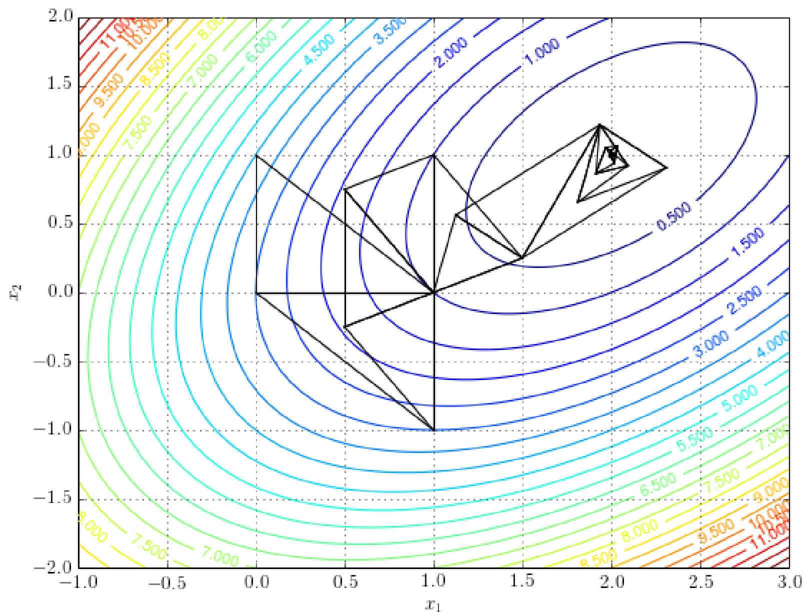
# Laboratorium 11

Minimalizacja funkcji

Jan Rajczyk

31 maja 2021



# 1 Treści zadań

Znajdź minimum funkcji:

1. $f(x_1, x_2) = x_1^2 + (x_2 - 1)^2$
2. $f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$
3. $f(x_1, x_2) = 837.9658 - x_1 \sin\sqrt{|x_1|} - x_2 \sin\sqrt{|x_2|}$,
   dla $-500 \leq x_1, x_2 \leq 500$.

# 2 Rozwiązania

1. Rozwiązanie metodą *simplex*
   Do rozwiązania zadań metodą *simplex* skorzystałem z programu napisanego w języku *Python* i użyłem w nim funkcji *minimize* z biblioteki *scipy*, gdzie jako parametr określający metodę podałem metodę Neldera-Meada, która jest tożsama z metodą *simplex*. Poniższy kod przedstawia znajdowanie minimów wybranych funkcji metodą *simplex*:

```
import math

import numpy as np
from scipy.optimize import minimize


def f1(x):
    return x[0]**2.0 + (x[1]-1)**2.0


def f2(x):
    return 100*(x[0]**2.0 - x[1])**2.0 + (1-x[0])**2.0


def f3(x):
    return 837.9658 - x[0]*math.sin(math.sqrt(abs(x[0]))) -
    x[1]*math.sin(math.sqrt(abs(x[1])))


result1 = minimize(f1, (0.0, 0.0), method='nelder-mead')
result2 = minimize(f2, (0.0, 0.0), method='nelder-mead')
result3 = minimize(f3, (420.0, 420.0), method='nelder-mead')
#print('Total Evaluations: %d' % result1['nfev'])
solution1 = result1['x']
solution2 = result2['x']
solution3 = result3['x']
```

```
evaluation1 = f1(solution1)
evaluation2 = f2(solution2)
evaluation3 = f3(solution3)
print('Solution: f(%s) = %.5f' % (solution1, evaluation1))
print('Solution: f(%s) = %.5f' % (solution2, evaluation2))
print('Solution: f(%s) = %.5f' % (solution3, evaluation3))
```

Otrzymałem następujące wyniki:

(a) $min(f_1) = 0$, dla $x_1 = 0, x_2 = 1$

(b) $min(f_2) = 0$, dla $x_1 = 1, x_2 = 1$

(c) $min(f_3) = 0.00003$, dla $x_1 = 420.9687, x_2 = 420.9687$.

2. Rozwiązanie metodą gradientu sprzężonego (*conjugate gradient method*
Przy rozwiązaniu zadanych problemów metodą gradientu sprzężonego również posłużyłem się funkcją *minimize* z biblioteki *scipy*, tym razem natomiast jako metodą wskazałem metodę gradientu sprzężonego (*cg*). Kod programu prezentuje się następująco:

```
import math

import numpy as np
from scipy.optimize import minimize


def f1(x):
    return x[0]**2.0 + (x[1]-1)**2.0


def f2(x):
    return 100*(x[0]**2.0 - x[1])**2.0 + (1-x[0])**2.0


def f3(x):
    return 837.9658 - x[0]*math.sin(math.sqrt(abs(x[0]))) -
    x[1]*math.sin(math.sqrt(abs(x[1])))


result1 = minimize(f1, np.array([0.0, 0.0]), method='cg')
result2 = minimize(f2, np.array([0.0, 0.0]), method='cg')
result3 = minimize(f3, np.array([400.0, 400.0]), method='cg')
solution1 = result1['x']
solution2 = result2['x']
solution3 = result3['x']
evaluation1 = f1(solution1)
evaluation2 = f2(solution2)
evaluation3 = f3(solution3)
```

```
print('Solution: f(%s) = %.5f' % (solution1, evaluation1))
print('Solution: f(%s) = %.5f' % (solution2, evaluation2))
print('Solution: f(%s) = %.5f' % (solution3, evaluation3))
```

Po wykonaniu powyższego programu otrzymałem nastepujące wyniki:

(a) $min(f_1) = 0$, dla $x_1 = 0, x_2 = 1$

(b) $min(f_2) = 0$, dla $x_1 = 1, x_2 = 1$

(c) $min(f_3) = 0.00003$, dla $x_1 = 420.9687, x_2 = 420.9687$.

**Wnioski**: Jak widzimy metoda *simplex* dała poprawne rezultaty. Pomimo zastosowania funkcji bibliotecznej *minimize* możemy też zauważyć, że metoda ta nie jest trudna w samodzielnej implementacji. Widzimy tutaj zgodność z jej łacińskim tłumaczeniem: *simplex* - prosty. istnieją również pewne modyfikacje tej metody - między innymi *metoda wzmocnionego spadku Tsenga*.
Co do metody gradientu sprzężonego to również można zobaczyć, że daje ona poprawne wyniki. To co okazuje się bardzo istotne po wykonaniu tego ćwiczenia to znajomość różnych bibliotek, ponieważ z ich wykorzystaniem możemy bardzo łatwo przejść z trudnej przestrzeni implementacji wielu funkcji (często operujących na ciężkich i nieoczywistych macierzach) w prostą przestrzeń wykorzystania bogactw narzędzi, które oferuje nam język *Python*, jak i inne takie jak *R* czy też *C++*.
Sama optymalizacja jest tematym bardzo ważnym i ciekawym, ponieważ tworzy pewien fundament wykorzystywany w wielu konceptach, które znajdują swoje zastosowanie w projektach inżynierskich zarówno w przemyśle, jak i badaniach naukowych.

# 3 Bibliografia


- Katarzyna Rycerz *Wykład z przedmiotu Metody Obliczeniowe w Nauce i Technice*
- `https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method`
- `https://pl.wiktionary.org/wiki/simplex`
- Marian Bubak *PhD*
- `https://industrialengineer.online/operations-research/simplex-method/`
- `https://en.wikipedia.org/wiki/Conjugate_gradient_method`
- `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`