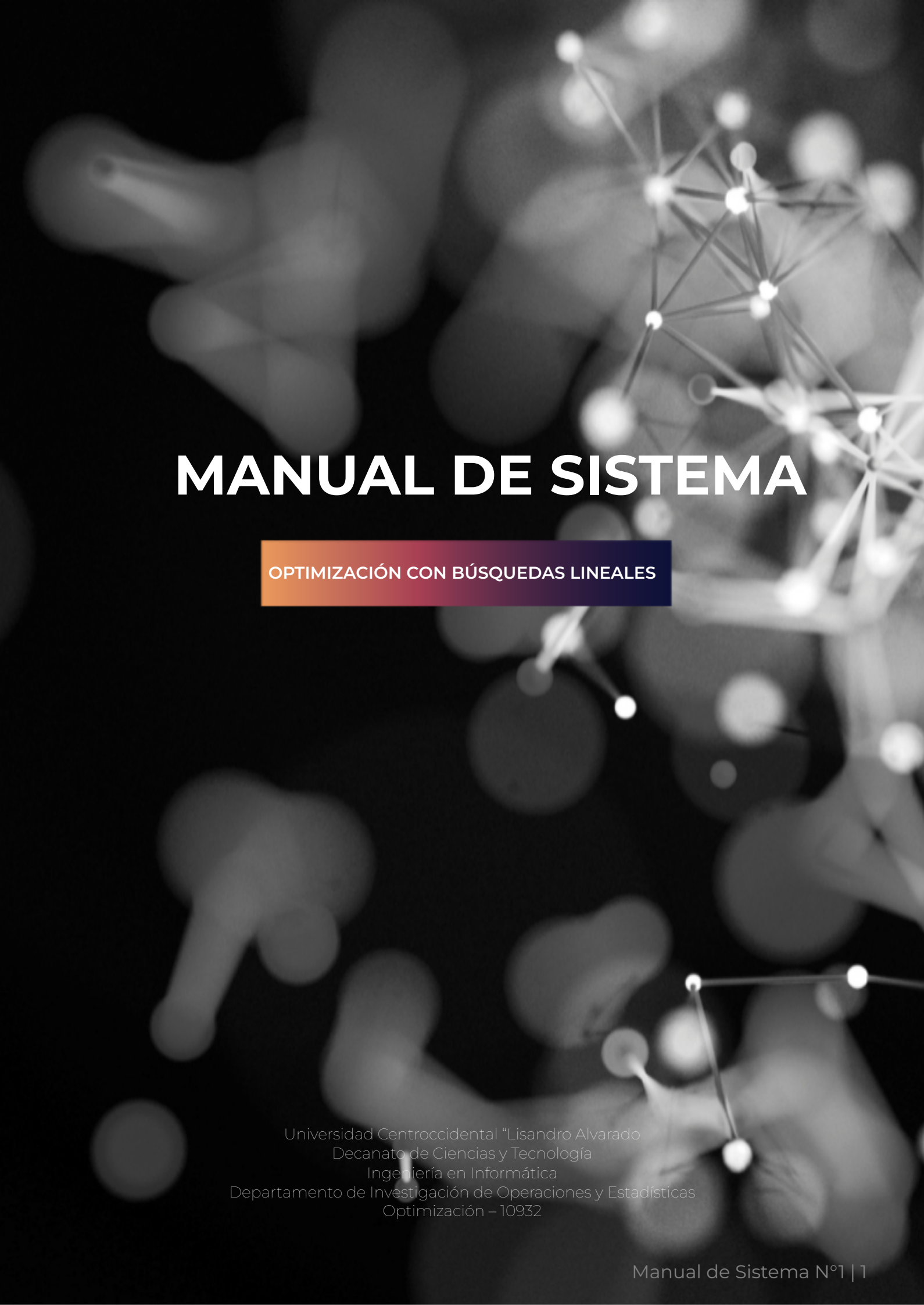
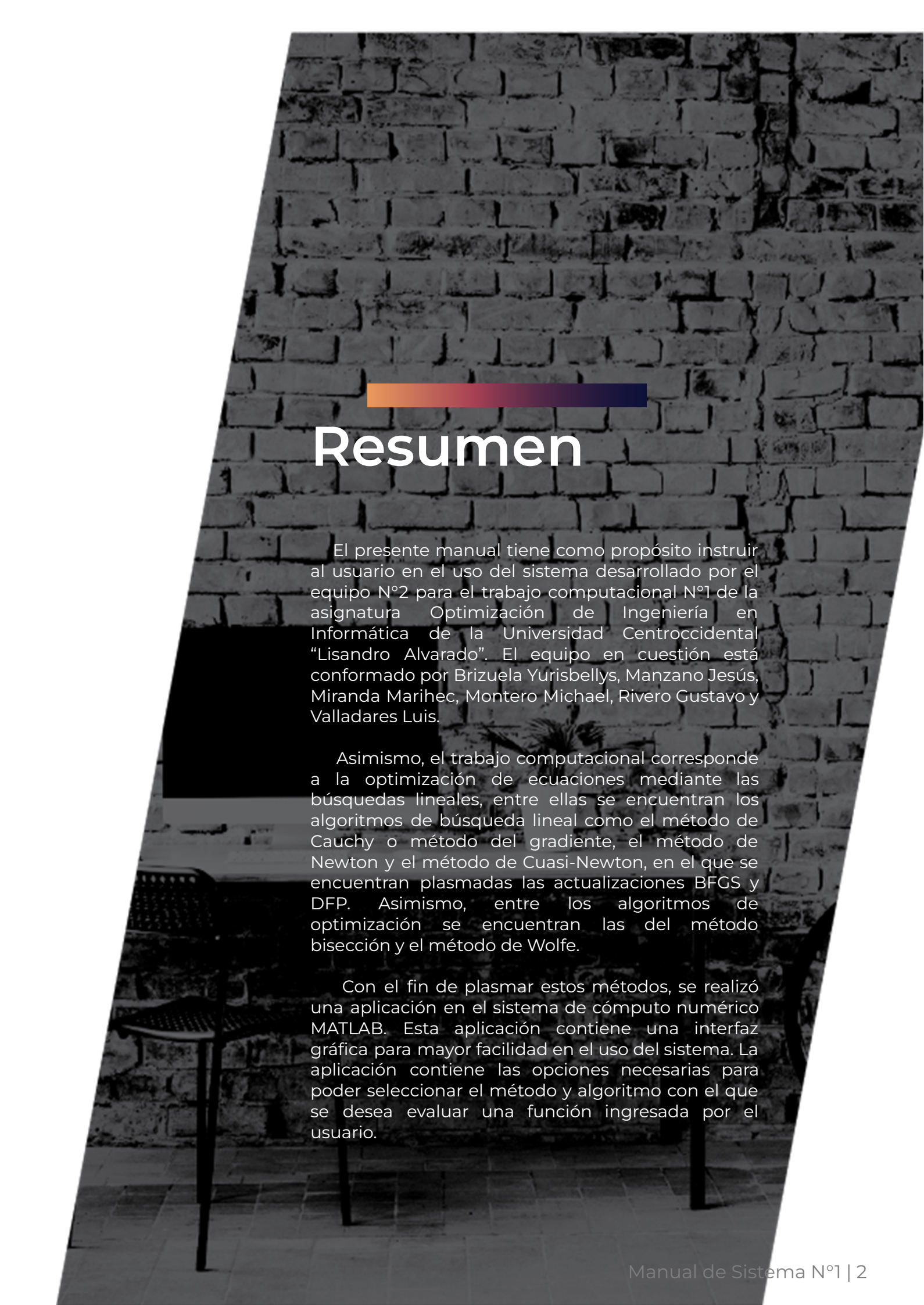


MANUAL DE SISTEMA

OPTIMIZACIÓN CON BÚSQUEDAS LINEALES

Universidad Centroccidental "Lisandro Alvarado
Decanato de Ciencias y Tecnología
Ingeniería en Informática
Departamento de Investigación de Operaciones y Estadísticas
Optimización – 10932



Resumen

El presente manual tiene como propósito instruir al usuario en el uso del sistema desarrollado por el equipo N°2 para el trabajo computacional N°1 de la asignatura Optimización de Ingeniería en Informática de la Universidad Centroccidental “Lisandro Alvarado”. El equipo en cuestión está conformado por Brizuela Yurisbellys, Manzano Jesús, Miranda Marihec, Montero Michael, Rivero Gustavo y Valladares Luis.

Asimismo, el trabajo computacional corresponde a la optimización de ecuaciones mediante las búsquedas lineales, entre ellas se encuentran los algoritmos de búsqueda lineal como el método de Cauchy o método del gradiente, el método de Newton y el método de Cuasi-Newton, en el que se encuentran plasmadas las actualizaciones BFGS y DFP. Asimismo, entre los algoritmos de optimización se encuentran las del método bisección y el método de Wolfe.

Con el fin de plasmar estos métodos, se realizó una aplicación en el sistema de cómputo numérico MATLAB. Esta aplicación contiene una interfaz gráfica para mayor facilidad en el uso del sistema. La aplicación contiene las opciones necesarias para poder seleccionar el método y algoritmo con el que se desea evaluar una función ingresada por el usuario.

Tabla de contenido

Requerimientos

Para el proyecto actual, se hará uso del programa MATLAB, en su versión 2021, contando con apoyo del lenguaje de programación que el mismo provee. Dicho programa es utilizado principalmente para el cálculo numérico y el análisis de datos, sirve para resolver problemas matemáticos y científicos, y es utilizado en diversas aplicaciones, como el análisis de señales y sistemas, el procesamiento de imágenes, la simulación y la optimización. Para hacer uso de los metodos de optimizacion con busquedas no lineales que se estudiarán en el proyecto, se debe contar con lo siguiente:

- **MATLAB** (versión R2021a o superior).
- **Archivo del proyecto** "*Optimización de búsquedas no lineales*".

Además, contar con un equipo de cómputo con las siguientes características:

- Sistema operativo compatible (Windows, MacOS o Linux).
- Procesador de 64-bit de 4 núcleos de 2.7GHz (mínimo).
- 4 Gb de RAM (Mínimo) u 8 Gb de RAM (Recomendado).
- 5 Gb de espacio para instalar MATLAB y sus paquetes adicionales (como Symbolic Math Toolbox).

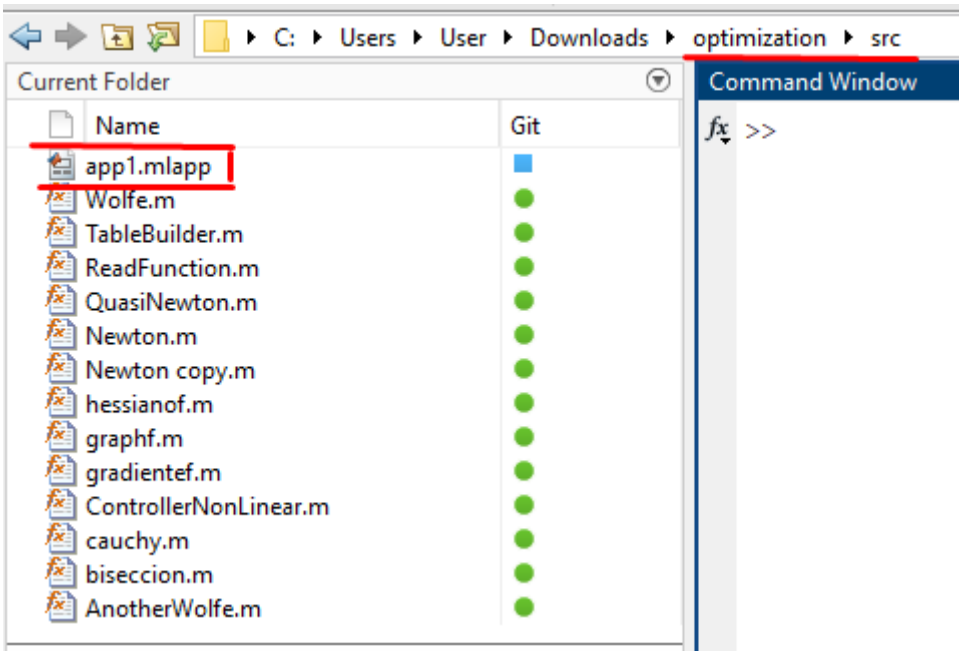
Ejecutar el programa

El programa cuenta con diversos archivos que pueden ser ejecutados desde la consola o ejecutar la interfaz gráfica la cual ofrece una mejor experiencia de usuario. A continuación explicaremos cómo ejecutar la interfaz de usuario.

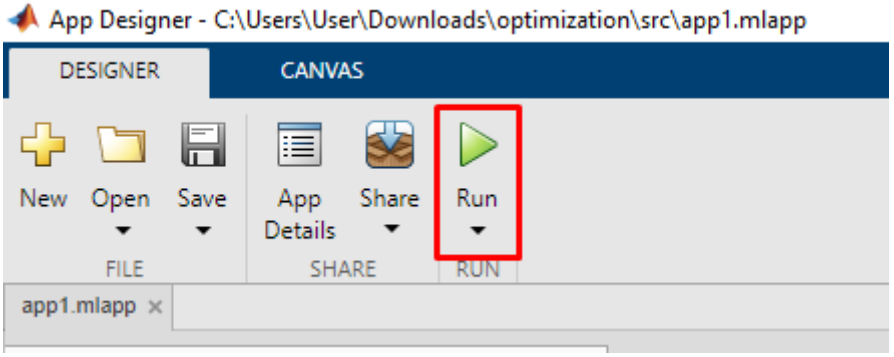
Pasos para ejecutar la interfaz de usuario:

1. Clonar el repositorio
   git clone https://github.com/Damurq/optimization

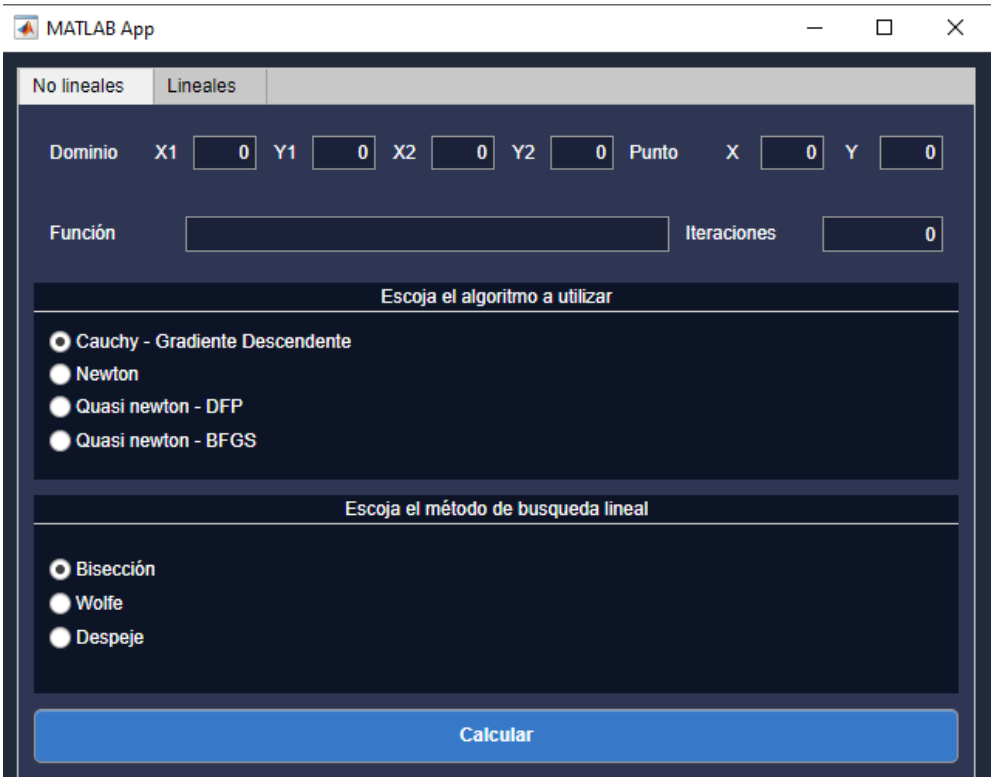
2. Abrir Matlab, entrar en la carpeta del proyecto, luego en la carpeta “src” y por último hacer doble click sobre el archivo *app1.mlapp*.



3. Se abrirá una nueva ventana, aquí se debe presionar el botón “Run” en la parte superior del menú de opciones.

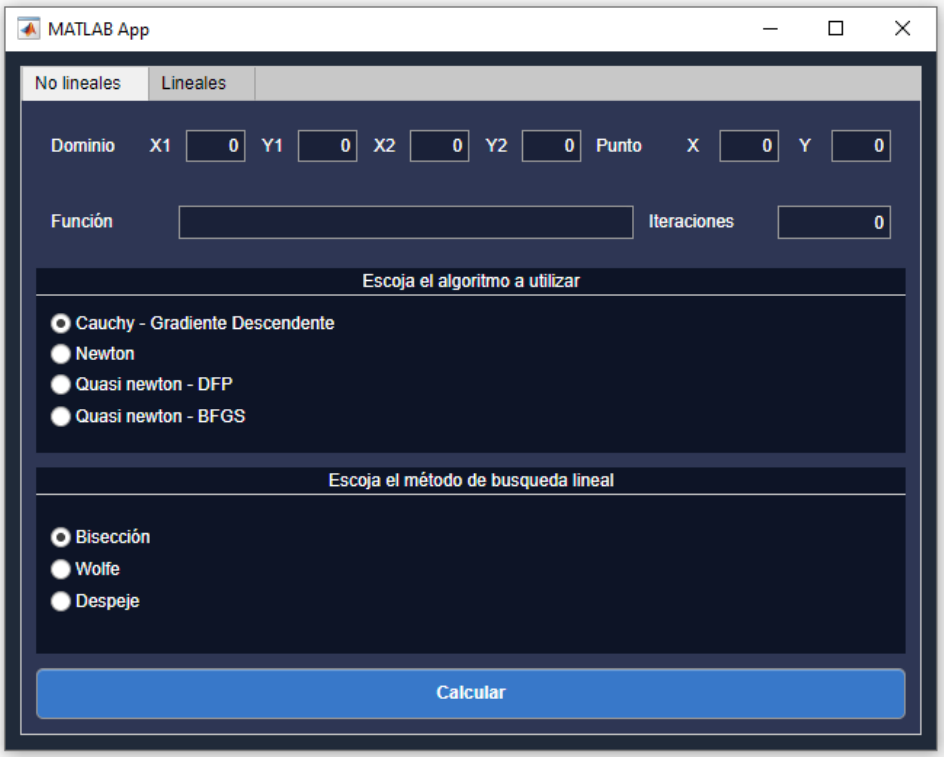


4. Nuestra aplicación se ha ejecutado, ahora es posible comenzar a usarla

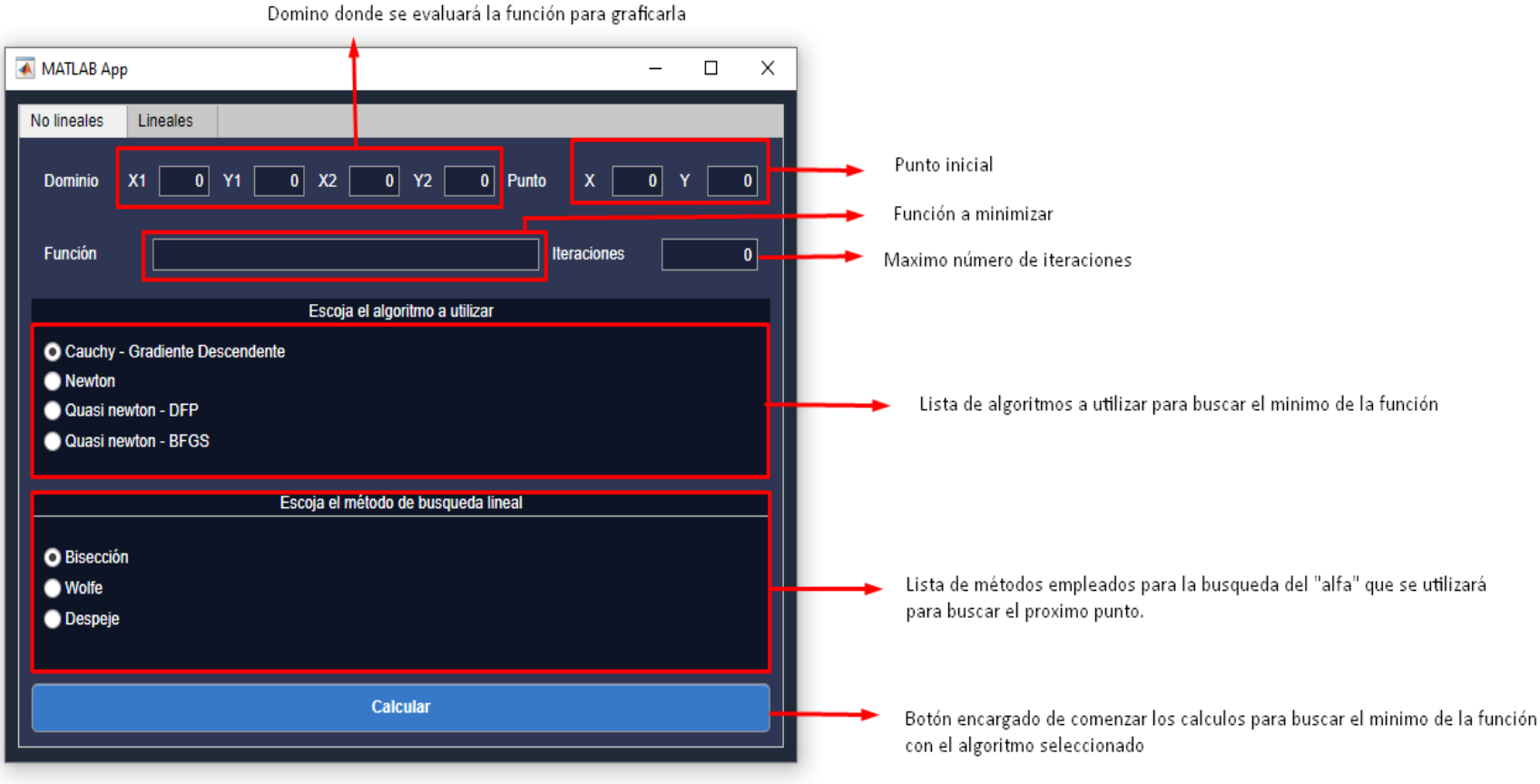


Interfaz gráfica

El programa desarrollado está compuesto de una ventana que posee varios campos para introducir los datos del problema a minimizar, así como una serie de radio buttons para seleccionar el algoritmo a emplear.



A continuación un **esquema detallado** de la interfaz.



Campos a llenar

Dominio

Representa el dominio que utilizará el programa para generar la gráfica de la superficie, para ello debemos introducir cuatro (4) valores numéricos: X_1, X_2, Y_1, Y_2.

Punto

Es el punto inicial desde el cual el algoritmo comenzará a iterar.

Función

La función objetivo a minimizar.

Iteración

Número máximo de iteraciones

Opciones a seleccionar

Algoritmo de optimización

En esta sección, se encuentran cuatro (4) algoritmos para minimizar una función, estos son: Cauchy (de descenso acelerado o de mayor pendiente), Newton, Quasi Newton (con actualizaciones: DFP y BFGS)

Búsquedas lineales

Aquí, se tienen tres (3) métodos para buscar "α " en la ecuación del punto k, $x^k = x^{k-1} + \alpha \cdot d$, la última opción llamada "despeje", es una opción adicional, la cual despeja el alfa si es empleada con Cauchy, mientras que para los otros tres (3) algoritmo toma el valor de "α " igual a uno (1).

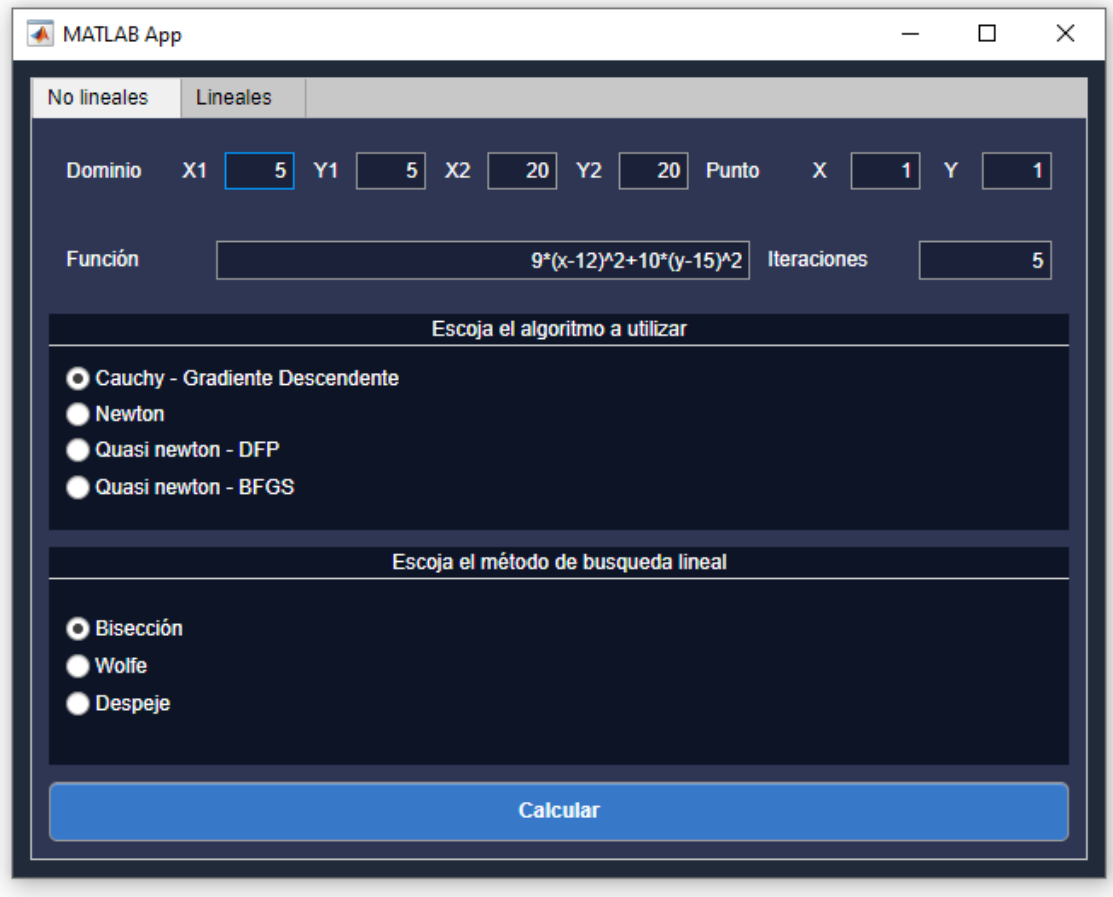
Ejecución del programa

Una vez que se han llenado los campos correspondientes y seleccionado la opción correcta se procede a pulsar el botón de "Calcular", esto hará que el programa comience a ejecutarse, de este modo se genera una gráfica de la superficie con una leyenda del punto inicial y final y una tabla con los datos y valores de cada iteración

Ejemplo:

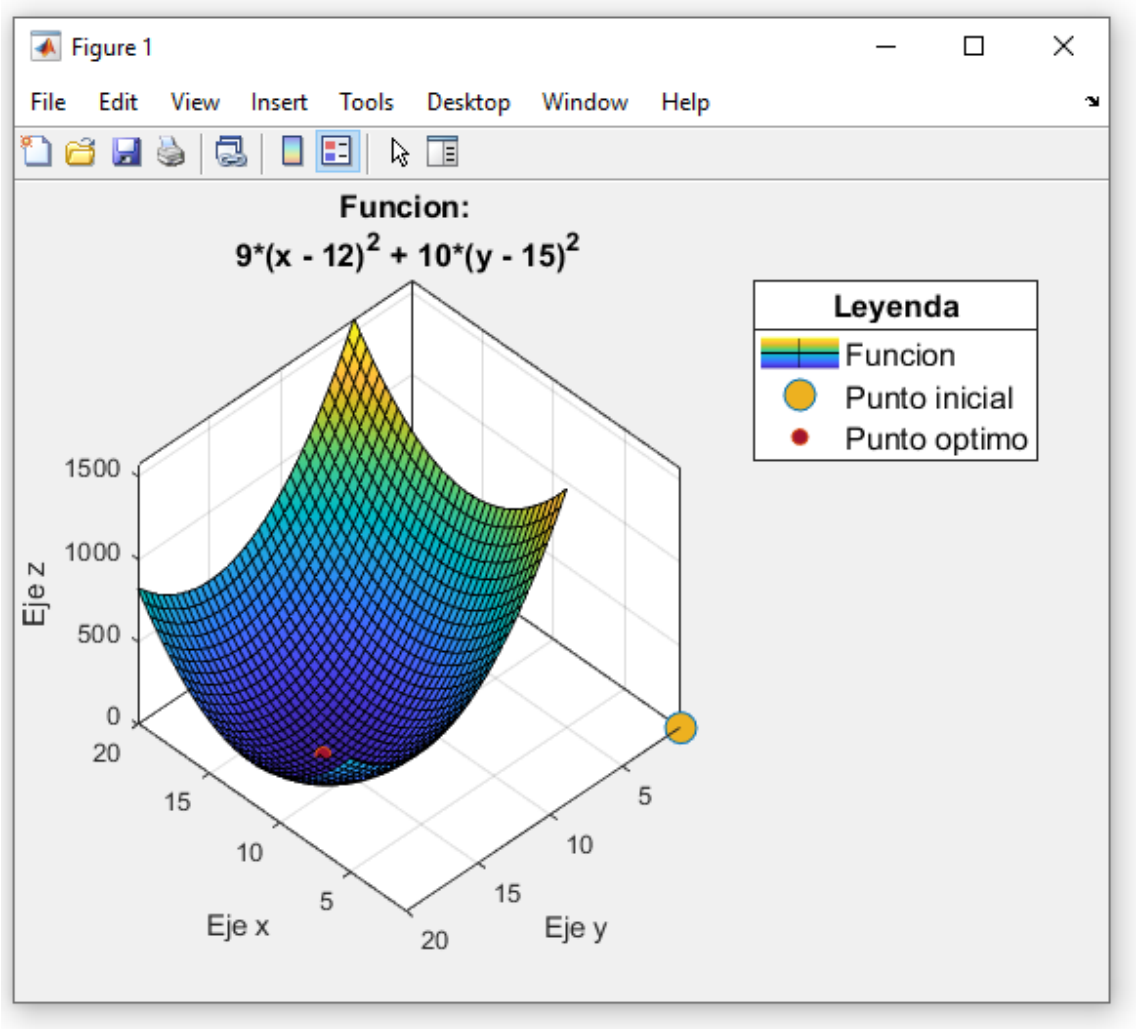
Datos a introducir:

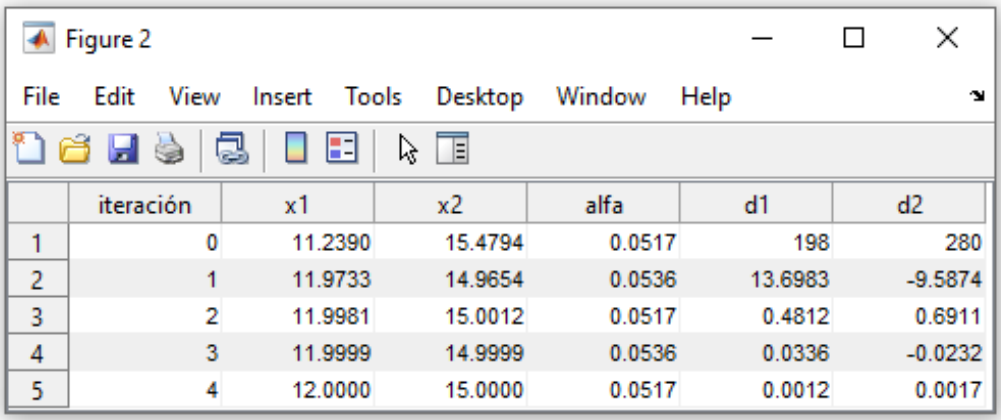
- f: $9(x-12)^2+10(y-15)^2$
- **Iteraciones:** 5
- **Punto:** $(1,1)$
- **Dominio para la gráfica:** $(x_1,x_2)\ [5,\ 20]\ -\ (y_1,y_2)\ [5,\ 20]$
- **Algoritmo de optimización:** Cauchy
- **Método de búsqueda lineal:** Bisección



El programa abrirá dos (2) pestañas, una contiene la gráfica de la superficie de la función con el punto de inicio y el punto final encontrado

por el algoritmo y la otra una tabla con los valores obtenidos en cada iteración.





	iteración	x1	x2	alfa	d1	d2
1	0	11.2390	15.4794	0.0517	198	280
2	1	11.9733	14.9654	0.0536	13.6983	-9.5874
3	2	11.9981	15.0012	0.0517	0.4812	0.6911
4	3	11.9999	14.9999	0.0536	0.0336	-0.0232
5	4	12.0000	15.0000	0.0517	0.0012	0.0017

Primer problema

Se solicita minimizar la siguiente función:

$$f(x_1, x_2) = \ -2x_1x_2 - 2x_2 + x_1^2 + x_2^2$$

Para el uso del sistema desarrollado, se realiza la siguiente adaptación a la función objetivo:

$$f(x, y) = \ -2xy - 2y + y^2 + x^2$$

Con el fin de utilizar el sistema, se lleva la función al siguiente formato:

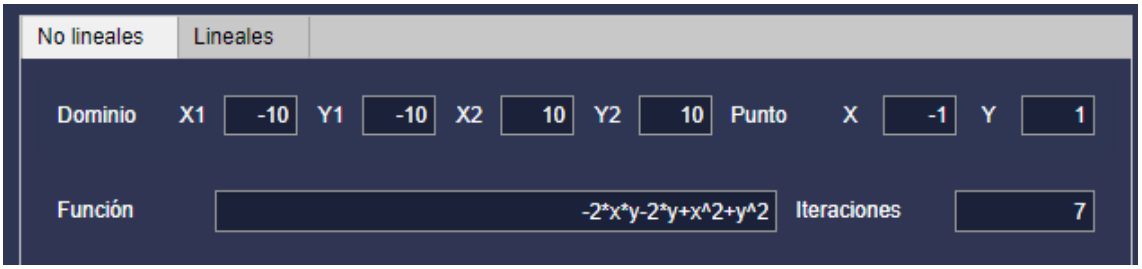
$$-2 * x * y - 2 * y + x\text{^}2 + y\text{^}2$$

Tras haber iniciado el sistema por medio del archivo *app1.mlapp*, se mostrará la interfaz gráfica del sistema. Se selecciona la pestaña superior para métodos no lineales, la cual lleva el nombre “No lineales” y se visualizará la pantalla en donde se ingresarán los datos.

Los datos a ingresar en el sistema con el fin de ilustrar la ejecución del mismo mediante el uso de diferentes métodos, son:

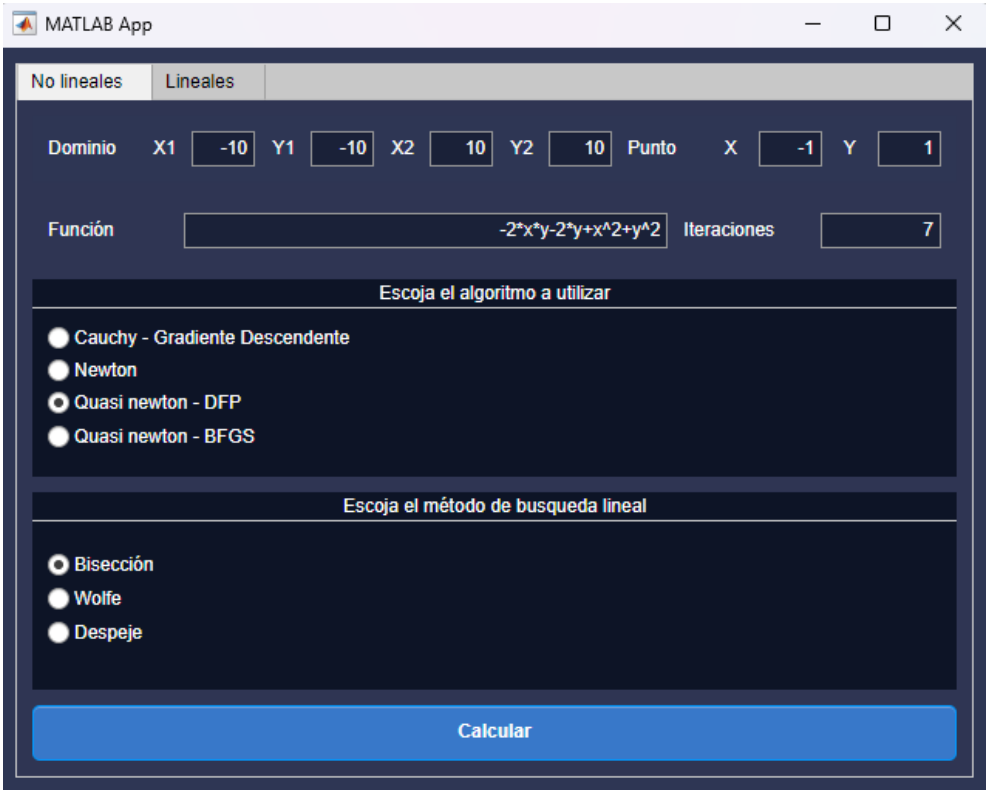
- **Punto:** $x^0 = [x_0, y_0] = [-1, 1]$
- **Iteraciones:** $k = 7$
- **Función**: $f(x, y) = -2 * x * y - 2 * y + x\text{^}2 + y\text{^}2$
- **Dominio:**
  - Eje x: $[-10, 10]$
  - Eje y: $[-10, 10]$

Nota: Los datos ingresados, exceptuando la función f, son valores arbitrarios con el fin de ilustrar el funcionamiento del sistema. Mientras más cercano sea el punto inicial x^0 al punto mínimo de la función, menos iteraciones se requieren para alcanzarlo. Asimismo, la cantidad de iteraciones k se propone como parámetro limitante con la intención de mitigar la carga exhaustiva sobre el procesador que crece conforme mayor sea la cantidad de iteraciones, es decir, con mayor cantidad de iteraciones, mayor necesidad de procesamiento en el equipo cómputo que ejecute el sistema, lo que significa, mayor tiempo en el cálculo de los valores o, incluso, lo suficiente como para congelar el sistema.



Búsqueda lineal: *Bisección*

Para la resolución con búsqueda lineal *bisección*, se hará uso del algoritmo de ***Quasi newton - DFP***, tal y como se muestra a continuación:



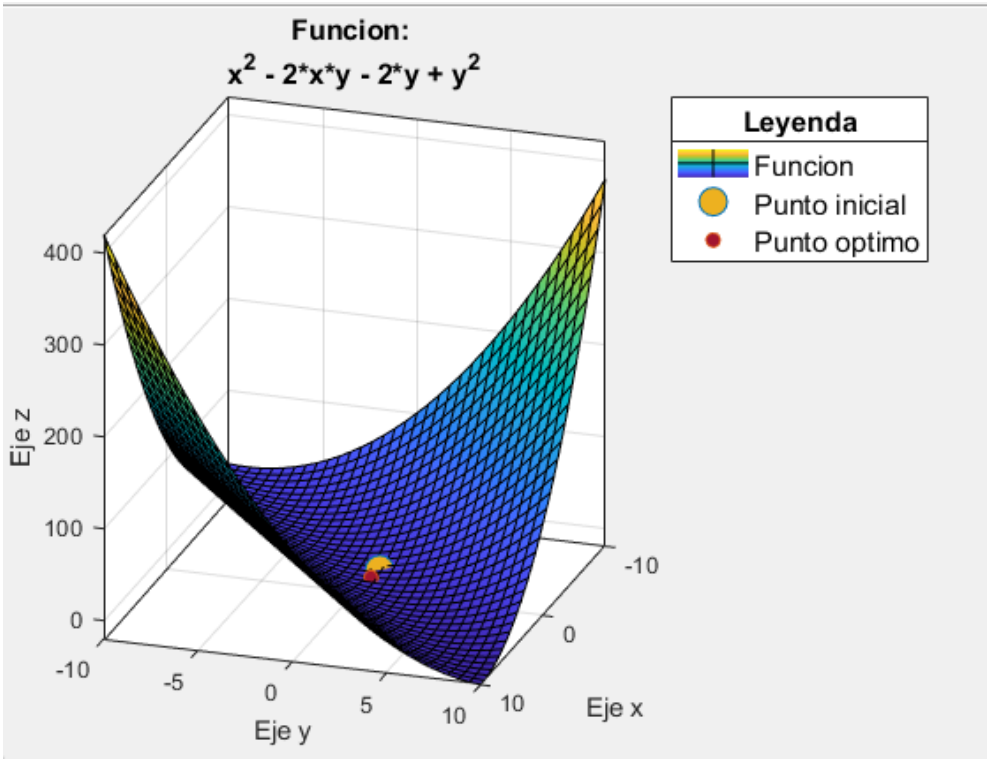
Al pulsar el botón **"Calcular"**, el sistema procesa los cálculos y luego de haber culminado, mostrará dos nuevas ventanas; la primera mostrará la tabla con los datos generados por el método de búsqueda lineal. La segunda ventana mostrará un gráfico tridimensional con el punto inicial y el punto óptimo. Esta última, posee una leyenda para reconocer los datos ilustrados.

Figure 2: Tabla del método Cuasi-Newton

File Edit View Insert Tools Desktop Window Help

	iteración	x1	x2	alfa	g(x)	f(x)
1	0	-1	1	0.2773	4.4721	2
2	1	0.1094	0.4453	0.2773	1.4884	-0.7778
3	2	0.1109	0.4445	0.0039	1.4904	-0.7778
4	3	0.0064	0.7588	0.6445	1.5841	-0.9515
5	4	0.0048	0.7611	0.0039	1.5892	-0.9503
6	5	0.2823	0.6200	0.4180	1.4869	-1.1259
7	6	0.2840	0.6191	0.0039	1.4892	-1.1259
8	7	0.1818	0.9366	0.6602	1.5872	-1.3035

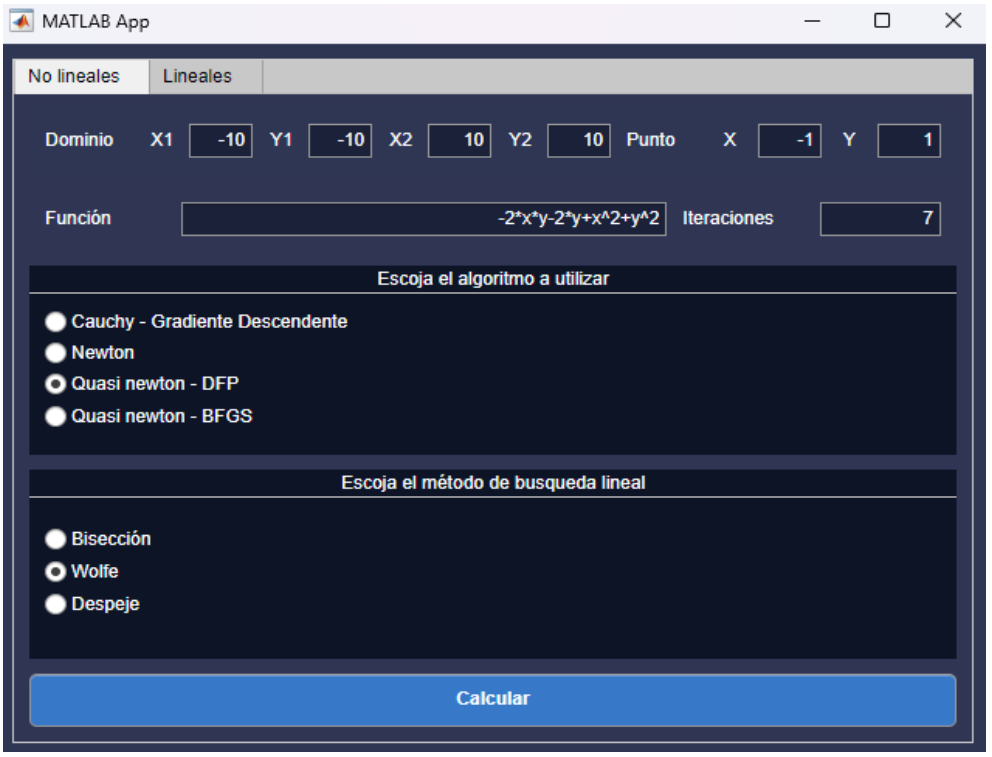
Tabla de datos para el método Quasi Newton



Gráfica de la función indicada por el usuario

Búsqueda lineal: *Wolfe*

Para la resolución con búsqueda lineal Wolfe, se hará uso del algoritmo de ***Quasi newton - DFP***, tal y como se muestra a continuación:



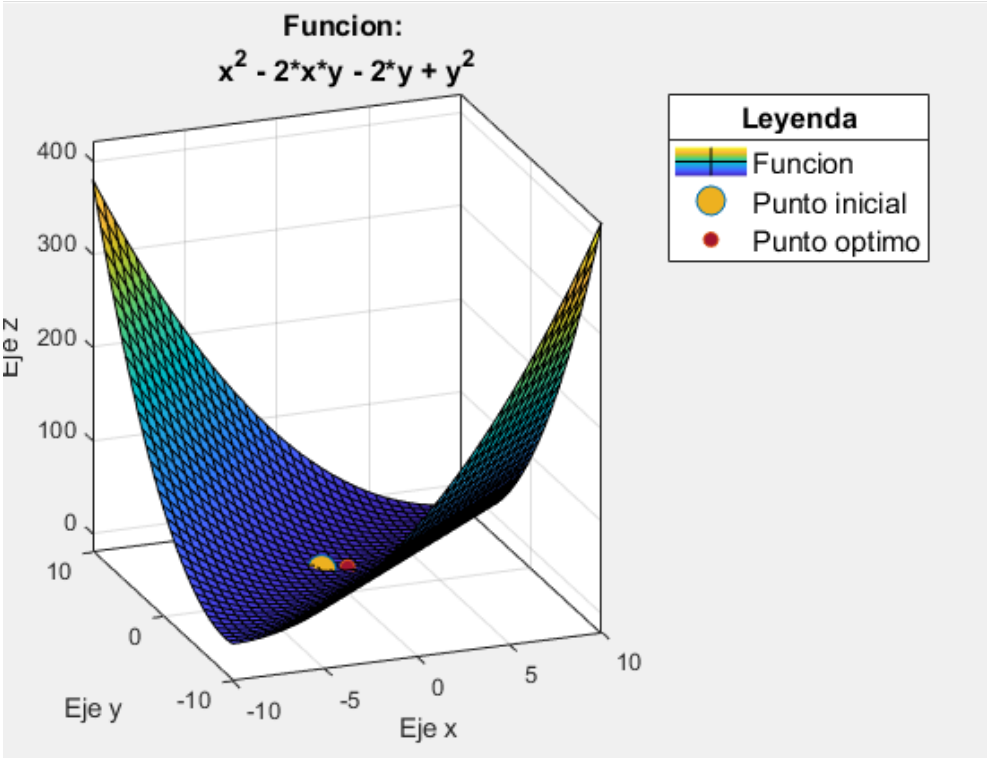
Al igual que para el cálculo anterior, se presiona el botón de **"Calcular"** y el sistema iniciará el procesamiento de datos. Tras un instante en el que culmina con este procesamiento, el sistema mostrará las dos ventanas para la tabla de datos calculados y la gráfica de la función con punto inicial y punto óptimo.

Figure 2: Tabla del método Cuasi-Newton

File Edit View Insert Tools Desktop Window Help

	iteración	x1	x2	alfa	g(x)	f(x)
1	0	-1	1	0.2501	4.4721	2
2	1	3.0000e-04	0.4999	0.2501	1.4142	-0.7501
3	2	0.0598	0.6162	0.9961	1.4232	-0.9228
4	3	0.1144	0.6212	0.2501	1.4143	-0.9856
5	4	0.1829	0.7368	0.9961	1.4224	-1.1668
6	5	0.2315	0.7398	0.2501	1.4144	-1.2213
7	6	0.3003	0.8552	0.9961	1.4227	-1.4024
8	7	0.3497	0.8567	0.2501	1.4144	-1.4563

Tabla de datos para el método Quasi Newton



Gráfica de la función indicada por el usuario

Observaciones

Se evidencia que la ejecución del sistema bajo los métodos de búsqueda lineal para ambos casos, se aproximan a los puntos $x_b = [0.1818,\ 0.9366]$ y $x_w = [0.3497,\ 0.8567]$, donde x_b corresponde al punto obtenido mediante el algoritmo de bisección y x_w al punto obtenido por el algoritmo de Wolfe. En ambos casos, los métodos van obteniendo puntos que minimizan a la función, por ejemplo, $f(x_b) = -1.3035$ y $f(x_w) = -1.4563$. De igual manera, se evidencia que, con la misma cantidad de iteraciones $k = 7$, la combinación correspondiente de Quasi Newton en su actualización DFP y el método de Wolfe, se aproximan con mayor velocidad al mínimo valor de la función.

Asimismo, el usuario está en libertad de realizar las diferentes combinaciones entre métodos y algoritmos con el fin de comprobar los diferentes resultados obtenidos, recordando que, todos apuntan en la misma dirección pero a velocidades y con comportamientos distintos.

Segundo problema

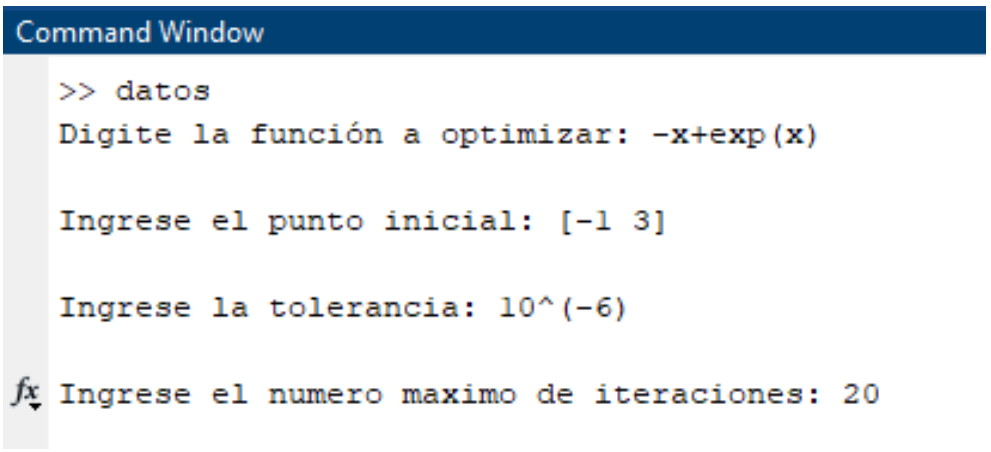
Se solicita aplicar el método de ***bisección*** para encontrar el x que minimice la función f en el intervalo $[-1, 3]$, donde f es:

$$f(x) = -x + e^{x}$$

Búsqueda lineal: *Bisección*

Tras haber iniciado la ejecución del script con nombre ***datos.m***, se visualizará en la ventana de comando de MATLAB la pantalla en donde se ingresarán los datos. Los datos que serán ingresados con el fin de ilustrar la ejecución y el funcionamiento del algoritmo de bisección, son:

- **Función**: $f(x, y) = -x + exp(x)$
- **Punto:** $x^{0} = [x_0, y_0] = [-1, 3]$
- **Iteraciones:** $k = 20$
- **Tolerancia:** $tol = 0.000001$



Datos ingresados por el usuario

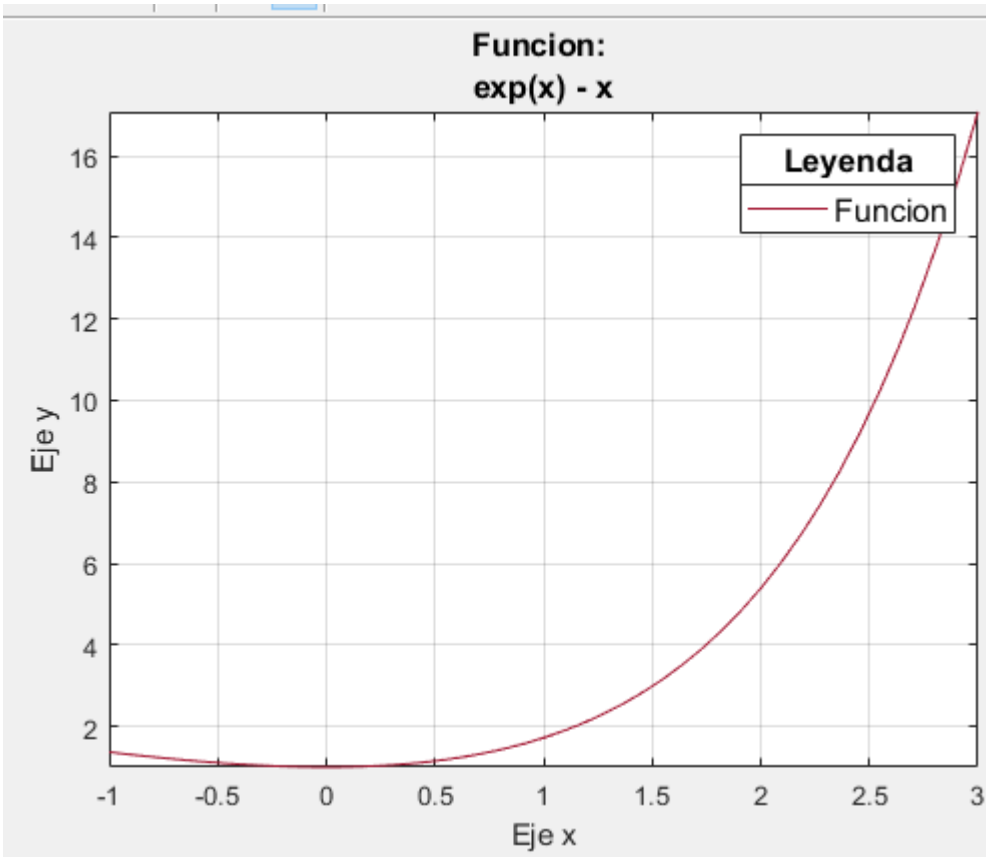
Luego de ingresar todos los datos requeridos, el sistema los procesará y ejecutará el algoritmo de bisección. Luego de haber culminado, se muestra en la consola de MATLAB los datos generados por

el método de bisección implementado y se despliega una nueva ventana que mostrará un gráfico de la función dada.

```
el punto optimo es (-1.000,3.000)

La raiz de la solucion es 1.000
```

Datos generados por el método de bisección.



Gráfica de la función indicada por el usuario

Cabe destacar que implementamos un algoritmo el cual funciona **bisecando** repetidamente el intervalo $[a, b]$ y **actualizando** el intervalo en función del signo de la función en el punto medio. Esto permite que el algoritmo converja en una solución (un punto donde la función es igual a cero) con un alto grado de precisión, siempre que la función sea continua y el intervalo $[a, b]$ contenga una *raíz*.