MANUAL DE USUARIO

Sistema de Optimización Numérica

Métodos: Unidimensional, Multidimensional y con Restricciones

---

ÍNDICE

---

1. INTRODUCCIÓN

¿Qué es la Optimización?

La optimización es el proceso matemático de encontrar el **mejor valor** (mínimo o máximo) de una función sujeta a ciertas condiciones. Es fundamental en:

- **Economía:** Maximizar beneficios, minimizar costos
- **Ingeniería:** Diseño óptimo, control de procesos
- **Logística:** Rutas óptimas, gestión de inventarios
- **Ciencia de Datos:** Entrenamiento de modelos, ajuste de parámetros

Características del Programa

Este software implementa tres tipos de optimización:

1. **Optimización Unidimensional**
   - Una variable de decisión: f(x)
   - Busca mínimo en un intervalo [a, b]
   - Método: Brent (muy eficiente)
2. **Optimización Multidimensional Sin Restricciones**
   - Múltiples variables: f(x, y, ...)
   - Sin condiciones adicionales
   - Método: BFGS (Quasi-Newton)

3. **Optimización con Restricciones**
   - Múltiples variables con condiciones
   - Restricciones de igualdad: g(x, y) = 0
   - Método: SLSQP (Sequential Least Squares)

¿Qué puede hacer este programa?

Minimizar funciones de una o varias variables

Trabajar con funciones no lineales complejas

Incorporar restricciones de igualdad

Proporcionar soluciones numéricas precisas

Mostrar resultados claros y comprensibles

No maximiza directamente (usar -f(x) para maximizar) ❌ No maneja restricciones de desigualdad (≤, ≥) ❌ No optimiza variables enteras

---

2. REQUISITOS DEL SISTEMA

Hardware Mínimo

- Procesador: 1 GHz o superior
- RAM: 2 GB mínimo (4 GB recomendado)
- Espacio en disco: 100 MB

Software Requerido

Python

- **Versión mínima**: Python 3.7
- **Versión recomendada**: Python 3.9 o superior

Verificar instalación:

```
python --version
```

Bibliotecas Necesarias

- **NumPy**: Para cálculos numéricos
- **SciPy**: Para optimización (opcional, futuras extensiones)

Instalación de Dependencias

Windows:

pip install numpy scipy

Linux/Mac:

pip3 install numpy scipy

Verificar instalación:

python -c "import numpy; print('NumPy:', numpy.__version__)"

---

3. INSTALACIÓN Y CONFIGURACIÓN

Estructura de Archivos

```
Optimizacion_3/
├── main.py                # Programa principal (menú)
└── metodos/
    ├── unidimensional.py      # Optimización 1D
    ├── multidimensional.py    # Optimización nD sin restricciones
    └── restricciones.py       # Optimización con restricciones
```

Pasos de Instalación

1. **Descargar el proyecto**
   - Descomprimir el archivo .zip
   - Colocar en una carpeta de fácil acceso
2. **Instalar dependencias**
3. pip install numpy scipy
4. **Verificar archivos**
   - Asegurarse de que todos los archivos estén presentes
   - Verificar que la carpeta metodos/ esté en el mismo directorio que main.py

Métodos de Ejecución

Opción 1: Terminal (Recomendado)

Windows:

cd ruta\a\Optimizacion_3
python main.py

---

4. DESCRIPCIÓN DE LOS MÉTODOS

4.1 Optimización Unidimensional

¿Qué es?

Encuentra el valor de x que minimiza f(x) en un intervalo [a, b].

Algoritmo: Método de Brent

Combina tres técnicas:

- Búsqueda de la sección dorada
- Interpolación parabólica
- Búsqueda de bisección

Ventajas:

- No requiere derivadas
- Convergencia garantizada
- Muy rápido (convergencia super-lineal)
- Robusto numéricamente

Limitaciones:

- Solo una variable
- Requiere intervalo [a, b]
- Encuentra solo un mínimo (local o global)

¿Cuándo usarlo?

Casos ideales:

- Optimización de precios: f(precio) = beneficio
- Ajuste de parámetros: f(parámetro) = error
- Diseño 1D: f(dimensión) = costo/eficiencia

Ejemplo: Encontrar el precio que maximiza ingresos

```
Ingresos = precio × demanda(precio)
demanda(precio) = 1000 - 5×precio
```

4.2 Optimización Multidimensional Sin Restricciones

¿Qué es?

Encuentra los valores de x, y, ... que minimizan f(x, y, ...) sin condiciones adicionales.

Algoritmo: BFGS (Broyden-Fletcher-Goldfarb-Shanno)

Método Quasi-Newton que aproxima la matriz Hessiana.

Ventajas:

- Maneja múltiples variables
- No calcula segundas derivadas (más eficiente)
- Convergencia super-lineal

- Funciona bien en problemas no lineales

Limitaciones:

- Necesita función diferenciable
- Puede converger a mínimos locales
- Sensible al punto inicial

¿Cuándo usarlo?

Casos ideales:

- Diseño de productos con múltiples parámetros
- Ajuste simultáneo de varios parámetros
- Problemas de localización óptima

Ejemplo: Diseño de un envase cilíndrico

```
Minimizar material: f(radio, altura) = área_total
Sin restricciones de volumen (caso libre)
```

4.3 Optimización con Restricciones

¿Qué es?

Encuentra valores óptimos de variables que deben cumplir condiciones específicas.

Algoritmo: SLSQP (Sequential Least Squares Programming)

Método de programación cuadrática secuencial.

Ventajas:

- Maneja restricciones de igualdad: g(x,y) = 0
- Múltiples variables
- Convergencia cuadrática cerca del óptimo
- Muy usado en ingeniería

Limitaciones:

- Solo restricciones de igualdad en esta implementación
- Restricciones deben ser diferenciables
- Más sensible al punto inicial

¿Cuándo usarlo?

Casos ideales:

- Optimización con presupuesto limitado

- Diseño con especificaciones fijas
- Producción con cuotas obligatorias

Ejemplo: Maximizar beneficio con presupuesto fijo

```
Maximizar: Beneficio(x, y)
Restricción: Costo(x) + Costo(y) = Presupuesto
```

---

5. GUÍA DE USO PASO A PASO

Inicio del Programa

Al ejecutar `python main.py`, verás:

```
=== MENÚ DE OPTIMIZACIÓN ===
1. Optimización Unidimensional
2. Optimización Multidimensional (sin restricciones)
3. Optimización con Restricciones
4. Salir
Seleccione una opción:
```

5.1 Optimización Unidimensional

Ejemplo Guiado: Minimizar Costo de Producción

Función: $C(x) = x^2 - 6x + 10$

Paso 1: Seleccionar opción

```
Seleccione una opción: 1
```

Paso 2: Ingresar función

```
Ingrese la función f(x): (ej. x**2 + 3*x + 2): x**2 - 6*x + 10
```

Importante:

- Usar `x` como variable
- Usar `**` para potencias: x^2 = `x**2`
- Usar `*` para multiplicación: $3x$ = `3*x`

Paso 3: Definir intervalo

```
Límite inferior a: 0
Límite superior b: 10
```

Paso 4: Ver resultado

```
Resultado: f(x) mínima = 1.0000 en x = 3.0000
```

Interpretación:

- El costo mínimo es 1 (unidades monetarias)
- Se alcanza produciendo x = 3 (unidades)

5.2 Optimización Multidimensional

Ejemplo Guiado: Diseño Óptimo

Función: $f(x, y) = x^2 + y^2 + xy$

Paso 1: Seleccionar opción

Seleccione una opción: 2

Paso 2: Ingresar función

Ingrese la función f(x, y): (ej. x**2 + y**2 + x*y): x**2 + y**2 + x*y

Importante:

- Usar `x` e `y` como variables
- Separar con `*`: xy = `x*y`

Paso 3: Valores iniciales

Ingrese valores iniciales separados por coma (ej. 1,2): 1,1

Consejos para valores iniciales:

- Usa estimaciones razonables del óptimo
- Si no sabes, prueba [1, 1] o [0, 0]
- Si el resultado no tiene sentido, cambia los valores iniciales

Paso 4: Ver resultado

Resultado: f(x,y) mínima = 0.0000 en punto [0. 0.]

5.3 Optimización con Restricciones

Ejemplo Guiado: Producción con Presupuesto

Función: $f(x, y) = x^2 + y^2$ **Restricción:** $x + y = 10$

Paso 1: Seleccionar opción

Seleccione una opción: 3

Paso 2: Ingresar función objetivo

Ingrese la función f(x, y): (ej. x**2 + y**2): x**2 + y**2

Paso 3: Ingresar restricción

```
Restricción g(x, y)=0 (ej. x + y - 1): x + y - 10
```

IMPORTANTE: La restricción debe ser igual a cero

- Si tienes: x + y = 10
- Ingresa: x + y - 10

Paso 4: Valores iniciales

```
Ingrese valores iniciales separados por coma (ej. 0.5,0.5): 5,5
```

Consejo: Usa valores que cumplan (o estén cerca de cumplir) la restricción

Paso 5: Ver resultado

```
Resultado: f(x,y) mínima = 50.0000 en punto [5. 5.]
```

Verificación de la restricción:

```
g(5, 5) = 5 + 5 - 10 = 0 ✓
```

---

6. SINTAXIS Y EXPRESIONES

Operadores Básicos

Operación	Símbolo	Ejemplo	Resultado
Suma	+	x + 3	x más 3
Resta	-	x - 2	x menos 2
Multiplicación	*	3*x	3 por x
División	/	x/2	x dividido 2
Potencia	**	x**2	x al cuadrado
Paréntesis	()	(x+1)**2	(x+1) al cuadrado

Funciones Matemáticas

Todas las funciones deben usar el prefijo `np.`:

Función	Sintaxis	Ejemplo
Seno	np.sin()	np.sin(x)
Coseno	np.cos()	np.cos(x)
Tangente	np.tan()	np.tan(x)
Exponencial	np.exp()	np.exp(x)
Logaritmo natural	np.log()	np.log(x)
Raíz cuadrada	np.sqrt()	np.sqrt(x)

Función	Sintaxis	Ejemplo
Valor absoluto	np.abs()	np.abs(x)

Ejemplos de Funciones Válidas

Unidimensional (una variable: x)

✓ x**2 + 3*x + 2
✓ np.sin(x) + x**2
✓ np.exp(-x**2)
✓ x**3 - 6*x**2 + 9*x
✓ np.log(x) + 1/x

Multidimensional (variables: x, y)

✓ x**2 + y**2
✓ x**2 + y**2 + x*y
✓ (x-2)**2 + (y-3)**2
✓ np.sin(x) + np.cos(y)
✓ x**2*y + y**2*x

7. INTERPRETACIÓN DE RESULTADOS

7.1 Resultado de Optimización Unidimensional

Ejemplo de salida:

```
Resultado: f(x) mínima = 1.0000 en x = 3.0000
```

Componentes:

- **f(x) mínima = 1.0000:** Valor mínimo de la función
- **x = 3.0000:** Punto donde se alcanza el mínimo

Cómo verificar:

1. Sustituir x = 3 en la función original
2. Comprobar que da f(3) = 1
3. Probar valores cercanos (x=2.9, x=3.1) y verificar que f(x) > 1

7.2 Resultado de Optimización Multidimensional

Ejemplo de salida:

```
Resultado: f(x,y) mínima = 0.0000 en punto [2. 3.]
```

Componentes:

- **f(x,y) mínima = 0.0000:** Valor mínimo
- **punto [2. 3.]:** x=2, y=3

Formato del punto:

```
[x_óptimo  y_óptimo]
```

7.3 Resultado con Restricciones

Ejemplo de salida:

```
Resultado: f(x,y) mínima = 50.0000 en punto [5. 5.]
```

Verificación adicional necesaria:

1. Verificar función objetivo: f(5, 5) = ?
2. **Verificar restricción:** g(5, 5) = 0 ✓

Ejemplo de verificación: Si la restricción era `x + y - 10`:

```
g(5, 5) = 5 + 5 - 10 = 0 ✓ Cumple!
```

---

8. CASOS DE USO PRÁCTICOS

Caso 1: Precio Óptimo de Producto

Situación: Una tienda quiere maximizar ingresos. La demanda depende del precio:

- Demanda = 200 - 2×precio
- Ingresos = precio × demanda

Función a minimizar (para maximizar ingresos):

```
f(x) = -(x * (200 - 2*x))
     = -200*x + 2*x**2
```

En el programa:

1. Opción: 1 (Unidimensional)
2. Función: `-200*x + 2*x**2`
3. Intervalo: [0, 100]

Resultado esperado: x = 50, ingresos máximos = 5000

Caso 2: Diseño de Caja con Área Mínima

Situación: Diseñar una caja rectangular (x × y) que contenga 100 unidades cúbicas usando mínimo material.

Sin altura definida, área aproximada:

```
f(x, y) = x*y + 200/x + 200/y
```

En el programa:

1. Opción: 2 (Multidimensional)
2. Función: `x*y + 200/x + 200/y`
3. Valores iniciales: [10, 10]

Caso 3: Asignación de Recursos con Presupuesto

Situación: Asignar recursos a dos proyectos (x, y) para minimizar riesgo total:

- Riesgo = $x^2 + 2y^2$
- Presupuesto total: x + y = 30

En el programa:

1. Opción: 3 (Con Restricciones)
2. Función: `x**2 + 2*y**2`
3. Restricción: `x + y - 30`
4. Valores iniciales: [15, 15]

Versión: 1.0
Fecha: Octubre 2025
Compatibilidad: Python 3.7+, NumPy 1.19+, SciPy 1.5+