3.

OPTIMIZATION TECHNIQUES FOR ONE VARIABLE UNCONSTRAINED FUNCTIONS

3.1 Introduction

If the objective function whose extreme point must be determined depends only on one variable, in the absence of additional restrictions, we have to solve the equation:

$$\frac{df(x)}{dx} = 0; \tag{3.1}$$

As we have already seen, this is the necessary condition for the existence of an extreme point, be it a minimum or a maximum. To solve Eq. 3.1, besides the fact that the function *f(x)* must be differentiable, the expression of the equation must be as simple as possible, so that we can determine its solution. In the following, some numerical optimization methods for determination of the optimal solution will be presented, even when the above conditions are not satisfied.

3.2 Finding the boundaries of the interval containing the optimal solution

In all numerical procedures presented in this chapter, we suppose that the boundaries of the interval that contain optimal solution are known in advance. If these are not yet known, then we need a method that finds them as simply as possible. We assume that the function $y = f(x)$ is unimodal[1] and the extreme point is situated in the positive range of the Ox axis, as can be seen Fig. 3.1. For this reason, we can set the left bound x_a of the interval containing the minimum point of the function $y = f(x)$ in the origin of the coordinate axis system. Let us assume that $x_a = \tau$, where $\tau =$

[1] The unimodal function has a single extreme point in the searching interval.

0,381966[2]. Let us calculate the values of $f(x)$ in $x_a = 0$ and $x_b = \tau$. If $f(x_a) < f(x_b)$ then, given that the function is unimodal and the minimum point is in the positive range of the Ox axis, we conclude that x_b is indeed the right bound of the interval containing this point and the search process is considered finished. If $f(x_a) > f(x_b)$, then we take point x_b as x_1 and we recalculate x_b:

$$x_b = (1 + \tau) \cdot x_1 + \tau \cdot x_a; \tag{3.2}$$

We recalculate the value of the function in the new point x_b. If $f(x_1) > f(x_b)$ then we will take x_1 as x_a, point x_b as x_1 and recalculate point x_b with Eq. 3.2 (see step 3 in Fig. 3.2). The numerical procedure continues in the same way, with the recalculation of points x_a, x_1 and x_b, until $f(x_1) < f(x_b)$.

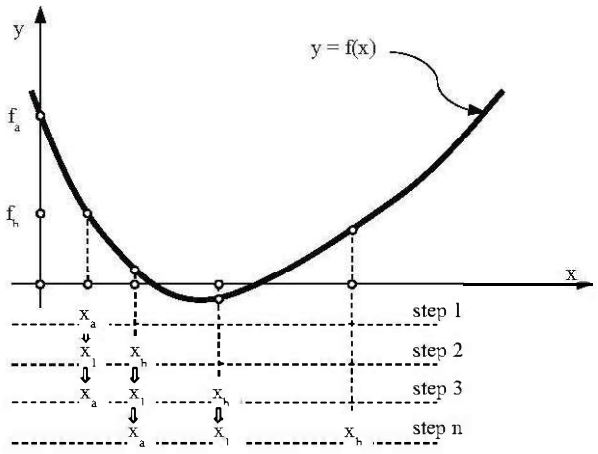

Fig. 3.1 Steps to find the interval containing the minimum point of the function $y = f(x)$.

```
1  %
2  %                    BOUNDARIES
3  % This program calculates the boundaries of the
4  % interval that includes the minimum point of
5  % an unimodal function, whose minimum point is
6  % located in the positive domain of Ox axis
7  %
8  % open a file to save results
```

[2] This reduction rate will be detailed later, in the paragraph about the golden section optimization method.

```
fp = fopen('results.txt','w');
fprintf(fp,'                    BOUNDARIES\n')'
fprintf(fp,'This program calculates the boundaries of
the\n');
fprintf(fp,'interval that includes the minimum point of
for\n');
fprintf(fp,'a unimodal function, whose minimum point
is\n');
fprintf(fp,'located in the positive domain of Ox
axis\n\n');
% initial left bound
xa = 0.0;
tau = 0.381966;
xb = tau;
fa = f(xa);
fb = f(xb);
fprintf(fp,'xa=%f   xb=%f   fa=%f   fb=%f\n',...
    xa,  xb,  fa,  fb);
if fa < fb
    left_bound = xa;
    right_bound = xb;
end
while fa > fb
    x1 = xb;
    f1 = fb;
    xb = (1+tau)*x1+tau*xa;
    fb = f(xb);
    fprintf(fp,'xa=%f   xb=%f   fa=%f   fb=%f\n',...
        xa,  xb,  fa,  fb);
    if f1 >= fb
        xa = x1;
        fa = f1;
    elseif f1 < fb
        left_bound = xa;
        right_bound = xb;
    end
end
% print to file the final results
fprintf(fp,'\n  Final result:\n');
fprintf(fp,'leftBound=%f    rightBound=%f\n',...
    left_bound,right_bound);
% close the file
fclose(fp);
%
```

Code 3.1 *boundaries.m*.

The program *boundaries.m* in Code 3.1 calculates the left and right bounds of the interval that contains the minimum point of a function. It makes a call to a function *f.m* in Code 3.2.

```
function [val] = f(x)
% the evaluation of function f in a point x
% f(x) represents the objective function
% f(x) = x*x - 4*x + 3;
val = x*x - 4*x + 3;
end
```

Code 3.2 *f.m*.

The objective function in Code 3.2 is a parabola of equation:

$$f(x) = x^2 - 4x + 3; \tag{3.3}$$

The function in Eq. 3.3 has a minimum point whose coordinates are [2,-1]. In this case, the program will start the search process taking as left bound the origin of the coordinate system and, step by step, will include within its boundaries this minimum point.

Table 3.1 History of iterations to find boundaries.

step	left bound x_a	right bound x_b	$f_a = f(x_a)$	$f_b = f(x_b)$
1	0.000000	0.381966	3.000000	1.618034
2	0.000000	0.527864	3.000000	1.167184
3	0.381966	0.875388	1.618034	0.264752
4	0.527864	1.411383	1.167184	-0.653530
5	0.875388	2.284852	0.264752	-0.918860
6	1.411383	3.696687	-0.653530	1.878748

After the execution of the program, the value of x_a = 1.411383 was determined as the left bound for the minimum point of the function in Eq. 3.3, and for the right bound x_b = 3.696687.

3.3 The grid method

Let us consider an unimodal function $y = f(x)$, and let us also admit as known the bounds a and b of the interval containing the minimum point of this function. We choose a number of n equal subdivisions of the interval $[a, b]$. The size δ of a subdivision (see Fig. 3.2) is: