Computational Physics SuSe 2023
9. Exercise Sheet

Handing out: 16.06.2023 Prof. H. Jelger Risselada
Submission: 23.06.2023 11 pm

Exercise 0: Comprehension questions 0 Points

1) Which methods for minimizing one-dimensional functions do you know? Discuss their advantages and disadvantages.

Exercise 1: Onedimensional optimization 9 Points

As with many basic procedures, there are several possible procedures for minimization. In this task, two different one-dimensional minimization algorithms are compared.
Implement the algorithm presented in the lecture

a) interval bisection method.

b) Newton's method

each with an accuracy bound of $x_a = 10^{-9}$ and test both methods on the function

$$f(x) = x^2 - 2 \; . \tag{1}$$

Use as initial values for the interval bisection procedure

$$x_0 = -0.5 \tag{2}$$
$$y_0 = -0.1 \tag{3}$$
$$z_0 = 2 \tag{4}$$

and for Newton's method

$$x_0 = 1 \; . \tag{5}$$

For an iterative formulation of the interval bisection method, have the iteration steps and for recursive formulation the function calls and compare them with those of the Newton method. For this, plot also the x_i, y_i, z_i of the interval bisection method, as well as the x_i of the Newton method against the step number.

Exercise 2: particle swarm optimization — 11 Points

gradient methods for extremum search have two key drawbacks: for non-differentiable functions, the gradients needed for the method can at best be approximated by multiple function evaluations, which is computationally intensive and imprecise, and for functions with multiple extrema, the methods can get stuck in local extrema. For such functions, it is convenient to use evolutionary algorithms. In this task, you are asked to minimize a function with many local minima using particle swarm optimization to familiarize yourself with the procedure.
The particle swarm consists of N positions $\vec{r}$, which in each iteration step are given by

$$\vec{r}_{n+1} = \vec{r}_n + \vec{v}_n \tag{6}$$

$$\vec{v}_{n+1} = \omega\vec{v}_n + c_1 r_1(\vec{r}_n^{\text{pers. best}} - \vec{r}_n) + c_2 r_2(\vec{r}^{\text{global best}} - \vec{r}_n) \tag{7}$$

be updated, where $\omega \in [0,1]$ and $c_1, c_2 \in \mathbb{R}_+$ are fixed weighting factors and $r_1, r_2 \in [0,1]$ are random scaling factors redrawn at each step and for each particle. Here we choose $\omega = 0.8$ and $c_1 = c_2 = 0.1$ and draw r_1 and r_2 from a uniform distribution on the interval $[0,1]$. The position $\vec{r}_n^{\text{pers. best}}$ stores the best position found so far for the nth particle and $\vec{r}^{\text{global best}}$ stores the best position found so far for all particles.
With the help of the particle swarm optimization you should now find the global minimum of the function

$$f(x,y) = (x-1.9)^2 + (y-2.1)^2 + 2\cos(4x+2.8) + 3\sin(2y+0.6) \tag{8}$$

determinedetermine

a) Initialize 10 particles randomly distributed on the interval $\vec{r}_i \in [-5,5]^2$ with random starting velocities $\vec{v}_i \in [-1,1]^2$. Run the particle swarm optimization for at least 100 iterations. *Note: If your algorithm works correctly, you should end up finding a global minimum with* $f(\vec{r}^{\,\text{global best}}) < -4$.

b) Plot in a single figure both the course of the best function value found $f(\vec{r}^{\,\text{global best}})$ and the course of the individual best values $f(\vec{r}_n^{\,\text{pers. best}})$ over the number of iterations.

c) Represent the course of the particle swarm search vividly using several snapshots and/or an animation. For this purpose, you should display the positions $\vec{r}_n$ of the particles (as markers) and their velocities $\vec{v}_n$ (as arrows) as well as the best positions $\vec{r}_n^{\text{pers. best}}$ and $\vec{r}^{\text{global best}}$ found so far (for example as different looking markers). It is useful to display the function $f(x,y)$ as a heatmap and/or contour plot in the background to be able to interpret the motion of the particle swarm.

Note: Feel free to apply the algorithm to other values of (ω, c_1, c_2) *and other initial values and compare with the parameters given here.*