# Одномерная минимизация функции

Владимир Руцкий, 3057/2

24 марта 2009 г.

# 1 Постановка задачи

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

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

используя *метод золотого сечения.*

Дана конкретная функция $\sin \frac{1}{x}$, и задан отрезок $[0.15, 0.6]$.

# 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

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

# 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, на котором имеется минимум, $[a,b]$ 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 найдуться две точки, делящие его в отношении золотой пропорции.

Величина отношения $\varphi$ 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 $\varepsilon$, с которой требуется найти минимум.
**Выход:** Точка $x^*$ — аргумент функции в которой достигается минимум с точностью $\varepsilon$.

{ 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** $|a_i - b_i| \geqslant \varepsilon$ **do**
  **if** $f(\lambda_i) \leqslant f(\mu_i)$ **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** $f(\lambda_i) \leqslant f(\mu_i)$ **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 $\alpha_i$ и $\mu_i$, таким образом функция может быть вычислена значительно меньшее число раз. Для простоты эта оптимизация не приводится в алгоритме.

# 4 Код программы

Исходный код 1: Код программы решения задачи поиска минимума унимодальной функции

```
# golden_section_search.m
# Golden section search algorithm.
# Vladimir Rutsky <altsysrq@gmail.com>
# 17.03.2009

precPows = [-3:-1:-6];
#precPows = [-3:-1:-3];
func = inline("sin(1./x)");
range = [0.15, 0.6];

function plotFunction( func, a, b, step )
```

```
  assert(a <= b);

  x = [a:step:b];
  y = func(x);

  plot(x, y, "-");
endfunction

function goldenSectionSearch( func, a, b, precision )
  assert(a <= b);

  figure();
  hold on;

  # Plotting function.
  plotFunction(func, a, b, 1e-3);

  # Searching minimum with golden section search (with visualization).
  phi = (5^0.5 + 1) / 2;

  originalRange = [a, b];

  assert(abs(phi) > 1e-10);
  x1 = b - (b - a) / phi;
  x2 = a + (b - a) / phi;
  y1 = func(x1);
  y2 = func(x2);
  while (abs(b - a) >= precision)
    assert(a < x1);
    assert(x1 < x2);
    assert(x2 < b);
    oldDist = b - a;

    # Drawing.
    yMax = max([func(a), func(x1), func(x2), func(b)]);
    #yMax = max([func(x1), func(x2)]);
    x = [a,    x1,   x2,   b   ];
    y = [yMax, yMax, yMax, yMax];
    plot(x, y, "+k");

    if (y1 <= y2)
      b  = x2;
      x2 = x1;
      x1 = b - (b - a) / phi;
      y2 = y1;
      y1 = func(x1);
    else
      a  = x1;
      x1 = x2;
      x2 = a + (b - a) / phi;
      y1 = y2;
      y2 = func(x2);
    endif

    newDist = b - a;
    assert(oldDist > newDist);
  endwhile

  if (y1 <= y2)
    res = x1;
  else
    res = x2;
  endif

  resStr = sprintf("Minimum at x=%9.7f, with precision of %5.2e", res, precision)
  title(resStr);

  drawnow();
endfunction

# Iterating through precisions.
```

```
precs = 10 .^ precPows;
for prec = precs
  goldenSectionSearch(func, range(1), range(2), prec)
endfor

input("Press enter to quit.");
```

## 5 Результаты решения

## 6 Возможные дополнительные исследования

## 7 Обоснование достоверности полученного результата