Instituto Politécnico Nacional
Escuela Superior de Cómputo

Algoritmos Bioinspirados
Práctica 2: Optimización usando herramientas en Python

Alumno: Luis Fernando Rodríguez Domínguez
Profesor: Jorge Luis Rosas Trigueros
Fecha de realización: 10 de marzo de 2025
Fecha de entrega: 10 de marzo de 2025

1. Marco Teórico

En la optimización de funciones multivariables, es fundamental identificar y clasificar los puntos críticos para determinar si corresponden a mínimos, máximos o puntos de silla. Para ello, se emplean dos métodos principales: el análisis de la matriz Hessiana y el método de los multiplicadores de Lagrange.

Análisis mediante la Matriz Hessiana

El método basado en la matriz Hessiana se utiliza para estudiar la curvatura de una función $f(x_1, x_2, \ldots, x_n)$ en sus puntos críticos. El proceso se resume en los siguientes pasos:

1. **Cálculo del Gradiente:** Se determina el gradiente ∇f y se resuelve la ecuación $\nabla f = 0$ para obtener los puntos críticos.

2. **Cálculo de la Matriz Hessiana:** Una vez obtenido un punto crítico, se construye la matriz Hessiana H, definida como:

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

3. **Evaluación y Análisis:** La Hessiana se evalúa en el punto crítico y se analiza la definitud de la misma. Esto se realiza mediante el criterio de los menores principales, que consiste en calcular el determinante de cada submatriz de orden $i \times i$ (para $i = 1, 2, \ldots, n$). Si todos los determinantes son positivos, la función tiene un mínimo local en ese punto.

Este enfoque formaliza la evaluación de la curvatura de f y permite clasificar rigurosamente los puntos críticos obtenidos.

Método de los Multiplicadores de Lagrange

El método de los multiplicadores de Lagrange es una técnica para encontrar los extremos de una función $f(x_1, x_2, \ldots, x_n)$ sujeta a restricciones, generalmente de la forma $g(x_1, x_2, \ldots, x_n) = c$. Para incorporar la restricción en el proceso de optimización, se define el **Lagrangiano**:

$$L(x_1, x_2, \ldots, x_n, \lambda) = f(x_1, x_2, \ldots, x_n) - \lambda \left(g(x_1, x_2, \ldots, x_n) - c\right),$$

donde λ es el multiplicador de Lagrange. El procedimiento consiste en:

1. Derivar el Lagrangiano con respecto a cada variable x_i y al multiplicador λ, obteniendo el sistema:

$$\frac{\partial L}{\partial x_i} = 0 \quad \text{para } i = 1, \ldots, n \quad \text{y} \quad \frac{\partial L}{\partial \lambda} = 0.$$

2. Resolver el sistema de ecuaciones resultante para encontrar los puntos críticos que cumplen la restricción.

Este método integra de manera elegante la restricción en el proceso de optimización, permitiendo encontrar soluciones que, de otro modo, serían difíciles de obtener mediante métodos directos.

Ambos métodos son esenciales en la optimización y se aplican en diversas áreas de la ingeniería y las ciencias. Su implementación en Python, utilizando librerías como `Sympy` para cálculos simbólicos, `NumPy` para operaciones numéricas y `Matplotlib` para la visualización, facilita la automatización y el análisis riguroso de problemas complejos.

2. Material y Equipo

- **Hardware:** Computadora personal (se optó por trabajar en un entorno local para tener control total sobre la configuración y las versiones de las librerías, en contraste con el uso de plataformas colaborativas como Google Colab).
- **Software:**
  - Python 3.9.14.
  - `NumPy` para operaciones numéricas.
  - `Matplotlib` para la representación gráfica.
  - `Sympy` para cálculos simbólicos.
  - Editor de código LaTeX (Overleaf) para la elaboración del reporte.

3. Desarrollo de la Práctica

La práctica se divide en dos grandes secciones, cada una de las cuales se desarrolló mediante la abstracción de funciones que permiten el cálculo y análisis de problemas de optimización de forma automatizada. En ambas secciones se generaron logs en consola (que ofrecen información relevante sobre el proceso y los resultados) y se realizaron gráficos que visualizan el comportamiento de las funciones analizadas. A continuación se detalla el desarrollo de cada parte, incluyendo fragmentos de código relevantes.

1. Optimización mediante el Método de la Matriz Hessiana

En esta sección se implementaron funciones para:

- Calcular el gradiente de una función simbólica.
- Calcular la matriz Hessiana, es decir, las segundas derivadas parciales.

- Evaluar la Hessiana en cada punto crítico (donde $\nabla f = 0$) y analizar los determinantes de las submatrices $i \times i$ para determinar la naturaleza de cada punto (mínimo local, máximo o punto de silla).

Se resolvieron tres problemas distintos:

Problema 1: Se analizó la función

$$f(x, y, z) = x^2 + y^2 + 7z^2 - xy.$$

Fragmento de código:

```
import sympy as sp

# Definir variables y funcion simbolica
x, y, z = sp.symbols('x y z')
f_sym = x**2 + y**2 + 7*z**2 - x*y
vars_list = [x, y, z]

# Calcular el gradiente y resolver gradiente = 0
grad_f = [sp.diff(f_sym, var) for var in vars_list]
crit_solutions = sp.solve(grad_f, vars_list, dict=True)
sp.pprint(crit_solutions)

# Calcular la matriz Hessiana
Hessian = sp.Matrix([[sp.diff(f_sym, var1, var2) for var1 in vars_list]
    for var2 in vars_list])
print("Matriz Hessiana simblica:")
sp.pprint(Hessian)
```

Listing 1: Cálculo del gradiente y la matriz Hessiana para f(x,y,z)

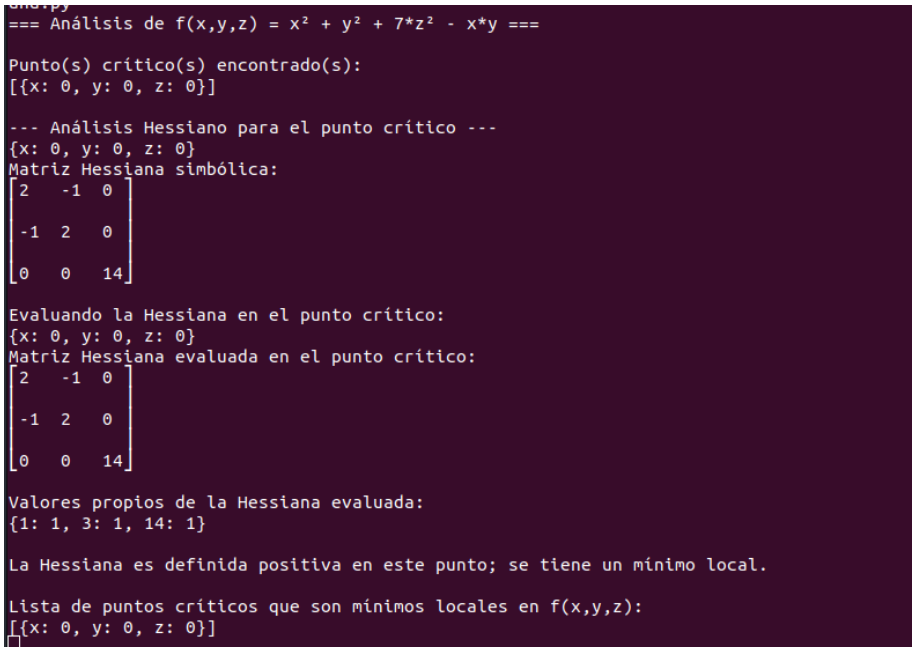

Figura 1: Resultado problema 1 hessiana

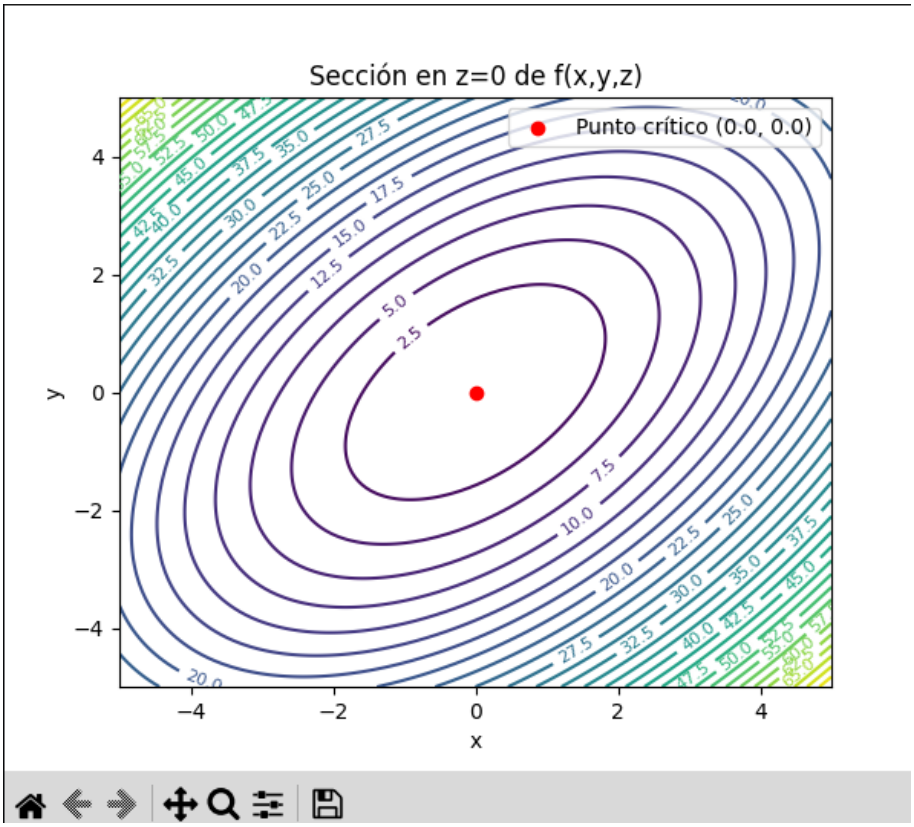

Figura 2: Gráfica con sección z=0 de la solución del problema 1

Problema 2: Se estudió la función

$$g(r, s, t, u) = r^2 (s + 1)^2 + t^2 (u - 1)^4.$$

Se siguió el mismo procedimiento: cálculo del gradiente, obtención de los puntos críticos, y evaluación de la matriz Hessiana en cada uno de ellos para determinar la definitud mediante los determinantes de las submatrices.

```
# Definir variables y funcion simbolica
r, s, t, u = sp.symbols('r s t u')
g_sym = r**2 * (s+1)**2 + t**2 * (u-1)**4
vars_list = [r, s, t, u]

# Calcular gradiente y puntos criticos
grad_g = [sp.diff(g_sym, var) for var in vars_list]
crit_solutions_g = sp.solve(grad_g, vars_list, dict=True)
sp.pprint(crit_solutions_g)

# Calcular la Hessiana y evaluarla en los puntos criticos
Hessian_g = sp.Matrix([[sp.diff(g_sym, var1, var2) for var1 in vars_list
    ] for var2 in vars_list])
print("Matriz Hessiana para g(r,s,t,u):")
sp.pprint(Hessian_g)
```

Listing 2: Cálculo del gradiente y Hessiana para g(r,s,t,u)

Para este problema, al tratarse de $\mathbb{R}^4$, se tuvo que extraer secciones en $\mathbb{R}^2$ para poder visualizar un poco sobre el problema.

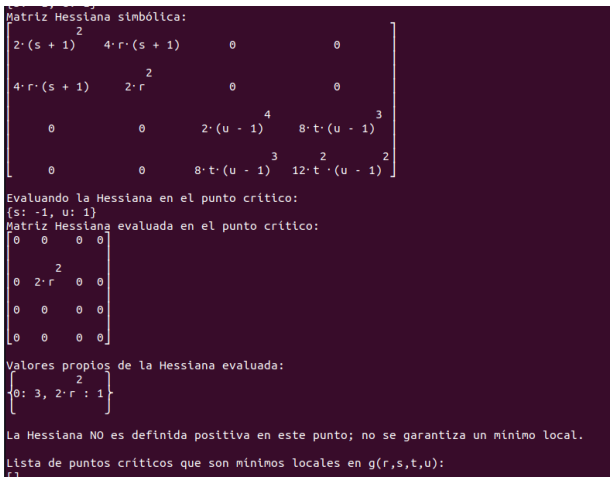

Figura 3: Resultados problema 2 hessiana

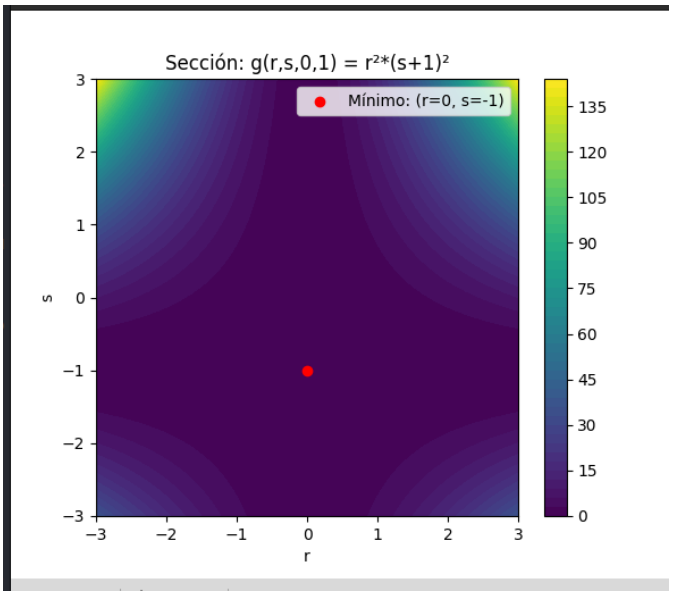

Figura 4: Gráfica de sección g(r,s,0,1) del problema 2 de hessiana

Problema 3 (Extra): Se analizó la función

$$f(x, y) = x^3 + 2y^3 - xy.$$

Además del procedimiento anterior, se generó una gráfica 3D de la superficie y se resaltó el mínimo local.

```
# Definir variables y funcion simbolica
x, y = sp.symbols('x␣y')
f_extra = x**3 + 2*y**3 - x*y
vars_list = [x, y]

# Calcular gradiente y puntos criticos
grad_f_extra = [sp.diff(f_extra, var) for var in vars_list]
crit_solutions_extra = sp.solve(grad_f_extra, vars_list, dict=True)
sp.pprint(crit_solutions_extra)

# Calcular la Hessiana
Hessian_extra = sp.Matrix([[sp.diff(f_extra, var1, var2) for var1 in
    vars_list] for var2 in vars_list])
```

```
print("Matriz Hessiana para f(x,y):")
sp.pprint(Hessian_extra)
```

Listing 3: Análisis de f(x,y) y visualización 3D

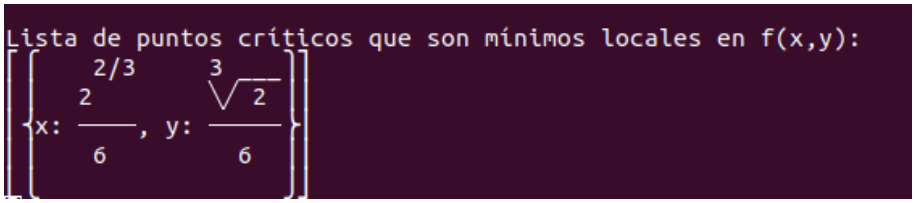

Figura 5: Resultados problema de ejemplo hessiana

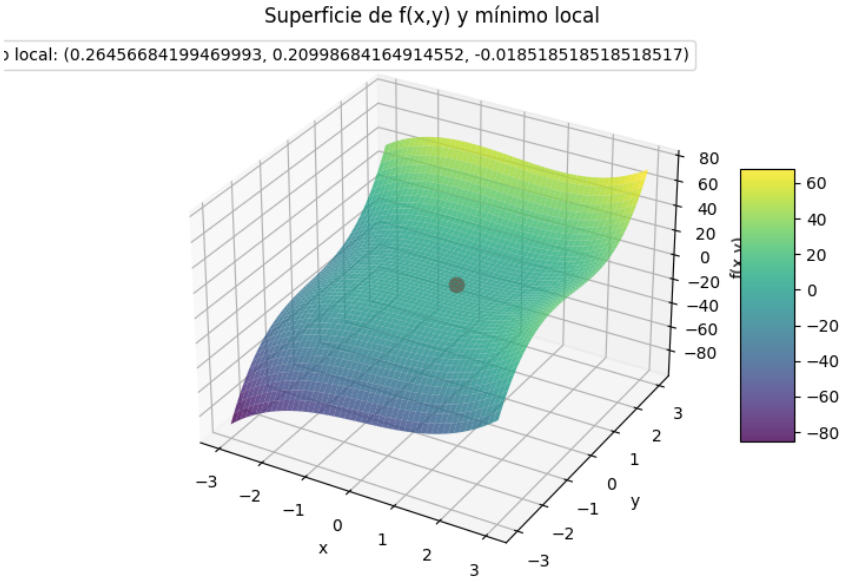

Figura 6: Gráfica del plano y solución del problema de ejemplo en clase de matriz hessiana

2. Optimización mediante el Método de los Multiplicadores de Lagrange

Se implementó una función abstracta que genera el Lagrangiano y resuelve el sistema de ecuaciones para encontrar los puntos críticos en problemas con restricciones. Se abordaron tres casos:

Problema 2.a: Minimización de la función

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

sujeta a la restricción $x^2 + y^2 = 9$. Se resolvió el sistema derivado del Lagrangiano y se determinó la solución óptima.

```python
def solve_lagrange(function, constraint, variables, lam):
    g_expr = constraint.lhs - constraint.rhs
    L = function - lam * g_expr
    eqs = [sp.diff(L, var) for var in variables] + [sp.diff(L, lam)]
    solutions = sp.solve(eqs, variables + [lam], dict=True)
    return solutions

# Definir variables y funcion
x, y, lam = sp.symbols('x y lam')
phi = (x - 2)**2 + (y - 2)**2
constraint = sp.Eq(x**2 + y**2, 9)

# Resolver el problema
solutions_2a = solve_lagrange(phi, constraint, [x, y], lam)
sp.pprint(solutions_2a)
```

Listing 4: Resolución del problema 2.a con multiplicadores de Lagrange

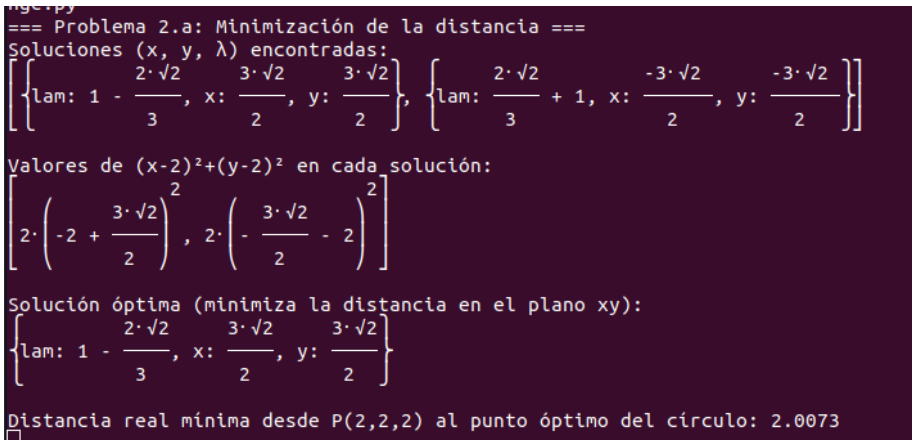

Figura 7: Resultados por consola problema 2a, lagrangiano

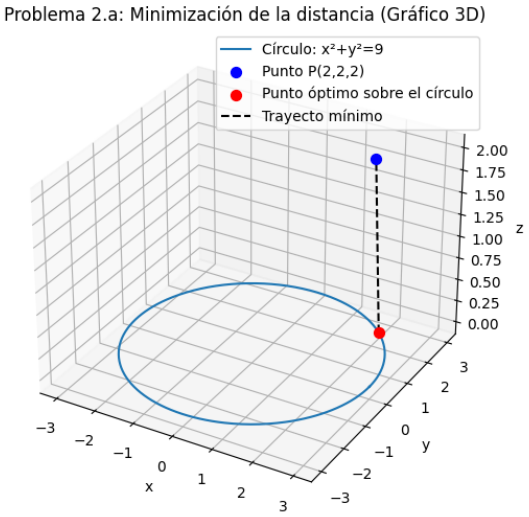

Figura 8: Gráfica de la solución del problema 2a

Problema 2.b: Optimización de la función

$$f(x, y) = x + y,$$

sobre el círculo unidad $x^2+y^2 = 1$. Se identificaron los puntos que maximizan y minimizan la función.

```
x, y, lam = sp.symbols('x y lam')
f_xy = x + y
constraint_unit = sp.Eq(x**2 + y**2, 1)

# Resolver el problema
solutions_2b = solve_lagrange(f_xy, constraint_unit, [x, y], lam)
sp.pprint(solutions_2b)
```

Listing 5: Resolución del problema 2.b con multiplicadores de Lagrange

```
=== Problema 2.b: Optimización de f(x,y)=x+y ===
Soluciones (x, y, λ) encontradas:
⎡⎧      -√2       -√2       -√2⎫  ⎧     √2     √2     √2⎫⎤
⎣⎨lam: ────, x: ────, y: ────⎬, ⎨lam: ──, x: ──, y: ──⎬⎦
⎩       2         2         2 ⎭  ⎩     2      2      2 ⎭

Valores de f(x,y) en cada solución:
[-√2, √2]

Solución que maximiza f(x,y):
⎧     √2     √2     √2⎫
⎨lam: ──, x: ──, y: ──⎬
⎩     2      2      2 ⎭

Solución que minimiza f(x,y):
⎧      -√2       -√2       -√2⎫
⎨lam: ────, x: ────, y: ────⎬
⎩       2         2         2 ⎭
```

Figura 9: Resultado por consola de problema 2b, lagrangiano

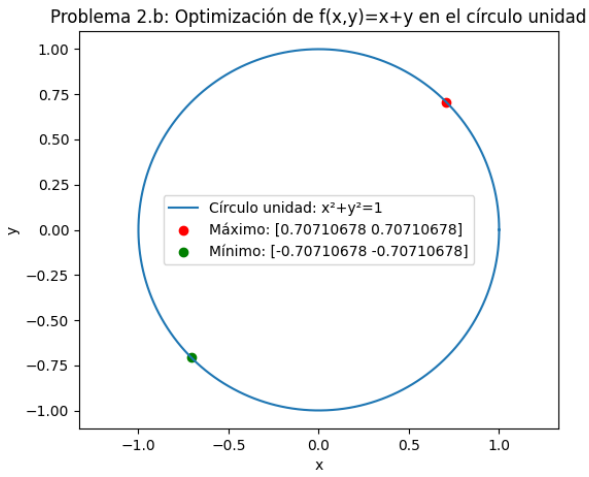

Figura 10: Gráfica solución problema 2b, lagrangiano

Problema Adicional: Optimización de la función

$$f(x, y) = 9 - x^2 - y^2,$$

sujeta a la restricción $x+y = 3$. Se resolvió mediante el método de Lagrange y se visualizó la superficie, la línea de restricción y el punto óptimo en una gráfica 3D.

```
x, y, lam = sp.symbols('x␣y␣lam')
f_xy_ad = 9 - x**2 - y**2
constraint_line = sp.Eq(x + y, 3)

# Resolver el problema
solutions_extra = solve_lagrange(f_xy_ad, constraint_line, [x, y], lam)
sp.pprint(solutions_extra)
```

Listing 6: Resolución del problema adicional con multiplicadores de Lagrange

```
=== Problema Adicional: Optimización de f(x,y)=9 - x² - y² ===
Soluciones (x, y, λ) encontradas:
[{lam: -3, x: 3/2, y: 3/2}]

Valores de f(x,y) en cada solución:
[9/2]

Solución óptima:
{lam: -3, x: 3/2, y: 3/2}
```

Figura 11: Resultado por consola del problema visto en clase, lagrangiano

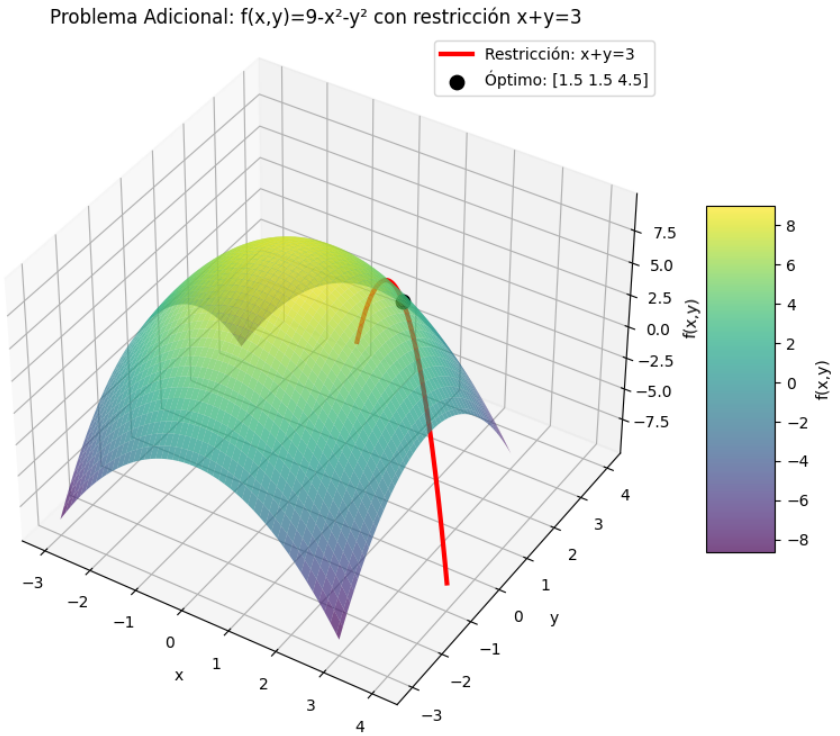

Figura 12: Gráfica del plano, restricción y resultado del problema visto en clase sobre lagrangiano

Listado de Imágenes Utilizadas

- **Imagen 1:** Salida en consola del análisis hessiano del Problema 1.
- **Imagen 2:** Gráfica de la sección en $z = 0$ de $f(x, y, z)$ con el punto crítico resaltado.
- **Imagen 3:** Salida en consola del análisis hessiano del Problema 2.
- **Imagen 4:** Gráfica representativa de una sección de $g(r, s, t, u)$.
- **Imagen 5:** Salida en consola del análisis del Problema 3 (Extra) en $f(x, y)$.
- **Imagen 6:** Gráfica 3D de la superficie $f(x, y)$ con el mínimo local resaltado.
- **Imagen 7:** Salida en consola del Problema 2.a (Lagrangiano) mostrando las soluciones y el valor óptimo.
- **Imagen 8:** Gráfico 3D del círculo $x^2 + y^2 = 9$, el punto $P(2, 2, 2)$ y el punto óptimo en $z = 0$.
- **Imagen 9:** Salida en consola del Problema 2.b (Lagrangiano) con las soluciones y valores de $f(x, y)$.
- **Imagen 10:** Gráfica 2D del círculo unidad $x^2 + y^2 = 1$ con los puntos extremos.
- **Imagen 11:** Salida en consola del Problema Adicional (Lagrangiano) mostrando la solución óptima.
- **Imagen 12:** Gráfica 3D de la superficie $9 - x^2 - y^2$ con la restricción $x + y = 3$ y el punto óptimo resaltado.

4. Conclusiones y Recomendaciones

- La práctica permitió afianzar el uso de herramientas de Python para resolver problemas de optimización que antes se abordaban manualmente.
- El análisis de la matriz Hessiana y el uso de multiplicadores de Lagrange resultan métodos efectivos para identificar puntos críticos y resolver restricciones en problemas de optimización.
- La integración de `Sympy` para cálculos simbólicos con `NumPy` y `Matplotlib` para la visualización proporciona un entorno robusto para el estudio y la validación de resultados.
- Se recomienda profundizar en estos métodos y explorar aplicaciones en problemas más complejos, lo que puede ampliar el conocimiento en áreas como algoritmos bioinspirados e inteligencia artificial.

Referencias

[1] NumPy Documentation, *NumPy Developers*. [En línea]. Disponible: `https://numpy.org/doc/`.

[2] Matplotlib Documentation, *Matplotlib Developers*. [En línea]. Disponible: `https://matplotlib.org/stable/contents.html`.

[3] Sympy Documentation, *Sympy Developers*. [En línea]. Disponible: `https://docs.sympy.org/latest/index.html`.

[4] Python Official Documentation, *Python Software Foundation*. [En línea]. Disponible: `https://docs.python.org/3/`.