|  |  |  |
| --- | --- | --- |
|  | **UNIVERSIDAD DE LOS ANDES**  **FACULTAD DE INGENIERÍA**  **DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN**  **Modelado, Simulación y Optimización**  **Profesor**  **Germán Montoya O.**  [**ga.montoya44@uniandes.edu.co**](mailto:ga.montoya44@uniandes.edu.co) |  |

|  |
| --- |
| **EXAMEN 2 - Solución** |

Julián Camilo Mora Valbuena

Luisa Fernanda Fuentes Ladino

# EJERCICIO 1 (10%): Método eConstraint en Pyomo

Se resolvió el mismo caso presentado en “multiobjetivoHopsCosts\_sumasPonderadas.py”, pero **implementando el método de eConstraint en Pyomo**.

Para ello, se realizaron los cambios correspondientes en la función objetivo, la cual solo puede incluir a una función. Además, la función restante se implementó dentro de las restricciones del problema, utilizando un épsilon. Para este caso, el épsilon es el número máximo de saltos que puede haber, más un valor de tolerancia, como se vio en clase.

Resultado:

**Gráfico, Gráfico de líneas

Descripción generada automáticamente**

**EJERCICIO 2 (20%): Encontrar el máximo global y el mínimo global de una función**

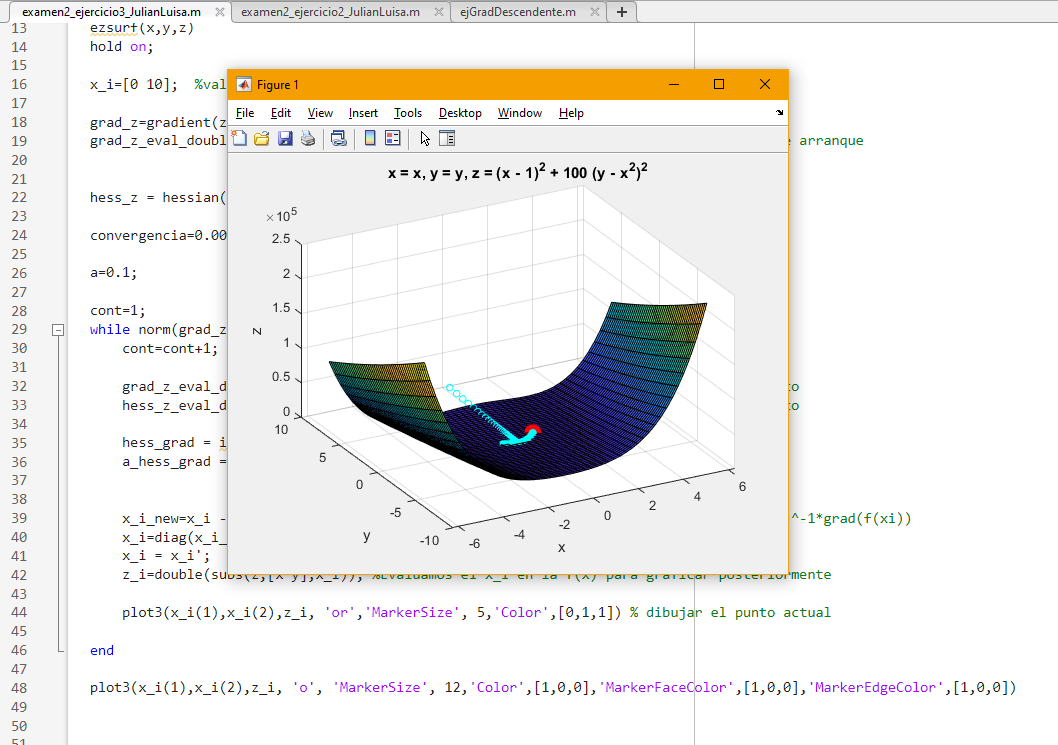
Utilizando el algoritmo de Newton Raphson, construimos una solución que recorre toda la función en el intervalo [-2.5,2.5]. Mientras lo recorre, utilizamos Newton Raphson para encontrar mínimos y máximos locales. Es decir, los puntos críticos. A su vez, estos puntos eran guardados en una lista de Matlab. Cuando el recorrido por la función finaliza, se buscan los mínimos y máximos globales evaluando los valores máximos y mínimos dentro del conjunto que contiene los puntos críticos. Finalmente, esos puntos se evalúan en la función inicial para obtener las coordenadas faltantes en y.

Interfaz de usuario gráfica, Gráfico, Aplicación, Gráfico de líneas

Descripción generada automáticamente

# EJERCICIO 3 (30%): Implementación de Newton Raphson para 3 dimensiones

Esta solución se realizo basada en el archivo ejNRaphson.m, el cual muestra la implementación de Newton Raphson para una función de una sola variable. Teniendo en cuenta lo visto en clase y el pseudocódigo dado, el archivo fue modificado para hacer los cálculos multivariables correspondientes al gradiente y el hessiano. Así, se obtuvo el punto crítico solicitado por el enunciado. El cual se encuentra en X y Y en coordenadas muy cercanas a 1 (0.999). Como se puede ver en la consola del programa.



# EJERCICIO 4: Implementación del Algoritmo Simplex (40%)

Esta solución se realizó basada en la Solución Geométrica a Problemas de

Optimización. Para esto primero se halló el polígono factible usando el método de solución geométrica. Luego, se hallaron los vértices (sus coordenadas) de dicho polígono, o sea, los FEVs. Después, se creó una estructura de datos donde se indiquen los vecinos de cada FEV. Así, las coordenadas de cada FEV y los vecinos de cada FEV se ingresan como parámetros al algoritmo que vas a diseñar. Con base en estas entradas y en la escogencia de un FEV aleatorio, el algoritmo automáticamente determina la solución óptima implementando la prueba de optimalidad del método Simplex para la siguiente ecuación y restricciones.

Texto

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente