Assignment 1

Numerical Optimization / Optimization for CS WS2021

Christian Kopf, christian.kopf@icg.tugraz.at
Lukas Erlbacher, lukas.erlbacher@student.tugraz.at
Thomas Wedenig, thomas.wedenig@student.tugraz.at

October 19, 2021

Deadline: Nov 16th, 2021 at 23:59

Submission: Upload your report and your implementation to the TeachCenter. Please use the provided framework-file for your implementation. Make sure that the total size of your submission does not exceed 50MB. Include **all** of the following files in your submission:

- `report.pdf`: This file includes your notes for the individual tasks. Keep in mind that we must be able to follow your calculation process. Thus, it is not sufficient to only present the final results. You are allowed to submit hand written notes, however a compiled LaTeX document is preferred. In the first case, please ensure that your notes are well readable.
- `main.py`: This file includes your python code to solve the different tasks. Please only change the marked code sections. Also please follow the instructions defined in `main.py`.
- `figures.pdf`: This file is generated by running `main.py`. It includes a plot of all mandatory figures on separate pdf pages. Hence, you do not have to embed the plots in your report.

1 Characterization of Functions (8 P.)

Given $\mathbf{x} = (x_1\ x_2)^\mathrm{T}$ and

a) $f(\mathbf{x}) = (\mathbf{a}^\mathrm{T}\mathbf{x} - d)^2$ where $\mathbf{a} = (-1\ 3)^\mathrm{T}$, $d = 2.5$

b) $f(\mathbf{x}) = (x_1 - 2)^2 + x_1 x_2^2 - 2$

c) $f(\mathbf{x}) = x_1^2 + x_1\|\mathbf{x}\|^2 + \|\mathbf{x}\|^2$

d) $f(\mathbf{x}) = \alpha x_1^2 - 2x_1 + \beta x_2^2$

and let $\|\cdot\|$ denote the ℓ_2-norm. For each of the given functions do:

In Python:

1. Plot the level sets of the above functions using a contour plot[1]. For d) use α, β of your choice.
2. Add markers to the stationary points (computed with Pen & Paper).

With Pen & Paper:

3. Compute the gradient and the Hessian.
4. Determine the set of stationary points.
5. For a) to c) characterize every stationary point whether it is a saddle point, (strict) local/global minimum or maximum.
6. For d) denote the intervals for α and β for which minima/maxima and saddle points are attained.

[1] `https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.contour.html`

2 Matrix Calculus (7.5 P.)

Given $\mathbf{x} \in \mathbb{R}^n$

a) $f(\mathbf{x}) = \frac{1}{4}\|\mathbf{x} - \mathbf{b}\|^4$ for $\mathbf{b} \in \mathbb{R}^n$

b) $f(\mathbf{x}) = \sum_{i=1}^{n} g\left((\mathbf{A}\mathbf{x})_i\right)$ with $g(z) = \frac{1}{2}z^2 + z$ for $z \in \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{n \times n}$, subscript i denoting the i-th element

c) $f(\mathbf{x}) = (\mathbf{x} \oslash \mathbf{b})^{\mathrm{T}} \boldsymbol{D} (\mathbf{x} \oslash \mathbf{b})$ for $\mathbf{b} \in \mathbb{R}^n, \mathbf{D} \in \mathbb{R}^{n \times n}$

The operator $\oslash$ denotes the Hadamard-Division which is the element-wise division of two vectors or matrices e.g.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \oslash \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} \frac{a_{11}}{b_{11}} & \frac{a_{12}}{b_{12}} \\ \frac{a_{21}}{b_{21}} & \frac{a_{22}}{b_{22}} \end{bmatrix} .$$

For each of the given functions:

With Pen & Paper:

1. Compute the gradient of $f(\mathbf{x})$. Compute the derivative for each element using the summation formulation as shown in the exercise lecture. Convert your result back to multivariate notation.
2. Compute the Hessian of $f(\mathbf{x})$. Proceed similarly to the gradient computation step. Show all your steps in the report.

3 Numerical Gradient Verification (4.5 P.)

To see whether the computed gradient is correct one can easily verify this by computing a numerical approximation of the gradient. For the case that $\mathbf{x} \in \mathbb{R}^2$ this can be achieved using central differences

$$\nabla f(\mathbf{x}) \approx \frac{1}{2\epsilon} \begin{bmatrix} f((x_1 + \epsilon,\ x_2)^{\mathrm{T}}) - f((x_1 - \epsilon,\ x_2)^{\mathrm{T}}) \\ f((x_1,\ x_2 + \epsilon)^{\mathrm{T}}) - f((x_1,\ x_2 - \epsilon)^{\mathrm{T}}) \end{bmatrix}$$

In Python:

1. Write a function to check the gradients for all the functions in Task 1 numerically as shown above. To do so, set $\epsilon = 0.0001$ and choose a random point for $\mathbf{x}$. For this point compare the result of your analytically computed gradient with the numerically approximation. Plot and compare your results in bar plot. Use the provided functions and variables to compute the gradient, function value and gradient approximation.
2. In a general setting for $\mathbf{x} \in \mathbb{R}^n$ choose $n = 5$ and again perform the verification but for your results from Task 2. This time use scipy's `approx_fprime()` function. Create a bar plot for each function $f(\mathbf{x})$ comparing your analytical result vs. the numerical approximation. Use the provided functions and variables to compute the gradient, function value and gradient approximation.

4 Scheduling Optimization Problem (5 P.)

As a well experienced software engineer assume you have a mobile computing system with two processing units (PUs) at your disposal. After some measurement has been taken, the task is to run computations comprised of eight processes at once. Both processing units (CPU, GPU) have different instruction sets which means that parts of some processes can only be executed on one PU but not on the other. In addition, depending on the process and which PU to execute it on, the energy consumption varies. Each process P_i is comprised of K_i instructions. The average estimate for the consumed energy for each PU and process is denoted in μWh per instruction and is stated in the table below.

	Processes							
	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
μWh per Instruction (CPU)	0.11	0.13	0.09	0.12	0.15	0.14	0.11	0.12
μWh per Instruction (GPU)	0.10	0.13	0.08	0.13	0.14	0.14	0.09	0.13
Number of Instructions K_i	1200	1500	1400	400	1000	800	760	1300

Your task is now to find the optimal scheduling plan which states how many instructions per process are scheduled on each PU. Because the battery capacity of this mobile computing system is limited, your goal is to keep the energy consumption as low as possible. The complexity of the processes P_0 to P_2 demands that at least 40% of each process needs to be executed on the CPU. Limit both PUs to a maximum of 4500 instructions each.

Tasks:

(a) Formulate the objective function of a linear program to compute a scheduling plan such that energy consumption is minimized.

(b) Formulate the corresponding constraints and state them in your report.

(c) Use `scipy`'s linear program solver[2] to solve the linear program.

(d) Report the scheduling plan as a matrix $M \in \mathbb{R}^{8\times 2}$ where the first column denotes how many instructions per process i are scheduled for execution on the CPU. Likewise the second column states the GPU scheduling.

(e) Check whether your solution fulfills all constraints.

(f) Compute and state the total energy consumption.

(g) Assume that the processes P_0, P_1, P_2 and P_7 need to be executed entirely on the CPU. Explain the results you get.

[2] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html`