|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
| Практическое задание № 1 | | |
| по дисциплине «Методы оптимизации» | | |
| **Методы одномерного поиска** | | |
|  | | |
|  | Бригада 1 | исакин даниил |
| Группа ПМ-13 | вострецова екатерина |
| Вариант 1 |  |
|  |  |
|  |  |
|  |  |
| Преподаватели | фИЛИППОВА елЕНА ВЛАДИМИРОВНА |
|  |  |
| Новосибирск, 2024 | | |

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**
2. Реализовать алгоритм поиска минимума унимодальной функции на заданном отрезке ***методом дихотомии***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 1 | 9,00000000 | 9,00000001 | 63,99999992 | 64,00000008 | -2,00000000 | 20,00000000 | 2,20E+01 |  |
| 2 | 3,50000000 | 3,50000001 | 6,24999999 | 6,25000004 | -2,00000000 | 9,00000001 | 1,10E+01 | 2,00E+00 |
| 3 | 0,75000000 | 0,75000001 | 0,06250000 | 0,06250000 | -2,00000000 | 3,50000001 | 5,50E+00 | 2,00E+00 |
| 4 | 2,12500000 | 2,12500001 | 1,26562500 | 1,26562502 | 0,75000000 | 3,50000001 | 2,75E+00 | 2,00E+00 |
| 5 | 1,43750000 | 1,43750001 | 0,19140625 | 0,19140626 | 0,75000000 | 2,12500001 | 1,38E+00 | 2,00E+00 |
| 6 | 1,09375000 | 1,09375001 | 0,00878906 | 0,00878906 | 0,75000000 | 1,43750001 | 6,88E-01 | 2,00E+00 |
| 7 | 0,92187500 | 0,92187501 | 0,00610352 | 0,00610351 | 0,75000000 | 1,09375001 | 3,44E-01 | 2,00E+00 |
| 8 | 1,00781250 | 1,00781251 | 0,00006104 | 0,00006104 | 0,92187500 | 1,09375001 | 1,72E-01 | 2,00E+00 |
| 9 | 0,96484375 | 0,96484376 | 0,00123596 | 0,00123596 | 0,92187500 | 1,00781251 | 8,59E-02 | 2,00E+00 |
| 10 | 0,98632812 | 0,98632813 | 0,00018692 | 0,00018692 | 0,96484375 | 1,00781251 | 4,30E-02 | 2,00E+00 |
| 11 | 0,99707031 | 0,99707032 | 0,00000858 | 0,00000858 | 0,98632812 | 1,00781251 | 2,15E-02 | 2,00E+00 |
| 12 | 1,00244140 | 1,00244141 | 0,00000596 | 0,00000596 | 0,99707031 | 1,00781251 | 1,07E-02 | 2,00E+00 |
| 13 | 0,99975586 | 0,99975587 | 0,00000006 | 0,00000006 | 0,99707031 | 1,00244141 | 5,37E-03 | 2,00E+00 |
| 14 | 1,00109863 | 1,00109864 | 0,00000121 | 0,00000121 | 0,99975586 | 1,00244141 | 2,69E-03 | 2,00E+00 |
| 15 | 1,00042724 | 1,00042725 | 0,00000018 | 0,00000018 | 0,99975586 | 1,00109864 | 1,34E-03 | 2,00E+00 |
| 16 | 1,00009155 | 1,00009156 | 0,00000001 | 0,00000001 | 0,99975586 | 1,00042725 | 6,71E-04 | 2,00E+00 |
| 17 | 0,99992370 | 0,99992371 | 0,00000001 | 0,00000001 | 0,99975586 | 1,00009156 | 3,36E-04 | 2,00E+00 |
| 18 | 1,00000763 | 1,00000764 | 0,00000000 | 0,00000000 | 0,99992370 | 1,00009156 | 1,68E-04 | 2,00E+00 |
| 19 | 0,99996567 | 0,99996568 | 0,00000000 | 0,00000000 | 0,99992370 | 1,00000764 | 8,39E-05 | 2,00E+00 |
| 20 | 0,99998665 | 0,99998666 | 0,00000000 | 0,00000000 | 0,99996567 | 1,00000764 | 4,20E-05 | 2,00E+00 |
| 21 | 0,99999714 | 0,99999715 | 0,00000000 | 0,00000000 | 0,99998665 | 1,00000764 | 2,10E-05 | 2,00E+00 |
| 22 | 1,00000238 | 1,00000239 | 0,00000000 | 0,00000000 | 0,99999714 | 1,00000764 | 1,05E-05 | 2,00E+00 |
| 23 | 0,99999976 | 0,99999977 | 0,00000000 | 0,00000000 | 0,99999714 | 1,00000239 | 5,26E-06 | 2,00E+00 |
| 24 | 1,00000107 | 1,00000108 | 0,00000000 | 0,00000000 | 0,99999976 | 1,00000239 | 2,63E-06 | 2,00E+00 |
| 25 | 1,00000042 | 1,00000043 | 0,00000000 | 0,00000000 | 0,99999976 | 1,00000108 | 1,32E-06 | 1,99E+00 |
| 26 | 1,00000009 | 1,00000010 | 0,00000000 | 0,00000000 | 0,99999976 | 1,00000043 | 6,66E-07 | 1,98E+00 |
| 27 | 0,99999992 | 0,99999993 | 0,00000000 | 0,00000000 | 0,99999976 | 1,00000010 | 3,38E-07 | 1,97E+00 |
| 28 | 1,00000001 | 1,00000002 | 0,00000000 | 0,00000000 | 0,99999992 | 1,00000010 | 1,74E-07 | 1,94E+00 |

1. Реализовать алгоритм поиска минимума унимодальной функции на заданном отрезке ***методом золотого сечения***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 0 | 6,4032522 | 11,5967478 | 29,1951349 | 112,2910629 | -2,0000000 | 20,0000000 | 2,20E+01 |  |
| 1 | 3,1934955 | 6,4032522 | 4,8114225 | 29,1951349 | -2,0000000 | 11,5967478 | 1,36E+01 | 1,61803399 |
| 2 | 1,2097567 | 3,1934955 | 0,0439979 | 4,8114225 | -2,0000000 | 6,4032522 | 8,40E+00 | 1,61803399 |
| 3 | -0,0162612 | 1,2097567 | 1,0327869 | 0,0439979 | -2,0000000 | 3,1934955 | 5,19E+00 | 1,61803399 |
| 4 | 1,2097567 | 1,9674775 | 0,0439979 | 0,9360128 | -0,0162612 | 3,1934955 | 3,21E+00 | 1,61803399 |
| 5 | 0,7414595 | 1,2097567 | 0,0668432 | 0,0439979 | -0,0162612 | 1,9674775 | 1,98E+00 | 1,61803399 |
| 6 | 1,2097567 | 1,4991803 | 0,0439979 | 0,2491810 | 0,7414595 | 1,9674775 | 1,23E+00 | 1,61803399 |
| 7 | 1,0308831 | 1,2097567 | 0,0009538 | 0,0439979 | 0,7414595 | 1,4991803 | 7,58E-01 | 1,61803399 |
| 8 | 0,9203332 | 1,0308831 | 0,0063468 | 0,0009538 | 0,7414595 | 1,2097567 | 4,68E-01 | 1,61803399 |
| 9 | 1,0308831 | 1,0992068 | 0,0009538 | 0,0098420 | 0,9203332 | 1,2097567 | 2,89E-01 | 1,61803399 |
| 10 | 0,9886568 | 1,0308831 | 0,0001287 | 0,0009538 | 0,9203332 | 1,0992068 | 1,79E-01 | 1,61803399 |
| 11 | 0,9625595 | 0,9886568 | 0,0014018 | 0,0001287 | 0,9203332 | 1,0308831 | 1,11E-01 | 1,61803399 |
| 12 | 0,9886568 | 1,0047858 | 0,0001287 | 0,0000229 | 0,9625595 | 1,0308831 | 6,83E-02 | 1,61803399 |
| 13 | 1,0047858 | 1,0147541 | 0,0000229 | 0,0002177 | 0,9886568 | 1,0308831 | 4,22E-02 | 1,61803399 |
| 14 | 0,9986251 | 1,0047858 | 0,0000019 | 0,0000229 | 0,9886568 | 1,0147541 | 2,61E-02 | 1,61803399 |
| 15 | 0,9948175 | 0,9986251 | 0,0000269 | 0,0000019 | 0,9886568 | 1,0047858 | 1,61E-02 | 1,61803399 |
| 16 | 0,9986251 | 1,0009783 | 0,0000019 | 0,0000010 | 0,9948175 | 1,0047858 | 9,97E-03 | 1,61803399 |
| 17 | 1,0009783 | 1,0024326 | 0,0000010 | 0,0000059 | 0,9986251 | 1,0047858 | 6,16E-03 | 1,61803399 |
| 18 | 1,0000794 | 1,0009783 | 0,0000000 | 0,0000010 | 0,9986251 | 1,0024326 | 3,81E-03 | 1,61803399 |
| 19 | 0,9995239 | 1,0000794 | 0,0000002 | 0,0000000 | 0,9986251 | 1,0009783 | 2,35E-03 | 1,61803399 |
| 20 | 1,0000794 | 1,0004228 | 0,0000000 | 0,0000002 | 0,9995239 | 1,0009783 | 1,45E-03 | 1,61803399 |
| 21 | 0,9998672 | 1,0000794 | 0,0000000 | 0,0000000 | 0,9995239 | 1,0004228 | 8,99E-04 | 1,61803399 |
| 22 | 1,0000794 | 1,0002106 | 0,0000000 | 0,0000000 | 0,9998672 | 1,0004228 | 5,56E-04 | 1,61803399 |
| 23 | 0,9999984 | 1,0000794 | 0,0000000 | 0,0000000 | 0,9998672 | 1,0002106 | 3,43E-04 | 1,61803399 |
| 24 | 0,9999483 | 0,9999984 | 0,0000000 | 0,0000000 | 0,9998672 | 1,0000794 | 2,12E-04 | 1,61803399 |
| 25 | 0,9999984 | 1,0000293 | 0,0000000 | 0,0000000 | 0,9999483 | 1,0000794 | 1,31E-04 | 1,61803399 |
| 26 | 0,9999793 | 0,9999984 | 0,0000000 | 0,0000000 | 0,9999483 | 1,0000293 | 8,10E-05 | 1,61803399 |
| 27 | 0,9999984 | 1,0000102 | 0,0000000 | 0,0000000 | 0,9999793 | 1,0000293 | 5,01E-05 | 1,61803399 |
| 28 | 0,9999911 | 0,9999984 | 0,0000000 | 0,0000000 | 0,9999793 | 1,0000102 | 3,10E-05 | 1,61803399 |
| 29 | 0,9999984 | 1,0000029 | 0,0000000 | 0,0000000 | 0,9999911 | 1,0000102 | 1,91E-05 | 1,61803399 |
| 30 | 0,9999956 | 0,9999984 | 0,0000000 | 0,0000000 | 0,9999911 | 1,0000029 | 1,18E-05 | 1,61803399 |
| 31 | 0,9999984 | 1,0000001 | 0,0000000 | 0,0000000 | 0,9999956 | 1,0000029 | 7,31E-06 | 1,61803399 |
| 32 | 1,0000001 | 1,0000012 | 0,0000000 | 0,0000000 | 0,9999984 | 1,0000029 | 4,52E-06 | 1,61803399 |
| 33 | 0,9999995 | 1,0000001 | 0,0000000 | 0,0000000 | 0,9999984 | 1,0000012 | 2,79E-06 | 1,61803399 |
| 34 | 1,0000001 | 1,0000005 | 0,0000000 | 0,0000000 | 0,9999995 | 1,0000012 | 1,73E-06 | 1,61803399 |
| 35 | 0,9999999 | 1,0000001 | 0,0000000 | 0,0000000 | 0,9999995 | 1,0000005 | 1,07E-06 | 1,61803400 |
| 36 | 1,0000001 | 1,0000003 | 0,0000000 | 0,0000000 | 0,9999999 | 1,0000005 | 6,59E-07 | 1,61803400 |
| 37 | 1,0000000 | 1,0000001 | 0,0000000 | 0,0000000 | 0,9999999 | 1,0000003 | 4,07E-07 | 1,61803398 |
| 38 | 1,0000000 | 1,0000000 | 0,0000000 | 0,0000000 | 0,9999999 | 1,0000001 | 2,52E-07 | 1,61803401 |
| 39 | 1,0000000 | 1,0000001 | 0,0000000 | 0,0000000 | 1,0000000 | 1,0000001 | 1,56E-07 | 1,61803401 |

1. Реализовать алгоритм поиска минимума унимодальной функции на заданном отрезке ***методом Фибоначчи***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| 0 | 6,4032522 | 11,5967478 | 29,1951349 | 112,2910629 | -2,0000000 | 20,0000000 | 2,20E+01 |  |
| 1 | 3,1934955 | 6,4032522 | 4,8114225 | 29,1951349 | -2,0000000 | 11,5967478 | 1,36E+01 | 1,6180E+00 |
| 2 | 1,2097567 | 3,1934955 | 0,0439979 | 4,8114225 | -2,0000000 | 6,4032522 | 8,40E+00 | 1,6180E+00 |
| 3 | -0,0162612 | 1,2097567 | 1,0327869 | 0,0439979 | -2,0000000 | 3,1934955 | 5,19E+00 | 1,6180E+00 |
| 4 | 1,2097567 | 1,9674775 | 0,0439979 | 0,9360128 | -0,0162612 | 3,1934955 | 3,21E+00 | 1,6180E+00 |
| 5 | 0,7414595 | 1,2097567 | 0,0668432 | 0,0439979 | -0,0162612 | 1,9674775 | 1,98E+00 | 1,6180E+00 |
| 6 | 1,2097567 | 1,4991803 | 0,0439979 | 0,2491810 | 0,7414595 | 1,9674775 | 1,23E+00 | 1,6180E+00 |
| 7 | 1,0308831 | 1,2097567 | 0,0009538 | 0,0439979 | 0,7414595 | 1,4991803 | 7,58E-01 | 1,6180E+00 |
| 8 | 0,9203332 | 1,0308831 | 0,0063468 | 0,0009538 | 0,7414595 | 1,2097567 | 4,68E-01 | 1,6180E+00 |
| 9 | 1,0308831 | 1,0992068 | 0,0009538 | 0,0098420 | 0,9203332 | 1,2097567 | 2,89E-01 | 1,6180E+00 |
| 10 | 0,9886568 | 1,0308831 | 0,0001287 | 0,0009538 | 0,9203332 | 1,0992068 | 1,79E-01 | 1,6180E+00 |
| 11 | 0,9625595 | 0,9886568 | 0,0014018 | 0,0001287 | 0,9203332 | 1,0308831 | 1,11E-01 | 1,6180E+00 |
| 12 | 0,9886568 | 1,0047858 | 0,0001287 | 0,0000229 | 0,9625595 | 1,0308831 | 6,83E-02 | 1,6180E+00 |
| 13 | 1,0047858 | 1,0147541 | 0,0000229 | 0,0002177 | 0,9886568 | 1,0308831 | 4,22E-02 | 1,6180E+00 |
| 14 | 0,9986251 | 1,0047858 | 0,0000019 | 0,0000229 | 0,9886568 | 1,0147541 | 2,61E-02 | 1,6180E+00 |
| 15 | 0,9948175 | 0,9986251 | 0,0000269 | 0,0000019 | 0,9886568 | 1,0047858 | 1,61E-02 | 1,6180E+00 |
| 16 | 0,9986251 | 1,0009783 | 0,0000019 | 0,0000010 | 0,9948175 | 1,0047858 | 9,97E-03 | 1,6180E+00 |
| 17 | 1,0009783 | 1,0024326 | 0,0000010 | 0,0000059 | 0,9986251 | 1,0047858 | 6,16E-03 | 1,6180E+00 |
| 18 | 1,0000794 | 1,0009783 | 0,0000000 | 0,0000010 | 0,9986251 | 1,0024326 | 3,81E-03 | 1,6180E+00 |
| 19 | 0,9995239 | 1,0000794 | 0,0000002 | 0,0000000 | 0,9986251 | 1,0009783 | 2,35E-03 | 1,6180E+00 |
| 20 | 1,0000794 | 1,0004228 | 0,0000000 | 0,0000002 | 0,9995239 | 1,0009783 | 1,45E-03 | 1,6180E+00 |
| 21 | 0,9998672 | 1,0000794 | 0,0000000 | 0,0000000 | 0,9995239 | 1,0004228 | 8,99E-04 | 1,6180E+00 |
| 22 | 1,0000794 | 1,0002106 | 0,0000000 | 0,0000000 | 0,9998672 | 1,0004228 | 5,56E-04 | 1,6180E+00 |
| 23 | 0,9999984 | 1,0000794 | 0,0000000 | 0,0000000 | 0,9998672 | 1,0002106 | 3,43E-04 | 1,6180E+00 |
| 24 | 0,9999483 | 0,9999984 | 0,0000000 | 0,0000000 | 0,9998672 | 1,0000794 | 2,12E-04 | 1,6180E+00 |
| 25 | 0,9999984 | 1,0000293 | 0,0000000 | 0,0000000 | 0,9999483 | 1,0000794 | 1,31E-04 | 1,6180E+00 |
| 26 | 0,9999793 | 0,9999984 | 0,0000000 | 0,0000000 | 0,9999483 | 1,0000293 | 8,10E-05 | 1,6180E+00 |
| 27 | 0,9999984 | 1,0000102 | 0,0000000 | 0,0000000 | 0,9999793 | 1,0000293 | 5,01E-05 | 1,6180E+00 |
| 28 | 0,9999911 | 0,9999984 | 0,0000000 | 0,0000000 | 0,9999793 | 1,0000102 | 3,10E-05 | 1,6180E+00 |
| 29 | 0,9999984 | 1,0000029 | 0,0000000 | 0,0000000 | 0,9999911 | 1,0000102 | 1,91E-05 | 1,6180E+00 |
| 30 | 0,9999956 | 0,9999984 | 0,0000000 | 0,0000000 | 0,9999911 | 1,0000029 | 1,18E-05 | 1,6180E+00 |
| 31 | 0,9999984 | 1,0000001 | 0,0000000 | 0,0000000 | 0,9999956 | 1,0000029 | 7,31E-06 | 1,6181E+00 |
| 32 | 1,0000001 | 1,0000012 | 0,0000000 | 0,0000000 | 0,9999984 | 1,0000029 | 4,52E-06 | 1,6180E+00 |
| 33 | 0,9999995 | 1,0000001 | 0,0000000 | 0,0000000 | 0,9999984 | 1,0000012 | 2,79E-06 | 1,6182E+00 |
| 34 | 1,0000001 | 1,0000005 | 0,0000000 | 0,0000000 | 0,9999995 | 1,0000012 | 1,73E-06 | 1,6176E+00 |
| 35 | 0,9999999 | 1,0000001 | 0,0000000 | 0,0000000 | 0,9999995 | 1,0000005 | 1,07E-06 | 1,6190E+00 |
| 36 | 1,0000001 | 1,0000003 | 0,0000000 | 0,0000000 | 0,9999999 | 1,0000005 | 6,60E-07 | 1,6154E+00 |
| 37 | 1,0000000 | 1,0000001 | 0,0000000 | 0,0000000 | 0,9999999 | 1,0000003 | 4,06E-07 | 1,6250E+00 |
| 38 | 1,0000000 | 1,0000000 | 0,0000000 | 0,0000000 | 0,9999999 | 1,0000001 | 2,54E-07 | 1,6000E+00 |
| 39 | 1,0000000 | 1,0000001 | 0,0000000 | 0,0000000 | 1,0000000 | 1,0000001 | 1,52E-07 | 1,6667E+00 |

Зависимость количества итераций от заданного eps

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Метод дихотомии | | | | | |
|  |  |  |  |  |  |
| 1,00E-01 | 1,00E-02 | 9 | 0,92054688 | 1,01644531 | 9,59E-02 |
| 1,00E-02 | 1,00E-03 | 13 | 0,99693408 | 1,00330493 | 6,37E-03 |
| 1,00E-03 | 1,00E-04 | 16 | 0,99974222 | 1,00051361 | 7,71E-04 |
| 1,00E-04 | 1,00E-05 | 19 | 0,99992234 | 1,00001627 | 9,39E-05 |
| 1,00E-05 | 1,00E-06 | 23 | 0,99999700 | 1,00000325 | 6,25E-06 |
| 1,00E-06 | 1,00E-07 | 26 | 0,99999975 | 1,00000050 | 7,56E-07 |
| 1,00E-07 | 1,00E-08 | 29 | 0,99999992 | 1,00000002 | 9,20E-08 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод золотого сечения | | | | |
|  |  |  |  |  |
| 1,00E-01 | 13 | 0,96255949 | 1,03088313 | 6,83E-02 |
| 1,00E-02 | 17 | 0,99481754 | 1,00478582 | 9,97E-03 |
| 1,00E-03 | 22 | 0,99952392 | 1,00042276 | 8,99E-04 |
| 1,00E-04 | 27 | 0,99994830 | 1,00002935 | 8,10E-05 |
| 1,00E-05 | 32 | 0,99999560 | 1,00000290 | 7,31E-06 |
| 1,00E-06 | 37 | 0,99999986 | 1,00000052 | 6,59E-07 |
| 1,00E-07 | 41 | 0,99999996 | 1,00000005 | 9,61E-08 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод Фибоначчи | | | | |
|  |  |  |  |  |
| 1,00E-01 | 12 | 0,91777618 | 1,03442686 | 1,17E-01 |
| 1,00E-02 | 17 | 0,99402054 | 1,00454470 | 1,05E-02 |
| 1,00E-03 | 22 | 0,99956867 | 1,00051760 | 9,49E-04 |
| 1,00E-04 | 27 | 0,99995138 | 1,00003695 | 8,56E-05 |
| 1,00E-05 | 32 | 0,99999456 | 1,00000228 | 7,72E-06 |
| 1,00E-06 | 36 | 0,99999951 | 1,00000064 | 1,13E-06 |
| 1,00E-07 | 41 | 0,99999996 | 1,00000006 | 1,02E-07 |

Таблица, показывающая поиск интервала, содержащего минимум

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 11,00000000 | 100,00000000 |
| 1 | 10,99999990 | 99,99999800 |
| 2 | 10,99999970 | 99,99999400 |
| 3 | 10,99999930 | 99,99998600 |
| 4 | 10,99999850 | 99,99997000 |
| 5 | 10,99999690 | 99,99993800 |
| 6 | 10,99999370 | 99,99987400 |
| 7 | 10,99998730 | 99,99974600 |
| 8 | 10,99997450 | 99,99949000 |
| 9 | 10,99994890 | 99,99897800 |
| 10 | 10,99989770 | 99,99795401 |
| 11 | 10,99979530 | 99,99590604 |
| 12 | 10,99959050 | 99,99181017 |
| 13 | 10,99918090 | 99,98361867 |
| 14 | 10,99836170 | 99,96723668 |
| 15 | 10,99672330 | 99,93447674 |
| 16 | 10,99344650 | 99,86897295 |
| 17 | 10,98689290 | 99,73802980 |
| 18 | 10,97378570 | 99,47640119 |
| 19 | 10,94757130 | 98,95417477 |
| 20 | 10,89514250 | 97,91384510 |
| 21 | 10,79028490 | 95,84967842 |
| 22 | 10,58056970 | 91,78731578 |
| 23 | 10,16113930 | 83,92647327 |
| 24 | 9,32227850 | 69,26031943 |
| 25 | 7,64455690 | 44,15013640 |
| 26 | 4,28911370 | 10,81826893 |
| 27 | -2,42177270 | 11,70852841 |

|  |  |  |
| --- | --- | --- |
| Интервал сод. мин | -2,42177270 | 7,64455690 |

Выводы

Метод Дихотомии является простым в реализации методом. Он находит решение за меньшее число итераций чем другие два метода, так как на каждой итерации делит область поиска пополам, но при этом вычисляет функцию два раза. Так же к недостаткам данного метода можно отнести сложность правильного подбора  так в силу проблемы представления чисел в машиной памяти может случиться так что числа ивысчитанные по формулам ,  не будут отличаться, следовательно и значения функции в этих точках.

Метод золотого сечения заключается в том, что отрезок поиска делится в пропорциях, соответствующих золотому сечению. Поэтому на каждой итерации интервал поиска уменьшается всего лишь в  вместо, но при этом на каждой итерации функция вычисляется всего лишь один раз, что бывает крайне эффективно в случае если самой дорогостоящей операцией вычисления является как раз определения значения функции в точке.

Если сравнивать метод Фибоначчи с дихотомией и золотым сечением, то можно сделать следующие выводы:

Метод Фибоначчи находит решение за большое число итераций, чем метод дихотомии, но при этом он сходится за такое же количество итераций, как и метод золотого сечения, при этом вычисляя на каждой итерации функцию лишь один раз. Но у методы Фибоначчи есть минусы. Один из них связан с тему, что нужно обязательно заранее узнать количество итераций, а второй с самим вычислением чисел Фибоначчи. Так как при больших отрезках поиска или маленькой желаемой погрешности неравенства   числа Фибоначчи не помещаются в машинную память, поэтому нужно использовать специальные типы значений для их хранения. К тому же подсчет чисел Фибоначчи по формуле Бинэ тоже является не самой легкой численной операцией. ???мб что-то про расходимость в конце еще написать???

Листинг программы

public static class MyMethods

{

private const double \_eps = 1e-7;

private const double \_delta = 1e-8;

private const int \_maxIter = 10000;

public static void Dichotomy(IFunctionOfaSingleVariable function, FunctionAreaOfSingleVariable interval)

{

double ak = interval.a;

double bk = interval.b;

int i;

for (i = 1; Abs(bk - ak) > \_eps && i < \_maxIter; i++)

{

double x1 = (ak + bk - \_delta) / 2;

double x2 = (ak + bk + \_delta) / 2;

double Fx1 = function.Calculate(x1);

if (double.IsNaN(Fx1))

throw new ArithmeticException();

double Fx2 = function.Calculate(x2);

//Console.WriteLine($"{bk}");

/\*Console.WriteLine($"{Fx1} {Fx2}");

Console.WriteLine($"{ak} {bk}");\*/

if (double.IsNaN(Fx2))

throw new ArithmeticException();

if (Fx1 > Fx2)

ak = x1;

else if(Fx2 > Fx1)

bk = x2;

//else

// throw new ArithmeticException();

}

double x\_min = (ak + bk) / 2;

double Fx\_min = function.Calculate(x\_min);

if (double.IsNaN(Fx\_min))

throw new ArithmeticException();

//Console.WriteLine($"Dichotomy Method:\t x = {x\_min}, F(x) = {Fx\_min}");

Console.WriteLine($"{i} {ak} {bk}");

}

public static void GoldenRatio(IFunctionOfaSingleVariable function, FunctionAreaOfSingleVariable interval)

{

double q = (3 - Sqrt(5)) / 2;

double p = (Sqrt(5) - 3) / 2;

double ak = interval.a;

double bk = interval.b;

double x1 = ak + q \* (bk - ak);

double x2 = bk + p \* (bk - ak);

double Fx1 = function.Calculate(x1);

if (double.IsNaN(Fx1))

throw new ArithmeticException();

double Fx2 = function.Calculate(x2);

if (double.IsNaN(Fx2))

throw new ArithmeticException();

int i;

for (i = 1; Abs(bk - ak) > \_eps && i < \_maxIter; i++)

{

//Console.WriteLine(bk);

if (Fx1 < Fx2)

{

bk = x2;

x2 = x1;

x1 = ak + q \* (bk - ak);

Fx2 = Fx1;

Fx1 = function.Calculate(x1);

if (double.IsNaN(Fx1))

throw new ArithmeticException();

}

else

{

ak = x1;

x1 = x2;

x2 = bk + p \* (bk - ak);

Fx1 = Fx2;

Fx2 = function.Calculate(x2);

if (double.IsNaN(Fx2))

throw new ArithmeticException();

}

}

double x\_min = (ak + bk) / 2;

double Fx\_min = function.Calculate(x\_min);

if (double.IsNaN(Fx\_min))

throw new ArithmeticException();

// Console.WriteLine($"GoldenRatio Method:\t x = {x\_min}, F(x) = {Fx\_min}, bk-ak = {bk - ak}");

Console.WriteLine($"{i} {ak} {bk}");

}

public static void Fibonacci(IFunctionOfaSingleVariable function, FunctionAreaOfSingleVariable interval)

{

static int GetFibonacciNumber(int n)

{

if (n <= 0)

throw new ArithmeticException();

double sqrt5 = Sqrt(5);

double part1 = Pow((1 + sqrt5) / 2, n);

double part2 = Pow((1 - sqrt5) / 2, n);

int result = (int)(Round((part1 - part2) / sqrt5, 0));

return result;

}

static double CalcKoef(int Nk1, int Nk2)

{

double tmp1 = GetFibonacciNumber(Nk1);

double tmp2 = GetFibonacciNumber(Nk2);

return tmp1 / tmp2;

}

double a0 = interval.a, ak = a0;

double b0 = interval.b, bk = b0;

double tmp1 = Log10(Sqrt(5) \* (b0 - a0) / \_eps);

double tmp2 = Log10((1 + Sqrt(5)) / 2);

double tmp3 = tmp1 / tmp2;

double tmp4 = Math.Ceiling(tmp3);

double FN\_2\_max = (b0 - a0) / \_eps;

int N = Convert.ToInt32(tmp4) - 2;

double fn = Convert.ToDouble(GetFibonacciNumber(N));

double fn\_1 = Convert.ToDouble(GetFibonacciNumber(N + 1));

double fn\_2 = Convert.ToDouble(GetFibonacciNumber(N + 2));

double x1 = a0 + fn / fn\_2 \* (b0 - a0);

double x2 = a0 + fn\_1 / fn\_2 \* (b0 - a0);

double Fx1 = function.Calculate(x1);

if (double.IsNaN(Fx1))

throw new ArithmeticException();

double Fx2 = function.Calculate(x2);

if (double.IsNaN(Fx2))

throw new ArithmeticException();

int k;

for (k = 1; k <= N; k++)

{

if (Fx1 < Fx2)

{

bk = x2;

x2 = x1;

x1 = ak + CalcKoef(N - k + 1, N - k + 3) \* (bk - ak);

Fx2 = Fx1;

Fx1 = function.Calculate(x1);

if (double.IsNaN(Fx1))

throw new ArithmeticException();

}

else

{

ak = x1;

x1 = x2;

x2 = ak + CalcKoef(N - k + 2, N - k + 3) \* (bk - ak);

Fx1 = Fx2;

Fx2 = function.Calculate(x2);

if (double.IsNaN(Fx2))

throw new ArithmeticException();

}

}

double x\_min = (ak + bk) / 2;

double Fx\_min = function.Calculate(x\_min);

if (double.IsNaN(Fx\_min))

throw new ArithmeticException();

//Console.WriteLine($"Fibonacci Method:\t x = {x\_min}, F(x) = {Fx\_min}, bk-ak = {bk-ak}");

Console.WriteLine($"{k} {ak} {bk}");

}

public static void FindRegionWithMin(IFunctionOfaSingleVariable function, double x0)

{

double h;

double Fx\_prev, Fx\_cur, Fx\_next;

double x\_prev = x0, x\_cur = x0, x\_next = x0 + \_delta;

Fx\_cur = function.Calculate(x\_cur);

Fx\_prev = Fx\_cur;

Fx\_next = function.Calculate(x\_next);

if (Fx\_cur > Fx\_next)

h = \_delta;

else if (Fx\_cur < Fx\_next)

h = -\_delta;

else

throw new ArithmeticException();

for (int flag=0, i=1 ; flag != 1; i++)

{

Console.WriteLine(x\_cur);

x\_next = x\_cur + h;

Fx\_next = function.Calculate(x\_next);

if (Fx\_cur > Fx\_next)

{

Fx\_cur = Fx\_next;

x\_prev = x\_cur;

x\_cur = x\_next;

h \*= 2;

}

else

flag = 1;

}

if(h > 0)

Console.WriteLine($"{x\_prev} {x\_next}");

else

Console.WriteLine($"{x\_next} {x\_prev}");

}

}