

INGENIERÍA MECATRÓNICA



DI\_CERO

DIEGO CERVANTES RODRÍGUEZ

INSTRUMENTACIÓN VIRTUAL

PYTHON 3.9.7, C# & LABVIEW

Graficación Matricial Múltiple

## Contenido

Instrucciones – <b>Graficación Matricial Múltiple:</b> .....	2
Código Python – Visual Studio Code (Logo Azul): .....	4
Resultado del Código Python .....	7
Código C# (.Net Framework) – Visual Studio (Logo Morado):.....	7
Código con resultado en GUI de Visual Studio .Net Framework: .....	7
Código C# de la interfaz gráfica (GUI) [Design .cs]:.....	9
Resultado de la GUI creada con Código C# en Visual Studio .....	10
Diagrama LabVIEW: .....	11

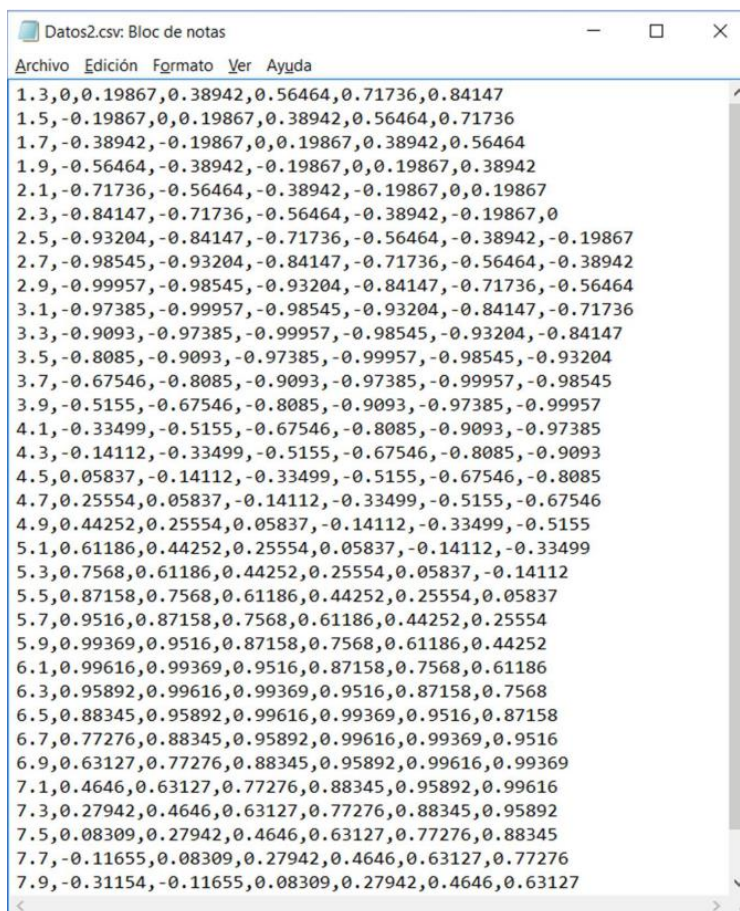


## Instrucciones – Graficación Matricial Múltiple:

Escriba un programa que lea un archivo con extensión csv o txt (Excel o Bloc de notas) y que grafique sus valores. Este archivo debe contener varias columnas y varios renglones con valores decimales. Además, cada valor que se asigne a una columna tiene que estar separado por una coma del siguiente valor (Si es que se utilizó el bloc de notas).

En la primera columna del archivo considere que se tienen los valores del eje x, el cual es común para todos los datos. En las columnas subsiguientes considere que se tienen los valores que corresponden a los ejes  $y_N$ , donde N es el número total de columnas, por lo tanto, de gráficas.

Escriba el programa de tal forma que pueda leer datos separados por una coma o por una tabulación, por si este proviene de un archivo de Excel o de Bloc de notas:



### Pseudocódigo

1. Abrir el archivo.
2. Leer cada uno de los renglones con a variable tipo string line.
  - a. En este ejemplo la lectura del primer renglón es **line** = “1, 1, 3, 9”, teniendo la siguiente matriz:

$$\mathbf{data} = \begin{bmatrix} 1 & 1 & 3 & 9 \\ 2 & 4 & 4 & 16 \end{bmatrix}$$

3. Guardar los valores de la variable line en el vector mat1 de tipo string.
  - a. Este vector, en este ejemplo, tiene dos elementos que son, donde todos sus valores son un solo valor grande de tipo string:

$$mat1[0] = ['1,1,3,9']$$

$$mat1[1] = ['2,4,4,16']$$

4. Obtenga el número de renglones del vector.
5. Con el método Split usando el parámetro (",") se produce otro vector interno que separa los valores (de tipo string) que se alojaron en el vector.
  - a. Es decir:

$$mat1[0].split(",") = [1, 1, 3, 9]$$

$$mat1[1].split(",") = [2, 4, 4, 16]$$

- b. Luego por medio del método length obtenga el número total de columnas, que es igual al número de elementos del vector
6. Convertir la matriz de tipo string a que sea de tipo decimal.
7. Vaciar los valores del mat1 en la matriz data.
  - a. La matriz quedará como:

$$\mathbf{data} = \begin{bmatrix} 1 & 1 & 3 & 9 \\ 2 & 4 & 4 & 16 \end{bmatrix}$$

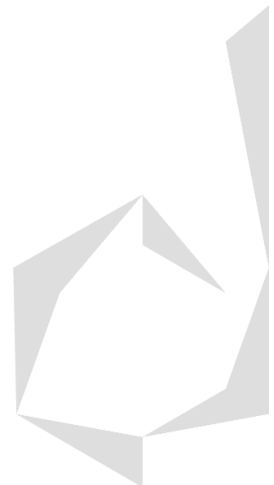
8. Obtener la transpuesta de la matriz Data.
  - a. La matriz quedará como:

$$\mathbf{data}^T = \begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 3 & 4 \\ 9 & 16 \end{bmatrix}$$

9. Asignar los renglones de la matriz data transpuesta a los vectores x o extraer dichos vectores por medio de un bucle for o for each:

$$\mathbf{x}_0 = [1, 2], \mathbf{y}_1 = [1, 4], \mathbf{y}_2 = [3, 4] \text{ y } \mathbf{y}_3 = [9, 16]$$

10. Graficar  $y_1$  vs  $x_0$ ,  $y_2$  vs  $x_0$  y  $y_3$  vs  $x_0$ , en la misma figura.



# Código Python – Visual Studio Code (Logo Azul):

```
# -*- coding: utf-8 -*-

#Comentario de una sola linea con el simbolo #, en Python para nada se deben poner acentos sino el programa
#puede fallar o imprimir raro en consola, la siguiente línea de código es para que no tenga error, pero aún
#así al poner un ángulo saldrá raro en consola, la línea debe ponerse tal cual como aparece y justo al inicio.

#GRAFICACIÓN MÚLTIPLE MATRICIAL CON MATPLOTLIB:
import matplotlib.pyplot as plt #matplotlib: Librería de graficación matemática.
import numpy as np #Librería numpy: Realiza operaciones matemáticas complejas (matriciales) y manejar datos.

#ABRIR y LEER UN ARCHIVO:
#Para leer una imagen o cualquier otro archivo se usa la dirección relativa o absoluta de un directorio:
# - Dirección relativa: Es una dirección que busca un archivo desde donde se encuentra la carpeta del
#   archivo python actualmente, esta se debe colocar entre comillas simples o dobles.
# - Dirección absoluta: Es una dirección que coloca toda la ruta desde el disco duro C o cualquier otro
#   que se esté usando hasta la ubicación del archivo, la cual se debe colocar entre comillas simples o
#   dobles.
# ..      : Significa que nos debemos salir de la carpeta donde nos encontramos actualmente.
# /        : Sirve para introducirnos a alguna carpeta cuyo nombre se coloca después del slash.
# .ext     : Se debe colocar siempre el nombre del archivo + su extensión.
#Es un archivo de Excel el que se va a crear, por eso tiene extensión .csv
filename = "Ejercicios Python con Archivos y Carpetas/16.-Graficación Matricial Múltiple/Datos2.csv"#Nombre del archivo a abrir
#open(): Método que sirve para abrir un archivo cualquiera, para ello es necesario indicar dos parámetros,
#el primero se refiere a la ruta relativa o absoluta del archivo previamente creado y la segunda indica qué
#es lo que se va a realizar con él, el contenido del archivo se asigna a una variable.
# - w: Sirve para escribir en un archivo, pero borrará la información que previamente contenía el archivo.
# - a: Sirve para escribir en un archivo sin que se borre la info anterior del archivo, se llama append.
# - r: Sirve para leer el contenido de un archivo.
new_file = open(filename, 'r') #Variable que guarda los datos que lea en el archivo de Excel.
#Los datos que están relacionados entre sí, se encuentran acomodados en forma de columna, pero el programa
#lo que hará es leer todas las filas de una en una para así crear una matriz de datos, esta matriz se irá
#guardando en la variable text.
text = ''
#Cada que se termine de leer una fila del archivo, se añadirá un valor a la variable rowsNumber, ya que esta
#indica cuántas filas existen en la matriz, para así después acomodar y relacionar todos los datos de una
#columna entre sí.
rowsNumber = 0
#La variable mat1 es una lista (array de python) vacía para guardar temporalmente todos los datos
#correspondientes a las distintas filas de la matriz creada con los datos de la matriz new_file extraída del
#archivo Excel, para así poder después transponer esta matriz y relacionar los datos entre sus columnas, para
#de esta forma exponer varias gráficas a través de una sola matriz en un solo figure.
mat1 = []
```

```

#Bucle for each para ir guardando temporalmente en la variable mat1 los datos de las columnas pertenecientes
#a la matriz de la variable new_file, extraídos del archivo de Excel.
for line in new_file:
    #De esta forma se agrega a la variable text de tipo String el contenido de la fila de datos.
    text += line
    #replace(): Método que reemplaza un caracter que se encuentra en un string por otro declarado por nosotros,
    #esto se ejecutará todas las veces que dicho caracter aparezca en el string.
    line = line.replace("\n", "") #Va a reemplazar los saltos de línea por un string vacío.
    line = line.replace(" ", "") #Quita los espacios por un string vacío.

    #find(): El método find devuelve el índice de la lista en donde se encuentre solamente el que primer caracter
    #indicado como su parámetro y si este no se encuentra en ningún lado, devuelve un -1.
    if(line.find(',')!=-1):
        #append(): Lo que hace es agregar un elemento a la lista (array de python).
        #split(): Método que crea una lista interna nueva en el index de la lista actual donde nos encontramos,
        #esto sucederá cuando se encuentre el caracter indicado como parámetro, creando así una lista interna
        #en cada una de las posiciones de la lista principal con los elementos que estén separados entre comas.
        mat1.append(line.split(',')) #Al encontrar una coma que separe los datos, se agrega una lista interna.
    if(line.find('\t')!=-1):
        mat1.append(line.split('\t')) #Al encontrar un tabulador que separe los datos, se agrega una lista interna.

    rowsNumber+=1 #Al terminar de analizar la fila de datos, se agrega un número a la variable rowsNumber.
    #print(): Imprimir un mensaje en consola.
    print("Fila número", rowsNumber, "extraída del archivo de Excel: ", "\n", line)

print("Matriz de filas extraídas del Archivo de Excel", "\n", mat1)
#var_file_open.close(): Método para cerrar un archivo previamente abierto con el método open(), es peligroso
#olvidar colocar este método, ya que la computadora lo considerará como si nunca hubiera sido cerrado, por lo
#cual no podré volver a abrirlo al dar clic sobre él.
new_file.close()

#El número de columnas será igual al número de elementos en cualquiera de las filas de los datos, recordemos que
#en cada una de las posiciones (index) de la variable mat1 existe una lista interna con los datos de cada una de
#las filas de la matriz de datos extraídas del archivo Excel.
#len(): Método que devuelve el número de elementos que tiene una lista (array de python).
columnsNumber = len(mat1[0])

#MATRIZ TRANSPUESTA: Se debe obtener la transpuesta de los datos extraídos porque siempre de los archivos de Excel
#estos son extraídos de forma vertical y los necesitamos de forma horizontal para que puedan ser graficados con
#python.
#np.asarray(): Convierte cualquier tipo de dato que pueda ser convertido en un array en un ndarray, que es un
#tipo de dato perteneciente a la librería numpy de Python.
data = np.asarray(mat1, dtype=float)

```

```

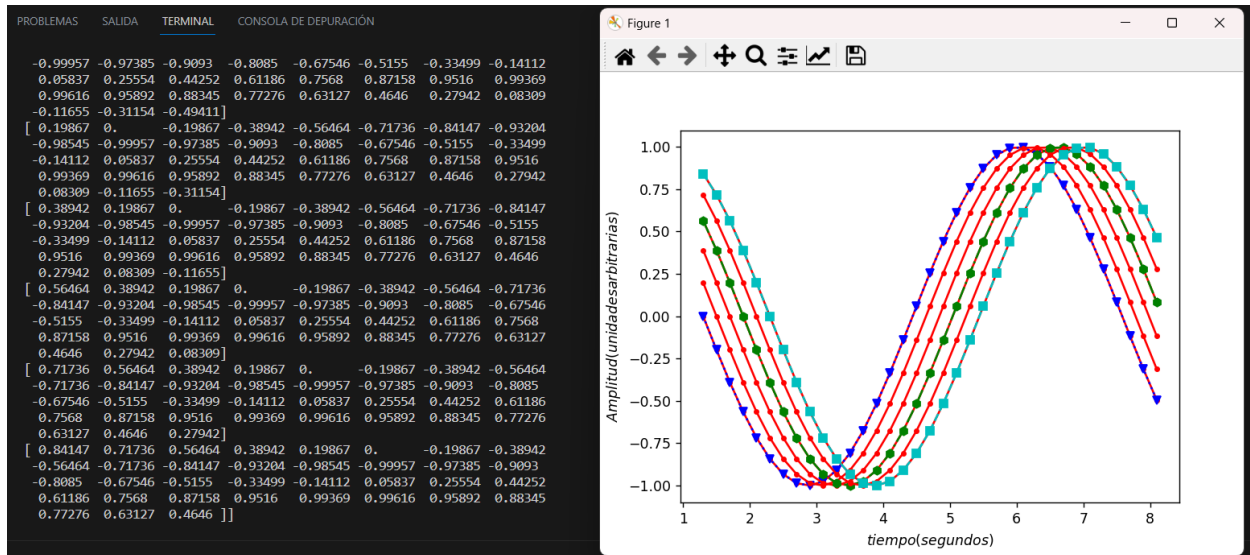
#ndarray.T: Método que obtiene el transpuesto de un ndarray, que pudo ser previamente obtenido con el método
#np.asarray().
data = data.T #Transpuesta de la matriz previamente obtenida en la variable mat1.
print("Matriz transpuesta de datos extraídos del Archivo de Excel", "\n",data)


#GRAFICAR LOS DATOS EN PYTHON:
#matplotlib.figure(): Método usado para crear la ventana de graficación (objeto pyplot).
fig = plt.figure()#Creación del objeto pyplot
#matplotlib.axes(): Método usado para crear la rejilla en la ventana de graficación.
ax = plt.axes()#Se crea un objeto ax perteneciente de la clase axes()
#Bucle for para graficar las distintas gráficas descritas por una matriz.
for i in range(1, columnsNumber):
    #axes.plot(): Método usado para crear la gráfica en la rejilla previamente creada en la ventana de graficación,
    #indicando como primer parámetro su eje horizontal, luego su eje vertical y finalmente el estilo de la gráfica:
    # - Colores:          C1: color naranja, r: color rojo, b: color azul, g: verde, c: cyan, m: morado,
    #   y: amarillo, k: negro, w: blanco.
    # - Colores: xkcd:aqua, xkcd:aquamarine, xkcd:azure, xkcd:beige, etc. Los colores se pueden obtener de este link:
    #   https://matplotlib.org/stable/tutorials/colors/colors.html
    # - Tipo de marcadores:  o: círculos, +: símbolos de más, .: puntos, v: Triángulo hacia abajo, h: Hexágono,
    #   etc.
    # - Tipo de Líneas:      -: sólida, --: punteada (líneas), :: punteada (puntos), -.: línea y punto,
    #   'or': Nada.
    ax.plot(data[0], data[i], 'r-.')# 'c.-' significa C1: color cyan, .: simbolo de punto -: línea sólida.
    #Si se quiere, se puede indicar un estilo específico para alguno de los gráficos con un if, en este caso se
    #está dando un estilo diferente a la primera gráfica, la de en medio y la última.
    if(i == 1):
        #Primera gráfica
        ax.plot(data[0], data[i], 'bv:')# 'bv:' significa b: color azul, v: simbolo de triángulos hacia abajo :: línea punteada.
    if(i == round(len(data)/2)):
        #Gráfica de en medio
        ax.plot(data[0], data[i], 'gh-.')# 'bv:' significa b: color azul, v: simbolo de triángulos hacia abajo :: línea punteada.
    if(i == len(data)-1):
        #Última gráfica
        ax.plot(data[0], data[i], 'cs--')# 'bv:' significa b: color azul, v: simbolo de triángulos hacia abajo :: línea punteada.
    #axes.set_xlabel(): Método para indicar el texto que aparece en el eje x.
    ax.set_xlabel(r'$tiempo (segundos)$') #Texto que aparece en el eje horizontal
    #axes.set_ylabel(): Método para indicar el texto que aparece en el eje y.
    ax.set_ylabel(r'$Amplitud (unidades arbitrarias)$') #Texto que aparece en el eje vertical
#matplotlib.show(): Método para mostrar la gráfica creada.
plt.show()

```



## Resultado del Código Python



## Código C# (.Net Framework) – Visual Studio (Logo Morado):

Código con resultado en GUI de Visual Studio .Net Framework:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

//Se usa la librería System.Numerics, agregada dando clic derecho en la parte de:
//Explorador de soluciones -> Referencias -> Agregar Referencia -> Nombre Librería.
using System.IO; //Librería utilizada para poder manipular archivos

namespace _8._Graficación_Matricial_Múltiple
{
    //CÓDIGO PARA LEER UN ARCHIVO Y APLICAR UNA GRAFICACIÓN MÚLTIPLE MATRICIAL:
    public partial class Form1 : Form
    {
        //Variables de instancia de la clase
        int rows; //Número de filas.
        int columns; //Número de columnas.
        double[,] data;
        public Form1()
        {
            InitializeComponent();
        } //Fin del constructor del GUI (Graphical User Interface) o Form.

        //Método que llena la matriz data con los datos leídos del archivo Excel.
        private void PopulateData(int rows, int columns, string[] lines)
        {
            string[] temp = new string[columns];
            data = new double[rows, columns];
            for (int i = 0; i < rows; i++)
            {
                temp = lines[i].Split(',', '\t');
                for (int j = 0; j < columns; j++)
                {
                    data[i, j] = Convert.ToDouble(temp[j]);
                } //2do Bucle for.
            } //1er Bucle for.

        } //Fin del método PopulateData()
    }
}
```



```

/*Método que se emplea para obtener la transpuesta de la matriz de datos, Se debe obtener la transpuesta de los
datos extraídos porque siempre de los archivos de Excel estos son extraídos de forma vertical y los necesitamos
de forma horizontal para que puedan ser graficados con C#./
private double[,] Transposed(int rows, int columns)
{
    double[,] transposed;
    transposed = new double[columns, rows];
    for (int i = 0; i < columns; i++)
    {
        for (int j = 0; j < rows; j++)
        {
            transposed[i, j] = data[j, i];
        }
    }
    return transposed;
} //Fin del método Transposed()

//Método que se usa en la creación de la gráfica matricial múltiple.
private void Plot(int row, int col)
{
    double[,] transposed = Transposed(row, col);
    string seriesNum;
    double x;
    double y;
    for (int i = 1; i < col; i++)
    {
        //Creación dinámica de gráficas.
        seriesNum = "Series" + Convert.ToString(i);
        Plt_Data.Series.Add(seriesNum);
        Plt_Data.Series[seriesNum].ChartType =
            System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Point;
        for (int j = 0; j < row; j++)
        {
            x = transposed[0, j];
            y = transposed[i, j];
            Plt_Data.Series[seriesNum].Points.AddXY(x, y);
        }
    }
} //Fin del método Plot()

//Evento que se produce cuando se oprime el botón de Btn_ReadPlotData
private void Btn_ReadPlotData_Click(object sender, EventArgs e)
{
    //Código para abrir el archivo de Excel.
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Multiselect = false;
    //Uso de una ventana emergente que muestre el explorador de archivos para seleccionar el archivo Excel.
    if (openFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
    {
        string line;
        string[] saveLine = new string[0];
        string[] temp;
        Txt_DisplayData.Text = "";
        System.IO.Stream stream = openFileDialog.OpenFile();
        using (System.IO.StreamReader reader = new System.IO.StreamReader(stream))
        {
            //Bucle while que lee todas las líneas del archivo Excel hasta que no encuentre más caracteres
            while ((line = reader.ReadLine()) != null)
            {
                Array.Resize(ref saveLine, saveLine.Length + 1);
                saveLine[saveLine.Length - 1] = line;
            } //Bucle while
        } //Instrucción using
        stream.Close(); //Close(): Método que sirve para cerrar el archivo abierto.

        /*Se asigna a la propiedad wordwrap un valor booleano False en la caja de texto para que se muestren las
        dos barras de desplazamiento, y que así se puedan ver completamente los datos leídos del archivo Excel.*/
        //Bucle for para desplegar en la caja de texto Txt_DisplayData los datos recopilados del archivo Excel.
        for (int i = 0; i < saveLine.Length; i++)
        {
            Txt_DisplayData.Text += saveLine[i] + "\r\n";
        }
        /*El método Split() se aplica a la primera línea leída de los datos provenientes del archivo Excel para así
        generar un vector, cada elemento de este vector corresponde al valor de una columna.
        Con el método Length aplicado al vector temp, se proporciona el número de columnas del archivo.*/
        temp = saveLine[0].Split(',', '\t');
        columns = temp.Length;
    }
}

```

```

        rows = saveLine.Length;
        //Se llena la matriz data con los datos recopilados.
        PopulateData(rows, columns, saveLine);
        Plt_Data.Series.Clear(); //Limpieza de la gráfica.
        Plot(rows, columns); //Gráfica de los datos.
    }
} //Fin del método Btn_ReadPlotData_Click()
} //Fin de la clase Form del GUI
} //Fin del namespace

```

## Código C# de la interfaz gráfica (GUI) [Design .cs]:

```

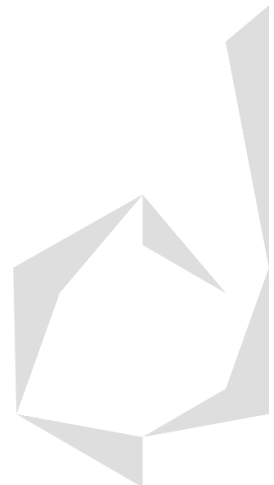
namespace _10._Graficación_Matricial_Múltiple
{
    partial class Form1
    {
        /// <summary>
        /// Variable del diseñador necesaria.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Limpiar los recursos que se estén usando.
        /// </summary>
        /// <param name="disposing">true si los recursos administrados se deben desechar; false en caso contrario.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Código generado por el Diseñador de Windows Forms

        /// <summary>
        /// Método necesario para admitir el Diseñador. No se puede modificar
        /// el contenido de este método con el editor de código.
        /// </summary>
        private void InitializeComponent()
        {
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea1 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend1 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Series series1 = new
System.Windows.Forms.DataVisualization.Charting.Series();
            this.button1 = new System.Windows.Forms.Button();
            this.Plt_Data = new System.Windows.Forms.DataVisualization.Charting.Chart();
            this.Txt_display_data = new System.Windows.Forms.TextBox();
            ((System.ComponentModel.ISupportInitialize)(this.Plt_Data)).BeginInit();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(79, 13);
            this.button1.Margin = new System.Windows.Forms.Padding(4);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(195, 53);
            this.button1.TabIndex = 0;
            this.button1.Text = "Leer y graficar datos";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.Btn_ReadPlotData_Click);
            //
            // Plt_Data
            //
            chartArea1.Name = "ChartArea1";
            this.Plt_Data.ChartAreas.Add(chartArea1);
            legend1.Name = "Legend1";
            this.Plt_Data.Legends.Add(legend1);
            this.Plt_Data.Location = new System.Drawing.Point(351, 13);
            this.Plt_Data.Margin = new System.Windows.Forms.Padding(4);
            this.Plt_Data.Name = "Plt_Data";

```



```

series1.ChartArea = "ChartArea1";
series1.Legend = "Legend1";
series1.Name = "Series1";
this.Plt_Data.Series.Add(series1);
this.Plt_Data.Size = new System.Drawing.Size(703, 528);
this.Plt_Data.TabIndex = 1;
this.Plt_Data.Text = "chart1";
this.Plt_Data.Click += new System.EventHandler(this.Btn_ReadPlotData_Click);
//
// Txt_display_data
//
this.Txt_display_data.Location = new System.Drawing.Point(13, 74);
this.Txt_display_data.Margin = new System.Windows.Forms.Padding(4);
this.Txt_display_data.Multiline = true;
this.Txt_display_data.Name = "Txt_display_data";
this.Txt_display_data.ReadOnly = true;
this.Txt_display_data.ScrollBars = System.Windows.Forms.ScrollBars.Both;
this.Txt_display_data.Size = new System.Drawing.Size(330, 467);
this.Txt_display_data.TabIndex = 2;
this.Txt_display_data.WordWrap = false;
//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
this.AutoScaleModeMode = System.Windows.Forms.AutoScaleModeMode.Font;
this.ClientSize = new System.Drawing.Size(1067, 554);
this.Controls.Add(this.Txt_display_data);
this.Controls.Add(this.Plt_Data);
this.Controls.Add(this.button1);
this.Margin = new System.Windows.Forms.Padding(4);
this.Name = "Form1";
this.Text = "Form1";
this.Click += new System.EventHandler(this.Btn_ReadPlotData_Click);
((System.ComponentModel.ISupportInitialize)(this.Plt_Data)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();
}

#endregion

private System.Windows.Forms.Button button1;
private System.Windows.Forms.DataVisualization.Charting.Chart Plt_Data;
private System.Windows.Forms.TextBox Txt_display_data;
}
}

```

## Resultado de la GUI creada con Código C# en Visual Studio

