

**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**

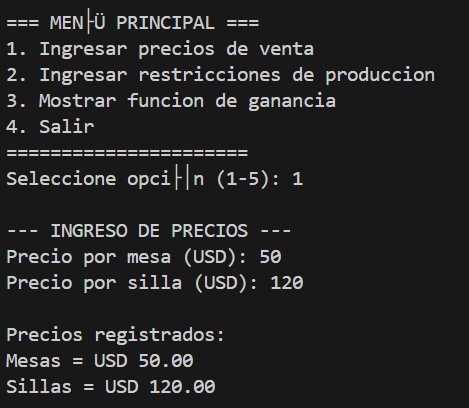
Opción 1: Ingreso de Precios de Venta

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 (precio\_mesa, precio\_silla) 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 (precio\_mesa > 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.

*Salida*



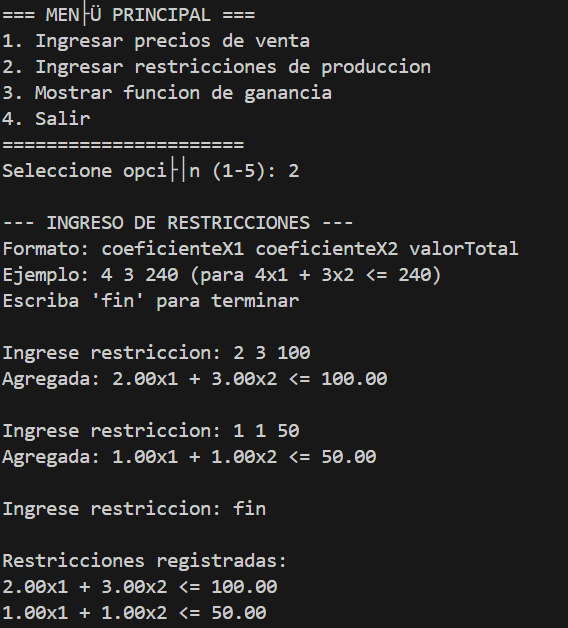
*Figura 1. Salida opción 1 en donde se ingresan los precios de venta.*

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.

*Salida*



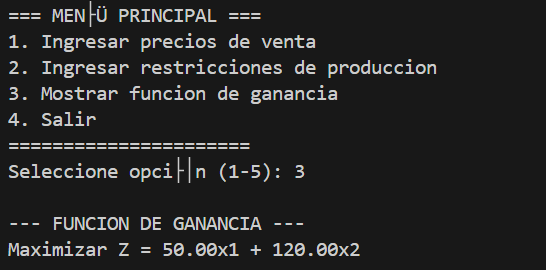
*Figura 2. Salida opción 2 en donde se ingresan las restricciones de producción.*

Opción 3: Mostrar Función de Ganancia

Esta función genera la ecuación objetivo Z = ax₁ + bx₂ basada en los precios previamente registrados. Primero verifica que existan valores válidos en precio\_mesa y precio\_silla, 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*



*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 *mejor\_z* y *mejor\_punto* 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 numeros 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.

Opción 5: Gráfica en Gnuplot

La función opcion5() 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 *Z=precio\_mesa⋅ x1+precio\_silla⋅ x2* 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.

**FUNCIONAMIENTO DE LA APLICACIÓN**

**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.

### **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**

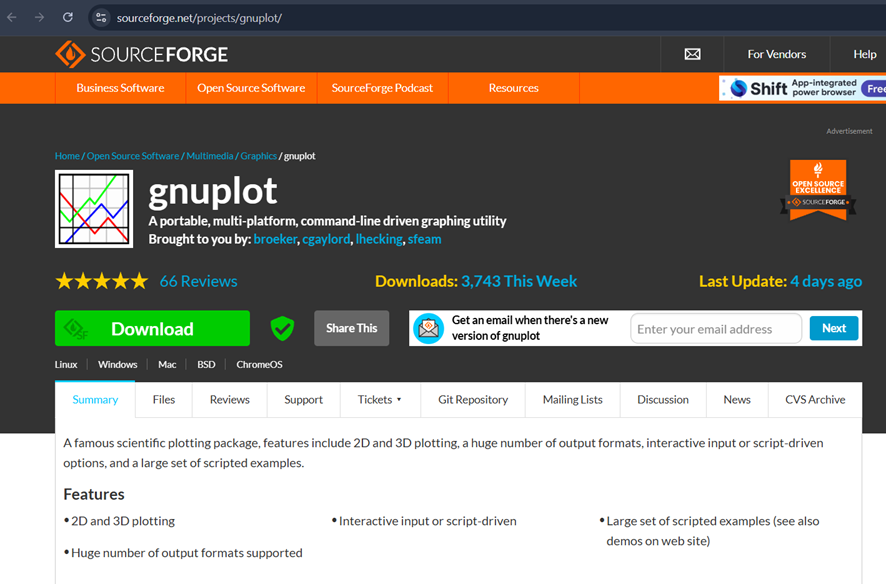
Admite datos desde archivos de texto, archivos binarios e incluso desde otros programas a través de pipes. Los datos pueden tener varias columnas, y se pueden seleccionar, procesar y transformar antes de graficarlos.

### **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.

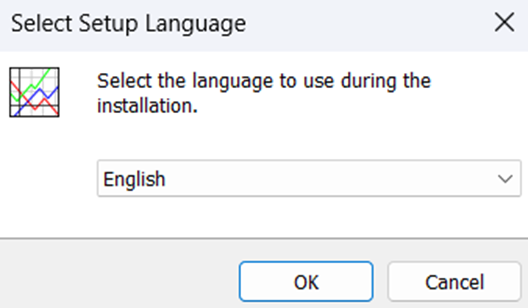
**INSTALACIÓN DE LA APLICACIÓN**

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



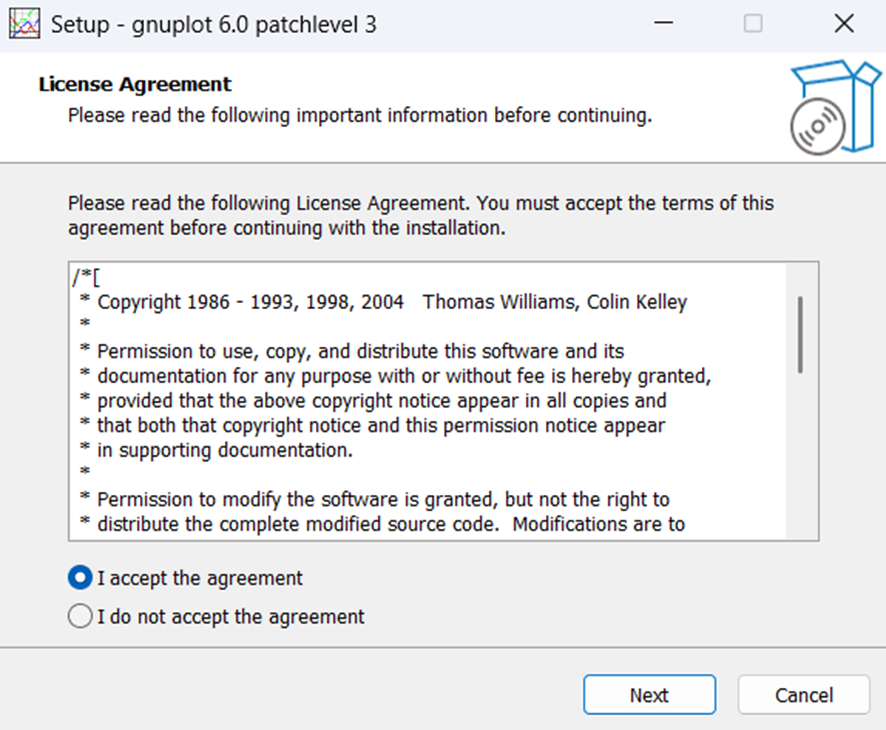
**Figura 1*.*** *Ventana inicial de instalación de Gnuplot solicitando selección de idioma. Nota. Captura de pantalla realizada durante la instalación de Gnuplot.*

1. 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.
2. Cuando se acepta el mensaje aparece la siguiente ventana, en la que se debe colocar el idioma inglés.



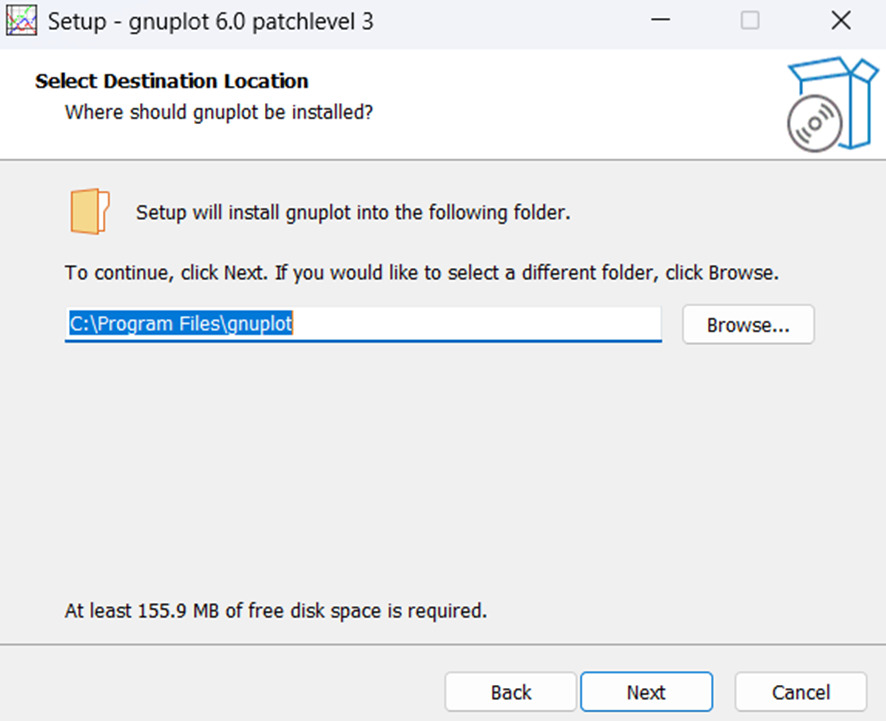
**Figura 2.** *Ventana de instalación donde se selecciona el idioma.*

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



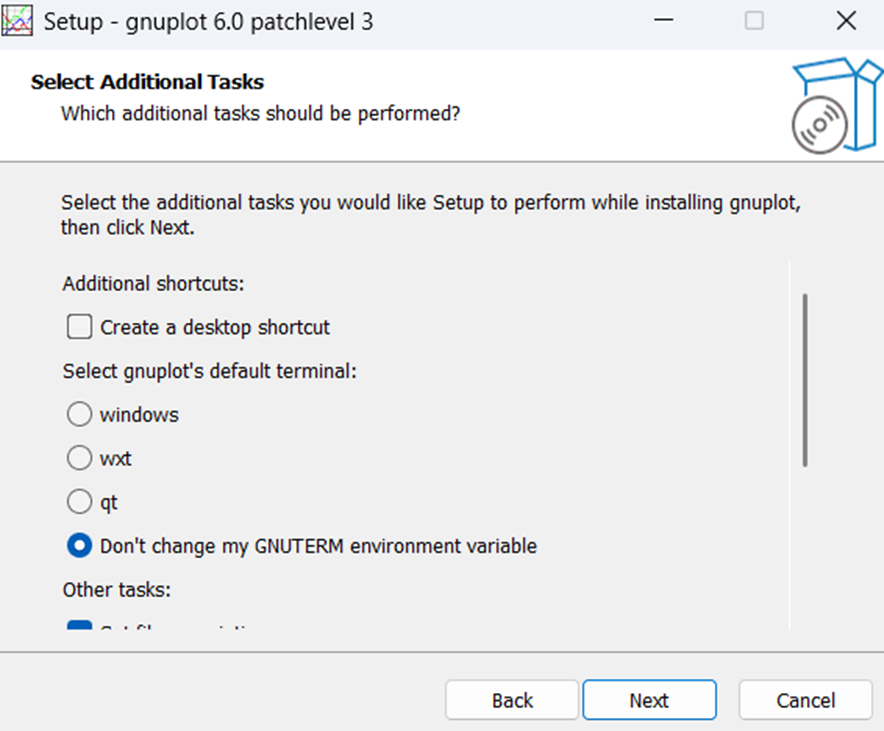
**Figura 3.** *Pantalla de aceptación del acuerdo de licencia.*

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



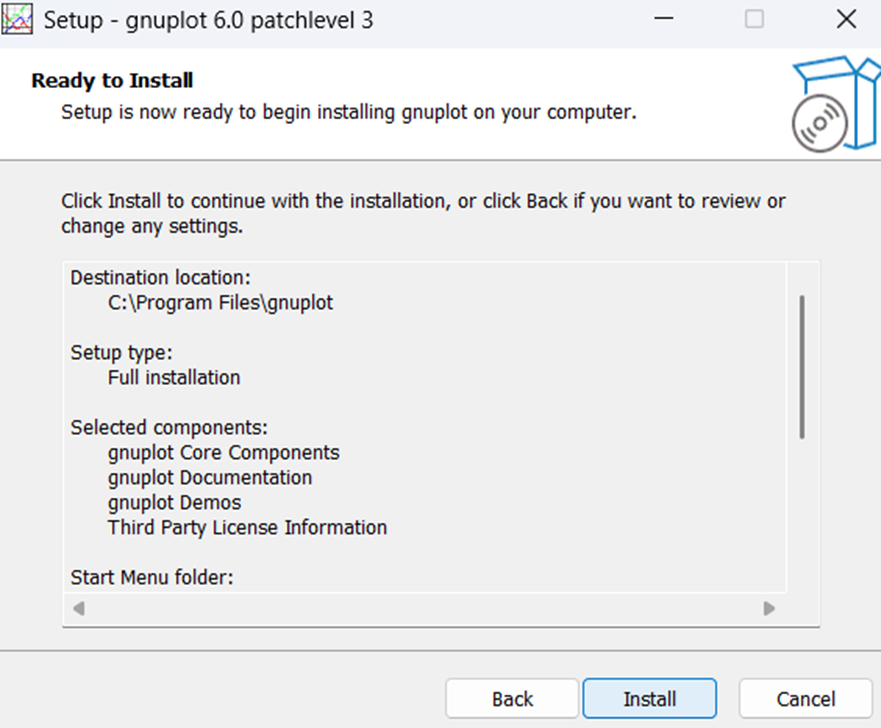
**Figura 4.** *Selección de carpeta de instalación para Gnuplot.*

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



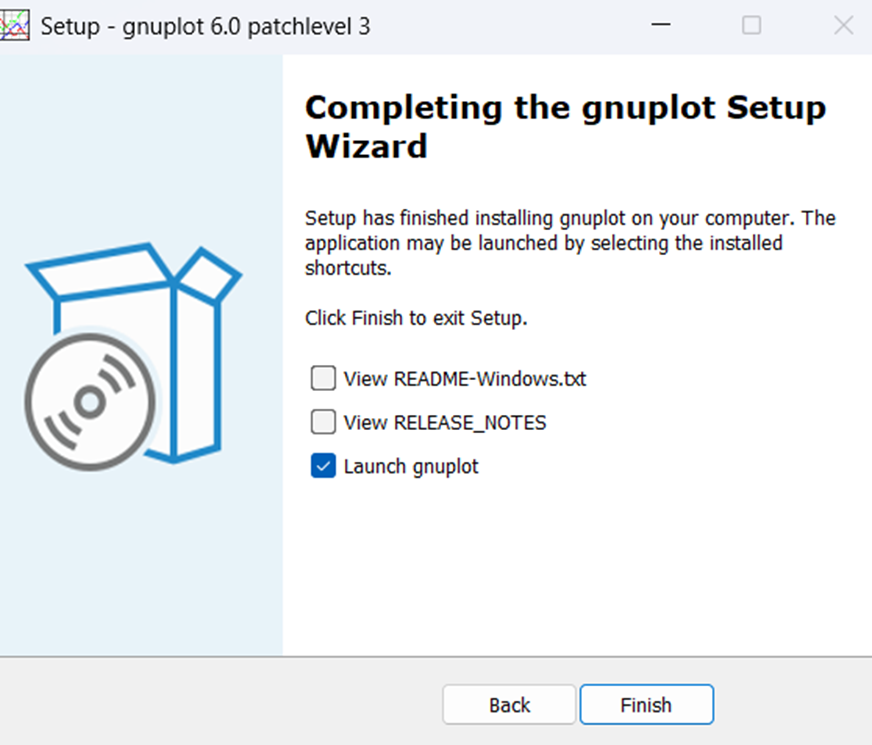
**Figura 5.** *Ventana emergente con tareas adicionales de instalación.*

1. Aparecerán más ventanas en las cuales se debe presionar “Next”.
2. Finalmente, se debe presionar el botón Instalar.



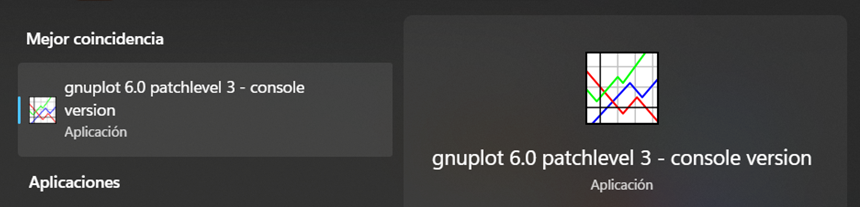
**Figura 6.**  *Pantalla de confirmación antes de iniciar la instalación.*

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



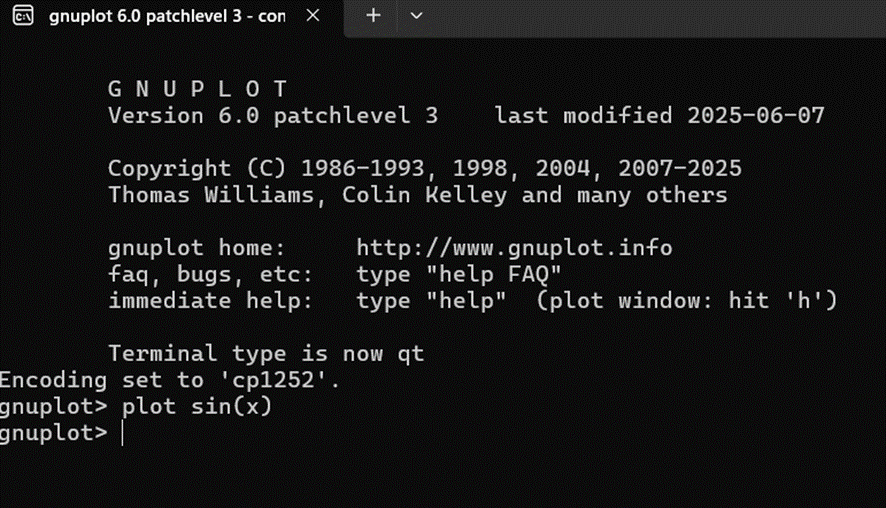
**Figura 7.** *Ventana con la opción “Launch gnuplot” al finalizar la instalación.*

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



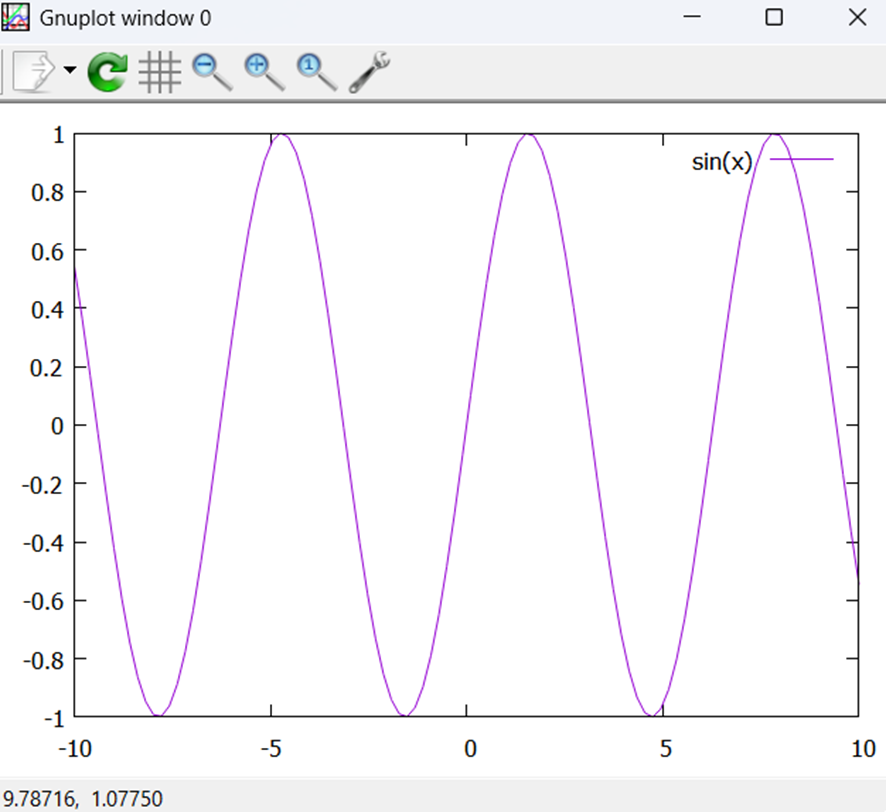
**Figura 8.** *Acceso a la aplicación Gnuplot 6.0 patchlevel 3 desde el menú de inicio de Windows.*

1. 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).



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

1. 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.



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

**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>