**. Conocidos los valores de x_{Tss} y x_{Nss}, podemos averiguar sin dificultad el valor de q_{ss} de la ecuación **VI**. De la ecuación **II** extraeríamos el valor de c_{Nss}, y una vez conocido éste, del ratio entre las ecuaciones **IV** y **V** obtendríamos el valor de c_{Tss}. Nos quedarían sin comprobar dos ecuaciones que son la **I** y la **VIII**. Como son dos los valores que hemos supuesto, podemos iterar en los valores de k_{Tss} y k_{Nss}, de modo que si las dos ecuaciones finales no se cumplen probamos con otro par de valores de k_{Tss} y k_{Nss}, y así sucesivamente hasta que todas la ecuaciones se cumplen. Como se puede observar esto es exactamente lo que hace el programa **sscpo.m**.

1.2.2 Dinámica

La dinámica esta programada en **dincpo.m** cuya construcción es idéntica a la de **sscpo.m**. Comienza realizando una asignación de parámetros que en

este caso contiene algunos adicionales muy importantes, como son **T**, que es la dimensión de la truncación y el par de valores de stock de capital inicial **kT0** y **kN0**. Luego se realiza la asignación de la semilla a cada una de las variables. La semilla fue cuidadosamente construida en el programa **inver.m.** La lógica sobre el funcionamiento de este algoritmo es idéntica al de los ya presentados y por tanto no nos extenderemos más sobre su contenido.

Llegados a este punto ya sólo queda resolver el modelo calibrado y comparar los resultados con los generados por la economía real. En este momento comienzan otros campos de investigación relacionados con el modelo básico. Restricción de crédito, posibilidades de fallar en la devolución de la deuda, estudiar los efectos del ciclo económico, y un largo etcétera es lo que faltaría para que nuestro estudio sobre este tipo de economías fuera completamente exhaustivo. Pero esto es investigación futura en la que vosotros posiblemente queráis participar.