Sprawozdanie - Metody numeryczne i optymailzacja

Jakub Andryszczak 259519,
Jakub Żak 244255,
Maciej Cierpisz 249163

Spis treści

1 Zadanie nr. 1

2 Zadanie nr. 2

3 Zadanie nr. 3

Rozwiąż poniższy układ równań nieliniowych:

$$\begin{cases} 2x_1 = x_2 = exp(-x_1) \\ -x_1 + 2x_2 = exp(-x_2) \end{cases} \tag{1}$$

dla punktu startowego $x_0 = [-55]^t$.
Poniżej kod realizujący zadanie:

```
import numpy as np
from scipy.optimize import fsolve

# Define the system of nonlinear equations
def equations(vars):
    x1, x2 = vars
    eq1 = 2 * x1 - x2 - np.exp(-x1)
    eq2 = -x1 + 2 * x2 - np.exp(-x2)
    return [eq1, eq2]

# Initial guess
x0 = [-5, -5]

# Solve the system of equations
solution = fsolve(equations, x0)

# Print the solution
print("Solution:")
print(f"x1 = {solution[0]}")
print(f"x2 = {solution[1]}")
```

Wyniki:

$$\begin{cases} x_1 = 0.5671432904097838 \\ x_2 = 0.567143290409784 \end{cases} \tag{2}$$

4 Zadanie nr. 4

Znajdź rozwiązanie minimalizujące funkcję celu
$\sum_{k=1}^{10}(2 + 2k - exp(kx_1) - exp(kx_2))^2$
dla punktu początkowego $x_0 = [0.3, 0.4]^T$.
Poniżej kod realizujący zadanie:

```python
import numpy as np
from scipy.optimize import minimize

# Define the objective function
def objective(x):
    x1, x2 = x
    total = 0
    for k in range(1, 11):
        total += (2 + 2*k - np.exp(k*x1) - np.exp(k*x2))**2
    return total

# Initial guess
x0 = [0.3, 0.4]

# Minimize the objective function
result = minimize(objective, x0)

# Print the solution
print("Solution:")
print(f"x1 = {result.x[0]}")
print(f"x2 = {result.x[1]}")
```

Wyniki:

$$\begin{cases} x_1 = 0.25782520984040996 \\ x_2 = 0.2578252098334402 \end{cases} \tag{3}$$

5 Zadanie nr. 5

Tłumienie fal elektromagnetycznych propagowanej w środowisku pozamiejskim w odległości d (w kilometrach) od stacji bazowej może być w przybliżeniu opisane modelem:

$$L = 128.3 + 32.5 \log d + c, \quad [\text{dB}]$$

gdzie $c \sim \mathcal{N}(0, \sigma^2)$ odzwierciedla efekty tłumienia spowodowany przez zjawisko powolnego zanikania (slow fading). Niech $c = 3$ oraz $d = [0.1, 0.2, \ldots, 10]$ [km]. Wykreśl tłumienie L w funkcji odległości d dla wybranych σ. Następnie dopasuj modele:

$$y = \alpha + \beta \log d, \quad \text{(logarytmiczny)}$$

$$y = \alpha + \beta d + \gamma d^2, \quad \text{(kwadratowy)}$$

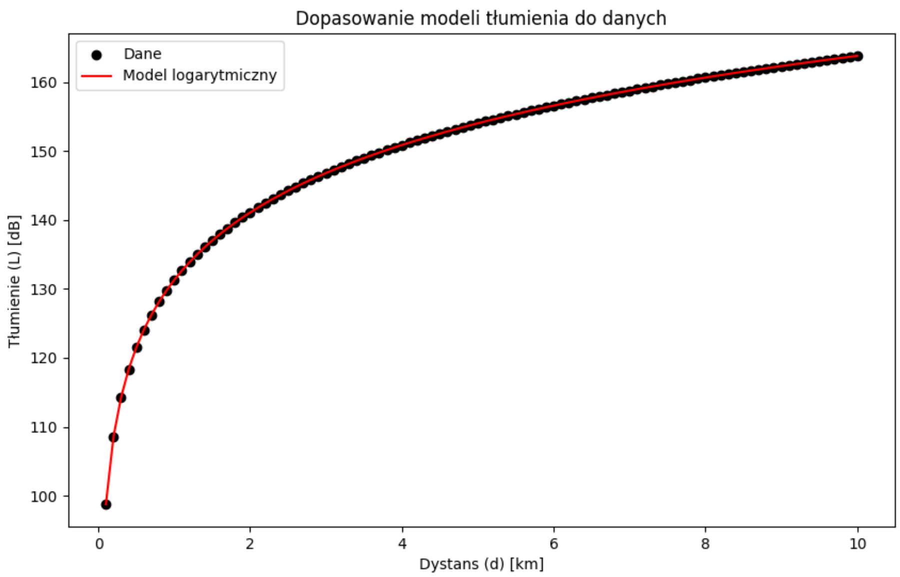
$$y = \alpha + \beta d + \gamma d^2 + \delta d^3, \quad \text{(trzeciego stopnia)}$$

do obserwowanych danych w sensie metryki najmniejszych kwadratów oraz wyznacz parametry $\{\alpha, \beta, \gamma, \delta\}$.

Do wykonania tego zadania posłużono się pythonem w celu wykreślenia wszystkich.

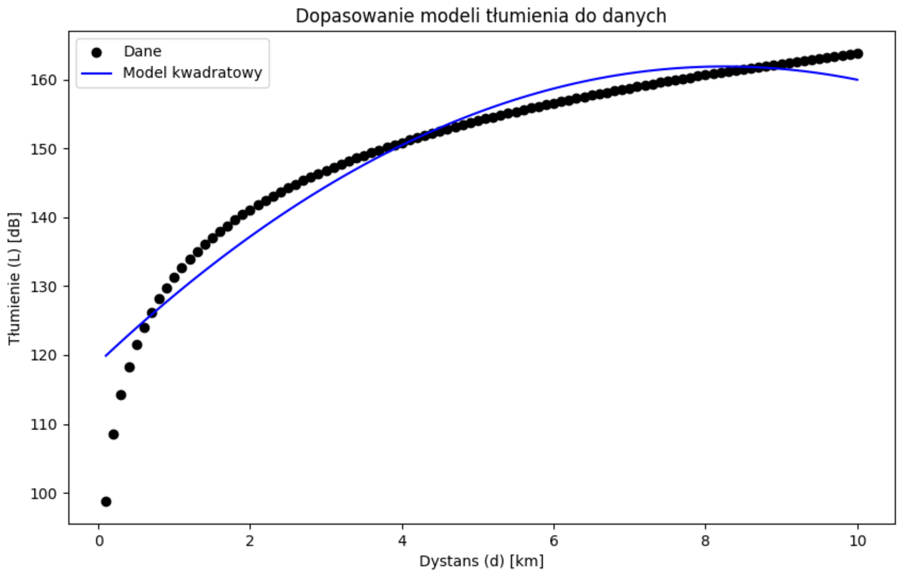
```
  import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Generowanie danych zgodnie z modelem
c = 3 # łStaa śćwarto c zgodnie z zadaniem
d = np.arange(0.1, 10.1, 0.1)
L = 128.3 + 32.51 * np.log10(d) + c # Dodajemy łstaą śćwarto c

# Definicje modeli
def log_model(d, alpha, beta):
    return alpha + beta * np.log10(d)

def quadratic_model(d, alpha, beta, gamma):
    return alpha + beta * d + gamma * d**2

def cubic_model(d, alpha, beta, gamma, delta):
    return alpha + beta * d + gamma * d**2 + delta * d**3

# Dopasowanie modeli do danych
popt_log, _ = curve_fit(log_model, d, L)
popt_quad, _ = curve_fit(quadratic_model, d, L)
popt_cubic, _ = curve_fit(cubic_model, d, L)

# Predykcja przy użyciu dopasowanych modeli
L_log_fit = log_model(d, *popt_log)
L_quad_fit = quadratic_model(d, *popt_quad)
L_cubic_fit = cubic_model(d, *popt_cubic)

# Wykres danych i dopasowanych modeli
plt.figure(figsize=(10, 6))
plt.scatter(d, L, label='Dane', color='black')
plt.plot(d, L_log_fit, label='Model logarytmiczny', color='red')
plt.plot(d, L_quad_fit, label='Model kwadratowy', color='blue')
plt.plot(d, L_cubic_fit, label='Model trzeciego stopnia', color='green')
plt.xlabel('łśćOdlego (d) [km]')
plt.ylabel('łTumienie (L) [dB]')
plt.legend()
plt.title('Dopasowanie modeli łtumienia do danych')
plt.show()

# śWywietlenie parametrów
print("Parametry modelu logarytmicznego: alpha = {:.4f}, beta =
    {:.4f}".format(*popt_log))
print("Parametry modelu kwadratowego: alpha = {:.4f}, beta = {:.4f},
    gamma = {:.4f}".format(*popt_quad))
print("Parametry modelu trzeciego stopnia: alpha = {:.4f}, beta =
    {:.4f}, gamma = {:.4f}, delta = {:.4f}".format(*popt_cubic))
```

Wykonane wykresy oraz punkty powstały podstawiając do wzoru $L = 128.3 + 32.5 logd + c$ za c = 3. Powstałe wykresy prezentują się następująco: Parametry



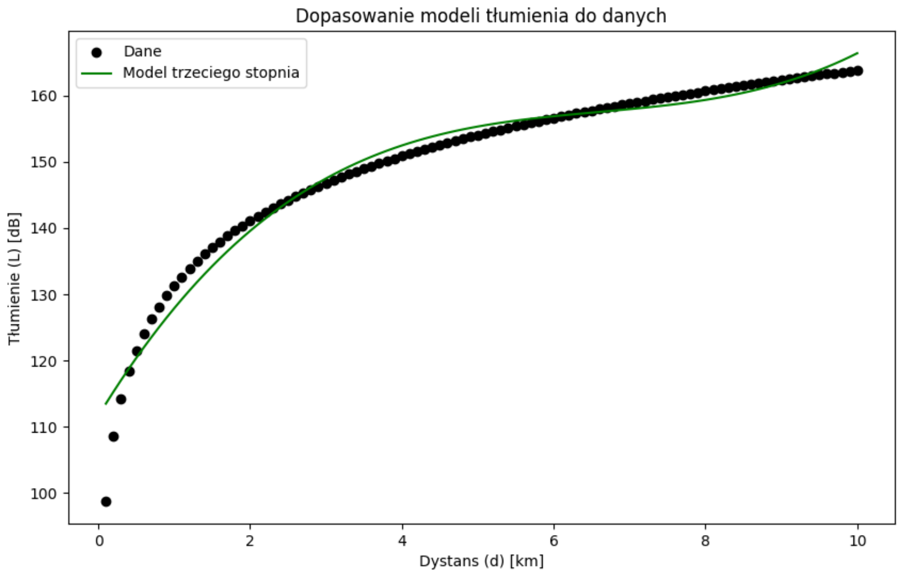
Rys 5.1 Wykres modelu logarytmicznego dopasowany do danych dla c stałego

modelu logarytmicznego: $\alpha = 131.3000, \beta = 32.5100$



Rys 5.2 Wykres modelu kwadratowego dopasowany do danych dla c stałego

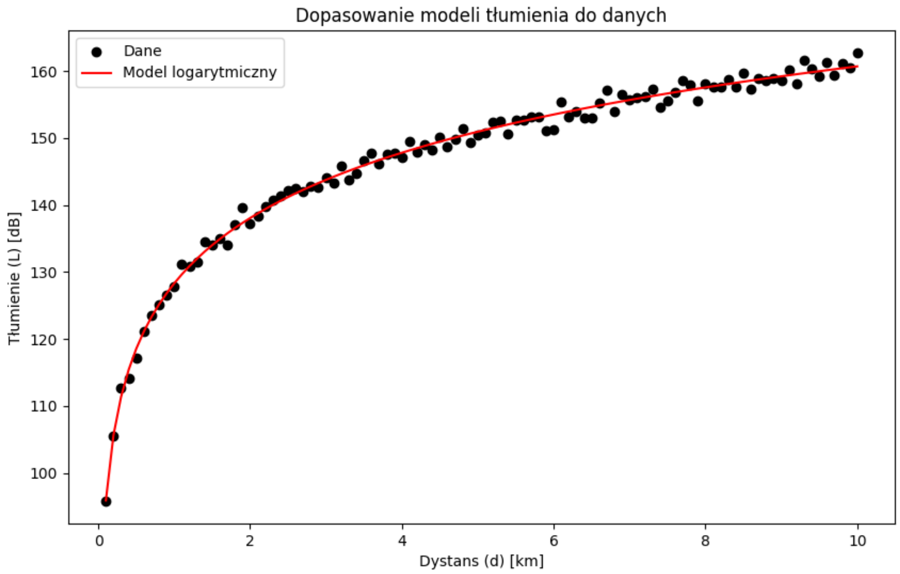
Parametry modelu kwadratowego: $\alpha = 118.8711, \beta = 10.4237, \gamma = -0.6314$



Rys 5.3 Wykres modelu trzeciego stopnia dopasowany do danych dla c stałego

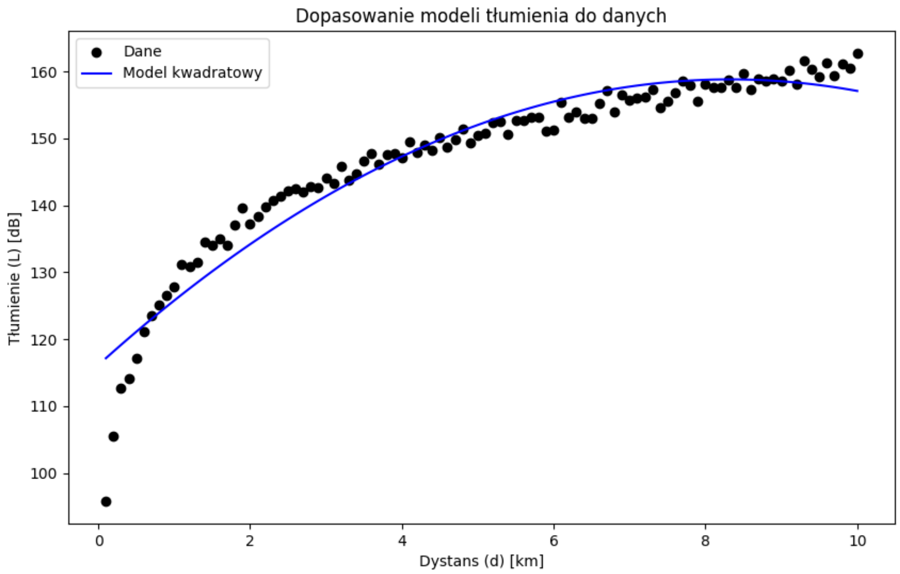
Parametry modelu trzeciego stopnia: $\alpha = 111.6540, \beta = 18.7910, \gamma = -2.6923, \delta = 0.1360$

W momencie gdy za c podstawimy $\mathcal{N}(0, \sigma^2)$, a za $\sigma = 1$ otrzymane wykresy i parametry modeli wyglądają następująco



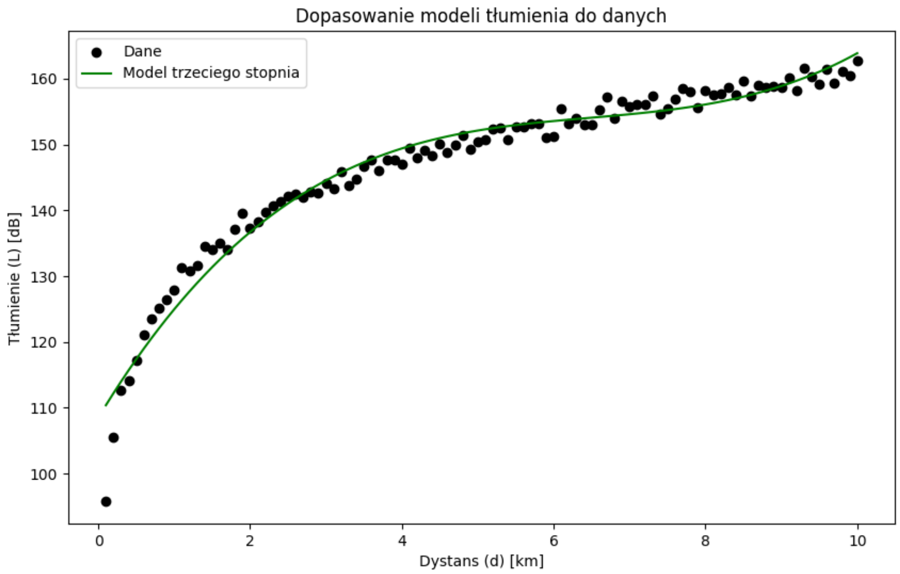
Rys 5.3 Wykres modelu logarytmicznego dopasowany do danych dla$c \sim \mathcal{N}(0, \sigma^2)$

Parametry modelu logarytmicznego: $\alpha = 128.2009$, $\beta = 32.6622$



Rys 5.3 Wykres modelu kwadratowego dopasowany do danych dla $c \sim \mathcal{N}(0, \sigma^2)$

Parametry modelu kwadratowego: alpha = 115.5702, beta = 10.5278, gamma = -0.6384



Rys 5.3 Wykres modelu trzeciego stopnia dopasowany do danych dla $c \sim \mathcal{N}(0, \sigma^2)$

Parametry modelu trzeciego stopnia: alpha = 108.6487, beta = 18.5524, gamma = -2.6148, delta = 0.1305

Dla wartości $c \sim \mathcal{N}(0, \sigma^2)$ trudniej jest opisać, który z modeli jest dokładniejszy dla podanego zadania. Można jednak zauważyć, że w przypadku modelu logarytmicznego parametry pokrywają się z parametrami funkcji L. Jednak zakładając że c = 3, wartości stają się jeszcze bardziej dokładniejsze. Wychodzi wtedy, że model loarytmiczny idealnie sprawdza się w tym typie zadania, gdzie pozostałe lekko odbiegają od jego poprawności.