(RCIA) | vol. 2(2):diciembre 2023

Método Simplex en VBA: una propuesta didáctica para la enseñanza de la programación lineal

López Martínez, C.M.[II]
DOI: 10.56643/rcia.v2i2.168

Resumen

Como mencionan Coll y Blasco (2010), la enseñanza tradicional ha estado centrada principalmente en la figura del profesor y a menudo no integra herramientas tecnológicas que faciliten el aprendizaje práctico. Este artículo desarrolla una herramienta pedagógica diseñada para mejorar la comprensión y aplicación de la programación lineal en estudiantes de ingeniería y administración. Aunque existen diversos estudios que utilizan la función Solver de Microsoft Excel para resolver problemas de programación lineal, la misma presenta desventajas importantes, como la falta de adaptabilidad y personalización. En el contexto educativo actual, que demanda métodos de enseñanza interactivos y prácticos, la propuesta de este estudio se destaca por integrar el Método Simplex en el entorno familiar de Microsoft Excel, utilizando Visual Basic for Applications (VBA). Esto permite a los estudiantes interactuar directamente con el código y observar los resultados del algoritmo en tiempo real, lo que facilita una comprensión más profunda de los conceptos subyacentes y promueve un aprendizaje significativo. Este artículo no sólo describe el desarrollo y las funcionalidades de la herramienta, sino que también proporciona el código Nota completo, fomentando su replicación científica y su mejora continua por parte de la comunidad educativa. Aunque la propuesta se centra en el uso de Excel y vba, se reconoce la necesidad de adaptabilidad a diferentes entornos tecnológicos y estilos de aprendizaje. A través de este trabajo, se invita a la comunidad educativa a participar en la evolución de las estrategias de enseñanza de la programación lineal, preparando a los estudiantes para enfrentar con confianza los retos analíticos y tecnológicos del futuro.

Palabras clave: Investigación de Operaciones, Excel, Programación VBA, optimización, ingeniería, administración

Abstract

As Coll and Blasco (2010) mention, traditional teaching has been primarily centered around the figure of the teacher, and often does not integrate technological tools that facilitate practical learning. This article develops a pedagogical tool designed to improve the understanding and application of linear programming in engineering and management students. Although there are numerous studies that use Microsoft Excel's Solver function to solve linear programming problems, this function has major disadvantages such as lack of adaptability and customization. In the current educational context, which demands interactive and practical teaching methods, the proposal of this study stands out for integrating the Simplex Method into the familiar environment of Microsoft Excel, using Visual Basic for Applications (VBA). This allows students to interact directly with the code and observe the algorithm's results in real time, thus facilitating a deeper understanding of the underlying concepts and promoting meaningful learning. This article not only describes the development and functionalities of the tool but also provides the complete source code, encouraging scientific replication and continuous improvement by the educational community. Although the proposal focuses on the use of Excel and VBA, the need for adaptability in different technological environments and learning styles is recognized. Through this work, the educational community is invited to participate in the evolution of teaching strategies for linear programming, preparing students to confidently face the analytical and technological challenges of the future.

Keywords: Operations Research, Excel, Programming VBA, optimization, engineering, administration

Cómo citar este artículo: López Martínez, C. M. (2024) Método Simplex en VBA: una propuesta didáctica para la enseñanza de la programación lineal. Revista Científica de Ingenierías y Arquitectura. 2(1). 67-116. DOI

[II] Licenciado en Matemáticas Aplicadas y Computación, maestro en Ingeniería en Tecnologías de la Información, doctor en Ciencias de la Administración. Profesor de asignatura en la Escuela de Ingeniería y Arquitectura Universidad La Salle Oaxaca; profesor de asignatura de la Facultad de Contaduría y Administración (fca) de la Universidad Autónoma "Benito Juárez" de Oaxaca (uabjo) (México). Autor para correspondencia: miguel.lopez.beca@gmail.com
Recibido: 14/10/23 | Aceptado: 06/11/23| 20/12/2023

Introducción

Dantzig (1963) define la programación lineal como una rama de las matemáticas aplicadas centrada en la optimización de funciones lineales, sujeta a restricciones también lineales. Esta técnica ha demostrado ser esencial en diversos campos, que incluyen desde la ingeniería y la logística hasta la economía y la administración, debido a su capacidad para resolver problemas complejos relacionados con la asignación de recursos limitados. La esencia de la programación lineal radica en encontrar el punto óptimo, ya sea máximo o mínimo, de una función objetivo, considerando un conjunto de restricciones que definen un espacio factible.

Taha (2012) destaca la importancia de la programación lineal en la toma de decisiones, pues permite a las organizaciones determinar la mejor manera de utilizar sus recursos — mano de obra, materiales y tiempo—, para maximizar las ganancias o minimizar los costos. En el ámbito de la logística, ayuda a determinar rutas óptimas para la distribución de productos, mientras que en el sector financiero puede ser utilizada para la optimización de carteras de inversión.

Fuson (2020) concebía una instrucción en la cual los estudiantes construyen significado para los conceptos matemáticos y procedimientos que están investigando, y participan en actividades significativas de resolución de problemas. Esto, a diferencia de la instrucción tradicional basada en libros de texto, se centra, principalmente, en el aprendizaje de memoria y la práctica de habilidades. Si bien estas herramientas han sido efectivas para introducir a los estudiantes en los conceptos básicos, a menudo carecen de la flexibilidad y la capacidad para manejar problemas más complejos y realistas. Además, en un mundo cada vez más digitalizado, es esencial que los estudiantes estén familiarizados con el uso y la creación de herramientas tecnológicas que les permitan aplicar sus conocimientos de manera práctica y eficiente.

Orozco (2004, citado en López Noriega, Lagunes Huerta y Herrera Sánchez, 2006) menciona que la tecnología ha venido a cambiar por completo el panorama tradicional según el cual se hacían, se veían y se enseñaban las matemáticas. En este contexto, Microsoft Excel emerge como una herramienta poderosa y accesible. Su amplia adopción en el mundo académico y empresarial, lo convierte en una plataforma ideal para la enseñanza por su naturaleza, no sólo de la programación lineal, sino de muchas áreas de la matemática aplicada. Aunque Excel por sí solo ofrece capacidades limitadas para la programación lineal, su lenguaje de programación integrado, vba, amplía enormemente su potencial. vba permite a los usuarios crear soluciones personalizadas, automatizar tareas y desarrollar herramientas didácticas específicas, como es el caso del método simplex.

Barr y Scott (2008) mencionan la funcionalidad de programación incorporada de Excel, conocida como vba, ideal para escribir herramientas de enseñanza basadas en simulaciones en estadísticas. Su estudio en la Universidad de Ciudad del Cabo mostró que este enfoque innovador es popular entre los estudiantes y efectivo para transmitir ideas estadísticas fundamentales a un amplio espectro de capacidades estudiantiles.

La integración de vba a la enseñanza de la programación lineal no sólo ofrece a los estudiantes una herramienta práctica para resolver problemas; también les brinda habilidades valiosas en programación y análisis de datos. Al familiarizarse con VBA de Microsoft Excel, los estudiantes aprenden sobre programación lineal y adquieren competencias transferibles que son altamente valoradas en el mercado laboral.

Aliane (2008) destaca el uso de Microsoft Excel como una plataforma alternativa que permite desarrollar herramientas de aprendizaje interactivas para la educación en control. Las hojas de cálculo modernas, como Excel, incluyen un lenguaje macro que permite la inclusión de código informático estándar. Esto permite a los usuarios incorporar programas específicos para gestionar y simular sistemas complejos, lo que lo convierte en una herramienta didáctica valiosa para la enseñanza de conceptos abstractos y complejos.

Justificación del estudio

La justificación de este estudio se basa en la necesidad de contar con una propuesta didáctica innovadora que complemente los métodos existentes para la resolución de problemas de programación lineal (ppl) en Microsoft Excel. Aunque existen diversos estudios, como los de Arboleda (2016), Bofil (2019) y Cruz (2007) que utilizan Microsoft Excel para solucionar ppl, la mayoría de ellos utilizan el complemento Solver. Este estudio se distingue de los mencionados pues se enfoca no sólo en encontrar soluciones, sino también en facilitar una comprensión más profunda y un aprendizaje efectivo de la programación lineal. La integración de vba para Excel supera a la función Solver en términos de personalización y flexibilidad. Además, profundiza en el conocimiento de

ISSN: En trámite

los estudiantes sobre vba y programación lineal, al tiempo que cultiva habilidades analíticas y de razonamiento lógico-deductivo transferibles a una amplia gama de tareas profesionales. Al aprender a programar en vba en Excel, siendo este una herramienta que se encuentra en casi todos los ámbitos profesionales en la actualidad, los estudiantes desarrollan lógica algorítmica, mejorando su capacidad para estructurar y desglosar problemas complejos. Este enfoque didáctico brinda a los estudiantes una base sólida para la comprensión teórica y la aplicación práctica en situaciones reales, preparándolos para enfrentar desafíos tecnológicos y analíticos con confianza y competencia. Por lo tanto, este estudio busca presentar una propuesta para la enseñanza de la programación lineal y contribuir al desarrollo de competencias esenciales para el éxito en el entorno profesional contemporáneo.

Objetivo general

Desarrollar una propuesta didáctica para la enseñanza de la programación lineal mediante la construcción de una herramienta basada en el Método Simplex implementado en vba para Microsoft Excel, con el fin de mejorar el proceso de enseñanza y aprendizaje de la programación lineal en estudiantes de ingeniería y administración.

Objetivos específicos

Facilitar la comprensión conceptual y práctica de la programación lineal a través de la interacción directa con el código y la visualización inmediata de los resultados del algoritmo.

Promover el desarrollo de habilidades analíticas y de razonamiento lógico-deductivo en los estudiantes mediante la programación en vba, destacando la transferencia de estas competencias a diversas situaciones profesionales.

Contribuir a la literatura pedagógica en programación lineal, ofreciendo una herramienta que pueda ser adaptada y personalizada a diferentes niveles educativos y contextos de aprendizaje.

Proporcionar el código Nota completo del desarrollo del método simplex en vba, para fomentar su replicación, personalización y mejora continua por parte de la comunidad educativa y científica.

Limitantes

Es importante reconocer que el desarrollo del método simplex como una propuesta didáctica para la enseñanza de la programación lineal se basa única y exclusivamente en el uso de Microsoft Excel y vba, lo que puede restringir su aplicabilidad a entornos que no utilizan este *software*, limitando así su generalización. Además, en tanto no se pretende realizar una evaluación empírica integral de la propuesta, no pueden hacerse afirmaciones concluyentes sobre su efectividad en el mejoramiento del aprendizaje de la programación lineal, por lo que surge como una alternativa más a los métodos existentes. La efectividad de la propuesta también puede verse influenciada por la familiaridad previa de los usuarios con vba y Excel, lo que podría representar una barrera para aquellos sin experiencia previa. Además, la diversidad de estilos de aprendizaje entre los estudiantes sugiere que la herramienta podría no ser universalmente efectiva para todos los usuarios. Estas limitaciones deben ser consideradas al interpretar los resultados y evaluar la aplicabilidad de la herramienta en diferentes contextos educativos.

Metodología

Este estudio tiene un enfoque cualitativo, ya que está centrado en desarrollar y describir una herramienta pedagógica para la enseñanza de la programación lineal. La metodología cualitativa es adecuada para explorar en profundidad cómo los estudiantes interactúan con la herramienta y cómo afecta su aprendizaje y comprensión del tema. Dado que la propuesta describe el desarrollo y la implementación de una herramienta pedagógica en un momento específico, este tipo de estudio recoge datos en un único punto en el tiempo, en lugar de seguir a los mismos sujetos a lo largo del tiempo y, aunque la propuesta actual es transversal, también abre la puerta a futuros estudios longitudinales. Tales estudios podrían seguir a los estudiantes a lo largo de un periodo, para ver cómo la implementación de la herramienta afecta su aprendizaje y su retención de conocimientos a largo plazo.

De la misma manera, si se decide incorporar la recopilación y análisis de datos numéricos, como resultados de exámenes, tasas de retención de estudiantes, o estadísticas de uso de la herramienta, en el futuro el estudio podría tener, también, un componente cuantitativo. Esto sería útil para medir de manera más objetiva la eficacia de la herramienta.

Actualmente, la metodología es de tipo cualitativo y transversal, pero tiene el potencial de

expandirse a metodologías longitudinales y cuantitativas, dependiendo de cómo se elija implementar y evaluar la herramienta en el futuro.

Antecedentes

La programación lineal, como disciplina matemática, ha sido una herramienta esencial para la toma de decisiones y la optimización desde mediados del siglo xx. Su relevancia en diversos campos ha llevado al desarrollo de métodos y técnicas que faciliten su comprensión y aplicación.

El método simplex fue creado en 1947 por el estadounidense George Bernard Dantzig y el ruso Leonid Vitalievich Kantorovich, con el ánimo de concebir un algoritmo capaz de solucionar problemas de m restricciones y n variables. Se lo considera uno de los algoritmos más importantes de la historia y hoy por hoy sigue siendo la base en la que se fundamenta la mayor parte de los solucionadores de modelos de programación lineal (Salazar, 2019). Aunque la programación lineal como concepto ya había sido formulada anteriormente, el método simplex ofreció una técnica sistemática y eficiente para encontrar soluciones óptimas (Martínez, 2013). A pesar de la aparición de otros métodos con el transcurso de los años, el simplex sigue siendo ampliamente utilizado debido a su robustez y eficiencia en una amplia variedad de problemas.

Uso previo de herramientas computacionales para la enseñanza de la programación lineal

Con la evolución de la tecnología y la computación, la enseñanza de la programación lineal ha experimentado cambios significativos. Inicialmente, los problemas se resolvían manualmente o con la ayuda de calculadoras básicas. Sin embargo, tras el advenimiento de las computadoras en los años sesenta y setenta, comenzaron a desarrollarse *software* especializados que permitían resolver problemas más complejos y grandes. Paquetes como lindo, cplex y gams se convirtieron en herramientas estándar en la academia y la industria. Estos programas no sólo permiten resolver problemas, sino que también ofrecen una plataforma para la enseñanza y el aprendizaje, facilitando la visualización y la comprensión de conceptos abstractos. Estos paquetes de *software* han evolucionado y se han mantenido actualizados a lo largo de los años, y continúan siendo extensamente utilizados en la investigación, la industria y la educación para abordar problemas de optimización complejos. Por otro lado, para Windows aparece el tora, un *software* educativo empleado para resolver problemas en operaciones de investigación (IO). La sigla "tora" proviene de "TOols for Research in Operations", que se traduce como "Herramientas para la Investigación en Operaciones" (Salazar, 2019). Es muy usado en cursos académicos y por profesionales para modelar y resolver problemas de IO.

Beneficios y limitaciones de las herramientas tradicionales

Las herramientas tradicionales, como el método gráfico o el tabular, ofrecen una introducción básica y conceptual a la programación lineal. Son excelentes para enseñar los fundamentos y permiten a los estudiantes visualizar soluciones y entender la estructura de los problemas. Sin embargo, presentan limitaciones significativas. Por ejemplo, el método gráfico es útil sólo para problemas con dos variables y el método tabular puede volverse tedioso y propenso a errores cuando se trata de problemas más grandes.

Por otro lado, las herramientas computacionales tradicionales, aunque poderosas, a menudo requieren una curva de aprendizaje empinada. Además, su costo y la necesidad de licencias pueden operar como barreras para su adopción en algunas instituciones educativas.

Mientras que las herramientas tradicionales han sido fundamentales para la enseñanza de la programación lineal, la evolución de la tecnología y las necesidades cambiantes de los estudiantes han llevado a la búsqueda de alternativas más modernas, flexibles y accesibles, como Microsoft Excel y vba.

Desde su concepción, Microsoft Excel ha sido una herramienta revolucionaria en el mundo de la informática y la gestión de datos. Su diseño basado en celdas, organizado en filas y columnas, lo convierte en una matriz que facilita la captura, organización y análisis de datos de manera intuitiva y eficiente. En el contexto de la programación lineal, uno de los puntos fuertes de Excel es su capacidad para manejar un gran número de variables sin restricciones significativas. Es cierto que existen limitantes técnicas en cuanto al número de filas y columnas. Según Microsoft (2023) Excel puede manejar 1 048 576 filas y 16 384 columnas, límites lo suficientemente amplios como para satisfacer las necesidades de la mayoría de los usuarios.

ISSN: En trámite

(RCIA) | vol. 2(2):diciembre 2023

La programación lineal es una herramienta esencial para la toma de decisiones y la optimización en diversos campos. El método simplex, en particular, ha sido un algoritmo estándar para resolver problemas de programación lineal desde su introducción. Sin embargo, la implementación y enseñanza de este método requieren herramientas adecuadas que faciliten su comprensión y aplicación. Aquí es donde la combinación de Microsoft Excel y vba se presenta como una solución poderosa y versátil.

Excel como la puerta de entrada y salida de datos

Microsoft Excel, con su diseño basado en una matriz de celdas, ofrece un entorno intuitivo para la entrada y salida de datos. Los usuarios pueden fácilmente introducir coeficientes, restricciones y funciones objetivo en una hoja de cálculo de Excel, aprovechando su estructura tabular. Esta disposición matricial es especialmente útil para representar problemas de programación lineal, donde las relaciones entre variables, restricciones y objetivos son intrínsecas.

Además, Excel proporciona una visualización clara de los datos, permitiendo a los usuarios ver rápidamente las relaciones y los patrones. Una vez que se ha resuelto un problema utilizando el método simplex, los resultados pueden presentarse de manera ordenada y estructurada en la misma hoja de cálculo, lo que facilita su interpretación y análisis.

VBA como motor de cálculo y automatización

Mientras que Excel sirve como interfaz para la entrada y salida de datos, vba trabaja como el motor que actúa detrás de la interfaz gráfica de la hoja de cálculo, permitiendo a los usuarios crear macros y scripts personalizados para automatizar tareas y realizar cálculos complejos.

Al utilizar vba para implementar el método simplex, se pueden crear soluciones personalizadas que se adapten a las necesidades específicas de un problema. Los usuarios pueden definir y ajustar parámetros, establecer criterios de parada y, lo más importante, programar el algoritmo del método simplex para encontrar soluciones óptimas. Además, con vba es posible incorporar características adicionales, como la identificación de soluciones alternativas o la realización de análisis de sensibilidad.

Desarrollo de la propuesta

Como menciona Duart (2005, citado en Ferro, Martínez y Otero, 2009), las nuevas tecnologías de la información (tic) permiten un acceso más rápido y eficaz de docentes y estudiantes a la información, reduciendo el grado de obsolescencia de la información y permitiendo un uso más eficiente de las distintas Notas informativas existentes en la red. Así, la enseñanza de la programación lineal ha evolucionado con el tiempo, adaptándose a las herramientas y tecnologías emergentes. En este contexto, la combinación de la investigación de operaciones, vba y Excel se presenta como una propuesta didáctica innovadora, que busca no sólo facilitar la comprensión de conceptos inherentes a los modelos de investigación operativa, sino también proporcionar habilidades prácticas y aplicables en el mundo real. Por ejemplo, posibilitan la creación de interfaces de usuario personalizadas que facilitan la entrada de datos y la programación de modelos que permiten, a su vez, la automatización de cálculos y procesos repetitivos en el procesamiento de datos para la optimización de las operaciones mediante la personalización de la programación lineal.

Las tic transforman sustancialmente formas y tiempos de interacción entre docentes y estudiantes, que pueden tener lugar tanto de forma sincrónica como asincrónica. Este hecho favorece e incrementa los flujos de información y la colaboración entre ellos más allá de los límites físicos y académicos de la universidad a la que pertenecen (Ferro et al., 2009).

Almenar (2009) menciona que la elección de vba en Excel para la enseñanza de programación lineal se fundamenta en varias razones. Primero, Excel es una herramienta ampliamente utilizada en el mundo profesional y académico, lo que facilita su acceso y familiaridad para los estudiantes. Además, vba permite automatizar tareas y personalizar funciones en Excel, convirtiéndose en una herramienta poderosa para resolver problemas complejos de programación lineal.

Asimismo, Torres (2016) hace énfasis en que la interfaz gráfica intuitiva de Excel permite a los estudiantes visualizar datos y resultados de manera clara y organizada. A diferencia de otros lenguajes de programación, vba en Excel no requiere un aprendizaje extenso de sintaxis, lo que facilita su adopción por parte de estudiantes que no tienen experiencia previa en programación.

Ulloa (2005) define al método simplex como "un método sistemático altamente eficiente que permite resolver rápidamente, mediante el uso de programas de computadoras, problemas muy grandes, con gran número de variables, de restricciones o de ambos".

El método simplex es una técnica de resolución específica diseñada para abordar problemas de programación lineal (ppl) que involucran la optimización de una función lineal sujeta a restricciones lineales. A través del uso del método simplex es posible resolver una amplia variedad de problemas de programación lineal, incluyendo, pero no limitados a, los siguientes:

Maximización y minimización: como describe Weber (1984, citado en Alvarado, 2009), un problema de programación lineal trata de la maximización o minimización de una función lineal de varias variables primarias, llamada función objetivo, con sujeción a un conjunto de igualdades o desigualdades lineales denominadas restricciones, con la condición adicional de que ninguna de las variables puede ser negativa. Puede utilizarse para maximizar beneficios, ingresos u otros objetivos, así como para minimizar costos, gastos o recursos utilizados.

Modelo de asignación: Shen (2007) describe en su investigación el uso del método simplex en la solución de un modelo de asignación dinámica dirigido a optimizar el tráfico en casos de evacuación de emergencia. Los modelos de asignación se utilizan para asignar eficientemente recursos limitados a una serie de tareas o actividades, minimizando o maximizando una función objetivo.

Modelo de transporte: Flores (2021) formula y resuelve mediante el uso de un complemento de Microsoft Excel llamado Solver un modelo matemático de programación lineal de transporte para la empresa Holcim S. A.; toma en cuenta las unidades o cantidad de sacos de cemento disponibles en las plantas productoras para enviarlos a las distintas distribuidoras. Es importante mencionar que la función Solver utiliza el método simplex para encontrar soluciones óptimas a problemas lineales. En los problemas de transporte, el método simplex se emplea para determinar la forma más económica de transportar productos desde múltiples Notas a múltiples destinos, minimizando los costos de transporte o maximizando los ingresos asociados con el transporte.

Problemas de la estabilidad del suelo: Karaulov (2021) estudia cómo este tipo de problemas son formulados como tareas de programación lineal; se demuestra que los sistemas de ecuaciones dados son lineales con respecto a las incógnitas y pueden resolverse utilizando el método simplex, comparándolo con esquemas clásicos para calcular la estabilidad del suelo, se muestra que los resultados obtenidos por el método simplex son los más confiables.

Problemas de flujo máximo costo mínimo: Dadush (2020) ofrece una visión técnica de los análisis suavizados del método simplex de vértice sombra para la programación lineal (LP). Se revisan primero las propiedades del método simplex de vértice sombra y su geometría asociada. La discusión sobre el análisis suavizado comienza con un análisis del algoritmo de camino más corto sucesivo para el problema de flujo máximo de mínimo costo bajo perturbaciones objetivas, una clásica instanciación del método simplex de vértice sombra. Luego, se pasa a la programación lineal general y se ofrece un análisis de un algoritmo basado en el vértice sombra para la programación lineal bajo perturbaciones gaussianas de restricciones. El método simplex se utiliza en problemas de flujo máximo para determinar la cantidad máxima de flujo que puede circular a través de una red o sistema, minimizando o maximizando el flujo de acuerdo con restricciones de capacidad y demanda.

Problemas de finanzas y economía: en finanzas, el método simplex puede ser utilizado para optimizar carteras de inversión, gestionar riesgos, como también en la planificación financiera. En economía, se aplica para resolver problemas de asignación de recursos, producción y costos, y en la teoría de juegos, entre otros.

Por lo anterior, el aprendizaje y desarrollo del método simplex resulta una herramienta fundamental para resolver una amplia gama de problemas en diferentes campos y aplicaciones. Su capacidad para manejar restricciones y objetivos lineales lo hace una técnica esencial para la toma de decisiones informadas y eficientes en situaciones del mundo real. Principio del formulario

Taha (2012) refiere que el modelo algebraico en que se visualiza un ppl, en general con n variables y m restricciones, se contempla en el siguiente formato genérico:

$$\textit{Maximizar o Minimizar la función } Z: \sum_{j=1}^{n} c_j x_j$$

La maximización o minimización están sujetas a ciertas restricciones:

ISSN: En trámite

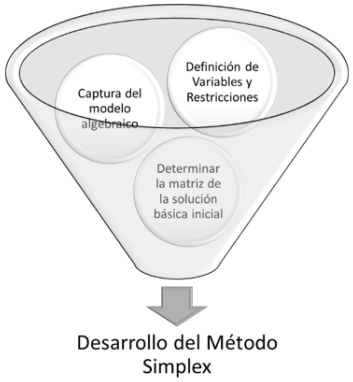
(RCIA) | vol. 2(2):diciembre 2023

$$\sum_{j=1}^{n} a_{ij} x_j [\leq, \geq, =] b_i, \; i = 1, 2, \ldots, m$$

$$x_j \geq 0, \; j = 1, 2, \ldots, n$$

Para el desarrollo del método simplex en vba, dividiré la explicación del algoritmo en tres momentos previos. Al final del documento se anexa el código con todos los procedimientos utilizados para el desarrollo de esta solución, a fin de que puedan ser replicados y personalizados por el estudiante.

Figura 1.
Momentos del desarrollo de la propuesta.



Nota: elaboración propia.

Definición de variables y restricciones. En un primer momento, una vez formulado el modelo PL, el algoritmo simplex inicia conociendo el número de variables (n) y el número de restricciones (m) para poder generar una subrutina que permita capturar el modelo de ppl. Para ello, se utilizarán las celdas B2 para el número de variables y B4 para el número de restricciones; en todo momento estas celdas funcionarán para la entrada de dichos datos. Es muy importante establecer que en el código se utiliza la notación matricial para referir las celdas de Excel, es decir, *Celda* (*fila, columna)*, por lo que tendríamos que la celda B2=Celda (2,2) y la celda B4=Celda (4,2).

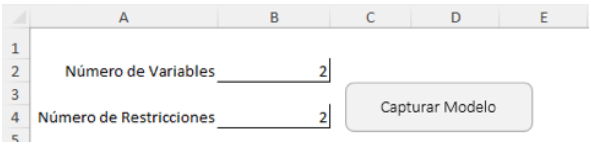


Figura 2.
Introducción de datos iniciales.
Nota: elaboración propia.

La figura 2 nos muestra la forma en que se define el número de variables y restricciones del modelo, contándose con un botón "*Capturar Modelo*"; la subrutina o macro asignada al botón se llama "*Iniciar*".

La subrutina "*Iniciar"* automatiza la generación de la estructura de captura del modelo algebraico de PL propuesto por Taha (2012). Dicha subrutina está diseñada para configurar una hoja de cálculo Excel destinada al análisis de modelos de programación lineal utilizados por el método simplex. Comienza limpiando la hoja de cálculo mediante otra subrutina denominada *"Limpiar"* y luego recopila el número de variables *nVar* y de restricciones *nRestricciones* del modelo a partir de las celdas específicas B2 y B4. Si alguno de estos valores es cero, se solicita al usuario que introduzca un número válido. Posteriormente, la subrutina procede a configurar la hoja para la entrada del modelo. Esto incluye el establecimiento de áreas para la función objetivo y las restricciones, utilizando bucles *For* para generar dinámicamente las celdas correspondientes a las variables y las restricciones. Además, se hace un llamado a la subrutina "*Desigualdad"* dentro de *"Iniciar"*; ésta se utiliza para configurar validaciones de entrada en las celdas de las restricciones, permitiendo al usuario elegir entre desigualdades como <=, =, o >=. Ello garantiza que las entradas de las restricciones sean consistentes con los requisitos del modelo de programación lineal. Finalmente, la subrutina culmina con la creación y configuración de un botón *"Resolver"* en la hoja de cálculo, que, al ser pulsado, ejecutará la subrutina *"Matriz_Inicial"* para procesar los datos ingresados en el modelo. Este enfoque hace que la herramienta sea interactiva y fácil de utilizar por los usuarios, permitiéndoles introducir y modificar de forma eficiente modelos de programación lineal en Excel.

Código VBA subrutina *Iniciar*

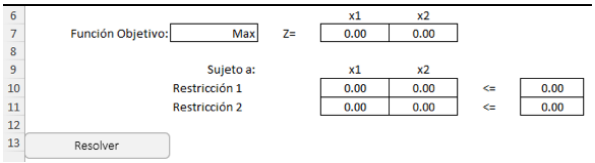
Sub Iniciar()

```
Dim i, j, Fila, Columna As Integer
Dim nVar As Integer
Dim nRestricciones As Integer
Call Limpiar
nVar = Cells(2, 2).Value
nRestricciones = Cells(4, 2).Value
If nVar = 0 Then
MsgBox "Introduzca el número de Variables del modelo"
Range("B2").Select
Else
 If nRestricciones = 0 Then
   MsgBox "Introduzca el número de Restricciones del modelo"
  Range("B4").Select
 Else
   Range("A7").Select
```

```
        ActiveCell.FormulaR1C1 = "Función Objetivo:"
        Call Celda_Captura(7, 2, "D")
        Range("C7").Select
        ActiveCell.FormulaR1C1 = "Z="
        Range("B9").Select
        ActiveCell.FormulaR1C1 = "Sujeto a:"
        Call Desigualdad(7, 2, "Max,Min", "Max")
        For i = 1 To nVar
          Cells(6, 3 + i).Value = "x" & i
          Call Celda_Captura(7, 3 + i, "D")
          Cells(9, 3 + i).Value = "x" & i
        Next i
        For i = 1 To nRestricciones
          Cells(9 + i, 2).Value = "Restricción " & i
           For j = 1 To nVar
             Call Celda_Captura(9 + i, 3 + j, "D")
           Next j
              Call Desigualdad(9 + i, 3 + nVar + 1, "<=,=,>=", "<=")
           Call Celda_Captura(9 + i, 3 + nVar + 2, "D")
        Next i
        ActiveSheet.Shapes.Range(Array(1)).Select
        Selection.Copy
        Cells(9 + nRestricciones + 2, 1).Select
        ActiveSheet.Paste
        Selection.OnAction = "Matriz_Inicial"
           Selection.ShapeRange(1).TextFrame2.TextRange.Characters.Text = "Resolver"
        Selection.ShapeRange(1).Name = "Boton"
        Range("B7").Select
      End If
    End If
End Sub
```

La *captura del modelo algebraico* de ppl constituye el segundo momento. Como se observa en la figura 3, en la celda (7,2) se establece el objetivo, esto es, si se trata de maximización o minimización. Asimismo, a partir de la celda (7,4) y hasta la celda (7,4+n-1) se establecen los coeficientes de las variables en la función objetivo. Para las restricciones, se toma la celda (10,4) como celda inicial hasta la celda (10+m-1,4+n-1) como celda final de la matriz de coeficientes de las restricciones. Los signos de desigualdad comenzarán en la celda (10,4+n) hasta la celda (10+m-1,4+n). Por último, se considera la celda (10,4+n+1) como el inicio del vector y la celda (10+m-1,4+n+1) como el final, correspondiente al lado derecho de las desigualdades. Es muy importante identificar las celdas de captura que dependen directamente del número de variables y restricciones, dado que es un proceso que se automatizará mediante código VBA.

Figura 3.
Estructura de captura del modelo.



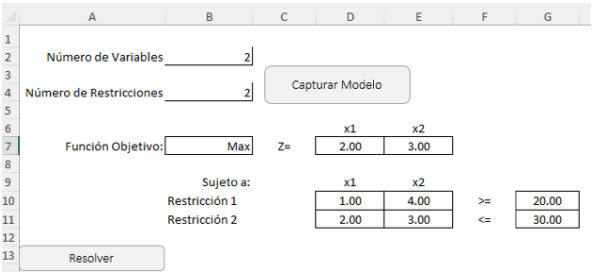
Nota: elaboración propia.

Supongamos un problema simple como ya se ha planteado, es decir, dos variables con dos restricciones.

$$Max\ Z = 2x_1 + 3x_2$$

$$S.a. \quad x_1 + 4x_2 \geq 20$$
$$2x_1 + 3x_2 \leq 30$$
$$x_1, x_2 \geq 0$$

Figura 4.
Modelo algebraico de PPL capturado



Nota: elaboración propia.

Una vez capturado el ppl, se genera el tercer momento de preparación previo al desarrollo del método simplex. Se hace el llamado a la subrutina "*Matriz_Inicial*", la cual se encarga de generar la matriz inicial simplex. Sin embargo, antes de continuar, se debe entender que, en el contexto del método simplex en la programación lineal, las holguras positivas, las holguras negativas y las variables artificiales son conceptos esenciales para transformar un problema de programación lineal en una forma que pueda ser resuelta utilizando el algoritmo simplex.

Como describe Frederick (2010), el procedimiento algebraico se basa en la solución de sistemas de ecuaciones. Por lo tanto, el primer paso para preparar el método simplex es convertir las restricciones funcionales de desigualdad en restricciones de igualdad equivalentes. A continuación, se explica cada uno de los conceptos mencionados.

Holguras positivas (variables de holgura). Cuando una restricción es de tipo "menor o igual que" (≤),

ISSN: En trámite

se añade una variable de holgura positiva denotada con la variable +Si, para convertir la desigualdad en una ecuación. Esta variable representa la "holgura" o el "espacio extra" disponible en esa restricción. Por ejemplo, considerando la restricción x1 + 4x2 ≤ 20, al agregar una variable de holgura S1, la restricción se convierte en x1 + 4x2 + S1 ≤ 20, donde S1 es la cantidad de "holgura" o "recurso no utilizado". Así, el número de variables de holgura positivas dependerá del número de restricciones "menores o iguales". Las variables de holgura positivas formarán parte de la solución inicial.

Holguras negativas (variables de exceso). Cuando una restricción es de tipo "mayor o igual que" (≥), se introduce una variable de holgura negativa (o variable de exceso) para convertir la desigualdad en una ecuación. Esta variable representa el "exceso" en esa restricción. Por ejemplo, en una segunda restricción 2x1 + 3x2 ≥ 5, al agregar una variable de exceso h1, la restricción se convierte en 2x1 + 3x2 – h1 = 5, donde h1 es la cantidad en que la combinación de x1 y x2 excede el valor de 5.

Variables artificiales. Las variables artificiales se introducen para manejar restricciones que no tienen una solución básica factible inicial obvia. Específicamente, se utilizan para restricciones de tipo "igual a" (=) o "mayor o igual que" (≥) cuando no se puede identificar una solución básica factible inicial. Estas variables permiten comenzar el método simplex en una solución básica factible, aunque su objetivo es que, en la solución óptima, tengan un valor de cero. Si una variable artificial tiene un valor positivo en la solución óptima, ello indica que el problema original no tiene solución factible. Por ejemplo, en el caso de la restricción 2x1 + 3x2 = 6, se podría introducir una variable artificial A1 para representar una solución básica factible inicial, aunque el objetivo es que A1 sea cero en la solución final. De manera que, en caso de que la restricción sea del tipo "igual a", por ejemplo, 2x1 + 3x2 = 6, la restricción quedaría 2x1 + 3x2 + A1= 6; por otro lado, si la restricción es del tipo "mayor o igual que", por ejemplo, 2x1 + 3x2 ≥ 5, la restricción quedaría 2x1 + 3x2 + A1 – h1 = 6, aplicando la variable de holgura negativa. Las variables artificiales formarán parte de la solución inicial.

Winston (2004) argumenta la importancia de mencionar que, al introducir estas variables, se debe ajustar la función objetivo para garantizar que las variables artificiales no afecten la solución óptima. Para el método simplex, esto se logra asignando un coeficiente muy grande (en el caso de minimización) o uno muy pequeño (en el caso de maximización) a las variables artificiales y un coeficiente de cero en el caso de las variables de holgura en la función objetivo.

En el ejercicio de ejemplo se intenta maximizar la función Z; por lo tanto, se asigna un valor de -10000 al coeficiente de la variable artificial. Queda de la forma siguiente:

$$Max\ Z = 2x_1 + 3x_2 - 10000A_1 + 0h_1 + 0S_1$$

$$\begin{aligned} S.a. \quad & x_1 + 4x_2 + A_1 - h_1 = 20 \\ & 2x_1 + 3x_2 + S_1 = 30 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Igualando a cero la función objetivo

$$Z - 2x_1 - 3x_2 + 10000A_1 - 0h_1 - 0S_1 = 0$$

$$\begin{aligned} S.a. \quad & x_1 + 4x_2 + A_1 - h_1 = 20 \\ & 2x_1 + 3x_2 + S_1 = 30 \end{aligned}$$

Así, la automatización del método mediante la subrutina "*Matriz_Inicial*" en código VBA generan la siguiente tabla inicial del método simplex, iniciando en la fila 10+m+2, columna 4, en tanto la posición final depende de las variables de holgura positivas, negativas y artificiales para los valores de la matriz inicial, quedando de la siguiente manera:

Figura 5.
Matriz inicial

	x1	x2	A1	h1	S1	Solución
Z	-2	-3	10000	0	0	0
A1	1	4	1	-1	0	20
S1	2	3	0	0	1	30

Nota: elaboración propia.

Para obtener una solución inicial, se utilizan operaciones con reglones de las variables artificiales, a fin de que su coeficiente sea 1 y todos los elementos de la columna sean 0, en este caso, sólo la fila de la función objetivo, tal y como se realiza en el método de Gauss - Jordán.

Figura 6.
Matriz inicial con solución básica.

	x1	x2	A1	h1	S1	Solución
Z	-10002	-40003	0	10000	0	-200000
A1	1	4	1	-1	0	20
S1	2	3	0	0	1	30

Nota: elaboración propia.

La subrutina privada encargada de realizar la matriz inicial con una solución básica es "*Matriz_*

Inicial",

A continuación, se presenta el código vba de esta subrutina, que resulta ser un componente clave para configurar y resolver problemas de programación lineal utilizando el método simplex en una hoja de cálculo Excel. Su funcionamiento puede describirse atendiendo los siguientes pasos:

Se definen varias variables para manejar el número de variables *nV*, restricciones *nR*, así como para contar holguras, variables artificiales y otros componentes del modelo.

El código vuelve a leer los valores de las variables y restricciones directamente de la hoja de cálculo, aunque bien podrían ser parámetros de entrada.

Se preparan matrices y arreglos para representar la matriz del problema, etiquetas para filas y columnas, adaptándose al número de variables y restricciones.

El tipo de cada restricción (<=, =, >=) se identifica mediante bucles, para configurar las holguras y variables artificiales correspondientes.

Dependiendo de si el problema es de minimización o maximización, se establecen los coeficientes adecuados en la matriz.

La matriz se completa con los coeficientes de las restricciones y la función objetivo, incorporando las condiciones para holguras y variables artificiales.

Si el modelo incluye variables artificiales, se realiza un ajuste en la matriz con un valor 'M' grande, un enfoque común en el método simplex para tratar ciertas restricciones.

Finalmente, la matriz preparada se pasa a otra subrutina *"SimplexM"*, a fin de realizar el desarrollo del método simplex.

"Matriz_Inicial" es una subrutina estructurada diseñada para preparar y configurar todos los aspectos inherentes a un problema de programación lineal para su posterior resolución mediante el método simplex. Desde la preparación de la matriz hasta el manejo de variables artificiales y la interacción con la interfaz de usuario, esta subrutina juega un papel crucial en el establecimiento de las bases que permitan una ejecución eficiente y correcta del algoritmo de optimización.

Código VBA subrutina *Matriz_Inicial*

```
Private Sub Matriz_Inicial()
    Dim nV, nR, i, j, Ren, Col As Integer
    Dim nIgual, nMenor_Igual, nMayor_Igual As
Integer
    Dim nArtificiales As Integer
    Dim nHolguras As Integer
    Dim nHolguras_art As Integer
    Dim nFilas, nColumnas As Integer
    Dim cDesigualdad As String
    Dim Valor_M As Double
    Dim nMultiplicador As Integer
    Dim n_Ms As Integer
    Dim nCol_Pivote, nRen_Pivote As Integer
    Dim nInverso As Double
    Dim lShowTbl As Boolean
    nTab = 0
    nV = Cells(2, 2).Value
    nR = Cells(4, 2).Value
    nHolguras = 0
    nHolguras_art = 0
    nArtificiales = 0
    nIgual = 0
    nFilas = nR + 1
    ReDim Filas(nFilas) As String
    If Cells(7, 2).Value = "Min" Then
    Filas(1) = "-Z"
     Else
    Filas(1) = "Z"
    End If
    For i = 1 To nR
      cDesigualdad = Cells(9 + i, 3 + nV + 1).Value
          If cDesigualdad = "<=" Then
            nHolguras = nHolguras + 1
            Filas(i + 1) = "S" & nHolguras
          Else
            If cDesigualdad = "=" Then
              nArtificiales = nArtificiales + 1
              Filas(i + 1) = "A" & nArtificiales
            Else
               nArtificiales = nArtificiales + 1
               Filas(i + 1) = "A" & nArtificiales
               nHolguras_art = nHolguras_art + 1
            End If
          End If
    Next i
    nColumnas = nV + nHolguras + nArtificiales +
nHolguras_art + 1
    ReDim Columnas(nColumnas) As String
    ReDim Matriz(nFilas, nColumnas) As Double
    i = 1
    nHolguras = 0
    nHolguras_art = 0
    nArtificiales = 0
    j = 1
    While j <= nColumnas
    If j = nColumnas Then
      Columnas(j) = "Solución"
```

(RCIA) | vol. 2(2):diciembre 2023

ISSN: En trámite

(RCIA) | vol. 2(2):diciembre 2023

```
    Else
    If j <= nV Then
     Columnas(j) = "x" & j
    Else
     cDesigualdad = Cells(9 + i, 3 + nV + 1).Value
     If cDesigualdad = "<=" Then
       nHolguras = nHolguras + 1
       Columnas(j) = "S" & nHolguras
     Else
       If cDesigualdad = "=" Then
         nArtificiales = nArtificiales + 1
         Columnas(j) = "A" & nArtificiales
       Else
         nArtificiales = nArtificiales + 1
         Columnas(j) = "A" & nArtificiales
         nHolguras_art = nHolguras_art + 1
         j = j + 1
         Columnas(j) = "h" & nHolguras_art
       End If
     End If
     i = i + 1
    End If
    End If
    j = j + 1
    Wend
    For Ren = 1 To nFilas - 1
     Col = 1
     While Col <= nColumnas - 1
     cDesigualdad = Cells(9 + Ren, 3 + nV + 1).Value
      If Col <= nV Then
       Matriz(Ren + 1, Col) = Cells(9 + Ren, 3 + Col)
      Else
       If Filas(Ren + 1) = Columnas(Col) Then
         Matriz(Ren + 1, Col) = 1
       Else
          If (cDesigualdad = ">=" And Mid(Columnas(Col), 1, 1) = "h") Then
           Matriz(Ren + 1, Col) = -1
         Else
          Matriz(Ren + 1, Col) = 0
         End If
       End If
      End If

      Col = Col + 1
     Wend
     Matriz(Ren + 1, nColumnas) = Cells(9 + Ren, 3 + nV + 2).Value
    Next Ren
    Valor_M = 10000
    If nArtificiales > 0 Then
     FrmDatos.Show
     Valor_M = Cells(1, 2).Value
    End If
    If Cells(7, 2).Value = "Min" Then
     nMultiplicador = 1
    Else
     nMultiplicador = -1
    End If
    For Col = 1 To nColumnas - 1
      If Col <= nV Then
        Matriz(1, Col) = Cells(7, 3 + Col).Value * nMultiplicador
      Else
        If Mid(Columnas(Col), 1, 1) = "A" Then
          If Cells(7, 2).Value = "Min" Then
            Matriz(1, Col) = Valor_M * nMultiplicador
          Else
            Matriz(1, Col) = Valor_M * nMultiplicador * -1
          End If
        Else
         Matriz(1, Col) = 0
        End If
      End If
    Next Col
    Cells(8, 4).Select
    Call Dibuja_Tabla(Matriz, Filas, Columnas, nColumnas, nR)
    If nArtificiales > 0 Then
    n_Ms = 0
    For i = 1 To nColumnas - 1
     If (i > nV) Then
      If (Mid(Columnas(i), 1, 1) = "A" And Matriz(1, i) <> 0) Then
        n_Ms = n_Ms + 1
        nCol_Pivote = i
       Exit For
      End If
     End If
    Next i
    While n_Ms > 0
      For j = 2 To nFilas
        If Matriz(j, nCol_Pivote) = 1 Then
         nRen_Pivote = j
        End If
       Next j
       For i = 1 To nRen_Pivote - 1
         nInverso = Matriz(i, nCol_Pivote) * -1
         For j = 1 To nColumnas
          Matriz(i, j) = nInverso * Matriz(nRen_Pivote, j) + Matriz(i, j)
         Next j
       Next i
       For i = nFilas To nRen_Pivote + 1 Step -1
       nInverso = Matriz(i, nCol_Pivote) * -1
         For j = 1 To nV + nR + 1
          Matriz(i, j) = nInverso * Matriz(nRen_Pivote, j) + Matriz(i, j)
         Next j
       Next i
      n_Ms = 0
      For i = 1 To nColumnas - 1
       If (i > nV) Then
         If (Mid(Columnas(i), 1, 1) = "A" And Ma-
```

```
triz(1, i) <> 0) Then
            n_Ms = n_Ms + 1
               nCol_Pivote = i
            Exit For
          End If
          End If
        Next i
      Wend
    Call Dibuja_Tabla(Matriz, Filas, Columnas, nColumnas, nR)
    End If
          Call SimplexM(Matriz, Filas, Columnas, nFilas, nColumnas)
End Sub
```

La macro "Matriz_Inicial" es una subrutina en código vba que genera la solución básica inicial y prepara la hoja de cálculo Excel para resolver un problema de programación lineal utilizando el método simplex. Recoge datos de entrada, como el número de variables y restricciones, configura matrices para la función objetivo y restricciones, y añade variables de holgura y artificiales según sea necesario. Determina si el problema es de maximización o minimización y ajusta los coeficientes de la función objetivo. Luego, dibuja una tabla inicial en la hoja de cálculo y aplica transformaciones para manejar variables artificiales en caso de que estén presentes. Una vez que obtiene la solución inicial, invoca otra subrutina llamada "*SimplexM*" para ejecutar el algoritmo simplex y encontrar la solución óptima al problema.

Una vez cubiertos los tres momentos iniciales de preparación, se cuenta con todos los elementos para el desarrollo del método simplex. El llamado a la subrutina o macro "*SimplexM*" desarrolla el método simplex, como se explicó previamente. Es necesario determinar la variable entrante en el sistema, lo que se realiza identificando la columna con el coeficiente más negativo en la fila de la función objetivo. La variable correspondiente a esta columna será la variable entrante a la base y la columna se denominará columna pivote. Asimismo, se debe determinar la variable saliente realizando la prueba del cociente mínimo, para lo que únicamente se utilizan los coeficientes positivos de las restricciones de la columna pivote a fin de dividir los valores correspondientes de la columna solución.

	x1	x2	A1	h1	S1	Solución
Z	-10002	-40003	0	10000	0	-200000
A1	1	4	1	-1	0	20
S1	2	3	0	0	1	30

Figura 7. *Columna, fila y elemento pivote.* Nota: elaboración propia.

En la fila Z, el valor más negativo se encuentra en la columna correspondiente a la variable x_2, lo que indica que x_2 es la variable que entra en la base. Para determinar la variable que sale, calculamos los cocientes de los elementos del lado derecho con los de la columna de x_2:

$$\frac{20}{4} = 5 \; y \; \frac{30}{3} = 10.$$

El menor de estos cocientes es 5, que se encuentra en la fila de A_1. Por lo tanto, *A*1 es la variable que sale de la base. El cruce de la columna de x_2 y la fila de A_1 nos da el elemento pivote, que es el coeficiente en la intersección de la columna entrante y la fila saliente.

Una vez que se cuenta con el elemento pivote, se realizan operaciones de fila para hacer que el elemento pivote sea 1 y todos los otros elementos de la columna pivote sean 0.

Figura 10.
Operaciones de fila

	x1	x2	A1	h1	S1	Solución
Z	-1 1/4	0	10000 3/4	- 3/4	0	15
x2	1/4	1	1/4	- 1/4	0	5
S1	1 1/4	0	- 3/4	3/4	1	15

Nota: elaboración propia.

Al terminar de realizar las operaciones de fila, es necesario verificar que en la fila de Z (función objetivo) ya no existan valores negativos. Si éste es el caso, el método concluye, de lo contrario, realizamos otra vez el cuarto momento. Así, en la siguiente iteración la variable que entra es X1 y la variable que sale es S1.

Figura 11.
Elección de la columna, fila y elemento pivote.

	x1	x2	A1	h1	S1	Solución
Z	-1 1/4	0	10000 3/4	- 3/4	0	15
x2	1/4	1	1/4	- 1/4	0	5
S1	1 1/4	0	- 3/4	3/4	1	15

Nota: elaboración propia.

Realizando las operaciones de renglón, se observa que ya no existen valores negativos en la fila de Z, por lo tanto, el método termina, obteniendo un valor óptimo para la maximización de la función Z = 30, con x1=12 y x2=2.

(RCIA) | vol. 2(2):diciembre 2023

ISSN: En trámite

(RCIA) | vol. 2(2):diciembre 2023

Código VBA subrutina *SimplexM*

```
Private Sub SimplexM(Matriz, Filas, Columnas, nFilas, nColumnas)
Dim i, j, nNegativos, nCol_Pivote, nRen_Pivote, nTab As Integer
Dim nPivote, nMenor, nInverso As Double
Dim lPaso As Boolean
Dim Solucion As String
nTab = 2
lPaso = True
nNegativos = 0
For i = 1 To nColumnas - 1
 If Matriz(1, i) < 0 Then
  nNegativos = nNegativos + 1
 End If
Next i
While nNegativos > 0
   nMenor = 0
   nCol_Pivote = 0
   For i = 1 To nColumnas - 1
      If Matriz(1, i) < nMenor Then
         nMenor = Matriz(1, i)
         nCol_Pivote = i
      End If
   Next i
 nRen_Pivote = 0
 nMenor = 1000000
   For j = 2 To nFilas
    If Matriz(j, nCol_Pivote) > 0 Then
      If (Matriz(j, nColumnas) / Matriz(j, nCol_Pivote)) < nMenor Then
         nMenor = (Matriz(j, nColumnas) / Matriz(j, nCol_Pivote))
         nRen_Pivote = j
       End If
     End If
   Next j
If nRen_Pivote > 0 Then
   Filas(nRen_Pivote) = Columnas(nCol_Pivote)
   nPivote = Matriz(nRen_Pivote, nCol_Pivote)
   If nPivote <> 0 Then
      For j = 1 To nColumnas
         Matriz(nRen_Pivote, j) = Matriz(nRen_Pivote, j) / nPivote
      Next j

      For i = 1 To nRen_Pivote - 1
        nInverso = Matriz(i, nCol_Pivote) * -1
        For j = 1 To nColumnas
         Matriz(i, j) = nInverso * Matriz(nRen_Pivote, j) + Matriz(i, j)
        Next j
      Next i

      For i = nFilas To nRen_Pivote + 1 Step -1
      nInverso = Matriz(i, nCol_Pivote) * -1
         For j = 1 To nColumnas
            Matriz(i, j) = nInverso * Matriz(nRen_Pivote, j) + Matriz(i, j)
         Next j
      Next i
     Call Dibuja_Tabla(Matriz, Filas, Columnas, nColumnas, nFilas - 1)

      nNegativos = 0
      For i = 1 To nColumnas - 1

       If Matriz(1, i) < 0 Then
        nNegativos = nNegativos + 1
       End If

      Next i
    Else
      nNegativos = 0
      lPaso = False
   End If
 Else
 nNegativos = 0
 lPaso = False
  End If
 Wend
 If lPaso Then
 Solucion = "Solución:  " & Chr(13) & Chr(13)
 Cells(9, 24 + nFilas - 1 + 3).Value = "Solución"
 nTab = 1
 For j = 1 To nFilas
 If j = 1 Then
  Cells(9 + nTab, 24 + nFilas - 1 + 3).Value = "Z="
   Cells(9 + nTab, 24 + nFilas - 1 + 4).Value = Abs(Matriz(j, nColumnas))
  Call Borde_Grueso(9 + nTab, 24 + nFilas - 1 + 4, "D")
  Solucion = Solucion & "      Z=" & Round(Matriz(j, nColumnas), 4) & Chr(13)
  nTab = nTab + 1
 Else
  If Mid(Filas(j), 1, 1) = "x" Then
   Cells(9 + nTab, 24 + nFilas - 1 + 3).Value = Filas(j) & "= "
  Cells(9 + nTab, 24 + nFilas - 1 + 4).Value = Matriz(j, nColumnas)
 ' Call Borde_Grueso(9 + nTab, 4 + nR + 4, "D")
  Solucion = Solucion & "     " & Filas(j) & "=" & Round(Matriz(j, nColumnas), 4) & Chr(13)
  nTab = nTab + 1
  End If
 End If
 Next j
 MsgBox Solucion
 Else
  MsgBox "El Modelo no tiene solución"
 End If
End Sub
```

La subrutina "*SimplexM"* en vba está diseñada para ejecutar el método simplex. Ésta maneja meticulosamente el proceso de optimización lineal, desde la selección de variables pivote hasta la actualización de la matriz y la presentación de la solución óptima o la falta de ella. A continuación, se detalla la descripción del código en una serie de pasos:

Se definen múltiples variables para el manejo de la matriz del simplex, incluyendo contadores y variables para el pivote y otros elementos clave del proceso.

El código comienza revisando la primera fila de la matriz (*función objetivo*) para identificar elementos negativos, los cuales indican la necesidad de más iteraciones en el método simplex.

Se busca el elemento más negativo en la fila de costos. El índice de este elemento define la columna de pivote, que es la variable que entrará en la base de la solución.

Una vez identificada la columna de pivote, se busca en las filas, excluyendo la fila de costos, para encontrar la fila de pivote, es decir, aquella que determinará qué variable sale de la base. Esto se hace mediante la regla del mínimo, comparando las proporciones de los elementos en la columna de solución y la columna de pivote.

Si se encuentra una fila de pivote válida, la subrutina actualiza esta fila para que el elemento de pivote sea 1 y ajusta los demás elementos de la fila en consecuencia.

Se actualizan todas las otras filas de la matriz para que el elemento en la columna de pivote sea cero, lo que refleja el cambio en la base de la solución.

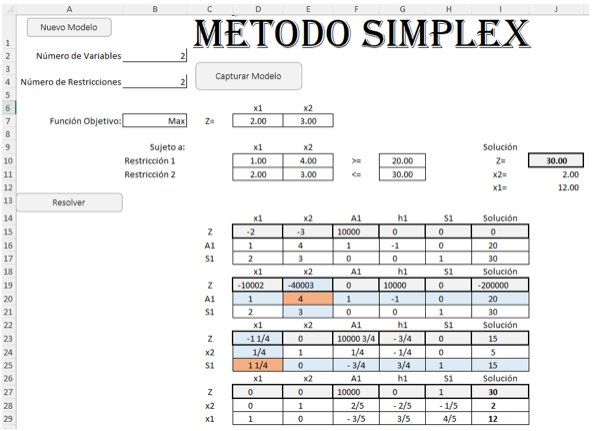
El proceso de encontrar un pivote y actualizar la matriz se repite hasta que no haya más elementos negativos en la fila *Z* de costos, lo que indica que se ha alcanzado la solución óptima.

Al finalizar el proceso, si se encuentra una solución, la subrutina muestra los valores de la función objetivo (*Z*) y las variables de decisión en un mensaje. En caso contrario, informa que el modelo no tiene solución.

Durante el proceso, la subrutina interactúa con la hoja de cálculo, llamando a las subrutinas *"Dibuja_Tabla"* y *"Borde_Grueso"* para dar formato y visualizar cada paso del método simplex y mostrar los resultados finales.

Una vez terminado el proceso de ejecución de la subrutina "*SimplexM*", la hoja de cálculo mostrará los resultados como aparecen en la figura 8, con todo el desarrollo y la solución óptima del problema de programación lineal.

Figura 8.
Desarrollo completo y automatizado del método simplex.



Nota: elaboración propia

Resultados

Una vez concluido el desarrollo completo y automatizado del método simplex, se pueden destacar los siguientes puntos:

Resolución automatizada de problemas: el código vba proporciona una solución automatizada para resolver problemas de programación lineal utilizando el método simplex. Esto permite a los estudiantes centrarse en la comprensión conceptual en lugar de en los cálculos manuales tediosos.

Visualización clara: al representar la matriz y las soluciones en Excel, los estudiantes pueden visualizar claramente cada paso del proceso, lo que facilita la comprensión.

Interactividad: los estudiantes pueden ingresar diferentes problemas y ver cómo se resuelven en tiempo real, lo que fomenta su aprendizaje activo.

Comparación de resultados académicos con y sin el uso de VBA

A menudo los estudiantes se sienten abrumados por los cálculos manuales y pueden cometer errores fácilmente, lo que lleva a una comprensión incorrecta del método simplex.

ISSN: En trámite

Como sugiere Djamila (2017) en su artículo "Excel spreadsheet in teaching numerical methods", en el que discute la importancia de comprender los algoritmos de métodos numéricos para estudiantes universitarios, aunque el cálculo manual es importante para entender el procedimiento, puede ser tedioso y propenso a errores, especialmente cuando se considera el procedimiento de iteración utilizado en muchos métodos numéricos. El artículo se centra en cómo, con el uso de hojas de cálculo de Excel, se ofrece un nivel inicial de programación que puede ser utilizado por los estudiantes sin distracciones para escribir códigos. Esto sugiere que el uso de herramientas computacionales puede mejorar la comprensión de los estudiantes en áreas técnicas, lo cual podría ser extrapolable al aprendizaje del método simplex.

Al programar en vba, los estudiantes pueden centrarse en la lógica y la conceptualización del método simplex; bajo esta premisa, los errores de cálculo se reducen significativamente, lo que lleva a una mejor comprensión y a mejores resultados académicos.

Ventajas de la adaptabilidad y personalización del método para diferentes niveles educativos

Según Taha (2012), el método simplex se describe como una técnica matemática fundamental para resolver problemas de programación lineal. Su adaptabilidad y personalización lo hacen particularmente valioso en el ámbito educativo, donde los estudiantes de diferentes niveles pueden beneficiarse de su aplicación. A continuación, se desarrollan las ventajas mencionadas.

Flexibilidad: la programación en vba permite a profesores y estudiantes adaptar el método simplex a una variedad de problemas de programación lineal. Por ejemplo, en un nivel introductorio, el código puede simplificarse para resolver problemas con pocas variables y restricciones, facilitando la comprensión de los conceptos básicos. En niveles más avanzados, el mismo código puede expandirse para abordar problemas más complejos con múltiples restricciones y variables, proporcionando una experiencia de aprendizaje progresiva y desafiante.

Personalización: los profesores tienen la libertad de modificar el código vba para enfocarse en los aspectos específicos del método simplex más relevantes para sus cursos. Esto significa que pueden diseñar ejercicios que resalten la importancia de ciertas técnicas matemáticas o que demuestren la aplicación del método simplex en contextos del mundo real, como la logística o la gestión de recursos.

Aplicabilidad en diferentes cursos: el método simplex no se limita a las clases de matemáticas. Se puede integrar en una amplia gama de cursos, que incluyen desde la toma de decisiones en la administración de empresas hasta la optimización en la ingeniería. La capacidad de adaptar el código vba a diferentes aplicaciones hace que sea una herramienta versátil y práctica en diversas disciplinas académicas.

Inclusión de funcionalidades adicionales: a medida que los estudiantes progresan en su educación, pueden encontrarse con conceptos más avanzados, como los de dualidad en programación lineal o la necesidad de abordar problemas de programación entera. El código vba puede ser enriquecido con estas funcionalidades adicionales, proporcionando una plataforma para que los estudiantes apliquen lo que han aprendido a situaciones más complejas y cercanas a la investigación avanzada o a las aplicaciones industriales.

La personalización y adaptabilidad del método simplex implementado a través de vba permiten una experiencia de aprendizaje dinámica y profundamente integrada con las necesidades y niveles de habilidad de los estudiantes. Esta flexibilidad es esencial en aras de prepararlos para aplicar la programación lineal en situaciones reales y complejas que encontrarán en sus carreras profesionales.

Conclusiones

La adopción de vba en la instrucción de matemáticas aplicadas responde a la necesidad actual de métodos educativos que enfaticen la interacción práctica y el aprendizaje significativo. Este trabajo contribuye a la diversificación de las estrategias pedagógicas y alienta la exploración de nuevas formas de integrar la tecnología en la educación. Destacamos la relevancia de la adaptabilidad tecnológica en la enseñanza, evidenciando cómo la programación en Excel con vba puede ser una herramienta eficaz en el proceso educativo. Este enfoque flexible permite una enseñanza personalizada que se alinea con los objetivos curriculares y fomenta la aplicación de conocimientos en escenarios reales y diversos.

La implementación de esta propuesta de en-

señanza se presenta como una alternativa pedagógica orientada a la mejor comprensión de la programación lineal, con un enfoque particular en los estudiantes de ingeniería y administración. Dicho método promueve un aprendizaje interactivo y centrado en el alumno, alejándose de las prácticas tradicionales centradas en la exposición del instructor.

Para apoyar la adaptación y mejora continua de este recurso educativo, se proporciona el código completo (desarrollado en su totalidad por el autor), anexando las subrutinas complementarias a los procesos principales descritos a lo largo de este documento. Con esto, reafirmo mi compromiso con una educación abierta y colaborativa, e invito a la comunidad educativa a participar activamente en la refinación e innovación de nuestras estrategias didácticas. Animo a los educadores a seguir explorando y ampliando el uso de herramientas tecnológicas que permitan enriquecer el aprendizaje y preparar a los estudiantes para los retos tecnológicos actuales y futuros. La interacción con el código y la visualización inmediata de los resultados fomentan una comprensión intuitiva de los conceptos. La capacidad de Excel para mostrar la tabla del simplex y otros elementos gráficos hace más tangible y accesible el proceso de optimización. Además, la automatización de los cálculos permite a los estudiantes profundizar en la teoría y las aplicaciones del método, más allá de la aritmética. La accesibilidad de Excel promueve la práctica independiente y el aprendizaje autodirigido. Así, no sólo se profundiza el conocimiento de los estudiantes sobre esta herramienta de *software* esencial, sino que también se cultivan habilidades de análisis y razonamiento lógico-deductivo transferibles a una amplia gama de tareas profesionales. Al aprender a programar en vba, los estudiantes ejercitan la lógica algorítmica, lo que mejora su capacidad para estructurar y desglosar problemas complejos en componentes manejables. Esta habilidad es fundamental para la resolución de problemas y la toma de decisiones. La programación fomenta un enfoque metódico para el análisis de situaciones, promoviendo un pensamiento sistemático y secuencial, pero también adaptable y creativo, al enfrentar desafíos inesperados. Además, el acto de programar puede revelar la belleza de la matemática aplicada, mostrando cómo las abstracciones numéricas y las fórmulas se traducen en soluciones prácticas y efectivas.

Por lo anterior, este estudio cumple con los objetivos de facilitar la comprensión de la programación lineal y fomentar habilidades analíticas valiosas a través de la implementación del método simplex en vba para Excel. Aunque las limitaciones reconocidas sugieren un alcance de aplicación específico, la contribución de este trabajo al campo educativo y su potencial para adaptarse y evolucionar con las necesidades cambiantes de estudiantes y profesionales, subrayan su valor y relevancia en la enseñanza contemporánea de la programación lineal.

Futuras investigaciones

Alentamos la continuación de la investigación en este campo, buscando formas innovadoras de explotar el potencial de herramientas tecnológicas en la enseñanza de la programación lineal, una disciplina con aplicaciones extensas y significativas en múltiples aspectos de la vida cotidiana y profesional.

Si bien el código presentado es funcional, siempre hay espacio para mejoras y optimizaciones. Sería beneficioso explorar la integración de interfaces gráficas más intuitivas, la incorporación de funciones adicionales o la adaptación del código a problemas más complejos. Además, sería interesante investigar la eficacia de esta herramienta en comparación con otros métodos de enseñanza o *software* dedicado. De la misma manera, la implementación de comentarios dentro del mismo podría facilitar su comprensión para aquellos menos familiarizados con vba. Aunado a ello, la realización de talleres o sesiones prácticas donde los estudiantes puedan modificar y jugar con el código podría enriquecer aún más su experiencia de aprendizaje.

Investigaciones futuras podrían incluir estudios empíricos que evalúen el impacto de esta herramienta en el rendimiento y la comprensión de los estudiantes, así como su capacidad para transferir habilidades aprendidas a problemas del mundo real.

Por último, se sugiere la integración de metodologías de aprendizaje mixto y enfoques de gamificación para examinar cómo estas estrategias pueden aumentar el compromiso y la retención de conocimientos en los estudiantes.

(RCIA) | vol. 2(2):diciembre 2023

ISSN: En trámite

Referencias

Aliane, N. (2008). Spreadsheet-based control system analysis and design [Focus on Education]. *Control Systems*, ieee, 28, 108-113. https://doi.org/10.1109/MCS.2008.927960Almenar

Alvarado Boirivant, J. (2009). La programación lineal aplicación de las pequeñas y medianas empresas. *Reflexiones*, *88*(1), 89-105. Universidad de Costa Rica. Recuperado de http://www.redalyc.org/articulo.oa?id=72912559007

Arboleda Molina, O., y Sotelo, S. (2016). Construcción de aplicativos de programación por restricciones en Microsoft Solver Foundation y Windows Azure. *Scientia Et Technica*, *21*(4), 336-341. Universidad Tecnológica de Pereira.

Barr, G., y Scott, L. (2008). A new approach to teaching fundamental statistical concepts and an evaluation of its application at uct. https://hdl.handle.net/10520/EJC99106

Bofill-Pérez, M., García-Noa, E., & Sariego-Toledo, Y. (2019). Optimización en la producción de surtidos de helados Alondra. *Tecnología Química*, 39(3), 508-523. Universidad de Oriente.

Coll Serrano, V., y Blasco Blasco, O. (2010). El uso de gráficos interactivos en Excel para facilitar la comprensión de conceptos básicos de Estadística. *@tic. revista d'innovació educativa*, (5), 30-34. Universitat de València.

Cruz T, E. A., Restrepo, J. H., y Medina V, P. D. (2007). Un problema logístico de ruteo de vehículos y una solución con Solver de Excel: Un caso de estudio. *Scientia Et Technica*, *XIII*(37), 369-372. Universidad Tecnológica de Pereira.

Dantzig, George (1963). *Linear Programming and Extensions.* Princeton University Press. https://doi.org/10.1515/9781400884179

Djamila, H. (2017). Excel spreadsheet in teaching numerical methods. *Journal of Physics: Conference Series*, *890*(1), 012093. https://doi.org/10.1088/1742-6596/890/1/012093

Ferro, C., Martínez, A. I., y Otero, M. C. (2009). Ventajas del uso de las tic en el proceso de enseñanza aprendizaje desde la óptica de los docentes universitarios españoles. *Edutec: Revista Electrónica de Tecnología Educativa, 29.*

Fuson, K. C., Carroll, W. M., y Drueck, J. V. (2000). Achievement results for second and third graders using the Standards-based curriculum Everyday Mathematics. *Journal for Research in Mathematics Education*, 31.

https://globaljournals.org/eBooks/A_Trajectory_for_the_Teaching_and_Learning_of_the_Didactics_of_Mathematics_using_ICT.pdf

Karaulov, A., Nemtzev, D., Konkov, A., y Shekhov, V. (2021). Study of the soil stability theory problems by the simplex method. *Journal of Physics: Conference Series*, *2131*(3), 032019. https://dx.doi.org/10.1088/1742-6596/2131/3/032019

Llongo, V., y Hernández Sancho, F. (2009). Excel como herramienta docente de las asignaturas de Microeconomía. *@tic. revista d'innovació educativa*, (3), 108-114. Universidad de Valencia. http://www.redalyc.org/articulo.oa?id=349532299017

López Noriega, M., Lagunes Huerta, C., y Herrera Sánchez, S. (2006). Excel como una herramienta asequible en la enseñanza de la Estadística. *Teoría de la Educación. Educación y Cultura en la Sociedad de la Información*, *7*(1). Universidad de Salamanca.

Martínez P. E. (2013). Un modelo de programación discreta para minimizar el costo de la transportación de cargas. *Economía y Desarrollo, 149*(1), 158-165. Universidad de La Habana.

Microsoft (2023). Especificaciones y límites de Excel. https://support.microsoft.com/es-es/office/especificaciones-y-l%C3%ADmites-de-excel-1672b34d-7043-467e-8e27-269d656771c3 Fecha de consulta: 01 de octubre, 2023

Orozco, J. (2004). Uso Pedagógico de los programas Derive 6.1 y Cabri Geometry II plus, en las clases de Matemáticas. Proyecto de Innovaciones Tecnológicas en la enseñanza de las matemáticas y Ciencias. Colombia. URL: http://www.scm.org.co/Subidos/855.Resumen.pdf

Salazar López, B. (2019, junio 11). Método Simplex. Ingeniería Industrial Online. https://www.ingenieriaindustrialonline.com/investigacion-de-operaciones/metodo-simplex/, Fecha de consulta: 05 de octubre 2023.

Shen, W., Nie, Y., y Zhang, H. M. (2007). Dynamic Network Simplex Method for Designing Emergency Evacuation Plans. *Transportation Research Record, 2022*(1), 83-93. https://doi.org/10.3141/2022-10

Taha, H. A. (2012). *Investigación de operaciones* (9a ed.). Pearson Educación. ISBN: 978-607-32-0796-6.

Torres, M. (2016). *Aplicaciones vba con Excel.* Editorial Macro EIRL. ISBN 978-612-304-265-3, e-ISBN 978-612-304-349-0.

Ulloa, L., y Protti, M. (2005). *Investigación de Operaciones.* Universidad Nacional a Distancia Costa Rica

Winston, W. L. (2004). *Operations research: Applications and algorithms.* Brooks/Cole-Thomson Learning.

Anexo 1.

Subrutinas complementarias para el desarrollo del método simplex.

Declaración de variable pública *nTab*

```
Public nTab As Integer
```

Código VBA subrutina *Limpiar*

```
Sub Limpiar()
    Range("A6:xl000").Select
      With Selection.Interior
      .Pattern = xlSolid
      .PatternColorIndex = xlAutomatic
      .ThemeColor = xlThemeColorDarkl
      .TintAndShade = 0
      .PatternTintAndShade = 0
    End With
    Selection.Font.Bold = False
    Selection.Borders(xlEdgeLeft).LineStyle = xl-
None
     Selection.Borders(xlEdgeTop).LineStyle = xl-
None
    Selection.Borders(xlEdgeBottom).LineStyle =
xlNone
     Selection.Borders(xlEdgeRight).LineStyle =
xlNone
    Selection.Borders(xlInsideVertical).LineStyle
= xlNone
     Selection.Borders(xlInsideHorizontal).Line-
Style = xlNone
    Selection.ClearContents
      With Selection.Validation
      .Delete
        .Add Type:=xlValidateInputOnly, Alert-
Style:=xlValidAlertStop, Operator _
      :=xlBetween
      .IgnoreBlank = True
      .InCellDropdown = True
      .InputTitle = ""
      .ErrorTitle = ""
      .InputMessage = ""
      .ErrorMessage = ""
      .ShowInput = True
      .ShowError = True
    End With
     If ActiveSheet.Shapes.Count >= 3 Then
     ActiveSheet.Shapes.Range(Array("Boton")).
Select
     Selection.Delete
   End If
    Range("B2").Select
End Sub
```

Código VBA subrutina *Celda_Captura*

```
Private Sub Celda_Captura(F, C, Tipo)
   Cells(F, C).Select
   If Selection.Value = "" Then
     Selection.Value = 0
   End If
   Selection.Font.Bold = False
   Selection.Borders(xlEdgeLeft).LineStyle = xl-
Continuous
    Selection.Borders(xlEdgeTop).LineStyle = xl-
Continuous
   Selection.Borders(xlEdgeBottom).LineStyle =
xlContinuous
     Selection.Borders(xlEdgeRight).LineStyle =
xlContinuous
    If Tipo = "D" Then
     Selection.Style = "Comma"
    Selection.NumberFormat = "0.00"
   Else
    Selection.NumberFormat = "# ?/?"
    End If
    With Selection.Validation
      .Delete
          .Add Type:=xlValidateDecimal, Alert-
Style:=xlValidAlertStop, Operator _
       :=xlBetween, Formula1:="-1000000", For-
mula2:="1000000"
      .IgnoreBlank = True
      .InCellDropdown = True
      .InputTitle = ""
      .ErrorTitle = ""
      .InputMessage = ""
      .ErrorMessage = "Solo Valores Numéricos"
      .ShowInput = True
      .ShowError = True
    End With
End Sub
```

Código VBA subrutina *Desigualdad*

```
Private Sub Desigualdad(F, C, Cadena, Valor)
   Cells(F, C).Select
   With Selection.Validation
      .Delete
       .Add Type:=xlValidateList, AlertStyle:=xl-
ValidAlertStop, Operator:= _
      xlBetween, Formula1:=Cadena
      .IgnoreBlank = False
      .InCellDropdown = True
      .InputTitle = ""
      .ErrorTitle = ""
      .InputMessage = "" 'Cadena
        .ErrorMessage = "Tiene que incuir una
desigualdad : " & Cadena
      .ShowInput = True
```


ISSN: En trámite

(RCIA) | vol. 2(2):diciembre 2023

(RCIA) | vol. 2(2):diciembre 2023

```
        .ShowError = True
    End With
    Cells(F, C).Value = Valor
End Sub
```

Código VBA subrutina *Dibuja_Tabla*

```
Private Sub Dibuja_Tabla(Matriz, Filas, Columnas, nColumnas, nR)
    Dim i, Fil, Col As Integer
    nTab = nTab + 1
    For i = 1 To nColumnas
     Cells(9 + nTab + (nR + 1) * nTab + 1, 3 + i).Value = Columnas(i)
      Call Borde_Grueso(10 + nTab + (nR + 1) * nTab + 1, 3 + i, "F")
    Next i
    For Fil = 1 To nR + 1
     Cells(10 + nTab + (nR + 1) * nTab + Fil, 3).Value = Filas(Fil)
    For Col = 1 To nColumnas
       Cells(10 + nTab + (nR + 1) * nTab + Fil, 3 + Col).Value = Matriz(Fil, Col)
        Call Celda_Captura(10 + nTab + (nR + 1) * nTab + Fil, 3 + Col, "F")
    Next Col
    Next Fil
End Sub
```

Código VBA subrutina *Borde_Grueso*

```
Private Sub Borde_Grueso(F, C, Tipo)
    Cells(F, C).Select
       With Selection.Interior
       .Pattern = xlSolid
       .PatternColorIndex = xlAutomatic
       .ThemeColor = xlThemeColorDark1
       .TintAndShade = -4.99893185216834E-02
       .PatternTintAndShade = 0
    End With
    Selection.Font.Bold = True
        Selection.Borders(xlDiagonalDown).LineStyle = xlNone
     Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
       .LineStyle = xlContinuous
       .Weight = xlMedium
    End With
    With Selection.Borders(xlEdgeTop)
       .LineStyle = xlContinuous
       .Weight = xlMedium
    End With
    With Selection.Borders(xlEdgeBottom)
       .LineStyle = xlContinuous
       .Weight = xlMedium
    End With
    With Selection.Borders(xlEdgeRight)
       .LineStyle = xlContinuous
       .Weight = xlMedium
    End With
    Selection.Borders(xlInsideVertical).LineStyle = xlNone
     Selection.Borders(xlInsideHorizontal).LineStyle = xlNone
    If Tipo = "D" Then
     Selection.Style = "Comma"
     Selection.NumberFormat = "0.00"
    Else
     Selection.NumberFormat = "# ?/?"
    End If
    If Selection.Value = "" Then
     Selection.Value = 0
    End If
    With Selection.Validation
       .Delete
            .Add Type:=xlValidateDecimal, AlertStyle:=xlValidAlertStop, Operator _
        :=xlBetween, Formula1:="-1000000", Formula2:="1000000"
       .IgnoreBlank = True
       .InCellDropdown = True
       .InputTitle = ""
       .ErrorTitle = ""
       .InputMessage = ""
       .ErrorMessage = ""
       .ShowInput = True
       .ShowError = True
    End With
End Sub
```

(RCIA) | vol. 2(2):diciembre 2023

ISSN: En trámite