Introducción a Matlab y Aplicaciones a Programación Dinámica

Hriday Karnani

March 19, 2024

Magíster y Doctorado en Economía. Facultad de Economía y Negocios, Universidad de Chile

Contenidos a cubrir

Introducción a Matlab y el curso

Programación Dinámica Determinística

- Aplicación 1: Modelo Neoclásico Determinístico de Crecimiento
- Aplicación 2: Modelo de Búsqueda

Programación Dinámica Estocástica

- Aplicación: Modelo Neoclásico Estocástico de Crecimiento

Introducción a Matlab y el curso

Qué haremos y qué esperar de estas sesiones?

- Aprenderemos lo básico de Matlab
  - su *environment*, crear funciones, trabajar con matrices, estructuras, condiciones lógicas, loops, algo de manipulación de datos, gráficos, etc.
  - útil para sus cursos de Econometría y Macro del magíster.
  - levemente distinto a R y Python. Distinto a Stata.
- Aplicaciones en Matlab a problemas de Programación Dinámica Determinística.
  - ejemplo: encontrar computacionalmente decisiones óptimas de consumo e inversión
  - ejemplo: modelo de búsqueda de empleo: encontrar salario de reserva
- Aplicaciones en Matlab a problemas de Programación Dinámica Estocástica.
  - ejemplo: encontrar computacionalmente decisiones óptimas de consumo e inversión bajo choques estocásticos

Qué esperar de estas sesiones?

- Qué esperar: Aprender a realizar ejercicios básicos en Matlab. Entender cómo pasar de lo lo teórico a lo práctico en problemas de Programación Dinámica.
- **Importante:** Pregunten.
  - primeras sesiones: enfóquense en entender la base de Matlab y qué hace cada código.
  - en Programación Dinámica: entender cómo se relacionan los códigos con la teoría.
- Todo el material estará disponible en Github (link). También será subido a Docencia Web.
- Disclaimer: Usaré material de bastantes fuentes en términos de código y del material de estas slides.

Metodología de esta parte del curso

- Clases:
  - *learning by doing*: tendremos un *script*, pero lo haremos desde cero, paso a paso.
  - Tenemos bloques de dos horas. Haremos un mini break de 5 minutos después de los primeros 70 minutos.
- Evaluaciones:
  - TBD

Qué es y Por qué Matlab?

- Matrix Laboratory (MatLab: Entorno de programación y software numérico)
  - entorno interactivo que permite realizar cálculos numéricos, implementar algoritmos, modelos matemáticos, etc.
  - se centra en la manipulación de matrices.
  - ofrece herramientas especializadas para la optimización.
- Por qué Matlab?
  - eficiencia en cálculos númericos (+ o -).
  - sintaxis simplicada para programación dinámica. Permite centrarnos en la lógica del modelo en lugar de detalles de su implementación.
  - es un lenguaje mucho más *hands-on*, usarán pocas funciones, programarán uds. la mayoría. Los obliga a entender lo que hacen.
- Entrega un enfoque distinto para pensar los datos. Al programar, su mente empezará a pensar de forma matricial. Súper interesante!

Descargando Matlab

En este link pueden descargar la versión para estudiantes. Pagada, pero con un descuento.

Hay otras formas de descargarlo...

El *environment* de Matlab

- Esta slide debe complementarse abriendo Matlab.
- **Command Window**: Acá es donde se ejecutan los cálculos y comandos Es la interfaz primaria para interactuar con Matlab. Es como la pantalla principal de Stata. Corramos cosas sencillas, como 2+2.
- **Workspace**: Acá se pueden observar todas las variables creadas en la sesión actual. Creemos a = 2+2 y observemos.
- **Current Folder**: Esta seección muestra los archivos en el directorio de Matlab en el que se encuentran actualmente.
- **Script**: Es usado para escribir y guardar una secuencia de comandos de Matlab. Permiten ejecutra múltiples comandos de una y son esenciales para hacer códigos reproducibles. Es como el do file de Stata.
- Exploraremos el resto en clases.
- Acá pueden encontrar una serie de atajos muy útiles
- Recomendación personal: Cambiar el shortcut para correr comandos para que sea igual al de su software más cercano. Lo veremos en clases.

Tipos de objetos y ejercicios básicos

De acá en adelante dejaremos las slides sólo para cuando tengamos que introducir tópios con más detalle. El core de la clase estará en el código, el cual está muy comentado para que sea fácil de seguir.

Haremos ejercicios básicos de Matlab siguiendo el PDF `Matlab_Basics.pdf` y los códigos `Session1_IntroMatlab.m` y `Session2_IntroMatlab.m`

Programación Dinámica Determinística

Feedback

Mini introducción

- Muchos problemas macro no dependen del tiempo sino de condiciones iniciales.
  - Por ejemplo, la inversión y las políticas iniciales afectan la senda de crecimiento a largo plazo.
- La naturaleza de los problemas se puede resolver de forma recursiva.
- Variables de Estado: resumen por completo la situación actual del agente (predeterminada).
- Variables de Control: Se escogen (y es posible controlarlas, valga la rebundancia).
- Estas sesiones: Aplicaciones prácticas de lo que verán con el Prof. Engel. Asumiremos los resultados que vean con él como dados y los aplicaremos en Matlab.

- Partimos con una aplicación muy sencilla. Un hogar maximiza su utilidad (bien portada) esperada a lo largo de su vida.

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

$$s.t \qquad c_t + k_{t+1} \leq f(k_t) \qquad \forall t \geq 0$$

- ¿Cuánto debe destinar a consumo e inversión hoy, si la inversión se transforma en capital productivo y, por ende, en consumo para mañana?

Economía Robinson Crusoe (cont.)

- Único bien que se destina a c_t o i_t. $y_t = f(k_t) = k_t^{\alpha}$. $\delta = 1$. $u(c_t) = log(c_t)$.
- Note que hay un número ∞ de restricciones, una para cada período. La solución debe cumplir algunas condiciones. Queremos escoger $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ que maximice el problema planteado.
- Sea $V(k_0)$ el valor óptimo del problema dado un k_0 inicial:

$$V(k_0) = \sum_{t=0}^{\infty} \beta^t u(c_t^*) \qquad s.t. \qquad f(k_t^*) \geq c_t^* + k_{t+1}^*$$

Donde $V(k_0)$ es la función valor, paralelo a la utilidad indirecta.

- Si optimizamos hoy suponiendo que mañana, dado k_{t+1} se optimiza correctamente, el nuevo problema para cualquier t es:

$$\max_{c_t, k_{t+1}} u(c_t) + \beta V(k_{t+1})$$

$$s.t. \qquad f(k_t) \geq k_{t+1} + c_t, \qquad k_t \ \ dado$$

- La solución debería ser igual a $V(k_t)$
- Lo anterior sugiere que la función debe satisfacer la ecuación de Bellman:

$$V(k) = \max_{c,k'} \left[u(c) + \beta V(k')\right] \qquad s.t. \quad f(k) \geq k' + c$$

- donde $k_t \longrightarrow k$, $k_{t+1} \longrightarrow k'$. La ecuación de Bellman nos permite pasar de un problema de ∞ períodos a uno de dos períodos.
- $v(k)$ función valor. k variable de estado. c, k' variables de control.

Objetivo de la Programación Dinámica

El objetivo es encontrar funciones de política (policy functions) que sean tiempo invariantes, en función de la/s variable/s de Estado:

$$c_t = h(k_t)$$
$$k_{t+1} = g(k_t)$$

tal que con un k_0 dado, genere una secuencia $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ que satisfaga

$$f(k_t) \geq k_{t+1} + c_t \qquad \forall t$$

y maximice la función objetivo U(c)

Las policy functions surgen endógenamente

- Note que por definición, g y h están en el problema de maximización

$$V(k) = \max_{c,k'} \left[u(\underbrace{c(k)}_{h(k)}) + \beta V(\underbrace{k'(k)}_{g(k)}) \right] \qquad s.t. \qquad \underbrace{f(k)}_{g(k)+h(k)} \geq k' + c$$

- Dos métodos de solución típicamente usados:
  1. Guess and Verify (Método de Coef. Indeterminados)
  2. Value Function Iteration (VFI, Enfoque Secuencial)
- Nos enfocaremos en el segundo y lo aplicaremos en Matlab.

Value Function Iteration (VFI)

- Se comienza con una candidata (first guess): $V^{(0)}(k')$
- Solucionamos el problema:

$$V^{(1)}(k) = \max_{c,k'} \left[u(c) + \beta V^{(0)}(k') \right] \qquad s.t. \quad f(k) \geq k' + c$$

- usando la solución del problema anterior, $c^{(0)}$ y $k^{(0)}$

$$V^{(1)}(k) = \max_{c,k'} \left[u(c^{(0)}) + \beta V^{(0)}(k'^{(0)}) \right] \qquad s.t. \quad f(k) = k'^{(0)} + c^{(0)}$$

Value Function Iteration

- Se continua el proceso con $V^{(1)}(k)$ y así sucesivamente hasta que las iteraciones converjan $V^{(i+1)}(k) \approx V^{(i)}(k)$

$$V^{(i+1)}(k) = \max_{c,k'} \left[u(c^{(i)}) + \beta V^{(i)}(k'^{(i)}) \right] \qquad s.t. \quad f(k) = k'^{(i)} + c^{(i)}$$

- El proceso de value iteration se basa en la existencia de un punto fijo.

VFI en la práctica

- Algoritmo:
  1. Crear una grilla para el capital de hoy de k.
  2. Fijar un criterio $\epsilon > 0$ de tolerancia. Por ejemplo, $\epsilon = 0.000001$.
  3. Partir con un guess para la función valor. Por ejemplo, $v^{(0)}(k) = 0 \quad \forall k$.
  4. Encontrar $v^{(1)}$ tal como fue mencionado. Note que c se puede reemplazar por $f(k) - k'$ para simplificar:
$$V^{(1)}(k) = \max_{0 \leq k' \leq k^{\alpha}} \left[u(k^{\alpha} - k') + \beta V^{(0)}(k') \right]$$
  5. Calculamos la diferencia entre $V^{(i+1)}$ y $V^{(i)}$ hasta que sea menor a ϵ. Cuando suceda, encontramos un punto fijo: $V^{(i+1)} = V^{*}$.
  6. La función de política será:
$$g(k) = arg \max_{0 \leq k' \leq k^{\alpha}} \left[u(k^{\alpha} - k') + \beta V^{*}(k') \right]$$

Aplicación 1: Modelo Neoclásico de Crecimiento

Para que sea más fácil, nos movemos al archivo `Guia_Modelo_Neoclasico_Deterministico.pdf` y al mfile `modelo_neoc_determ.m`

Aplicación 2: Modelo de Búsqueda

- El modelo de búsqueda de McCall ayudó a transformar la manera en la que economistas piensan sobre los mercados laborales.
- Para clarificar nociones como el desempeo "involuntario", McCall modeló el problema de un trabajador desempleado en términos de factores que incluyen:
  - Salario corriente y posibles salarios futuros
  - Impaciencia
  - Compensación por desempleo (bono de cesantía)
- Para solucionar el problema, se pueden usar técnicas de Programación Dinámica

El modelo

- Un desempleado recibe en cada período una oferta de trabajo con salario w_t
- La secuencia de ofertas $\{w_t\}_{t \geq 0}$ es i.i.d., con $q(w)$ la probabilidad de observar $w \in \mathcal{W}$.
- El agente observa w_t al inicio de t
- El agente sabe que $\{w_t\}$ es i.i.d. con distribución común q y puede usar esto para computar sus expectativas.

El modelo (cont.)

En el período t, el agente tiene dos opciones:

- Aceptar la oferta y trabajar permanentemente a un salario constante w_t
- Rechazar la oferta, recibir un bono de cesantía c y considerar la siguiente oferta en el próximo período

Los agentes viven infinitamente y buscan maximizar la utilidad esperada descontada de sus ingresos

$$\mathbb{E} \sum_{t=0}^{\infty} \beta^t y_t$$

$\beta \in (0, 1)$ es el factor de descuento. y_t es el ingreso, igual a w_t cuando está empleado o c cuando está desempleado

Trade-off

El trabajador enfrenta el siguiente trade-off:

- Esperar mucho a una oferta es costoso, dado que el futuro está descontado
- Aceptar muy pronto una oferta es costoso, porque mejores ofertas pueden llegar

Para decidir óptimamente, usamos programación dinámica. Es un procedimiento en dos etapas donde

1. primero asignamos valores a los "estados" y
2. después deducimos acciones óptimas dados esos valores

Función Valor

Tenemos que pensar en (i) los pagos corrientes que obtenemos de diferentes decisiones; (ii) los diferentes estados a los que esas decisiones nos llevarán en el siguiente período

$v^*(w)$ es el valor total de por vida de un trabajador desempleado que entra en el período actual desempleado cuando el salario es de $w \in \mathcal{W}$.

Más preciso, $v^*(w)$ es el valor de la función objetivo cuando un agente en esa situación toma decisiones óptimas hoy y en el futuro.

Obviamente, $v^*(w)$ no es trivial de calcular porque no conocemos cuáles son las decisiones óptimas

Pero piensen en v^* como una función que asigna a cada posible salario el valor máximo de vida que puede obtenerse con esa oferta en la mano

Ecuación de Bellman

v^* debe satisfacer

$$v^*(w) = max\left\{\frac{w}{1-\beta}, c + \beta \sum_{w' \in \mathcal{W}} v^*(w')q(w')\right\}$$

Esta es la **Ecuación de Bellman**.

El primer término es el valor esperado de por vida de aceptar la oferta hoy

El segundo término es el valor esperado de por vida de no aceptar la oferta hoy y comportarse de forma óptima en el futuro.

Política Óptima

Ahora tenemos que elegir correctamente el lado derecho

La acción óptima en general es pensada como una **política**. Dado cualquier w, las decisiones son aceptar (=1) o rechazar (=0). Podemos escribir la función de política como sigue:

$$\sigma(w) := \mathbb{1}\left\{\frac{w}{1-\beta} \geq c + \beta \sum_{w' \in \mathcal{W}} v^*(w')q(w')\right\}$$

Donde $\mathbb{1}\{P\} = 1$ si P es verdadero y 0 si es falso. Podemos re-escribir esto como:

$$\sigma(w) := \mathbb{1}\{w \geq \bar{w}\}$$

donde

$$\bar{w} := (1-\beta)\left\{c + \beta \sum_{w' \in \mathcal{W}} v^*(w')q(w')\right\}$$

$\bar{w}$, llamado salario de reserva. Entonces, el agente debería aceptar sólo si y solo si el salario ofrecido es mayor al salario de reserva.

Luego, podemos computar el salario de reserva si podemos computar la función valor.

Computando la Política Óptima

Para simplificar notación, definamos

$$W := \{w_1, ..., w_n\} \qquad \text{y} \qquad v^*(i) := v^*(w_i)$$

La función valor está representada entonces por $v^* = (v^*(i))_{i=1}^n$ Entonces,

$$v^*(i) = max \left\{ \frac{w(i)}{1-\beta}, c + \beta \sum_{1 \leq j \leq n} v^*(j) q(j) \right\} \qquad \forall i = 1, ..., n$$

El algoritmo

Usamos el método VFI ya discutido, seguimos los siguientes pasos:

1. Elegir un guess inicial $v \in \mathbb{R}^n$
2. Definir un criterio de tolerancia.
3. Computar un nuevo vector $v' \in \mathbb{R}^n$
4. Calcular la distancia entre v y v'
5. Si la distancia es mayor que la tolerancia definida, volver al paso 3 hasta encontrar una distancia menor al criterio de tolerancia

Si se cumple el último punto, encontramos un v que aproxima la función v^*

Ahora lo implementaremos en Matlab, con el código `modelo_search.m`, siguiendo el PDF `Guia_Modelo_Search.m`

Programación Dinámica Estocástica

Motivación

- Las economías están sujetas a distintos tipos de shocks. Modelaremos una forma fácil de incluirlos: agregar un shock de productividad a la función de producción. $y_t = z_t k_t^{\alpha}$.
- z_t sigue un proceso **estocástico** i.i.d con $z_t > 0$. Podría seguir otros procesos, por ejemplo, que su ln() siga un $AR(1)$.
- Distintas historias detrás de z_t: desastres naturales, guerras, descubrimientos tecnológicos, malas cosechas, etc.

- Antes (sin incertidumbre): dado k_0 decidimos $\{k_t\}_{t=1}^{\infty}$.
- Ahora (con incertidumbre respecto a z_t):

$$
\begin{aligned}
k_1 &= f_0(k_0, z_0) \\
k_2 &= f_1(k_0, z_1, z_0) \\
k_3 &= f_2(k_0, z_2, z_1, z_0) \\
\vdots &= \quad \vdots
\end{aligned}
$$

- La información disponible al decidir k_{t+1}, $(z_1, z_2, ..., z_t)$, no se conoce en $t = 0$. Por ende, el planificador elige una **colección de planes contingentes a cada posible historia de realizaciones de shocks.**
- Ahora las herramientas de programación dinámica son mucho más atractivas. Podemos modelar lo que puede suceder ante distintos estados de la naturaleza.

- La ecuación de Bellman ahora es:

$$v(k_t, z_t) = arg \max_{0 \leq k_{t+1} \leq z_t k_t^{\alpha}} [u(z_t k_t^{\alpha} - k_{t+1}) + \beta \mathbb{E}[v(k_{t+1}, z_{t+1})]]$$

  donde la esperanza es sobre los valores de z_t.
- (k_t, z_t) son las variables de estado. k_{t+1} variable de control.
- Eliminando índices de tiempo:

$$v(k, z) = arg \max_{0 \leq k' \leq z k^{\alpha}} \left[u(z k^{\alpha} - k') + \beta \mathbb{E}[v(k', z')]\right]$$

- Notar que ahora k es una variable aleatoria. Por lo tanto no convergerá como vimos antes debido a la presencia de shocks de productividad.
- Pero si $log(z)$ es un proceso estacionario, k oscilará en torno a un estado estacionario k^{EE}.

Aplicación: Modelo Neoclásico Estocástico de Crecimiento

Nos movemos al archivo `Modelo_Neoclasico_Estocastico.pdf` y al mfile `modelo_neoc_estoc.m`

Programación Dinámica: Reflexiones

- Cuándo las herramientas de Programación Dinámica son útiles?
  - Problemas de horizonte infinito, tanto determinísticos como estocásticos.
  - Problemas invariantes en el tiempo: misma función valor a ambos lados de la Ecuación de Bellman.
- Solucionando la Ecuación de Bellman
  - En la mayoría de los casos no podremos encontrar una solución analítica para la función valor. Los ejemplos vistos hasta ahora son casos especiales.
  - Generalmente se recurre a algoritmos númericos. Vimos estos algoritmos utilizando MATLAB.