Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

Coordinación de políticas de precio y reposición para artículos con pérdida de valor

Autor: María Quirós Prieto

Tutor: Jesús Muñuzuri Sanz

Dpto. de Organización Industrial y Gestión de Empresas II
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

Coordinación de políticas de precio y reposición para artículos con pérdida de valor

Autor:

María Quirós Prieto

Tutor:

Jesús Muñuzuri Sanz

Catedrático

Dpto. de Organización y Gestión de Empresas II

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2022

Trabajo Fin de Máster: Coordinación de políticas de precio y reposición para artículos con pérdida de valor

Autor: María Quirós Prieto

Tutor: Jesús Muñuzuri Sanz

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi abuelo Juan.

AGRADECIMIENTOS

En primer lugar, quiero dar las gracias a mi tutor en este Trabajo Fin de Máster, D. Jesús Muñuzuri Sanz, a quien considero uno de los mejores profesores de esta Escuela. Su manera de transmitir sus conocimientos y el compromiso que muestra con sus alumnos, contribuyeron a que me decidiese por la especialidad de Organización y Producción durante el Grado. Supone una gran satisfacción para mí poner fin a este Máster y a la etapa universitaria con este proyecto dirigido por él. Aunque me habría gustado que nuestras conversaciones sobre este trabajo hubiesen sido presenciales, quiero agradecerle todo el tiempo invertido en cada correo, en cada duda, y también su paciencia y comprensión con los tiempos que he necesitado. Sin su apoyo no habría sido posible concluir este proyecto.

Agradecer también a José Ángel Gutiérrez de Ravé Millán sus contribuciones en la parte más temprana de este trabajo y que tanto me sirvieron en el ámbito de la programación en Python.

Gracias, de corazón, a mis amigos, los que me ha dado esta Escuela. Nunca dejaré de sentirme afortunada por haber compartido con ellos esta etapa y todas las que vienen y vendrán. Y por supuesto, también a mis amigos de toda la vida, porque a pesar de la distancia, han sido partícipes de cada momento importante.

A Carlos, por acompañarme en cada paso y quedarse siempre.

Por último, dar las gracias a mi familia, por haber estado conmigo enorgulleciéndose de cada uno de mis éxitos y apoyándome en cada lección que aprender. A mis padres, por ser siempre mi respaldo, mi refugio y mi motor. A mi hermana, por todo lo que hemos crecido y aprendido por el camino.

Y en especial, dar las gracias a la persona a la que está dedicado este trabajo: Mi abuelo Juan. Por enseñarme tantas cosas, por animarme a ir “al toro” siempre. Por sentirse orgulloso de mí sin condición. Por quererme tanto y por llevarme siempre por bandera. Sé que, aunque ahora lo haga desde el cielo, me celebra una vez más.

RESUMEN

Los modelos de gestión de inventarios suelen partir de una serie de suposiciones que en numerosas ocasiones distan de los casos de estudio reales, como podrían ser considerar la demanda independiente del precio del bien o que la mercancía pueda permanecer en el almacén indefinidamente. Además, lo común es considerar que los bienes tienen una vida útil infinita, algo que no siempre ocurre, puesto que son muchos los productos perecederos o que pierden propiedades o el interés de los consumidores con el tiempo.

No obstante, en las últimas décadas, algunos investigadores han trabajado en modelos que sí tienen en cuenta el deterioro que experimentan muchos productos. Esta literatura ha ido aumentando con los años y los modelos propuestos son cada vez más completos y complejos. Sin embargo, en todos ellos se considera que el deterioro se comienza a producir desde el momento en que se produce el artículo, lo cual no siempre es así.

No es hasta entrados los 2000 cuando se propone un primer modelo en el que se considera un periodo de tiempo en el que el producto no se estropea y mantiene su calidad y propiedades. Este fenómeno fue definido como "deterioro no instantáneo" por los autores Wu, Ouyang & Yang en su artículo *Coordinating replenishment and pricing policies for non-instantaneous deteriorating items with price-sensitive demand* (2009), en el que está basado este trabajo. El modelo propuesto tiene en cuenta de forma simultánea, además, dos situaciones reales: considerar una demanda elástica al precio y la coordinación de políticas de precio y reposición. Finalmente proponen un algoritmo matemático y varios ejemplos ilustrativos.

Este Trabajo Fin de Máster tiene como objetivo la programación de dicho modelo en Python, comprobar los resultados expuestos y obtener una herramienta que permita determinar la política óptima de precios y reposición en casos similares que se puedan encontrar en la vida real.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN Y OBJETO DEL TRABAJO

Los modelos de inventario tradicionales están sujetos al cumplimiento de una serie de condiciones, como que los bienes pueden permanecer almacenados sin límite temporal o el hecho de que la demanda sea independiente del precio del bien. Estos supuestos no siempre pueden considerarse en la vida real.

Un caso muy común es el de aquellos bienes que con el paso del tiempo sufren un deterioro gradual de sus propiedades físicas o químicas o simplemente cada vez son menos atractivos para el comprador, al quedar desfasados, pasados de moda o ser sustituidos por nuevos productos con mejores prestaciones o con un diseño más interesante.

Como se puede apreciar, el deterioro puede producirse por diferentes motivos:

- En el caso de los alimentos, el deterioro va relacionado con que se trata de productos perecederos que tienen fecha de caducidad o consumo preferente. Es bastante común encontrar en los establecimientos ofertas en productos como la carne o el pescado cuya fecha de caducidad está próxima, lo cual vuelve ese producto más atractivo para el cliente, que se plantea elegir ese en lugar de uno con una fecha de caducidad más tardía pudiendo beneficiarse de un descuento en el producto. También ocurre esto con las frutas y verduras, las cuales pueden presentar un deterioro visual cuando han pasado un mayor tiempo en las estanterías, aunque siguen siendo aptas para el consumo. En estos casos ocurre algo similar, es posible encontrar descuentos en estos productos, ya que al vendedor le interesa, en cualquier caso, darle salida con un precio que siempre es mejor que perder la venta por caducidad del producto.

- En el caso de la electrónica, el deterioro va más ligado a la obsolescencia o a que los productos van siendo continuamente sustituidos por nuevos modelos con mejores prestaciones que resultan más atractivos para el comprador. Para que las versiones anteriores sigan despertando interés en el consumidor, se suelen ofertar a precios más bajos.

- En el caso de la moda, el deterioro está más relacionado con el fin de temporada o que las prendas pasen de moda. Al final de cada temporada suele haber un periodo de rebajas, en el que se bajan los precios a una gran cantidad de productos, o bien, se vuelven a poner a la venta artículos de anteriores temporadas con un precio más ajustado, con la finalidad de dar salida al stock de productos más antiguos que continúan generando costes de almacén.

Normalmente, todo producto o bien tiene una vida útil determinada, dándose por hecho que llegará un momento en el que no pueda usarse o consumirse. Normalmente este fin de la vida útil no se da de un momento a otro, sino que los bienes van experimentando algún tipo de "degradación" progresiva. Como se puede observar, y entendiendo el deterioro no solo aplicado a bienes perecederos como se acaba de exponer, son numerosos los ejemplos de mercancías o productos que no mantienen de forma indefinida las características con las que son puestos en el mercado, incluso podría decirse que es más habitual encontrarse con esta casuística que con la de productos que mantienen sus propiedades intactas para siempre.

Desde el punto de vista de cualquier comerciante o vendedor, sería interesante estudiar la forma en que podría tratar de dar salida a un stock de productos que, a partir de momento dado, deja de tener las propiedades iniciales con las que se fabricó obteniendo el máximo beneficio posible.

En esta línea, Wu, Ouyang & Yang en su artículo *Coordinating replenishment and pricing policies for non-instantaneous deteriorating items with price-sensitive demand* (2009) desarrollaron un modelo de inventario centrado en bienes que experimentan deterioro con el tiempo y, en particular, bienes en los que ese deterioro no comienza hasta pasado un cierto tiempo desde que son fabricados. Esta cuestión, como se verá en el

capítulo siguiente, es lo que diferencia este modelo de los desarrollados hasta la fecha. Además, añade dos situaciones reales, como son una demanda elástica al precio y la idoneidad de coordinar políticas de precio y reposición.

En este Trabajo Fin de Master se desarrollará el análisis teórico que lleva a los autores a determinar las políticas de precio y reposición óptimas. Éste parte de la demostración de que, para un precio de venta determinado, el valor óptimo de duración del ciclo de reposición no solo existe, sino que, además, es único. A continuación, se demostrará que existe un único precio de venta que maximiza el beneficio total por unidad de tiempo para una duración del ciclo de reabastecimiento dada.

Adicionalmente, se explicará un algoritmo sencillo que los autores proponen y que permite encontrar la solución óptima al problema planteado. Proponen también dos ejemplos numéricos para ilustrar los resultados teóricos y un análisis de sensibilidad de la solución óptima con respecto a los principales parámetros del modelo.

El objeto de este TFM es, por un lado, poner en valor la aportación de Wu, Ouyang & Yang (2009) a la literatura sobre modelos de inventario y, por otro, contribuir a su trabajo con la programación en Python del algoritmo que proponen.

Los distintos códigos elaborados se utilizan en este proyecto para comprobar los resultados de los dos ejemplos prácticos y el análisis de sensibilidad expuestos en el artículo, aunque podrían aplicarse a la resolución de casos reales similares de productos con deterioro no instantáneo.

2 REVISIÓN DE LA LITERATURA

Este capítulo tiene como finalidad hacer un recorrido por la literatura que ha tratado de abordar la problemática de la gestión de inventarios de bienes con deterioro desde las primeras aproximaciones realizadas a mediados del siglo XX hasta nuestros días.

Resulta imprescindible comenzar mencionando el modelo fundamental y más utilizado en control de inventarios: La Cantidad Económica de Pedido ("Economic Order Quantity", EOQ), también conocida como Modelo de Wilson, se basa en hallar el punto de pedido para el que se igualan los costes de lanzar un pedido y los costes de mantenimiento del producto, tomando una demanda determinista (constante y conocida). Este modelo, aunque es ampliamente utilizado debido a su sencillez y robustez, presenta algunos inconvenientes, como considerar constantes tanto la demanda como la disminución del inventario, o que no tenga en cuenta roturas de stock ni descuentos por gran volumen. Por supuesto, no trata escenarios en los que se tienen productos con tasa de deterioro.

Dado que en la actualidad ha cobrado una gran importancia el control y mantenimiento de inventarios de bienes perecederos, muchos autores han tratado de abordar esta problemática desde mediados del siglo XX. Ghare y Schrader (1963) fueron los primeros en presentar un modelo EOQ para un inventario con deterioro constante y exponencial. Más tarde, la suposición de una tasa constante de deterioro fue relajada por Covert y Philip (1973), formulando un modelo con una tasa variable de deterioro usando una distribución Weibull de dos parámetros, el cual a posteriori fue generalizado por Philip (1974) usando una Weibull de tres parámetros.

Shah (1977) amplió el modelo de Philip (1974) considerando posibles roturas de stock. Goyal y Gunasekaran (1995) desarrollaron un modelo integrado de producción-inventario para artículos con deterioro. Lin, Tan y Lee (2000) propusieron un modelo EOQ para este tipo de bienes considerando una demanda variable con el tiempo y roturas de stock. Por último, destacar Goyal y Giri (2001), un recopilatorio de toda la literatura relativa a los bienes con pérdida de valor desde principios de los años 90s.

La política de precios es una de las principales estrategias que aplican los vendedores para obtener el máximo beneficio. El precio tiene un impacto directo sobre la demanda, y los modelos con demanda dependiente del precio ocupan un puesto predominante en la literatura sobre inventarios. Cohen (1977) determina el ciclo de reabastecimiento óptimo y el precio, conjuntamente, para un inventario sujeto a una tasa de deterioro constante en el tiempo. Wee (1995) estudió la política conjunta de precios y reposición para un inventario que se deteriora con una elasticidad precio de la demanda que disminuye con el tiempo. Abad (1996) consideró el problema dinámico de fijación de precios y dimensionamiento de lotes de un bien perecedero bajo pedidos parciales. Asumió que la fracción de unidades que se pide es variable y es una función decreciente del tiempo de espera.Wee en sus dos publicaciones posteriores (1997), (1999), amplía el modelo de Cohen (1977) para adaptarlo a una política de aprovisionamiento para artículos con pérdida progresiva de valor con una demanda dependiente del precio y una distribución Weibull para el deterioro con la posibilidad de tener o no en cuenta descuentos por cantidad o volumen. Wee y Law (2001) desarrollan un modelo de inventario para artículos con pérdida de valor con demanda dependiente del precio teniendo en cuenta, además, el valor temporal del dinero (TVM). Abad (2003) considera conjuntamente precio y tamaño de lote bajo condiciones de deterioro, producción finita y entregas parciales. Mukhopadhyay, Mukherjee y Chaudhuri (2004), (2005) reestablecieron el modelo de Cohen (1977) considerando tasa de elasticidad de la demanda y una tasa de deterioro proporcional al tiempo o bien, modelada por una distribución Weibull de dos parámetros, ambas opciones. Chang, Teng, Ouyang y Dye (2006) introdujeron un modelo de inventario con deterioro con una demanda dependiente del tiempo y del precio y entregas parciales. Dye (2007) propone políticas conjuntas de fijación de precio y de pedido para inventarios con deterioro con demanda dependiente del precio.

Todos los modelos de inventario para artículos en deterioro que se recogen en la literatura anteriormente mencionada tienen en común el asumir que el deterioro ocurre tan pronto como se produce el artículo y el nivel de inventario se reduce debido a ese deterioro al comienzo de cada ciclo de reabastecimiento. Sin

embargo, en la realidad existen muchos bienes cuyo deterioro no se produce hasta pasado un tiempo, es decir, durante un periodo de tiempo determinado mantienen su calidad o condición inicial sin sufrir deterioro alguno. Por ejemplo, las frutas y verduras de primera mano tienen un periodo de tiempo en el que, aunque es corto, mantienen su frescura. Después, algunas unidades de producto comenzarán a estropearse. O en el caso de las prendas de vestir, tampoco se considera que éstas se devalúen desde el principio, sino desde que acaba la temporada. Wu, Ouyang y Yang (2006) definieron este fenómeno como "deterioro no instantáneo", desarrollando una política de reabastecimiento para artículos con deterioro no instantáneo con una demanda dependiente del nivel de stock, de modo que el coste total de inventario por unidad de tiempo tenga un valor mínimo. También consideraron la posibilidad de tener entregas parciales de mercancía.

Este proyecto toma como base el artículo de Wu, Ouyang y Yang (2009), en el cual, se utiliza el modelo de Wu et al. (2006) para artículos con deterioro no instantáneo, aunque coordinando políticas de fijación de precio y reaprovisionamiento con una demanda sensible al precio.

El concepto de "deterioro no instantáneo" que definen estos autores despertó el interés de la comunidad investigadora, siendo en la década siguiente una cuestión sobre la que se continuó trabajando, desarrollándose nuevos modelos de gestión de inventarios que asumían diversas hipótesis.

Rezagholifam, Sadjadi, Heydari y Karimi (2022) realizan un recorrido por la literatura sobre este tipo de modelos de inventario desde Wu et al. (2009) hasta la actualidad. Uno de los más recientes es el de Patel y Gor (2019) quienes presentan un modelo para artículos con deterioro no instantáneo donde la tasa de deterioro viene definida por una Weibull de tres variables. Volviendo atrás en el tiempo, casi paralelamente a Wu et al. (2009), Yang, Ouyang y Wu (2009) propusieron un modelo de inventario para artículos con deterioro no instantáneo con demanda dependiente del precio que además consideraba entregas parciales. Posteriormente, fue propuesto un modelo aún más completo que no solo tenía en cuenta las hipótesis del anterior, sino que, además, también permitía escasez de artículos y retrasos en los pagos (Maihami y Abadi (2012); Maihami y Kamalabadi (2012)). En relación con la coordinación de políticas de precios y reposición para los sistemas de inventario de artículos con deterioro, algunas investigaciones interesantes han sido realizadas por Hsieh y Dye (2010), Vaghela y Shah (2020) y Wang y Huang (2014).

Otras contribuciones interesantes a la literatura de este tipo de modelos de inventarios han sido el estudio de los efectos de:

- La inversión en tecnología de conservación de los artículos (Dye, C. Y., 2013).
- La posibilidad de que algunos productos presenten imperfecciones de calidad desde un primer momento (Mahmoudinezhad, M., Ghoreishi, M., Mirzazadeh, A., & Ghodratnama, A., 2014).
- La inflación (Mahmoudinezhad, M., Ghoreishi, M., Mirzazadeh, A., & Ghodratnama, A., 2014), (Ghoreishi, M., Weber, G. W., & Mirzazadeh, A., 2015), (Shaikh, A. A., Mashud, A. H. M., Uddin, M. S., & Khan, M. A. A., 2017).
- Considerar varios almacenes (Tiwari, S., Cárdenas-Barrón, L. E., Khanna, A., & Jaggi, C. K., 2016), (Khan, M. A. A., Shaikh, A. A., Panda, G. C., & Konstantaras, I., 2019a).
- Considerar diferentes tasas de deterioro (Mashud, A., Khan, M., Uddin, M., & Islam, M., 2018).

Por su parte, la aportación de Rezagholifam, Sadjadi, Heydari y Karimi (2022) es la de considerar cambios de precio durante el horizonte de planificación, algo a lo que no se había prestado atención anteriormente.

Hace ya casi un siglo de los primeros artículos que se adentraban en este campo de la gestión de inventarios de productos que sufren deterioro o merman sus cualidades con el paso del tiempo. Las publicaciones y aportaciones anteriormente mencionadas son sólo una parte de la literatura existente hasta la fecha sobre esta temática que, como se ha visto, continúa despertando el interés de la comunidad investigadora.

3 HIPÓTESIS Y NOTACIÓN

Como se ha explicado anteriormente, el objetivo principal del modelo de Wu et al. (2009) es determinar las políticas óptimas de precio y reposición que permitan maximizar el beneficio total por unidad de tiempo en el caso de bienes que se deterioran aunque no desde el instante de su puesta en el mercado.

Se tienen en cuenta de forma simultánea dos situaciones reales:

- Demanda elástica al precio. Es decir, cómo varía porcentualmente la cantidad demandada ante cambios porcentuales en el precio.

- Coordinación de políticas de precio y reposición.

La función objetivo será maximizar el beneficio. Además, se completa el modelo incluyendo el caso en que el tiempo del ciclo de reposición es menor o igual al periodo de tiempo que el artículo permanece sin sufrir deterioro.

Recapitulando, el modelo matemático tendrá como base los siguientes supuestos:

- Se considera un único artículo con deterioro no instantáneo.

- La tasa de reposición es infinita (el pedido se entrega de una vez) y el plazo de entrega es cero (es decir, la entrega es inmediata).

- La demanda del producto es elástica y función del precio de venta:

$$D(p) = \alpha \cdot p^{-\beta}$$

  Donde α (>0) es el factor de escala y β (>1) es el índice de elasticidad del precio.

- No se permiten roturas de stock.

- Durante un periodo determinado, t_d, no se produce deterioro. Al final de dicho periodo, el inventario se deteriora a una tasa constante θ.

- Durante el periodo considerado no se producen sustituciones ni reparaciones de los artículos dañados.

- El horizonte temporal es infinito.

A continuación, se detallan los parámetros y símbolos que se usarán en adelante:

A = coste de pedido.

c = coste de compra unitario.

h = coste unitario de almacenamiento por unidad de tiempo.

p = precio unitario de venta.

T = tiempo de ciclo de reposición o reabastecimiento.

Q = cantidad de pedido.

θ= tasa de deterioro.

$I_1(t)$ = nivel de inventario en el instante t ($0 \leq t \leq t_d$) en el que el producto no experimenta deterioro.

$I_2(t)$ = nivel de inventario en el instante t ($t_d \leq t \leq T$) en el que el producto experimenta deterioro.

$Z(p,T)$ = beneficio total por unidad de tiempo del sistema de inventario.

p^* = precio de venta unitario óptimo.

T^* = tiempo óptimo del ciclo de reposición.

Q^* = cantidad óptima de pedido.

Z^* = beneficio total óptimo por unidad de tiempo, es decir, $Z^* = Z(p^*, T^*)$.

4 FORMULACIÓN DEL MODELO

Se considera un problema de reposición de un único bien que experimenta un deterioro no instantáneo. Al comienzo de cada ciclo llegan al sistema de inventario Q unidades del bien. Existen dos casos posibles en función del valor de T y t_d:

Caso 1: $T \geq t_d$

En este caso, el tiempo de ciclo de reposición, T, es mayor o igual que el periodo en el que el bien permanece sin sufrir deterioro alguno, t_d. Durante el intervalo de tiempo $[0, t_d]$, el nivel de inventario disminuye debido únicamente a la demanda variable. A partir del instante $t = t_d$, el bien comienza a deteriorarse, por lo que en el intervalo $[t_d, T]$, el inventario se reduce no solo debido a la demanda sino también a que los productos van experimentando una pérdida de calidad. Este proceso se repite en el tiempo.

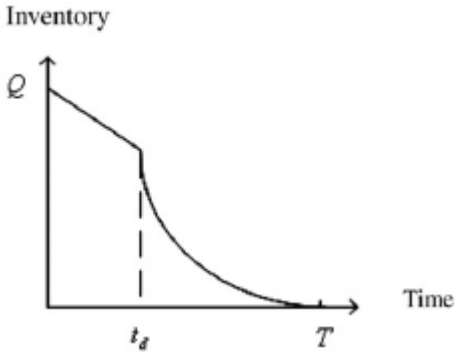


Figura 1: Caso 1 ($T \geq t_d$).

En la Figura 1 se observa lo descrito en el párrafo anterior. En el intervalo $[0, t_d]$ el inventario decrece debido exclusivamente a la demanda variable. Por ello, la ecuación diferencial que representa el nivel de inventario en ese periodo es:

$$\frac{dI_1(t)}{dt} = -D(p), \qquad 0 < t < t_d \tag{1}$$

Resolviendo la Ecuación (1) teniendo en cuenta la condición de contorno $I_1(0) = Q$ queda:

$$I_1(t) = Q - D(p)t, \qquad 0 \leq t \leq t_d \tag{2}$$

También se observa cómo en el intervalo $[t_d, T]$, el inventario disminuye a causa tanto de la demanda como del deterioro de los productos. En este caso, la ecuación diferencial que representa el nivel de inventario es:

$$\frac{dI_2(t)}{dt} + \theta I_2(t) = -D(p), \qquad t_d < t < T \tag{3}$$

Resolviendo la Ecuación (2) teniendo en cuenta la condición de contorno $I_2(T) = 0$ queda:

$$I_2(t) = \frac{D(p)}{\theta}\left[e^{\theta(T-t)} - 1\right], \qquad t_d \le t \le T \tag{4}$$

Para garantizar la continuidad de la función $I(t)$ en $t = t_d$, se igualan las ecuaciones (2) y (4):

$$I_1(t_d) = I_2(t_d) = Q - D(p)t_d = \frac{D(p)}{\theta}\left[e^{\theta(T-t_d)} - 1\right],$$

Lo que implica que la cantidad de pedido en cada ciclo es:

$$Q = \frac{D(p)}{\theta}\left[e^{\theta(T-t_d)} - 1\right] + D(p)t_d \tag{5}$$

Sustituyendo la Ecuación **¡Error! No se encuentra el origen de la referencia.** en la (2):

$$I_1(t) = \frac{D(p)}{\theta}\left[e^{\theta(T-t_d)} - 1\right] + D(p)(t_d - t), \qquad 0 \le t \le t_d \tag{6}$$

Por otra parte, el coste total de inventario y los ingresos por ventas por ciclo se componen de los siguientes cuatro elementos:

i. El coste de pedido es A.

ii. El coste de almacenamiento del inventario:

$$h\left[\int_0^{t_d} I_1(t)dt + \int_{t_d}^{T} I_2(t)dt\right] = hD(p)\left\{\frac{t_d}{\theta}\left[e^{\theta(T-t_d)} - 1\right] + \frac{t_d^2}{2} + \frac{1}{\theta^2}\left[e^{\theta(T-t_d)} - \theta(T-t_d) - 1\right]\right\} \tag{7}$$

iii. El coste de compra:

$$cQ = \frac{cD(p)}{\theta}\left[e^{\theta(T-t_d)} - \theta(T-t_d) - 1\right] + cD(p)T \tag{8}$$

iv. Los ingresos por ventas:

$$p\int_0^T D(p)dt = pD(p)T \tag{9}$$

Con todo, el beneficio total por unidad de tiempo (denotado por $Z_1(p,T)$) viene dado por:

$$
\begin{aligned}
Z_1(p,T) = &\left\{ \begin{matrix} Ingresos\ por\ ventas - Coste\ de\ pedido \\ -Coste\ de\ almacenamiento \\ -Coste\ de\ compra \end{matrix} \right\} \Big/ T \\
&= (p-c)D(p) \\
&- \frac{D(p)}{T}\left\{ \frac{ht_d}{\theta}\left[e^{\theta(T-t_d)} - 1\right] + \frac{ht_d^2}{2} \right. \\
&\left. + \frac{(h+\theta c)}{\theta^2}\left[e^{\theta(T-t_d)} - \theta(T-t_d) - 1\right] \right\} - \frac{A}{T}
\end{aligned}
\tag{10}
$$

Caso 2: $T \leq t_d$

En este caso, el tiempo de ciclo de reposición es menor o igual que el tiempo en el que el producto no se deteriora. Esto implica que la política de reposición óptima para el vendedor es dar salida a todo el stock antes de que su estado comience a empeorar.

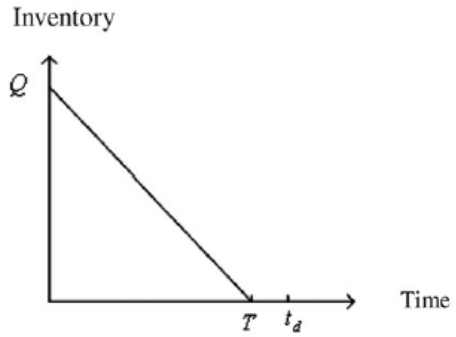


Figura 2. Caso 1 ($T \leq t_d$).

Bajo esta circunstancia, el modelo de inventario no es otro que el modelo tradicional, y el beneficio total por unidad de tiempo (denotado por $Z_2(p,T)$) se obtiene fácilmente mediante la siguiente expresión:

$$
Z_2(p,T) = (p-c)D(p) - \frac{hD(p)T}{2} - \frac{A}{T} \tag{11}
$$

Del razonamiento anterior se deduce que el beneficio total por unidad de tiempo del sistema de inventario es:

$$
Z(p,T) = \begin{cases} Z_1(p,T), & si\ T \geq t_d, \\ Z_2(p,T), & si\ T \leq t_d, \end{cases}
$$

Nótese que, para un valor de p dado, se cumple que $Z_1(p,t_d) = Z_2(p,t_d)$. Por tanto, la función del beneficio por unidad de tiempo $Z(p,T)$ es continua en el punto $T = t_d$.

5 RESULTADOS TEÓRICOS

Para obtener la política de precio y pedidos óptima que maximice el beneficio total por unidad de tiempo, se siguen los procedimientos que, a continuación, se describen, diferenciando los dos casos anteriormente expuestos.

Caso 1: $T = T_1 \geq t_d$

Para maximizar el beneficio total por unidad de tiempo dado por la Ecuación (10) debe cumplirse que las derivadas parciales del beneficio, Z_1, respecto al tiempo de ciclo de reposición, T, y el precio, p, sean iguales a cero simultáneamente. Esto es:

$$\frac{\partial Z_1(p,T)}{\partial T} = 0\,, \qquad \frac{\partial Z_1(p,T)}{\partial p} = 0$$

Desarrollando las expresiones, se obtienen:

$$\frac{\partial Z_1(p,T)}{\partial T} = \frac{D(p)}{T^2}\left\{\frac{(h+\theta h t_d+\theta c)}{\theta^2}\left[e^{\theta(T-t_d)} - \theta T e^{\theta(T-t_d)} - 1\right] + \frac{h t_d^2}{2} + \frac{(h+\theta c)t_d}{\theta}\right\} + \frac{A}{T^2} = 0 \tag{12}$$

$$\frac{\partial Z_1(p,T_1)}{\partial p} = (p-c)D'(p) + D(p) - \frac{D'(p)}{T_1}\left\{\frac{h t_d}{\theta}\left[e^{\theta(T_1-t_d)} - 1\right] + \frac{h t_d^2}{2} + \frac{(h+\theta c)}{\theta^2}\left[e^{\theta(T_1-t_d)} - \theta(T_1-t_d) - 1\right]\right\} = 0 \tag{13}$$

Donde $D'(p)$ es la derivada de la demanda, $D(p)$, respecto al precio, p.

No resulta sencillo encontrar una solución (p,T) a partir de las Ecuaciones (12) y (13). En su lugar, se resuelve el problema utilizando el siguiente procedimiento: Primero, se demuestra que, para un precio p dado, no solo existe un valor óptimo para el tiempo de ciclo de reposición T, sino que este es único. Además, para cualquier valor de T, existe un único p que maximiza el beneficio total por unidad de tiempo.

Siendo la variación de precio $\Delta(p) \equiv 2A - h t_d^2 D(p)$, se tiene el siguiente resultado.

Lema 1: *Para cualquier valor de p, se tiene:*

(a) Si $\Delta(p) \geq 0$, el beneficio total por unidad de tiempo $Z_1(p,T)$ tiene un único máximo global en el punto $T = T_1$, donde $T_1 \in [t_d, \infty)$ y satisface la Ecuación (12).

(b) Si $\Delta(p) < 0$, el beneficio total por unidad de tiempo $Z_1(p,T)$ tiene un máximo en el punto límite $T = t_d$.

Demostración:

(a) Para un valor dado de p, de la Ecuación (12):

$$F(x) = \frac{(h + \theta h t_d + \theta c)}{\theta^2}\left[e^{\theta(x-t_d)} - \theta x e^{\theta(x-t_d)} - 1\right] + \frac{h t_d^2}{2} + \frac{(h+\theta c)t_d}{\theta} + \frac{A}{D(p)}, \quad x \in [t_d, \infty)$$

Tomando la primera derivada de $F(x)$ con respecto a $x \in (t_d, \infty)$, se tiene:

$$\frac{dF(x)}{dx} = -(h + \theta h t_d + \theta c)\, x e^{\theta(x-t_d)} < 0$$

$F(x)$ es una función decreciente de x en el intervalo $[t_d, \infty)$. Además, para $x = t_d$ se tiene $F(t_d) = \frac{A}{D(p)} - \frac{h{t_d}^2}{2} = \frac{\Delta(p)}{2D(p)}$ siendo $\lim_{x\to+\infty} F(x) = -\infty$. Luego, si $\Delta(p) \geq 0$, usando el teorema de Bolzano[1], existe un único valor $T_1 \in [t_d, \infty)$ tal que $F(T_1) = 0$, esto es, T_1 es solución única de la Ecuación (12).

Después, para cualquier valor de p, tomando la segunda derivada de $Z_1(p,T)$ con respecto a T, y evaluándola en T_1, se obtiene:

$$\left.\frac{d^2 Z_1(p,T)}{dT^2}\right|_{T=T_1} = \frac{-(h + \theta h t_d + \theta c)e^{\theta(T_1 - t_d)} D(p)}{T_1} < 0$$

Por tanto, $T_1 \in [t_d, \infty)$ es el máximo global del beneficio total por unidad de tiempo.

(b) Si $\Delta(p) < 0$, entonces se tiene que $F(t_d) < 0$. Dado que $F(x)$ es una función estrictamente decreciente de $x \in [t_d, \infty)$, $F(T) < 0$ para todo $T \in [t_d, \infty)$. Así, para cualquier valor de p:

$$\frac{dZ_1(p,T)}{dT} = \frac{D(p)}{T^2}\left\{\frac{(h + \theta h t_d + \theta c)}{\theta^2}\left[e^{\theta(T-t_d)} - \theta T e^{\theta(T-t_d)} - 1\right] + \frac{h t_d^2}{2} + \frac{(h+\theta c)t_d}{\theta}\right\} + \frac{A}{T^2} = \frac{D(p)F(T)}{T^2} < 0$$

Lo que implica que $Z_1(p,T)$ es una función estrictamente decreciente de $T \in [t_d, \infty)$. Luego $Z_1(p,T)$ tiene un máximo en el punto $T = t_d$, como queda demostrado.

En el Caso 1, para cualquier valor de p, T_1^* denota el valor óptimo del ciclo de reabastecimiento, por lo que, del Lema 1 se tiene:

$$T_1^* = \begin{cases} T_1, & si\ \Delta(p) \geq 0, \\ t_d, & si\ \Delta(p) < 0. \end{cases}$$

A continuación, sustituyendo $T = T_1^*$ en la Ecuación (13) y tras unos cálculos sencillos, el valor de p (denotado por p_1) viene dado por la expresión:

[1] Sea $f: [a,b] \to \mathbb{R}$ una función real continua en $[a,b]$ con $f(a) < 0 < f(b)$ entonces existe al menos un punto $c \in (a,b)$ tal que $f(c) = 0$ (Teorema de Bolzano).

$$p_1 = \frac{\beta}{\beta-1}\left\{\frac{1}{T_1^*}\left\{\frac{ht_d}{\theta}\left[e^{\theta(T_1^*-t_d)}-1\right]+\frac{ht_d^2}{2}+\frac{(h+\theta c)}{\theta^2}\right.\right.$$
$$\left.\left.\times\left[e^{\theta(T_1^*-t_d)}-\theta(T_1^*-t_d)-1\right]\right\}+c\right\} \quad (14)$$

Tomando la segunda derivada de $Z_1(p,T_1^*)$ con respecto a p, y evaluándola en el punto $p=p_1$, se obtiene:

$$\left.\frac{d^2Z_1(p,T_1^*)}{dp^2}\right|_{p=p_1} = \frac{(1-\beta)D(p_1)}{p_1} < 0.$$

Así, $p=p_1$ es la solución óptima que maximiza $Z_1(p,T_1^*)$. De este modo, se obtiene el siguiente resultado.

Lema 2: *Siendo $T_1^* \in [t_d,\infty)$, el beneficio total por unidad de tiempo $Z_1(p,T_1^*)$ tiene un único máximo global en el punto $p=p_1$, que viene dado por la Ecuación (14).*

Caso 2: $T = T_2 \leq t_d$

De formar similar al Caso 1, para p y $T \in (0,t_d]$ dados, tomando las derivadas de primer y segundo orden de $Z_2(p,T)$ en la Ecuación (11) con respecto a T, se tiene:

$$\frac{dZ_2(p,T)}{dT} = -\frac{hD(p)}{2}+\frac{A}{T^2} \quad (15)$$

$$\frac{d^2Z_2(p,T)}{dT^2} = \frac{-2A}{T^3} \quad (16)$$

Dado que la segunda derivada es menor que cero, se puede afirmar que la función es cóncava. Por lo tanto, el valor óptimo de T (denotado por T_2) que maximiza $dZ_2(p,T)$ puede hallarse resolviendo la ecuación $dZ_2(p,T)/dT=0$, quedando:

$$T_2 = \sqrt{\frac{2A}{hD(p)}} \quad (17)$$

Para garantizar que $T_2 \leq t_d$, se sustituye T_2 en la inecuación:

$$\sqrt{\frac{2A}{hD(p)}} \leq t_d$$

Recuperando la definición que anteriormente se dio para $\Delta(p)$ y despejando $D(p)$, se obtiene:

$$\Delta(p) \equiv 2A - ht_d^2D(p) \rightarrow \cdots \rightarrow D(p) = \frac{2A-\Delta(p)}{ht_d^2}$$

Sustituyendo $D(p)$ en la inecuación, se tiene:

$$\sqrt{\frac{2A}{h\dfrac{2A-\Delta(p)}{ht_d^2}}} \le t_d \;\rightarrow\; \frac{2At_d^2}{2A-\Delta(p)} \le t_d^2 \;\rightarrow$$

$$\rightarrow\; 2A \le 2A-\Delta(p) \;\rightarrow\; \Delta(p) \le 0$$

Por tanto, para $\Delta(p) \le 0$ se cumple que $T_2 \le t_d$.

Por otro lado, en el caso de que $\Delta(p) > 0$, despejando el coste de pedido, A:

$$\Delta(p) \equiv 2A - ht_d^2 D(p) > 0 \;\rightarrow\; A > \frac{ht_d^2 D(p)}{2}$$

Sustituyendo en la Ecuación (15), se tiene:

$$\frac{dZ_2(p,T)}{dT} > \frac{-hD(p)}{2} + \frac{ht_d^2 D(p)}{2T^2} = \frac{hD(p)}{2T^2}\left(t_d^2 - T^2\right) \ge 0$$

Lo que implica que $Z_2(p,T)$ es una función estrictamente creciente de $T \in (0, t_d]$. En consecuencia, $Z_2(p,T)$ tiene un máximo en el punto $T = t_d$.

Resumiendo lo anterior, se obtiene el siguiente resultado.

Lema 3: *Para cualquier valor de p, se tiene que:*

(a) Si $\Delta p \le 0$, el beneficio total por unidad de tiempo $Z_2(p,T)$ tiene un único máximo global en el punto $T = T_2$, dado por la Ecuación (17).

(b) Si $\Delta p > 0$, entonces el beneficio total por unidad de tiempo $Z_2(p,T)$ tiene un máximo en el punto límite $T = t_d$.

En el Caso 2, para cualquier valor de p, T_2^* denota el valor óptimo del ciclo de reabastecimiento, por lo que, del Lema 3 se tiene:

$$T_2^* = \begin{cases} T_2, & si\ \Delta(p) \le 0, \\ t_d, & si\ \Delta(p) > 0. \end{cases}$$

A continuación, para un T_2^* dado, la condición necesaria para maximizar $Z_2(p, T_2^*)$ es:

$$\frac{dZ_2(p,T_2^*)}{dp} = (p-c)D'(p) + D(p) - \frac{hD'(p)T_2^*}{2} = 0 \qquad (18)$$

A partir de la Ecuación (18) y tras unos cálculos sencillos, se obtiene el valor de p (denotado por p_2):

$$p_2 = \frac{\beta}{\beta - 1}\left(\frac{hT_2^*}{2} + c\right) \qquad (19)$$

Tomando la derivada de segundo orden de $Z_2(p, T_2^*)$ con respecto a p, y evaluándola en el punto p_2, se obtiene:

$$\left.\frac{d^2 Z_2(p, T_2^*)}{dp^2}\right|_{p=p_2} = \frac{(1-\beta)D(p_2)}{p_2} < 0$$

Así, se tiene que $p = p_2$ es la solución óptima que maximiza $Z_2(p, T_2^*)$. Es decir, se tiene el siguiente resultado.

Lema 4: *Para un valor de T_2^* tal que $T_2^* \in (0, t_d]$, el beneficio total por unidad de tiempo $Z_2(p, T_2^*)$ tiene un único máximo global en el punto $p = p_2$, que viene definido por la Ecuación (19).*

Combinando los Lemas 1 y 3:

Teorema 1*: Para un valor de p dado, se tiene:*

(a) Si $\Delta(p) > 0$, el valor óptimo para el tiempo de ciclo de reposición es T_1.

(b) Si $\Delta(p) < 0$, el valor óptimo para el tiempo de ciclo de reposición es T_2.

(c) Si $\Delta(p) = 0$, el valor óptimo para el tiempo de ciclo de reposición es t_d.

Demostración: Se deduce directamente de los Lemas 1 y 3 y del hecho de que $Z_1(p, t_d) = Z_2(p, t_d)$.

Observación: Cuando $\Delta(p) \equiv 2A - ht_d^2 D(p) < 0$, es decir, $\frac{ht_d^2 D(p)}{2} > A$ (el coste total de almacenamiento durante el periodo de tiempo del ciclo de reposición en el que no se produce deterioro es mayor que el coste de pedido) el distribuidor venderá todo su stock antes de que el producto comience a perder calidad.

Visto todo lo anterior, se puede establecer el siguiente algoritmo que permite obtener la solución óptima (p^*, T^*) del problema estudiado.

Algoritmo:

Paso 1: Se comienza con $j = 0$ y tomando el valor inicial de p_j como $p_j = c$.

Paso 2: Se calcula $\Delta(p_j) = 2A - ht_d^2 D(p_j)$ para un valor dado de p_j.

Paso 2.1: Si $\Delta(p_j) > 0$, determinar el valor de $T_{1,j}$ resolviendo la Ecuación (12). Sustituir $T_{1,j}$ en la Ecuación **¡Error! No se encuentra el origen de la referencia.** para obtener el valor correspondiente del precio de venta unitario: $p_{1,j+1}$. Los valores del precio unitario y el tiempo de ciclo de reposición se actualizan a los calculados: $p_{j+1} = p_{1,j+1}$ y $T_j = T_{1,j}$. Ir al Paso 3.

Paso 2.2: Si $\Delta(p_j) < 0$, determinar el valor de $T_{1,j}$ resolviendo la Ecuación (17). Sustituir $T_{2,j}$ en la Ecuación (19) para obtener el valor correspondiente del precio de venta unitario: $p_{2,j+1}$. Los valores del precio unitario y el tiempo de ciclo de reposición se actualizan a los calculados: $p_{j+1} = p_{2,j+1}$ y $T_j = T_{2,j}$. Ir al Paso 3.

Paso 2.3: Si $\Delta(p_j) = 0$, igualar $T_{1,j}$ (o $T_{2,j}$) a t_d y sustituir el valor en las Ecuaciones (14) ó (19) para obtener el valor correspondiente del precio de venta unitario: $p_{1,j+1}$ (o $p_{2,j+1}$). Los valores del precio unitario y el tiempo de ciclo de reposición se actualizan a los calculados: $p_{j+1} = p_{1,j+1}$ y $T_j = T_{1,j+1}$. Ir al Paso 3.

Paso 3: Si la diferencia entre p_j y p_{j+1} es lo suficientemente pequeña, $|p_j - p_{j+1}| \leq 10^{-5}$, en ese caso se fijan los valores de p y T como óptimos: $p^* = p_j$ y $T^* = T_j$. Así se llega a la solución óptima (p^*, T^*). En otro caso, se incrementa el valor de j en 1, $j = j + 1$ para realizar una nueva iteración,

volviendo al Paso 2.

Una vez se obtiene la solución óptima (p^*, T^*), es posible determinar el tamaño de lote óptimo Q^* a partir de la Ecuación (5) y el valor óptimo de la función objetivo, el beneficio, $Z^* = Z(p^*, T^*)$, usando las Ecuaciones (10) u (11).

Para mayor claridad, se incluye, a continuación, un diagrama de flujo del algoritmo. El capítulo siguiente se centrará en analizar la implementación del algoritmo en Python.

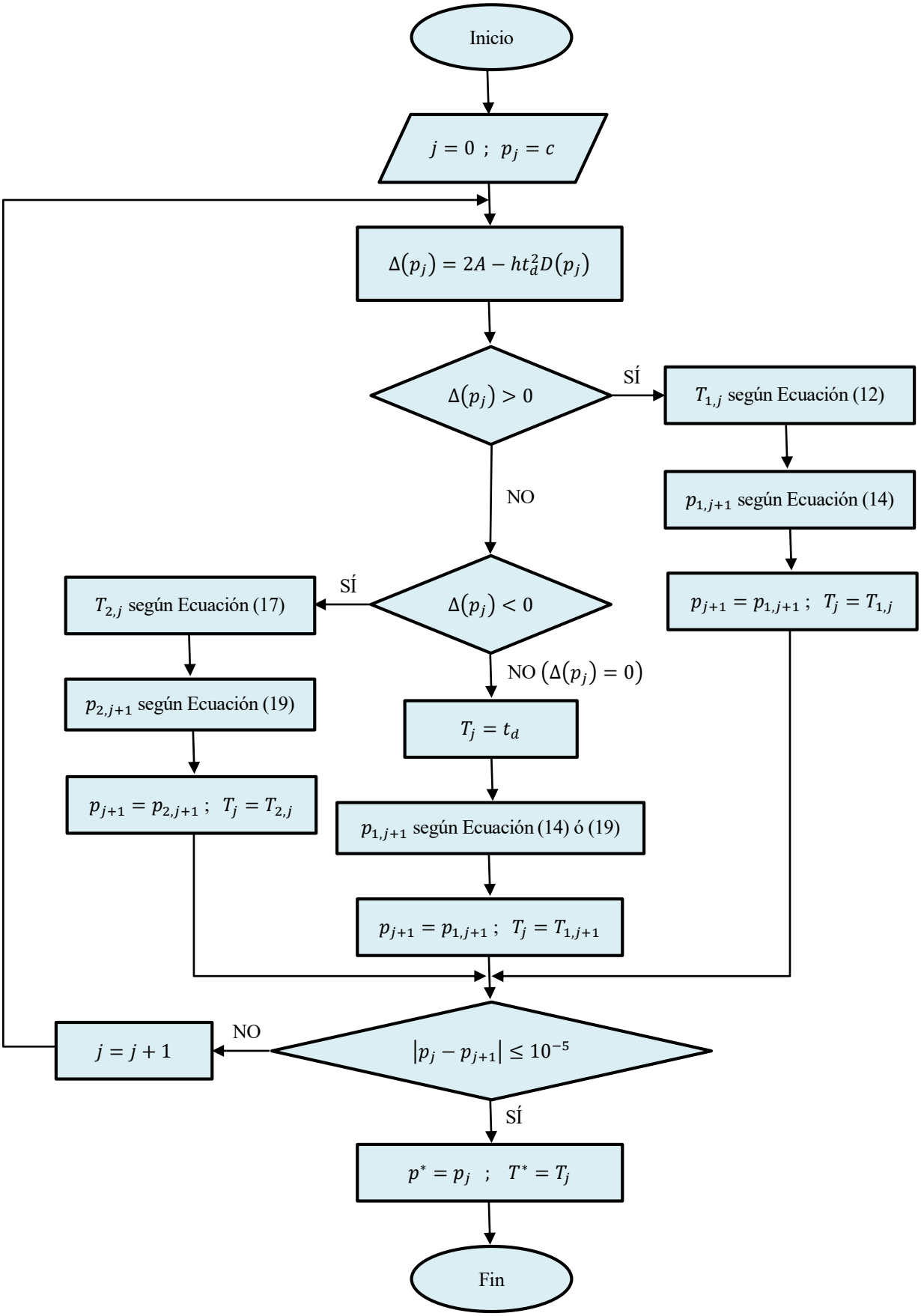


Figura 3. Diagrama de flujo del algoritmo.

6 PROGRAMACIÓN DEL ALGORITMO

Como se indicaba al final del capítulo anterior, la programación del algoritmo se lleva a cabo en Python. Se opta por este lenguaje porque actualmente es uno de los lenguajes de programación más extendidos, lo cual permite encontrar una gran cantidad de material de ayuda en la red, además de contar con una documentación oficial muy extensa y en constante actualización. Python es un lenguaje muy intuitivo gracias a que su sintaxis es más clara y limpia que la de otros lenguajes y tiene la gran ventaja de ser multiplataforma, algo que podría ser útil ya que permitiría hacer uso del código generado en otros sistemas operativos. Al ser un lenguaje interpretado, es suficiente con que el sistema integre un intérprete de Python.

Tanto los ejemplos numéricos como el análisis de sensibilidad se programan por separado en *scripts* independientes.

En los siguientes subcapítulos se detalla la estructura de los códigos implementados.

6.1 Programación Ejemplo 1

En primer lugar, como en cualquier programa, es necesario incluir las librerías que se van a utilizar:

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Dec 21 12:40:05 2020
4
5  @author: Maria
6  """
7
8  import math
9  import sympy
10 import numpy as np
11 from sympy import Symbol
12 from sympy.solvers import solve
13 import re
```

Figura 4. Ejemplo 1: Librerías.

1. Math: Permite el acceso a las funciones matemáticas definidas en el estándar de C.

2. Sympy: Biblioteca de matemáticas simbólicas que funciona como un sistema de álgebra por computadora. De ella se importan la clase Symbol, que nos permitirá definir variables simbólicas (estas variables son un objeto de la clase Symbol) y el solver que nos permitirá resolver una expresión compleja integrada en el algoritmo.

3. Numpy: Significa "Python Numérico" y es la librería que se utiliza para computación científica y análisis de datos.

4. Re: Permite buscar coincidencias de expresiones regulares como patrones o cadenas de texto. En este caso es necesaria para las funciones de lectura de datos desde y escritura de resultados en .txt.

A continuación, se definen las funciones de lectura de datos y escritura de resultados.

La función de lectura de datos recibe como argumento el nombre del fichero donde se recogen los datos. En ella se incluye el código que permite trasladar la información cargada manualmente en el fichero .txt a las variables definidas en el programa. Finalmente, devuelve todas esas variables.

```
def read_datos(filename):

    '''    Lectura Datos Ejemplo 1      '''

    file = open(filename, 'r')
    for filas,line in enumerate(file):
        print("filas:", filas)
        print("line:", line)
        numberline = re.split("\s+", line)
        print("numberline:", numberline)
        if filas == 3:
             datos=(list(map(float, numberline[0:5]))) #Se pone flotante porque algunos valores son decimales
             A = datos[0]
             h = datos[1]
             c = datos[2]
             alpha = datos[3]
             beta = datos[4]
        if filas == 6:
            td_values = (list(map(float, numberline[0:3]))) #dado en días
            size = len(td_values)
            for val in range(0,size):
                td_values[val] = td_values[val]/365 #para convertir td de días a años
            print("td:", td_values)
        if filas == 9:
            theta_values = (list(map(float, numberline[0:3])))

    file.close()
    return A, h, c, alpha, beta, td_values, theta_values
```

Figura 5. Ejemplo 1: Función de lectura de datos.

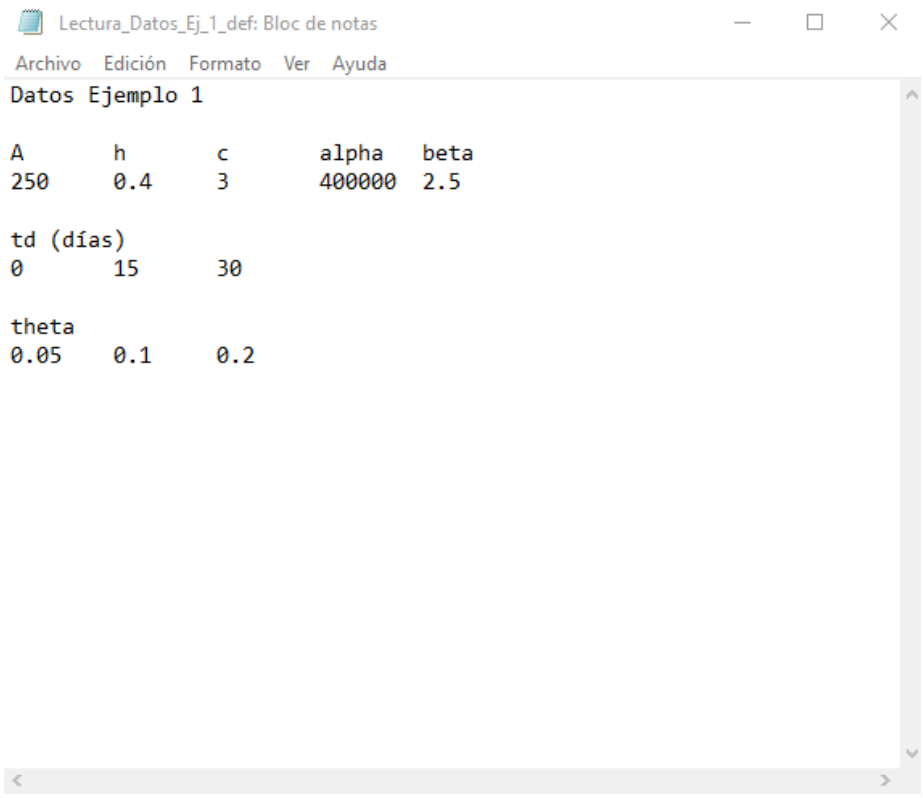


Figura 6. Ejemplo 1: Fichero de datos.

La Figura 7 muestra la función mediante la que se escriben los resultados obtenidos en cada iteración del algoritmo en otro fichero .txt bastante sencillo pero que permite visualizarlos de forma ordenada por casos. Esta función recibe como argumentos tanto los parámetros que determinan los distintos casos de estudio (θ, t_d) como aquellas variables que representan la solución de cada uno de ellos. Ya que los argumentos cambian su valor a lo largo de la ejecución del algoritmo, esta función es llamada tantas veces como casos se analizan. No devuelve ningún valor en Python.

```
46 def write_resultados(theta,td,p_opt,T_opt,Q_opt,Z_opt,it):
47
48     '''    Escritura Resultados Ejemplo 1      '''
49
50     if theta == theta_values[0] and td == td_values[0]:
51         fichero=open("Resultados_Ej_1_def.txt","w")
52         msg = "theta \t\ttd \t\tp* \t\tT* \t\tQ* \t\tZ* \t\tIteraciones\n\n"; fichero.write("%s" %msg);
53         fichero.close()
54
55     fichero=open("Resultados_Ej_1_def.txt","a")
56     if td == td_values[0]:
57         msg = str(theta); fichero.write("%s" %msg); fichero.write("\t\t");
58     else:
59         fichero.write("\t\t")
60
61     msg = str(round(td,4)); fichero.write("%s" %msg); fichero.write("\t\t");
62     msg = str(round(p_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
63     msg = str(round(T_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
64     msg = str(round(Q_opt,2)); fichero.write("%s" %msg); fichero.write("\t\t");
65     msg = str(round(Z_opt,1)); fichero.write("%s" %msg); fichero.write("\t\t");
66     msg = str(it+1); fichero.write("%s" %msg); fichero.write("\n\n");
67
68     fichero.close()
```

Figura 7. Ejemplo 1: Función de escritura de resultados.

El fichero .txt de resultados que se obtiene es el siguiente:

Resultados_Ej_1_def: Bloc de notas

Archivo Edición Formato Ver Ayuda

theta	td	p*	T*	Q*	Z*	Iteraciones
0.05	0.0	5.17031	0.3695	2454.11	12933.0	5
	0.0411	5.16044	0.36934	2460.02	12971.8	5
	0.0822	5.15232	0.37057	2473.79	13006.3	5
0.1	0.0	5.19325	0.32768	2167.96	12756.6	5
	0.0411	5.17354	0.32751	2179.05	12833.5	5
	0.0822	5.15782	0.3298	2203.92	12901.0	5
0.2	0.0	5.23321	0.27475	1803.3	12455.0	6
	0.0411	5.19402	0.27465	1822.88	12606.3	6
	0.0822	5.16447	0.2787	1865.01	12735.6	6

Figura 8. Ejemplo 1: Fichero de resultados.

La parte principal del código comienza con una llamada a la función de lectura de datos anteriormente descrita. De este modo, se obtienen los datos para la posterior ejecución del algoritmo, y se define una matriz de ceros donde se irán guardando los resultados, que finalmente serán extraídos a través del fichero de texto.

Se utiliza un doble bucle *for* que permitirá realizar todas las combinaciones de θ y t_d que se pretenden analizar, donde se hace uso de dos contadores:

- *cont*: indica el número de caso en el que nos encontramos (solo para la visualización en el terminal de Python)

- *it*: cuenta el número de iteraciones que realiza el algoritmo en cada caso.

A continuación, se programan los pasos del algoritmo:

Paso 1: Se pone el contador de iteraciones a cero y se inicializa $p = c$. Se incluyen algunos *print* con la única finalidad de ver cómo va evolucionando el algoritmo a través del terminal de Python mientras se está ejecutando el código.

Paso 2: Este paso se incluye dentro de un bucle *while* que permite volver al segundo *for* y pasar al caso siguiente cuando se llega al óptimo, usando *break*.

La función *p_variation* calcula Δp. Según si el valor obtenido es mayor, menor o igual a cero, se utilizan las ecuaciones correpondientes para calcular el tiempo de ciclo de reposición y el precio.

El caso más complejo es en el que $\Delta p > 0$, donde para el cálculo de T es necesario recurrir a una función *solver* de la librería Sympy y trabajar con variables simbólicas.

Paso 3: Está integrado en cada uno de los *if* del paso 2. En él se comprueba si la diferencia entre el nuevo valor de p que se acaba de calcular y el anterior es menor que $1 \cdot 10^{-5}$, en cuyo caso se toman los valores de T y p calculados cómo los óptimos, T^* y p^*, y se calculan la cantidad de pedido y el beneficio óptimos, Q^* y Z^*. Finalmente se llama a la función que escribe el fichero de resultados. Se actualiza el valor de *cont* para indicar que pasamos al siguiente caso y se utiliza un *break* para salir del bucle *while*. En caso de que la diferencia sea mayor que $1 \cdot 10^{-5}$, habría que iterar de nuevo, por lo que se aumenta el contador de iteraciones *it* y se actualiza el valor de p al calculado en esta iteración.

En la Figura 9 se muestra el código completo del Ejemplo 1.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 21 12:40:05 2020

@author: Maria
"""

import math
import sympy
import numpy as np
from sympy import Symbol
from sympy.solvers import solve
import re


def read_datos(filename):

    '''    Lectura Datos Ejemplo 1      '''

    file = open(filename, 'r')
    for filas,line in enumerate(file):
        print("filas:", filas)
        print("line:", line)
        numberline = re.split("\s+", line)
        print("numberline:", numberline)
        if filas == 3:
             datos=(list(map(float, numberline[0:5]))) #Se pone flotante porque algunos valores son decimales
             A = datos[0]
             h = datos[1]
             c = datos[2]
             alpha = datos[3]
             beta = datos[4]
        if filas == 6:
            td_values = (list(map(float, numberline[0:3]))) #dado en días
            size = len(td_values)
            for val in range(0,size):
                td_values[val] = td_values[val]/365 #para convertir td de días a años
            print("td:", td_values)
        if filas == 9:
            theta_values = (list(map(float, numberline[0:3])))

    file.close()
    return A, h, c, alpha, beta, td_values, theta_values


def write_resultados(theta,td,p_opt,T_opt,Q_opt,Z_opt,it):

    '''    Escritura Resultados Ejemplo 1      '''

    if theta == theta_values[0] and td == td_values[0]:
        fichero=open("Resultados_Ej_1_def.txt","w")
        msg = "theta \t\ttd \t\tp* \t\tT* \t\tQ* \t\tZ* \t\tIteraciones\n\n"; fichero.write("%s" %msg); #Escribe la cabecera
        fichero.close()

    fichero=open("Resultados_Ej_1_def.txt","a")
    if td == td_values[0]:
        msg = str(theta); fichero.write("%s" %msg); fichero.write("\t\t");
    else:
        fichero.write("\t\t")

    msg = str(round(td,4)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(round(p_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(round(T_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(round(Q_opt,2)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(round(Z_opt,1)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(it+1); fichero.write("%s" %msg); fichero.write("\n\n");

    fichero.close()


''' FUNCIÓN PRINCIPAL '''

A, h, c, alpha, beta, td_values, theta_values = read_datos("Lectura_Datos_Ej_1_def.txt")

matriz_resultados = np.zeros(((len(theta_values)*len(td_values)),6))


cont = 0

for theta in theta_values:

    for td in td_values:

        #Step 1
        print("\n")
        print("CASO %d:" % (cont+1))
        print("td = ", td)
```

```
print("theta = ",theta)

it = 0
p = c

while True:

    print("\n")
    print("Iteración %d (Z*)" % (it+1))
    print("td = ", td)
    print("theta = ",theta)


    #Step 2
    p_variation = 2*A - h*(td**2)*alpha*p**(-beta)

    print("p_variation = ", p_variation)

    if p_variation > 0:

        x = Symbol('x')
        T1_lista = solve((alpha*p**(-beta)/x**2)*(((h+theta*h*td+theta*c)/theta**2)*(sympy.exp(theta*(x-td))-theta*x*sympy
        T1 = T1_lista[-1] # x tiene dos soluciones, nos quedamos con la última componente, que es la positiva

        p1 = (beta/(beta-1))*(1/T1*(h*td/theta*(sympy.exp(theta*(T1-td))-1)+h*td**2/2+(h+theta*c)/theta**2*(sympy.exp(thet

        #Step 3
        if abs(p-p1) <= 1e-5:

            p_opt = p1
            T_opt = T1
            Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
            Z1 = (p_opt-c)*alpha*p_opt**(-beta)-((alpha*p_opt**(-beta))/T_opt)*((h*td/theta)*(math.exp(theta*(T_opt-td))-1

            Z_opt = Z1

            write_resultados(theta,td,p_opt,T_opt,Q_opt,Z_opt,it)

            print("El óptimo de la F.O. es: ", Z_opt)
            print("p_opt = ", p_opt)
            print("T_opt = ", T_opt)
            print("Q_opt = ", Q_opt)

            resultados_caso = np.array([theta,td,p_opt,T_opt,Q_opt,Z_opt])
            matriz_resultados[cont] = resultados_caso
            print("Resultados caso:", resultados_caso)
            print("Resultados acum.:", matriz_resultados)

            cont = cont + 1

            break

        else:
            it = it+1
            p = p1

    else:

        if p_variation < 0:

            T2 = math.sqrt((2*A)/(h*alpha+p**(-beta)))
            p2 = (beta/(beta-1))*((h*T2)/2+c)

            #Step 3
            if abs(p-p2) <= 1e-5:

                p_opt = p2
                T_opt = T2
                Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
                Z2 = (p_opt-c)*alpha*p_opt**(-beta)-(h*alpha*(p_opt**(-beta))*T_opt)/2-A/T_opt

                Z_opt = Z2

                write_resultados(theta,td,p_opt,T_opt,Q_opt,Z_opt,it)

                print("El óptimo de la F.O. es: ", Z_opt)
                print("p_opt = ", p_opt)
                print("T_opt = ", T_opt)
                print("Q_opt = ", Q_opt)

                resultados_caso = np.array([theta,td,p_opt,T_opt,Q_opt,Z_opt])
                matriz_resultados[cont] = resultados_caso
                print("Resultados caso:", resultados_caso)
                print("Resultados acum.:", matriz_resultados)

                cont = cont + 1
```

```
            break

        else:
            it = it+1
            p = p2

    else:

        if p_variation == 0:

            T3 = td
            p3 = (beta/(beta-1))*((h*T3)/2+c)

            #Step 3
            if abs(p-p3) <= 1e-5:

                p_opt = p3
                T_opt = T3
                Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
                #Z se puede calcular por cualquiera de las dos expresiones
                Z1 = (p_opt-c)*alpha*p_opt**(-beta)-((alpha*p_opt**(-beta))/T_opt)*((h*td/theta)*(math.exp(theta*(T_opt-td))-
                Z2 = (p_opt-c)*alpha*p_opt**(-beta)-(h*alpha*(p_opt**(-beta))*T_opt)/2-A/T_opt

                Z_opt = Z1

                write_resultados(theta,td,p_opt,T_opt,Q_opt,Z_opt,it)

                print("El óptimo de la F.O. es: ", Z_opt)
                print("p_opt = ", p_opt)
                print("T_opt = ", T_opt)
                print("Q_opt = ", Q_opt)

                resultados_caso = np.array([theta,td,p_opt,T_opt,Q_opt,Z_opt])
                matriz_resultados[cont] = resultados_caso
                print("Resultados caso:", resultados_caso)
                print("Resultados acum.:", matriz_resultados)

                cont = cont + 1

                break

            else:
                it = it+1
                p = p3
```

Figura 9. Código Ejemplo 1.

6.2 Programación Ejemplo 2

La programación del Ejemplo 2 se va a obviar, ya que es muy similar a la del Ejemplo 1. La principal diferencia entre ambos son las variables que definen los casos de estudio (se tratará más detalladamente en el Capítulo 7) por lo que los archivos de texto involucrados son ligeramente diferentes:

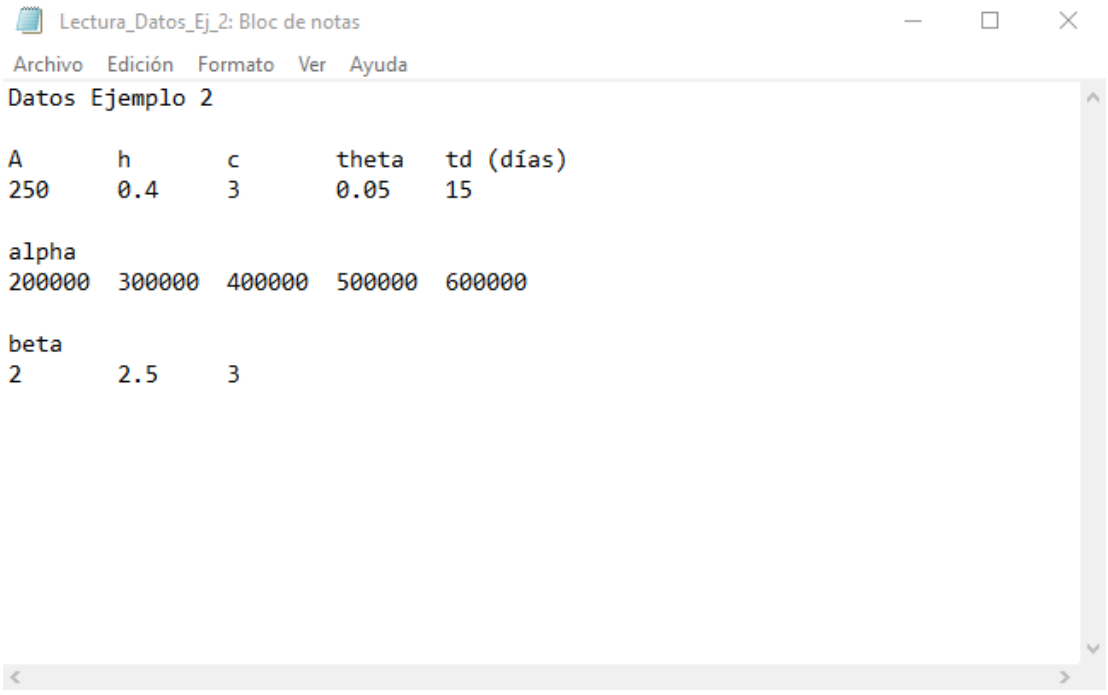
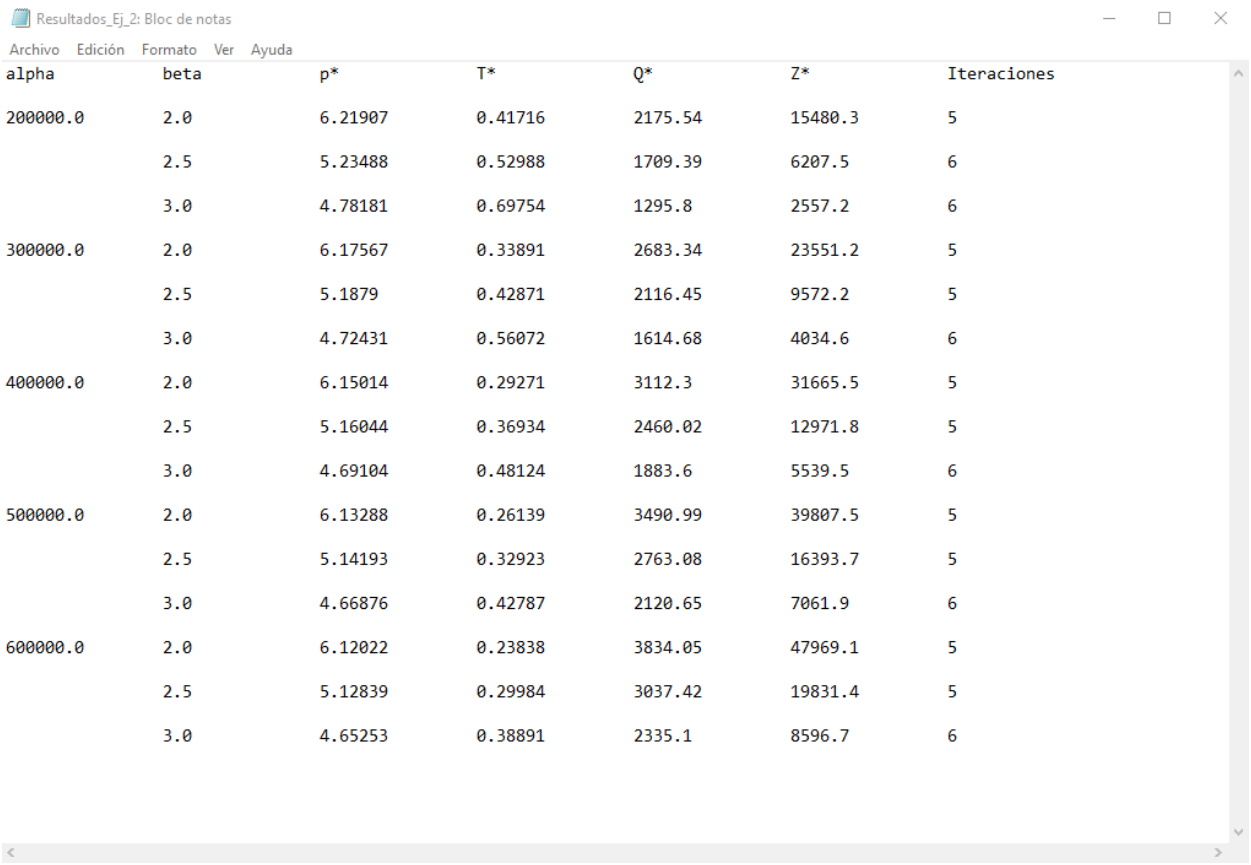


Figura 10. Ejemplo 2: Fichero de datos.



alpha	beta	p*	T*	Q*	Z*	Iteraciones
200000.0	2.0	6.21907	0.41716	2175.54	15480.3	5
	2.5	5.23488	0.52988	1709.39	6207.5	6
	3.0	4.78181	0.69754	1295.8	2557.2	6
300000.0	2.0	6.17567	0.33891	2683.34	23551.2	5
	2.5	5.1879	0.42871	2116.45	9572.2	5
	3.0	4.72431	0.56072	1614.68	4034.6	6
400000.0	2.0	6.15014	0.29271	3112.3	31665.5	5
	2.5	5.16044	0.36934	2460.02	12971.8	5
	3.0	4.69104	0.48124	1883.6	5539.5	6
500000.0	2.0	6.13288	0.26139	3490.99	39807.5	5
	2.5	5.14193	0.32923	2763.08	16393.7	5
	3.0	4.66876	0.42787	2120.65	7061.9	6
600000.0	2.0	6.12022	0.23838	3834.05	47969.1	5
	2.5	5.12839	0.29984	3037.42	19831.4	5
	3.0	4.65253	0.38891	2335.1	8596.7	6

Figura 11. Ejemplo 2: Fichero de resultados.

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Jan 24 19:22:54 2021

@author: Maria
"""

import math
import sympy
import numpy as np
from sympy import Symbol
from sympy.solvers import solve
import re


def read_datos(filename):

    '''    Lectura Datos Ejemplo 2      '''

    file = open(filename, 'r')
    for filas,line in enumerate(file):
        print("filas:", filas)
        print("line:", line)
        numberline = re.split("\s+", line)
        print("numberline:", numberline)
        if filas == 3:
             datos=(list(map(float, numberline[0:5]))) #Se pone flotante porque algunos valores son decimales
             A = datos[0]
             h = datos[1]
             c = datos[2]
             theta = datos[3]
             td = datos[4]/365 #viene dado en días, lo pasamos a años
        if filas == 6:
            alpha_values = (list(map(float, numberline[0:5])))

        if filas == 9:
            beta_values = (list(map(float, numberline[0:3])))

    file.close()
    return A, h, c, theta, td, alpha_values, beta_values


def write_resultados(alpha,beta,p_opt,T_opt,Q_opt,Z_opt):

    if alpha == alpha_values[0] and beta == beta_values[0]:
        fichero=open("Resultados_Ej_2_def.txt","w")
        msg = "alpha \t\tbeta \t\tp* \t\tT* \t\tQ* \t\tZ* \t\tIteraciones\n\n"; fichero.write("%s" %msg); #Escribe la cabecera
        fichero.close()

    fichero=open("Resultados_Ej_2_def.txt","a")
    if beta == beta_values[0]:
        msg = str(alpha); fichero.write("%s" %msg); fichero.write("\t");
    else:
        fichero.write("\t\t")

    msg = str(round(beta,5)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(round(p_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(round(T_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(round(Q_opt,2)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(round(Z_opt,1)); fichero.write("%s" %msg); fichero.write("\t\t");
    msg = str(it+1); fichero.write("%s" %msg); fichero.write("\n\n");

    fichero.close()


A, h, c, theta, td, alpha_values, beta_values = read_datos("Lectura_Datos_Ej_2_def.txt")

matriz_resultados = np.zeros(((len(alpha_values)*len(beta_values)),6))
cont = 0


for alpha in alpha_values:

    for beta in beta_values:

        #Step 1
        print("\n")
        print("CASO %d:" % (cont+1))
        print("alpha = ", alpha)
        print("beta = ",beta)

        it = 0
        p = c

        while True:

            print("\n")
```

```
print("Iteración %d (Z*)" % (it+1))
print("alpha = ", alpha)
print("beta = ", beta)


#Step 2
p_variation = 2*A - h*(td**2)*alpha*p**(-beta)

print("p_variation = ", p_variation)

if p_variation > 0:

    x = Symbol('x')
    T1_lista = solve((alpha*p**(-beta)/x**2)*(((h+theta*h*td+theta*c)/theta**2)*(sympy.exp(theta*(x-td))-theta*x*sympy.exp(th
    #print("T1 lista = ", T1_lista) #x tiene dos soluciones, nos quedamos con la positiva
    T1 = T1_lista[-1] #nos quedamos con la última componente, que es la positiva
    #print("T1 = ", T1)

    p1 = (beta/(beta-1))*(1/T1*(h*td/theta*(sympy.exp(theta*(T1-td))-1)+h*td**2/2+(h+theta*c)/theta**2*(sympy.exp(theta*(T1-t

    #Step 3
    if abs(p-p1) <= 1e-5:

        p_opt = p1
        T_opt = T1
        Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
        Z1 = (p_opt-c)*alpha*p_opt**(-beta)-((alpha*p_opt**(-beta))/T_opt)*((h*td/theta)*(math.exp(theta*(T_opt-td))-1)+(h*td

        Z_opt = Z1

        write_resultados(alpha,beta,p_opt,T_opt,Q_opt,Z_opt)

        print("El óptimo de la F.O. es: ", Z_opt)
        print("p_opt = ", p_opt)
        print("T_opt = ", T_opt)
        print("Q_opt = ", Q_opt)

        resultados_caso = np.array([alpha,beta,p_opt,T_opt,Q_opt,Z_opt])
        matriz_resultados[cont] = resultados_caso
        print("Resultados caso:", resultados_caso)
        print("Resultados acum.:", matriz_resultados)

        cont = cont+1

        break

    else:
        it = it+1
        p = p1

else:

    if p_variation < 0:

        T2 = math.sqrt((2*A)/(h*alpha+p**(-beta)))
        p2 = (beta/(beta-1))*((h*T2)/2+c)

        #Step 3
        if abs(p-p2) <= 1e-5:

            p_opt = p2
            T_opt = T2
            Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
            Z2 = (p_opt-c)*alpha*p_opt**(-beta)-(h*alpha*(p_opt**(-beta))*T_opt)/2-A/T_opt

            Z_opt = Z2

            write_resultados(alpha,beta,p_opt,T_opt,Q_opt,Z_opt)

            print("El óptimo de la F.O. es: ", Z_opt)
            print("p_opt = ", p_opt)
            print("T_opt = ", T_opt)
            print("Q_opt = ", Q_opt)

            resultados_caso = np.array([alpha,beta,p_opt,T_opt,Q_opt,Z_opt])
            matriz_resultados[cont] = resultados_caso
            print("Resultados caso:", resultados_caso)
            print("Resultados acum.:", matriz_resultados)

            cont = cont+1

            break

        else:
            it = it+1
            p = p2
```

```
else:

    if p_variation == 0:

        T3 = td
        p3 = (beta/(beta-1))*((h*T3)/2+c)

        #Step 3
        if abs(p-p3) <= 1e-5:

            p_opt = p3
            T_opt = T3
            Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
            #Z se puede calcular por cualquiera de las dos expresiones
            Z1 = (p_opt-c)*alpha*p_opt**(-beta)-((alpha*p_opt**(-beta))/T_opt)*((h*td/theta)*(math.exp(theta*(T_opt-td
            Z2 = (p_opt-c)*alpha*p_opt**(-beta)-(h*alpha*(p_opt**(-beta))*T_opt)/2-A/T_opt
            print("Z1 = ", Z1)
            print("Z2 = ", Z2)

            Z_opt = Z1

            write_resultados(alpha,beta,p_opt,T_opt,Q_opt,Z_opt)

            print("El óptimo de la F.O. es: ", Z_opt)
            print("p_opt = ", p_opt)
            print("T_opt = ", T_opt)
            print("Q_opt = ", Q_opt)

            resultados_caso = np.array([alpha,beta,p_opt,T_opt,Q_opt,Z_opt])
            matriz_resultados[cont] = resultados_caso
            print("Resultados caso:", resultados_caso)
            print("Resultados acum.:", matriz_resultados)

            cont = cont + 1

            break

        else:
            it = it+1
            p = p3
```

Figura 12. Código Ejemplo 2.

6.3 Programación Análisis de Sensibilidad

Este código presenta una estructura similar a los de los Ejemplos 1 y 2, con ligeras diferencias.

Se importan las mismas librerías que en los programas anteriores, a excepción de Numpy, ya que en este caso no se hace uso de *arrays*.

El fichero de lectura de datos es el que se muestra en la Figura 13. De él se extraen los datos mediante la función correspondiente *read_datos* como en los casos anteriores:

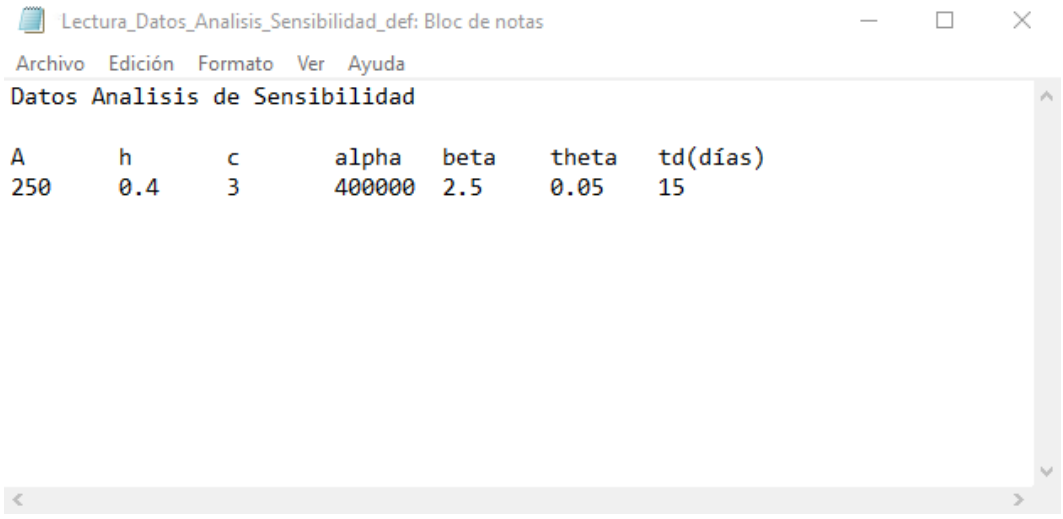
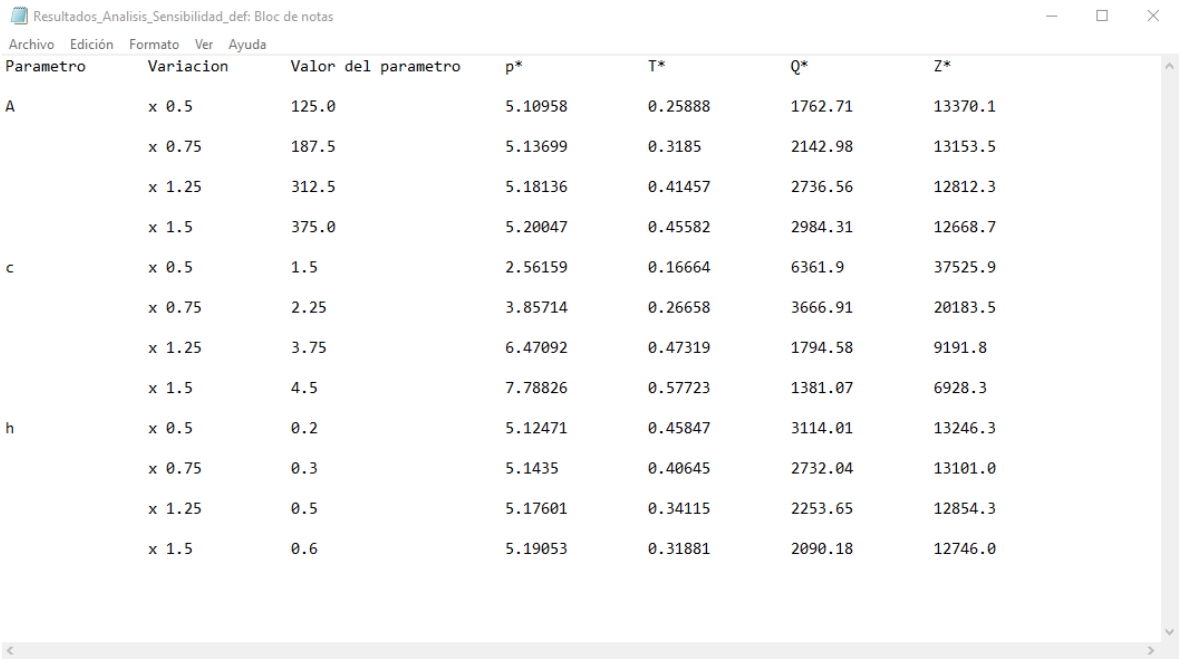


Figura 13. Análisis de Sensibilidad: Fichero de datos.

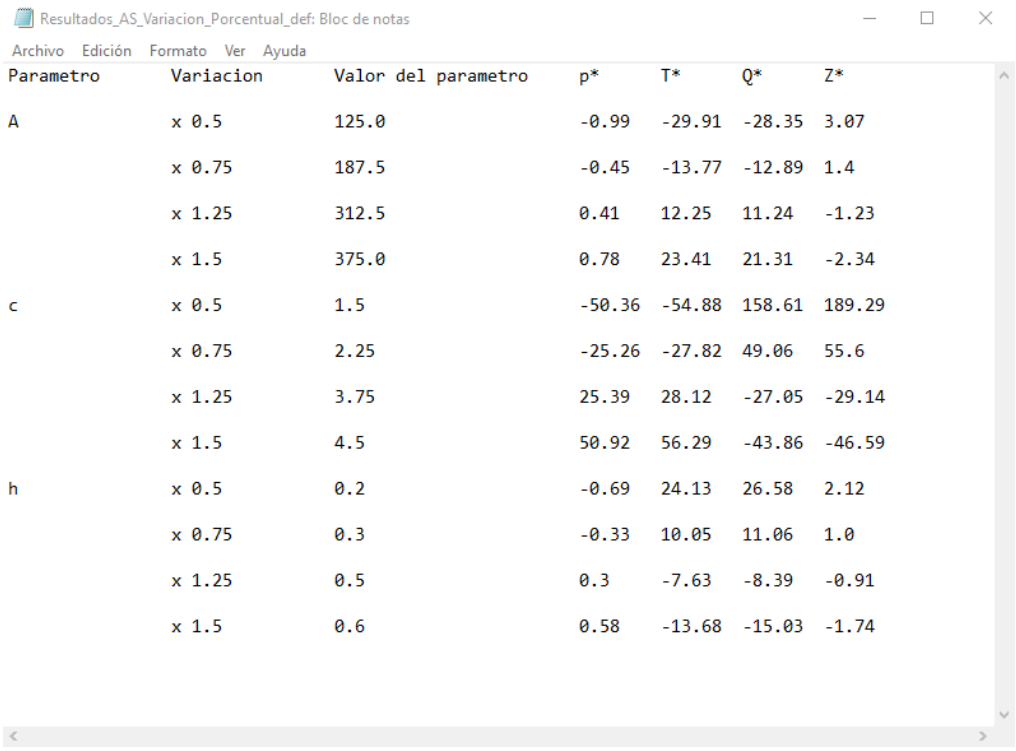
En el análisis de sensibilidad se generan dos ficheros de resultados. Uno de ellos presenta los valores que toman todas las variables al aumentar o disminuir los parámetros A, c y h. El otro proporciona la variación porcentual que experimentan todos los parámetros del modelo. Para obtener estos ficheros se definen dos funciones de escritura de datos: *write_resultados* y *variacion_porcentual*. Aquí las funciones de escritura son un poco más complejas ya que es necesario hacer distinción entre las tres variables modifican su valor para que el fichero obtenido tenga una estructura lógica. Para ello se incorporan tres *if* que diferencian si el parámetro que se está variando es A, c o h. En el caso de la función *variación_porcentual* además es necesario incluir p^*, T^*, Q^* y Z^* inicializados a los valores que toman cuando A, c y h tienen sus valores iniciales (para mayor claridad, ver la Figura 18). Estos datos se toman de cualquiera de los dos ejemplos que ya se han visto. De este modo se pueden obtener las variaciones porcentuales aplicando la fórmula correspondiente (la que se indica en el Capítulo 8).


Resultados_Analisis_Sensibilidad_def: Bloc de notas

Archivo Edición Formato Ver Ayuda

Parametro	Variacion	Valor del parametro	p*	T*	Q*	Z*
A	x 0.5	125.0	5.10958	0.25888	1762.71	13370.1
	x 0.75	187.5	5.13699	0.3185	2142.98	13153.5
	x 1.25	312.5	5.18136	0.41457	2736.56	12812.3
	x 1.5	375.0	5.20047	0.45582	2984.31	12668.7
c	x 0.5	1.5	2.56159	0.16664	6361.9	37525.9
	x 0.75	2.25	3.85714	0.26658	3666.91	20183.5
	x 1.25	3.75	6.47092	0.47319	1794.58	9191.8
	x 1.5	4.5	7.78826	0.57723	1381.07	6928.3
h	x 0.5	0.2	5.12471	0.45847	3114.01	13246.3
	x 0.75	0.3	5.1435	0.40645	2732.04	13101.0
	x 1.25	0.5	5.17601	0.34115	2253.65	12854.3
	x 1.5	0.6	5.19053	0.31881	2090.18	12746.0

Figura 14. Análisis de Sensibilidad: Resultados.



Resultados_AS_Variacion_Porcentual_def: Bloc de notas

Archivo Edición Formato Ver Ayuda

Parametro	Variacion	Valor del parametro	p*	T*	Q*	Z*
A	x 0.5	125.0	-0.99	-29.91	-28.35	3.07
	x 0.75	187.5	-0.45	-13.77	-12.89	1.4
	x 1.25	312.5	0.41	12.25	11.24	-1.23
	x 1.5	375.0	0.78	23.41	21.31	-2.34
c	x 0.5	1.5	-50.36	-54.88	158.61	189.29
	x 0.75	2.25	-25.26	-27.82	49.06	55.6
	x 1.25	3.75	25.39	28.12	-27.05	-29.14
	x 1.5	4.5	50.92	56.29	-43.86	-46.59
h	x 0.5	0.2	-0.69	24.13	26.58	2.12
	x 0.75	0.3	-0.33	10.05	11.06	1.0
	x 1.25	0.5	0.3	-7.63	-8.39	-0.91
	x 1.5	0.6	0.58	-13.68	-15.03	-1.74

Figura 15. Análisis de Sensibilidad: Variaciones porcentuales.

En cuanto a la función principal, lo que la diferencia de las funciones principales de los códigos anteriores es que se incluye una primera parte en la que se definen dos vectores:

- *vector_variacion*: incluye las variaciones a aplicar en tanto por uno: 0,5 (-50%); 0,75 (-25%); 1,25 (+25%); 1,50 (+50%).

- *vector_parametros*: guarda los valores iniciales de A, c y h, que se leen directamente del fichero de datos.

```
170 vector_variacion = [0.5, 0.75, 1.25, 1.5]
171 vector_parametros = [A_inicial, c_inicial, h_inicial]
```

Figura 16. Vectores variación parámetros A, c y h.

Estos vectores, además de utilizarse en la función principal, son los que se usan para escribir los ficheros de resultados pudiendo diferenciar los casos.

A continuación, dentro del doble bucle *for* y, nuevamente, mediante unos *if* se determina qué variable, A, c o h va ser modificada, haciendo posible que se den todas las combinaciones entre parámetro e incremento/decremento que se le aplica.

```
176 for parametro in vector_parametros:
177
178     for variacion in vector_variacion:
179
180         #Step 1
181         print("\n")
182         print("CASO %d:" % (cont+1))
183
184         if parametro == vector_parametros[0]:
185             A = parametro*variacion
186             c = c_inicial
187             h = h_inicial
188             print("Valor de A = ", A)
189             print("Valor de c = ", c)
190             print("Valor de h = ", h)
191
192         if parametro == vector_parametros[1]:
193                 A = A_inicial
194                 c = parametro*variacion
195                 h = h_inicial
196                 print("Valor de A = ", A)
197                 print("Valor de c = ", c)
198                 print("Valor de h = ", h)
199
200         if parametro == vector_parametros[2]:
201                 A = A_inicial
202                 c = c_inicial
203                 h = parametro*variacion
204                 print("Valor de A = ", A)
205                 print("Valor de c = ", c)
206                 print("Valor de h = ", h)
```

Figura 17. Triple *if* para la diferenciación del parámetro incrementado/decrementado.

El resto del código es muy similar al de los Ejemplos 1 y 2. La Figura 18 presenta el código completo.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar  4 17:09:51 2021

@author: Maria
"""

import math
import sympy
from sympy import Symbol
from sympy.solvers import solve
import re


def read_datos(filename):

    '''    Lectura Datos Análisis de Sensibilidad      '''

    file = open(filename, 'r')
    for filas,line in enumerate(file):
        print("filas:", filas)
        print("line:", line)
        numberline = re.split("\s+", line)
        print("numberline:", numberline)
        if filas == 3:
             datos=(list(map(float, numberline[0:7]))) #Se pone flotante porque algunos valores son decimales
             #print("Datos:", datos)
             A_inicial = datos[0]
             h_inicial = datos[1]
             c_inicial = datos[2]
             alpha = datos[3]
             beta = datos[4]
             theta = datos[5]
             td = datos[6]/365 #viene dado en días, lo pasamos a años
             #print("Datos:", A,h,c,alpha,beta)

    file.close()
    return A_inicial, h_inicial, c_inicial, alpha, beta, theta, td


def write_resultados(A,c,h,p_opt,T_opt,Q_opt,Z_opt):

    if parametro == vector_parametros[0]:
        if variacion == vector_variacion[0]:
            fichero=open("Resultados_Analisis_Sensibilidad_def.txt","w")
            msg = "Parametro \tVariacion \tValor del parametro \tp* \t\tT* \t\tQ* \t\tZ*\n\n"; fichero.write("%s" %msg); #Escrib
            msg = "A\t\t"; fichero.write("%s" %msg);
        else:
            fichero=open("Resultados_Analisis_Sensibilidad_def.txt","a")
            fichero.write("\t\t")

        msg = "x "; fichero.write("%s"%msg);
        msg = str(variacion); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(A,5)); fichero.write("%s" %msg); fichero.write("\t\t\t");
        msg = str(round(p_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(T_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(Q_opt,2)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(Z_opt,1)); fichero.write("%s" %msg); fichero.write("\n\n");

        fichero.close()

    if parametro == vector_parametros[1]:
        if variacion == vector_variacion[0]:
            fichero=open("Resultados_Analisis_Sensibilidad_def.txt","a")
            msg = "c\t\t"; fichero.write("%s" %msg);
        else:
            fichero=open("Resultados_Analisis_Sensibilidad_def.txt","a")
            fichero.write("\t\t")

        msg = "x "; fichero.write("%s"%msg);
        msg = str(variacion); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(c,5)); fichero.write("%s" %msg); fichero.write("\t\t\t");
        msg = str(round(p_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(T_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(Q_opt,2)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(Z_opt,1)); fichero.write("%s" %msg); fichero.write("\n\n");

        fichero.close()

    if parametro == vector_parametros[2]:
        if variacion == vector_variacion[0]:
            fichero=open("Resultados_Analisis_Sensibilidad_def.txt","a")
            msg = "h\t\t"; fichero.write("%s" %msg);
        else:
            fichero=open("Resultados_Analisis_Sensibilidad_def.txt","a")
            fichero.write("\t\t")
```

```
        msg = "x "; fichero.write("%s"%msg);
        msg = str(variacion); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(h,5)); fichero.write("%s" %msg); fichero.write("\t\t\t");
        msg = str(round(p_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(T_opt,5)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(Q_opt,2)); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(Z_opt,1)); fichero.write("%s" %msg); fichero.write("\n\n");

        fichero.close()

def variacion_porcentual(A,c,h,p_opt,T_opt,Q_opt,Z_opt):
    "Valores del caso sin variar los parámetros, tomados de los resultados del Ejemplo 1"
    p_opt_ej1 = 5.16044
    T_opt_ej1 = 0.36934
    Q_opt_ej1 = 2460.01833
    Z_opt_ej1 = 12971.77824

    p_opt_var = (p_opt-p_opt_ej1)/p_opt_ej1*100
    T_opt_var = (T_opt-T_opt_ej1)/T_opt_ej1*100
    Q_opt_var = (Q_opt-Q_opt_ej1)/Q_opt_ej1*100
    Z_opt_var = (Z_opt-Z_opt_ej1)/Z_opt_ej1*100

    if parametro == vector_parametros[0]:
        if variacion == vector_variacion[0]:
            fichero=open("Resultados_AS_Variacion_Porcentual_def.txt","w")
            msg = "Parametro \tVariacion \tValor del parametro \tp* \tT* \tQ* \tZ*\n\n"; fichero.write("%s" %msg); #Escribe la ca
            msg = "A\t\t"; fichero.write("%s" %msg);
        else:
            fichero=open("Resultados_AS_Variacion_Porcentual_def.txt","a")
            fichero.write("\t\t")

        msg = "x "; fichero.write("%s"%msg);
        msg = str(variacion); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(A,5)); fichero.write("%s" %msg); fichero.write("\t\t\t");
        msg = str(round(p_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(T_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(Q_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(Z_opt_var,2)); fichero.write("%s" %msg); fichero.write("\n\n");

        fichero.close()

    if parametro == vector_parametros[1]:
        if variacion == vector_variacion[0]:
            fichero=open("Resultados_AS_Variacion_Porcentual_def.txt","a")
            msg = "c\t\t"; fichero.write("%s" %msg);
        else:
            fichero=open("Resultados_AS_Variacion_Porcentual_def.txt","a")
            fichero.write("\t\t")

        msg = "x "; fichero.write("%s"%msg);
        msg = str(variacion); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(c,5)); fichero.write("%s" %msg); fichero.write("\t\t\t");
        msg = str(round(p_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(T_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(Q_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(Z_opt_var,2)); fichero.write("%s" %msg); fichero.write("\n\n");

        fichero.close()

    if parametro == vector_parametros[2]:
        if variacion == vector_variacion[0]:
            fichero=open("Resultados_AS_Variacion_Porcentual_def.txt","a")
            msg = "h\t\t"; fichero.write("%s" %msg);
        else:
            fichero=open("Resultados_AS_Variacion_Porcentual_def.txt","a")
            fichero.write("\t\t")

        msg = "x "; fichero.write("%s"%msg);
        msg = str(variacion); fichero.write("%s" %msg); fichero.write("\t\t");
        msg = str(round(h,5)); fichero.write("%s" %msg); fichero.write("\t\t\t");
        msg = str(round(p_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(T_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(Q_opt_var,2)); fichero.write("%s" %msg); fichero.write("\t");
        msg = str(round(Z_opt_var,2)); fichero.write("%s" %msg); fichero.write("\n\n");

        fichero.close()


''' FUNCIÓN PRINCIPAL '''

A_inicial, h_inicial, c_inicial, alpha, beta, theta, td = read_datos("Lectura_Datos_Analisis_Sensibilidad_def.txt")

vector_variacion = [0.5, 0.75, 1.25, 1.5]
vector_parametros = [A_inicial, c_inicial, h_inicial]

cont = 0
```

```
for parametro in vector_parametros:

    for variacion in vector_variacion:

        #Step 1
        print("\n")
        print("CASO %d:" % (cont+1))

        if parametro == vector_parametros[0]:
            A = parametro*variacion
            c = c_inicial
            h = h_inicial
            print("Valor de A = ", A)
            print("Valor de c = ", c)
            print("Valor de h = ", h)

        if parametro == vector_parametros[1]:
                A = A_inicial
                c = parametro*variacion
                h = h_inicial
                print("Valor de A = ", A)
                print("Valor de c = ", c)
                print("Valor de h = ", h)

        if parametro == vector_parametros[2]:
                A = A_inicial
                c = c_inicial
                h = parametro*variacion
                print("Valor de A = ", A)
                print("Valor de c = ", c)
                print("Valor de h = ", h)

        it = 0
        p = c

        while True:

            print("\n")
            print("Iteración %d (Z*)" % (it+1))


            #Step 2
            p_variation = 2*A - h*(td**2)*alpha*p**(-beta)


            if p_variation > 0:

                x = Symbol('x')
                T1_lista = solve((alpha*p**(-beta)/x**2)*(((h+theta*h*td+theta*c)/theta**2)*(sympy.exp(theta*(x-td))-theta*x*sym
                print("T1 lista = ", T1_lista) #x tiene dos soluciones, nos quedamos con la positiva
                T1 = T1_lista[-1] #nos quedamos con la última componente, que es la positiva

                p1 = (beta/(beta-1))*(1/T1*(h*td/theta*(sympy.exp(theta*(T1-td))-1)+h*td**2/2+(h+theta*c)/theta**2*(sympy.exp(th

                #Step 3
                if abs(p-p1) <= 1e-5:

                    p_opt = p1
                    T_opt = T1
                    Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
                    Z1 = (p_opt-c)*alpha*p_opt**(-beta)-((alpha*p_opt**(-beta))/T_opt)*((h*td/theta)*(math.exp(theta*(T_opt-td))

                    Z_opt = Z1

                    write_resultados(A,c,h,p_opt,T_opt,Q_opt,Z_opt)
                    variacion_porcentual(A,c,h,p_opt,T_opt,Q_opt,Z_opt)

                    print("El óptimo de la F.O. es: ", Z_opt)
                    print("p_opt = ", p_opt)
                    print("T_opt = ", T_opt)
                    print("Q_opt = ", Q_opt)

                    cont = cont+1

                    break

                else:
                    it = it+1
                    p = p1

            else:

                if p_variation < 0:

                    T2 = math.sqrt((2*A)/(h*alpha+p**(-beta)))
                    p2 = (beta/(beta-1))*((h*T2)/2+c)
```

```
    #Step 3
    if abs(p-p2) <= 1e-5:

        p_opt = p2
        T_opt = T2
        Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
        Z2 = (p_opt-c)*alpha*p_opt**(-beta)-(h*alpha*(p_opt**(-beta))*T_opt)/2-A/T_opt

        Z_opt = Z2

        write_resultados(A,c,h,p_opt,T_opt,Q_opt,Z_opt)
        variacion_porcentual(A,c,h,p_opt,T_opt,Q_opt,Z_opt)

        print("El óptimo de la F.O. es: ", Z_opt)
        print("p_opt = ", p_opt)
        print("T_opt = ", T_opt)
        print("Q_opt = ", Q_opt)

        cont = cont+1

        break

    else:
        it = it+1
        p = p2

else:

    if p_variation == 0:

        T3 = td
        p3 = (beta/(beta-1))*((h*T3)/2+c)

        #Step 3
        if abs(p-p3) <= 1e-5:

            p_opt = p3
            T_opt = T3
            Q_opt = ((alpha*p_opt**(-beta))/theta)*(math.exp(theta*(T_opt-td))-1) + alpha*(p_opt**(-beta))*td
            #Z se puede calcular por cualquiera de las dos expresiones
            Z1 = (p_opt-c)*alpha*p_opt**(-beta)-((alpha*p_opt**(-beta))/T_opt)*((h*td/theta)*(math.exp(theta*(T_
            Z2 = (p_opt-c)*alpha*p_opt**(-beta)-(h*alpha*(p_opt**(-beta))*T_opt)/2-A/T_opt
            print("Z1 = ", Z1)
            print("Z2 = ", Z2)

            Z_opt = Z1

            write_resultados(A,c,h,p_opt,T_opt,Q_opt,Z_opt)
            variacion_porcentual(A,c,h,p_opt,T_opt,Q_opt,Z_opt)

            print("El óptimo de la F.O. es: ", Z_opt)
            print("p_opt = ", p_opt)
            print("T_opt = ", T_opt)
            print("Q_opt = ", Q_opt)

            cont = cont + 1

            break

        else:
            it = it+1
            p = p3
```

Figura 18. Código Análisis de Sensibilidad.

7 EJEMPLOS NUMÉRICOS

7.1 Ejemplo 1

En este caso, se analizan los resultados que se obtienen de la combinación de distintos valores del periodo de tiempo en el que no se produce deterioro t_d y la constante de deterioro θ.

- $t_d = \left\{0;\ \frac{15}{365} = 0{,}0411;\ \frac{30}{365} = 0{,}0822\right\}$ años
- $\theta = \{0{,}05;\ 0{,}1;\ 0{,}2\}$

Los costes toman los valores siguientes:

- A = 250 $ / pedido
- h = 0,4 $ / unidad
- c = 3 $ / unidad

La demanda viene representada por la función $D(p) = \alpha p^{-\beta}$ donde α y β toman los valores 400.000 y 2,5 respectivamente, quedando la expresión $D(p) = 400.000\, p^{-2,5}$.

En la Tabla 1 se representan los resultados obtenidos:

θ	t_d	p^*	T^*	Q^*	Z^*	Iteraciones
0,05	0	5,17031	0,3695	2.454,11	12.933,0	5
	0,0411	5,16044	0,36934	2.460,02	12.971,8	5
	0,0822	5,15232	0,37057	2.473,79	13.006,3	5
0,1	0	5,19325	0,32768	2.167,96	12.756,6	5
	0,0411	5,17354	0,32751	2.179,05	12.833,5	5
	0,0822	5,15782	0,3298	2.203,92	12.901,0	5
0,2	0	5,23321	0,27475	1.803,3	12.455,0	6
	0,0411	5,19402	0,27465	1.822,88	12.606,3	6
	0,0822	5,16447	0,2787	1.865,01	12.735,6	6

Tabla 1. Resultados Ejemplo 1

A la vista de los resultados expuestos para este ejemplo, se pueden extraer las siguientes conclusiones:

(a) Para un valor fijo de θ, al aumentar t_d, la cantidad óptima de pedido, Q^*, y el beneficio total por unidad, Z^*, aumentan mientras el precio óptimo de venta, p^*, disminuye. Desde el punto de vista de la gestión de inventarios, cuanto mayor es el periodo de tiempo en el que la mercancía no se deteriora,

menor es el precio de venta y mayores las cantidades óptimas de pedido, así como el beneficio total por unidad de tiempo obtenido. Además, cuanto menor es la tasa de deterioro, θ, menor es el impacto de la duración del periodo de no deterioro de los productos, t_d, en el beneficio. En cambio, al aumentar θ, el beneficio total por unidad de tiempo es significativamente mayor al aumentar t_d.

(b) Cuando la tasa de deterioro θ aumenta sin hacerlo el resto de parámetros del modelo, los valores óptimos del tiempo de ciclo de reposición, T^*, la cantidad de pedido, Q^*, y el beneficio total por unidad de tiempo Z^* disminuyen mientras el precio de venta, p^*, aumenta. Esto quiere decir que, si el vendedor consigue disminuir la tasa de deterioro de sus productos, conseguirá un mayor beneficio.

7.2 Ejemplo 2

La diferencia con el Ejemplo 1 reside en que, en este caso, las variables que toman distintos valores serán los parámetros que definen la función de demanda: α y β.

- $\alpha = \{200.000;\ 300.000;\ 400.000;\ 500.000;\ 600.000\}$
- $\beta = \{2;\ 2{,}5;\ 3\}$

Los costes A, h y c toman valores idénticos a los del ejemplo anterior. Por último, se toman $\theta = 0{,}05$ y $t_d = \frac{15}{365} = 0{,}0411$ años.

De la combinación de los valores de α y β se obtienen 15 casos diferentes, cuyos resultados se recogen en la Tabla 2:

α	β	p^*	T^*	Q^*	Z^*	Iteraciones
200.000	2	6,21907	0,41716	2.175,54	15.480,3	5
	2,5	5,23488	0,52988	1.709,39	6.207,5	6
	3	4,78181	0,69754	1.295,8	2.557,2	6
300.000	2	6,17567	0,33891	2.683,34	23.551,2	5
	2,5	5,1879	0,42871	2.116,45	9.572,2	5
	3	4,72431	0,56072	1.614,68	4.034,6	6
400.000	2	6,15014	0,29271	3.112,3	31.665,5	5
	2,5	5,16044	0,36934	2.460,02	12.971,8	5
	3	4,69104	0,48124	1.883,6	5.539,5	6
500.000	2	6,13288	0,26139	3.490,99	39.807,5	5
	2,5	5,14193	0,32923	2.763,08	16.393,7	5
	3	4,66876	0,42787	2.120,65	7.061,9	6
600.000	2	6,12022	0,23838	3.834,05	47.969,1	5

α	β	p^*	T^*	Q^*	Z^*	Iteraciones
	2,5	5,12839	0,29984	3.037,42	19.831,4	5
	3	4,65253	0,38891	2.335,1	8.596,7	6

Tabla 2. Resultados Ejemplo 2

De los resultados obtenidos se puede concluir que cuando el factor de escala, α, aumenta manteniéndose β invariable, la cantidad óptima de pedido Q^*y el beneficio total por unidad de tiempo Z^* aumentan, mientras que el precio de venta p^* y la duración óptima del ciclo de reposición T^* disminuyen.

Por otro lado, cuando β aumenta manteniéndose constante α, el precio de venta óptimo p^*, la cantidad óptima de pedido y el beneficio total por unidad de tiempo Z^* disminuyen, mientras la duración del ciclo de reposición aumenta.

La conclusión desde el punto de vista de la gestión de inventarios es que el incremento del factor de escala de la demanda, α, hace que la demanda se vea también incrementada. Así, la cantidad óptima de pedido por ciclo de reposición y el beneficio total por unidad de tiempo aumentarán. Por su parte, cuando lo que varía es la elasticidad del precio, β, su incremento causa una disminución tanto de la cantidad de pedido por ciclo de reposición como del beneficio total por unidad de tiempo.

8 ANÁLISIS DE SENSIBILIDAD

En este capítulo se estudian los efectos de la variación de los parámetros A, c y h en el precio de venta óptimo p^*, la duración del ciclo de reposición T^*, la cantidad de pedido Q^* y el beneficio total por unidad de tiempo Z^* tomando como base el Ejemplo 1 (donde $\alpha = 400.000$, $\beta = 2.5$, $t_d = \frac{15}{365} = 0{,}0411$ años y $\theta = 0{,}05$ toman valores fijos).

El análisis de sensibilidad en este caso se hace variando los valores de los parámetros indicados en porcentajes del -50%, -25%, +25%, +50%, cambiando solo uno de ellos cada vez, y dejando el resto invariantes.

Los resultados obtenidos se muestran en la Tabla 3. Estos han sido calculados aplicando la fórmula de la variación porcentual:

$$\%\ variación = \frac{v_{inicial} - v_{final}}{v_{inicial}} \times 100$$

Si el resultado es positivo hay un incremento porcentual, es decir, el valor final es mayor en ese tanto por ciento con respecto al valor inicial. Si es negativo, se produce una disminución porcentual.

Parámetro			p^*		T^*		Q^*		Z^*	
	% variación	Valor	% variación	Valor	% variación	Valor	% variación	Valor	% variación	Valor
A	-50	125,0	-0,99	5,10958	-29,91	0,25888	-28,35	1.762,71	3,07	13.370,1
	-25	187,5	-0,45	5,13699	-13,77	0,3185	-12,89	2.142,98	1,4	13.153,5
	+25	312,5	0,41	5,18136	12,25	0,41457	11,24	2.736,56	-1,23	12.812,3
	+50	375,0	0,78	5,20047	23,41	0,45582	21,31	2.984,31	-2,34	12.668,7
c	-50	1,5	-50,36	2,56159	-54,88	0,16664	158,61	6.361,9	189,29	37.525,9
	-25	2,25	-25,26	3,85714	-27,82	0,26658	49,06	3.666,91	55,6	20.183,5
	+25	3,75	25,39	6,47092	28,12	0,47319	-27,05	1.794,58	-29,14	9.191,8
	+50	4,5	50,92	7,78826	56,29	0,57723	-43,86	1.381,07	-46,59	6.928,3
h	-50	0,2	-0,69	5,12471	24,13	0,45847	26,58	3.114,01	2,12	13.246,3
	-25	0,3	-0,33	5,1435	10,05	0,40645	11,06	2.732,04	1,0	13.101,0
	+25	0,5	0,3	5,17601	-7,63	0,34115	-8,39	2.253,65	-0,91	12.854,3
	+50	0,6	0,58	5,19053	-13,68	0,31881	-15,03	2.090,18	-1,74	12.746,0

Tabla 3. Análisis de Sensibilidad

Observando los resultados, es posible extraer las siguientes conclusiones:

(a) El precio de venta óptimo p^* aumenta cuando aumentan los parámetros A, c y h. Además, el incremento de p^* es más acusado cuando cambia c que cuando lo hacen A y h, lo cual tiene sentido, dado que el coste del producto influye de manera más significativa en el precio de venta.

(b) El tiempo de ciclo de reposición T^* aumenta con el incremento de los parámetros A, c y h, siendo el que más influye el coste del bien, c.

(c) La cantidad óptima de pedido Q^* es ligeramente sensible a las variaciones de A y h, mientras que varía con más fuerza con el incremento de c. Q^* disminuye conforme aumenta c, lo cual es lógico. A mayor coste del producto, menor número de unidades del mismo se pedirán de una vez.

(d) El beneficio total por unidad Z^* disminuye levemente con las variaciones de los parámetros A y h, mientras que lo hace de forma más significativa cuando aumenta c. Es obvio que el beneficio total dismunuya al aumentar el coste del producto.

9 CONCLUSIONES

El artículo de Wu, Ouyang y Yang (2006) en el que se basa este trabajo contribuyó a la metodología existente de varias formas:

1. En primer lugar, aborda el problema de los bienes con deterioro no instantáneo bajo la premisa de que la demanda es dependiente del precio, un asunto que no había sido tratado por la literatura hasta la fecha.

2. En segundo lugar, desarrolla varios lemas útiles y proporciona un algoritmo que determina el precio óptimo y la duración de un ciclo de reabastecimiento.

3. En tercer lugar, de los resultados teóricos se deduce que el minorista puede determinar la cantidad de pedido y el precio de venta óptimos negociando la política de vender todas sus existencias antes/después de que los productos empiecen a deteriorarse para los artículos que no se deterioran instantáneamente. La regla de decisión consiste en comparar el coste total de mantenimiento durante el periodo en el que no se produce el deterioro ($ht_d^2 D(p)/2$) con el coste de lanzar un pedido (A).

4. En cuarto lugar, los ejemplos numéricos muestran que cuanto más largo sea el tiempo de no deterioro de los productos, mayor es la cantidad de pedido y el beneficio total por unidad de tiempo. El precio de venta disminuye al hacerlo el coste de compra, mientras que la cantidad de pedido y el beneficio total óptimo por unidad de tiempo aumentan al disminuir el coste de compra.

El modelo propuesto puede ser utilizado en el control de inventarios de artículos con deterioro no instantáneo tales como alimentos, componentes electrónicos, moda u otros. El gestor de inventario, usando el modelo obtenido en este estudio, puede determinar de manera efectiva el precio y la cantidad de pedido óptimos.

10 REFERENCIAS

Abad, P.L. (1996). Optimal Pricing and Lot Sizing Under Conditions of Perishability and Partial Backordering. *Management Science, 42*, 1093-1104.

Abad, P.L. (2003). Optimal Pricing and Lot-sizing Under Conditions of Perishability, Finite Production and Partial Backordering and Lost Sales. *European Journal of Operational Research, 144*, 677-686.

Cantidad económica de pedido. (s.f.). Obtenido de Wikipedia, la Enciclopedia Libre: https://es.wikipedia.org/wiki/Cantidad_econ%C3%B3mica_de_pedido

Chang, H.H., Teng, J.T., Ouyang, L.Y., and Dye, C.Y. (2006). Retailer's Optimal Pricing and Lot-sizing Policies for Deteriorating Items with Partial Backlogging. *European Journal of Operational Research, 168*, 51-64.

Cohen, M.A. (1977). Joint Pricing and Ordering Policy for Exponentially Decaying Inventory with Known Demand. *Naval Research Logistic Quarterly, 24*, 257-268.

Covert, R.P., and Philip, G.C. (1973). An EOQ Model for Items with Weibull Distribution Deterioration. *AIIE Transaction, 5*, 323-326.

Dye, C. Y. (2013). The effect of preservation technology investment on a non-instantaneous deteriorating inventorymodel. *Omega, 41*(5), 872–880.

Dye, C.Y. (2007). Joint Pricing and Ordering Policy for Deteriorating Inventory with Partial Backlogging. *Omega, 35*, 184-189.

El tutorial de Python. (s.f.). Obtenido de Documentación de Python: https://docs.python.org/es/3/tutorial/

Ghare, P.M., and Schrader, G.H. (1963). A Model for Exponentially decaying Inventory System. *International Journal of Production Research, 21*, 449-460.

Ghoreishi, M., Weber, G. W., & Mirzazadeh, A. (2015). Annals of Operations Research. *An inventory model for non-instantaneous deteriorating items with partial backlogging, permissible delay in payments, inflation-and selling price-dependent demand and customer returns., 226*(1), 221–238.

Goyal, S.K., and Giri, B.C. (2001). Recent Trends in Modeling of Deteriorating Inventory. *European Journal of Operational Research, 134*, 1-16.

Goyal, S.K., and Gunasekaran, A. (1995). An Integrated Production-inventory-marketing Model for Deteriorating Items. *Computers & Industrial Engineering, 28*, 755-762.

Hsieh, T. P., & Dye, C. Y. (2010). Pricing and lot-sizing policies for deteriorating items with partial backlogging under inflation. *Expert Systems with Applications, 37*(10), 234–7242.

Jason, F. (2022). *Economic Order Quantity (EOQ)*. Obtenido de Investopedia: https://www.investopedia.com/terms/e/economicorderquantity.asp

Khan, M. A. A., Shaikh, A. A., Panda, G. C., & Konstantaras, I. (2019a). Two-warehouse inventory model for deteriorating items with partial backlogging and advance payment scheme. *RAIRO-Operations Research, 53*(5), 1691–1708.

La Librería Numpy. (s.f.). Obtenido de Aprende con Alf: https://aprendeconalf.es/docencia/python/manual/numpy/

Librería math - Funciones matemáticas. (s.f.). Obtenido de Documentación de Python: https://docs.python.org/es/3/library/math.html

Librería re - Operaciones con expresiones regulares. (s.f.). Obtenido de Documentación de Python: https://docs.python.org/es/3/library/re.html

Librería Sympy. (s.f.). Obtenido de Sympy.org: https://www.sympy.org/es/

Lin, C., Tan, B., and Lee, W.-C. (2000). An EOQ Model for Deteriorating Items with Time-varying Demand and Shortages. *International Journal of Systems Science, 31*, 391-400.

Mahmoudinezhad, M., Ghoreishi, M., Mirzazadeh, A., & Ghodratnama, A. (2014). An inventory model for non-instantaneous deteriorating items with imperfect quality, delay in payments and time value money. *Journal of Industrial Engineering and Management Studies, 1*(1), 58–71.

Maihami, R., & Abadi, I. N. K. (2012). Joint control of inventory and its pricing for non-instantaneously deteriorating items under permissible delay in payments and partial backlogging. *Mathematical and Computer Modelling, 55*(5), 1722–1733.

Maihami, R., & Kamalabadi, I. N. (2012). Joint pricing and inventory control for non-instantaneous deteriorating itemswith partial backlogging and time and price dependent demand. *International Journal of Production Economics, 136*(1), 116–122.

Mashud, A., Khan, M., Uddin, M., & Islam, M. (2018). A non-instantaneous inventory model having different deterioration rates with stock and price dependent demand under partially backlogged shortages. *Uncertain Supply Chain Management, 6*(1), 49–64.

Mukhopadhyay, S., Mukherjee, R.N., and Chaudhuri, K.S. (2004). Joint Pricing and Ordering Policy for a Deteriorating Inventory. *Computers and Industrial Engineering, 47*, 339-349.

Mukhopadhyay, S., Mukherjee, R.N., and Chaudhuri, K.S. (2005). An EOQ Model with Two-parameter Weibull Distribution Deterioration and Price-Dependent Demand. *International Journal of Mathematical Education in Science and Technology, 36*, 25-33.

Patel, H., & Gor, A. (2019). Salvage value and three variable Weibull deteriorating rate for non-instantaneous deteriorating items. *Asian-European Journal of Mathematics, 12*(05).

Paz Soldán Marcelo and Villarroel Jaime. (2009). La elasticidad precio de la demanda para algunos productos de la economía boliviana.

Philip G.C. (1974). A Generalized EOQ Model for Items with Weibull Distribution. *AIIE Transaction, 6*, 159-162.

Rezagholifam, M., Sadjadi, S. J., Heydari, M., & Karimi, M. (2022). Optimal pricing and ordering strategy for non-instantaneous deteriorating items with price and stock sensitive demand and capacity constraint. *International Journal of Systems Science: Operations & Logistics, 9*(1), 121-132.

Shah, Y.K. (1977). An Order-level Lot Size Inventory Model for Deteriorating Items. *AIIE Transaction, 9*, 108-112.

Shaikh, A. A., Mashud, A. H. M., Uddin, M. S., & Khan, M. A. A. (2017). Non-instantaneous deterioration inventory model with price and stock dependent demand for fully backlogged shortages under inflation. *International Journal of Business Forecasting and Marketing Intelligence, 3*(2), 152–164.

Sonntag, M. (s.f.). *Inventory Replenishment: Definition, Explanation, & Best Practices*. Obtenido de Repsly: https://www.repsly.com/blog/inventory-replenishment-definition-explanation-best-practices

Teorema de Bolzano. (s.f.). Obtenido de Wikipedia, la Enciclopedia Libre: https://es.wikipedia.org/wiki/Teorema_del_valor_intermedio

Tiwari, S., Cárdenas-Barrón, L. E., Khanna, A., & Jaggi, C. K. (2016). Impact of trade credit and inflation on retailer's ordering policies for non-instantaneous deteriorating items in a two-warehouse environment. *International Journal of Production Economics, 176*, 154–169.

Vaghela, C. R., & Shah, N. H. (2020). Dynamic pricing, advertisement investment and replenishment model for deteriorating items. *Optimization and inventory management asset analytics (performance and safety management)*, 81–92.

Wang, C., & Huang, R. (2014). Pricing for seasonal deteriorating products with price-and ramp-type time-dependent demand. *Computers & Industrial Engineering, 77*, 29–34.

Wee, H.M. (1995). Joint Pricing and Replenishment Policy for Deteriorating Inventory with Declining Market. *International Journal of Production Economics, 40*, 163-171.

Wee, H.M. (1997). A Replenishment Policy for Items with a Price-dependent Demand and a Varying Rate of Deterioration. *Production Planning and Control, 8*, 494-499.

Wee, H.M. (1999). Deteriorating Inventory Model with Quantity Discount, Pricing and Partial Backordering. *International Journal of Production Economics, 59*, 511-518.

Wee, H.M., and Law, S.T. (2001). Replenishment and Pricing Policy for Deteriorating Items Taking into Account the Time Value of Money. *International Journal of Production Economics, 71*, 213-220.

Wu, K.S., Ouyang L.Y., & Yang C.T. (2006). An Optimal Replenishment Policy for Non-instantaneous Deteriorating Items with Stock-dependent Demand and Partial Backlogging. *International Journal of Production Economics, 101*, 369–384.

Wu, K.S., Ouyang L.Y., & Yang C.T. (2009). Coordinating replenishment and pricing policies for non-instantaneous deteriorating items with price-sensitive demand. *International Journal of Systems Science, 40*(12), 1273-1281.

Yang, C. T., Ouyang, L. Y., & Wu, H. H. (2009). Retailer's optimal pricing and ordering policies for non-instantaneous deteriorating items with price-dependent demand and partial backlogging. *Mathematical Problems in Engineering*.