

Решение задачи одномерной минимизации функции

Владимир Руцкий, 3057/2

1 Постановка задачи

Требуется найти с наперёд заданной точностью минимум (локальный) одномерной функции $f(x)$ на заданном отрезке $[a, b]$:

$$\min f(x), \quad x \in [a, b],$$

используя *метод золотого сечения*.

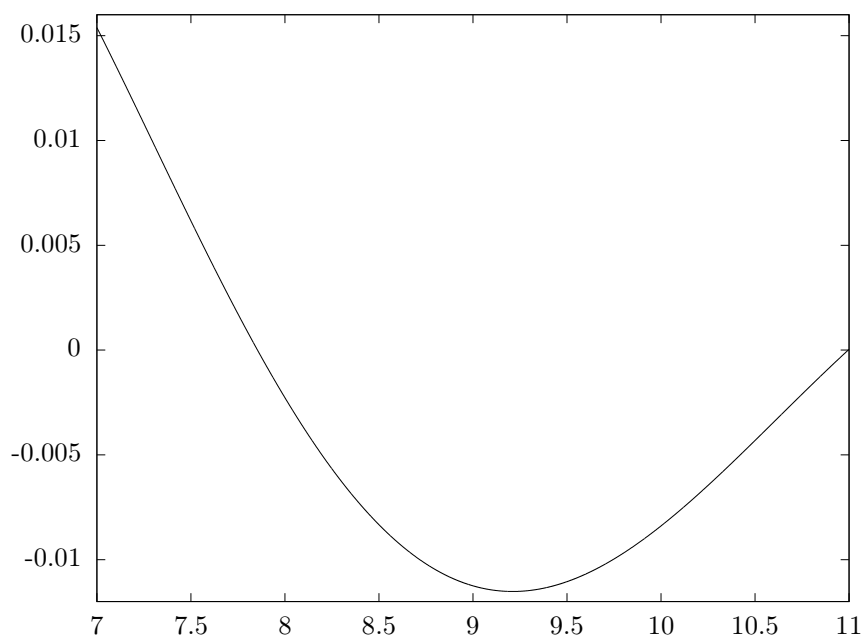
Исходная функция: $f(x) = \frac{\cos x}{x^2}$, на отрезке $[7, 11]$.

2 Исследование применимости метода

Алгоритмы поиска минимума одномерной функции $f(x)$ на отрезке $[a, b]$, использующие только значения функции в некоторых точках (исследуемый метод золотого сечения в их числе), применимы, только если заданная функция является *унимодальной*, а именно

$$\exists! x^* \in [a, b] : \begin{cases} \forall x_1, x_2 \in [a, b] : & x^* \leq x_1 \leq x_2 \Rightarrow f(x^*) \leq f(x_1) \leq f(x_2) \\ \forall x_1, x_2 \in [a, b] : & x^* \geq x_1 \geq x_2 \Rightarrow f(x^*) \geq f(x_1) \geq f(x_2) \end{cases}.$$

Рис. 1: График функции $f(x)$



Из графика видно, что исходная функция унимодальна на исследуемом промежутке.

3 Описание алгоритма

Общая идея поиска минимума унимодальной функции на отрезке заключается в том, что отрезок с минимумом можно разбить на части и гарантировано определить в какой части находится минимум, тем самым позволив сузить область для поиска минимума.

В случае метода золотого сечения, отрезок $[a, b]$, на котором имеется минимум, делится в отношении *золотой пропорции*. *Золотая пропорция* — это деление непрерывной величины на части в таком отношении, что большая часть относится к меньшей, как большая ко всей величине.

$x \in [a, b]$ делит $[a, b]$ золотым сечением, если

$$\begin{cases} x - a > b - x & \frac{x-a}{b-x} = \frac{b-a}{x-a} = \varphi \\ x - a < b - x & \frac{b-x}{x-a} = \frac{b-a}{b-x} = \varphi \end{cases},$$

т.е. для любого непустого отрезка найдутся две точки, делящие его в отношении золотой пропорции.

Величина отношения φ определена единственным образом, и равна $\frac{\sqrt{5}+1}{2}$.

Поиск минимума унимодальной функции методом золотого сечения можно описать алгоритмом 1:

Алгоритм 1 FindMinimumByGoldenSectionSearch. Поиск минимума унимодальной функции.

Вход: Исследуемая функция f ; отрезок $[a, b]$, содержащий минимум; точность ε , с которой требуется найти минимум.

Выход: Точка x^* — аргумент функции в которой достигается минимум с точностью ε .

```
{ Инициализируем первоначальное деление отрезка  $[a, b]$  на три отрезка золотым сечением. }
 $\alpha := \frac{1}{\varphi}$  { с данной величиной далее будет удобно работать }
 $i := 0$  { счетчик итераций }
{  $[a_i, b_i]$  — отрезок содержащий минимум на  $i$ -м шаге. }
 $a_i := a$ 
 $b_i := b$ 
{  $\lambda_i < \mu_i$  — точки делящие отрезок на  $i$ -м шаге золотым сечением. }
 $\lambda_i := a_i + (1 - \alpha)(b_i - a_i)$ 
 $\mu_i := a_i + \alpha(b_i - a_i)$ 
{ Подразбиваем отрезок пока его длина не станет меньше необходимой точности. }
while  $|a_i - b_i| \geq \varepsilon$  do
  if  $f(\lambda_i) \leq f(\mu_i)$  then
    { Минимум лежит на  $[a_i, \mu_i]$ , подразбиваем этот отрезок и продолжаем работу алгоритма. }
     $a_{i+1} := a_i$ 
     $b_{i+1} := \mu_i$ 
    { Из-за особенностей золотого сечения необходимо перевычислять лишь одну новую точку. }
     $\mu_{i+1} := \lambda_i$ 
     $\lambda_{i+1} := a_{i+1} + (1 - \alpha)(b_{i+1} - a_{i+1})$ 
  else
    { Минимум лежит на  $[\lambda_i, b_i]$ . }
     $a_{i+1} := \lambda_i$ 
     $b_{i+1} := b_i$ 
     $\lambda_{i+1} := \mu_i$ 
     $\mu_{i+1} := a_{i+1} + \alpha(b_{i+1} - a_{i+1})$ 
  end if
   $i := i + 1$ 
end while
{ Все точки на результирующем отрезке равны точке минимума с точностью  $\varepsilon$ , выбираем точку с меньшим значением функции из  $\lambda_i$  и  $\mu_i$ . }
if  $f(\lambda_i) \leq f(\mu_i)$  then
  return  $\lambda_i$ 
else
  return  $\mu_i$ 
end if
```

Приведённый алгоритм 1 может быть легко оптимизирован, если сохранять вычисленные значения функции f в точках α_i и μ_i , таким образом функция может быть вычислена лишь минимально необходимое число раз, для простоты эта оптимизация не приводится в алгоритме.

4 Код программы

Исходный код 1: Решение задачи поиска минимума унимодальной функции

```

1 # golden_section_search.m
2 # Golden section search algorithm.
3 # Vladimir Rutsky <altsysrq@gmail.com>
4 # 17.03.2009
5
6 precPows = [-3:-1:-8];
7
8 load data/function.mat      # func
9 load data/function_der.mat  # funcDer
10 load data/segment.mat      # range
11
12 function plotFunction( func, a, b, step )
13     assert(a <= b);
14
15     x = [a:step:b];
16     y = func(x);
17
18     plot(x, y, "-");
19 endfunction
20
21 function retval = goldenSectionSearch( func, a, b, precision )
22     assert(a <= b);
23
24     #figure(); # drawing graphic
25     #hold on; # drawing graphic
26
27     # Plotting function.
28     plotFunction(func, a, b, 1e-3); # drawing graphic
29
30     # Searching minimum with golden section search (with visualization).
31     phi = (5^0.5 + 1) / 2;
32     alpha = 1 / phi;
33
34     originalRange = [a, b];
35
36     fcalls = 0;
37
38     x1 = a + (1 - alpha) * (b - a);
39     x2 = a + (alpha) * (b - a);
40     y1 = func(x1); fcalls++;
41     y2 = func(x2); fcalls++;
42     i = 0;
43     while (abs(b - a) >= precision)
44         assert(a < x1);
45         assert(x1 < x2);
46         assert(x2 < b);
47         oldDist = b - a;
48
49         # Drawing.
50         yMax = max([func(a), func(x1), func(x2), func(b)]);
51         #yMax = max([func(x1), func(x2)]);
52         x = [a, x1, x2, b];
53         y = [yMax, yMax, yMax, yMax];
54         plot(x, y, "+k"); # drawing graphic
55
56         if (y1 <= y2)
57             b = x2;
58             x2 = x1;
59             x1 = a + (1 - alpha) * (b - a);
60             y2 = y1;
61             y1 = func(x1); fcalls++;
62         else
63             a = x1;
64             x1 = x2;
65             x2 = a + alpha * (b - a);
66             y1 = y2;
67             y2 = func(x2); fcalls++;
68         endif
69
70         newDist = b - a;

```

```

71     assert(oldDist > newDist);
72
73     i++;
74 endwhile
75
76     if (y1 <= y2)
77         res = x1;
78     else
79         res = x2;
80     endif
81
82     resStr = sprintf("Minimum_at_x=%11.9f, with_precision_of_%5.2e", res, precision);
83     title(resStr);
84
85     #drawnow(); # drawing graphic
86
87     retval = [res, i, fcalls];
88 endfunction
89
90 # Outputting to a file by the way.
91 filename = "../output/result.tex";
92 fid = fopen(filename, "w");
93
94 # Iterating through precisions.
95 prec = 10.^ precPows;
96
97 prevFuncInit = 0;
98 prevFunc = 0;
99
100 for prec = prec
101     retval = goldenSectionSearch(func, range(1), range(2), prec)
102
103     res = retval(1);
104     fres = func(res);
105
106     # Table: precision, iterations, function calls, result.
107     buf = sprintf("%2.1e_%d_%d_%11.9f_%11.9f_", prec, retval(2), retval(3), res,
108         fres);
109     fputs(fid, buf);
110
111     # Table: function value delta.
112     if (prevFuncInit)
113         buf = sprintf("%e", fres - prevFunc);
114         fputs(fid, buf);
115     endif
116
117     # Table: function derivative.
118     buf = sprintf("_%11.9e_\\n", funcDer(res));
119     fputs(fid, buf);
120
121     prevFuncInit = 1;
122     prevFunc = fres;
123 endfor
124 fclose(fid);
125
126 #input("Press enter to quit."); # drawing graphic

```

5 Результаты решения

Результаты решения приведены в таблице 1.

Таблица 1: Результаты работы метода золотого сечения

| Точность | Итерации | Вызовы функции | x | $f(x)$ | $f(x_i) - f(x_{i-1})$ | $f'(x)$ |
|----------|----------|----------------|-------------|--------------|-----------------------|------------------|
| 1.0e-03 | 18 | 20 | 9.210999446 | -0.011518238 | | 4.133279241e-07 |
| 1.0e-04 | 23 | 25 | 9.210960866 | -0.011518238 | -7.172261e-12 | -4.151203903e-08 |
| 1.0e-05 | 27 | 29 | 9.210964345 | -0.011518238 | -7.307176e-14 | -4.988289602e-10 |
| 1.0e-06 | 32 | 34 | 9.210964345 | -0.011518238 | 0.000000e+00 | -4.988289602e-10 |
| 1.0e-07 | 37 | 39 | 9.210964391 | -0.011518238 | -1.214306e-17 | 4.072474880e-11 |
| 1.0e-08 | 42 | 44 | 9.210964391 | -0.011518238 | 0.000000e+00 | 4.072474880e-11 |

6 Возможные дополнительные исследования

Вследствии особенностей золотого сечения, на каждой итерации длина исследуемого промежутка уменьшается в φ раз, что позволяет заранее оценить необходимое число итераций для нахождения минимума с определённой точностью:

$$l_i = l_0 \frac{1}{\varphi^i} = \varepsilon,$$

$$i = \log_{\varphi} \frac{l_0}{\varepsilon}.$$

На каждой итерации функция вычисляется ровно один раз.

7 Обоснование достоверности полученного результата

Согласно графику, исследуемая функция выпукла вблизи локального минимума, и полученный результат с допустимой точностью обращает производную исследуемой функции в ноль, значит найденная точка является точкой локального минимума.