[EEC3600-001] 수치해석		
소속: 전기전자공학부	학번: 12191529	이름: 장준영
Python Programming for Numerical Analysis		HW number: #7

7.1. Optimization with Constraints (1)

a. 문제

Problem 7.1 Minimize the function

$$F(x, y) = (x-1)^2 + (y-1)^2$$

subject to the constraints $x + y \geq 1$ and $x \geq 0.6$.

b. 풀이

sol) 만약 제약조건이 없다면: $(x_{op}, y_{op}) = (1, 1)$

이제 (1,1)이 제약조건을 만족하는지 살펴보자.

① $x+y \geq 1$: $1+1 = 2 \geq 1$ (만족)

② $x \geq 0.6$: $1 \geq 0.6$ (만족)

$\boxed{\text{답: } F(1,1) = 0}$

7.2. **Optimization with Constraints (2)**

a. 문제

Problem 7.2 Find the minimum of the function

$$F(x, y) = 6x^2 + y^3 + xy$$

in $y \geq 0$. Verify the result analytically.

b. 풀이

sol) $\min_{y \geq 0} F(x,y) = ??$

① (편미분) = 0 찾기.

대입

- $\frac{\partial F}{\partial x} = 12x + y = 0$; $y = -12x$
- $\frac{\partial F}{\partial y} = 3y^2 + x = 0$; $3(144x^2) + x = 0$; $x(432x+1) = 0$

∴ 가능한 해 $\begin{cases} x = 0 \rightarrow y = 0 \\ x = -\frac{1}{432} \rightarrow y = \cancel{+}\frac{\cancel{+12}}{\cancel{12}\cdot 12 \cdot 3} = \frac{1}{36} \end{cases}$ → 두 케이스 모두 제약조건 $(y \geq 0)$ 만족!

② 두 점에서 함수값 비교

1) $(0,0)$: $F(0,0) = 0$

2) $\left(-\frac{1}{432}, \frac{1}{36}\right)$: $F\left(-\frac{1}{432}, \frac{1}{36}\right) = 6\left(-\frac{1}{432}\right)^2 + \left(\frac{1}{36}\right)^3 + \left(-\frac{1}{432}\right)\left(\frac{1}{36}\right) = -\frac{1}{93312}$

∴ 제약 조건을 만족하면서 최소가 되는 케이스는

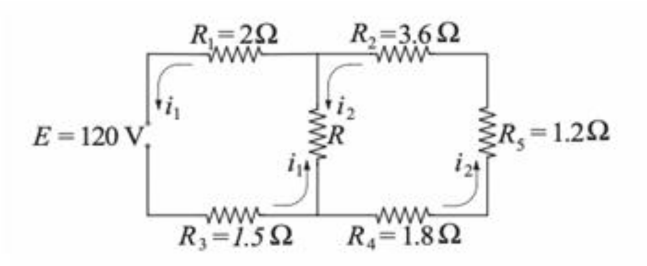
$F\left(-\frac{1}{432}, \frac{1}{36}\right) = -\frac{1}{93312}$

$\boxed{\text{답}: -\frac{1}{93312}}$

7.3. Optimization with Constraints (3)

a. 문제

Problem 7.3



Kirchoff's equations for the two loops of the electrical circuit are

$$R_1 i_1 + R_3 i_1 + R(i_1 - i_2) = E$$
$$R_2 i_2 + R_4 i_2 + R_5 i_2 + R(i_2 - i_1) = 0$$

3.6 1.8 1.2

Find the resistance R that maximizes the power dissipated by R.

b. 풀이

Sol) 목표: 저항 R에서 소모되는 전력 $P = R(i_1 - i_2)^2$을 최대화하자.

① 식 정리: 각 값들을 대입하면,

- $3.5 i_1 + R(i_1 - i_2) = 120$
- $6.6 i_2 + R(i_2 - i_1) = 0$

② i_1을 i_2에 대한 식으로 표현:

- $(6.6+R) i_2 = R i_1$; $i_1 = \frac{6.6+R}{R} i_2 \cdots (i)$
- $(3.5+R)\left(\frac{6.6+R}{R} i_2\right) - R i_2 = 120$; $\left\{(3.5+R)(6.6+R) - R^2\right\} \cdot \frac{i_2}{R} = 120$

  ; $i_2 = \frac{120R}{(3.5+R)(6.6+R)} \cdots (ii)$

③ 전력 함수 $P = (i_1 - i_2)^2 R$

$\Rightarrow P = R(i_1 - i_2)^2 = R\left\{\left(\frac{6.6+R}{R} - 1\right) i_2\right\}^2$

$= \frac{43.56}{R} i_2^2 \quad (R > 0 \ \& \ (ii))$

(수치적 최적화 과정은 Python 코드 참고!)

답: $R = 2.2871(\Omega)$ 일 때

$P = 672.14(W)$

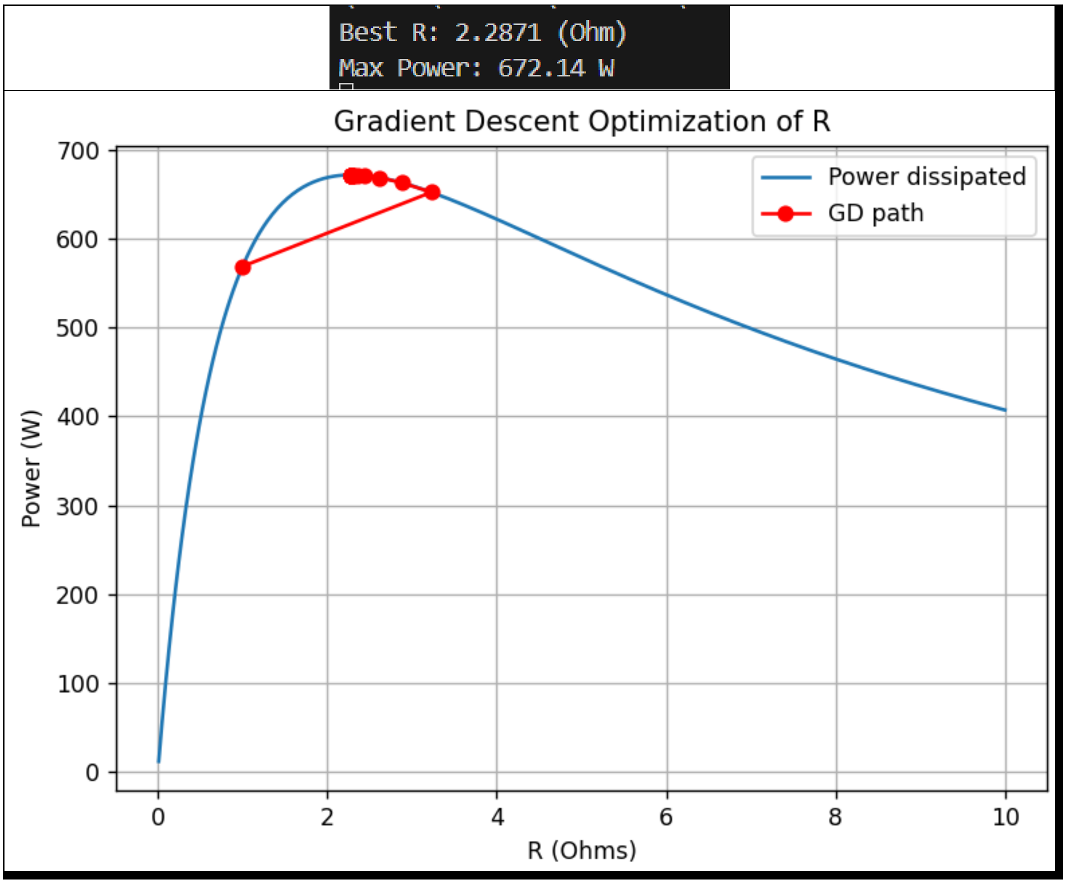
c. Python Code

```python
import numpy as np
import matplotlib.pyplot as plt

# 목적 함수: f(R) = -P(R)
def f(R):
    if R <= 0:
        return 1e6  # 정의역 밖
    num = 120 * R
    denom = (3.5 + R) * (6.6 + R) - R**2
    i2 = num / denom
    return - (43.56 / R) * i2**2

# 수치 미분을 이용한 gradient 계산
def grad_f(R, h=1e-5):
    return (f(R + h) - f(R - h)) / (2 * h)

# Line search descent method
def gradient_descent(initial_R, alpha=0.01, tol=1e-6, max_iter=1000):
    R = initial_R
    history = [R]
    for i in range(max_iter):
        g = grad_f(R)
        if abs(g) < tol:
            break
        R = R - alpha * g
        history.append(R)
    return R, -f(R), history  # 최소화한 -f는 실제 최대값

# 실행
optimal_R, max_power, trajectory = gradient_descent(initial_R=1.0)
print(f"Best R: {optimal_R:.4f} (Ohm)")
print(f"Max Power: {max_power:.2f} W")
# 경로 시각화
R_vals = np.linspace(0.01, 10, 500)
P_vals = [-(f(R)) for R in R_vals]

plt.plot(R_vals, P_vals, label="Power dissipated")
plt.plot(trajectory, [-(f(R)) for R in trajectory], 'ro-', label="GD path")
plt.xlabel("R (Ohms)")
plt.ylabel("Power (W)")
plt.grid(True)
plt.title("Gradient Descent Optimization of R")
plt.legend()
plt.tight_layout()
plt.show()
```

■ **Gradient Descent 반복**

- 초기값 R_0설정
- 방향 벡터 $d_n = -\nabla f(R_n)$
- 적절한 step size α를 선택 (간단히 고정값 사용)
- 다음 점 $R_{n+1} = R_n + \alpha d_n$
- 수렴 조건(gradient norm < ε)이 만족될 때까지 반복

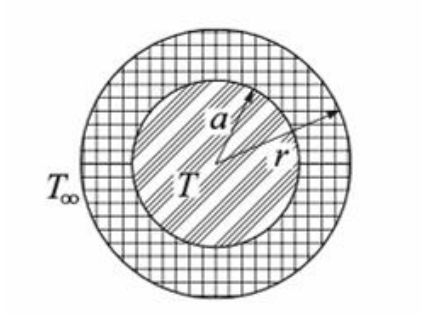
d. Python Code 실행 결과



7.4. Optimization with Constraints (4)

a. 문제

Problem 7.4



A wire carrying an electric current is surrounded by rubber insulation of outer radius r. The resistance of the wire generates heat, which is conducted through the insulation and convected into the surrounding air. The temperature of the wire can be shown to be

$$T = \frac{q}{2\pi}\left(\frac{\ln(r/a)}{k} + \frac{1}{hr}\right) + T_\infty$$

where

$q =$ rate of heat generation in wire $= 50$ W/m

$a =$ radius of wire $= 5$ mm

$k =$ thermal conductivity of rubber $= 0.16$ W/m · K

$h =$ convective heat-transfer coefficient $= 20$ W/m^2 · K

$T_\infty =$ ambient temperature $= 280$ K

Find r that minimizes T.

b. 풀이

sol) $\min_{r>a} T(r)$ 을 찾자!

• 예상되는 값 : $r = \frac{k}{h} = 0.008$

(수치 해 알고리즘을 사용하여 결과를 찾는 과정은 Python 코드 참고!)

답: $r = 8.001$ (mm) 일 때

$T_{min} = 353.1120$ (K)

c. Python Code

```python
import numpy as np
import matplotlib.pyplot as plt

# Constants for T(r)
q = 50
a = 0.005
k = 0.16
h = 20
T_inf = 280

# Objective function
def T(r):
    if r <= a:
        return 1e6  # invalid
    return (q / (2 * np.pi)) * (np.log(r / a) / k + 1 / (h * r)) + T_inf
# Brent's Method (simplified and pure)
def brent(f, a, b, tol=1e-5, max_iter=100):
    gr = (3 - np.sqrt(5)) / 2  # golden ratio factor

    x = w = v = a + gr * (b - a)
    fx = fw = fv = f(x)
    d = e = b - a

    for _ in range(max_iter):
        m = 0.5 * (a + b)
        tol1 = tol * abs(x) + 1e-10
        tol2 = 2 * tol1

        # Check convergence
        if abs(x - m) <= tol2 - 0.5 * (b - a):
            return x, f(x)

        # Parabolic fit condition
        r = q = p = 0
        if x != w and x != v and w != v:
            # Inverse quadratic interpolation
            r = (x - w) * (fx - fv)
            q = (x - v) * (fx - fw)
            p = (x - v) * q - (x - w) * r
            q = 2 * (q - r)
            if q != 0:
                p /= q
            u = x + p
        else:
            u = None  # fallback to golden
        # Accept interpolation step?
        accept_interp = (
            u is not None and
            a + tol1 <= u <= b - tol1 and
            abs(u - x) < e / 2
        )

        if accept_interp:
            d = abs(u - x)
        else:
            # Golden-section step
            if x < m:
                u = x + gr * (b - x)
                e = b - x
            else:
                u = x - gr * (x - a)
                e = x - a
            d = abs(u - x)

        fu = f(u)
```

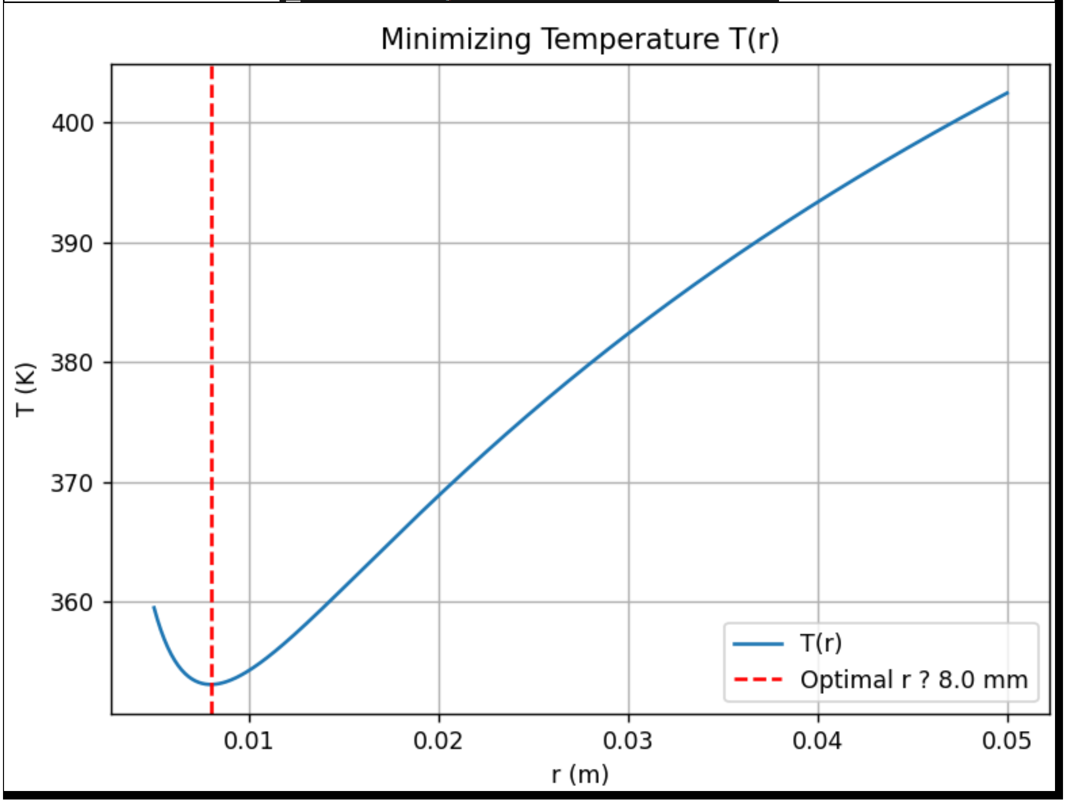
```python
        # Update a, b, x, w, v
        if fu <= fx:
            if u < x:
                b = x
            else:
                a = x
            v, fv = w, fw
            w, fw = x, fx
            x, fx = u, fu
        else:
            if u < x:
                a = u
            else:
                b = u
            if fu <= fw or w == x:
                v, fv = w, fw
                w, fw = u, fu
            elif fu <= fv or v == x or v == w:
                v, fv = u, fu

    raise RuntimeError("Brent's method did not converge.")

# Run Brent's method
r_min, T_min = brent(T, a + 1e-6, 0.05)
print(f"Brent's Method (manual):")
print(f"Optimal r = {r_min:.6f} m")
print(f"Minimum T = {T_min:.4f} K")
# Plot T(r)
r_vals = np.linspace(0.00501, 0.05, 300)
T_vals = [T(r) for r in r_vals]
plt.plot(r_vals, T_vals, label="T(r)")
plt.axvline(r_min, color='r', linestyle='--', label=f"Optimal r ? {r_min:.4f}")
plt.xlabel("r (m)")
plt.ylabel("T (K)")
plt.title("Brent's Method (manual) for Minimizing T(r)")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

■ Brent's Method

- 초기 구간 $[r_{min}, r_{max}]$ 설정
- 함수 $T(r)$의 세 점을 선택하고, 포물선을 근사하여 극값 추정.
  만약 포물선 근사 실패 시, Golden section 방법으로 안전하게 탐색.
- 수렴 조건: 양 끝 구간이 충분히 가까워질 때까지 반복.
  또는 (함수값 변화량) < ε일 때 중단
- 최적 해 반환:
  - $r^* = argmin\, T(r)$
  - $T(r^*)$: 최소 온도 값.

d. Python 출력 결과

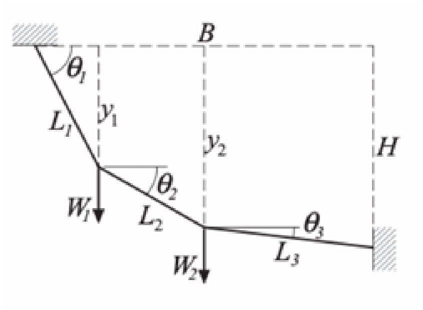
```
Optimal r: 8.001 mm
Minimum Temperature: 353.1120 K
```



7.5. Prediction via Polynomial Interploation

a. 문제

Problem 7.5



A cable supported at the ends carries the weights W_1 and W_2. The potential energy of the system is

$$\begin{aligned} V &= -W_1 y_1 - W_2 y_2 \\ &= -W_1 L_1 \sin\theta_1 - W_2 (L_1 \sin\theta_1 + L_2 \sin\theta_2) \end{aligned}$$

and the geometric constraints are

$$\begin{aligned} L_1\cos\theta_1 + L_2\cos\theta_2 + L_3\cos\theta_3 = B \\ L_1\sin\theta_1 + L_2\sin\theta_2 + L_3\sin\theta_3 = H \end{aligned}$$

The principle of minimum potential energy states that the equilibrium configuration of the system is the one that satisfies geometric constraints and minimizes the potential energy.

Determine the equilibrium values of θ_1, θ_2, and θ_3 given that $L_1 = 1.2$ m, $L_2 = 1.5$ m, $L_3 = 1.0$ m, $B = 3.5$ m, $H = 0$, $W_1 = 20$kN, and $W_2 = 30$kN.

b. 풀이

sol) ① θ_3 제거하기: 제약조건에서

- $\cos\theta_3 = \dfrac{B - L_1\cos\theta_1 - L_2\cos\theta_2}{L_3}$
- $\sin\theta_3 = \dfrac{-L_1\sin\theta_1 - L_2\sin\theta_2}{L_3}$

$\Rightarrow$ $\therefore \cos^2\theta_3 + \sin^2\theta_3 = 1$

; $\left(\dfrac{B - L_1\cos\theta_1 - L_2\cos\theta_2}{\cancel{L_3} 1}\right)^2 + \left(\dfrac{-L_1\sin\theta_1 - L_2\sin\theta_2}{\cancel{L_3} 1}\right)^2 = 1$

; $B^2 - 2B(L_1\cos\theta_1 + L_2\cos\theta_2) + \underbrace{(L_1\cos\theta_1 + L_2\cos\theta_2)^2}_{L_1^2\cos^2\theta_1 + 2L_1L_2\cos\theta_1\cos\theta_2 + L_2\cos^2\theta_2}$
$+ \underbrace{(L_1\sin\theta_1 + L_2\sin\theta_2)^2}_{L_1^2\sin^2\theta_1 + 2L_1L_2\sin\theta_1\sin\theta_2 + L_2^2\sin^2\theta_2} = 1$

; $B^2 - 2B(L_1\cos\theta_1 + L_2\cos\theta_2) + (L_1^2 + L_2^2 + 2L_1L_2\cos(\theta_1 - \theta_2)) = 1$

② 수치 해 찾기 : Python 코드 참고.

답: $\theta_1 = 21.2806°$, $\theta_2 = 1.3478°$, $\theta_3 = -28.0865°$

$\rightarrow V = -22.83457$ (kJ)

c. Python Code

```python
import numpy as np
from scipy.optimize import root_scalar
import matplotlib.pyplot as plt

# --- Constants ---
L1, L2, L3 = 1.2, 1.5, 1.0
W1, W2 = 20000, 30000
B = 3.5
L3_sq = L3**2

# --- θ₁ search range ---
theta1_range_deg = np.linspace(1, 89, 500)
theta1_range_rad = np.radians(theta1_range_deg)

# --- Store θ₂ and V ---
theta2_results = []
V_results = []

# --- Main loop over θ₁ ---
for theta1 in theta1_range_rad:
    def constraint(theta2):
        term1 = B**2 - 2*B*(L1*np.cos(theta1) + L2*np.cos(theta2))
        term2 = L1**2 + L2**2 + 2*L1*L2*np.cos(theta1 - theta2)
        return term1 + term2 - L3_sq

    try:
        sol = root_scalar(constraint, bracket=[np.radians(1), np.radians(89)], method='brentq')
        if sol.converged:
            theta2 = sol.root
            V = -(W1 + W2) * L1 * np.sin(theta1) - W2 * L2 * np.sin(theta2)
            theta2_results.append(np.degrees(theta2))
            V_results.append(V)
        else:
            theta2_results.append(None)
            V_results.append(None)
    except:
        theta2_results.append(None)
        V_results.append(None)

# --- Find optimal index ---
valid_indices = [i for i, v in enumerate(V_results) if v is not None]
min_index = min(valid_indices, key=lambda i: V_results[i])
theta1_opt = theta1_range_rad[min_index]
theta2_opt = np.radians(theta2_results[min_index])
V_min = V_results[min_index]

# --- Compute theta3 ---
cos_theta3 = (B - L1 * np.cos(theta1_opt) - L2 * np.cos(theta2_opt)) / L3
sin_theta3 = (-L1 * np.sin(theta1_opt) - L2 * np.sin(theta2_opt)) / L3
theta3_rad = np.arctan2(sin_theta3, cos_theta3)
theta3_deg = np.degrees(theta3_rad)
# --- Print result ---
print(f"Optimal θ₁: {np.degrees(theta1_opt):.4f}°")
print(f"Optimal θ₂: {np.degrees(theta2_opt):.4f}°")
print(f"Optimal θ₃: {theta3_deg:.4f}°")
print(f"Minimum potential energy: {V_min:.2f} J")

# --- Plot V vs θ₁ ---
plt.figure(figsize=(10, 5))
plt.plot(theta1_range_deg, [v if v is not None else np.nan for v in V_results], label='Potential Energy V(θ₁)')
plt.scatter(np.degrees(theta1_opt), V_min, color='red', label='Minimum')
plt.xlabel("θ₁ (degrees)")
plt.ylabel("Potential Energy (J)")
plt.title("Potential Energy vs θ₁ (θ₂ from constraint)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

■ 알고리즘 단계

- $\theta_1 \in [1°, 89°]$ 범위로 일정 간격 샘플링. 이후 각 θ_1에 대해 θ_2를 결정.

- 다음 등식이 성립하도록 θ_2를 찾음:

$$B^2 - 2B(L_1 cos\theta_1 + L_2 cos\theta_2) + \left({L_1}^2 + {L_2}^2 + 2L_1 L_2 \cos(\theta_1 - \theta_2)\right) - 1 = 0$$

- 위 식을 θ_2에 대한 함수로 정리하고, rood_scalar()로 θ_2를 구함.
- Potential Energy($V(\theta_1, \theta_2)$) 계산.
- 가능한 지점 중 Potential Energy가 가장 작은 지점 선택.
- θ_3 계산.

d. Python 출력 결과

```
Optimal θ₁: 21.2806°
Optimal θ₂: 1.3478°
Optimal θ₃: -28.0865°
Minimum potential energy: -22834.57 J
```