КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Кафедра інтелектуальних та інформаційних систем

Лабораторна робота № 1
з дисципліни
“Методи синтезу та оптимізації”

Виконав студент
групи КН- 31
Пашковський Павло Володимирович

Київ-2020

Завдання:

$$1.\quad y = k_1(x - k_2)^2 + k_3\,,\qquad k_1, k_2, k_3 = \overline{0,10}$$

Рис. 1. Завдання

Функція **y = 6(x – 6)^2 + 6**; точність **e=10^(-6).**

Методи:

- метод ділення інтервалу навпіл (метод дихотомічного ділення)
- метод “золотого перетину”
- метод Н’ютона

Знаходження початкового наближення:

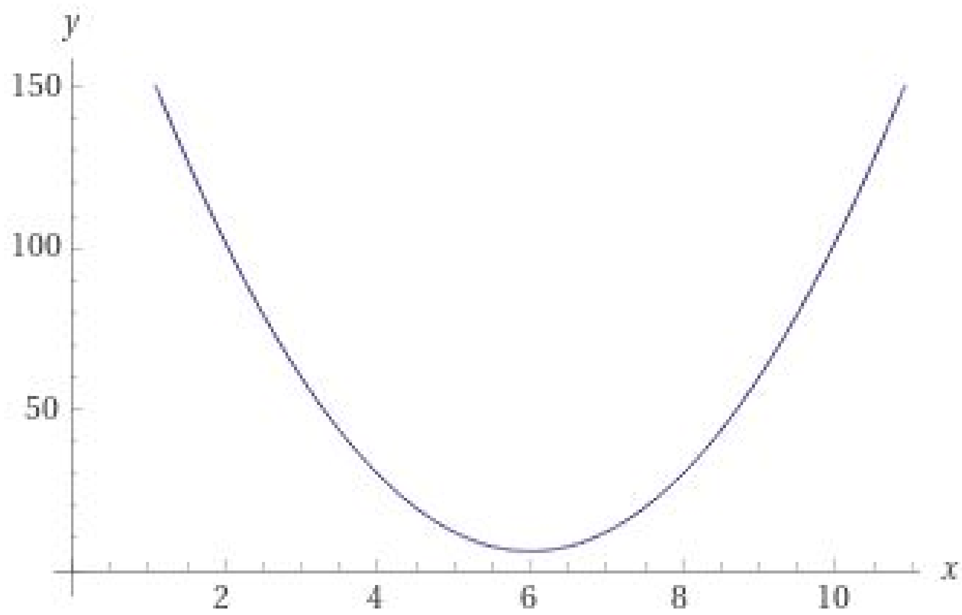

Рис. 2. Графік функції

Оберемо наступне початкове наближення: **a=4, b=8.**

Метод дихотомічного ділення

Алгоритм:

Крок 1. Покласти $x_m = (a + b) / 2$; довжина інтервалу $L = b - a$. Обчислити значення $f(x_m)$.

Крок 2. Покласти $x_1 = a + L / 4$; $x_2 = b - L / 4$. Обчислити значення $f(x_1)$ та $f(x_2)$.

Крок 3. Порівняти $f(x_1)$ та $f(x_m)$.

Якщо $f(x_1) < f(x_m)$, то виключити інтервал $(x_m, b]$, далі покласти $b = x_m$, $x_m = x_1$.

Отже, серединою точкою нового інтервалу пошуку стає колишня точка x_1.Перейти до кроку 5.

Якщо $f(x_1) \geq f(x_m)$, то перейти до кроку 4.

Крок 4. Порівняти $f(x_2)$ та $f(x_m)$.

Якщо $f(x_2) < f(x_m)$, то

виключити інтервал $[a, x_m)$,

далі покласти $a = x_m$, $x_m = x_2$.

Перейти до кроку 5.

Якщо $f(x_2) \geq f(x_m)$, то виключити $[a, x_1)$ та $(x_2, b]$, покласти $a = x_1$, $b = x_2$.

Перейти до кроку 5.

Крок 5. Обчислити $L = b - a$.

Якщо величина |L| менше заздалегідь встановленої точності ε,

тобто достатньо мала, то закінчити пошук.

У протилежному разі повернутися до кроку 2.

Результат роботи коду:

```
Метод дихотомічного ділення
Ітерація: 1, x1 - 5.0, x2 - 7.0
Ітерація: 2, x1 - 5.5, x2 - 6.5
Ітерація: 3, x1 - 5.75, x2 - 6.25
Ітерація: 4, x1 - 5.875, x2 - 6.125
Ітерація: 5, x1 - 5.9375, x2 - 6.0625
Ітерація: 6, x1 - 5.96875, x2 - 6.03125
Ітерація: 7, x1 - 5.984375, x2 - 6.015625
Ітерація: 8, x1 - 5.9921875, x2 - 6.0078125
Ітерація: 9, x1 - 5.99609375, x2 - 6.00390625
Ітерація: 10, x1 - 5.998046875, x2 - 6.001953125
Ітерація: 11, x1 - 5.9990234375, x2 - 6.0009765625
Ітерація: 12, x1 - 5.99951171875, x2 - 6.00048828125
Ітерація: 13, x1 - 5.999755859375, x2 - 6.000244140625
Ітерація: 14, x1 - 5.9998779296875, x2 - 6.0001220703125
Ітерація: 15, x1 - 5.99993896484375, x2 - 6.00006103515625
Ітерація: 16, x1 - 5.999969482421875, x2 - 6.000030517578125
Ітерація: 17, x1 - 5.9999847412109375, x2 - 6.0000152587890625
Ітерація: 18, x1 - 5.999992370605469, x2 - 6.000007629394531
Ітерація: 19, x1 - 5.999996185302734, x2 - 6.000003814697266
Ітерація: 20, x1 - 5.999998092651367, x2 - 6.000001907348633
Ітерація: 21, x1 - 5.999999046325684, x2 - 6.000000953674316
Ітерація: 22, x1 - 5.999999523162842, x2 - 6.000000476837158
Остаточна відповідь (середнє значення) : 6.0, кількість ітерацій: 22
```

Рис. 3. Результат методу дихотомічного ділення

Метод золотого перетину

Алгоритм:

Покроковий алгоритм методу пошуку мінімуму на відрізку $[a, b]$:

Крок 1. Обчислюємо коефіцієнт дроблення відрізка $\tau \approx 1.618$.

Крок 2. $x_1 = a + (1 - 1 / \tau)(b - a)$, обчислити $f(x_1)$.

Крок 3. $x_2 = a + 1 / \tau (b - a)$, обчислити $f(x_2)$.

Крок 4. Якщо $|x_2 - x_1| < \varepsilon$, де ε – задане відхилення, то $x_m = (x_1 + x_2) / 2$, обчислити $f(x_m)$ і закінчити пошук.

Якщо $|x_2 - x_1| > \varepsilon$, то перейти до кроку 5.

Крок 5. Якщо $f(x_1) > f(x_2)$, то виключити з розгляду інтервал (a, x_1), далі встановити $a = x_1$, $x_1 = x_2$ і $f(x_1) = f(x_2)$. Перейти до кроку 3, потім до кроку 4.

Якщо $f(x_1) < f(x_2)$, то виключити з розгляду інтервал (x_2, b), далі встановити $b = x_2$, $x_2 = x_1$ і $f(x_2) = f(x_1)$. Перейти до кроку 2 і 4.

Результат роботи коду:

```
Метод золотого перетину:
Ітерація 1, x1 - 5.527812113720643, x2 - 6.472187886279357
Ітерація 2, x1 - 6.472071763733401, x2 - 7.055740349987242
Ітерація 3, x1 - 6.111408930786625, x2 - 6.47214353292126
Ітерація 4, x1 - 5.88850235912607, x2 - 6.1114532875158325
Ітерація 5, x1 - 6.111425872973677, x2 - 6.249220019073653
Ітерація 6, x1 - 6.0262795617512905, x2 - 6.111442816448433
Ітерація 7, x1 - 5.973655141959951, x2 - 6.0262900336145515
Ітерація 8, x1 - 6.026283561511176, x2 - 6.058814396897208
Ітерація 9, x1 - 6.006181977282092, x2 - 6.026287561575067
Ітерація 10, x1 - 5.993758254025551, x2 - 6.006184449509467
Ітерація 11, x1 - 6.006182921556825, x2 - 6.013862894043793
Ітерація 12, x1 - 6.001437282166017, x2 - 6.006183865903328
Ітерація 13, x1 - 5.998504254112365, x2 - 6.001437865816514
Ітерація 14, x1 - 6.001437505093091, x2 - 6.003250614922602
Ітерація 15, x1 - 6.000317140997858, x2 - 6.00143772803711
Ітерація 16, x1 - 5.999624703361742, x2 - 6.000317278787733
Ітерація 17, x1 - 6.000317193627118, x2 - 6.000745237771734
Ітерація 18, x1 - 6.000052694873099, x2 - 6.000317246260377
Ітерація 19, x1 - 5.999889222219194, x2 - 6.000052727402926
Ітерація 20, x1 - 6.000052707297964, x2 - 6.000153761181607
Ітерація 21, x1 - 5.9999902636770255, x2 - 6.000052719723775
Ітерація 22, x1 - 5.999951670586209, x2 - 5.99999027135676
Ітерація 23, x1 - 5.999990266610323, x2 - 6.000014123699661
Ітерація 24, x1 - 5.999975524742027, x2 - 5.999990269543844
Ітерація 25, x1 - 5.99999026773079, x2 - 5.999999380710898
Ітерація 26, x1 - 5.999999379590347, x2 - 6.000005011840105
Ітерація 27, x1 - 5.9999958992879945, x2 - 5.9999993802829
Ітерація 28, x1 - 5.99999937985487, x2 - 6.000001531273229
Ітерація 29, x1 - 5.999998050441811, x2 - 5.999999380119412
Ітерація 30, x1 - 5.999999379955913, x2 - 6.000000201759128
Ітерація 31, x1 - 6.000000201658077, x2 - 6.000000709571065
Ітерація 32, x1 - 5.999999887806447, x2 - 6.000000201720531
Остаточна відповідь (середнє значення) : 6.000000000000012, кількість ітерацій: 32
Метод Ньютона
```

Рис. 4. Результат методом золотого перетину

Метод Ньютона

Алгоритм:

Крок 1. Наступне наближення до стаціонарної точки x^* визначається за формулою

$$x_{k+1} = x_k - [f'(x_k) / f''(x_k)].$$

Крок 2. Обчислити $f'(x_{k+1}), f''(x_{k+1})$.

Крок 3. Якщо $|f'(x_{k+1})| < \varepsilon$, то закінчити пошук. У протилежному разі необхідно повернутися до кроку 1.

Як виявляється з алгоритму, цільова функція $f(x)$ повинна бути двічі диференційована.

Результат роботи коду:

Рис. 5. Результат роботи методу Н’ютона

Код (Python):

```
eps = 0.000001
def func(x):
  return 6* (x - 6.0)**2 + 6
def first_derivative(x):
  return 12*x - 72
def second_derivative_2(x):
  return 12
def dichotomy(a, b):
  l = b - a
  xm = (a + b) / 2.0
  iteration = x1 = x2 = 0
  print('Метод дихотомічного ділення')
  while abs(l) > eps:
    x1 = a + l / 4.0
    x2 = b - l / 4.0
    if func(x1) < func(xm):
      b = xm
      xm = x1
    elif func(x2) < func(xm):
      a = xm
      xm = x2
    else:
      a = x1
      b = x2
    l = b - a
    iteration += 1
    print(f'Ітерація: {iteration}, x1 - {x1}, x2 - {x2}')
  print(f'Остаточна відповідь (середнє значення) : {func((x1 + x2) / 2.0)}, кількість ітерацій:
{iteration}')
```

```
def gr(a, b):
   print('Метод золотого перетину:')
   t = 1.618
   iteration = 0
   x1 = x2 = 0
   while abs(b - a) > eps:
       x1 = b - (b - a) / t
       x2 = a + (b - a) / t
       if func(x1) >= func(x2):
           a = x1
       else:
           b = x2
       iteration += 1
       print(f'Ітерація {iteration}, x1 - {x1}, x2 - {x2}')
   print(f'Остаточна відповідь (середнє значення) : {func((x1 + x2) / 2.0)}, кількість ітерацій:
{iteration}')
def newton(a, b):
   print('Метод Ньютона')
   x = (a + b) / 2
   iteration = 0
   while abs(first_derivative(x)) > eps:
       x = x - first_derivative(x) / second_derivative(x)
       iteration += 1
       print(f'Ітерація: {iteration}, x - {x}')
   print(f'Остаточна відповідь : {func(x)}, кількість ітерацій: {iteration}')
dichotomy(4, 8)
gr(4, 8)
newton(4,8)
```

Висновки: під час виконання даної лабораторної роботи я дослідила роботу різних методів одновимірної оптимізації. Порівнюючи результат роботи методів можна зробити висновок, що метод Н'ютона є найбільш ефективним, оскільки дає результат за найменшу кількість ітерацій. Найменш оптимальним методом виявився метод "золотого перетину", оскільки для отримання остаточного результату довелося провести 32 ітерації.