Optymalizacja funkcji jednej zmiennej metodami bezgradientowymi

1. Cel ćwiczenia.

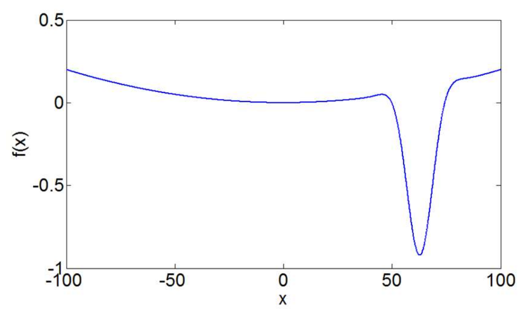
Celem ćwiczenia jest zapoznanie się z metodami bezgradientowymi poprzez ich implementację oraz wykorzystanie do rozwiązania jednowymiarowego problemu optymalizacji.

2. Testowa funkcja celu.

Funkcja celu dana jest wzorem:

$$f(x) = -cos(0{,}1x) \cdot e^{-(0{,}1x-2\pi)^2} + 0{,}002 \cdot (0{,}1x)^2$$

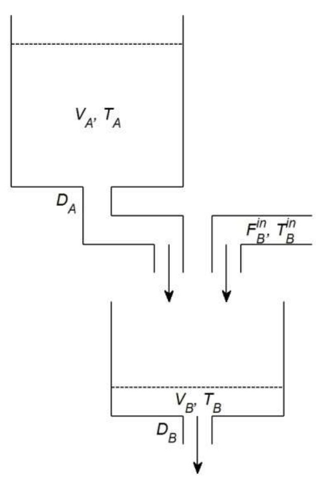
Jej wykres przedstawiony jest poniżej.



Punkt startowy powinien należeć do przedziału $x^{(0)} \in [-100, 100]$.

3. Problem rzeczywisty.

Są dwa zbiorniki z wodą A (górny) i B (dolny).



Zbiornik A ma pole podstawy $P_A = 2m^2$ i zawiera $V_A^0 = 5m^3$ wody o temperaturze $T_A^0 = 95°C$. Zbiornik B ma pole podstawy $P_B = 1m^2$ i zawiera $V_B^0 = 1m^3$ wody o temperaturze $T_B^0 = 20°C$. Woda ze zbiornika A wlewa się do B poprzez otwór o polu przekroju D_A. Dodatkowo, do zbiornika B wlewa się woda o temperaturze $T_B^{in} = 20°C$ z szybkością $F_B^{in} = 10\, {}^{l}/_{s}$. Ze zbiornika B woda wylewa się poprzez otwór o polu przekroju $D_B = 36.5665cm^2$. Zmiana objętość wody w zbiorniku spowodowana jej wypływem przez otwór o polu przekroju D dana jest wzorem:

$$\frac{dV}{dt} = -a \cdot b \cdot D \cdot \sqrt{2g\frac{V}{P}},$$

gdzie: $a = 0.98$ – współczynnik odpowiadający za lepkość cieczy, $b = 0.63$ – współczynnik odpowiadający za zwężenie strumienia cieczy, $g = 9.81\, {}^{m}/_{s^2}$ – przyspieszenie ziemskie.

Zmiana temperatury wody w zbiorniku dana jest wzorem:

$$\frac{dT}{dt} = \frac{V^{in}}{V} \cdot \left(T^{in} - T\right),$$

gdzie: V^{in}, T^{in} – objętość i temperatura wpływającej wody, V, T – objętość i temperatura wody w zbiorniku.

Celem optymalizacji jest znalezienie takiego pola przekroju D_A, dla którego maksymalna temperatura wody w zbiorniku B będzie równa 50°C. Punkt startowy $D_A^{(0)} \in [1, 100]cm^2$. Symulacje należy przeprowadzać dla czasu od $t_0 = 0$ do $t_{end} = 2000s$ z krokiem $dt = 1s$.

W celu sprawdzenia poprawności implementacji modelu, można przeprowadzić symulację dla $D_A = 50cm^2$. Maksymalna temperatura wody w zbiorniku B powinna wynosić około 79.14°C.

4. Algorytmy optymalizacji.

Do wstępnego oszacowania przedziału poszukiwań należy wykorzystać zmodyfikowaną metodę ekspansji. Do wyznaczenia minimum w otrzymanym przedziale należy zastosować metodę Fibonacciego oraz metodę opartą na interpolacji Lagrange'a.

5. Zadanie do samodzielnego wykonania.

a. Testowa funkcja celu.

Zadanie polega na wykonaniu 100 optymalizacji dla trzech różnych współczynników ekspansji startując z losowego punktu startowego (jeżeli w dwóch sprawozdaniach pojawią się identyczne punkty startowe będą one ocenione na 0 punktów). Po wstępnym zawężeniu przedziału poszukiwań, należy przeprowadzić optymalizację dwoma wymienionymi metodami porównując ich dokładność i szybkość zbieżności. Ponadto, należy przeprowadzić optymalizację nie wykonując początkowego zawężenia przedziału poszukiwań. Wyniki należy zestawić pliku xlsx w tabeli 1. Wartości średnie należy przedstawić w tabeli 2. Dodatkowo, dla przypadku bez wstępnego zawężania przedziału poszukiwań należy narysować wykres przedstawiający długość przedziału $[a, b]$ jako funkcję numeru iteracji (na jednym wykresie dla obydwóch metod poszukiwania minimum).

b. Problem rzeczywisty.

Zadanie polega na przeprowadzeniu optymalizacji wykorzystując metodę Fibonacciego oraz metodę opartą na interpolacji Lagrange'a. Wyniki należy zestawić w tabeli 3. Dla znalezionego, optymalnego pola przekroju D_A należy przeprowadzić symulację, a jej wyniki wstawić do arkusza Symulacja. Na ich podstawie należy narysować wykresy przedstawiające objętość wody w zbiorniku A i B oraz temperaturę wody w zbiorniku B.

6. Sprawozdanie.

Sprawozdanie powinno zostać przygotowane w formacie docx (lub doc) albo pdf i powinno zawierać parametry poszczególnych algorytmów, dyskusję wyników oraz wnioski. Dodatkowo, w sprawozdaniu należy umieścić kod zaimplementowanych metod, funkcję lab1 oraz funkcje wykorzystane do obliczenia funkcji celu i pochodnych podczas rozwiązywania równań różniczkowych. Wyniki optymalizacji oraz wykresy należy przygotować w formacie xlsx (lub xls).

Pseudokod metody ekspansji.

Dane wejściowe: dowolny punkt startowy $x^{(0)}$, odległość $d = x^{(1)} - x^{(0)}$, $d > 0$, współczynnik ekspansji $\alpha > 1$, maksymalna liczba wywołań funkcji celu N_{max}

```
i = 0
x^(1) = x^(0) + d
if f(x^(1)) = f(x^(0)) then
     return [x^(0), x^(1)]
end if
if f(x^(1)) > f(x^(0)) then
     d = -d
     x^(1) = x^(0) + d
     if f(x^(1)) ≥ f(x^(0)) then
          return [x^(1), x^(0) – d]
     end if
end if
repeat
     if f_calls > N_max
          return error
     end if
     i = i + 1
     x^(i+1) = x^(0) + α^i·d
until f(x^(i)) ≤ f(x^(i+1))
if d > 0
     return [x^(i-1), x^(i+1)]
end if
return [x^(i+1), x^(i-1)]
```

Pseudokod metody Fibonacciego.

Dane wejściowe: przedział poszukiwań $[a, b]$, dokładność obliczeń $\varepsilon > 0$

```
znajdź najmniejszą liczbę k spełniającą nierówność φ_k > (b - a) / ε
a^(0) = a, b^(0) = b
c^(0) = b^(0) - φ_(k-1) / φ_k·(b^(0) - a^(0))
d^(0) = a^(0) + b^(0) - c^(0)
for i = 0 to k – 3 do
     if f(c^(i)) < f(d^(i)) then
          a^(i+1) = a^(i)
          b^(i+1) = d^(i)
     else
          b^(i+1) = b^(i)
          a^(i+1) = c^(i)
     end if
     c^(i+1) = b^(i+1) - φ_(k-i-2) / φ_(k-i-1)·(b^(i+1) - a^(i+1))
     d^(i+1) = a^(i+1) + b^(i+1)- c^(i+1)
end for
return x* = c^(i+1);
```

Pseudokod metody opartej na interpolacji Lagrange'a.

Dane wejściowe: przedział poszukiwań $[a, b]$ i punkt wewnętrzny c, dokładności obliczeń $\varepsilon > 0$ i $\gamma > 0$, maksymalna liczba wywołań funkcji celu N_{max}

```
i = 0
a^(0) = a, b^(0) = b, c^(0) = c
repeat
     l= f(a^(i))((b^(i))^2- (c^(i))^2) + f(b^(i))((c^(i))^2 - (a^(i))^2) + f(c^(i))((a^(i))^2
     - (b^(i))^2)
     m = f(a^(i))(b^(i) - c^(i)) + f(b^(i))(c^(i) - a^(i)) + f(c^(i))(a^(i) - b^(i))
     if m ≤ 0
          return error
     end if
     d^(i) = 0,5·l / m
     if a^(i) < d^(i) < c^(i) then
          if f(d^(i)) < f(c^(i)) then
               a^(i+1) = a^(i)
               c^(i+1) = d^(i)
               b^(i+1) = c^(i)
          else
               a^(i+1) = d^(i)
               c^(i+1) = c^(i)
               b^(i+1) = b^(i)
          end if
```

```
    else
        if c^(i) < d^(i) <b^(i) then
            if f(d^(i)) < f(c^(i)) then
                a^(i+1) = c^(i)
                c^(i+1) = d^(i)
                b^(i+1) = b^(i)
            else
                a^(i+1) = a^(i)
                c^(i+1) = c^(i)
                b^(i+1) = d^(i)
            end if
        else
            return error
        end if
    end if
    i = i + 1
    if f_calls > N_max then
        return error
    end if
until b^(i) – a^(i) < ε or |d^(i) – d^(i-1)| < γ
return x* = d^(i)
```