Escuela Politécnica Nacional

Programación 1

Proyecto Bimestre 2

Integrantes GRUPO 2:

Esteban López

Bryan Magarisca

Doménica Martínez

Sebastián Muñoz

Ronald Pilataxi

Melany Pullas

Jade Revelo

Paula Romo

GR2CD

2025

FUNCIONAMIENTO DEL PROGRAMA

Opción 1: Ingreso de Precios de Venta

Esta función implementa el registro de precios unitarios según los requisitos del proyecto, utiliza un sistema de validación robusto con bucles while que garantizan entradas numéricas positivas para ambos productos.

El código incluye manejo de errores para casos no exitosos, como valores negativos o caracteres no numéricos, aplicando `cin.clear()` y `cin.ignore()` para limpieza del buffer, estos precios se almacenan en variables globales (`precioMesa, precioSilla`) y se muestran confirmados con formato de dos decimales. La implementación sigue principios de código limpio con nombres descriptivos y una estructura modular.

La validación de entrada se realiza mediante un bucle infinito que solo se rompe cuando se introduce un valor válido, asegurando la robustez del sistema; para el precio de mesas, se verifica que el valor sea mayor que cero (`precioMesa` > 0) y de tipo numérico. Si la entrada no cumple estos requisitos, se muestra un mensaje de error específico y se limpia el buffer de entrada para evitar problemas en iteraciones posteriores, este mismo proceso se repite para el precio de sillas, garantizando consistencia en la validación de datos. Finalmente, los precios se muestran con formato monetario estándar (dos decimales) para una presentación profesional y clara al usuario. Asimismo, se implementó el uso de un try catch que se enfoca en validar las entradas diferentes a números, tales como letras o alguna otra simbología similar.

Salida

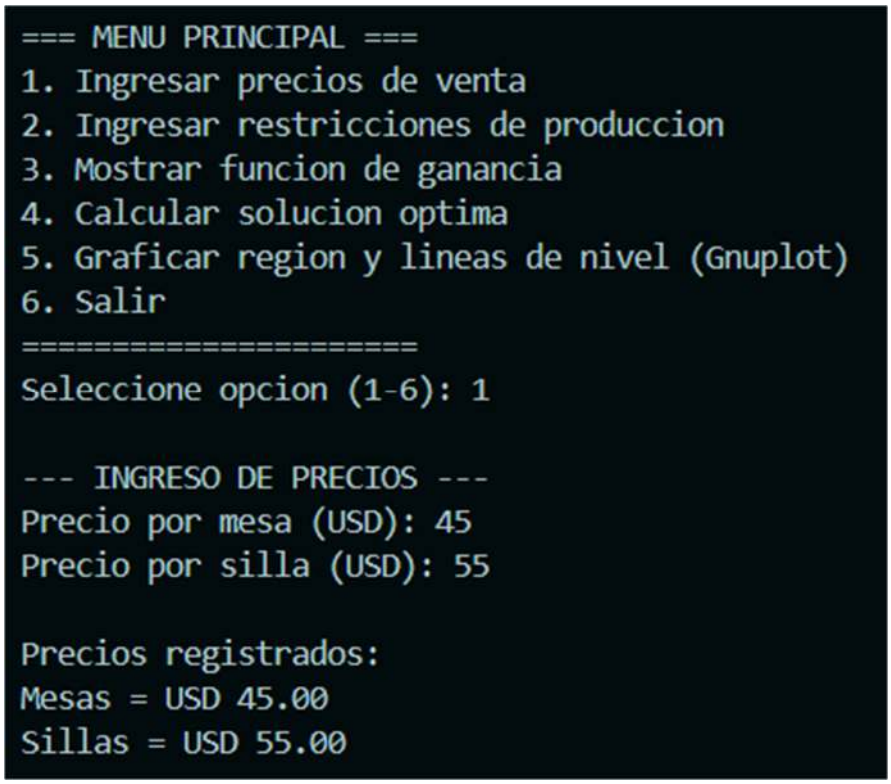


Figura 1. Salida opción 1 en donde se ingresan los precios de venta.

Opción 2: Ingreso de Restricciones de Producción

Esta sección permite el registro de múltiples restricciones lineales en el formato especificado (coeficienteX1 coeficienteX2 valorTotal). El sistema utiliza getline y sscanf para procesar cada entrada, validando exactamente tres valores numéricos no negativos, así pues, en cada restricción válida se almacena en un vector de arreglos para su posterior procesamiento; la función incluye mensajes de error descriptivos para formatos incorrectos y ofrece una confirmación visual de cada restricción agregada. Al finalizar la entrada (con el comando 'fin'), se muestra el listado completo de restricciones registradas.

El proceso comienza con una explicación clara del formato requerido, seguido de un bucle principal que procesa cada línea de entrada. Para cada restricción, el código verifica primero si el usuario ha escrito 'fin' para terminar la entrada; luego, utiliza sscanf para analizar la cadena de texto y extraer los tres valores numéricos, seguido se implementan múltiples validaciones: verificación del número de valores extraídos (debe ser exactamente 3), comprobación de que todos los valores sean no negativos, y confirmación del formato correcto. Cada restricción válida se añade al vector restricciones usando `push_back`, y se muestra inmediatamente en formato matemático para confirmación visual. Además, se implementó el uso del “try – catch” para las entradas invalidas que sean diferentes a números o en un orden ilegible para el sistema.

Salida

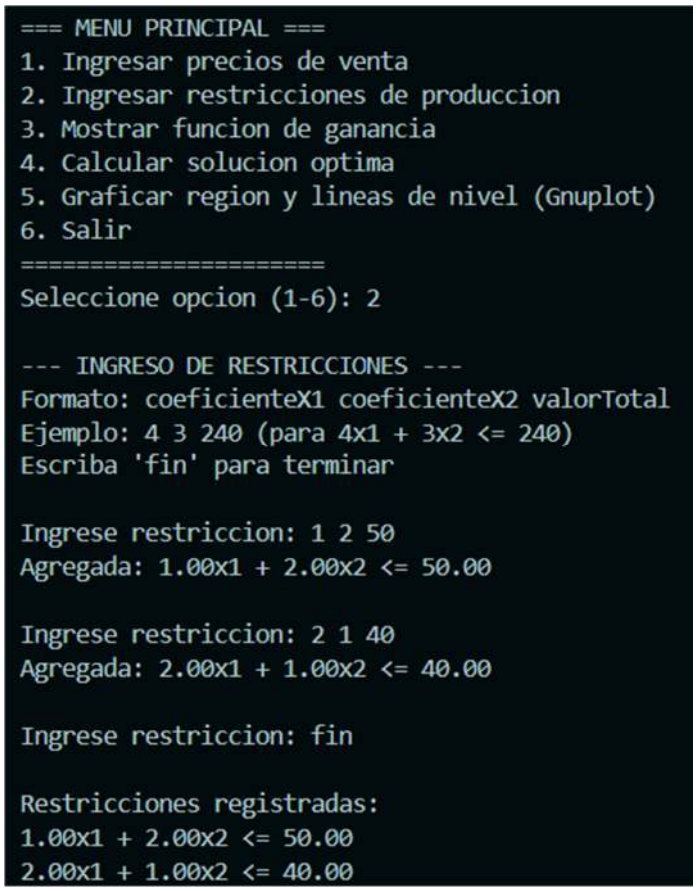


Figura 2. Salida opción 2 en donde se ingresan las restricciones de producción

Opción 3: Mostrar Función de Ganancia

Esta función genera la ecuación objetivo $Z = ax_1 + bx_2$ basada en los precios previamente registrados. Primero verifica que existan valores válidos en precioMesa y precioSilla, mostrando un mensaje de error instructivo si no se han ingresado datos. Cuando los datos son válidos, presenta la función de maximización con el formato exacto requerido (`Maximizar Z = 7x₁ + 5x₂`); así mismo la implementación es clara y concisa, reutilizando las variables globales establecidas en la Opción 1 y sirviendo como puente hacia las opciones de cálculo posterior (solución óptima).

La función comienza con una verificación de estado, asegurando que los precios necesarios hayan sido establecidos previamente. Esta validación previene errores y guía al usuario a través del flujo correcto del programa, pues cuando los precios están disponibles, se construye la cadena de texto que representa la función objetivo utilizando los valores almacenados; además el formato de salida es consistente con los estándares matemáticos y el mensaje de error incluye instrucciones claras sobre cómo corregir la situación (indicando que primero se debe ejecutar la Opción 1).

Salida

```
=== MENU PRINCIPAL ===
1. Ingresar precios de venta
2. Ingresar restricciones de produccion
3. Mostrar funcion de ganancia
4. Calcular solucion optima
5. Graficar region y lineas de nivel (Gnuplot)
6. Salir
======================
Seleccione opcion (1-6): 3

--- FUNCION DE GANANCIA ---
Maximizar Z = 45.00x1 + 55.00x2
```

Figura 3. Salida opción 3 en donde se muestra la función de ganancia.

Opción 4: Cálculo de la solución óptima

La función `opcion4()` implementa el cálculo de solución óptima para programación lineal mediante validación inicial de datos (`precio_mesa, precio_silla` y mínimo 2 restricciones), seguida de generación sistemática de puntos candidatos incluyendo el origen (0,0) y las intersecciones con ejes coordenados calculadas como `restricciones[i][2] / restricciones[i][0]` para el eje x1 y `restricciones[i][2] / restricciones[i][1]` para el eje x2.

El algoritmo principal resuelve la intersección entre las dos primeras restricciones mediante un sistema de ecuaciones lineales 2x2 utilizando la regla de Cramer, donde se extraen los coeficientes `a1, b1, c1` de la primera restricción y `a2, b2, c2` de la segunda, se calcula el determinante `det = a1*b2 - a2*b1`, y si es diferente de cero se obtienen las coordenadas `x1=(c1*b2 - c2*b1)/det` y `x2=(a1*c2 - a2*c1)/det`, agregando el punto resultante al vector de candidatos solo si cumple las condiciones de factibilidad `x1 >= 0 && x2 >= 0`.

La implementación utiliza estructuras de control como bucles `for` para iterar sobre restricciones, condicionales if anidadas para validaciones, vectores bidimensionales `vector<vector<double>>` para manejo dinámico de datos, y técnicas de depuración mediante verificación preventiva de división por cero y validación de entrada.

x1	x2	Fuente
0	0	Origen
x	0	Intersección de la restricción 1 con el eje x
0	y	Intersección de la restricción 2 con el eje y
0	50	Intersección entre restricción 1 y restricción 2

Tabla 1: Puntos candidatos para obtener la solución óptima.

Estos puntos provienen del origen, de las intersecciones de cada restricción con los ejes coordenados, y del cruce entre las dos primeras restricciones. Solo se incluyen aquellos puntos que cumplen todas las restricciones del problema.

Tras la evaluación de puntos candidatos, el código evaluara cada punto almacenándolos en el vector puntos para así poder determinar una solución óptima, las variables *mejorZ* y *mejorPunto* que evaluarán la ganancia máxima, después repite esto sobre todos los puntos candidatos generados previamente, incluidos los de la *Tabla 1* como es (0,0), intersecciones con ejes y puntos de cruce entre condiciones, para cada punto se valida contra las restricciones teniendo una tolerancia de 0.001 para evitar errores de redondeo en números flotantes, los puntos que pasan esta validación son evaluados en la función ganancia, mediante este proceso se mantiene un registro dinámico entre el punto y la máxima ganancia encontrada, el cual seguirá actualizando la solución óptima en tiempo real.

Durante el proceso la función imprime en consola los posibles puntos factible y su correspondiente ganancia, al mismo tiempo, se actualiza de manera dinámica la solucion óptima si se detecta una que genere una mayor ganancia, además si el código detecta al menos un punto válido, presenta la combinación óptima entre mesas y sillas, caso contrario muestra la ausencia de una solución factible.

Salida

```
=== MENU PRINCIPAL ===
1. Ingresar precios de venta
2. Ingresar restricciones de produccion
3. Mostrar funcion de ganancia
4. Calcular solucion optima
5. Graficar region y lineas de nivel (Gnuplot)
6. Salir
======================
Seleccione opcion (1-6): 4

--- CALCULO DE LA SOLUCION OPTIMA EN INTERSECCIONES ---

Evaluando intersecciones:
(10.00, 20.00) -> Z = 1550.00

=== SOLUCION OPTIMA EN INTERSECCION ===
Mesas (x1) = 10.00
Sillas (x2) = 20.00
Ganancia maxima = 1550.00
```

Figura 4. Salida opción 4 en donde se muestra la solución óptima del problema.

Opción 5: Gráfica en Gnuplot

La función graficarSolucion() está diseñada para resolver gráficamente un problema de programación lineal mediante el uso de Gnuplot. Comienza verificando que el usuario haya ingresado los precios de los productos (mesa y silla) y al menos dos restricciones. Luego, calcula puntos relevantes que podrían formar parte de la región factible, incluyendo intersecciones con los ejes y entre las primeras dos restricciones, utilizando determinantes para encontrar el punto de cruce. Posteriormente, filtra estos puntos, verificando que cumplan todas las restricciones impuestas, y evalúa cada uno con la función objetivo para encontrar el punto que maximiza el beneficio (Z óptimo).

Una vez identificado el punto óptimo, se genera un archivo (region.dat) con los puntos válidos que conforman la región factible, cerrando el polígono para que pueda ser graficado

correctamente. Luego se calculan tres líneas de nivel (una óptima y dos paralelas) para mostrar visualmente el avance de la función objetivo. Con ayuda de Gnuplot, se genera una gráfica que incluye: la región factible coloreada, las restricciones como líneas, las líneas de nivel que representan diferentes valores de Z, y el punto óptimo marcado visualmente.

Salida

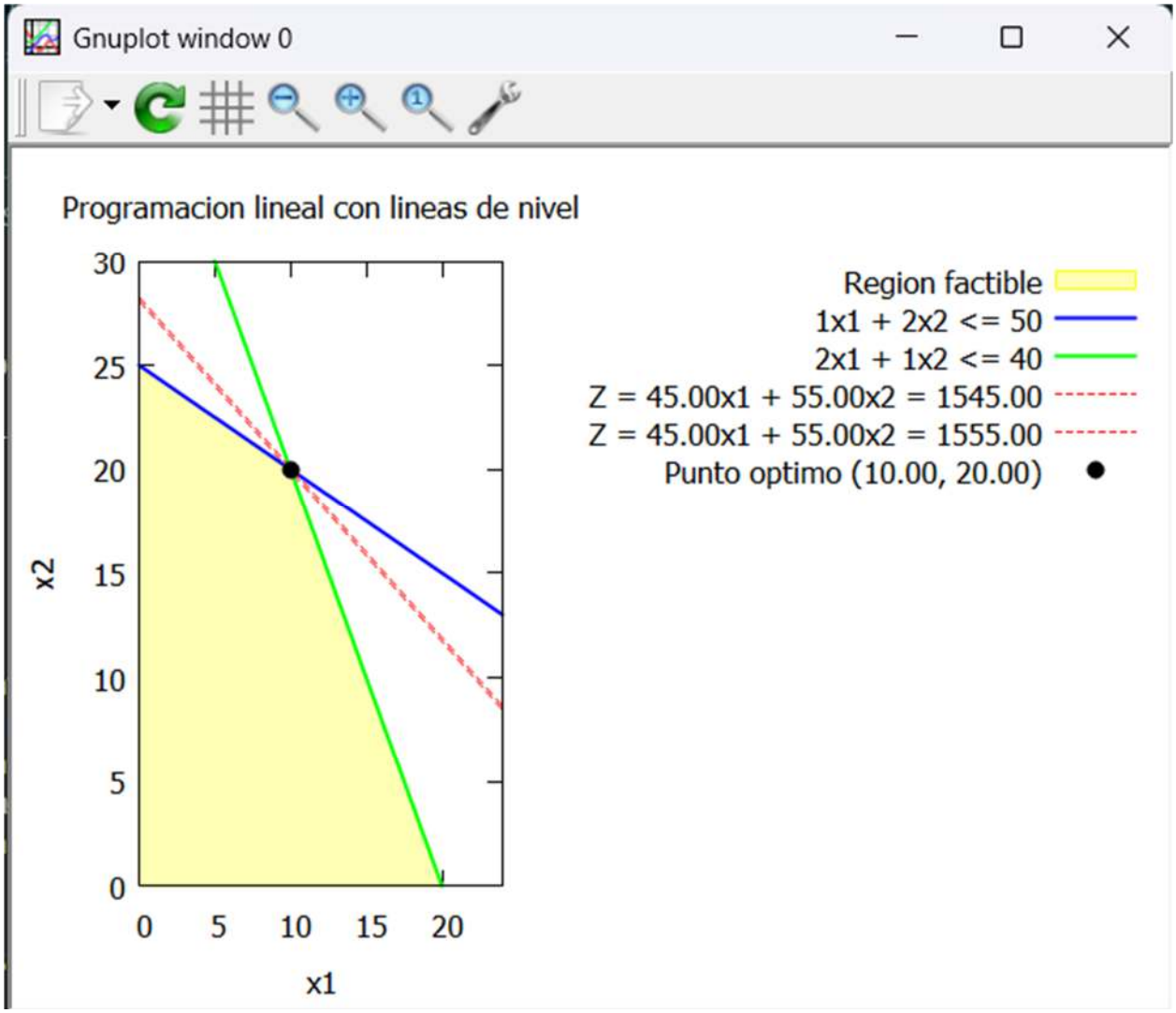


Figura 5. Salida opción 5 en donde se muestra la gráfica del problema, las ecuaciones tomadas en cuenta y el punto de la solución óptima.

FUNCIONAMIENTO DE LA APLICACIÓN GRÁFICA

Gnuplot es una herramienta gráfica multiplataforma que se ejecuta desde la línea de comandos y está disponible para sistemas como Linux, Windows, macOS, OS/2, VMS, entre otros. Fue desarrollada inicialmente con el propósito de ayudar a científicos y estudiantes a representar visualmente datos y funciones matemáticas de forma interactiva. Con el tiempo, ha ampliado sus capacidades para incluir usos no interactivos, como la generación de gráficos mediante scripts en la web. Además, es empleada como motor gráfico en programas externos como Octave (Gnuplot, s.f.).

Funcionamiento de Gnuplot

Gnuplot es una herramienta que funciona mediante comandos escritos en una terminal. Puedes usarla de forma interactiva o mediante scripts automáticos. Esta flexibilidad permite generar gráficos rápida y eficientemente, sin necesidad de una interfaz gráfica.

Relación entre Gnuplot y C++

Los programas de C++ pueden generar archivos .dat con datos numéricos y luego se puede usar Gnuplot para leer esos archivos y generar gráficos. Para ejecutar comandos de Gnuplot directamente desde el código de C++ se puede usar popen(), el cual abre un canal entre el programa y Gnuplot con el objetivo de enviar comandos uno por uno a Gnuplot desde el programa.

Para poder usar Gnuplot desde C++ con system() o popen(), se necesita que el ejecutable de gnuplot esté disponible en las variables de entorno del sistema. Para ello hay que agregar al PATH la ruta: C:\Program Files\gnuplot\bin, y después comprobar en consola que ha sido configurado correctamente.

Tipos de Gráficos

Gnuplot permite crear una amplia variedad de gráficos, como gráficos 2D y 3D, superficies, histogramas, mapas de calor, diagramas de caja y funciones paramétricas. También puede generar gráficos geográficos o con datos temporales.

Salidas y Exportación

Los gráficos pueden mostrarse directamente en pantalla o exportarse a formatos como PNG, PDF, SVG, EPS, GIF animado o incluso código para LaTeX/TikZ. Esto es útil para integrarlos en documentos o páginas web.

Entrada de Datos

Admite datos desde archivos de texto, archivos binarios e incluso desde otros programas a través de pipes. Los datos pueden tener varias columnas, y se pueden seleccionar, procesar y transformar antes de graficarlos.

Integración con Otros Lenguajes

Gnuplot puede ser utilizado desde lenguajes como Python, Julia, Perl, Java, Fortran y muchos más. Además, se integra como motor gráfico en programas como GNU Octave, Maxima o gretl.

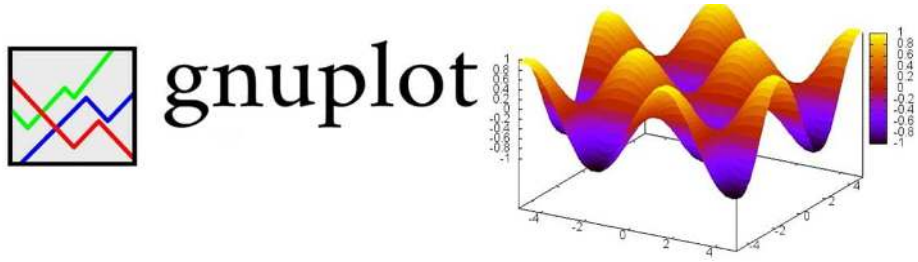
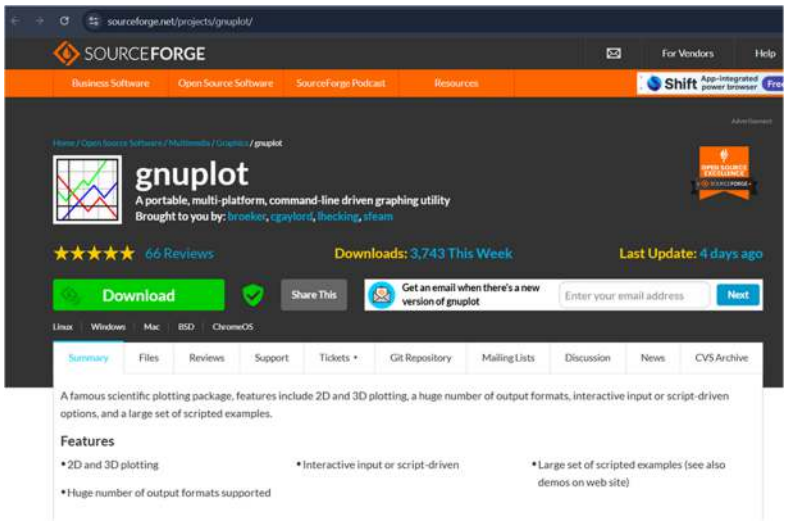


Figura 6. Logotipo de la librería grafica utilizada

INSTALACIÓN DE LA APLICACIÓN

1. Abrir el siguiente enlace en una pestaña del navegador https://sourceforge.net/projects/gnuplot/
2. Una vez abierto el enlace, presionar el botón que dice download.

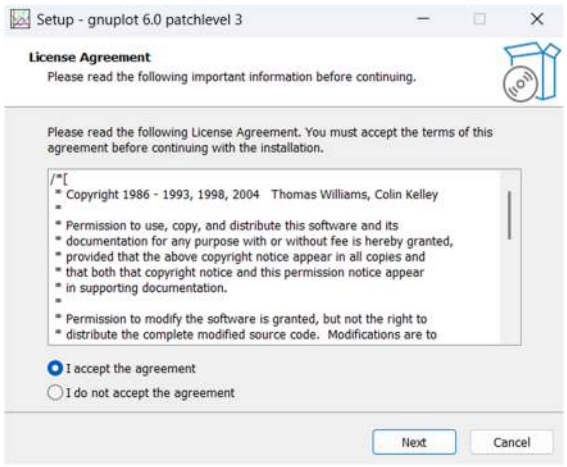


Nota 1. *Ventana inicial de instalación de Gnuplot solicitando selección de idioma. Nota. Captura de pantalla realizada durante la instalación de Gnuplot.*

3. Después de que la aplicación se ha terminado de instalar, se debe aceptar el mensaje que aparece que dice que se permite los cambios en el dispositivo.
4. Cuando se acepta el mensaje aparece la siguiente ventana, en la que se debe colocar el idioma inglés.

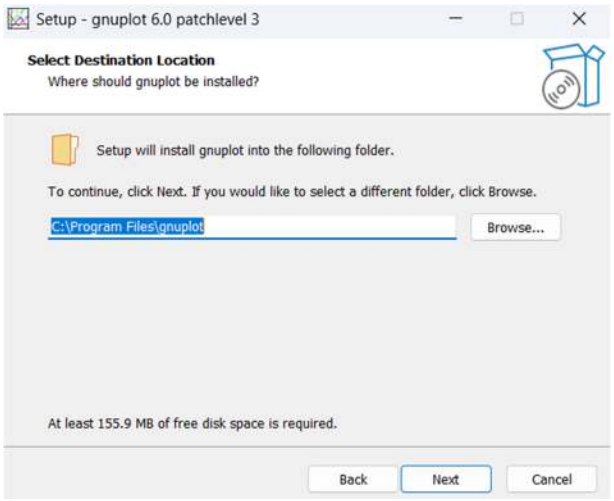
Nota 2. *Ventana de instalación donde se selecciona el idioma.*

5. Se debe señalar la opción de “aceptó el acuerdo”.



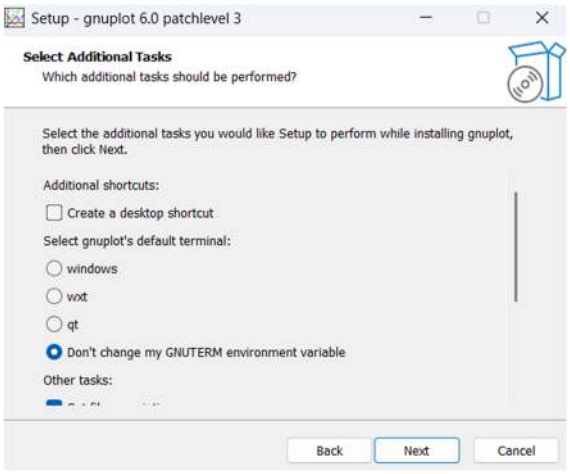
Nota 3. *Pantalla de aceptación del acuerdo de licencia.*

6. Elegir en que carpeta se quiere guardar la aplicación.



Nota 4. *Selección de carpeta de instalación para Gnuplot.*

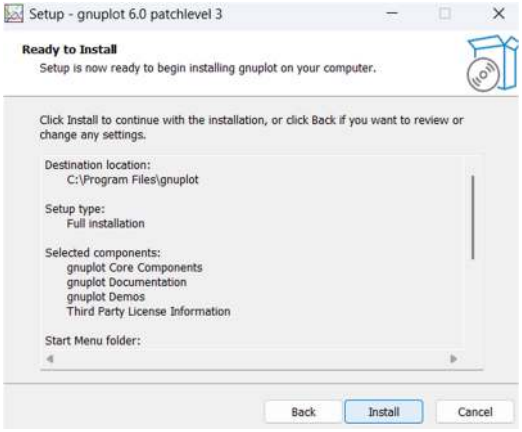
7. Posteriormente aparecerá una ventana para seleccionar tareas adicionales. Donde se debe marcar la opción “Don’t change my GNUTERM enviroment variable”.



Nota 5. *Ventana emergente con tareas adicionales de instalación.*

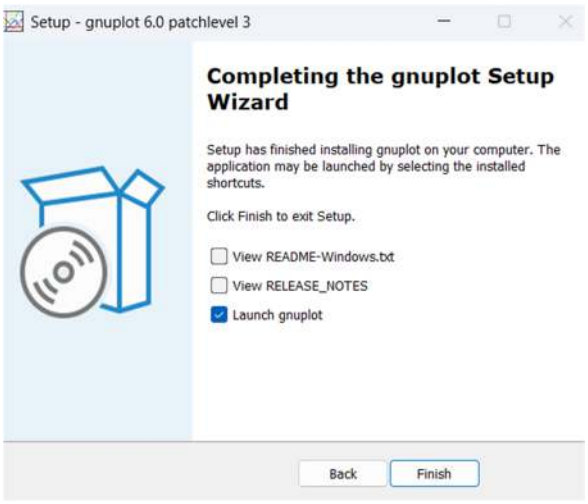
8. Aparecerán más ventanas en las cuales se debe presionar “Next”.

9. Finalmente, se debe presionar el botón Instalar.



Nota 6. *Pantalla de confirmación antes de iniciar la instalación.*

10. Se visualizará una ventana donde se debe seleccionar “Launch gnuplot” y luego finalizar.



Nota 7. *Ventana con la opción "Launch gnuplot" al finalizar la instalación.*

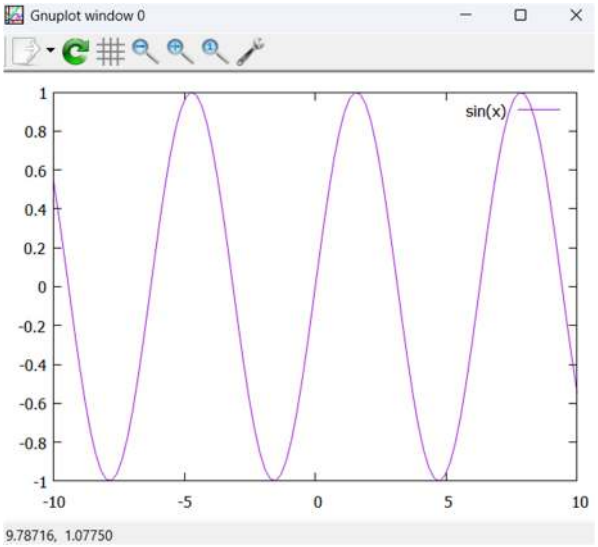
11. Para comprobar su funcionamiento se debe abrir la aplicación en la computadora.

Notaa 8. *Acceso a la aplicación Gnuplot 6.0 patchlevel 3 desde el menú de inicio de Windows.*

12. Se desplegará la terminal de la app donde se puede colocar un ejemplo, en este caso el sin(x) para comprobar el funcionamiento de la aplicación. Esto debe escribirse seguido de gnuplot>, colocando plot sin(x).

Nota 9. *Terminal de Gnuplot con el comando plot sin(x) ingresado.*

13. Al dar enter al comando, aparecerá una ventana que muestra la gráfica de la función que se colocó anteriormente. Comprobando el funcionamiento de la app.



Nota 10. *Gráfica de la función seno generada en Gnuplot.*

JUSTIFICACIÓN

Hemos decidio usar Gnuplot por una variedad de razones que listaremos a continuación:

- **Integración Sencilla con C/C++**
  – Gnuplot funciona vía tuberías (popen/_popen), de modo que desde el propio código se pueden generar datos y comandos en tiempo real, sin depender de interfaces gráficas complejas.
- **Ligereza y Portabilidad**
  – Es una herramienta de línea de comandos muy ligera y con pocas dependencias.
  – Corre en Windows, macOS y Linux sin apenas modificaciones en el código.
- **Rica Capacidad de Personalización**
  – Permite controlar títulos, etiquetas de ejes, rangos, estilos de línea, colores y transparencias, fundamentales para dibujar regiones factibles y líneas de nivel con claridad.
  – Soporta “filled curves” para sombrear áreas factibles, trazado de puntos y líneas de contorno, todo con unas pocas líneas de script.
- **Reproducibilidad y Mantenimiento**
  – Al separar la lógica de cálculo (C++) de la de visualización (Gnuplot), el código es más modular y fácil de mantener.

– Cada gráfico se genera de forma determinista usando el mismo archivo de datos y comandos, ideal para presentar resultados consistentes.

- **Comunicación de Resultados**
  – Las gráficas de Gnuplot son de calidad "publicación": claras, ajustables en resolución y exportables en formatos como PNG, SVG o PDF, útiles para informes o presentaciones académicas.
  – Facilita el entendimiento visual de la región factible, la evolución de la función objetivo y la localización del punto óptimo.

CASOS DE PRUEBA

Caso de Prueba 1 (Exitoso):

a) Opción 1: Precios precioMesa=25, precioSilla=35.

```
=== MENU PRINCIPAL ===
1. Ingresar precios de venta
2. Ingresar restricciones de produccion
3. Mostrar funcion de ganancia
4. Calcular solucion optima
5. Graficar region y lineas de nivel (Gnuplot)
6. Salir
======================
Seleccione opcion (1-6): 1

--- INGRESO DE PRECIOS ---
Precio por mesa (USD): 25
Precio por silla (USD): 35

Precios registrados:
Mesas = USD 25.00
Sillas = USD 35.00
```

b) Opción 2: Restricciones $4x1+2x2 \leq 100$, $1x1+5x2 \leq 80$.

```
--- INGRESO DE RESTRICCIONES ---
Formato: coeficienteX1 coeficienteX2 valorTotal
Ejemplo: 4 3 240 (para 4x1 + 3x2 <= 240)
Escriba 'fin' para terminar

Ingrese restriccion: 4 2 100
Agregada: 4x1 + 2x2 <= 100

Ingrese restriccion: 1 5 80
Agregada: 1x1 + 5x2 <= 80

Ingrese restriccion: fin

Restricciones registradas:
4x1 + 2x2 <= 100
1x1 + 5x2 <= 80
```

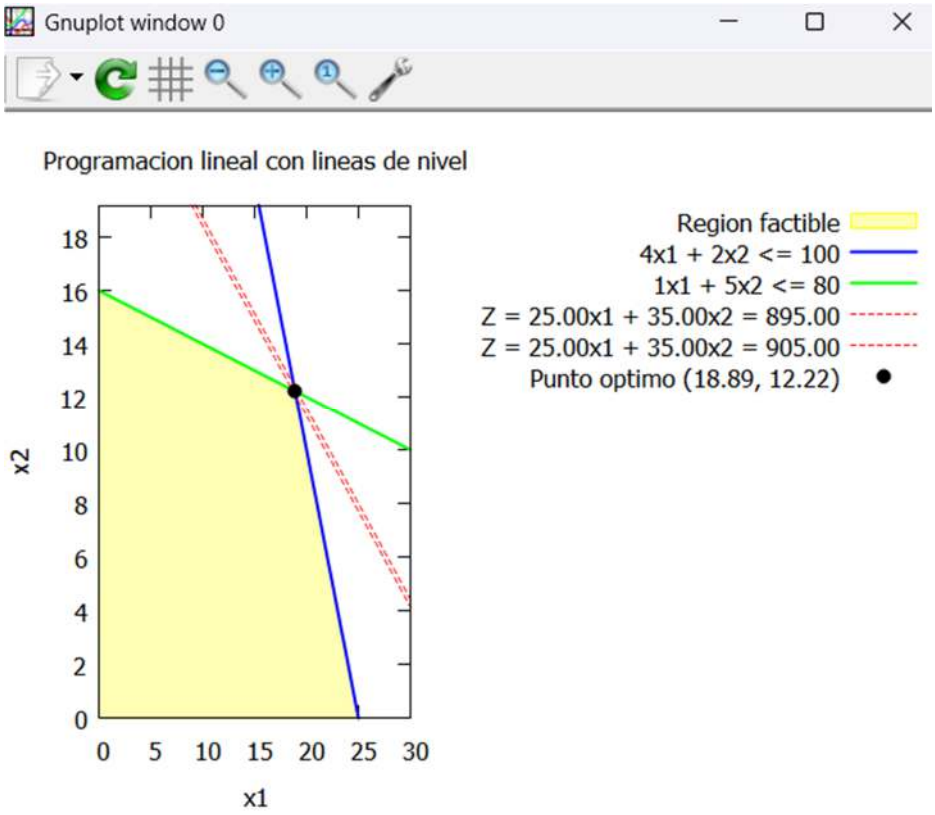
c) Opción 3: Función Z=25x1+35x2.

```
--- FUNCION DE GANANCIA ---
Maximizar Z = 25.00x1 + 35.00x2
```

d) Opción 4: Calcular solución óptima con datos previos (x1=18.89, x2=12.22, Z=900).

```
=== SOLUCION OPTIMA EN INTERSECCION ===
Mesas (x1) = 18.89
Sillas (x2) = 12.22
Ganancia maxima = 900.00
```

e) Opción 5: Mostrar gráfico con las restricciones, el área factible y el punto de solución óptima (18.89, 12.22).



Consideramos este caso como relevante por lo siguiente:
El mejor punto (18,12) no se encuentra sobre ningún eje ni en un intercepto sencillo, sino en la intersección de ambas restricciones. Garantiza que el programa:

- Calcule correctamente esa intersección vía Cramer.
- Evalúe la función objetivo en todos los vértices factibles, no sólo en extremos triviales.

Caso de Prueba 2 (Exitoso):

a) Opción 1: Precios precioMesa=60, precioSilla=20.

```
=== MENU PRINCIPAL ===
1. Ingresar precios de venta
2. Ingresar restricciones de produccion
3. Mostrar funcion de ganancia
4. Calcular solucion optima
5. Graficar region y lineas de nivel (Gnuplot)
6. Salir
======================
Seleccione opcion (1-6): 1

--- INGRESO DE PRECIOS ---
Precio por mesa (USD): 60
Precio por silla (USD): 20

Precios registrados:
Mesas = USD 60.00
Sillas = USD 20.00
```

b) Opción 2: Restricciones $3x1+4x2 \leq 120$, $5x1+1x2 \leq 150$.

```
--- INGRESO DE RESTRICCIONES ---
Formato: coeficienteX1 coeficienteX2 valorTotal
Ejemplo: 4 3 240 (para 4x1 + 3x2 <= 240)
Escriba 'fin' para terminar

Ingrese restriccion: 3 4 120
Agregada: 3.00x1 + 4.00x2 <= 120.00

Ingrese restriccion: 5 1 150
Agregada: 5.00x1 + 1.00x2 <= 150.00

Ingrese restriccion: fin

Restricciones registradas:
3.00x1 + 4.00x2 <= 120.00
5.00x1 + 1.00x2 <= 150.00
```

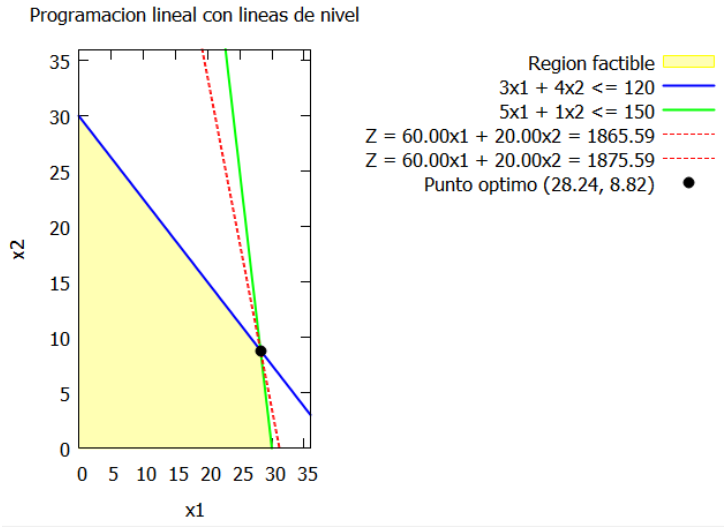
c) Opción 3: Función $Z=60x1+20x2$.

```
--- FUNCION DE GANANCIA ---
Maximizar Z = 60.00x1 + 20.00x2
```

d) Opción 4: Calcular solución óptima con datos previos ($x1=28.24$, $x2=8.82$, $Z=1870.59$).

```
--- CALCULO DE LA SOLUCION OPTIMA EN INTERSECCIONES

Evaluando intersecciones:
(28.24, 8.82) -> Z = 1870.59

=== SOLUCION OPTIMA EN INTERSECCION ===
Mesas (x1) = 28.24
Sillas (x2) = 8.82
Ganancia maxima = 1870.59
```

e) Opción 5: Mostrar gráfico con las restricciones, el área factible y el punto de solución óptima (28.24, 8.82).



Consideramos este caso como relevante por lo siguiente:

- La línea punteada roja, un nivel de Z sólo un peldaño arriba, ya no toca la región amarilla. Muestra perfectamente cómo, si intentáramos mejorar la ganancia, no habría ninguna solución factible. Es un excelente indicativo visual de que se ha encontrado el máximo global.
- Vemos que la línea de nivel “corta” entre ambas restricciones. Eso explica por qué el óptimo cae justo en la intersección: su pendiente está entre las de las dos fronteras.

Caso de Prueba 3 (Fallido):

a) Opción 1: Precios precioMesa=10, precioSilla=5.

```
=== MENU PRINCIPAL ===
1. Ingresar precios de venta
2. Ingresar restricciones de produccion
3. Mostrar funcion de ganancia
4. Calcular solucion optima
5. Graficar region y lineas de nivel (Gnuplot)
6. Salir
======================
Seleccione opcion (1-6): 1

--- INGRESO DE PRECIOS ---
Precio por mesa (USD): 10
Precio por silla (USD): 5

Precios registrados:
Mesas = USD 10.00
Sillas = USD 5.00
```

b) Opción 2: Restricciones 1x1+1x2≤10, 2x1+2x2≤20.

```
--- INGRESO DE RESTRICCIONES ---
Formato: coeficienteX1 coeficienteX2 valorTotal
Ejemplo: 4 3 240 (para 4x1 + 3x2 <= 240)
Escriba 'fin' para terminar

Ingrese restriccion: 1
Error: Formato incorrecto. Use 3 numeros separac

Ingrese restriccion: 1 1 10
Agregada: 1.00x1 + 1.00x2 <= 10.00

Ingrese restriccion: 2 2 20
Agregada: 2.00x1 + 2.00x2 <= 20.00

Ingrese restriccion: fin

Restricciones registradas:
1.00x1 + 1.00x2 <= 10.00
2.00x1 + 2.00x2 <= 20.00
```

c) Opción 3: Función Z=10x1+5x2.

```
--- FUNCION DE GANANCIA ---
Maximizar Z = 10.00x1 + 5.00x2
```

d) Opción 4: Calcular solución óptima con datos previos (**No se encontraron intersecciones factibles** (determinante = 0).

```
--- CALCULO DE LA SOLUCION OPTIMA EN INTERSECCIONES ---
No se encontraron intersecciones factibles entre restricciones.
```

Caso de Prueba 4 (Fallido):

a) Opción 1: Precios precioMesa=20, precioSilla=15.

```
=== MENU PRINCIPAL ===
1. Ingresar precios de venta
2. Ingresar restricciones de produccion
3. Mostrar funcion de ganancia
4. Calcular solucion optima
5. Graficar region y lineas de nivel (Gnuplot)
6. Salir
======================
Seleccione opcion (1-6): 1

--- INGRESO DE PRECIOS ---
Precio por mesa (USD): 20
Precio por silla (USD): 15

Precios registrados:
Mesas = USD 20.00
Sillas = USD 15.00
```

b) Opción 2: Restricciones 1x1+2x2≤8, 0.5x1+1x2≤4.

```
--- INGRESO DE RESTRICCIONES ---
Formato: coeficienteX1 coeficienteX2 valorTotal
Ejemplo: 4 3 240 (para 4x1 + 3x2 <= 240)
Escriba 'fin' para terminar

Ingrese restriccion: 1 2 8
Agregada: 1.00x1 + 2.00x2 <= 8.00

Ingrese restriccion: 0.5 1 4
Agregada: 0.50x1 + 1.00x2 <= 4.00

Ingrese restriccion: fin

Restricciones registradas:
1.00x1 + 2.00x2 <= 8.00
0.50x1 + 1.00x2 <= 4.00
```

c) Opción 3: Función Z=20x1+15x2.

```
--- FUNCION DE GANANCIA ---
Maximizar Z = 20.00x1 + 15.00x2
```

d) Opción 4: Calcular solución óptima con datos previos (**No se encontraron intersecciones factibles** (pendientes iguales).

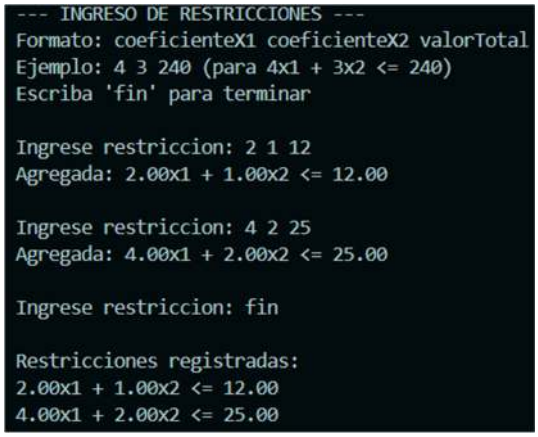
```
--- CALCULO DE LA SOLUCION OPTIMA EN INTERSECCIONES ---
No se encontraron intersecciones factibles entre restricciones.
```

Caso de Prueba 5 (Fallido):

a) Opción 1: Precios precioMesa=30, precioSilla=10.

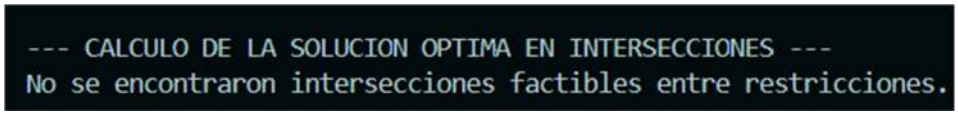
```
=== MENU PRINCIPAL ===
1. Ingresar precios de venta
2. Ingresar restricciones de produccion
3. Mostrar funcion de ganancia
4. Calcular solucion optima
5. Graficar region y lineas de nivel (Gnuplot)
6. Salir
======================
Seleccione opcion (1-6): 1

--- INGRESO DE PRECIOS ---
Precio por mesa (USD): 30
Precio por silla (USD): 10

Precios registrados:
Mesas = USD 30.00
Sillas = USD 10.00
```

b) Opción 2: Restricciones 2x1+1x2≤12, 4x1+2x2≤25.



c) Opción 3: Función Z=30x1+10x2.

d) Opción 4: Calcular solución óptima con datos previos (**No se encontraron intersecciones factibles** (rectas paralelas).



La razón del porque fallan estos casos de prueba es porque el método de Cramer detecta determinante cero y el programa correctamente informa “No se encontraron intersecciones factibles” antes de intentar graficar o calcular un óptimo.

Repositorio en Línea con todo el Proyecto: https://github.com/ArlRonald995/Proyecto-Segundo-Bimestre-Programaci-n1.git

Nota: Los comandos realizados para realizar los comits de las diferentes versiones se encuentran en un archivo llamado “Comandos_de_Git.txt”, de esta forma se puede evidenciar el correcto uso de esta herramienta muy útil para este tipo de proyectos y a su vez observar la evolución del mismo conforme se implementaban más mejoras.

BIBLIOGRAFÍA

- *Gnuplot*. (n.d.). https://webs.um.es/mira/maxima/gnuplot.php
- Hillier, F. S., & Lieberman, G. J. (2021). *Introduction to Operations Research* (11th ed.). McGraw-Hill Education. https://info.mheducation.com/rs/128-SJW-347/images/Preface_Hillier_Intro_Operations_Research_11e.pdf
- *Ayuda:Gnuplot - Wikipedia, la enciclopedia libre*. (s.f.). Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Ayuda:Gnuplot#:~:text=Gnuplot%20es%20un%20software%20de,se%20representa%20por%20la%20y.)
- (s.f.). Bienvenido a FAMAF - Facultad de Astronomía, Matemática, Física y Computación. https://www.famaf.unc.edu.ar/~serra/man_gnuplot.pdf
- (s.f.). Servidor de software libre de la Universidad de Zaragoza. https://softlibre.unizar.es/manuales/aplicaciones/gnuplot/manual-gnuplot.pdf
- *Gnuplot*. (s.f.). Leibniz-Rechenzentrum (LRZ). http://doku.lrz.de/gnuplot-10746446.html