|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Лабораторная работа № 1 | | |
| по дисциплине «Методы оптимизации» | | |
| **Методы одномерного поиска** | | |
|  | | |
|  | Бригада 2 | Буданцев Дмитрий |
| Группа ПМ-13 | голубь Андрей |
| Вариант 2 |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | Филиппова Елена Владимировна |
|  |  |
| Новосибирск,2024 | | |

1. **Цель работы**:

Ознакомиться с методами одномерного поиска, используемыми в многомерных методах минимизации функции n переменных. Сравнить различные алгоритмы по эффективности на тестовых примерах.

1. **Задания:**
   1. Реализовать методы дихотомии, золотого сечения, исследовать их сходимость и провести сравнение по числу вычислений функции для достижения заданной точности  от  до . Построить график зависимости количества вычислений минимизируемой функции от десятичного логарифма задаваемой точности .
   2. Реализовать алгоритм поиска интервала, содержащего минимум функций.
   3. Реализовать алгоритм поиска интервала, содержащего минимум функции.



Метод дихотомии:

Точки  выбираются на расстоянии  от середины отрезка:

 (1)

За одну итерацию интервал неопределенности уменьшается примерно в два раза. После  итераций длина интервала будет равна примерно . Для достижения точности  потребуется  итераций.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | **x1** | **x2** | **f(x1)** | **f(x2)** | **a** | **b** | **len** | **prev/now** |
| **0** | 9,00000 | 9,00000 | 49,00000 | 49,00000 | -2,000000 | 9,000000 | 1,10E+01 | 2,0000000 |
| **1** | 3,50000 | 3,50000 | 2,25000 | 2,25000 | -2,000000 | 3,500000 | 5,50E+00 | 2,0000000 |
| **2** | 0,75000 | 0,75000 | 1,56250 | 1,56250 | 0,750000 | 3,500000 | 2,75E+00 | 2,0000000 |
| **3** | 2,12500 | 2,12500 | 0,01563 | 0,01563 | 0,750000 | 2,125000 | 1,38E+00 | 2,0000000 |
| **4** | 1,43750 | 1,43750 | 0,31641 | 0,31641 | 1,437500 | 2,125000 | 6,88E-01 | 1,9999999 |
| **5** | 1,78125 | 1,78125 | 0,04785 | 0,04785 | 1,781250 | 2,125000 | 3,44E-01 | 1,9999999 |
| **6** | 1,95312 | 1,95313 | 0,00220 | 0,00220 | 1,953125 | 2,125000 | 1,72E-01 | 1,9999997 |
| **7** | 2,03906 | 2,03906 | 0,00153 | 0,00153 | 1,953125 | 2,039063 | 8,59E-02 | 1,9999994 |
| **8** | 1,99609 | 1,99609 | 0,00002 | 0,00002 | 1,996094 | 2,039063 | 4,30E-02 | 1,9999988 |
| **9** | 2,01758 | 2,01758 | 0,00031 | 0,00031 | 1,996094 | 2,017578 | 2,15E-02 | 1,9999977 |
| **10** | 2,00684 | 2,00684 | 0,00005 | 0,00005 | 1,996094 | 2,006836 | 1,07E-02 | 1,9999954 |
| **11** | 2,00146 | 2,00146 | 0,00000 | 0,00000 | 1,996094 | 2,001465 | 5,37E-03 | 1,9999907 |
| **12** | 1,99878 | 1,99878 | 0,00000 | 0,00000 | 1,998779 | 2,001465 | 2,69E-03 | 1,9999814 |
| **13** | 2,00012 | 2,00012 | 0,00000 | 0,00000 | 1,998779 | 2,000122 | 1,34E-03 | 1,9999628 |
| **14** | 1,99945 | 1,99945 | 0,00000 | 0,00000 | 1,999451 | 2,000122 | 6,71E-04 | 1,9999255 |
| **15** | 1,99979 | 1,99979 | 0,00000 | 0,00000 | 1,999786 | 2,000122 | 3,36E-04 | 1,9998511 |
| **16** | 1,99995 | 1,99995 | 0,00000 | 0,00000 | 1,999954 | 2,000122 | 1,68E-04 | 1,9997022 |
| **17** | 2,00004 | 2,00004 | 0,00000 | 0,00000 | 1,999954 | 2,000038 | 8,40E-05 | 1,9994046 |
| **18** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 1,999996 | 2,000038 | 4,20E-05 | 1,9988099 |
| **19** | 2,00002 | 2,00002 | 0,00000 | 0,00000 | 1,999996 | 2,000017 | 2,10E-05 | 1,9976225 |
| **20** | 2,00001 | 2,00001 | 0,00000 | 0,00000 | 1,999996 | 2,000007 | 1,05E-05 | 1,9952564 |
| **21** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 1,999996 | 2,000001 | 5,30E-06 | 1,9905575 |
| **22** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 1,999999 | 2,000001 | 2,67E-06 | 1,9812917 |
| **23** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 1,999999 | 2,000000 | 1,36E-06 | 1,9632705 |
| **24** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 1,999999 | 2,000000 | 7,10E-07 | 1,9291435 |
| **25** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,000000 | 2,000000 | 3,80E-07 | 1,8676638 |
| **26** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,000000 | 2,000000 | 2,10E-07 | 1,7662599 |
| **27** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,000000 | 2,000000 | 1,30E-07 | 1,6210869 |
| **28** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,000000 | 2,000000 | 9,00E-08 | 1,4504178 |

Из таблицы можно подтвердить то, что интервал неопределённости уменьшается примерно в два раза, кроме последних итераций. Это происходит потому, что длина интервала уменьшается, и ближе к концу таблицы изменение длины интервала примерно равно 0, а .

Уменьшение длины интервала совпадает с теоретическим представлением:



Количество итераций совпадает с теоретическим представлением:



Метод золотого сечения:

Точки  находятся симметрично относительно середины отрезка  и делят его в пропорции золотого сечения, когда длина всего отрезка относится к длине большей его части так же, как длина большей части относится к длине меньшей части

Следовательно,  находятся:



За одну итерацию интервал неопределенности уменьшается всего в  раза. Для достижения точности  потребуется итераций.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | **x1** | **x2** | **f(x1)** | **f(x2)** | **a** | **b** | **len** | **prev/now** |
| **0** | 6,40325 | 11,59675 | 19,38863 | 92,09757 | -2,00000 | 11,59675 | 1,36E+01 | 1,61803 |
| **1** | 3,19350 | 6,40325 | 1,42443 | 19,38863 | -2,00000 | 6,40325 | 8,40E+00 | 1,61803 |
| **2** | 1,20976 | 3,19350 | 0,62448 | 1,42443 | -2,00000 | 3,19350 | 5,19E+00 | 1,61803 |
| **3** | -0,01626 | 1,20976 | 4,06531 | 0,62448 | -0,01626 | 3,19350 | 3,21E+00 | 1,61803 |
| **4** | 1,20976 | 1,96748 | 0,62448 | 0,00106 | 1,20976 | 3,19350 | 1,98E+00 | 1,61803 |
| **5** | 1,96748 | 2,43577 | 0,00106 | 0,18990 | 1,20976 | 2,43577 | 1,23E+00 | 1,61803 |
| **6** | 1,67805 | 1,96748 | 0,10365 | 0,00106 | 1,67805 | 2,43577 | 7,58E-01 | 1,61803 |
| **7** | 1,96748 | 2,14635 | 0,00106 | 0,02142 | 1,67805 | 2,14635 | 4,68E-01 | 1,61803 |
| **8** | 1,85693 | 1,96748 | 0,02047 | 0,00106 | 1,85693 | 2,14635 | 2,89E-01 | 1,61803 |
| **9** | 1,96748 | 2,03580 | 0,00106 | 0,00128 | 1,85693 | 2,03580 | 1,79E-01 | 1,61803 |
| **10** | 1,92525 | 1,96748 | 0,00559 | 0,00106 | 1,92525 | 2,03580 | 1,11E-01 | 1,61803 |
| **11** | 1,96748 | 1,99357 | 0,00106 | 0,00004 | 1,96748 | 2,03580 | 6,83E-02 | 1,61803 |
| **12** | 1,99357 | 2,00970 | 0,00004 | 0,00009 | 1,96748 | 2,00970 | 4,22E-02 | 1,61803 |
| **13** | 1,98361 | 1,99357 | 0,00027 | 0,00004 | 1,98361 | 2,00970 | 2,61E-02 | 1,61803 |
| **14** | 1,99357 | 1,99974 | 0,00004 | 0,00000 | 1,99357 | 2,00970 | 1,61E-02 | 1,61803 |
| **15** | 1,99974 | 2,00354 | 0,00000 | 0,00001 | 1,99357 | 2,00354 | 9,97E-03 | 1,61803 |
| **16** | 1,99738 | 1,99974 | 0,00001 | 0,00000 | 1,99738 | 2,00354 | 6,16E-03 | 1,61803 |
| **17** | 1,99974 | 2,00119 | 0,00000 | 0,00000 | 1,99738 | 2,00119 | 3,81E-03 | 1,61803 |
| **18** | 1,99884 | 1,99974 | 0,00000 | 0,00000 | 1,99884 | 2,00119 | 2,35E-03 | 1,61803 |
| **19** | 1,99974 | 2,00029 | 0,00000 | 0,00000 | 1,99884 | 2,00029 | 1,45E-03 | 1,61803 |
| **20** | 1,99939 | 1,99974 | 0,00000 | 0,00000 | 1,99939 | 2,00029 | 8,99E-04 | 1,61803 |
| **21** | 1,99974 | 1,99995 | 0,00000 | 0,00000 | 1,99974 | 2,00029 | 5,56E-04 | 1,61803 |
| **22** | 1,99995 | 2,00008 | 0,00000 | 0,00000 | 1,99974 | 2,00008 | 3,43E-04 | 1,61803 |
| **23** | 1,99987 | 1,99995 | 0,00000 | 0,00000 | 1,99987 | 2,00008 | 2,12E-04 | 1,61803 |
| **24** | 1,99995 | 2,00000 | 0,00000 | 0,00000 | 1,99995 | 2,00008 | 1,31E-04 | 1,61803 |
| **25** | 2,00000 | 2,00003 | 0,00000 | 0,00000 | 1,99995 | 2,00003 | 8,11E-05 | 1,61803 |
| **26** | 1,99998 | 2,00000 | 0,00000 | 0,00000 | 1,99998 | 2,00003 | 5,01E-05 | 1,61803 |
| **27** | 2,00000 | 2,00001 | 0,00000 | 0,00000 | 1,99998 | 2,00001 | 3,10E-05 | 1,61803 |
| **28** | 1,99999 | 2,00000 | 0,00000 | 0,00000 | 1,99999 | 2,00001 | 1,91E-05 | 1,61803 |
| **29** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 1,99999 | 2,00000 | 1,18E-05 | 1,61803 |
| **30** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 7,31E-06 | 1,61803 |
| **31** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 4,52E-06 | 1,61803 |
| **32** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 2,79E-06 | 1,61803 |
| **33** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 1,73E-06 | 1,61803 |
| **34** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 1,07E-06 | 1,61803 |
| **35** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 6,60E-07 | 1,61803 |
| **36** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 4,10E-07 | 1,61803 |
| **37** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 2,50E-07 | 1,61803 |
| **38** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 1,60E-07 | 1,61803 |
| **39** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 1,00E-07 | 1,61803 |

За одну итерацию интервал уменьшается в 1.61803, что совпадает с теоретическим представлением.

Количество итераций совпадает с теоретическим представлением:



В отличие от метода дихотомии имеет большее количество итераций в одинаковых условиях. Что также предсказано теоретически, так как в обоих методах скорость изменения интервалов разное и в методе дихотомии он изменяется быстрей.

Метод Фибоначчи:

В силу того, что в асимптотике , метод золотого сечения может быть трансформирован в метод Фибоначчи. Однако при этом в силу свойств чисел Фибоначчи количество итераций строго ограничено и выбирается из соотношения .

Точки  вычисляются так:



где . При k равном n процесс заканчивается.

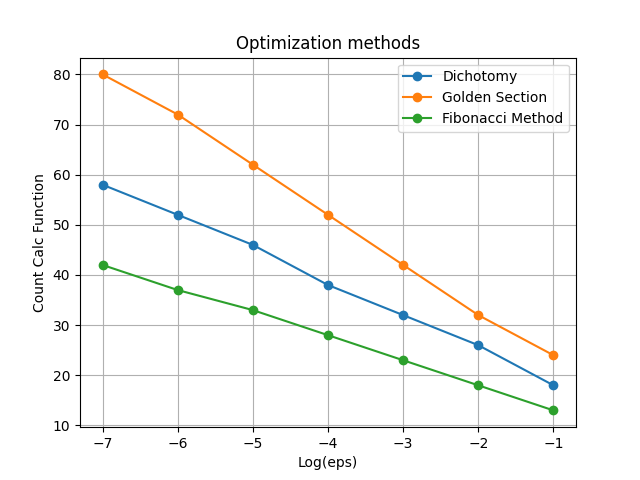
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **i** | **x1** | **x2** | **f(x1)** | **f(x2)** | **a** | **b** | **len** | **prev/now** |
| **0** | 3,19350 | 6,40325 | 1,42443 | 19,38863 | -2,00000 | 11,59675 | 1,36E+01 | 1,61803 |
| **1** | 1,20976 | 3,19350 | 0,62448 | 1,42443 | -2,00000 | 6,40325 | 8,40E+00 | 1,61803 |
| **2** | -0,01626 | 1,20976 | 4,06531 | 0,62448 | -2,00000 | 3,19350 | 5,19E+00 | 1,61803 |
| **3** | 1,20976 | 1,96748 | 0,62448 | 0,00106 | -0,01626 | 3,19350 | 3,21E+00 | 1,61803 |
| **4** | 1,96748 | 2,43577 | 0,00106 | 0,18990 | 1,20976 | 3,19350 | 1,98E+00 | 1,61803 |
| **5** | 1,67805 | 1,96748 | 0,10365 | 0,00106 | 1,20976 | 2,43577 | 1,23E+00 | 1,61803 |
| **6** | 1,96748 | 2,14635 | 0,00106 | 0,02142 | 1,67805 | 2,43577 | 7,58E-01 | 1,61803 |
| **7** | 1,85693 | 1,96748 | 0,02047 | 0,00106 | 1,67805 | 2,14635 | 4,68E-01 | 1,61803 |
| **8** | 1,96748 | 2,03580 | 0,00106 | 0,00128 | 1,85693 | 2,14635 | 2,89E-01 | 1,61803 |
| **9** | 1,92525 | 1,96748 | 0,00559 | 0,00106 | 1,85693 | 2,03580 | 1,79E-01 | 1,61803 |
| **10** | 1,96748 | 1,99357 | 0,00106 | 0,00004 | 1,92525 | 2,03580 | 1,11E-01 | 1,61803 |
| **11** | 1,99357 | 2,00970 | 0,00004 | 0,00009 | 1,96748 | 2,03580 | 6,83E-02 | 1,61803 |
| **12** | 1,98361 | 1,99357 | 0,00027 | 0,00004 | 1,96748 | 2,00970 | 4,22E-02 | 1,61803 |
| **13** | 1,99357 | 1,99974 | 0,00004 | 0,00000 | 1,98361 | 2,00970 | 2,61E-02 | 1,61803 |
| **14** | 1,99974 | 2,00354 | 0,00000 | 0,00001 | 1,99357 | 2,00970 | 1,61E-02 | 1,61803 |
| **15** | 1,99738 | 1,99974 | 0,00001 | 0,00000 | 1,99357 | 2,00354 | 9,97E-03 | 1,61803 |
| **16** | 1,99974 | 2,00119 | 0,00000 | 0,00000 | 1,99738 | 2,00354 | 6,16E-03 | 1,61803 |
| **17** | 1,99884 | 1,99974 | 0,00000 | 0,00000 | 1,99738 | 2,00119 | 3,81E-03 | 1,61804 |
| **18** | 1,99974 | 2,00029 | 0,00000 | 0,00000 | 1,99884 | 2,00119 | 2,35E-03 | 1,61803 |
| **19** | 1,99939 | 1,99974 | 0,00000 | 0,00000 | 1,99884 | 2,00029 | 1,45E-03 | 1,61804 |
| **20** | 1,99974 | 1,99995 | 0,00000 | 0,00000 | 1,99939 | 2,00029 | 8,99E-04 | 1,61802 |
| **21** | 1,99995 | 2,00008 | 0,00000 | 0,00000 | 1,99974 | 2,00029 | 5,55E-04 | 1,61811 |
| **22** | 1,99987 | 1,99995 | 0,00000 | 0,00000 | 1,99974 | 2,00008 | 3,43E-04 | 1,61806 |
| **23** | 1,99995 | 2,00000 | 0,00000 | 0,00000 | 1,99987 | 2,00008 | 2,12E-04 | 1,61797 |
| **24** | 2,00000 | 2,00003 | 0,00000 | 0,00000 | 1,99995 | 2,00008 | 1,31E-04 | 1,61787 |
| **25** | 1,99998 | 2,00000 | 0,00000 | 0,00000 | 1,99995 | 2,00003 | 8,11E-05 | 1,61814 |
| **26** | 2,00000 | 2,00001 | 0,00000 | 0,00000 | 1,99998 | 2,00003 | 5,01E-05 | 1,61775 |
| **27** | 1,99999 | 2,00000 | 0,00000 | 0,00000 | 1,99998 | 2,00001 | 3,10E-05 | 1,61831 |
| **28** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 1,99999 | 2,00001 | 1,91E-05 | 1,61730 |
| **29** | 1,99999 | 2,00000 | 0,00000 | 0,00000 | 1,99999 | 2,00000 | 1,18E-05 | 1,61877 |
| **30** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 1,99999 | 2,00000 | 7,32E-06 | 1,61611 |
| **31** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 4,49E-06 | 1,62838 |
| **32** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 2,77E-06 | 1,62116 |
| **33** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 1,72E-06 | 1,60990 |
| **34** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 1,08E-06 | 1,59630 |
| **35** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 6,60E-07 | 1,63148 |
| **36** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 4,20E-07 | 1,58359 |
| **37** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 2,50E-07 | 1,70005 |
| **38** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 1,50E-07 | 1,67929 |
| **39** | 2,00000 | 2,00000 | 0,00000 | 0,00000 | 2,00000 | 2,00000 | 1,00E-07 | 2,47214 |

Если сравнить последний столбец этой таблицы с таблицей для метода золотого сечения, то увидим совпадения в скорости изменения интервалов, что совпадает с теорией, но ближе к концу видим отклонение. На последней итерации, поскольку изменение происходит по другому правилу, отклонение больше.

Длина на n итерации совпадает с теоретическим представлением:



В отличие от остальных методов требует меньше вычислений минимизируемой функции, что видно из графика:



И видно то, что метод Фибоначчи требует почти в 2 раза меньше вычислений минимизируемой функции, чем в методе золотого сечения.

Алгоритм поиска интервала, содержащего минимум функции:

В отличие от выше рассмотренных методах, этот не требует знать начальный отрезок, содержащую точку минимума. Поиск минимума функции заключается в том, что возрастание по величине шаги осуществляется до тех пор, пока не будет пройдена точка минимума функции.

:

|  |  |  |
| --- | --- | --- |
| **i** | **xi** | **f(xi)** |
| **0** | 19 | 289 |
| **1** | 18 | 256 |
| **2** | 17 | 225 |
| **3** | 16 | 196 |
| **4** | 15 | 169 |
| **5** | 14 | 144 |
| **6** | 13 | 121 |
| **7** | 12 | 100 |
| **8** | 11 | 81 |
| **9** | 10 | 64 |
| **10** | 9 | 49 |
| **11** | 8 | 36 |
| **12** | 7 | 25 |
| **13** | 6 | 16 |
| **14** | 5 | 9 |
| **15** | 4 | 4 |
| **16** | 3 | 1 |
| **17** | 2 | 0 |
| Интервал: [3; 1] | | |

В данном случае  подобрана удачно, и метод останавливается на своём минимуме. При других значениях  метод может перескочить или наоборот не дойти до минимума, но в любом случае интервал будет верным. Так, что метод эффективен для нахождения ближайшего к  интервала, содержащего минимум функции.

1. Программа на Python:

from math import sqrt

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

def f(x):

    return (x-2)\*\*2

def delta(eps):

    calc\_delta = eps / 2

    if calc\_delta < eps:

        return calc\_delta

    else:

        raise "delta считается не правильно"

def dichotomy(start, end, eps):

    a, b, iter, func\_calc = start, end, 0, 0

    df = pd.DataFrame(columns=np.array(['x1','x2', 'f(x1)', 'f(x2)','a', 'b', 'len', 'prev/now']))

    while abs(b-a) > eps:

        a\_step, b\_step = a, b

        x1 = (a + b - delta(eps)) / 2

        x2 = (a + b + delta(eps)) / 2

        if f(x1) > f(x2):

            a = x1

        else:

            b = x2

        iter += 1

        func\_calc += 2

        df.loc[len(df)] = np.array([x1, x2, f(x1), f(x2), a, b, b-a, (b\_step - a\_step)/(b-a)])

    #print(f'Минимальный отрезок: [{a}, {b}]')

    return df, func\_calc

def golden\_section(start, end, eps):

    a, b, iter, func\_calc = start, end, 0, 0

    df = pd.DataFrame(columns=np.array(['x1','x2', 'f(x1)', 'f(x2)', 'a', 'b', 'len', 'prev/now']))

    val = sqrt(5)

    while abs(a - b) > eps:

        a\_step, b\_step = a, b

        x1 = a + (3 - val) \* (b - a) / 2

        x2 = a + (val - 1) \* (b - a) / 2

        if f(x1) > f(x2):

            a = x1

        else:

            b = x2

        iter += 1

        func\_calc += 2

        df.loc[len(df)] = np.array([x1, x2, f(x1), f(x2), a, b, b-a, (b\_step - a\_step)/(b-a)])

    #print(f'Минимальный отрезок: [{a}, {b}]')

    return df, func\_calc

def find\_n(a, b, eps):

    n = 0

    while  (b - a) / eps >= Fib\_num(n):

        n += 1

    return n

def Fib\_num(n):

    val = sqrt(5)

    return (((1 + val)/2)\*\*n - ((1-val)/2)\*\*2) / val

def Fib\_method(start, end, eps):

    n = find\_n(start, end, eps)

    l = start + Fib\_num(n-2) \* (end - start)/Fib\_num(n)

    m = start + Fib\_num(n-1) \* (end - start)/Fib\_num(n)

    a, b, k, calc\_func = start, end, 0, 2

    df = pd.DataFrame(columns=np.array(['x1','x2', 'f(x1)', 'f(x2)','a', 'b', 'len', 'prev/now']))

    func\_l, func\_m = f(l), f(m)

    while k != n - 3:

        a\_step, b\_step = a, b

        if func\_l > func\_m:

            a = l

            l = m

            m = a + Fib\_num(n - k - 2) \* (b - a) / Fib\_num(n - k - 1)

            func\_l = func\_m

            func\_m = f(m)

        else:

            b = m

            m = l

            l = a + Fib\_num(n - k - 3) \* (b - a) / Fib\_num(n - k - 1)

            func\_m = func\_l

            func\_l = f(l)

        k += 1

        calc\_func += 1

        df.loc[len(df)] = np.array([l, m, f(l), f(m), a, b, b-a, (b\_step - a\_step)/(b-a)])

    m = l + eps

    func\_m = f(m)

    calc\_func += 1

    if func\_l >= func\_m:

        a = l

    else:

        b = m

    #print(f'Минимальный отрезок: [{a}, {b}]')

    df.loc[len(df)] = np.array([l, m, f(l), f(m), a, b, b-a, (b\_step - a\_step)/(b-a)])

    return df, calc\_func

def find\_direction(x, delta):

    if f(x) > f(x + delta):

        return delta

    return -delta

def find\_min\_interval(x, delta):

    h = 2 \* find\_direction(x, delta)

    xk, k, func\_calc = x + h, 0, 0

    df = pd.DataFrame(columns=np.array(['xi', 'f(xi)']))

    while f(x) > f(xk):

        x, xk = xk, xk + h

        k += 1

        func\_calc += 2

        df.loc[len(df)] = np.array([x, f(x)])

    #print(f'Минимальный отрезок: [{x - h}, {xk}], кол-во итераций: {k}')

    return df

def grafic():

    a, b = -2, 20

    axis\_x = np.array([np.log10(eps) for eps in [10\*\*(-i) for i in reversed(range(1, 8))]])

    axis\_y\_dec = np.array([dichotomy(a, b, eps)[1] for eps in [10\*\*(-i) for i in reversed(range(1, 8))]])

    axis\_y\_gold = np.array([golden\_section(a, b, eps)[1] for eps in [10\*\*(-i) for i in reversed(range(1, 8))]])

    axis\_y\_fib = np.array([Fib\_method(a, b, eps)[1] for eps in [10\*\*(-i) for i in reversed(range(1, 8))]])

    fig = plt.figure()

    ax = fig.add\_subplot(111)

    ax.patch.set\_facecolor('#2D3332')

    ax.plot(axis\_x, axis\_y\_dec,'o-', label='Dichotomy', color='#E3B414')

    ax.plot(axis\_x, axis\_y\_gold,'o-', label='Golden Section', color='#DD14E3')

    ax.plot(axis\_x, axis\_y\_fib,'o-', label="Fibonacci Method", color='red')

    ax.set\_xlabel("Log(eps)")

    ax.set\_ylabel("Count Calc Function")

    ax.grid(color='#14E39E')

    ax.spines['right'].set\_color('#14E39E')

    ax.spines['top'].set\_color('#14E39E')

    ax.spines['bottom'].set\_color('#14E39E')

    ax.spines['left'].set\_color('#14E39E')

    ax.xaxis.label.set\_color('white')

    ax.tick\_params(colors='#14E39E')

    ax.yaxis.label.set\_color('white')

    ax.legend(facecolor='#2D3332', labelcolor='white')

    fig.set\_facecolor('#2D3332')

    plt.title("Optimization methods", color='white')

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    grafic()

    with pd.ExcelWriter("OutputFile.xlsx") as writer:

        dichotomy(-2, 20, 1e-7)[0].to\_excel(writer, sheet\_name="Dichotomy Method", index\_label='i', float\_format='%.8f')

        golden\_section(-2, 20, 1e-7)[0].to\_excel(writer, sheet\_name="Golden Section Method", index\_label='i', float\_format='%.8f')

        Fib\_method(-2, 20, 1e-7)[0].to\_excel(writer, sheet\_name="Fibonacci Method", index\_label='i', float\_format='%.8f')

        find\_min\_interval(20, 0.5).to\_excel(writer, sheet\_name="Minimum interval", float\_format='%.2f', index\_label='i')