Bisection Algorithm

This algorithm is implemented in C++ in the class *RBisectionLineSearchAlgorithm*. A simple but somewhat computationally inefficient method to solve the 1D optimization problem

$$x^* = \arg\min\{F(x)\} \tag{1}$$

and the 1D root-finding problem

$$f(x) = \frac{dF(x)}{dx} = 0 \tag{2}$$

is to repeatedly cut in half intervals along the x-axis where the solution is known to exist. The algorithm starts with a user-given guess for the interval in which the solution is assumed to exist.

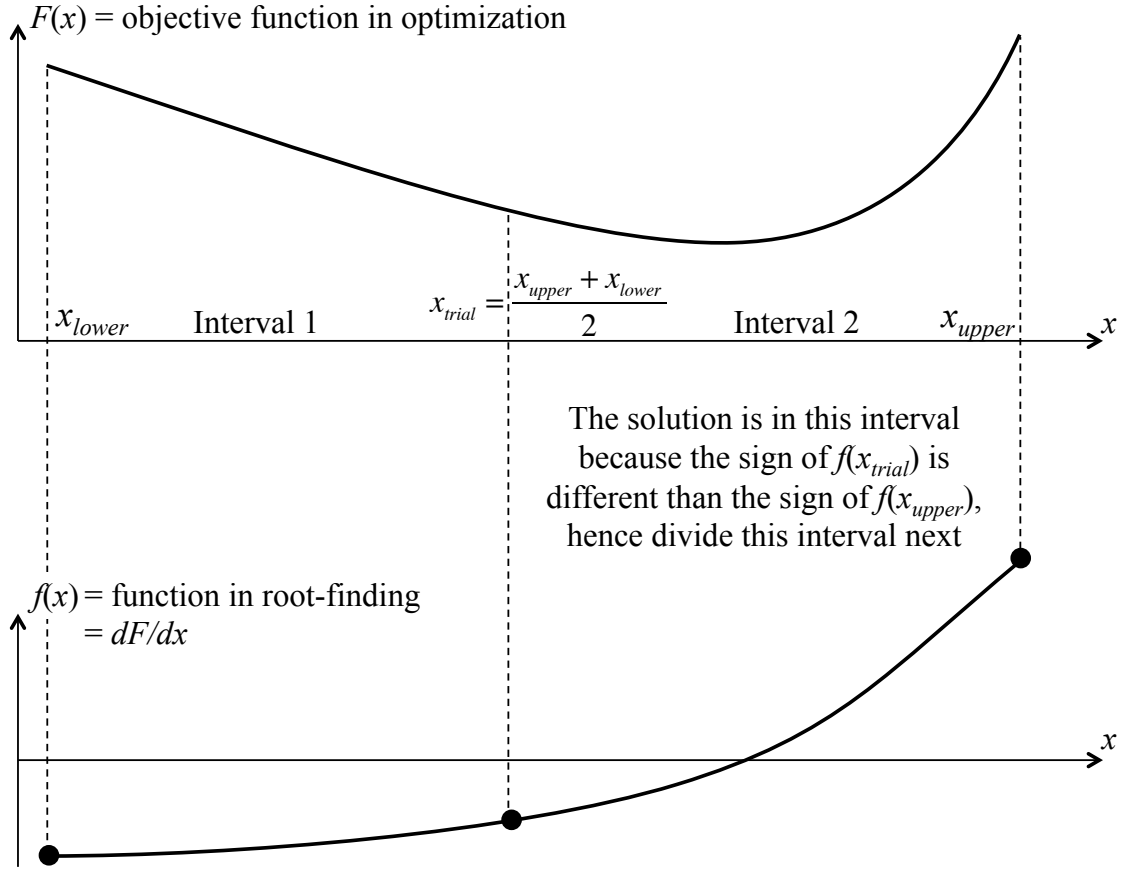

Figure 1: Dividing of intervals in the bisection algorithm.

Starting with the initial bounds x_{lower} and x_{upper}, and the calculation of the corresponding function values $f(x_{lower})$ and $f(x_{upper})$ the interval midpoint

$$x_{trial} = \frac{x_{upper} + x_{lower}}{2} \tag{3}$$

and the corresponding value $f(x_{trial})$ is computed. If $f(x_{trial})$ and $f(x_{lower})$ has a different signs then the solution is known to be in the left-most interval, and vice versa.