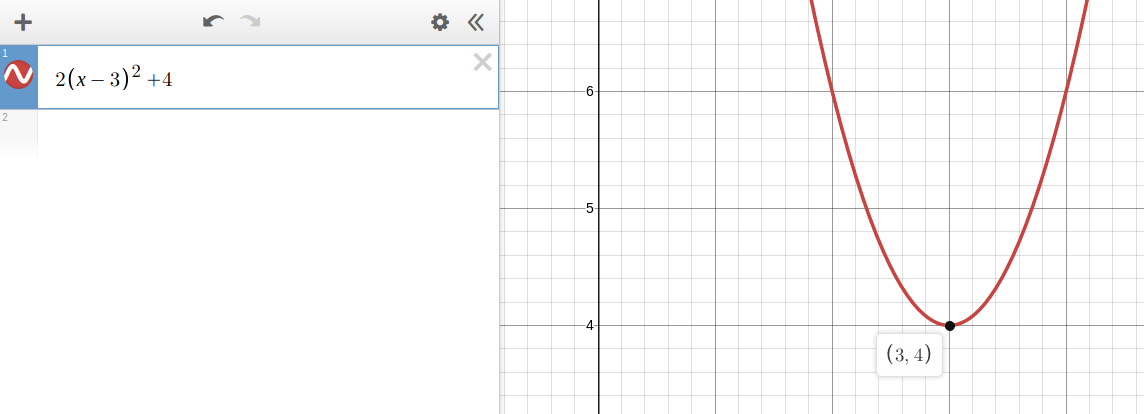
1. Цільова функція

Покладемо , тоді

2. Результат роботи метода дихотомічного ділення для початкового інтервалу [0, 5]

0. f(2.5) = 4.5

1. f(3.75) = 5.125

2. f(3.125) = 4.03125

3. f(2.8125) = 4.0703125

4. f(2.96875) = 4.001953125

5. f(3.046875) = 4.00439453125

6. f(3.0078125) = 4.0001220703125

7. f(2.98828125) = 4.000274658203125

8. f(2.998046875) = 4.000007629394531

9. f(3.0029296875) = 4.000017166137695

10. f(3.00048828125) = 4.000000476837158

11. f(2.999267578125) = 4.000001072883606

12. f(2.9998779296875) = 4.000000029802322

13. f(3.00018310546875) = 4.000000067055225

14. f(3.000030517578125) = 4.000000001862645

15. f(2.9999542236328125) = 4.000000004190952

16. f(2.9999923706054688) = 4.000000000116415

17. f(3.000011444091797) = 4.0000000002619345

18. f(3.000001907348633) = 4.000000000007276

19. f(2.999997138977051) = 4.000000000016371

20. f(2.999999523162842) = 4.000000000000455

21. f(3.0000007152557373) = 4.000000000001023

22. f(3.0000001192092896) = 4.000000000000028

3. Результат роботи метода золотого перетину для початкового інтервалу [0, 5]

0. f(2.5) = 4.5

1. f(3.4549150281252627) = 4.4138953656284174

2. f(2.8647450843757887) = 4.036587784401025

3. f(3.2294901687515774) = 4.1053314751072545

4. f(3.0040653093778915) = 4.000033053480676

5. f(2.8647450843757882) = 4.036587784401025

6. f(2.950849718747371) = 4.004831500294425

7. f(3.0040653093778915) = 4.000033053480676

8. f(2.9711762656368297) = 4.001661615325277

9. f(2.9915028125262877) = 4.000144404389927

10. f(3.0040653093778915) = 4.000033053480676

11. f(2.996301259340037) = 4.000027361364939

12. f(3.001099706153786) = 4.00000241870725

13. f(2.99813410292968) = 4.000006963143754

14. f(2.999966946519324) = 4.0000000021850655

15. f(3.001099706153786) = 4.00000241870725

16. f(3.0003996221986045) = 4.000000319395803

17. f(2.999966946519324) = 4.0000000021850655

18. f(3.0002343547952246) = 4.00000010984434

19. f(3.0000690873918447) = 4.000000009546135

20. f(2.999966946519324) = 4.0000000021850655

21. f(3.0000300730501825) = 4.000000001808777

22. f(2.99999105870852) = 4.000000000159893

23. f(3.000015170897716) = 4.000000000460313

24. f(3.00000026874525) = 4.000000000000145

25. f(2.9999910587085203) = 4.000000000159893

26. f(2.999996750824257) = 4.000000000021115

27. f(3.00000026874525) = 4.000000000000145

28. f(2.9999980945505067) = 4.000000000007262

29. f(2.999999438276756) = 4.0000000000006315

30. f(3.00000026874525) = 4.000000000000145

31. f(2.9999997554874946) = 4.00000000000012

32. f(3.0000000726982323) = 4.000000000000011

4. Результат роботи метода Ньютона для початкової точки х = 123

0. f(123) = 28804

2. f(2.998730936789457) = 4.000003221042864

3. f(2.999999999980175) = 4.0

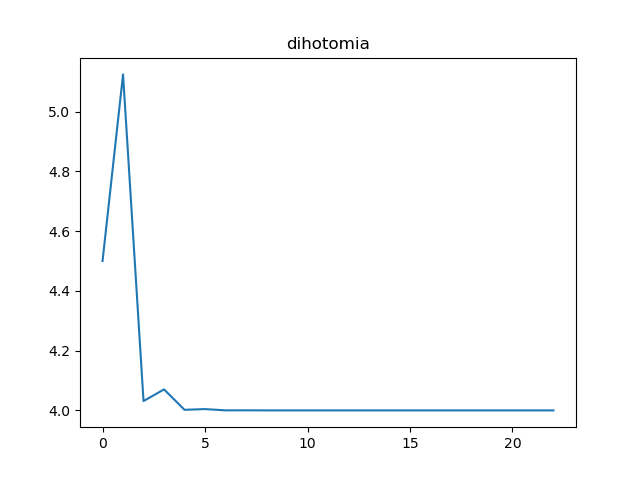
4. f(3.000000000000159) = 4.0

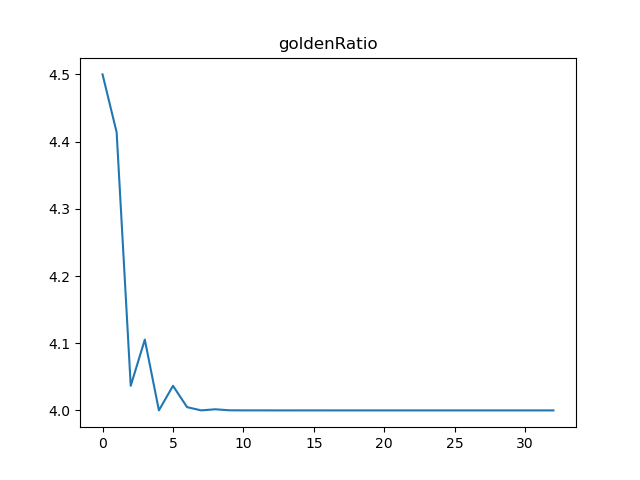
5. Аналіз отриманих результатів

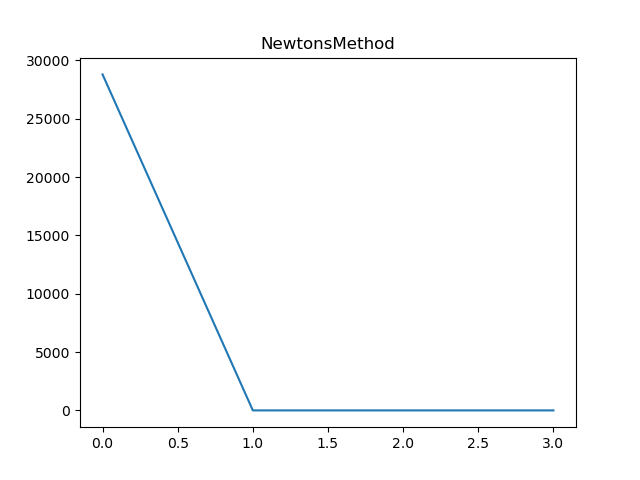
Метод дихотомічного ділення та золотого перетину дійшли до результату за більшу кількість ітерацій, ніж метод Ньютона. Це викликано тим, що швидкість збіжності починає падати пропорційно зменшенню довжини області пошуку. Проте дані методи не потребують розрахунку першої та другої похідних, тому їх можна використовувати на більшому класі функцій.

Метод Ньютона має квадратичну швидкість збіжності, проте потребує розрахунку першої та другої похідних функції. Особливо це буде відчуватися на функціях від багатьох змінних, оскільки треба буде рахувати зворотній Гессіан.

6. Графіки збіжності методів







7. Код програми (Python)

import matplotlib.pyplot as plt

from math import \*

eps = 1e-6

def f(x):

k1 = 2

k2 = 3

k3 = 4

return k1\*(x-k2)\*\*2 + k3

def dfdx(f, x):

return (f(x+1e-4) - f(x-1e-4))/2e-4

def df2dx2(f, x):

return (f(x+1e-4) - 2\*f(x) + f(x-1e-4))/1e-8

def dihotomia(a, b, f):

it = [f((a+b)/2)]

print('0. f({}) = {}'.format((a+b)/2, f((a+b)/2)))

while True:

c = (a+b)/2

if(f(c-eps) < f(c+eps)):

b = c

else:

a = c

if b-a <= eps:

break

print('{}. f({}) = {}'.format(len(it), (a+b)/2, f((a+b)/2)))

it.append(f((a+b)/2))

plt.plot(it)

plt.title('dihotomia')

plt.show()

plt.clf()

def goldenRatio(a, b, f):

it = [f((a+b)/2)]

print('0. f({}) = {}'.format((a+b)/2, f((a+b)/2)))

golden = (1+sqrt(5))/2

while True:

x1 = b - (b-a)/golden

x2 = a + (b-a)/golden

if(f(x1) > f(x2)):

a = x1

else:

b = x2

if(abs(b-a) <= eps):

break

print('{}. f({}) = {}'.format(len(it),(a+b)/2, f((a+b)/2)))

it.append(f((a+b)/2))

plt.plot(it)

plt.title('goldenRatio')

plt.show()

plt.clf()

def NewtonsMethod(x, f):

it = [f(x)]

print('0. f({}) = {}'.format(x, f(x)))

while True:

x1 = x - dfdx(f, x) / df2dx2(f, x)

it.append(f(x1))

print('{}. f({}) = {}'.format(len(it), x1, f(x1)))

if abs(x1 - x) < eps:

break

x = x1

plt.plot(it)

plt.title('NewtonsMethod')

plt.show()

plt.clf()

print('dihotomia')

dihotomia(0, 5, f)

print('goldenRatio')

goldenRatio(0, 5, f)

print('NewtonsMethod')

NewtonsMethod(123, f)