|  |  |  |
| --- | --- | --- |
|  | | |
| 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 7 | буров евгений |
| Группа ПМ-23 | Гайченко максим |
|  | кобылин дмитрий |
|  |  |
|  |  |
|  |  |
| Преподаватель | Лемешко борис юрьевич |
|  |  |
| Новосибирск, 2025 | | |

1. Цель работы

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

1. Задание
   1. Реализовать методы дихотомии, золотого сечения, исследовать их сходимость и провести сравнение по числу вычислений функции для достижения заданной точности ε от 10-1 до 10-7. Построить график зависимости количества вычислений минимизируемой функции от десятичного логарифма задаваемой точности ε.
   2. Реализовать алгоритм поиска интервала, содержащего минимум функции.
   3. Реализовать метод Фибоначчи, сравнить его с методами дихотомии и золотого сечения.

Вариант 7:

1. Таблицы с результатами исследований
   1. Метод дихотомии

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 1 | 9 | 9 | 4 | 4 | -2 | 9 | 11 | 2 |
| 2 | 3.5 | 3.5 | 12.25 | 12.25 | 3.5 | 9 | 5.5 | 4 |
| 3 | 6.25 | 6.25 | 0.5625 | 0.5625 | 6.25 | 9 | 2.75 | 6 |
| 4 | 7.625 | 7.625 | 0.390625 | 0.390625 | 6.25 | 7.625 | 1.375 | 4 |
| 5 | 6.9375 | 6.9375 | 0.003906253 | 0.003906246 | 6.9375 | 7.625 | 0.6875 | 4 |
| 6 | 7.28125 | 7.28125 | 0.07910155 | 0.07910158 | 6.9375 | 7.28125 | 0.34375 | 4 |
| 7 | 7.109375 | 7.109375 | 0.01196289 | 0.0119629 | 6.9375 | 7.109375 | 0.171875 | 5.999999 |
| 8 | 7.023437 | 7.023438 | 0.0005493154 | 0.0005493178 | 6.9375 | 7.023438 | 0.08593755 | 9.999995 |
| 9 | 6.980469 | 6.980469 | 0.0003814705 | 0.0003814686 | 6.980469 | 7.023438 | 0.0429688 | 3.999997 |
| 10 | 7.001953 | 7.001953 | 3.814617e-06 | 3.814813e-06 | 6.980469 | 7.001953 | 0.02148442 | 3.999993 |
| 11 | 6.991211 | 6.991211 | 7.724798e-05 | 7.72471e-05 | 6.991211 | 7.001953 | 0.01074224 | 3.999986 |
| 12 | 6.996582 | 6.996582 | 1.168265e-05 | 1.168231e-05 | 6.996582 | 7.001953 | 0.005371144 | 5.999953 |
| 13 | 6.999268 | 6.999268 | 5.364718e-07 | 5.363985e-07 | 6.999268 | 7.001953 | 0.002685597 | 9.999832 |
| 14 | 7.00061 | 7.00061 | 3.725041e-07 | 3.725651e-07 | 6.999268 | 7.00061 | 0.001342823 | 3.999888 |
| 15 | 6.999939 | 6.999939 | 3.727788e-09 | 3.721685e-09 | 6.999939 | 7.00061 | 0.0006714367 | 3.999777 |
| 16 | 7.000275 | 7.000275 | 7.542589e-08 | 7.545336e-08 | 6.999939 | 7.000275 | 0.0003357434 | 3.999553 |
| 17 | 7.000107 | 7.000107 | 1.140433e-08 | 1.141501e-08 | 6.999939 | 7.000107 | 0.0001678967 | 5.998511 |
| 18 | 7.000023 | 7.000023 | 5.22933e-10 | 5.252223e-10 | 6.999939 | 7.000023 | 8.397334e-05 | 9.994641 |
| 19 | 6.999981 | 6.999981 | 3.645786e-10 | 3.626717e-10 | 6.999981 | 7.000023 | 4.201167e-05 | 3.99643 |
| 20 | 7.000002 | 7.000002 | 3.560369e-12 | 3.751559e-12 | 6.999981 | 7.000002 | 2.103083e-05 | 3.992868 |
| 21 | 6.999991 | 6.999991 | 7.402061e-11 | 7.316276e-11 | 6.999991 | 7.000002 | 1.054042e-05 | 3.985769 |
| 22 | 6.999997 | 6.999997 | 1.127828e-11 | 1.094495e-11 | 6.999997 | 7.000002 | 5.295209e-06 | 5.952788 |
| 23 | 6.999999 | 6.999999 | 5.412696e-13 | 4.701986e-13 | 6.999999 | 7.000002 | 2.672604e-06 | 9.831625 |
| 24 | 7.000001 | 7.000001 | 3.31306e-13 | 3.913652e-13 | 6.999999 | 7.000001 | 1.361302e-06 | 3.889811 |
| 25 | 7 | 7 | 6.409474e-15 | 9.035548e-16 | 7 | 7.000001 | 7.056511e-07 | 3.78743 |
| 26 | 7 | 7 | 6.138817e-14 | 8.86648e-14 | 7 | 7 | 3.778255e-07 | 3.602991 |
| 27 | 7 | 7 | 7.031423e-15 | 1.791678e-14 | 7 | 7 | 2.139128e-07 | 4.831299 |
| 28 | 7 | 7 | 3.599357e-18 | 2.693319e-15 | 7 | 7 | 1.319564e-07 | 6.589782 |
| 29 | 7 | 7 | 1.527324e-15 | 1.192247e-16 | 7 | 7 | 9.097819e-08 | 2.351253 |

* 1. Метод золотого сечения

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 1 | 3.193496 | 6.403252 | 14.48948 | 0.3561079 | -2 | 11.59675 | 13.59675 | 1.618034 |
| 2 | 6.403252 | 8.386991 | 0.3561079 | 1.923744 | 3.193496 | 11.59675 | 8.403252 | 2.618034 |
| 3 | 5.177234 | 6.403252 | 3.322475 | 0.3561079 | 3.193496 | 8.386991 | 5.193496 | 4.236068 |
| 4 | 6.403252 | 7.160973 | 0.3561079 | 0.02591232 | 5.177234 | 8.386991 | 3.209757 | 6.854102 |
| 5 | 7.160973 | 7.62927 | 0.02591232 | 0.395981 | 6.403252 | 8.386991 | 1.983739 | 11.09017 |
| 6 | 6.871549 | 7.160973 | 0.01649955 | 0.02591232 | 6.403252 | 7.62927 | 1.226018 | 17.94427 |
| 7 | 6.692676 | 6.871549 | 0.09444814 | 0.01649955 | 6.403252 | 7.160973 | 0.7577208 | 29.03444 |
| 8 | 6.871549 | 6.982099 | 0.01649955 | 0.0003204309 | 6.692676 | 7.160973 | 0.4682972 | 46.97871 |
| 9 | 6.982099 | 7.050423 | 0.0003204309 | 0.002542485 | 6.871549 | 7.160973 | 0.2894236 | 76.01316 |
| 10 | 6.939873 | 6.982099 | 0.003615246 | 0.0003204309 | 6.871549 | 7.050423 | 0.1788736 | 122.9919 |
| 11 | 6.982099 | 7.008197 | 0.0003204309 | 6.718631e-05 | 6.939873 | 7.050423 | 0.11055 | 199.005 |
| 12 | 7.008197 | 7.024326 | 6.718631e-05 | 0.0005917421 | 6.982099 | 7.050423 | 0.06832364 | 321.9969 |
| 13 | 6.998228 | 7.008197 | 3.138422e-06 | 6.718631e-05 | 6.982099 | 7.024326 | 0.04222633 | 521.0019 |
| 14 | 6.992068 | 6.998228 | 6.292135e-05 | 3.138422e-06 | 6.982099 | 7.008197 | 0.02609731 | 842.9988 |
| 15 | 6.998228 | 7.002036 | 3.138422e-06 | 4.145243e-06 | 6.992068 | 7.008197 | 0.01612902 | 1364.001 |
| 16 | 6.995875 | 6.998228 | 1.701358e-05 | 3.138422e-06 | 6.992068 | 7.002036 | 0.009968285 | 2207 |
| 17 | 6.998228 | 6.999683 | 3.138422e-06 | 1.006196e-07 | 6.995875 | 7.002036 | 0.006160739 | 3571 |
| 18 | 6.999683 | 7.000582 | 1.006196e-07 | 3.382978e-07 | 6.998228 | 7.002036 | 0.003807546 | 5778 |
| 19 | 6.999127 | 6.999683 | 7.616391e-07 | 1.006196e-07 | 6.998228 | 7.000582 | 0.002353193 | 9349 |
| 20 | 6.999683 | 7.000026 | 1.006196e-07 | 6.822696e-10 | 6.999127 | 7.000582 | 0.001454353 | 15127 |
| 21 | 7.000026 | 7.000238 | 6.822696e-10 | 5.679049e-08 | 6.999683 | 7.000582 | 0.0008988397 | 24476 |
| 22 | 6.999895 | 7.000026 | 1.102892e-08 | 6.822696e-10 | 6.999683 | 7.000238 | 0.0005555135 | 39603 |
| 23 | 7.000026 | 7.000107 | 6.822696e-10 | 1.148511e-08 | 6.999895 | 7.000238 | 0.0003433262 | 64079 |
| 24 | 6.999976 | 7.000026 | 5.745766e-10 | 6.822696e-10 | 6.999895 | 7.000107 | 0.0002121873 | 103682 |
| 25 | 6.999945 | 6.999976 | 3.017089e-09 | 5.745766e-10 | 6.999895 | 7.000026 | 0.0001311389 | 167761 |
| 26 | 6.999976 | 6.999995 | 5.745766e-10 | 2.340058e-11 | 6.999945 | 7.000026 | 8.104832e-05 | 271443 |
| 27 | 6.999995 | 7.000007 | 2.340058e-11 | 4.882343e-11 | 6.999976 | 7.000026 | 5.009062e-05 | 439204 |
| 28 | 6.999988 | 6.999995 | 1.475141e-10 | 2.340058e-11 | 6.999976 | 7.000007 | 3.09577e-05 | 710647 |
| 29 | 6.999995 | 7 | 2.340058e-11 | 1.028782e-13 | 6.999988 | 7.000007 | 1.913291e-05 | 1149851 |
| 30 | 7 | 7.000002 | 1.028782e-13 | 6.104399e-12 | 6.999995 | 7.000007 | 1.182479e-05 | 1860498 |
| 31 | 6.999998 | 7 | 4.185952e-12 | 1.028782e-13 | 6.999995 | 7.000002 | 7.308123e-06 | 3010349 |
| 32 | 7 | 7.000001 | 1.028782e-13 | 5.55762e-13 | 6.999998 | 7.000002 | 4.516668e-06 | 4870847 |
| 33 | 6.999999 | 7 | 9.598498e-13 | 1.028782e-13 | 6.999998 | 7.000001 | 2.791454e-06 | 7881196 |
| 34 | 7 | 7 | 1.028782e-13 | 7.485966e-15 | 6.999999 | 7.000001 | 1.725214e-06 | 1.275204e+07 |
| 35 | 7 | 7 | 7.485966e-15 | 1.143973e-13 | 7 | 7.000001 | 1.066241e-06 | 2.063324e+07 |
| 36 | 7 | 7 | 4.766653e-15 | 7.485966e-15 | 7 | 7 | 6.58973e-07 | 3.338528e+07 |
| 37 | 7 | 7 | 2.728569e-14 | 4.766653e-15 | 7 | 7 | 4.072677e-07 | 5.401852e+07 |
| 38 | 7 | 7 | 4.766653e-15 | 9.257121e-17 | 7 | 7 | 2.517053e-07 | 8.74038e+07 |
| 39 | 7 | 7 | 9.257121e-17 | 7.345138e-16 | 7 | 7 | 1.555624e-07 | 1.414223e+08 |
| 40 | 7 | 7 | 1.04443e-15 | 9.257121e-17 | 7 | 7 | 9.614287e-08 | 2.288261e+08 |

* 1. Метод Фибоначчи

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 1 | 3.193496 | 6.403252 | 14.48948 | 0.3561079 | -2 | 11.59675 | 13.59675 | 1.618034 |
| 2 | 6.403252 | 8.386991 | 0.3561079 | 1.923744 | 3.193496 | 11.59675 | 8.403252 | 2.618034 |
| 3 | 5.177234 | 6.403252 | 3.322475 | 0.3561079 | 3.193496 | 8.386991 | 5.193496 | 4.236068 |
| 4 | 6.403252 | 7.160973 | 0.3561079 | 0.02591232 | 5.177234 | 8.386991 | 3.209757 | 6.854102 |
| 5 | 7.160973 | 7.62927 | 0.02591232 | 0.395981 | 6.403252 | 8.386991 | 1.983739 | 11.09017 |
| 6 | 6.871549 | 7.160973 | 0.01649955 | 0.02591232 | 6.403252 | 7.62927 | 1.226018 | 17.94427 |
| 7 | 6.692676 | 6.871549 | 0.09444814 | 0.01649955 | 6.403252 | 7.160973 | 0.7577208 | 29.03444 |
| 8 | 6.871549 | 6.982099 | 0.01649955 | 0.0003204309 | 6.692676 | 7.160973 | 0.4682972 | 46.97871 |
| 9 | 6.982099 | 7.050423 | 0.0003204309 | 0.002542485 | 6.871549 | 7.160973 | 0.2894236 | 76.01316 |
| 10 | 6.939873 | 6.982099 | 0.003615246 | 0.0003204309 | 6.871549 | 7.050423 | 0.1788736 | 122.9919 |
| 11 | 6.982099 | 7.008197 | 0.0003204309 | 6.718631e-05 | 6.939873 | 7.050423 | 0.11055 | 199.005 |
| 12 | 7.008197 | 7.024326 | 6.718631e-05 | 0.0005917421 | 6.982099 | 7.050423 | 0.06832364 | 321.9969 |
| 13 | 6.998228 | 7.008197 | 3.138422e-06 | 6.718631e-05 | 6.982099 | 7.024326 | 0.04222633 | 521.0019 |
| 14 | 6.992068 | 6.998228 | 6.292135e-05 | 3.138422e-06 | 6.982099 | 7.008197 | 0.02609731 | 842.9988 |
| 15 | 6.998228 | 7.002036 | 3.138422e-06 | 4.145243e-06 | 6.992068 | 7.008197 | 0.01612902 | 1364.001 |
| 16 | 6.995875 | 6.998228 | 1.701358e-05 | 3.138422e-06 | 6.992068 | 7.002036 | 0.009968285 | 2207 |
| 17 | 6.998228 | 6.999683 | 3.138422e-06 | 1.006196e-07 | 6.995875 | 7.002036 | 0.006160739 | 3571 |
| 18 | 6.999683 | 7.000582 | 1.006196e-07 | 3.382978e-07 | 6.998228 | 7.002036 | 0.003807546 | 5778 |
| 19 | 6.999127 | 6.999683 | 7.616391e-07 | 1.006196e-07 | 6.998228 | 7.000582 | 0.002353193 | 9349 |
| 20 | 6.999683 | 7.000026 | 1.006196e-07 | 6.822697e-10 | 6.999127 | 7.000582 | 0.001454353 | 15127 |
| 21 | 7.000026 | 7.000238 | 6.822697e-10 | 5.679049e-08 | 6.999683 | 7.000582 | 0.0008988397 | 24476 |
| 22 | 6.999895 | 7.000026 | 1.102892e-08 | 6.822697e-10 | 6.999683 | 7.000238 | 0.0005555135 | 39603 |
| 23 | 7.000026 | 7.000107 | 6.822697e-10 | 1.148511e-08 | 6.999895 | 7.000238 | 0.0003433262 | 64079 |
| 24 | 6.999976 | 7.000026 | 5.745778e-10 | 6.822697e-10 | 6.999895 | 7.000107 | 0.0002121873 | 103682 |
| 25 | 6.999945 | 6.999976 | 3.017087e-09 | 5.745778e-10 | 6.999895 | 7.000026 | 0.000131139 | 167761 |
| 26 | 6.999976 | 6.999995 | 5.745778e-10 | 2.340013e-11 | 6.999945 | 7.000026 | 8.104831e-05 | 271443.1 |
| 27 | 6.999995 | 7.000007 | 2.340013e-11 | 4.882249e-11 | 6.999976 | 7.000026 | 5.009065e-05 | 439203.8 |
| 28 | 6.999988 | 6.999995 | 1.475174e-10 | 2.340013e-11 | 6.999976 | 7.000007 | 3.095766e-05 | 710648 |
| 29 | 6.999995 | 7 | 2.340013e-11 | 1.030398e-13 | 6.999988 | 7.000007 | 1.913298e-05 | 1149847 |
| 30 | 7 | 7.000002 | 1.030398e-13 | 6.105542e-12 | 6.999995 | 7.000007 | 1.182468e-05 | 1860516 |
| 31 | 6.999998 | 7 | 4.183785e-12 | 1.030398e-13 | 6.999995 | 7.000002 | 7.308307e-06 | 3010273 |
| 32 | 7 | 7.000001 | 1.030398e-13 | 5.572733e-13 | 6.999998 | 7.000002 | 4.51637e-06 | 4871169 |
| 33 | 6.999999 | 7 | 9.563367e-13 | 1.030398e-13 | 6.999998 | 7.000001 | 2.791938e-06 | 7879832 |
| 34 | 7 | 7 | 1.030398e-13 | 8.024735e-15 | 6.999999 | 7.000001 | 1.724432e-06 | 1.275782e+07 |
| 35 | 7 | 7 | 8.024735e-15 | 1.128478e-13 | 7 | 7.000001 | 1.067506e-06 | 2.060879e+07 |
| 36 | 7 | 7 | 5.572733e-15 | 8.024735e-15 | 7 | 7 | 6.569265e-07 | 3.348929e+07 |
| 37 | 7 | 7 | 2.457575e-14 | 5.572733e-15 | 7 | 7 | 4.105791e-07 | 5.358286e+07 |
| 38 | 7 | 7 | 5.572733e-15 | 5.57273e-17 | 7 | 7 | 2.463474e-07 | 8.930476e+07 |

1. График зависимости количества вычислений функции от логарифма точности
2. Таблица, показывающая процесс поиска интервала, содержащего минимум

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 1 | 8649 | 100 |
| 2 | 8464 | 99 |
| 3 | 8100 | 97 |
| 4 | 7396 | 93 |
| 5 | 6084 | 85 |
| 6 | 3844 | 69 |
| 7 | 900 | 37 |
| 8 | 1156 | -27 |
| Интервал, содержащий минимум: [-27; 69] | | |

1. Выводы

При сравнении реализованных методов поиска минимума функции на отрезке можно  
сказать, что метод дихотомии, хотя и сошёлся до необходимой точности быстрее прочих  
методов в смысле количества итераций, но потребовал значительно большего количества вычислений исследуемой функции, что при трудоёмкости её вычисления делает этот метод неэффективным. Методы золотого сечения и Фибоначчи похожи по скорости сходимости и количеству вычислений целевой функции, но последний несколько более эффективен.

Графики зависимости количества вычислений целевой функции от логарифма задаваемой точности показывают линейную зависимость.

Реализованный метод поиска интервала, содержащего минимум исследуемой функции, даже при заметном удалении от точного значения минимума начальной точки, сходится достаточно быстро в силу удвоения шага, но с другой стороны, это значительно увеличивает получаемый интервал.

1. Текст программы
   1. functions.h

#pragma once

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <iomanip>

class FunctionMinimization

{

private:

double a;

double b;

double eps;

double function(double x);

double nthFibonacciNumber(int n);

double smallestFibonacciNumber();

public:

FunctionMinimization(std::string file\_name);

double dichotomyMethod();

double goldenRatioMethod();

double fibonacciMethod();

void intervalSearch(double x0, double delta);

};

* 1. functions.cpp

#include "functions.h"

double FunctionMinimization::function(double x)

{

return pow(x - 7, 2);

}

FunctionMinimization::FunctionMinimization(std::string file\_name)

{

std::ifstream input\_stream(file\_name);

if (!input\_stream)

{

throw std::string("Couldn\'t open file: " + file\_name);

}

double a0;

double b0;

double eps0;

input\_stream >> a0 >> b0 >> eps0;

if (a0 > b0)

{

throw std::string("Incorrect parameters: a must be less than b");

}

if (eps0 <= 0)

{

throw std::string("Incorrect parameters: epsilon must be positive");

}

a = a0;

b = b0;

eps = eps0;

}

double FunctionMinimization::dichotomyMethod()

{

double minimum = 0;

double delta = eps / 2;

double left\_edge = a;

double right\_edge = b;

double left\_edge\_prev=a;

double right\_edge\_prev=b;

double x1;

double x2;

double f1;

double f2;

int iterations=0;

std::ofstream output\_stream("dichotomy.txt");

output\_stream << std::setprecision(7);

for(int i=1;right\_edge - left\_edge >= eps;i++)

{

x1 = (left\_edge + right\_edge - delta) / 2;

x2 = (left\_edge + right\_edge + delta) / 2;

f1 = function(x1);

f2 = function(x2);

if (f1 >= f2)

{

left\_edge\_prev = left\_edge;

left\_edge = x1;

}

else

{

right\_edge\_prev = right\_edge;

right\_edge = x2;

}

output\_stream << i << "\t" << x1 << "\t" << x2 << "\t" << f1 << "\t" \

<< f2 << "\t" << left\_edge << "\t" << right\_edge << "\t" \

<< right\_edge - left\_edge << "\t" << (right\_edge\_prev - left\_edge\_prev) / (right\_edge - left\_edge) << "\n";

iterations = i;

}

minimum = (right\_edge + left\_edge) / 2;

output\_stream.close();

std::cout << "eps: " << eps << " iterations:" << iterations << "\n";

return minimum;

}

double FunctionMinimization::goldenRatioMethod()

{

double minimum = 0;

double delta = eps / 2;

double left\_edge = a;

double right\_edge = b;

double left\_edge\_prev = a;

double right\_edge\_prev = b;

double x1;

double x2;

double f1;

double f2;

const double sqrt5 = sqrt(5);

int iterations = 0;

x1 = left\_edge + (right\_edge - left\_edge) \* (3 - sqrt5) / 2;

x2 = left\_edge + (right\_edge - left\_edge) \* (sqrt5 - 1) / 2;

f1 = function(x1);

f2 = function(x2);

std::ofstream output\_stream("golden\_ratio.txt");

output\_stream << std::setprecision(7);

for (int i=1;right\_edge - left\_edge >= eps;i++)

{

if (f1 >= f2)

{

left\_edge = x1;

x1 = x2;

f1 = f2;

x2 = left\_edge + (right\_edge - left\_edge) \* (sqrt5 - 1) / 2;

f2 = function(x2);

}

else

{

right\_edge = x2;

x2 = x1;

f2 = f1;

x1 = left\_edge + (right\_edge - left\_edge) \* (3 - sqrt5) / 2;

f1 = function(x1);

}

output\_stream << i << "\t" << x1 << "\t" << x2 << "\t" << f1 << "\t" \

<< f2 << "\t" << left\_edge << "\t" << right\_edge << "\t" \

<< right\_edge - left\_edge << "\t" << (right\_edge\_prev - left\_edge\_prev) / (right\_edge - left\_edge) << "\n";

iterations = i;

}

minimum = (right\_edge + left\_edge) / 2;

output\_stream.close();

std::cout << "eps: " << eps << " iterations:" << iterations << "\n";

return minimum;

}

double FunctionMinimization::nthFibonacciNumber(int n)

{

double fibonacci\_number = 0;

const double sqrt5 = sqrt(5);

fibonacci\_number = (pow((1+sqrt5)/2, n) - pow((1-sqrt5)/2, n)) / sqrt5;

return fibonacci\_number;

}

double FunctionMinimization::smallestFibonacciNumber()

{

const double sqrt5 = sqrt(5);

double result = 1;

double number\_to\_compare = (b-a) / eps;

for (int n = 2;result <= number\_to\_compare;n++)

{

result = (pow((1 + sqrt5) / 2, n) - pow((1 - sqrt5) / 2, n)) / sqrt5;

}

return result;

}

double FunctionMinimization::fibonacciMethod()

{

double minimum = 0;

double delta = eps / 2;

double left\_edge = a;

double right\_edge = b;

double left\_edge\_prev = a;

double right\_edge\_prev = b;

double x1;

double x2;

double f1;

double f2;

const double sqrt5 = sqrt(5);

double fib\_n\_plus\_2 = 0;

double number\_to\_compare = (b - a) / eps;

int n = 0;

int iterations = 0;

do

{

n++;

fib\_n\_plus\_2 = (pow((1 + sqrt5) / 2, n) - pow((1 - sqrt5) / 2, n)) / sqrt5;

} while (fib\_n\_plus\_2 <= number\_to\_compare);

n -= 2;

x1 = a + (b - a) \* (nthFibonacciNumber(n)/ fib\_n\_plus\_2);

x2 = a+b-x1;

f1 = function(x1);

f2 = function(x2);

std::ofstream output\_stream("fibonacci.txt");

output\_stream << std::setprecision(7);

for (int k = 2,i=1;k<n;k++,i++)

{

if (f1 >= f2)

{

left\_edge = x1;

x1 = x2;

f1 = f2;

x2 = left\_edge + (b - a) \* (nthFibonacciNumber(n-k+2) / fib\_n\_plus\_2);

f2 = function(x2);

}

else

{

right\_edge = x2;

x2 = x1;

f2 = f1;

x1 = left\_edge + (b - a) \* (nthFibonacciNumber(n-k+1) / fib\_n\_plus\_2);

f1 = function(x1);

}

output\_stream << i << "\t" << x1 << "\t" << x2 << "\t" << f1 << "\t" \

<< f2 << "\t" << left\_edge << "\t" << right\_edge << "\t" \

<< right\_edge - left\_edge << "\t" << (right\_edge\_prev - left\_edge\_prev) / (right\_edge - left\_edge) << "\n";

iterations = i;

}

minimum = (right\_edge + left\_edge) / 2;

output\_stream.close();

std::cout << "eps: " << eps << " iterations:" << iterations << "\n";

return minimum;

}

void FunctionMinimization::intervalSearch(double x0, double delta)

{

std::ofstream output\_stream("interval.txt");

output\_stream << std::setprecision(7);

//if (function(x0 + delta) >= function(x0 - delta))

//{

// do

// {

// output\_stream << x0 << "\t" << function(x0) <<"\n";

// x0 -= delta;

// } while (function(x0 + delta) >= function(x0 - delta));

// std::cout << "a:" << x0 << " b:" << x0 + delta << "\n";

//}

//else

//{

// do

// {

// output\_stream << x0 << "\t" << function(x0) << "\n";

// x0 += delta;

// } while (function(x0 + delta) < function(x0 - delta));

// std::cout << "a:" << x0 - delta << " b:" << x0 << "\n";

//}

int k=1;

double h=0;

output\_stream << x0 << "\t" << function(x0) << "\n";

if (function(x0) > function(x0 + delta))

{

x0 += delta;

h = delta;

output\_stream << x0 << "\t" << function(x0) << "\n";

}

else

{

x0 -= delta;

h = -delta;

output\_stream << x0 << "\t" << function(x0) << "\n";

}

h \*= 2;

x0 += h;

output\_stream << x0 << "\t" << function(x0) << "\n";

while (function(x0 - h) > function(x0))

{

h \*= 2;

x0 += h;

k += 1;

output\_stream << x0 << "\t" << function(x0) <<"\n";

}

if (h > 0)

{

output\_stream << "a:" << x0-h << " b:" << x0 + h << "\n";

}

else

{

output\_stream << "a:" << x0+h << " b:" << x0 - h << "\n";

}

output\_stream.close();

}

* 1. main.cpp

#include "functions.h"

int main()

{

FunctionMinimization f\_m("parameters.txt");

std::cout << f\_m.dichotomyMethod();

std::cout << "\n";

std::cout << f\_m.goldenRatioMethod();

std::cout << "\n";

std::cout << f\_m.fibonacciMethod();

f\_m.intervalSearch(-1000, 1);

std::cout << "\n";

}