Universidad Politécnica de Cartagena

GRADO EN ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS

CURSO ACADÉMICO 2016-2017

SIMULACIÓN PARA MINIMIZAR LOS COSTES DE UNA EMPRESA MEDIANTE EL SOFTWARE MATLAB

Autor

ALEJANDRO ORTEGA MOLINA

Directores

ROBERTO CAÑAVATE BERNAL

JUAN CARLOS TRILLO MOLLA

ALFONSO ESCUDERO SOLANO

Tabla de ilustraciones

Agradecimientos

Este proyecto es fruto del esfuerzo de muchas personas y no sólo de los que hemos participado activamente en él, sino también de los que han creído y lo han apoyado en todo momento. Quiero dar las gracias en primer lugar a mis tutores, por su infinita paciencia y toda su ayuda.

A Roberto, profesor de Matemáticas y de redacción, por todo el esfuerzo realizado y por haber hecho posible que este proyecto haya seguido adelante.

A Juan Carlos, que me ha enseñado prácticamente todo lo que sé de Matlab, por haber estado siempre dispuesto a echar una mano, haber sentado las bases de este proyecto y ser siempre tan paciente y comprensivo.

A Alfonso, tutor y amigo, por haberme introducido y guiado en este trabajo. Muchas gracias por la oportunidad, por haber creído en mí y por todo el tiempo y esfuerzo dedicado.

Esto tampoco habría sido posible sin el apoyo de mis padres, que me han apoyado incondicionalmente en mis decisiones y me han ayudado a seguir siempre adelante, no perdiendo nunca la confianza ni el interés en mí y en mi trabajo.

A Sami, por animarme siempre y por haber vivido este proyecto conmigo, ayudándome a pasar los malos momentos y mejorando los buenos.

1. Introducción

En este trabajo se pretende aplicar Matlab a la asignatura de Microeconomía del Grado en ADE de la UPCT. Matlab es un programa y a su vez un lenguaje de programación, que permite la creación de scripts[1]. Existen muchos tipos de lenguajes de programación, pero básicamente se pueden dividir en dos categorías: lenguajes interpretados[2] y lenguajes compilados[3], aunque eventualmente la mayoría de los lenguajes pueden ser compilados. Matlab es un lenguaje interpretado y un programa intérprete[4], que nos permite, además de crear los scripts, interpretarlos y crear "aplicaciones" que pueden ser instaladas en cualquier otro ordenador que posea Matlab. No obstante, Mathworks, empresa desarrolladora y responsable de Matlab, ofrece un compilador externo que permite convertir los ficheros Matlab a otros lenguajes de programación y crear aplicaciones independientes. Esto nos permite considerarlo como un entorno de desarrollo en toda regla.

Aprovechando estas características, se ha creado una aplicación en la que se pueden introducir los datos de unos determinados tipos de funciones de producción: las funciones Cobb-Douglas, Leontief y de Bienes Sustitutivos Perfectos, con datos W, V, α, β y q_0, siendo W y V los precios de los factores trabajo y capital respectivamente, α y β las elasticidades de los factores trabajo y capital respectivamente, y q_0 la cantidad producida. A partir de los datos introducidos, la aplicación proporciona los costes mínimos, la cantidad óptima a emplear de cada factor, K y L (capital y trabajo respectivamente), algunos datos adicionales (senda de expansión, RMST[5], productividad marginal de cada factor, costes medios, totales y marginales a largo plazo y el tipo de rendimientos de escala), así como representaciones gráficas del mapa de isocostes, rectas isocuantas, sendas de expansión y costes totales,

[1] Secuencia de comandos que debe ser interpretada por un programa para poder ejecutarse (Colburn, 2003).

[2] Lenguaje en el que se lee el script en orden comando por comando, ejecutando un comando antes de pasar al siguiente (Colburn, 2003).

[3] Lenguaje que requiere que el script sea convertido en lenguaje máquina antes de ser interpretado.

[4] Programa que lee y ejecuta el script comando a comando, siguiendo un orden establecido (Colburn, 2003).

[5] Relación Marginal de Sustitución Técnica. Disminución en la cantidad empleada de un factor productivocuando se utiliza una unidad extra de otro factor productivo, de manera que el volumen de producción permanece constante (Mas-Colell, Whinston and Green, 1995). En adelante, RMST.

medios y marginales a largo plazo. Cuenta también con un ejemplo, así como una validación de datos que impide que se introduzcan valores ilógicos en las casillas de la interfaz, como precios o cantidades negativas. Este proyecto puede resultar interesante tanto por tener un gran valor didáctico, permitiendo analizar el comportamiento de las funciones desde un punto de vista matemático, gráfico y visual, como por motivar la introducción y el uso de Matlab en la Facultad de Ciencias de la Empresa de la UPCT, algo que puede ser muy beneficioso para los alumnos por muchas razones: por una parte, porque les proporcionaría una base de conocimientos en programación, además de una plataforma con la que desarrollar aplicaciones que resuelvan diversos problemas relacionados con las Matemáticas y la Economía; por otra parte supondría un apoyo extra a su aprendizaje, al poder usar estas aplicaciones además para comprobar soluciones de problemas y ejercicios de clase. Sin embargo, es importante señalar que se trata de un software bastante complejo y difícil de dominar dada la gran cantidad de comandos y opciones que posee, aunque cabe destacar que, al tener un uso muy generalizado y extendido, cuenta con gran cantidad de contenido de soporte creado por otros usuarios y por la propia Mathworks, del cual el alumno puede extraer sugerencias, ideas y probablemente soluciones a los problemas que le surjan a la hora de programar. Es importante considerar que la empresa del presente y en mayor medida aún la del futuro, se basa en el uso y dominio de aplicaciones. Las aplicaciones automatizan y simplifican procesos, y los procesos simplificados y automatizados minimizan costes, lo cual es un objetivo básico de la empresa. Por eso es muy importante que los estudiantes estén familiarizados con las bases de estas aplicaciones y que tenga una idea básica de cómo funcionan y como se crean, ya que suponen un importante elemento diferenciador para las empresas, además de una importante herramienta didáctica para su formación.

La elección del software Matlab está motivada principalmente por tratarse de un programa muy extendido a nivel académico. Como se ha mencionado anteriormente, además de ofrecer funcionalidades avanzadas en el ámbito matemático permite la programación de aplicaciones, que es el objetivo principal del trabajo. Mathworks ofrece también un software para convertir aplicaciones de Matlab en aplicaciones independientes, lo que aporta interesantes perspectivas de futuro a este proyecto. Actualmente la UPCT cuenta con licencia de Matlab, lo que también ha influido en su elección frente a otros programas, como Mathematica.

2. Marco conceptual

2.1 Breve reseña histórica

La minimización de costes es un problema que afecta a la humanidad desde sus comienzos. Incluso antes de que el dinero existiera tal y como se conoce hoy, los avances tecnológicos y en métodos de producción siempre han ido enfocados a mejorar tanto la eficacia como la eficiencia de la producción. El auge de las ciudades en la Edad Media, y el uso extendido de la moneda pone la preocupación por la minimización de costes en primer plano, ya que la población empieza a concentrarse, el número de artesanos aumenta y con ello la competencia, lo que lleva a la creación de los primeros gremios, que no son sino el germen de la empresa moderna, una manera de organizar, entrenar y en cierta medida ofrecer acceso más sencillo a medios de producción y materiales a los productores. Sin embargo, no sería hasta mediados del siglo XIX cuando, de acuerdo con Pindyck y Rubinfeld (2009), nacerían las primeras empresas similares a las que hoy en día se conocen.

2.2 Teorías de producción

A la hora de producir, según Pindyck y Rubinfeld (2009), las empresas tienen que tomar tres decisiones esenciales, que son similares en cierto modo a las que señala George Stigler (qué, cómo y cuánto producir):

1. Cómo transformar los factores de producción, tomando como principales el trabajo, capital y materias primas, en productos, a lo que llama *tecnología de producción.*
2. La restricción de costes de la empresa, es decir, cuánto puede producir con el presupuesto del que dispone. Siguiendo un supuesto de racionalidad, la empresa siempre querrá producir el máximo posible que le permita este presupuesto.
3. Relacionado con lo anterior, la empresa tendrá que decidir, en función del precio de los factores, qué cantidad de cada uno empleará para producir con el menor coste.

En teoría de la producción, y también en este trabajo, se considera, como norma general y salvo que se indique lo contrario, que la empresa emplea dos factores: trabajo (L) y

capital (K), y que combinando estos factores puede conseguir un determinado nivel de producción, limitado por la disponibilidad de los mismos. Para representar estas diferentes combinaciones y niveles de producción se utiliza una función que relaciona los inputs con el output, q(K,L). Es importante señalar que, en el corto plazo, esta función sólo tendrá un factor variable, el trabajo, mientras que a largo plazo ambos factores son variables. Esto tiene una explicación lógica: a corto plazo una empresa, en términos generales, no puede contar con tener una fábrica más grande o una fábrica nueva, por ejemplo, ya que no dispone del tiempo de construirla y ponerla en funcionamiento. En cambio, sí puede contratar más trabajadores, pagar horas extras, añadir turnos o ampliar los existentes. Destaca una anotación que realizan Pindyck y Rubinfeld (2009) en este aspecto: el largo y corto plazo son conceptos variables, y pese a que generalmente se considera largo plazo el período superior a un año y corto plazo el inferior, este dependerá en gran medida del tipo de empresa y del ciclo de producción, ya que no será el mismo, por ejemplo, para un puesto de churros, cuyo largo plazo puede ser un mes, que para una gran multinacional que realiza enormes inversiones de tiempo y dinero, y cuyo largo plazo puede ser a partir de los 5 ó 10 años. Este proyecto se centra exclusivamente en la producción a largo plazo, y por ello este marco conceptual estará enfocado en la misma.

2.3 Funciones de producción

La función de producción es una expresión matemática que relaciona la cantidad empleada de factores de producción con la producción obtenida. En la asignatura de Microeconomía del grado en ADE de la UPCT es habitual tratar tres tipos de funciones de producción: la función Cobb-Douglas, la función Leontief y la función de Bienes Sustitutivos Perfectos, y como ya se mencionó anteriormente, este proyecto está centrado en ellas. Matemáticamente cada una tiene un tratamiento distinto, siendo la función Cobb-Douglas en cierto modo la más compleja y diferenciada de las tres, mientras que las funciones Leontief y de Bienes Sustitutivos Perfectos tienen un tratamiento similar. Antes de profundizar en las tres funciones, es importante revisar algunos conceptos importantes primero.

2.3.1 La isocuanta

La isocuanta de nivel q_0 de una función de producción q(K,L) es el conjunto de combinaciones de K y L que generan la producción dada q_0. La representación de la isocuanta

de nivel q_0 de una función de producción q(K,L) se puede obtener despejando uno de los factores productivos en la igualdad asociada q(K,L)=q_0, generalmente K. Por ejemplo, si la función de producción de una empresa es $q(L,K) = L + K$, y se despeja K, se obtiene la expresión $K = q_0 - L$. Suponiendo, por ejemplo, que q_0=1000, se puede representar la recta K=1000-L entre los puntos (1000,0) y (0,1000), dado que siempre se supone que K y L toman valores no negativos. Esta gráfica representa todas las combinaciones posibles de K y L a partir de las cuales se obtiene una producción de 1000 unidades con la tecnología dada.

2.3.1 La productividad marginal de los factores y la RMST

Cuando se habla de la productividad marginal de los factores, se distingue entre la productividad marginal del trabajo (L) y del capital (K). La productividad marginal del trabajo, PMg_L, indica el aumento que experimenta la producción si se aumenta en una unidad el factor trabajo y se mantiene constante el factor capital. Matemáticamente, ésta se obtiene realizando la derivada parcial de la función de producción con respecto a L. Por otro lado, la productividad marginal del capital, PMg_K, indica el aumento que experimenta la producción si se aumenta en una unidad el factor capital y se mantiene constante el factor trabajo. De manera similar al caso anterior, se obtiene realizando la derivada parcial de la función de producción con respecto a K.

La RMST, definida anteriormente, muestra la relación a la que un factor productivo puede ser sustituido por otro mientras se mantiene el nivel de producción. Se obtiene a partir del coeficiente de las productividades marginales de K y L. $RMST = -\frac{PMg_L}{PMg_K}$, aunque en este trabajo utilizaremos, como suele ser habitual, el valor absoluto de la RMST para obviar el signo negativo. De su expresión matemática puede apreciarse que la RMST se corresponde con la pendiente de la recta tangente a la isocuanta en un punto (L,K) dado.

2.3.2 Los rendimientos de escala

Según Pindyck y Rubinfeld (2009), los rendimientos de escala son la tasa a la que aumenta la producción cuando se incrementan los factores productivos en la misma proporción simultáneamente. Estos rendimientos se diferencian en tres tipos: crecientes, constantes y decrecientes. Cuando son crecientes el factor de incremento de la producción es mayor que el de los insumos; cuando son constantes, la producción se incrementa en el mismo factor de

proporción que los insumos; y si son decrecientes de escala, el factor de incremento de la producción es menor que el de incremento de los insumos. Por ejemplo, si se duplicaran los factores productivos, en caso de existir rendimientos crecientes de escala la producción sería más que duplicada, en el caso de ser constantes los rendimientos, quedaría duplicada, y en el caso de ser los rendimientos de escala decrecientes, la producción no llegaría a duplicarse.

2.4 Costes

Los costes pueden enfocarse desde diversas perspectivas, estableciéndose así una clasificación de los costes en tres tipos: coste de oportunidad, coste económico y coste contable.

- El coste contable comprende los gastos reales más los gastos de depreciación del equipo de capital (Pindyck y Rubinfeld, 2009). La evaluación de los costes desde esta perspectiva tiene como objetivo principal seguir la evolución de los activos y pasivos de la empresa e informar sobre los resultados pasados de la misma.
- El coste económico, a diferencia del contable, es una perspectiva en la que los costes se valoran pensando en el futuro, ya que se tienen en cuenta todos los costes de utilizar los factores de producción, incluido el coste de oportunidad. El objetivo en este caso no es informar, sino evaluar los costes que puede controlar la empresa y los que no, así como saber cuáles son sus posibles costes futuros, de cara a poder reorganizar la producción para reducirlos (Pindyck y Rubinfeld, 2009).
- El coste de oportunidad es el coste correspondiente a las oportunidades que pierde la empresa cuando no utiliza los recursos para el mejor fin alternativo. Como señalan Pindyck y Rubinfeld (2009), una empresa que posee un edificio de oficinas y por ello no paga alquiler, en realidad tiene que asumir un coste por emplear ese edificio que es, por ejemplo, la cantidad que podría obtener por arrendarlo. Este alquiler que se deja de ganar es un ejemplo perfecto de coste de oportunidad que, desde un punto de vista contable de costes, no existe.

En este proyecto se abordan los costes desde una perspectiva económica y no contable, por ello, cuando se haga referencia al coste será al coste económico en todo momento.

2.4.1 Minimización de costes y maximización del beneficio

Los costes totales de una empresa vienen dados por la expresión $CT(K,L) = WL + VK$, que es la suma del coste total de los factores trabajo y capital. Por otra parte, los ingresos totales se expresan como la multiplicación del precio del producto por la producción total, q_0. La diferencia entre los ingresos y los costes totales es el beneficio económico. En este proyecto se asume que la empresa desea producir una cantidad fija de producto final minimizando sus costes totales, ya que este es un objetivo común a la mayoría de las empresas.

2.4.2 La recta isocoste

De manera similar a la isocuanta mencionada anteriormente, una recta isocoste es el conjunto de todas las combinaciones posibles de trabajo y capital que pueden adquirirse con un coste total dado, C_0 (Pindyck y Rubinfeld, 2009). Dada una función de costes totales $CT(L,K) = WL + VK$, y un nivel de coste dado, C_0, se puede representar la recta isocoste correspondiente despejando uno de los factores productivos, generalmente K, en función de C_0 y del otro factor productivo, generalmente L. La expresión obtenida en este caso es $K = \frac{C_0}{V} - \frac{W}{V}L$, una recta de pendiente $-\frac{W}{V}$. Esta pendiente indica a su vez la proporción a la que se pueden sustituir ambos factores para mantener dicho nivel de costes, es decir, la empresa puede sustituir una unidad de factor trabajo por $\frac{W}{V}$ unidades de capital sin que varíe el coste total.

2.4.3 La senda de expansión

El punto de corte entre una isocuanta y una recta isocoste en el que ambas curvas tienen la misma pendiente se denomina punto de tangencia. La senda de expansión es el lugar geométrico de los puntos de tangencia que se generan cuando la empresa aumenta su producción manteniendo constantes los precios de los factores productivos.

2.4.4 La elección de la empresa

Es importante comprender cómo los conceptos de isocuanta e isocoste se relacionan con la elección de la empresa, es decir, qué cantidades de los factores L y K van a ser empleadas en la producción. Como el objetivo de la empresa es minimizar costes, siempre va a emplear las cantidades de L y K que le proporcionen la producción deseada a coste mínimo. El problema

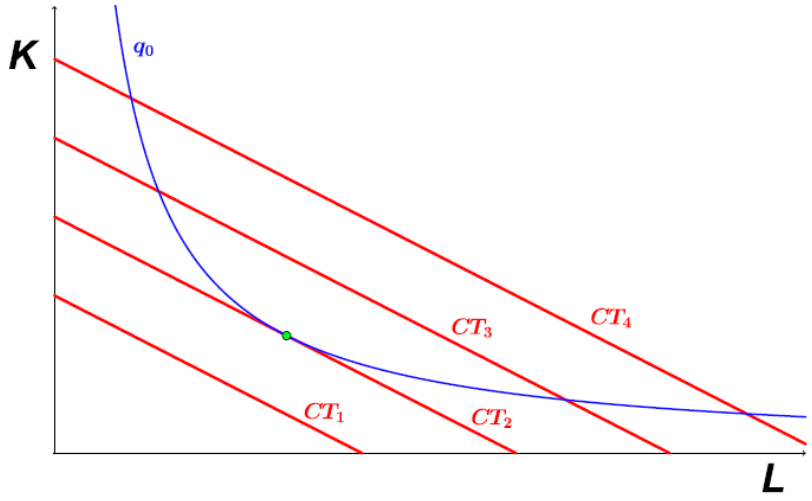

Imagen 1: Representación de la condición de tangencia en un mapa de isocostes.

surge a la hora de escoger el punto de la isocuanta, es decir, los valores K y L, en el que se consigue este mínimo coste. Para obtener este punto se igualan la RMST y el valor $\frac{W}{V}$, que son, respectivamente, la pendiente de la recta tangente a la isocuanta en un punto (L,K) y el coeficiente de precios, el cual a su vez se corresponde con la pendiente de la recta isocoste. A esto se lo conoce como condición de tangencia ya que el punto que se obtiene es el punto de tangencia mencionado anteriormente.

Para poder visualizar este procedimiento mejor se procede a mostrar un mapa de rectas isocostes, CT_1, CT_2, CT_3, CT_4, con una isocuanta de nivel q_0 dibujada sobre él. Las rectas isocostes que se sitúan más arriba representan niveles de costes mayores. El objetivo de la condición de tangencia es obtener el punto de tangencia, es decir, el punto en el que la isocuanta tiene la misma pendiente que la recta isocoste. En este punto, por tanto, la isocuanta corta con la recta isocoste situada más hacia abajo posible, y las siguientes por debajo representan niveles de costes imposibles para la producción dada.

El objetivo principal de la aplicación que se ha desarrollado en este proyecto es hallar esta condición de tangencia a partir de la cual podrán obtenerse todos los demás datos mencionados.

2.4.5 Costes de los factores productivos

Como ya se mencionó anteriormente, en este trabajo se considera que una empresa hace uso de dos factores productivos, trabajo (L) y capital (K). Estos factores tienen unos costes asociados a los que se denominarán W para el coste unitario del factor trabajo, y V para el coste unitario del factor capital. Un ejemplo del coste del factor trabajo puede ser, por

ejemplo, el salario por hora del operador de una grúa, mientras que un ejemplo del coste del factor capital es el coste por hora de alquilar dicha grúa.

2.4.6 Costes totales a largo plazo

La función de coste total a largo plazo $CT = CT(W, V, q)$ mide el coste total mínimo contraído por la empresa para un conjunto cualquiera de los precios de los factores y para un nivel cualquiera de producción (Nicholson and Cole, 2008). Esta función se obtiene de una manera diferente para cada una de las funciones de producción tratadas en este proyecto.

2.4.7 Costes medios y marginales

El coste medio es el resultado de dividir los costes totales de la empresa entre la cantidad producida, es decir, el coste de producción por unidad.

El coste marginal, también conocido como incremental, es el aumento que experimenta el coste cuando se produce una unidad más sobre el nivel de producción actual y los precios de los factores productivos permanecen invariables. Se obtiene derivando la función del coste total con respecto a la cantidad (q).

2.5 La función Cobb-Douglas

La función Cobb-Douglas fue probada y evidenciada estadísticamente por los economistas Charles Cobb y Paul Douglas en el año 1928, en el artículo "A theory of production". Esta función, aplicada al ámbito de este proyecto, representa dos bienes que se sustituyen a una ratio variable a los que se añade una constante A. La función tiene la siguiente forma

$$q(L, K) = AL^{\alpha}K^{\beta}$$

donde Q es la producción total, L y K son trabajo y capital respectivamente, α y β son las elasticidades de trabajo y capital respectivamente, y A es un factor de productividad que representa el progreso tecnológico. El resultado de sumar α y β indicará si los rendimientos de escala son crecientes, constantes o decrecientes: Si la suma es mayor que uno, serán crecientes, si igual a uno, constantes y si menor que uno, decrecientes, según demuestra Ramírez M. (1991).

La función Cobb-Douglas es homogénea de grado t ya que

$$q(\lambda L, \lambda K) = \lambda^t q(L, K), \forall \lambda > 0$$

dado que se cumple que

$$q(\lambda L, \lambda K) = \mathrm{A}(\lambda L^{\alpha})(\lambda K^{\beta}) = \lambda^{\alpha+\beta} A L^{\alpha} K^{\beta} = \lambda^{\alpha+\beta} q(L, K)$$

queda demostrado que la función Cobb-Douglas es homogénea de grado α+β, lo que implica que si, por ejemplo, se decide duplicar el input, el output quedaría multiplicado por $2^{\alpha+\beta}$. A continuación, se muestra un ejemplo gráfico de la isocuanta para el nivel de producción dado (verde), mapa de isocostes, isocoste en la que se cumple la condición de tangencia (rojo) y la senda de expansión (amarillo) en el eje (L,K). La imagen 2 se corresponde a la representación de los elementos citados anteriormente para la siguiente función Cobb-Douglas: $(L, K) = 10L^{0,4}K^{0,6}$, nivel de producción q=1000, empleando la aplicación Matlab que hemos desarrollado.

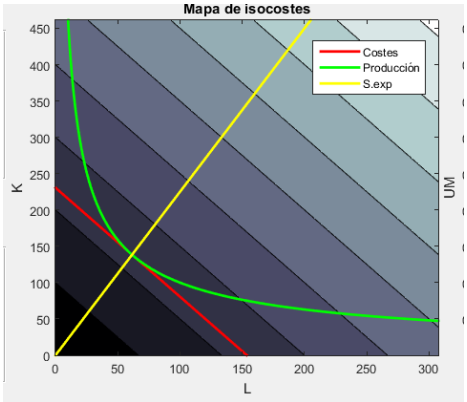

Imagen 2: Ejemplo de representación gráfica del mapa de isocoste, recta de isocoste que minimiza el coste total, isocuanta para la producción dada y senda de expansión de una función Cobb-Douglas.

En la función Cobb-Douglas, se puede obtener la expresión matemática de la curva de costes totales a largo plazo despejando la variable K en la ecuación de la condición de tangencia ($RMST = \frac{W}{V}$), obteniendo así K en función de L. Una vez obtenido K en función de L, se iguala la función de producción a q quedando $q(L, K) = AL^{\alpha}K^{\beta} = q$. Se sustituye K en la

misma y se despeja L, obteniendo una expresión de L en función de q. Sustituyendo la ecuación que se acaba de obtener en la primera expresión de K en función de L, se obtendrá K en función de q. Una vez se tienen K y L en función de q, al sustituirlas en la función de costes se obtendrá una expresión de los costes en función de la producción q y el precio de los factores de producción $CT = CT(q, V, W)$. La expresión obtenida sólo será lineal en el caso de los rendimientos constantes a escala, y será una curva cóncava o convexa en función del tipo de rendimiento a escala. La aplicación creada en este proyecto es capaz de representar estas curvas, como se muestra en las imagen 3 sobre los ejes (q,CT(Unidades Monetarias)), para a partir de ella obtener las funciones de costes medios y marginales a largo plazo. Nuestra aplicación también es capaz de representarlos, tal y como se muestra en la imagen 4.

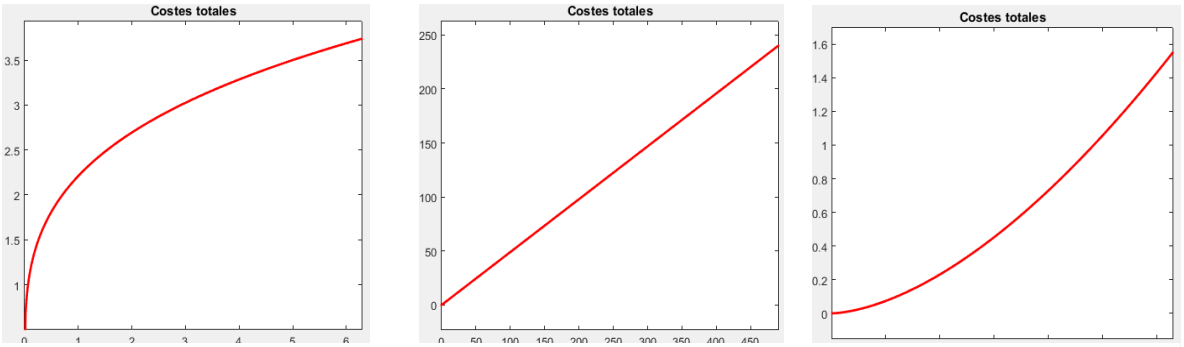

Imagen 3: Representación de los costes totales a largo plazo para tres empresas cuya tecnología de producción es Cobb-Douglas. De izquierda a derecha, empresa con rendimientos crecientes, constantes y decrecientes a escala.

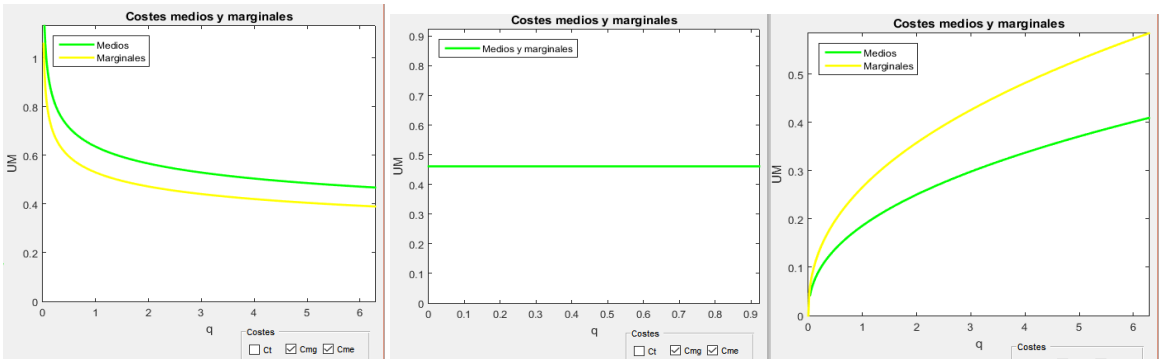

Imagen 4: Representación de los costes medios y marginales a largo plazo para tres empresas cuya tecnología de producción es Cobb-Douglas. De izquierda a derecha, empresa con rendimientos crecientes, constantes y decrecientes de escala.

2.6 La función de Bienes Sustitutivos Perfectos

La segunda función a tratar en este proyecto es la de Bienes Sustitutivos Perfectos. Representa una producción sujeta a dos insumos que son indiferentes desde el punto de vista productivo para la empresa. La forma de esta función es la siguiente

$$q(L,K) = \alpha L + \beta K$$

donde L y K son trabajo y capital respectivamente, y α y β son las ratios de sustitución de cada uno. Al tratarse de una función lineal, es una función homogénea de grado 1. Puede demostrarse siguiendo un procedimiento similar al empleado en la función Cobb-Douglas

$$q(\lambda L, \lambda K) = \lambda^t q(L,K), \forall \lambda > 0$$

Dado que se cumple que

$$q(\lambda L, \lambda K) = \alpha\lambda L + \beta\lambda K = \lambda^1\, q(L,K)$$

se trata de una función homogénea de grado 1, es decir, siempre ofrece rendimientos de escala constantes. Esto significa que, si se multiplican los inputs por un número λ, la producción quedará multiplicada por λ^1, esto es, será multiplicada por λ.

La elección de la empresa en este caso muestra tres casos particulares. Como se ha mencionado en apartados anteriores, la elección de la empresa viene dada por la condición de tangencia. En este tipo de funciones de producción se alcanza una solución esquina o infinitas soluciones, dependiendo de la relación entre la RMST y el coeficiente de precios de

los factores de producción $\frac{W}{V}$. Cuando la RMST sea mayor que el coeficiente de precios, se producirá empleando únicamente el factor L, cuando sea menor, únicamente el factor K, y si ambos son iguales existirán infinitas soluciones sujetas a la isocuanta para el nivel de producción dado.

En las imagen 6 se muestra un ejemplo gráfico de la isocuanta para el nivel de producción dado (verde), mapa de isocostes y recta isocoste en la que se minimizan los costes para la producción dada (rojo) en el eje (L,K), en funciones de Bienes Sustitutivos Perfectos con RMST menor, mayor e igual al coeficiente de precios.

Los costes totales a largo plazo para este tipo de función se obtiene igualando la función de producción a q_0, $q(L,K) = \alpha L + \beta K = q_0$, despejando la variable correspondiente según la relación entre la RMST y el coeficiente de precios, considerando 0 el valor de la otra variable (solución esquina) y sustituyendo en la función de costes $CT(L,K) = WL + VK$. Si RMST>$\frac{W}{V}$, se produce empleando únicamente L, y los costes totales a largo plazo serán $CT(q,W,V) = W\left(\frac{q_0}{\alpha}\right)$. En el caso de RMST<$\frac{W}{V}$, se produce empleando únicamente el factor K, y siguiendo el procedimiento descrito se obtiene que $CT(q,W,V) = V\left(\frac{q_0}{\beta}\right)$. Finalmente, si RMST = $\frac{W}{V}$, la isocuanta y la recta de isocoste que minimiza el coste para la producción dada tienen la misma pendiente en todo momento, por lo que existirán infinitas combinaciones posibles de K y L sujetas a $CT(L,K) = WL + VK = C_0$. Los costes medios y marginales se obtendrán del mismo modo que en la función Cobb-Douglas, y coincidirán en este tipo de funciones. En la imagen 5 se muestra una representación gráfica de los costes totales, medios y marginales a largo plazo para la siguiente función de Bienes Sustitutivos Perfectos

$$q(L,K) = 3L + 5K$$

$$sujeto\ a\ CT(L,K) = 2L + 3K, nivel\ de\ producción\ q_0 = 1000$$

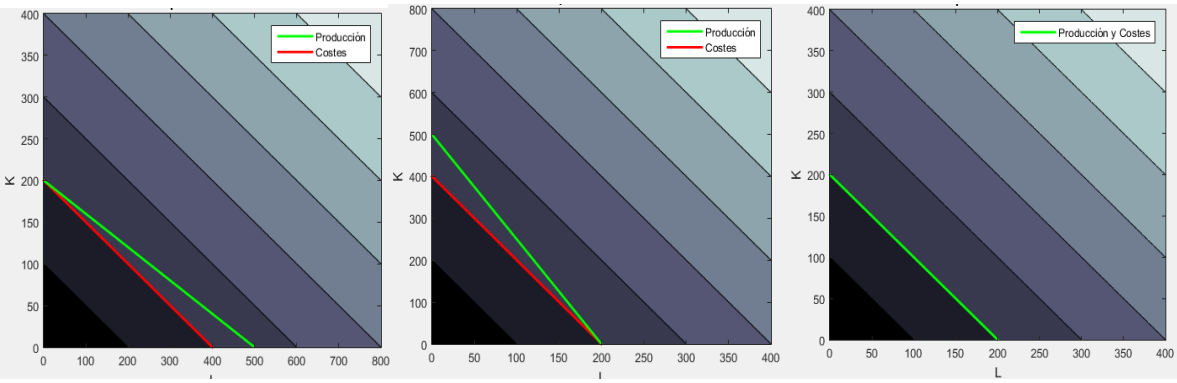

Imagen 5: Ejemplo de representación gráfica del mapa de isocoste, recta de isocoste que minimiza el coste total e isocuanta para la producción dada de tres funciones de producción de Bienes Sustitutivos Perfectos con RMST menor, mayor e igual al coeficiente de precios.

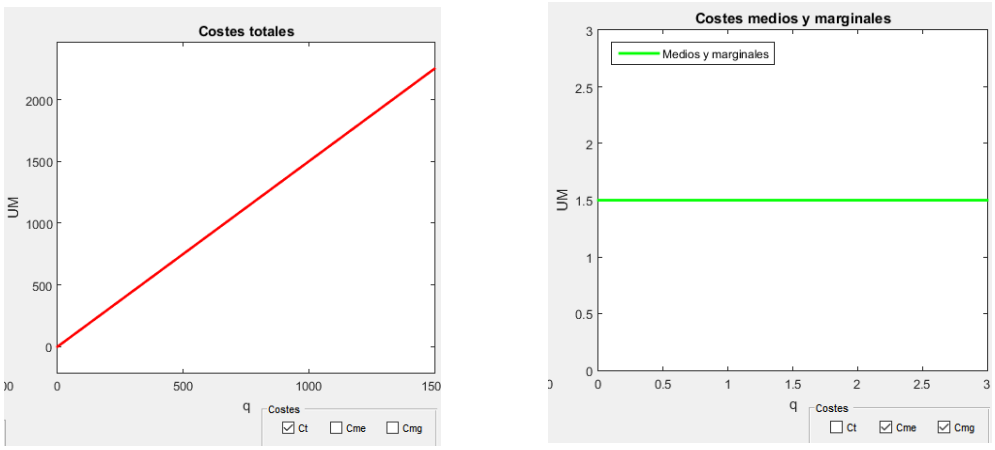

Imagen 6: Representación de los costes totales a largo plazo (izquierda) medios y marginales (derecha) a largo plazo para una empresa cuya tecnología de producción es la de Bienes Sustitutivos Perfectos.

2.7 La función Leontief

La última función a tratar es la función Leontief, que relaciona dos inputs que son complementarios, es decir, se requiere una determinada combinación de inputs para producir una unidad de Q. Un ejemplo de este caso sería una maquinaria pesada que requiere dos trabajadores para funcionar, o lo que es lo mismo, dos trabajadores por cada máquina o una máquina por cada dos trabajadores. La función Leontief, por lo tanto, tiene la siguiente forma

$$q(L,K) = \min\{\alpha L, \beta K\}$$

donde Q es la cantidad producida, L y K son los factores trabajo y capital respectivamente y α y β la cantidad de L que se necesita por cada unidad de K y viceversa.

Los rendimientos de escala en este tipo de función, al igual que en la de Bienes Sustitutivos Perfectos, son constantes. Para ello comprobaremos que esta función de producción es homogénea, esto es, que se cumple

$$q(\lambda L, \lambda K) = \lambda^t q(L,K), \forall \lambda > 0$$

con t=1. Efectivamente, esto es lo que ocurre, tal y como se muestra a continuación:

$$q(\lambda L, \lambda K) = \min\{\alpha\lambda L, \beta\lambda K\} = \lambda^1\, q(L,K)$$

Dicen Nicholson y Cole (2008) que en la función Leontief se sabe que la producción tendrá lugar en el vértice de las isocuantas en forma de L donde $q = \alpha L = \beta K$. Despejando L y K en función de q, se tiene que $L = \frac{q}{\alpha}$ y $K = \frac{q}{\beta}$. Sustituyendo estas expresiones en la función de costes $CT(L,K) = WL + VK$ se tiene que $CT(V,W,q) = W\frac{q}{\alpha} + V\frac{q}{\beta}$, y simplificando se obtiene que la función de costes totales a largo plazo para una función de producción Leontief es $CT(W,V,q) = q\left(\frac{V}{\alpha} + \frac{W}{\beta}\right)$. En la imagen 7 se muestra un ejemplo gráfico de la isocuanta para el nivel de producción dado (amarillo), mapa de isocostes, recta isocoste en la que se minimizan los costes para la producción dada (rojo) y senda de expansión (verde) en el eje (L,K), para la siguiente función Leontief

$$Q(L,K) = \min\{50L, 50k\}$$

$$sujeto\ a\ CT(L,K) = 2L + K$$

Los costes medios y marginales, que coincidirán, se calculan del mismo modo que en los casos anteriores. Nuestra aplicación representa estos costes como se muestra en la imagen 8.

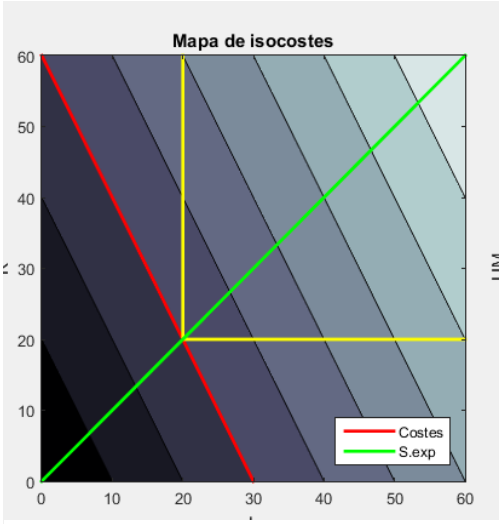

Imagen 7: Ejemplo de representación gráfica del mapa de isocoste, recta de isocoste que minimiza el coste total e isocuanta para la producción dada de una función Leontief

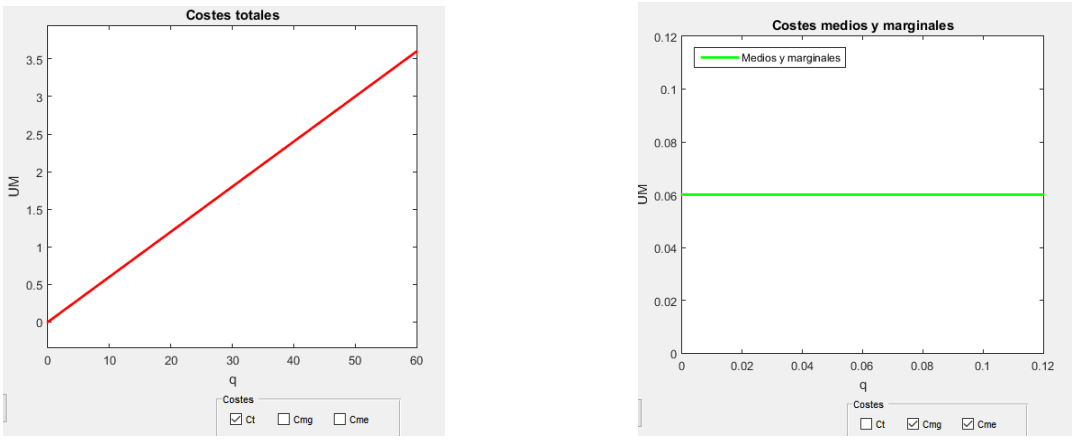

Imagen 8: Representación de los costes totales a largo plazo (izquierda) medios y marginales (derecha) a largo plazo para una empresa cuya tecnología de producción es la de Leontief.

3. Desarrollo de la aplicación

El desarrollo de la aplicación se divide en tres fases principales: creación del script principal, dibujo de la interfaz y desarrollo del script de la interfaz.

3.1 Creación del script principal

Como se ha mencionado anteriormente, en este proyecto se va a trabajar con tres tipos de funciones de costes a largo plazo: la función Cobb-Douglas, la función de Bienes Sustitutivos Perfectos y la de bienes complementarios, también conocida como Leontief. Estas funciones se basan en los mismos fundamentos, pero requieren un tratamiento distinto en el programa, especialmente la función de Cobb-Douglas, la más difícil de tratar, ya que, al no ser una ecuación lineal, en ocasiones el programa puede hallar soluciones que implican números complejos, las cuales no tienen sentido en el marco de la Microeconomía, por lo que habrá que forzar al programa a elegir siempre una solución en el entorno real y positivo. La función de Bienes Sustitutivos Perfectos presenta tres casos diferentes en función de la relación entre la RMST y el coeficiente de precios y hay que conseguir que el programa distinga cada uno. La función Leontief es quizás la más sencilla de programar ya que carece de casos particulares y soluciones complejas, aunque las dificultades surgen a la hora de dibujar el gráfico, ya que éste es significativamente más complejo que los anteriores y requiere diseñar una configuración específica usando la función *plot* que ofrece Matlab. No obstante, para profundizar más en el uso del programa y sus posibilidades, se ha añadido a esta función la opción de maximizar la producción a partir de unos costes establecidos. Por estos motivos, el script principal consta de tres archivos con extensión *.m* en los que se recogen todas las funciones y secuencias de comandos. Estos archivos son los que se usan para crear una aplicación de Matlab, y son los que el programa emplea para realizar los cálculos. Se componen de tres partes esenciales cada uno: definición de la función, formulación y diseño, y configuración de los gráficos.

A la hora de definir la función, se tienen en cuenta tres factores muy importantes, como son las variables de entrada, las de salida y el nombre de la función.

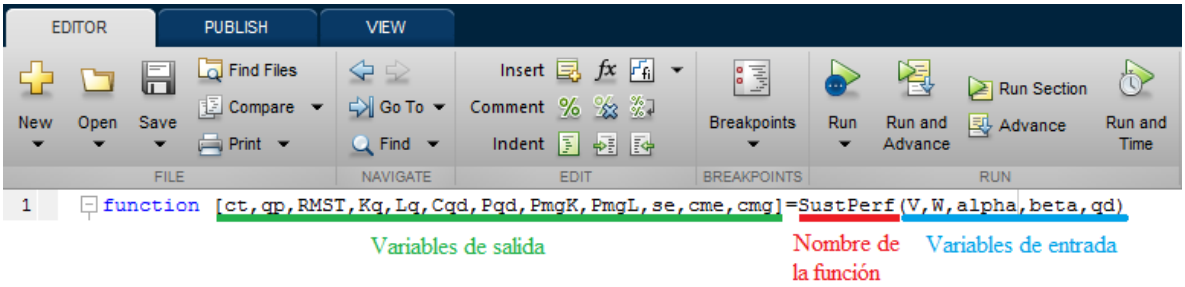

Imagen 9: Elementos a definir en una función de Matlab.

- Las variables de entrada son aquellas que se conocen previamente y que se emplearán en el programa para obtener los datos finales. Es importante definir bien estas variables, ya que serán las que se introduzcan posteriormente en la interfaz a fin de ser empleadas por el programa para realizar los cálculos.

- Las variables de salida son aquellos resultados que se obtienen a partir de los datos ya conocidos

- El nombre de la función deberá coincidir con el nombre del fichero *.m* donde se guarde, si no el programa mostrará el error *Undefined function* (función no definida). Matlab guarda los ficheros por defecto con el mismo nombre que la función principal.

Se pueden añadir además variables simbólicas. Éstas son de ayuda especialmente a la hora de crear gráficos, ecuaciones, derivar e integrar. Por último, se pueden crear objetos y otorgarles propiedades y características, o incluso asociarles valores y funciones. Son realmente útiles, y serán las principales herramientas a la hora de crear el script, ya que permiten desde asociar unas propiedades a un gráfico hasta crear órdenes de ejecución y condicionales. Para ilustrar esto procedemos a presentar un ejemplo: se va a crear un objeto

llamado *Grafico*, al que será asignada una función (imagen 10). A partir de este objeto Matlab generará un gráfico empleando su configuración predeterminada.

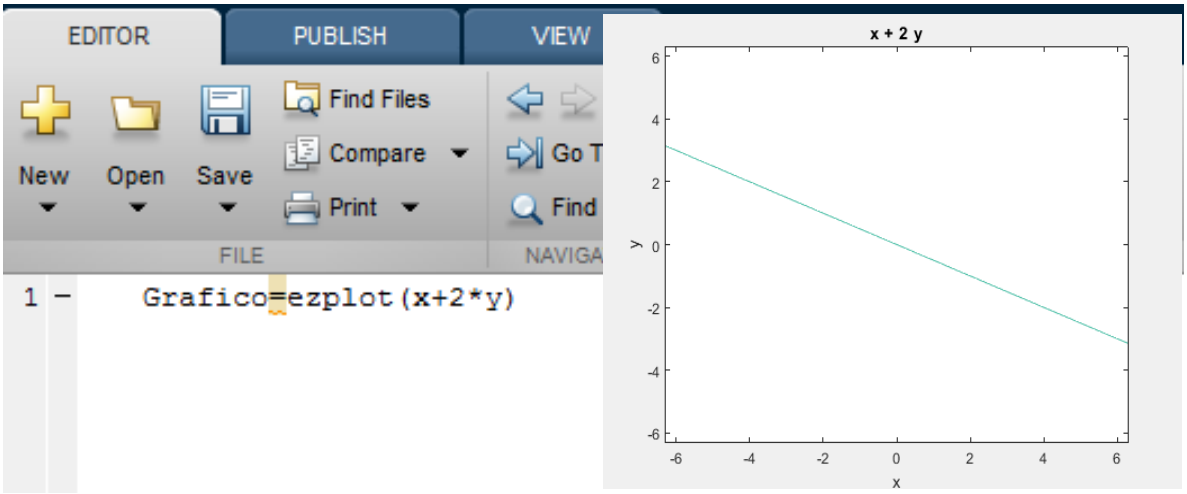

Imagen 10: Asignación de elementos a un objeto.

A continuación, usando el comando *set*, se asignan unas propiedades a *Grafico*. En este ejemplo se cambia el color de línea y el grosor:

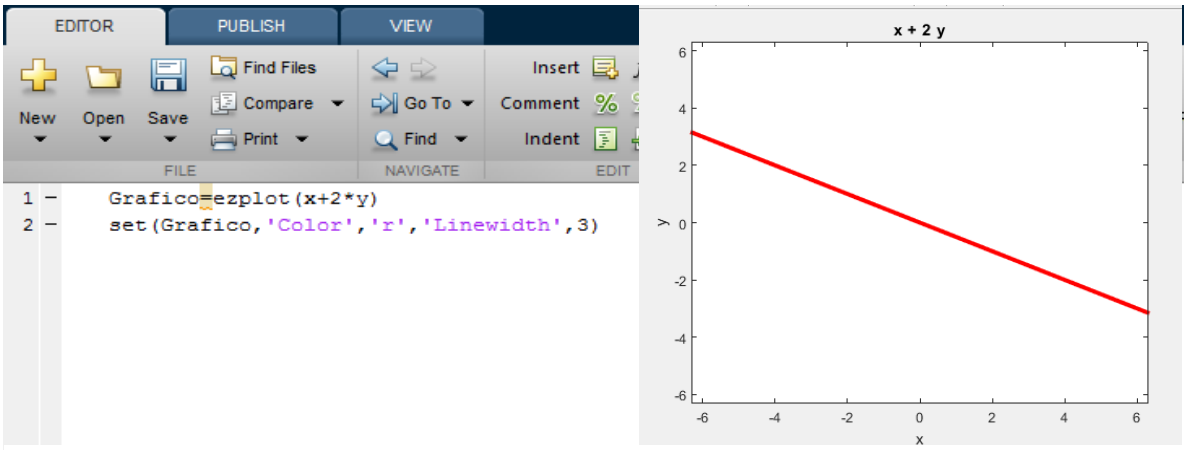

Imagen 11: Asignación de propiedades a un objeto.

Estos objetos permiten además la creación de cadenas de comandos algo más complejos, que serán muy útiles, sobre todo a la hora de crear la interfaz.

Para la formulación y diseño se han empleado las siguientes funciones de Matlab:

- *diff*: Sirve para realizar derivadas de distinto grado. El comando tiene la siguiente forma *diff(Variable,grado,dimension)*. Necesario para calcular la RMST y las

productividades marginales, permitiendo realizar derivadas parciales indicando la/s variable/s a derivar.

- *solve*: Resuelve una ecuación. Su forma es *solve(ecuación,variable a despejar,nombre 1,valor 1,…,nombre n, valor n,condición 1,valor condición 1,…, condición m, valor de condición m)*. Lo único obligatorio a especificar en esta función es la ecuación, de no escribir el resto, Matlab lo determinará automáticamente. Los apartados de nombre y valor sirven para asignar valores a distintas variables en ecuaciones que cuenten con más de una. Por ejemplo, en la ecuación 2x+3y+z, si se escribe solve(2*x+3*y+z, x , y , 1 , z, 2) , el programa calculará el valor de X para esa ecuación asumiendo que y=1 y z=2. En las condiciones puede establecerse que se devuelvan, por ejemplo, soluciones reales, complejas, o mayores que un determinado número. Tomando como ejemplo la ecuación $x^2 - 1 = 0$, si se quisiera resolver en Matlab y obtener solo valores positivos y reales para x se emplearía el comando *solve* de la siguiente manera: *solve((x^2)-1,x>0,'Real',true)*. En este caso se muestran dos condiciones, la primera que no es de tipo lógico, por lo que no se le añade un valor *true* o *false* y la segunda, que sí lo es. Habrá que añadir un *true* si se quieren obtener únicamente soluciones reales, o un *false* si se quieren obtener todas las soluciones posibles.

- *double*: De acuerdo con Mathworks (2016), convierte un resultado a doble precisión. El fundamento teórico es que Matlab almacena por defecto los valores numéricos en formato 32 bits, lo que permite realizar cálculos con potencias de hasta 2^{32} y 2^{-32}, mientras que usando el formato 64 bits, las posibilidades se amplían hasta 2^{64} y 2^{-64} (Petzold, 1999).

- *assume*: Obliga al programa a asumir unas condiciones determinadas por el usuario para una variable simbólica. Indispensable cuando se quiere que la solución de una ecuación que implica variables simbólicas se encuentre en un determinado rango de valores, y mucho más efectivo cuando además se combina con las condiciones de *solve*. Su forma general es la siguiente *assume(expresión/variable,condición 1,valor 1,…,condición n,valor n)*. Las condiciones y los valores se comportan de manera similar a los de *solve.*

- Operadores básicos: "+", "-", "^" y "/". Matlab permite realizar las operaciones básicas bien usando los signos mostrados o usando funciones específicas, como *sum*, *minus* o *product*, aunque estas últimas generalmente son más útiles a la hora de operar con vectores o matrices.

- %: Sirve para añadir un comentario. Útil a la hora de añadir descripciones o anotaciones. Añadir *%%* seguido de un espacio crea una sección, que quedará resaltada sobre lo demás cuando sea seleccionada, y se extenderá hasta el siguiente *%%*. Dividir el script en secciones permite organizarlo, ya que Matlab ofrece la posibilidad de acceder directamente a una sección en lugar de buscar manualmente la parte del código en cuestión. A la hora de depurar[6], el programa permite ejecutar una sección del código en concreto, lo que en ocasiones puede ahorrar bastante tiempo.

- *if, elseif, else*: Permite establecer condiciones. Con *if* se establece la condición principal y su consecuencia. *elseif* sirve para añadir otro par condición/consecuencia adicional, y con *else* se establece una condición y su respectiva consecuencia que sólo se ejecuta cuando no se cumple ninguna de las establecidas previamente. Se debe tener en cuenta que un comando de este tipo tiene un sólo *if*, puede tener ningún, uno o varios *elseif* y a lo sumo un *else*. Se debe tener en cuenta que un comando de este tipo tiene un sólo *if*, puede tener ningún, uno o varios *elseif* y a lo sumo un *else.* Se debe tener en cuenta que un comando de este tipo tiene un solo *if*, puede tener ningún, uno o varios *elseif* y a lo sumo un *else*. A la hora de crear una condición en Matlab, puede hacerse únicamente con *if*, siendo los otros dos elementos únicamente imprescindibles cuando se quiere distinguir entre diversos casos o que el programa tome un rumbo de ejecución específico. Siempre será necesario escribir *end* al final de un *if, elseif, else* para indicarle al programa cuando acaba, o de lo contrario interpretará todo lo que se siga escribiendo como parte de las condiciones/consecuencias.

[6] Procedimiento que consiste en la identificación y reparación de errores en el script o código fuente, ya sean errores de sintaxis o lógicos, así como la eliminación de líneas innecesarias y/o la optimización de las existentes. Conocido también por el término inglés *Debug* o por la españolización informal de este término, *Debuggear*.

Las funciones empleadas en la configuración de los gráficos son:

- *plot*: Es la herramienta por defecto para diseñar gráficos de Matlab, y la que permite el mayor nivel de personalización y configuración. Su principal ventaja es que permite crear prácticamente cualquier forma por compleja que sea, aunque, por otro lado, su programación es más compleja. Imprescindible para crear el gráfico de la función Leontieff.

- *ezplot*: Es una abreviatura de "easy plot", y permite crear un gráfico con parámetros elegidos automáticamente por el programa. Es ideal a la hora de representar rectas, curvas y otros gráficos sencillos, ya que requiere poca configuración y su programación es sencilla, pero por otro lado presenta serias limitaciones cuando se quieren representar figuras algo más complejas. Este comando se representa de la siguiente manera: *ezplot(función, [min x max x min y max y], figura en la que queremos representarlo).*

- *ezcontour*: Crea un contorno, especialmente útil a la hora de crear mapas de curvas. La sintaxis es prácticamente la misma que la de *ezplo*t.

- *ezcontourf*: Crea contornos al igual que *ezcontour*, pero los rellena de color. La sintaxis es la misma que la de *ezcontour*. Además, usando el comando *colormap* se puede ajustar el mapa de colores y la forma en que varían los colores en función de los valores máximos y mínimos de la función, como, por ejemplo, que los valores más próximos al mínimo sean de azul oscuro intenso y que se hagan más claros cuanto más se aproximen al máximo.

- *legend*: Permite asignar una leyenda a un gráfico. La forma de este comando es: *legend([objeto1,objeto2…objeto n], 'nombre1','nombre2'…'nombre n').*

- *hold on/off*: Al crear un gráfico, se crea una figura con ejes como, por ejemplo, la imagen 11. Si se quisiera añadir otro gráfico más a dicha figura sobre los mismos ejes se emplearía el comando *hold on* tras definir el primer gráfico. Todos los gráficos definidos a continuación se irán añadiendo a la misma figura sin cambiar los ejes

hasta que se añada un *hold off*. Después de añadir este comando el siguiente gráfico que se defina creará una figura y unos ejes nuevos.

- *xlim/ylim*: establece unos límites superiores para los ejes x e y de una figura, respectivamente.

3.2 Dibujo de la interfaz

El software Matlab cuenta con su propia herramienta para la creación de interfaces de usuario, llamada GUIDE. Esta se puede ejecutar desde el menú principal, haciendo click en *new* y seleccionando *Graphical User Interface* o escribiendo *guide* en la ventana de comandos. Se abre entonces una ventana con una plantilla cuadriculada en la que se pueden dibujar y posicionar los distintos elementos de la interfaz. Estos elementos disponibles se encuentran en la columna de la izquierda. Crear una interfaz genera dos archivos con el mismo nombre pero diferente formato: un archivo *.fig*, que contiene la plantilla de la interfaz y un archivo *.m*, que se actualiza automáticamente cada vez que se añaden elementos a la plantilla y se guardan los cambios.

Los elementos empleados en el dibujo de la interfaz son los siguientes:

- Edit text: Se puede identificar generalmente como un cuadro de texto con el fondo blanco (cuando esta activo) o gris (cuando se desactiva). Su principal diferencia con el texto estático es que permite su edición directa desde la interfaz de usuario, mientras que el texto estático solo puede ser editado o bien desde GUIDE, o bien desde el script de la interfaz. La función principal que cumple en este proyecto es la entrada y salida de datos, por los motivos explicados anteriormente. En adelante será llamado *edit*.

- Static text: Su función principal es nombrar otros elementos del panel, aunque también puede emplearse para la salida de datos. Una ventaja destacable que ofrece sobre un elemento *edit* es la protección que ofrece a los datos, ya que no podrán editarse de ningún modo si no es desde GUIDE o desde el script. En este proyecto, resulta realmente útil cuando los datos de salida tienen forma de texto, por ejemplo, a la hora de indicar los rendimientos a escala.

- Panel: Su función principal en este proyecto es agrupar elementos de la interfaz. Los elementos agrupados en un panel quedan vinculados a él, por lo que si se hace invisible o móvil al panel, los elementos en su interior también lo serán, esto es, se moverán con éste y aparecerán y desaparecerán a la vez.

- Checkbox y Radio buttons: A priori ambos elementos pueden parecer idénticos, y su elección depender de una preferencia estética, e incluso Matlab no nos aporta ninguna indicación sobre el correcto uso de cada uno de los elementos. Sin embargo, de acuerdo con reconocidos desarrolladores web y de interfaces de usuario como (Microsoft, 2016), cada uno se empleará en un caso determinado. Según (Nielsen, 2004) «Usaremos Radio buttons cuando hay una lista de dos o más opciones que son mutuamente exclusivas, y Checkbox cuando exista una serie de opciones y el usuario tenga la opción de seleccionar cualquier número de ellas o ninguna». En este proyecto se han seguido estas directrices a la hora de elegir entre ambas figuras.

- Axes: La función principal de este elemento es recoger y realizar representaciones gráficas, pero también puede usarse para implementar imágenes, siempre y cuando estas se encuentren en la misma carpeta que el archivo *.m*.

- Push button: Se trata de un botón que ejecuta un procedimiento al pulsarlo, desde realizar operaciones y cálculos hasta resetear la interfaz o abrir ficheros externos.

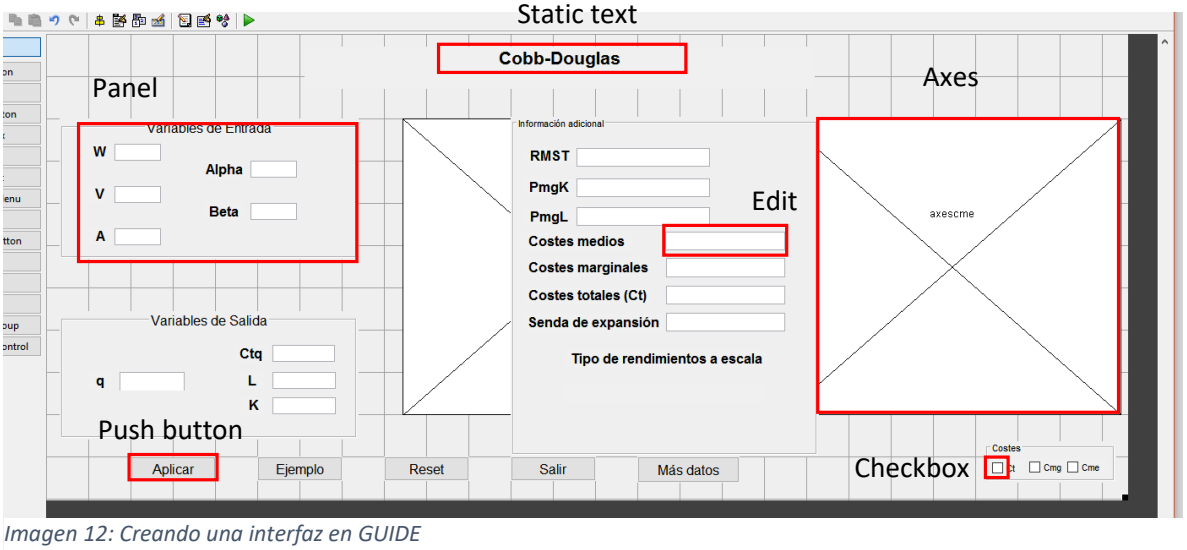

Imagen 12: Creando una interfaz en GUIDE

3.3 Desarrollo del script de la interfaz

Como se ha mencionado anteriormente, una UI (*User Interface* o Interfaz de Usuario) de Matlab cuenta con dos ficheros: uno de formato *.fig*, que contiene el diseño o "dibujo" de la interfaz, y uno de formato *.m* vinculado a éste, que contiene el script asociado, es decir, las órdenes que se ejecutan al pulsar los botones, la vinculación de los *edits* a los scripts de cálculo y la configuración de los ejes, entre otros. Esta es quizás la parte más compleja de todo el proyecto, ya que se compone de gran cantidad de elementos interdependientes, así como muchísimos detalles que pese a poder parecer insignificantes, pueden causar errores críticos que impidan la resolución de los cálculos. La interfaz y el script interactúan a través del *callback*, que es una función que se crea automáticamente en el fichero *.m* al dibujar un elemento en el archivo *.fig* y guardar los cambios. Todos los comandos que se escriben en un *callback* se ejecutan al activar el elemento con el que está asociado. Así mismo, si se quiere asignar propiedades, como por ejemplo visibilidad o color, habrá de hacerse en este espacio también. Un elemento importante del *callback* es el *tag* o etiqueta, y que es el "nombre" del elemento dentro del script, lo que significa que siempre que se quiera hacer referencia a éste, debe hacerse empleando este *tag*. Matlab asigna un *tag* por defecto a cada elemento, por ejemplo, a los elementos *edit* los denominará *edit1*, *edit2*,…, *editn*, por lo que es muy conveniente a la hora de poder organizarse en la programación sustituir este *tag* por uno personalizado e identificable por el programador. Esto puede hacerse desde el inspector de propiedades

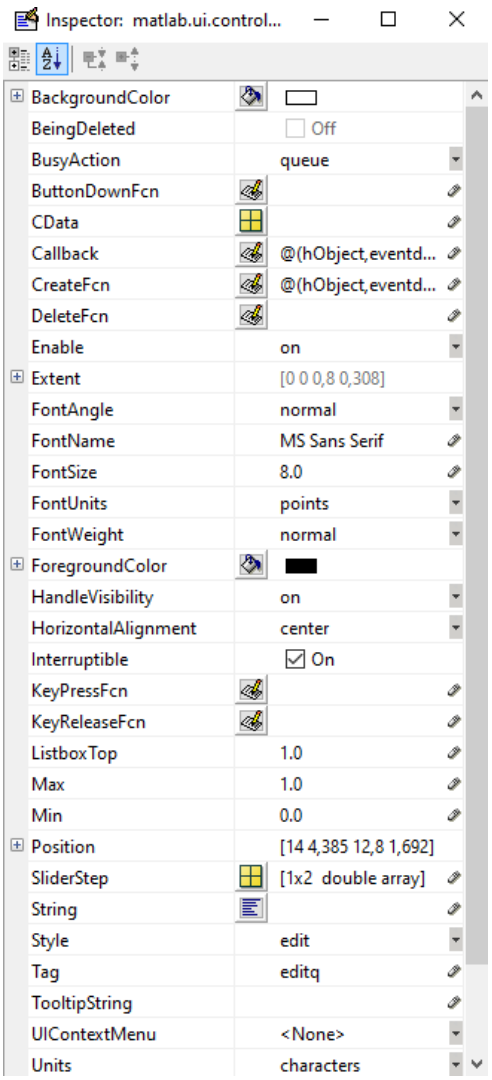

Imagen 13: Inspector de propiedades

(*property inspector*), un menú con el que es especialmente importante familiarizarse ya que servirá de guía en la creación del script.

En la imagen 13 se muestra el inspector de propiedades del *edit* que recoge los datos de *q*. Todas estas propiedades se pueden editar desde este inspector o usando el comando *set* de la siguiente forma: *set(handles.etiqueta,'propiedad','valor')*. Por ejemplo, si se quisiera desactivar este *edit* para que los datos no puedan ser editados, se escribiría de la siguiente forma: *set(handles.editq,'Enable','off')*. Los valores en forma de texto (como *on* y *off*) se escribirán entrecomillados, mientras que los numéricos no. Esta forma de configuración se puede aplicar a la mayoría de los elementos de este panel. La orden complementaria a *set* es *get*, que devolverá el valor de un parámetro. Por ejemplo, en el caso de los *edits*, puede emplearse para recoger si hay algo escrito en ellos o no. La sintaxis empleada es muy similar a la de *set*.

4. Partes destacadas del desarrollo

El proceso de desarrollo de la aplicación es largo y complejo. Como todo, tiene un objetivo, la creación de una aplicación con determinadas funcionales, unos medios, Matlab, y un procedimiento, que desafortunadamente no está preestablecido, ni tampoco existe un método óptimo predefinido. Al haberse empleado este proyecto para el aprendizaje de Matlab, quizás alguno de los procedimientos no sea el más óptimo y existan maneras más sencillas de resolver algunos de los problemas planteados, no obstante, los procedimientos seguidos en este proyecto han demostrado ser completamente válidos.

4.1 Gráfico de la función Leontief

Se ha elegido el gráfico de esta función como ejemplo de desarrollo porque ha sido el que más trabajo ha requerido. Éste se caracteriza por tener forma de L, una función en la que convergen dos rectas en un mínimo, punto en el que corta con la función de costes y la senda de expansión. El principal problema en este caso es el no poder hacer uso de la función *ezplot*, pues como ya se comentó, solo sirve para hacer representaciones sencillas, así que se ha usado una función *plot* compuesta de varias funciones:

```
X0=[0 0];
Y0=[0 0];
X1=[Lqd Lqd];
Y1=[Kqd ydib];
X2=[Lqd xdib];
Y2=[Kqd Kqd];

plot(X0,Y0,X1,Y1,X2,Y2)
```

Imagen 14: Extracto del código del gráfico de la función Leontief.

1. La función [X0, Y0] es la coordenada [0, 0], necesaria para que el gráfico muestre solo la parte positiva del eje de coordenadas. De otra manera se mostraría el eje completo y se vería descentrado.

2. *xdib* e *ydib* son dos objetos creados para establecer los límites superiores y laterales del eje.

3. Las funciones [X1, X2] e [Y1, Y2] representan las coordenadas de L y K respectivamente. X1 e Y2 son los puntos mínimos de L y K y X2 e Y1 son rectas que van desde ambos mínimos hasta q_0, que es la cantidad producida.

4. Por último, hay que unir estas coordenadas en la función *plot*, y asignarle las propiedades que se deseen.

4.2 Gráficos seleccionables con título y leyenda variables:

Visualmente quizás sea ésta una de las partes más atractivas del trabajo, aunque su diseño fue especialmente largo y complejo. Se presentaban varias opciones, entre ellas poner diferentes ejes que fueran apareciendo y desapareciendo, aunque esta suponía dibujar nueve ejes y no era 100% fiable, además de hacer más difícil su configuración. Se optó entonces por lo siguiente:

1. Se crean tres objetos: *ctqcheck*, *cmecheck* y *cmgcheck*, asociados a los *checkbox* del gráfico dos, que detectan qué casilla está marcada. El valor de estos objetos puede ser 0 si la casilla no está marcada y 1 si lo está. Se crea un último objeto, *chk*, que es un vector formado por los 3 objetos anteriores, y que tiene nueve combinaciones posibles de 1 y 0. Cada combinación representa un gráfico. Además, a cada gráfico (costes medios, totales y marginales) se le asigna un objeto, *p4*, *p5* y *p6*. Por ejemplo, la combinación [0,0,0] significa que ninguna casilla ha sido marcada, mientras que la [1,1,1] significa que todas han sido marcadas, o la [1,0,1] significa que se representarán los costes totales y marginales. Todas estas combinaciones se enlazan con condicionales *if* y *elseif*, para que el programa reconozca cada caso (el total de esta parte del código supone más de 70 líneas, por lo que solo se muestran dos de ellos), y a cada uno se le ajusta o no, dependiendo de las necesidades de cada representación, un límite diferente. También se ajusta a cada caso el color de la línea y el grosor.

```
442 -   ctqcheck=get(handles.rbctq,'Value');
443 -   cmecheck=get(handles.RbCme,'Value');
444 -   cmgcheck=get(handles.RbCmg,'Value');
445 -   chk=[ctqcheck,cmecheck,cmgcheck]; %creamos un vector para seleccionar todas as posibles combinaciones de grafico
```

Imagen 15: Creación del gráfico móvil: parte 1a

```
483 -       elseif chk==[1,0,1]
484 -           p6=ezplot(ctq,[0 Cqd]);
485 -           set(p6,'Color','r','Linewidth',2)
486 -           hold on
487 -           p5=ezplot(cmg, [0, Cqd]);
488 -           set(p5,'Color','y','Linewidth',2)
489 -           hold off
490
491 -       elseif chk==[0,1,1]
492 -           p4=ezplot(cme, [0, Cqd]);
493 -           set(p4,'Color','g','Linewidth',2)
494 -           hold on
495 -           p5=ezplot(cmg, [0, Cqd]);
496 -           set(p5,'Color','y','Linewidth',2)
497 -           xlim([0, double(2*cme)])
498 -           ylim([0, double(2*cme)])
499 -           hold off
```

Imagen 16: Creación del gráfico móvil: parte 1b

.

2. La segunda parte consiste en ajustar las leyendas y el título variables. El procedimiento es similar al utilizado anteriormente, creando cuatro objetos: *A*, *B*, *C* y *D* (imagen 17). *A*, *B* y *C* tienen un rango de valores posibles de 0 a 8, mientras que *D* solo puede ser 0 o 1. La función de estos objetos es detectar qué gráfico se ha dibujado, y ajustar la leyenda correspondiente. *A* detecta si se han representado los costes medios, *B* si se han representado los costes marginales, *C* si se han representado los costes totales y *D* si coinciden los costes marginales y los medios.

   El comando *exist* devuelve un valor entre 0 y 8. Si es 0, significa que el objeto no existe, y los valores entre 1 y 8 significan que el objeto existe e indican en qué forma (si el objeto es parte del espacio de trabajo, si es un archivo independiente, etc). Empleado en conjunto con los objetos mencionados anteriormente, permite saber si se han seleccionado costes medios, marginales o totales, dependiendo de que el valor del objeto asociado a cada coste sea 0 (no existe, luego no se ha seleccionado ese tipo de coste para la representación gráfica y no se mostrará en la leyenda) o mayor que 0 (existe, luego se ha señalado ese tipo de coste para la representación gráfica y se mostrará en la leyenda). Por ello se decidió usar nuevamente el condicional *if-elseif* combinado con &&, que permite crear casos en los que estrictamente se tienen que cumplir las condiciones encadenadas con &&. Se ha tenido que hacer así porque el comando *legend*, mediante el cual se crea la leyenda, no funcionará si alguno de los objetos que tienen que aparecer en la leyenda no existe. Por ello se tienen que crear

todos los escenarios posibles y combinaciones de *legend*, para evitar que el programa intente representar en la leyenda un objeto que no existe y falle. Con esto se consigue también que el título se ajuste a lo mostrado en el gráfico, mediante la función *title* (como se puede observar en la imagen 18).

```
507 -       A=exist('p4');
508 -       B=exist('p5');
509 -       C=exist('p6');
510 -       D=isequal(cmg,cme);
```

Imagen 17: Creación del gráfico móvil: parte 2a

```
720 -   if C>0 && B>0 && A>0 && D==0
721 -       legend([p6,p5,p4],'Totales','Marginales','Medios','Location','northwest')
722 -       title('Costes totales, medios y marginales')
723 -   elseif C>0 && B>0 && A>0 && D==1
724 -       legend([p6,p4],'Totales','Medios y marginales','Location','northwest')
725 -       title('Costes totales, medios y marginales')
726 -       set(p4,'Color','g','Linewidth',2)
727 -       set(p5,'Color','g','Linewidth',2)
728 -   elseif A>0 && C>0 && B==0
729 -       legend([p4,p6],'Medios','Totales','Location','northwest')
730 -       title('Costes medios y totales')
```

Imagen 18: Creación del gráfico móvil: parte 2b

4.3 Maximización de la producción a partir de unos costes dados

Aunque supone un objetivo prácticamente inverso al original del proyecto, merecía la pena comprobar si era viable combinar ambas funcionalidades en una misma interfaz. Por restricciones de tiempo esta funcionalidad sólo se ha implantado en la función Leontieff, ya que supone prácticamente rehacer toda una parte del proyecto para su implantación. Afortunadamente, tras un largo y complicado proceso que ha supuesto remodelar prácticamente todo el código de esta función y su interfaz, se ha implantado con éxito y funcionando sin errores conocidos hasta el momento. El proceso comienza hallando las funciones para obtener la producción mediante los costes:

1. Se añade una nueva variable de entrada, los costes establecidos, y una nueva de salida, la producción a partir de costes. A ellas se añaden sus respectivas funciones asociadas

en el script, así como las nuevas fórmulas de K y L que surgen a partir de estos cambios. Se crea también un condicional que determina qué gráfico usar, si el que depende de la producción o el que depende de los costes. Esta elección dependerá de que el programa detecte que hay costes establecidos o no.

2. En la interfaz se producen varios casos. Como ya se mencionó antes, ésta cuenta con una validación de datos para impedir que se introduzcan datos ilógicos o que se ejecute el programa sin datos de entrada. Cada uno genera una condición, pero al añadir este nuevo caso el número de condiciones aumenta considerablemente y hay que crear un vector similar al empleado en el gráfico de tres variables explicado anteriormente, esta vez con seis opciones binarias posibles. En este caso, se combina el condicional *if-elseif* con ||, que significa *ó*. Se emplea cuando se quiere aplicar una misma condición a diferentes casos.

```
159    %establecemos los condicionales y un vector para que el
160    %programa sepa si trabajar con qd o cs
161    % Comprobamos que no está vacío y que es positivo
162
163 -  csz=get(handles.checkboxctq,'Value');%nos dice si el radio button de ctq esta activo
164 -  s=(str2num(get(handles.editctq,'String'))); %#ok<ST2NM> %nos da el valor de la casilla Ctq
165
166    %Establecemos las condiciones para que nos reconozca si hay un valor
167    %válido en la casilla de ctq
168
169 -  if s<=0
170 -      cscheck=0;
171 -  elseif s>0
172 -      cscheck=1;
173 -  else
174 -      cscheck=0;
175 -  end
176
177    %Si cscheck=1, el valor de Cqd será válido, si es 0, no
178    %Establecemos las condiciones para que nos reconozca si hay un valor
179    %válido en la casilla de qd
180
181 -  if qd<=0
182 -      qchk=0;
183 -  elseif qd>0
184 -      qchk=1;
185 -  else
186 -      qchk=0;
187 -  end
188
189    %Si qchk=1, el valor de qd será válido, si es 0, no
190 -  m=isempty(cs);
191 -  v=isempty(qd); %Para saber si la casilla de qd está vacía. Si v=1, lo está
192 -  qmz=get(handles.checkboxq,'Value');%Para saber si hemos marcado la casilla q. Si es 1, la hemos marcado
193
194    %qmz=1 radiobutton de q marcado / qmz=0 radiobutton de q desmarcado
195    %v=1 editq vacio / v=0 editq relleno
196    %qchk=1 valor válido para qd /qchk=0 valor no válido para qd
197    %csz=1 radiobutton de ctq activo / csz=0 radiobutton de Ctq off
198    %cschek=1 valor válido para cs / cschk= 0 valor no válido
199    %m=1 la casilla de ctq esta vacia/ m=0 casilla rellena
200
201 -  qsz=[qmz,v,qchk,csz,cscheck,m]; %Vector que nos junta las 4 condiciones para que salte el error
202
```

Imagen 19: Elección entre minimización de costes o maximización de la producción.

En la imagen 19 se muestra cómo se crean los objetos que ayudan al programa a identificar cada caso y en la imagen 20 se muestra cómo se usan en la práctica estos objetos.

```
210     %valor válido para qd, sin ctq
211
212 -   elseif qsz==[1,0,1,0,1,1] | qsz==[1,0,1,0,0,0] | qsz==[1,0,1,0,1,0] | [1,0,1,0,0,1]
213
```

Imagen 20: Ejemplo de identificación de caso mediante un vector.

En el caso mostrado, el programa sigue la línea estándar de operaciones: el usuario marca la casilla de *qd*[7] e introduce un número válido en la misma. La interpretación de los vectores es la siguiente: la condición se ejecuta si el usuario marca el radio button de *qd*, la rellena con un valor válido, el radio button de *Ctq*[8] está desmarcada (activar el de *qd* desactiva automáticamente el de *Ctq*) y haya algo o no escrito, ya sea válido o no, en el *edit* de *Ctq*. En total se han tenido que realizar 3 casos más como el mostrado anteriormente: uno para cuando se introduzca un valor no válido en el *edit* de *qd* o *Ctq*, otro cuando no se introduzca ningún valor en el *edit* de *qd* (para ejecutar el programa con un valor estándar de 1000) y otro para cuando se marque la casilla de *Ctq* y se quiera maximizar *q*.

[7] Cantidad dada, se ha llamado así al nivel de producción a partir del cual se va a obtener el coste mínimo
[8] Coste para la cantidad dada, se ha llamado así al coste mínimo para un nivel de producción q.

5. Uso del programa

5.1 Minimización de costes

A continuación, se muestra un ejemplo del funcionamiento de la aplicación. Se toma el módulo de la función Leontief para, a partir de una producción dada, minimizar costes. La función de este ejemplo será la siguiente:

$$Q(L,K) = \min\{50L, 50K\}.\ Valor\ de\ Q = 1000$$

$$sujeto\ a\quad CT(L,K) = 2L + K$$

1. La primera pantalla a la que se accede tras ejecutar la aplicación es el menú. Desde aquí se debe seleccionar la función a la que se está enfrentando el usuario, en este caso, una función Leontief. Por ello, se procede a hacer click en el botón "Leontief"

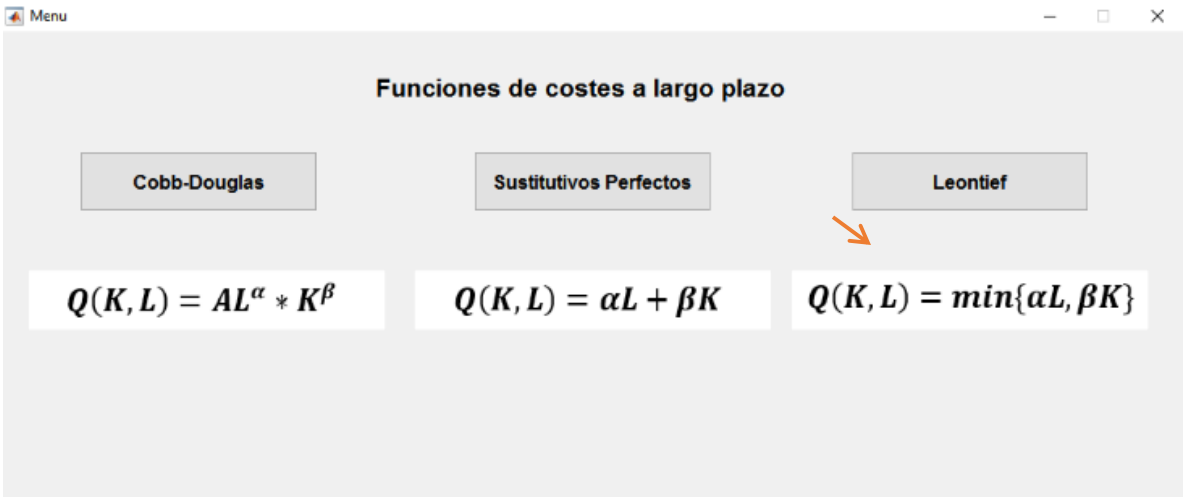

Imagen 21: Menú principal de la aplicación.

2. La segunda pantalla, a la que se accede tras hacer click en el botón "Leontief", es la interfaz de usuario para la función en cuestión. Se procede entonces a introducir los datos en las casillas correspondientes. Como en este caso lo que se pretende es minimizar costes, es decir, se conoce el valor de la producción, se selecciona el *radio button* de *q*. Adicionalmente, se puede seleccionar qué gráficos de costes se quiere que represente el programa, a elegir entre costes totales, medios y marginales (o todos a la vez). En este ejemplo no se elegirán ningunos, por lo que el programa muestra

por defecto los costes totales. Finalmente, se presiona el botón aplicar para obtener los resultados.

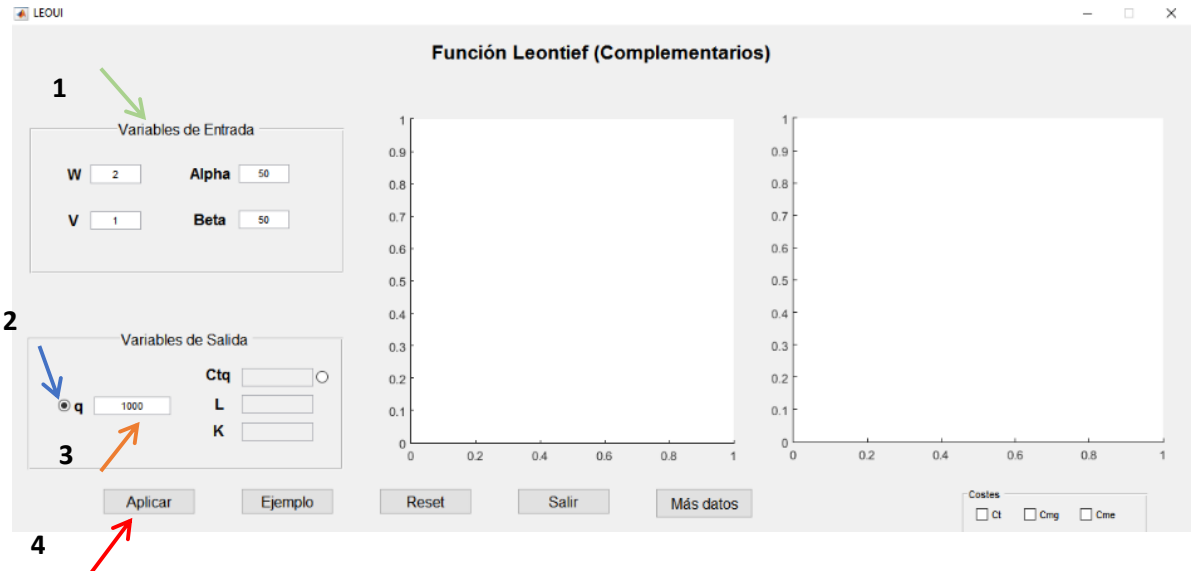

Imagen 22: Interfaz de usuario de la función Leontief, minimización de costes.

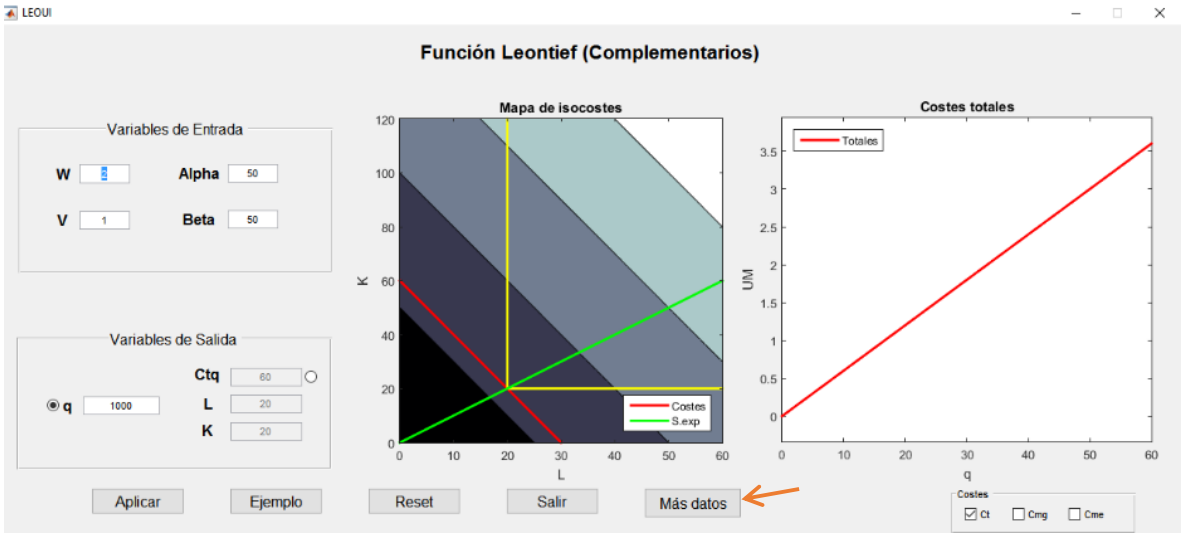

Imagen 23: Pantalla de resultados de la función Leontief, minimización de costes.

3. En la pantalla de resultados, se puede hacer click en el botón “Más datos” para desplegar un panel con datos adicionales.

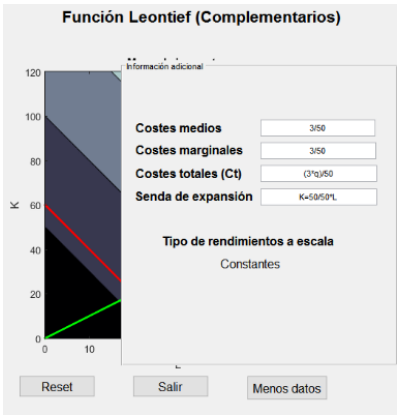

Imagen 24: Panel de datos adicionales de la función Leontief, minimización de costes.

5.2 Maximización de la producción (función Leontief)

La interfaz de la función Leontief es la única de las tres que ofrece la posibilidad de maximizar la producción a partir de un coste dado. El funcionamiento es el mismo que el explicado en el apartado anterior, con la diferencia de que en lugar de seleccionar el *radio button* asociado a q habrá que seleccionar el asociado a Ctq. En el siguiente ejemplo, se tomarán los mismos parámetros que en el anterior, salvo que en lugar de la producción se conocerá el coste, 60.

1. En primer lugar, se introducen todos los datos, como en el caso anterior, y se selecciona el *radio button* de *Ctq*, como se indica en la imagen. Posteriormente, se introduce el coste dado y se selecciona el gráfico de costes deseado, en este ejemplo, el de costes medios. Finalmente se hace click en el botón de “Aplicar”, como se muestra en la imagen 25.

2. Al igual que en el ejemplo anterior, se dispone de un panel de datos adicionales que puede ser desplegado haciendo click en el botón de “Más datos” (imagen 27).

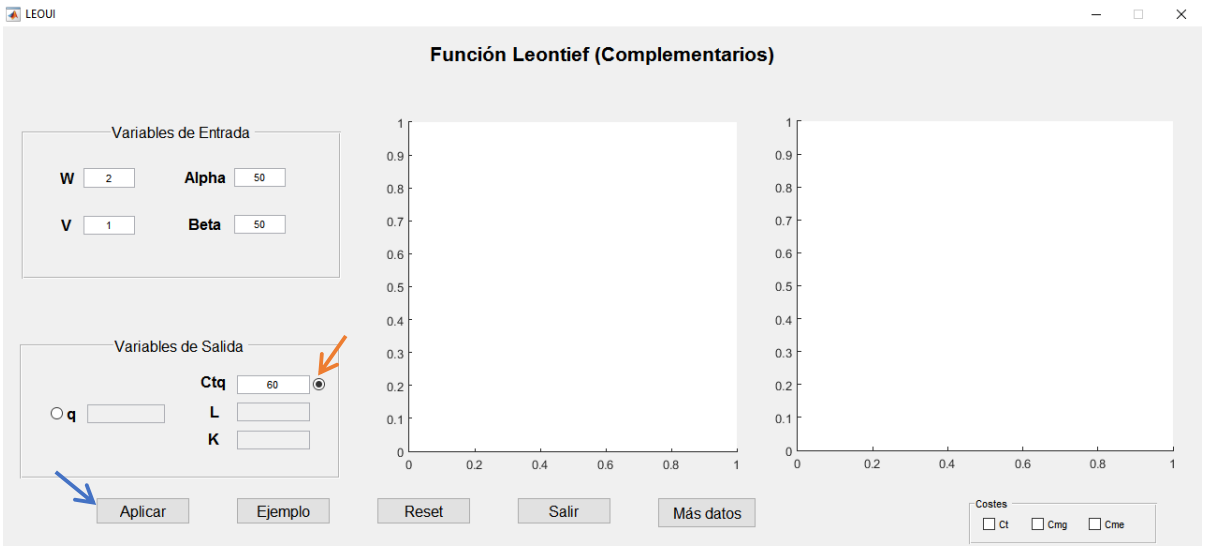

Imagen 25: Interfaz de usuario de la función Leontief, maximización de la producción.

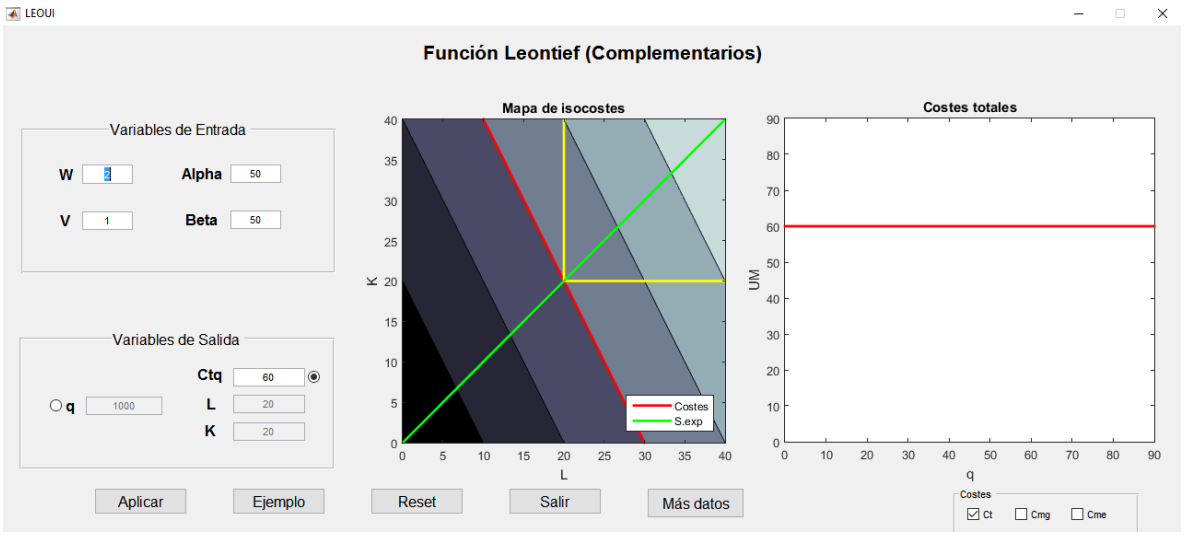

Imagen 26: Pantalla de resultados de la función Leontief, maximización de la producción.

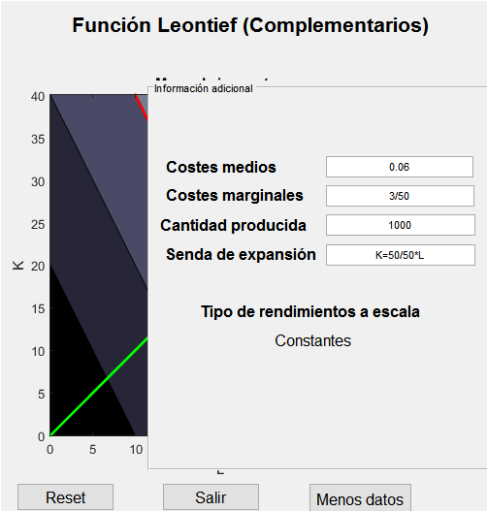

Imagen 27: Panel de datos adicionales de la función Leontief, maximización de la producción.

6. Conclusiones

Matlab ha demostrado ser un software muy útil en el campo de la Microeconomía, ya que hemos conseguido programar tanto las funciones empleadas habitualmente en la asignatura de Microeconomía del grado en ADE de la UPCT como sus representaciones gráficas, facilitando así al usuario la obtención de gran cantidad de datos asociados a dichas funciones. Estas representaciones gráficas incluyen el mapa de isocostes, la recta de isocoste correspondiente al nivel de costes dado, la isocuanta para el nivel de producción dado, la senda de expansión y los costes totales, medios y marginales a largo plazo. Los datos que se pueden llegar a obtener en función de cada caso, son las cantidades de los factores trabajo (L) y capital (K) que minimizan los costes para una producción dada, las productividades marginales para el nivel de producción dado, el tipo de rendimientos de escala, la RMST y las expresiones matemáticas asociadas a los elementos representados gráficamente.

Esta aplicación está pensada principalmente para su uso en la docencia, facilitando por un lado la labor del profesor al enseñar a los alumnos el comportamiento de dichas funciones y, por otro, la del alumno a la hora de estudiar autónomamente y comprobar algunos resultados de ejercicios. Por todo lo mencionado anteriormente y porque además es sencilla de usar, creemos que realmente resultará de utilidad en la enseñanza de dicha asignatura puesto que no requiere que el usuario tenga algún conocimiento previo de Matlab. No obstante, estamos confiados en que podría animar a los usuarios a experimentar autónomamente con este software.

El aspecto más positivo del programa Matlab es la gran cantidad de herramientas que ofrece, la opción de crear interfaces gráficas y la posibilidad de crear una aplicación independiente que no requiera de Matlab para funcionar. Por supuesto, también cabe destacar algunos aspectos negativos de este software, siendo el más relevante de ellos que se trata de un software que requiere licencia, es decir, no es gratuito. Por otro lado, requiere más tiempo para dominarlo que otros programas más sencillos, si bien luego el potencial de éste compensa con creces el tiempo invertido.

Por motivos de tiempo no ha sido posible avanzar en este proyecto en algunas líneas de actuación que consideramos especialmente interesantes. Una de ellas sería añadir la opción de maximizar la producción a partir de costes a los scripts de las funciones Cobb-Douglas y Bienes Sustitutivos Perfectos. Otra alternativa sería añadir la opción de generar funciones automáticamente para proporcionar a los estudiantes una fuente de ejercicios generados aleatoriamente y cuyas soluciones puedan comprobarse. Esperamos disponer del tiempo suficiente en el futuro para desarrollar estos aspectos.

7. Anexo I

Lenguaje máquina: lenguaje nativo del ordenador, en forma de instrucciones binarias, directamente legibles por el microprocesador (Pcmag.com, 2016).

Código fuente: secuencia de comandos que debe ser compilada por un programa para poder ejecutarse

Script: (Gilat and Macías Iglesias, 2006) secuencia de comandos que debe ser interpretada por un programa para poder ejecutarse (Colburn, 2003).

Programa intérprete: programa que lee y ejecuta el script comando a comando, siguiendo un orden establecido (Colburn, 2003).

Lenguaje interpretado: lenguaje en el que se lee el script en orden comando por comando, ejecutando un comando antes de pasar al siguiente (Colburn, 2003).

Programa compilador: Convierte el script en lenguaje máquina y lo almacena, bien en un archivo ejecutable (.exe) o bien en un formato intermedio (por ejemplo, bytecode) (Colburn, 2003).

Lenguaje compilado: Lenguaje que requiere que el script sea convertido en lenguaje máquina antes de ser interpretado.

Variable simbólica: variable sin valor numérico asignado.

Depurar (el código): Procedimiento que consiste en la identificación y reparación de errores en el script o código fuente, ya sean errores de sintaxis o lógicos, así como la eliminación de líneas innecesarias y/o la optimización de las existentes. Conocido también por el término inglés *Debug* o por la españolización informal de este término, *Debuggear*.

Relación Marginal de Sustitución Técnica (RMST): Según Pindyck y Rubinfeld (2009), es la cantidad en la que puede reducirse el capital cuando se utiliza una unidad más de trabajo, o viceversa.

Productividad marginal (del trabajo o del capital): Aumento de la producción al aumentar en una unidad uno de los inputs.

8. Bibliografía

- Borrell i Nogueras, G. (2007). *Introducción informal a Matlab y Octave*. 1st ed.
- Es.mathworks.com. (2016). *GUI de MATLAB - MATLAB & Simulink*. [online] Available at: https://es.mathworks.com/discovery/matlab-gui.html [Accessed 16 Dec. 2016].
- Colburn, R. (2003). *Sams teach yourself CGI in 24 hours*. 1st ed. Indianapolis, Ind.: Sams.
- Facultad de informática, Universidad Politécnica de Madrid (2016). *Lenguajes de guiones (Scripting languages)*. [online] Available at: http://lml.ls.fi.upm.es/ep/script.html [Accessed 20 Dec. 2016].
- Gilat, A. and Macías Iglesias, J. (2006). *Matlab*. 1st ed. Barcelona: Reverté.
- Pcmag.com. (2016). *machine language Definition from PC Magazine Encyclopedia*. [online] Available at: http://www.pcmag.com/encyclopedia/term/46445/machine-language [Accessed 20 Dec. 2016].
- Es.mathworks.com. (2016). *MathWorks - MATLAB and Simulink for Technical Computing - MATLAB & Simulink*. [online] Available at: https://es.mathworks.com/ [Accessed 21 Dec. 2016].
- Petzold, C. (1999). *Code*. 1st ed. Redmond, Wash.: Microsoft Press.
- Msdn.microsoft.com. (2016). *Radio Buttons (Windows)*. [online] Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/dn742436(v=vs.85).aspx [Accessed 23 Dec. 2016].
- Nielsen, J. (2004). *Checkboxes vs. Radio Buttons*. [online] Nngroup.com. Available at: https://www.nngroup.com/articles/checkboxes-vs-radio-buttons/ [Accessed 23 Dec. 2016].

- Pindyck, R. and Rubinfeld, D. (2009). *Microeconomía*. 1st ed. Madrid, España: Pearson Educación.
- Ramírez M., P. (1991). *Las funciones homogéneas y su uso en economía*. [online] Medellín: Universidad de Antioquía, pp.2-3. Available at: https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=5&cad=rja&uact=8&ved=0ahUKEwjc8oD1wZnRAhXD6xoKHYwtDtkQFgg4MAQ&url=https%3A%2F%2Fdialnet.unirioja.es%2Fdescarga%2Farticulo%2F4833884.pdf&usg=AFQjCNHBuVyTlKNzHmhzickye5_Eo5Om2A&sig2=aP44QDsRZmDtBozKBLvF1A [Accessed 29 Dec. 2016].
- Mas-Colell, A., Whinston, M. and Green, J. (1995). *Microeconomic theory*. 1st ed. New York: Oxford University Press.
- Cobb, C., & Douglas, P. (1928). A Theory of Production. *The American Economic Review*, *18*(1), 139-165. Retrieved from http://www.jstor.org/stable/1811556
- Nicholson, W. and Cole, J. (2008). *Teoría microeconómica*. México, D.F.: CENGAGE Learning.