2
Programación Dinámica Práctica

Kamal A. Romero S.

Sumario

Se repasa la implementación numérica de la iteración de la función valor, método sencillo que nos permitirá la resolución de problemas que admitan representación recursiva. Se analizan los casos determinista y estocástico.

Contenido

1. Introducción

El objetivo del presente capítulo es presentar uno de los métodos que nos permitirán resolver modelos recursivos, a saber, la iteración de la función valor. Este método nos permitirá obtener las reglas de decisión y función valor óptimas de un problema que admita representación recursiva.

La exposición de este capítulo no es exhaustiva, no se exponen los principios teóricos de la programación dinámica[1], ni los de la aproximación numérica[2]. El objetivo es que el lector adquiera una idea rápida de los métodos de resolución a utilizar a lo largo del texto, y así pueda comenzar a utilizar las herramientas lo antes posible.

Asimismo, no expondremos otros métodos tales como la iteración de la función de política o los métodos de proyección. No obstante, dado que muchos de los problemas interesantes que involucran agentes heterogéneos suelen presentar no convexidades, suelen ser aproximados numéricamente mediante la iteración de la función valor.

Presentaremos la versión determinista y su extensión estocástica, utilizando como ejemplo el modelo neoclásico de crecimiento.

2. Programación Dinámica Determinista

Vamos a comenzar con la versión no estocástica del modelo neoclásico de crecimiento con oferta de trabajo exógena.

2.1. La Economía

Entorno

Existe un continuo de consumidores idénticos de medida 1 que viven infinitos períodos y que poseen acceso a una tecnología de producción.

Dado que los individuos son idénticos, y que se cumplen todas las condiciones a través del cual son equivalentes el equilibrio competitivo de descentralizado y el problema del planificador, trabajamos bajo la representación de agente representativo.

Consumidores

El ordenamiento de preferencias viene representado por una función de utilidad estándar:

$$U = \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{1}$$

donde $\beta \in (0, 1)$ es la tasa subjetiva de descuento.

[1] Para una exposición detallada ver Stokey and Lucas (1989)

[2] Puede encontrar un buen tratamiento de libro de texto en Judd (1998)

La función de utilidad instantánea $u(c_t)$ es crecientes, continua dos veces diferenciable, estrictamente cóncava y cumple la llamada condición de Inada[3].

La tecnología es neoclásica y depende sólo del capital, adopta la siguiente forma:

$$Y = F(K) = Ak_t^{\alpha} \tag{2}$$

donde $\alpha \in (0, 1)$

La representación secuencial del problema del consumidor viene dada por:

$$\begin{aligned} &\underset{\{c,k_{t+1}\}_{t=0}^{\infty}}{max} \sum_{t=0}^{\infty} \beta^t u(c_t) \\ &\text{sujeto a} \\ &c_t + i \leq Ak_t^{\alpha} \\ &k_0 \geq 0 \\ &c_t, k_t \geq 0 \ \ \forall t \end{aligned} \tag{3}$$

donde i es la inversión o acumulación de capital del consumidor

Podemos construir la representación recursiva de este problema secuencial una vez definida de manera adecuada la variable de estado y su ley de movimiento.

La variable de estado del problem 3 es el nivel de capital k_t. Si asumimos que el capital se deprecia a una tasa δ, la ecuación de movimiento de la variable de estado puede escribirse de la siguiente forma:

$$k_{t+1} = (1 - \delta)k_t + i \tag{4}$$

Ya poseemos los elementos necesarios escribir la representación recursiva de 3:

$$\begin{aligned} v(k) = &\underset{c,k'}{max}[u(c) + \beta v(k')] \\ &\text{sujeto a} \\ &c + i = F(k) \\ &k' = (1 - \delta)k + i \\ &k \in X = (0, \overline{k}) \end{aligned} \tag{5}$$

las primas representan el valor de la variable el próximo período.

La solución de este problema viene dado por:

- Una función valor $v^*(k)$, $v : X \Rightarrow \mathbb{R}$
- Funciones de políticas asociadas $k' = g^k(k)$, $g : X \Rightarrow X$ y $c = g^c(k)$, $g : X \Rightarrow X$ que resuelvan el problema arriba enunciado.

[3] $\lim_{c \to 0} u'(c) \to +\infty$

El objetivo de este capítulo es resolver numéricamente el problema 5 en una computadora a través del lenguaje MATLAB.

Para ello debemos discretizar el espacio de los estados (y el de los controles, que en este caso es el mismo). Por lo tanto, X será un vector de tamaño n.

Asimismo, la función valor $v^*(k)$, que nos indica el máximo de la función de retorno para cada nivel de k, también será un vector. Y siguiendo un razonamiento similar, las funciones de política $g^k(k)$ y $g^c(k)$ que relacionan las decisiones óptimas de c y k' con el nivel de capital, a su vez serán vectores.

3. Programación Dinámica

El funcional que aparece en el problema 5 es la llamada ecuación de Bellman. Esta ecuación funcional debe ser resuelta para dos funciones desconocidas $v(k)$ y $g^k(k)$. La teoría de programación dinámica estudia los métodos utilizados para obtener dichas funciones.

Bajo ciertas restricciones de concavidad en la función $u(c)$ y la correspondencia $\Gamma = \{k' \leq F(k) + (1-\delta)k - c\}$ Los principales resultados de esta teoría son:

1. La ecuación funcional de 5 posee una solución única estrictamente cóncava.
2. La solución es aproximada en el limite $n \to \infty$ mediante iteraciones de la forma
$$v^{n+1} = \max_{k'}\{u(k,k') + \beta v^n(k')\}$$
comenzando desde cualquier v^0 sujeto a Γ
3. Existe una única función de política invariante $g^k(k)$ tal que maximize el lado derecho de la ecuación de Bellman.

4. Iteración de la Función Valor

4.1. Caso Determinista

Sustituyendo la restricción en la ecuación 5 obtenemos:

$$v(k) = \max_{k'}[u(F(k) + (1-\delta)k - k') + \beta v(k')] \quad (6)$$
sujeto a
$$k' \in X = (0, \overline{k})$$

Según los resultados teóricos enunciados en la sección anterior, dado un valor inicial para la función v, por ejemplo v^0, la secuencia de funciones definida por el operador del máximo:

$$v^{n+1} = \max_{k'}\{u(F(k) + (1-\delta)k - k') + \beta v^n(k')\}$$

Es una contracción que converge a la solución v^*.

La idea de la iteración de la función valor es implementar numéricamente este operador a través de una proceso iterativo, en el cual luego de un determinado número de iteraciones obtengamos la función valor óptima y sus funciones de política asociadas.

Lo primero que debemos hacer es discretizar el espacio de los estados. Como podemos observar en 6, todo el problema queda planteado en términos de k, por lo tanto definimos un vector de tamaño n que nos indique los diferentes valores que puede tomar el capital.

Una vez construido este vector podemos obtener valores para el consumo, la función de utilidad y por ende la función valor v.

Para poder obtener valores concretos del consumo y la utilidad, debemos asignar formas funcionales a u y $F(k)$, y valores a sus parámetros.

Siguiendo la tradición de los modelos de crecimiento neoclásico utilizaremos la función de producción de la expresión 2, mientras que para la función de utilidad utilizaremos una función de elasticidad de sustitución constante de la forma:

$$u = \frac{c_t^{1-\sigma} - 1}{1 - \sigma} \tag{7}$$

donde $\sigma \geq 0$.

Con todos estos elementos podemos implementar la iteración de la función valor, que nos permita resolver el problema 5.

4.1.1. Implementación Numérica

Como hemos expuesto arriba debemos discretizar el problema, calcular el consumo, la utilidad y la función valor. Una vez obtenido esto construimos un bucle que realize iteraciones de la expresión 6 dado un valor incial para v.

Sistematizando la anterior, los pasos a realizar son los siguientes:

1. Definir los parámetros de la función de utilidad y producción.
2. Construir un vector de valores (grid) para el capital (k) de tamaño n
3. Utilizar el grid del capital para construir una matriz de consumo de tamaño $(n \times n)$ vía la expresión $c(k, k') = Ak_t^{\alpha} + (1 - \delta)k - k'$
4. Utilizar la matriz de consumo del punto anterior y construir una matriz de utilidades de tamaño $(n \times n)$ $U(k, k') = \frac{c_t^{1-\sigma}-1}{1-\sigma}$
5. Para iniciar la iteración sobre la expresión 6 debemos definir un vector de tamaño $(1 \times n)$ que represente el valor inicial de la función valor v^0. A partir de v^0 construimos el primer paso de la expresión 6 $v^1(k, k') = \underset{k'}{max}\{u(Ak_t^{\alpha} + (1 - \delta)k - k') + \beta v^0\}$
6. Iteramos en la ecuación anterior hasta que $|v^{n+1} - v^n| \leq tol$

7. Una vez que converga la función v^* obtenemos la función de política para el capital $k' = g^k(k)$ aplicando el operador $argmax$, $g^k(k) = argmax\{u(Ak_t^\alpha + (1-\delta)k - k') + \beta v^*\}$

8. Calculamos la función de política para el consumo a través de la restricción de presupuesto $c = g^c(k) = Ak_t^\alpha + (1-\delta)k - g^k(k)$

4.1.2. Pasos Computacionales

Enumeramos los pasos computacionales a realizar en cada paso.

- Paso 1: Una vez parametrizado el modelo cosntruimos un vector (grid) de tamaño $1 \times n$, que asignen valores al capital.

Podemos construir el grid de varias formas, asignando límites arbitrarios (por ejemplo un mínimo de cero y un máximo obtenido por ensayo y error) o aproximando los niveles de capital alrededor del nivel de estado estacionario.

Construimos primero un grid con límites arbitrarios:

```
%GENERACION DEL GRID
%Nivel minimo de capital en el grid
MINK=0.1;
%Nivel maximo de capital en el grid
MAXK=20;
%Tamaño del incremento en el grid
INK=0.1;
GRID=[MINK:INK:MAXK];
%Número de celdas en el grid
NK=length(GRID);
```

Si se desea un número especifico de valores para el grid `NK`, con la única condición que los espacios entre los distintos puntos del capital sean los mismos, podemos utilizar el comando `linespace` de MATLAB.

```
%Numero de elementos en el grid
%NK=200;
GRID=linspace(MINK,MAXK,NK);
```

Por último, si estamos trabajando con un modelo de crecimiento podemos construir el grid en un entorno del nivel de capital de estado estacionario. En este caso lo primero es determinar dicho nivel, por la teoría de crecimiento económico sabemos que viene dado por:

$$k^* = \left(\frac{1-(1-\delta)\beta}{\alpha\beta A}\right)^{\frac{1}{\alpha-1}} \tag{8}$$

Una vez obtenido el valor de k^*=`Kss` generamos el grid:

```
%Nivel minimo de capital en el grid
MINK=0.95*Kss;
```

```
%Nivel maximo de capital en el grid
MAXK=1.01*Kss;
%Número de celdas en el grid
NK=200;
%Tamaño del incremento en el grid
INK=((MAXK-MINK)/(NK-1));
GRID=[MINK:INK:MAXK];
```

En este caso en lugar de imponer el incremento en el grid, lo definimos de manera tal que el espacio entre los niveles de capital sean iguales.

- Paso 2: Utilizamos la expresión para el consumo $c(k,k') = Ak_t^{\alpha} + (1-\delta)k - k'$ y construimos una matriz $n \times n$. El grid del capital tambien sirve para representar los valores de k'.

Construimos una matriz en lugar de un vector porque el consumo depende de dos variables, k y k', las cuales se encuentran representadas por el mismo vector.

Tomar en consideración que el consumo es creciente en k y decreciente en k'.

El objeto es crear una matriz de la forma:

$$C(k,k') = \underset{\downarrow}{k'} \quad \overbrace{\left\{ \begin{pmatrix} C(k_1,k_1') & C(k_2,k_1') & \dots & C(k_n,k_1') \\ C(k_1,k_2') & C(k_2,k_2') & \dots & C(k_n,k_2') \\ \vdots & \vdots & \ddots & \vdots \\ C(k_1,k_n') & C(k_2,k_n') & \dots & C(k_n,k_n') \end{pmatrix} \right.}^{k \rightarrow}$$

Donde el consumo aumenta conforme nos movemos a la derecha de la matriz y disminuye conforme nos movemos hacia abajo.

Podemos construir esta matriz a través de un bucle o a través de un producto matricial. Escogemos la segunda estrategia ya que implica una ganancia en términos de tiempo de computación.

Definimos un vector de unos de tamaño $(1 \times n)$

```
I=ones(1,NK);
```

Escribimos la expresión $k' + (1-\delta)k$

```
inversion=GRID'*I-((1-DELTA)*(I'*GRID));
```

Cuando realizamos la multiplicación `GRID'*I` creamos una matriz de tamaño $(n \times n)$ donde las columnas adoptan los valores del grid

Ejemplo 4.1. Supongamos que el grid de k viene dado por:

```
187.8174 191.7714 195.7255 199.6795
```

Si realizamos la multiplicación `GRID'*I` obtenemos:

```
187.8174  187.8174  187.8174  187.8174
191.7714  191.7714  191.7714  191.7714
195.7255  195.7255  195.7255  195.7255
199.6795  199.6795  199.6795  199.6795
```

Con la inversión escribimos el consumo

```
consumo=I'*(A*GRID.^ALFA)-inversion;
```

Ejemplo 4.2. Utilizando el vector del ejemplo anterior, obtenemos la siguiente matriz de consumo:

```
51.2205 55.3947 59.5627 63.7249
47.2665 51.4406 55.6087 59.7708
43.3124 47.4866 51.6546 55.8168
39.3584 43.5325 47.7006 51.8627
```

- Paso 3: Utilizar la matriz de consumo del punto anterior y construir una matriz de utilidades de tamaño $(n \times n)$ $U(k, k') = \frac{c_t^{1-\sigma}-1}{1-\sigma}$

Antes de construir la matriz de consumo debemos verificar si se cumple la restricción de que el consumo no sea negativo. Si existe para determinado niveles de k y k' un consumo menor a cero, le asignamos el número real negativo más grande que tiene MATLAB, de manera tal que cuando realicemos la maximización dichos niveles de capital nunca serán escogidos.

```
   for i=1:NK
for j=1:NK
if consumo(i,j)<=0;
consumo=-realmax;
end;
end;
end;
```

Posteriormente evaluamos la matriz de consumo con nuestra función de utilidad.

```
   %CALCULO DE LA UTILIDAD;
if SIGMA==1
U=log(consumo);
else
U=1/(1-SIGMA)*(consumo.^(1-SIGMA));
end;
```

Ejemplo 4.3. Utilizando la matriz de consumo del ejemplo anterior, obtenemos la siguiente matriz de utilidades:

```
-0.0195 -0.0181 -0.0168 -0.0157
-0.0212 -0.0194 -0.0180 -0.0167
-0.0231 -0.0211 -0.0194 -0.0179
-0.0254 -0.0230 -0.0210 -0.0193
```

- Paso 4: Para iniciar la iteración sobre la expresión 6 debemos definir un vector de tamaño $(1 \times n)$ que represente el valor inicial de la función valor v^0. A partir de v^0 construimos el primer paso de la expresión 6 $v^1(k, k') = \underset{k'}{max}\{u(Ak_t^\alpha + (1-\delta)k - k') + \beta v^0\}$

Definimos la función valor inicial:

`v1=zeros(1,NK);`

Calculamos el primer paso de la ecuación de Bellman. Previamente definimos un vector de tamaño $(n \times 1)$, para poder realizar el producto βv^0.

`o=ones(NK,1);`

`Tv1=max(U+(o*BETA*v1)');`

El producto `o*BETA` es un vector columna de tamaño $(n \times 1)$ donde todos sus elementos son iguales a β. Y la multiplicación `o*BETA*v1` (recordar que `v1` es un vector fila de tamaño $(1 \times n)$) es una matriz $(n \times n)$ de la siguiente forma:

$$\begin{pmatrix} U(k_1,k'_1) & U(k_2,k'_1) & \dots & U(k_n,k'_1) \\ U(k_1,k'_2) & U(k_2,k'_2) & \dots & U(k_n,k'_2) \\ \vdots & \vdots & \ddots & \vdots \\ U(k_1,k'_n) & U(k_2,k'_n) & \dots & U(k_n,k'_n) \end{pmatrix} + \beta \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

$$\begin{pmatrix} U(k_1,k'_1)+\beta\times 0 & U(k_2,k'_1)+\beta\times 0 & \dots & U(k_n,k'_1)+\beta\times 0 \\ U(k_1,k'_2)+\beta\times 0 & U(k_2,k'_2)+\beta\times 0 & \dots & U(k_n,k'_2)+\beta\times 0 \\ \vdots & \vdots & \ddots & \vdots \\ U(k_1,k'_n)+\beta\times 0 & U(k_2,k'_n)+\beta\times 0 & \dots & U(k_n,k'_n)+\beta\times 0 \end{pmatrix} \quad (9)$$

El comando `max` escoge el valor máximo de las columnas de una matriz. En el caso de la matriz 9, el comando `max` crea un vector que contiene el argumento máximo de cada columna:

$$\begin{pmatrix} v_1(k_1) \\ v_1(k_2) \\ v_1(k_3) \\ v_1(k_4) \end{pmatrix} = \underset{k'}{max} \left\{ \begin{pmatrix} U(k_1,k'_1)+\beta\times 0 & U(k_2,k'_1)+\beta\times 0 & \dots & U(k_n,k'_1)+\beta\times 0 \\ U(k_1,k'_2)+\beta\times 0 & U(k_2,k'_2)+\beta\times 0 & \dots & U(k_n,k'_2)+\beta\times 0 \\ \vdots & \vdots & \ddots & \vdots \\ U(k_1,k'_n)+\beta\times 0 & U(k_2,k'_n)+\beta\times 0 & \dots & U(k_n,k'_n)+\beta\times 0 \end{pmatrix} \right\}$$

El vector $v_1(k_i)$ nos dice cual es el nivel de k' que maximiza la expresión $v^1(k, k') = \underset{k'}{max}\{u(k, k') + \beta v^0\}$ para cada k_i

Ejemplo 4.4. A partir de nuestra matriz de utilidades y nuestra función valor inicial de `0 0 0 0`, obtenemos la siguiente matriz $u(k,k') + \beta v^0$

```
-0.0195 -0.0181 -0.0168 -0.0157
-0.0212 -0.0194 -0.0180 -0.0167
-0.0231 -0.0211 -0.0194 -0.0179
-0.0254 -0.0230 -0.0210 -0.0193
```

Que no es mas que la misma matriz de utilidades.

Realizando el primer paso de la iteración de la ecuación de Bellman obtenemos el vector $v_1(k_i)$

```
-0.0195 -0.0181 -0.0168 -0.0157
```

Que es el valor máximo de cada columna de la matriz de la ecuación de Bellman. En este caso el nivel de k' que maximiza $v(k,k')$ es el correspondiente al primer espacio del grid, es decir, el nivel más bajo de capital.

Este resultado es completamente lógico cuando $v^0 = 0$, dado que el consumo cae al escoger un mayor k' cuando no existe ningún efecto intertemporal.

- Paso 5: Iteramos en $v^{n+1}(k,k') = \underset{k'}{max}\{u(Ak_t^\alpha + (1-\delta)k - k') + \beta v^n\}$ hasta que $|v^{n+1} - v^n| \leq tol$

Esta es la etapa principal de la implementación numérica, en la cual aplicando el operador max generamos una secuencia de funciones valor v^n hasta localizar un punto fijo.

Suele ser muy sencillo implementar cualquier tipo de iteración en MATLAB a través de un bucle, en el cual repetimos un número determinado de veces una operación.

En nuestro caso deseamos repetir la operación $v^{n+1}(k,k') = \underset{k'}{max}\{u(Ak_t^\alpha + (1-\delta)k - k') + \beta v^n\}$ hasta que $v^{n+1} \backsimeq v^n$.

En el paso anterior ya hemos realizado el primer paso de la iteración tomando como dado un valor inicial para v^0, obteniendo v^1.

Comparamos v^1 y v^0 utilizando alguna métrica, si esta es menor a cierto nivel de tolerancia ya hemos terminado. En caso contrario, utilizamos v^1 como $v(k')$ y obtenemos a través de la expresión $v^2(k,k') = \underset{k'}{max}\{u(Ak_t^\alpha + (1-\delta)k - k') + \beta v^1\}$ un valor para v^2, el cual comparamos con v^1 y repetimos la operación anterior.

Para construir el bucle que nos implemente lo arriba expuesto, lo primero que debemos hacer es definir un nivel de tolerancia, un valor inicial para la métrica con la cual medimos la distancia entre v^{n+1} y v^n e inicializar el contador del número de iteraciones.

```
   %Iniciar el contador
n=1;
%Numero maximo de iteraciones
```

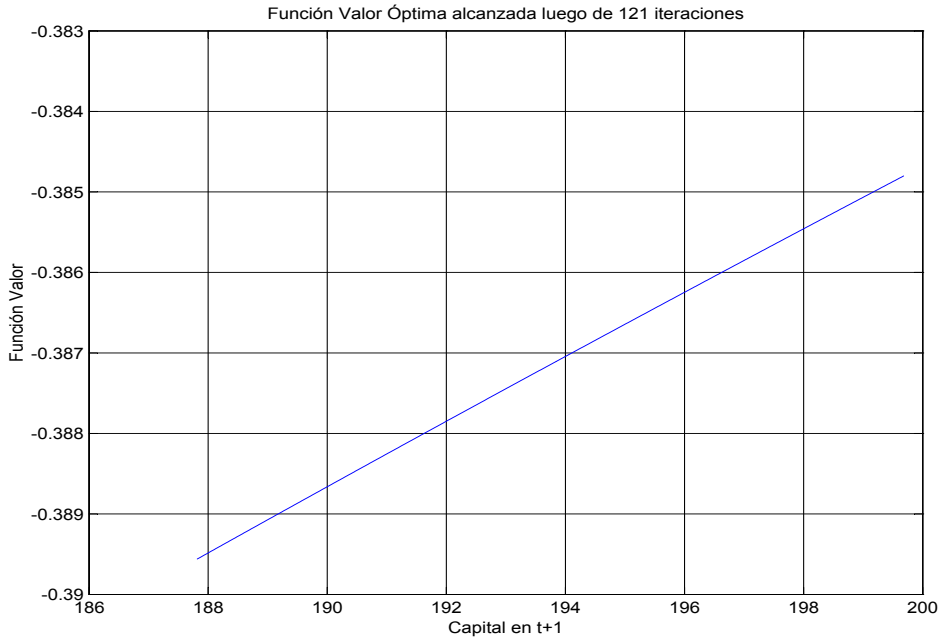

Figura 1: Función Valor Caso Determinista

```
MAXIT=2500;
%Nivel de tolerancia
TOL=1*10^(-3);
%Valor inicial de la metrica
distance1=1;
```

Definimos el(los) criterio(s) de parada.

```
while distance1>TOL & n<MAXIT;
```

Ejecutamos $v^{n+1}(k,k') = \underset{k'}{max}\{u(Ak_t^{\alpha} + (1-\delta)k - k') + \beta v^n\}$.

```
Tv1=max(U+(o*BETA*v1)');
```

Evaluamos la distancia entre v^{n+1} y v^n

```
distance1=norm(abs(Tv1-v1));
```

De no cumplirse el condicional `while`

```
v1=Tv1;
n=n+1;
```

En caso contrario

```
end
```

- Paso 6: Una vez que converga la función v^* obtenemos la función de política para el capital $k' = g^k(k)$ aplicando el operador $argmax$, $g^k(k) = argmax\{u(Ak_t^{\alpha} + (1-\delta)k - k') + \beta v^*\}$

Recordar que el comando `max` de MATLAB nos da el valor máximo de cada columna de una matriz, en nuestro problema eso equivale a el valor máximo de la función valor para cada nivel de capital hoy k. Esto nos permite obtener valores para la función valor.

Asimismo nos interesa saber no sólo el valor máximo de la función valor para cada nivel de k, sino a que nivel de capital el próximo período k' se encuentra asociado dicho valor máximo.

Analíticamente, esta información viene dada por la función de política $k' = g^k(k)$, la cual nos indica que niveles de k' escoge el agente para cada nivel de k. Dado que $g^k(k)$ se define como una solución del problema 5, el agente escogerá aquellos niveles de k' que maximicen la función valor.

O lo que es igual, aquellos asociados a los valores de v obtenidos con el operador max.

El comando `max` de MATLAB no sólo nos proporciona el valor máximo de cada columna, también nos puede indicar en que fila de la columna se encuentra localizado dicho valor, lo cual es equivalente a señalarnos a que nivel de k' se encuentra asociado ese valor[4] o la función de política $k' = g^k(k)$.

Sencillamente le señalamos al comando `max` que nos indique el valor y el índice de la fila del siguiente modo:

```
[valuefunction,j]=max(U+(o*BETA*v1)');
```

donde `j` es un vector que indica el número de fila en el cual se localiza el máximo de cada columna.

En nuestro caso `j` nos dice en que posición del grid se encuentra el valor de k' que maximiza la función valor para cada k.

Para obtener la función de política $g^k(k)$ sencillamente localizamos los valores en el grid en función de los índices `j`.

```
policyfunctionKap=GRID(j);
```

- Paso 7: Calculamos la función de política para el consumo a través de la restricción de presupuesto $c = g^c(k) = Ak_t^\alpha + (1-\delta)k - g^k(k)$

Para obtener la función $c = g^c(k)$ aplicamos la expresión del consumo y sustituimos la función $g^k(k)$ por k'

```
policyfunctionCons=(A*GRID.^ALFA)+(1-DELTA)*GRID-(policyfunctionKap);
```

4.2. Caso Estocástico

En el presente caso la economía descrita al inicio se encuentra sujeta a algún tipo de choque estocástico. Asumiremos que experimenta choques tecnológicos que afectan la capacidad de producir bienes en cada período.

Podemos modelizar lo anterior modificando la tecnología 2 del siguiente modo:

$$Y = F(K, z) = zAk_t^\alpha \tag{10}$$

[4]Recordar que los valores de k' se encuentran representados en las filas.

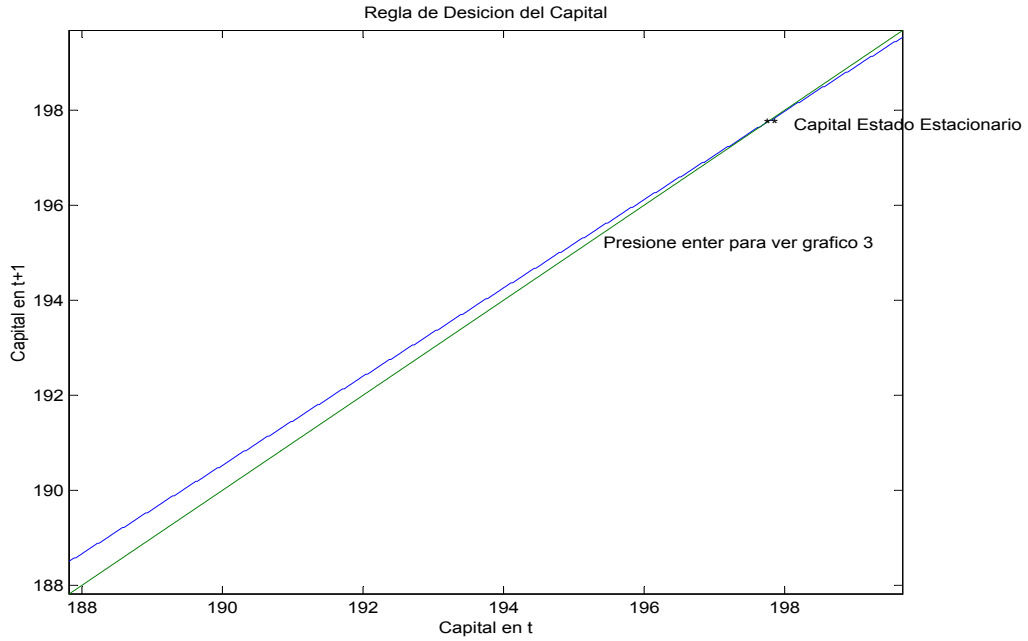

Figura 2: Función de Política del capital

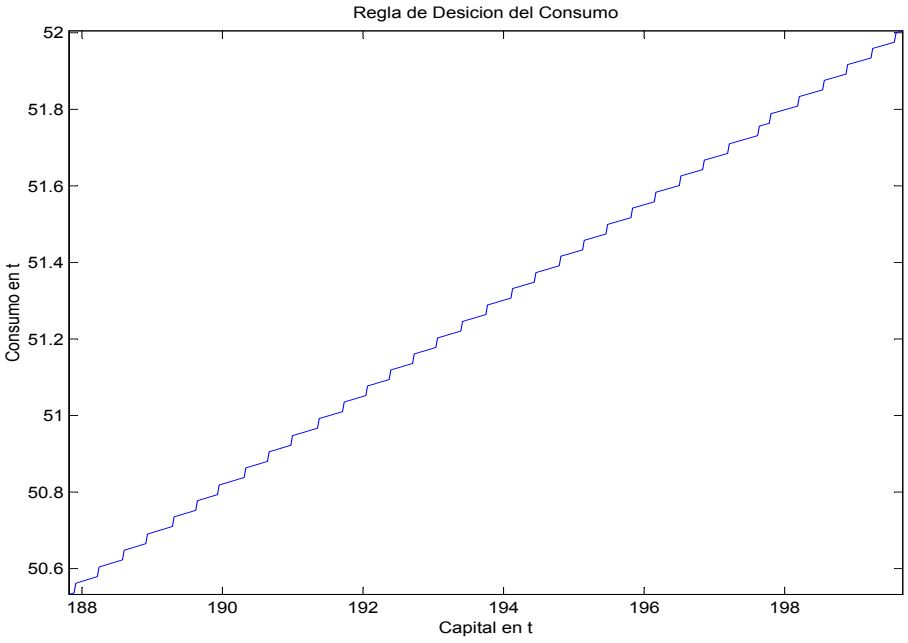

Figura 3: Función de Política del Consumo

Donde z es una variable aleatoria y representa la fuente de incertidumbre del modelo.

La tecnología pasa a ser una variable aleatoria, por lo tanto el consumo y la utilidad también lo serán.

En este caso para que el agente pueda calcular su utilidad y resolver el programa 3 con la tecnología 10, es necesario que conozca los posibles valores de z y sus probabilidades de ocurrencia.

Asumiremos que z siguen un proceso de Markov, debido a que mantendremos este supuesto a lo largo del texto por razones a ser expuestas más adelante.

En este caso necesitamos tres elementos:

- La distribución inicial de los estados z.
- La matriz de transición entre estados que resume la probabilidad de pasar a z_i dado z hoy $\pi(z_{t+1} = z_i \mid z_t = z)$
- Las probabilidades de ocurrencia de los estados. Con estos elementos es posible calcular tanto la matriz de transición como la distribución estacionaria de z

Distribución inicial $Z = \{1, 2\ ;\ 0, 8\}$

Matriz de transición P= $\begin{pmatrix} 0,8 & 0,2 \\ 0,5 & 0,5 \end{pmatrix}$

En este ejemplo sencillo la variable z sólo puede tomar dos valores, por lo que estamos en un modelo estocástico con dos estados de la naturaleza, los cuales llamaremos "bueno" y "malo".

En esta caso el consumo adopta dos valores para cada nivel de k y k':

$$\begin{aligned} c_t^m &= z_m k_t^\alpha + (1-\delta)k - k' \\ c_t^b &= z_b k_t^\alpha + (1-\delta)k - k' \end{aligned}$$

De igual modo la utilidad adopta dos valores para cada nivel de k y k':

$$\begin{aligned} U_t^m &= \frac{(c_t^m)^{1-\sigma} - 1}{1-\sigma} \\ U_t^b &= \frac{(c_t^b)^{1-\sigma} - 1}{1-\sigma} \end{aligned}$$

Por ende, tenemos dos valores para la función valor y dos pares de funciones de política asociada a cada una.

$$v(k, z_m) = \max_{c,k'} [u(c, z_m) + \beta E v(k', z')] \tag{11}$$

$$v(k, z_b) = \max_{c,k'} [u(c, z_b) + \beta E v(k', z')] \tag{12}$$

4.2.1. Implementación Numérica

Las dos principales diferencias a nivel numérico de este modelo con su versión determinista son:

1. Debemos calcular una matriz de consumo y utilidades por cada estado de la economía, e iterar en un número igual de funciones valor

2. Debemos computar el valor esperado de la función valor $Ev(k', z')$.

La primera diferencia es trivial, sólo es necesario repetir los mismos pasos para distintos valores del producto.

La segunda diferencia no representa mayor complicación, definiendo $Ev(k', z')$ por la expresión:

$$Ev(k', z') = v(k', z'_m)\pi_m + v(k', z'_b)\pi_b \tag{13}$$

que no es más que un simple producto matricial.

El procedimiento computacional es casi idéntico al caso estocástico, por lo que la mayoría de los pasos numéricos son repeticiones de este tal y como se observa a continuación:

1. Definir los parámetros de la función de utilidad y producción y el proceso estocástico z.

2. Construir un vector de valores (grid) para el capital (k) de tamaño n

3. Utilizar el grid del capital para construir s matrices de consumo de tamaño $(n \times n)$ vía la expresión $c(k, k', z_s) = z_s A k_t^{\alpha} + (1 - \delta)k - k'$, donde s es el número de estados de la naturaleza.

4. Utilizar las matrices de consumo del punto anterior y construir s matrices de utilidades de tamaño $(n \times n)$ $U(k, k', z_s) = \frac{(c_t^s)^{1-\sigma}-1}{1-\sigma}$

5. Definir s vectores de tamaño $(1 \times n)$ que represente el valor inicial de la función valor v^0. A partir de v^0 construimos el primer paso de las expresiones 11 $v(k, z_s) = \underset{c,k'}{max}[u(c, z_s) + \beta Ev(k', z')]$

6. Iteramos en la ecuaciones anteriores hasta que convergan.

7. Una vez que convergan la funciones v_s^* obtenemos las s funciones de política para el capital y el consumo.

4.2.2. Pasos Computacionales

Dado que la mayoría de los pasos computacionales son muy parecidos al caso estocástico la siguiente exposición va a ser un poco menos detallada al principio.

- Definimos los parámetros del proceso estocástico z

```
%PROCESO ESTOCASTICO;
%Distribución de Probabilidad del parámetro tecnológico
THETA=[1.2 0.8];
%Número de estados
S=length(THETA);
%Probabilidad de quedarse en el estado b (estando en el estado b)
Pbb=0.8;
%Probabilidad de transitar al estado m (estando en el estado b)
Pbm=1-Pbb;
%Probabilidad de quedarse en el estado m (estando en el estado m)
Pmm=0.5;
%Probabilidad de transitar al estado b (estando en el estado m)
Pmb=1-Pmm;
%Matriz de transición
Q=[Pbb Pbm; Pmb Pmm];
```

- Calculamos las s matrices de consumo y utilidad

```
%CALCULO DEL CONSUMO Y LA INVERSION;
I=ones(1,NK);
inversion=GRID'*I-((1-DELTA)*(I'*GRID));
consumoalto=I'*((THETA(1)*A*GRID).^ALFA)-inversion;
consumobajo=I'*((THETA(2)*A*GRID).^ALFA)-inversion;
%CALCULO DE LA UTILIDAD;
U=zeros(NK,NK);
if SIGMA==1
Ualto=log(consumoalto);
Ubajo=log(consumobajo);
else
Ualto=(1/(1-SIGMA))*(consumoalto.^(1-SIGMA));
Ubajo=(1/(1-SIGMA))*(consumobajo.^(1-SIGMA));
end
%Se le asigna un valor extremadamente bajo a los consumos no factibles
for i=1:NK
for j=1:NK
if consumoalto(i,j)<=0;
Ualto(i,j)=-realmax;
end;
if consumobajo(i,j)<=0;
Ubajo(i,j)=-realmax;
end;
end;
```

```
end;
```

- Calculamos el primer paso de las s funciones valor $v^1(k, z_s) = \underset{c,k'}{max}[u(c, z_s)+ \beta E v^0(k', z')]$

```
%FUNCION VALOR INICIAL;
v1=zeros(1,NK);
v2=zeros(1,NK);
o=ones(NK,1);
%Funcion valor en el estado bueno
Tv1=max(Ualto+(o*BETA*v1)');
%Funcion valor en el estado malo
Tv2=max(Ubajo+(o*BETA*v2)');
v1=Tv1;
v2=Tv2;
```

- Iteramos en las s ecuaciones valor $v^{n+1}(k, z_s) = \underset{c,k'}{max}[u(c, z_s)+\beta E v^n(k', z')]$ hasta la convergencia de las mismas

Al igual que el caso estocástico lo primero que se debe hacer es definir un nivel de tolerancia, inicializar el contador de iteraciones y un valor inicial para la métrica de la distancia $v_s^{n+1} - v_s^n$. Al depender la función valor de s debemos definir s valores iniciales para la métrica.

```
%Iniciar el contador
n=1;
%Numero maximo de iteraciones
MAXIT=2500;
%Nivel de tolerancia
TOL=1*10^(-8);
distance1=1;
distance2=1;
```

Definimos los criterios de parada

```
while distance1>TOL & distance2>TOL & n<MAXIT;
```

Como hemos mencionado arriba el otro paso adicional a los ya realizados en el caso estocástico era calcular el valor esperado $Ev(k', z')$, definido por la expresión 13.

13 puede ser calculado a partir de un producto vectorial, en el cual multiplicamos las probabilidades de transición por la función valor contingente al estado futuro

```
Q(1,1)*v1+Q(1,2)*v2
```

EL producto anterior es un vector de tamaño $(1 \times n)$, debemos convertirlo en un matriz $(1 \times n)$ para poder sumarlo a la matriz de utilidad.

Al igual que el caso determinista definimos un vector `o` de tamaño $(n \times 1)$ el cual multiplicamos al valor esperado.

```
(o*(Q(1,1)*v1+Q(1,2)*v2));
```

Ahora es posible definir la expresión completa para el valor esperado $\beta[v(k', z'_m)\pi_m + v(k', z'_b)\pi_b]$

```
%Valor presente función valor período siguiente en el estado bueno
hoy
Balto=BETA*(o*(Q(1,1)*v1+Q(1,2)*v2));
%Valor presente función valor período siguiente en el estado malo hoy
Bbajo=BETA*(o*(Q(2,1)*v1+Q(2,2)*v2));
```

Ejemplo 4.5. Con los parámetros de preferencias y tecnología de la versión determinista del modelo y los del proceso estocástico z, obtenemos los siguientes valores para el primer paso de la función valor:

```
v1 = -0.8730 -0.3934 -0.2580 -0.193
v2 = -1.0323 -0.4291 -0.2748 -0.2033
```

El valor presente para cada estado $v(k', z'_m)\pi_m + v(k', z'_b)\pi_b$ viene dado por:

```
Q(1,1)*v1+Q(1,2)*v2 = -0.9049 -0.4006 -0.2613 -0.1952
Q(2,1)*v1+Q(2,2)*v2 = -0.9527 -0.4113 -0.2664 -0.1982
```

Y la expresión $\beta[v(k', z'_m)\pi_m + v(k', z'_b)\pi_b]$ para cada s

```
Balto =
-0.8596 -0.3805 -0.2483 -0.1854
-0.8596 -0.3805 -0.2483 -0.1854
-0.8596 -0.3805 -0.2483 -0.1854
-0.8596 -0.3805 -0.2483 -0.1854

Bbajo =
-0.9050 -0.3907 -0.2531 -0.1883
-0.9050 -0.3907 -0.2531 -0.1883
-0.9050 -0.3907 -0.2531 -0.1883
-0.9050 -0.3907 -0.2531 -0.1883
```

Ya definidos los términos para el valor presente procedemos a realizar la iteración sobre las s funciones valor.

```
Tv1=max(Ualto+(Balto)');
Tv2=max(Ubajo+(Bbajo)');
V=[Tv1;Tv2];
distance1=norm(abs(Tv1-v1));
distance2=norm(abs(Tv2-v2));
v1=Tv1;
```

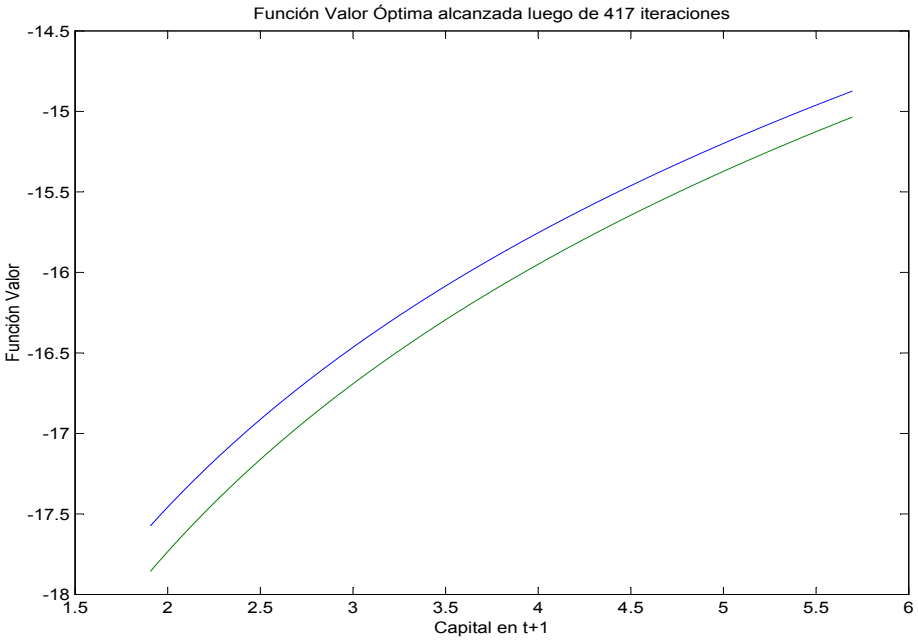

Figura 4: Funciones Valor Caso Estocástico

```
v2=Tv2;
n=n+1;
```

hasta la convergencia.

- Obtenemos las funciones de política

```
%CALCULO FUNCIONES DE POLITICA
[valuegood,j]=max(Ualto+(Balto)');
policygoodKap=GRID(j);
policygoodCons=(THETA(1)*A*GRID.^
ALFA)+((1-DELTA)*GRID)-(policygoodKap);
[valuebad,jj]=max(Ubajo+(Bbajo)');
policybadKap=GRID(jj);
policybadCons=(THETA(2)*A*GRID.^ALFA)+((1-DELTA)*GRID)-(policybadKap);
```

Ejercicio 4.6. Utilizar la función *determinista* y verifique que ocurre con las funciones de política si se incrementa la tasa subjetiva de descuento β

Ejercicio 4.7. Utilizar las función *estocastico* e intente incluir un tercer estado

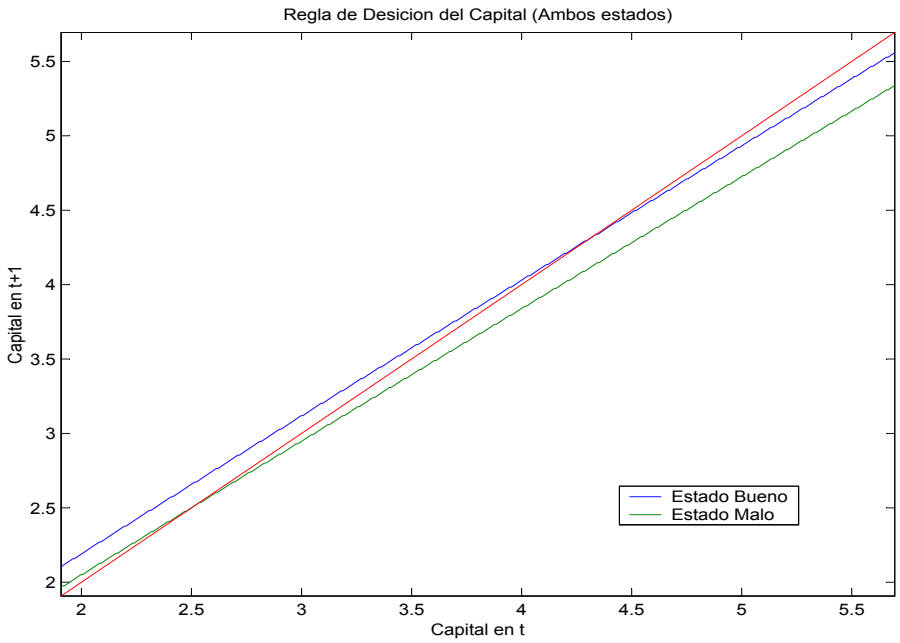

Figura 5: Funciones de Política del Capital Caso Estocástico

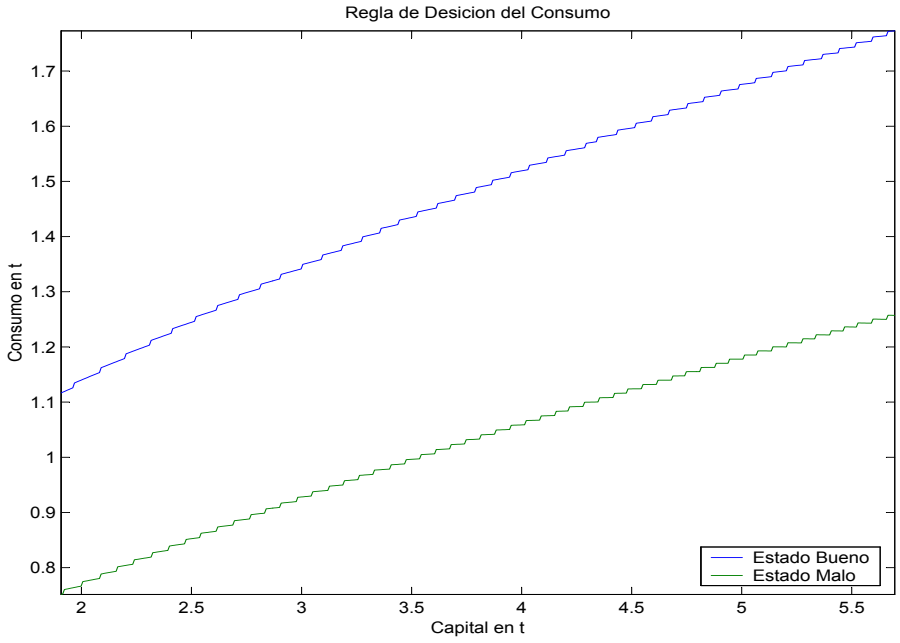

Figura 6: Funciones de Política del Consumo Caso Estocástico

de la naturaleza donde $z_3 = 4$ y la matriz de transición sea la siguiente:

$$\begin{pmatrix} 0,6 & 0,3 & 0,1 \\ 0,2 & 0,7 & 0,1 \\ 0,1 & 0,2 & 0,8 \end{pmatrix}$$

Referencias

Judd, K. L. (1998): *Numerical Methods in Economics*. The MIT Press.

Stokey, N. L., and R. E. Lucas (1989): *Recursive Methods in Economic Dynamics*. Harvard University Press.