1 Unconstrained Static Optimization

This exercise deals with two topics: First, the accuracy of numerical differentiation algorithms is to be investigated using the central difference quotient and complex function evaluation as examples. Furthermore, the minimum of a cost function $f(\mathbf{x})$ with respect to the optimization variable $\mathbf{x}$ is to be found in an unconstrained domain. For this purpose, the *Newton* method is to be implemented in MATLAB and subsequently compared, using two test functions, with the algorithms provided by the *Optimization Toolbox*.

This exercise is divided into two parts. The first part consists of preparatory tasks to be completed *before the exercise session*. In the second part, a numerical differentiation task and an unconstrained optimization problem are solved. These tasks are to be completed *during the exercise session*.

Complete the following tasks in preparation for the exercise session:

1. Implement the *Booth* function

$$f_{\text{Booth}}(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

   and the *Styblinski–Tang* function

$$f_{\text{Styblinski-Tang}}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i^4 - 16x_i^2 + 5x_i\right)$$

   for $n = 2$ in MATLAB as separate functions `[f,df,ddf]=calcf_booth(x)` and `[f,df,ddf]=calcf_styblinski_tang(x)`, respectively, which return the corresponding function value `f`, the analytically calculated gradient `df`, and the analytically calculated Hessian matrix `ddf`. Plot these two functions with the MATLAB command `mesh`, as shown in Figure 1.1. Furthermore, use the `plot3` command to draw a line in the graph.

   In addition, visualize the geometric shape of the functions by plotting their contour lines with the MATLAB command `contour`. The MATLAB command `meshc` combines the two commands `mesh` and `contour`.

2. Familiarize yourself with the *Optimization Toolbox*, particularly the command `fminunc`. Consult the MATLAB documentation for this command.

3. Study the theory of derivative calculation using difference quotients and complex differentiation, as well as the *Newton* method and line-search methods.

The following tasks are to be completed during the exercise session:

©, Automation and Control Institute, TU Wien

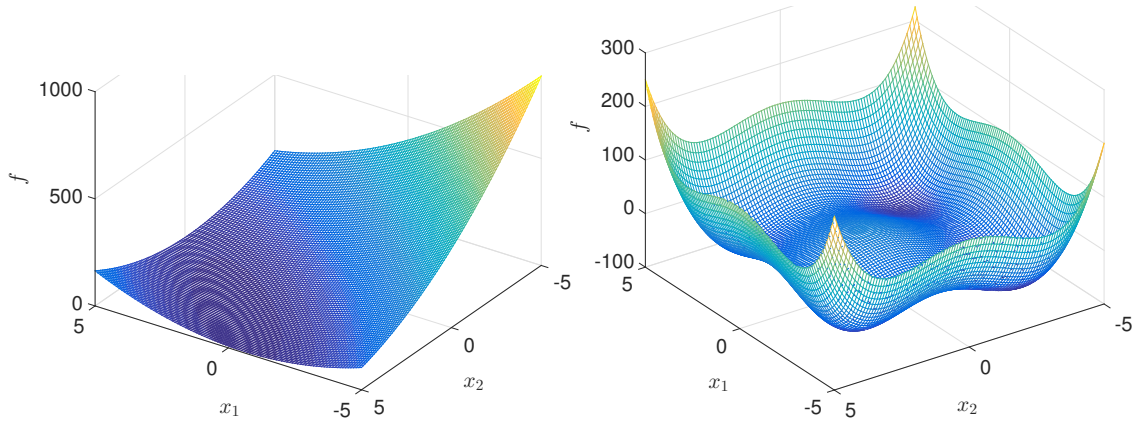


Figure 1.1: Booth function and Styblinski–Tang function.

1. Implement the two subroutines `[f,df]=diff_central(fun,x,h)` and `[f,df]=diff_complex(fun,x,h)`, which determine the gradient of a function `fun` using the central difference quotient and complex function evaluation, respectively. Note that the argument `x` is a vector. The subroutines are to determine the function value `f` and the gradient `df` with step size `h` at the point `x`, where the function `fun` is passed as a *function handle* (the `@` symbol in Matlab).

   The accuracy of the numerical algorithms is assessed using the directional derivative of the function $f_{\text{Styblinski-Tang}}(\mathbf{x})$ along a curve $\mathbf{x}(t) \in \mathbb{R}^2$ according to

   $$d(t) = \nabla f_{\text{Styblinski-Tang}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}(t)} \cdot \mathbf{x}'(t) \tag{1.1}$$

   with $\mathbf{x}'(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$. Choose $\mathbf{x}(t)$ as the straight line $\mathbf{x}(t) = \mathbf{a} + \mathbf{b}t$ with $\mathbf{a} = [-5, -5]^{\mathrm{T}}$ and $\mathbf{b} = \frac{1}{\sqrt{2}}[1, 1]^{\mathrm{T}}$.

   Calculate the gradient $\nabla f_{\text{Styblinski-Tang}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}(t)}$ for the directional derivative (1.1) analytically and with the subroutines `diff_central` and `diff_complex`. The direction of the straight line $\mathbf{x}'(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$ in (1.1) is to be calculated analytically.

   Furthermore, calculate the absolute errors between the analytical and numerically calculated directional derivatives for different step sizes $h \in [10^{-9}, 10^{-1}]$ at 10 points in the interval $t \in [0, 10\sqrt{2}]$, and average them for each step size.

   - How does the error behave as the step size h decreases?
   - In which ranges does the truncation error or the round-off error dominate? Where does cancellation occur?

2. Write a function `[x,fval]=newton(fun,x0)` that uses the *Newton* method to calculate the minimum of a cost function starting from the initial value $\mathbf{x}_0$. The cost function is to be passed as a *function handle* and the initial value as a vector `x0`. Choose a constant step size $\alpha_k = 1$ (undamped Newton method) and plot the iteration steps as a polygonal chain in a three-dimensional plot.

©, Automation and Control Institute, TU Wien

> **Remark:** Use the analytically calculated gradient and the analytically calculated Hessian matrix here.

3. Test your method using the *Booth* function and the *Styblinski–Tang* function. Start the iterations at the points $\mathbf{x}_0 = [5, 0]^{\mathrm{T}}$ and $\mathbf{x}_0 = [5, 5]^{\mathrm{T}}$, respectively.
   - Does the method converge to a global minimum?
   - Why might different points be reached depending on the initial value?
   - How can the convergence behavior be improved?

   > **Remark:** The global minimum is located at $\mathbf{x} = [1, 3]^{\mathrm{T}}$ for the Booth function and at $\mathbf{x} = [-2.903534, -2.903534]^{\mathrm{T}}$ for the Styblinski–Tang function.

4. Compare your results with the function `fminunc` provided by MATLAB using the *trust-region method*. Make sure that the options are passed correctly so that this method is actually used.

   > **Remark:** For the MATLAB command `fminunc`, set the option `Display` to `'iter'` (using `optimoptions`). This displays the progress of the optimization after each iteration of the algorithm.