Peter Fischer, k12102665

Project 1: Individual Submission

auxiliary percentage:

I solved the Problems i, ii, iv → 25/30 → 0.83 %

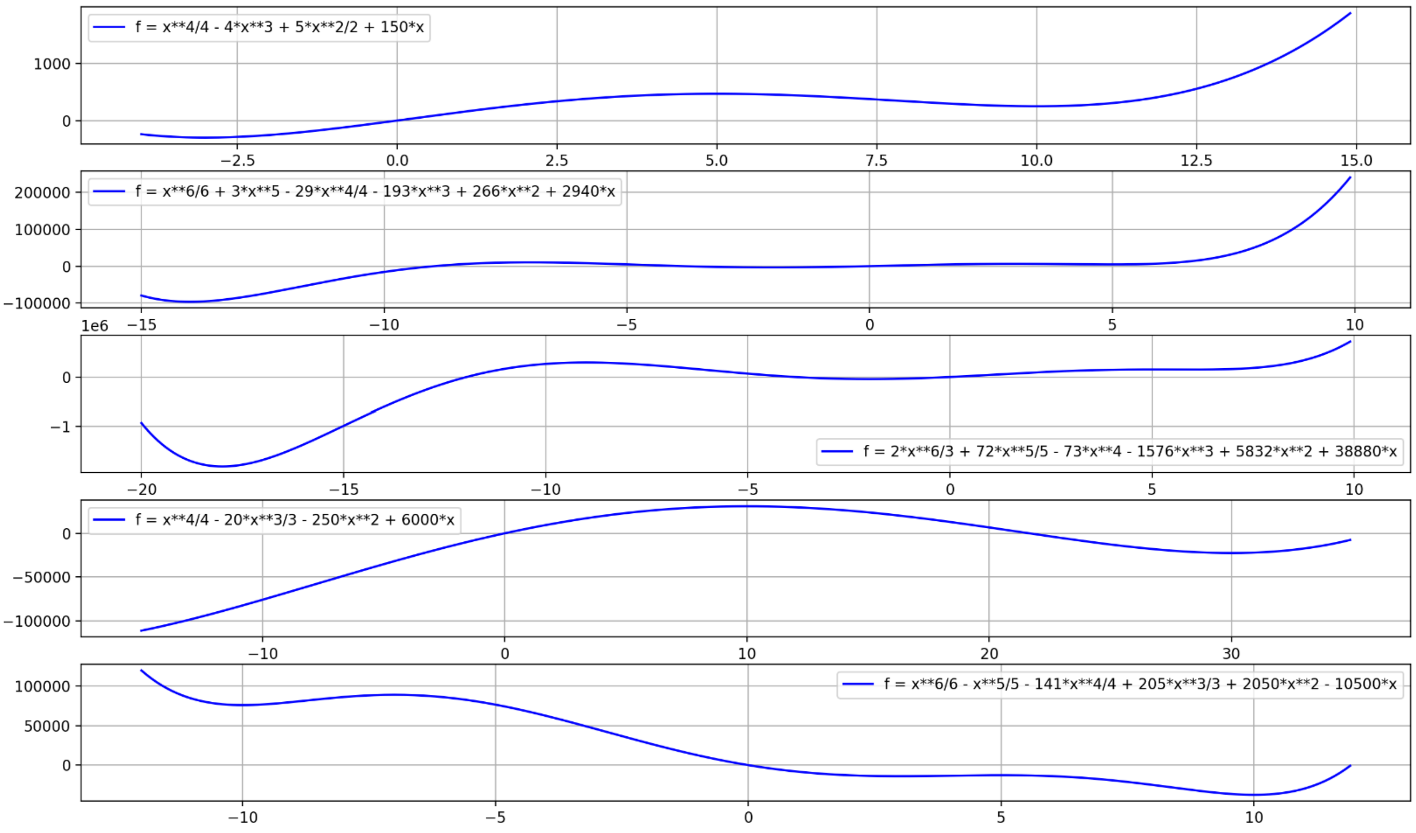
Generall:

I have decided to write a class called LineSearch, since the functionality of the two methods is very similar and can therefore be easily combined with slight modifications. During initialization, the method (SD or NM) to be applied later is specified. Later, methods are added that are responsible for the actual functionality: pk (for calculating the pk value), d2function (the actual line search function), alpha_strongWolfe (calculation of alpha), and methods for general mathematical operations (nabla, grad, hesse). The remaining methods: testi - testiv (perform the tasks), find_mindif (for calculating the next analytical zero point for numerical solution), and zeros (for calculating the analytical solution) are the programmed statements for testing the methods.

I am afraid that the alpha_strongWolfe function may have an error somewhere. I have rewritten it several times and it works sufficiently for tasks i and ii, although it sometimes takes several hours to run. Unfortunately, after much research, I have not found a better method for calculating alpha, which now means that I can no longer consider task iii as completed. The step size becomes very small, yet it still satisfies the Wolfe condition (no break in the loop is triggered). If you notice an error during your review, I would kindly ask you to let me know through Moodle comments or similar, as I have invested a lot of time and finding a solution would be very interesting.

i. I constructed the functions: $\int(x-5)(x+3)(x-10)dx$, $\int(x+2)(x-3)(x+14)(x+7)(x-5)dx$, $\int(2x-10)(x+2)(x-6)(x+18)(2x+18)dx$, $\int(x-10)(x+20)(x-30)dx$, $\int(x+10)(x-3)(x-10)(x+7)(x-5)dx$

The following picture shows these functions in the same order:



Results:

Steepest descent:

	Local minimizer (x*)	$\tilde{x}$	$\|\nabla f(\tilde{x})\|$	$\|\tilde{x} - x^*\|$	Iterations
f_1	-3	-2.99999999331382	6.95362796676591E-7	6.68E-9	46
f_2	-2	-2.00000000341598	0.0000717355578672141	3.416E-9	2001 (maximum)
f_3	-18	-18.0000001253166	0.0398446683248039	1.25E-7	2001 (maximum)
f_4	-20	-20.0000000298606	0.0000447908960268251	2.99E-8	2001 (maximum)
f_5	10	10.0000000089268	0.000106228486401960	8.93E-9	2001 (maximum)

Newton Method:

	Local minimizer (x*)	$\tilde{x}$	$\|\nabla f(\tilde{x})\|$	$\|\tilde{x} - x^*\|$	Iterations
f_1	-3	-3.00000000137805	1.43316866353851E-7	1.38E-9	10
f_2	-7	-7	3.84500534384366E-15	~0	8
f_3	-2	-2	1.13275170068110E-16	~0	6
f_4	10	9.99999999951881	2.88712496271832E-7	-4.81E-10	5
f_5	3	2.99999999999938	1.12068833681564E-9	-6.2E-13	6

ii. I have decided to increase the complexity of the functions from f1 - f5 from "low" to "high" in order to be able to recognise a possible connection between complexity and runtime or accuracy (if any).
I specified the following functions: $f_1 = x, f_2 = x^2, f_3 = (x-2) * (x+3) * (x-4), f_4 = 2^x, f_5 = e^x$
$n(f_1) = 2, n(f_2) = 4, n(f_3) = 5, n(f_4) = 10, n(f_5) = 20$ …. These are the degrees of the polynomials.
In this exercise I started with the Newton Method, because the computations were much faster than in the steepest descent (backtracking needed much time). The results of the newton method are:
$f_{1,NM}(x) = 0\, x^0 + 1.0\, x^1 + 0\, x^2$

$f_{2,NM}(x) = 4.44089209850063 * 10^{-16}\, x + 0.999999999999996\, x^2 + 1.11022302462516 * 10^{-16}\, x^4$

$f_{3,NM}(x) = 23.9999999999999 + -10.0000000000019\, x^1 + -2.99999999999998\, x^2 + 1.00000000000008\, x^3 + -2.94415704357337 * 10^{-16} x^4 + -7.16400403845391 * 10^{-16} x^5$

$f_{4,NM} = 0.973794292186562 + 0.779086998594097\, x^1 + 0.259145113288975\, x^2 + 0.0375827381119549\, x^3 + 0.00741107953559312\, x^4 + 0.00235296290623539\, x^5 + 0.000246234444571135\, x^6 + -7.53266064576015 * 10^{-06} x^7 + -3.58624557656449 * 10^{-07} x^8 + 3.06386146250456 * 10^{-07} x^9 + 2.00659978582769 * 10^{-08} x^{10}$

$f_{5,NM} = 1.0000112781233\, x^0 + 0.999921876364667\, x^1 + 0.499972691415259\, x^2 + 0.166726043331376\, x^3 + 0.0416777456102093\, x^4 + 0.00832011050521675\, x^5 + 0.0013871463367232\, x^6 + 0.000199755541111343\, x^7 + 2.49410758141261 * 10^{-05} x^8 + 2.68124505596779 * 10^{-06} x^9 + 2.69114562235893 * 10^{-07} x^{10} + 2.7515963592816 * 10^{-08} x^{11} + 2.27224735048317 * 10^{-09} x^{12} + 1.10080107022728 * 10^{-10} x^{13} + 8.1276119315975 * 10^{-12} x^{14} + 1.40685525456938 * 10^{-12} x^{15} + 8.59289219636535 * 10^{-14} x^{16} + -2.02360594459927 * 10^{-15} x^{17} + -1.0428620303939 * 10^{-16} x^{18} + 2.68803744402403 * 10^{-17} x^{19} + 1.33047317410625 * 10^{-18} x^{20}$

The results of the Steepest descent method are:

$$f_{1,SD}(x) = -0.000403123199522074 \ + 0.999999592010767\, x^1 + 6.57173428479047 * 10^{-6} x^2$$

$$f_{2,SD}(x) = 0.0102397641739058 + 0.00075409202296346\, x^1 + 0.221637572026657\, x^2 - 0.000782393321339397\, x^3 + 0.0098961632574229\, x^4$$

$$f_{3,SD}(x) = -6.92218042548813 * 10^{-5} x^0 + 0.00125153873002758\, x^1 - 0.00300164686054546\, x^2 + 0.0393297348305784\, x^3 - 0.0346360416633931\, x^4 + 0.0101282967691139\, x^5$$

$$f_{4,SD}(x) = (1.89511863540625\, e - 14) + (1.13331456416124\, e - 13)x^1 + (9.01982106200585\, e - 13)x^2 + (5.75106808302156\, e - 12)x^3 + (4.6263518030367\, e - 11)x^4 + (3.02467843061246\, e - 10)x^5 + (2.26958583997176\, e - 09)x^6 + (1.4993006936304\, e - 08)x^7 + (8.78572542325367\, e - 08)x^8 + (5.72932744912423\, e - 07)x^9 + (5.85958786216716\, e - 08)x^{10}$$

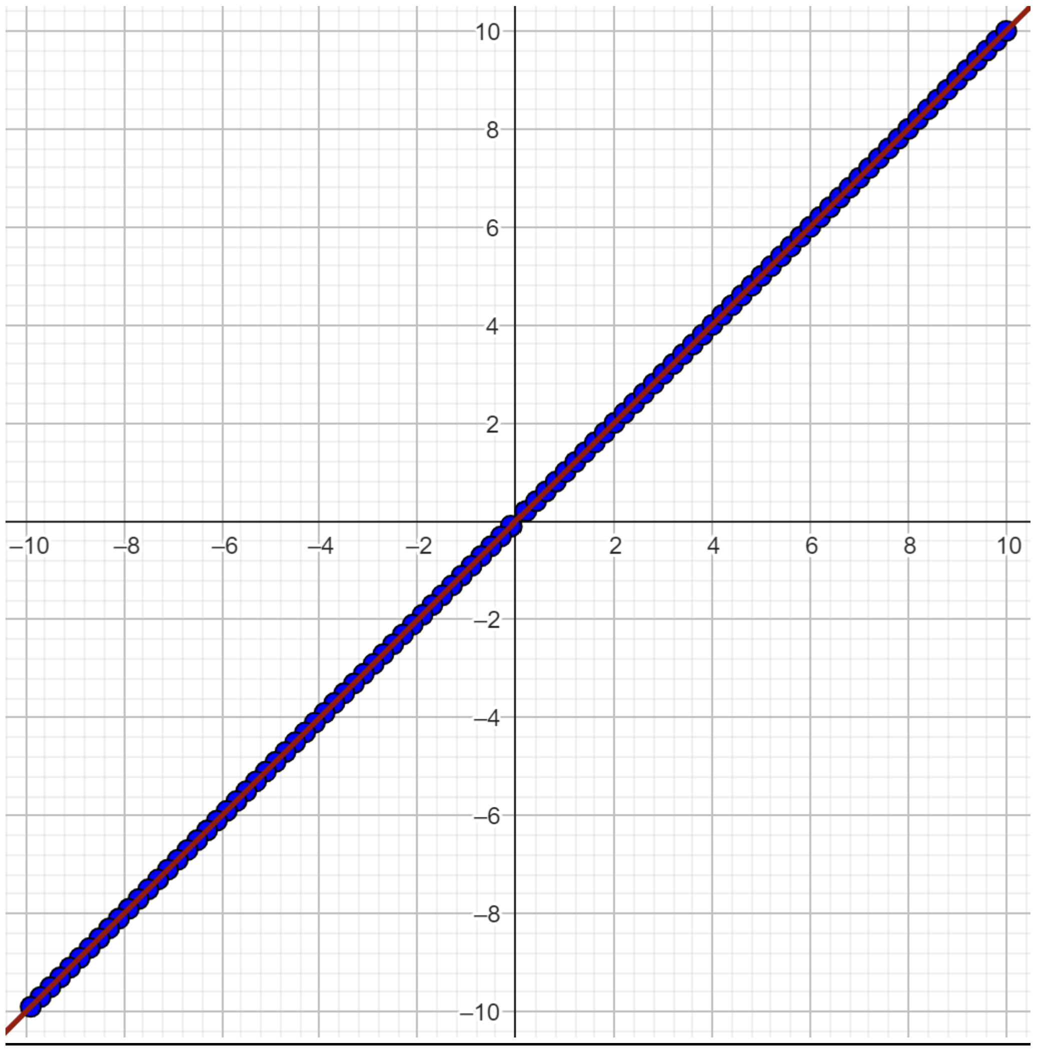
$$f_{5,SD}(x) = 6.32773144730604 * 10^{-32} + 4.87965809879419 * 10^{-31} x^1 + 4.23067301820466 * 10^{-30} x^2 + 3.31522514074241 * 10^{-29} x^3 + 2.94921463445721 * 10^{-28} x^4 + 2.3167239950346 * 10^{-27} x^5 + 2.09585424161297 * 10^{-26} x^6 + 1.63563068195347 * 10^{-25} x^7 + 1.49663368862708 * 10^{-24} x^8 + 1.14969137503101 * 10^{-23} x^9 + 1.05967005900058 * 10^{-22} x^{10} + 7.91248224928661 * 10^{-22} x^{11} + 7.31719544863631 * 10^{-21} x^{12} + 5.19886942515279 * 10^{-20} x^{13} + 4.79887263830713 * 10^{-19} x^{14} + 3.10091344106155 * 10^{-18} x^{15} + 2.82849573926505 * 10^{-17} x^{16} + 1.45047696186971 * 10^{-16} x^{17} + 1.26250339035003 * 10^{-15} x^{18} + 1.32898434350577 * 10^{-15} x^{19} + 1.21654214984592 * 10^{-16} x^{20}$$

Graphs:

Legend:

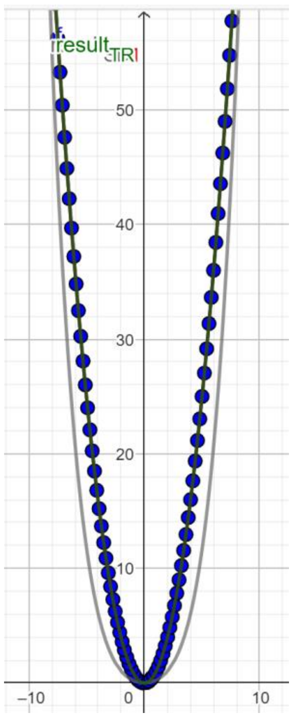
🟦	f_n	Input Function
🟦	P_n	Data Points
🟩	$f_{n,TR}$	Taylor expansion
🟥	$f_{n,NM}$	Newton Method
⬜	$f_{n,SD}$	Steepest Descent Method

f_1:



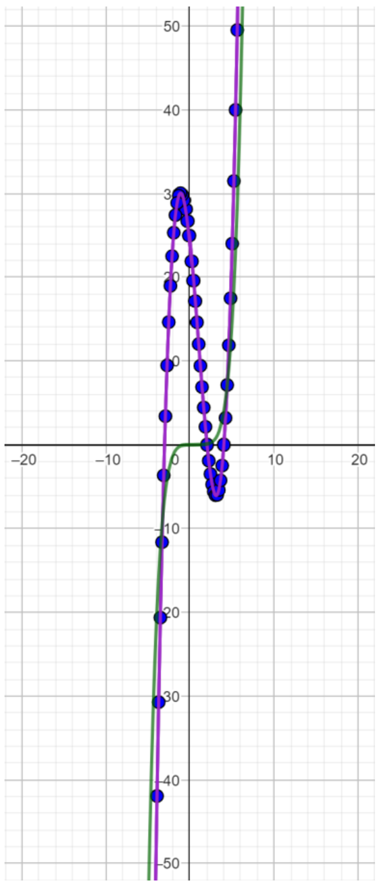
For the first function, it can be seen that both the Newton method and the Steepest Descent method give very good results. However, if the functions are now considered to infinity, the better result becomes visible (Newton method has a smaller error). In the interval where the measured points lie, both are very well approximated.

f$_2$:



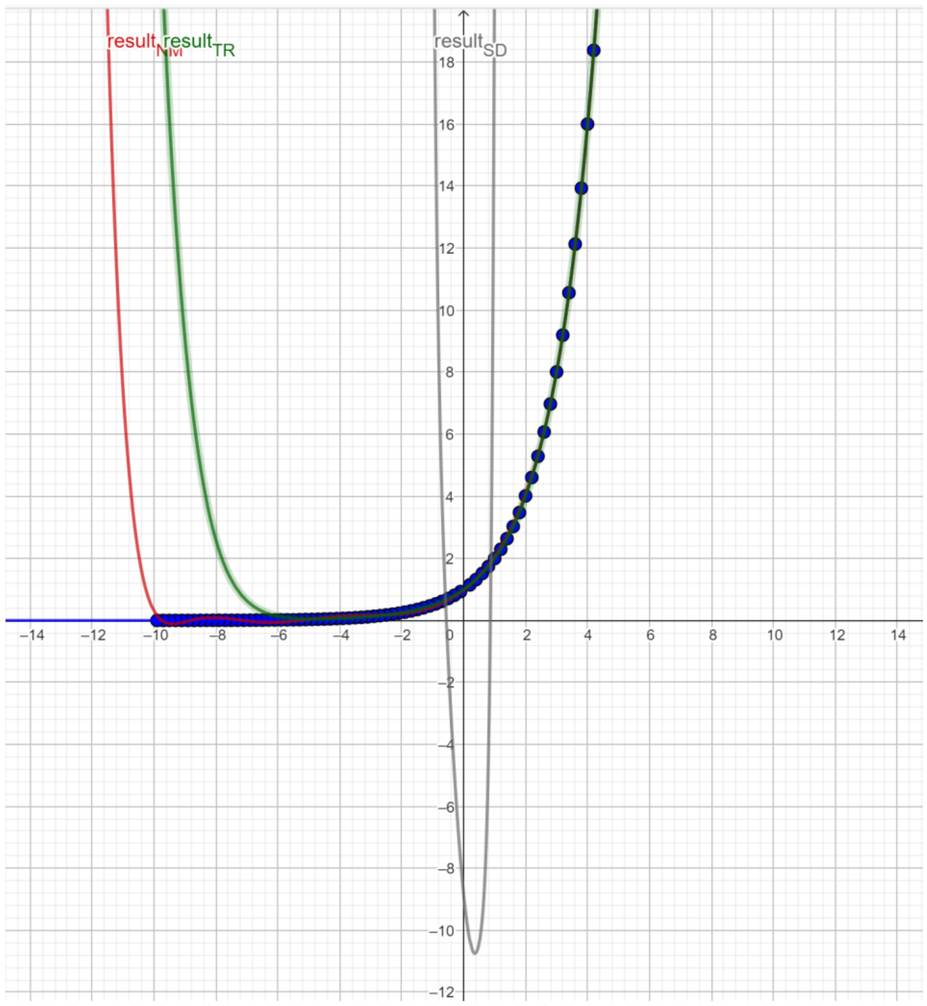
For the second function we can already see a difference between Newton and Steepest Descent method.

f$_3$:



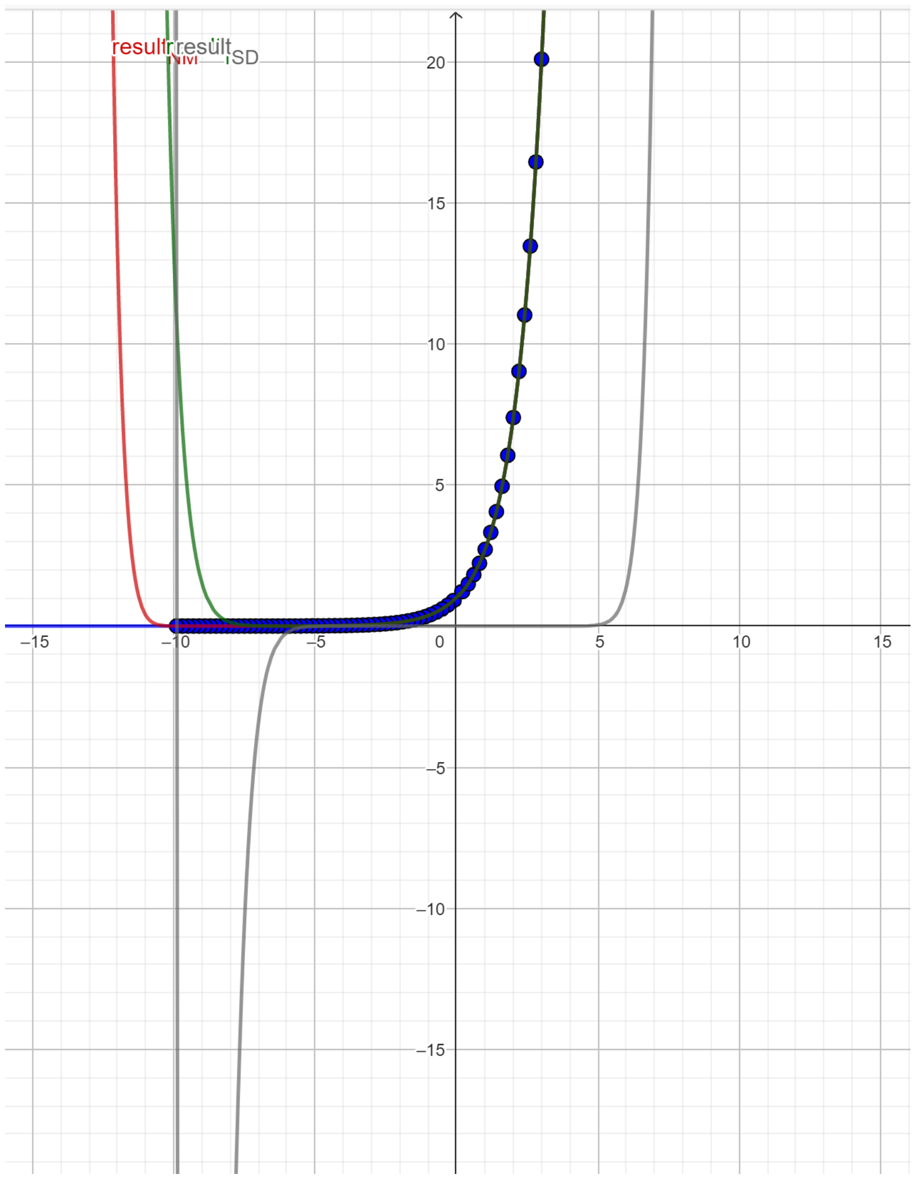
With the third function and the resulting higher complexity and polynomial degree, the advantages of Newton's method become clear. Both in the running time of the programme, which is much longer with the SD method, and in the accuracy.

f_4:



In this example, the Newton method has even higher accuracy than the Taylor expansion. With the Steepest Descent method, it is no longer possible to speak of a useful approximation.

f_5:



The last example underlines the advantages of the Newton method, also in contrast to Taylor expansion, which still gives a better result than the Steepest Descent method.

iii. I have tried this task many times and have written some methods for it, also to solve the task analytically. I have broken the problem down to the function for calculating alpha using the Steepest Descent method. For this I tried several different calculations for the Wolfe condition. The various methods for this have always led to different results, which were unfortunately all so far away from the analytical solution that I cannot accept the task as completed. Nevertheless, you can still find the methods I wrote for this task in the Python script.

iv. Problem 1: $(-10 * x^2 + 10 * y) ** 2 + (-x + 1)^2$
Problem 2: $(-x + 1)^2 + ((10x + 2(y + 6) * y)^2$
Problem 3: $((10x + 2(y + 6) * y)^2 + ((y + 1)^2)^2$
Problem 4: $((y + 1)^2)^2 + (-(5x)^2 + 8y + 4)^2$
Problem 5: $(-(5x)^2 + 8y + 4)^2 + ((x + y)^2)^2$

Problem	**Local minimizer**	$\tilde{x}$	$\|\nabla f(\tilde{x})\|$	$\|\tilde{x} - x^*\|$	**Iterations**
1	(1, 1)	(0.99, 0.99)	$7.5 * 10^{-7}$	$9.37 * 10^{-8}$	36
2	(1, -1)	(0.99, -0.99)	$8.2 * 10^{-7}$	$1.16 * 10^{-7}$	28
3	(1, -1)	(0.99, -0.99)	$9.8 *10^{-7}$	0.00057	49
4	(0, -0.5)	(0, -0.5)	$5.2 * 10^{-7}$	$4 * 10^{-9}$	28
5	(0.27, -0.27)	(0.27, -0.27)	$8.2 *10^{-7}$	0.000244	68

The differences between the local minimizers and the results of the Newton Method are quite small. I saved all results as txt-file named: “result4.txt” and add it my submission.