LING572 Hw5 and Hw6 (Optimization)
Due: 11pm on Feb 21, 2025
Points: 100 points for each assignment

Q1 (30 points): Write a script, **zero_order.sh**, that implements the three zero-order local optimization methods to **minimize** an objective function.

ANSWER: See hw.tar.gz

Q2 (30 points): Fill out Table 1 and answer the questions below.

Table 1: Zero-order methods

Expt id	func name	α	N	Method id	w^0	n	w^n	$func(w^n))$
Z1	f1	0.05	200	1	(1, 2)	200	(0.001, -0.005)	2
Z2	f1	0.05	200	2	(1, 2)	61	(-3.19e-16, -1.207e-15)	2
Z3	f1	0.05	200	3	(1, 2)	82	(-3.19e-16, -1.207e-15	2
Z4	f1	0.05	200	1	(1, 2.03)	200	(-0.002, -0.012)	2.001
Z5	f1	0.05	200	2	(1, 2.03)	62	(-3.19e-16, -0.02)	2.001
Z6	f1	0.05	200	3	(1, 2.03)	84	(-3.19e-16, -0.02)	2.001
Z7	f1	0.05	200	grad	(1, 2)	200	(7.055e-10, 1.411e-09)	2
Z8	f1	0.05	200	grad	(1, 2.03)	200	(7.055e-10, 1.432e-09)	2
Z9	f1	0.05	200	1	(100, 20)	200	(91.898, 14.92)	8669.8
Z10	f1	0.05	200	2	(100, 20)	200	(90, 20)	8502
Z11	f1	0.05	200	grad	(100, 20)	200	(7.055e-08, 1.411e-08)	2

Answer the following questions based on Table 1:

- (a) Z1-Z3: At the end of iterations, which of the three methods approach got close, but does not reach, a minimum point? Why? Which reaches the minumum point with the least number of iterations? Why?

  ANSWER: In this case presented by this data, all three functions reached the minimum point. This is not always the case with random search, however, which varies each run and often produces a result that isn't quite the minimum. This is likely because it is sampling randomly and as a result is getting diminishing returns each time it produces a better value.

  Coordinate Search appears to reach the minimum point the quickest. I believe that this is because it is descending quicker than Coordinate Descent, as it chooses a new direction among four potential directions instead of alternating axes like Coordinate Search.

- (b) Z4-Z6: At the end of iterations, which methods did not reach the minimum point? Why? Which method has the lowest function value? Why?

ANSWER: None of them reached a minimum point. For Coordinate Descent, and Coordinate search, this is likely because they started approaching and asymptote that caused their stopping criteria to terminate, as there was no visible improvement. I am not sure if these would eventually reach the minimum given python's limitations on overflow.

Random search also did not reach a minimum with a similarly asymptotic descent. Random search's asymptote appears to get slightly closer than the other two methods however. My interpretation is that, by having random sampling instead of concrete mathematical search, Random search's asymptote is a bit more volatile, allowing it to dip lower than the other methods by chance.

- (c) Z7-Z8: Does gradient descent reach the minimum point? Why or why not?

  ANSWER: Gradient descent leverages the directional information provided by the gradient of the function. This allows gradient descent to make more informed decisions than the other algorithms, as the slope of the function is accounted for.

- (d) Z9-Z11: Which method reaches the minimum point?

  ANSWER: Only gradient descent reaches the minimum point here. My thought is that this is due to gradient descent's more intelligent design, as described above.

- (e) Z1-Z11: What conclusion can you draw from those experiments?

  ANSWER: It seems that gradient descent is the most reliable method of finding a minumum. The other three may be computationally less expensive, but are hindered by their own design, giving answers that are close to, but don't necessarily reach the minimum. Gradient descent is also the least impacted by a w^0 that is distant from the minimum as seen in Z9-Z11, whereas the other three functions will be impacted by even a slight shift away from the optimal minimum point, as seen in Z4-Z6.

Q3 (30 points): Write a script, **grad_desc.sh**, that implements the gradient descent algorithm.

ANSWER: See hw.tar.gz

Q4 (30 points): Run your Q3 code with different initial values and fill out Table 2:

ANSWER: See Table 2

Q5 (40 points): What's your observation when coming the experiments in Q4? One-sentence answers should be sufficient. For instance, for (a), you can simply say that different w^0 values lead to different global minima.

- (a) E1 and E2:

  ANSWER: To quote the prompt here, it appears that different w^0 values lead to different global minima.

Table 2: Gradient Descent Results with $\alpha = 0.01$ and $N = 200$, but different w^0

Expt id	func name	w^0	converge?	w^{200}	func(w^{200})
E1	g1	-1.8	yes	-2.618	-1
E2	g1	1.8	yes	1.571	-1
E3	g2	-1.8	yes	-2.561	-0.33
E4	g2	1.8	yes	1.5366	-0.759
E5	g3	-1.8	yes	-0.031658	0.201
E6	g3	1.8	yes	0.031658	0.201
E7	g4	-1.8	yes-but-diverge	-1.83e+25	-6.132e+75
E8	g4	1.8	yes	0.15081	0.003
E9	g5	-1.8	yes	-1.4346	-0.161
E10	g5	1.8	no	0.91236	0.213
E11	g6	-1.8	yes-but-diverge	-3.601e+17	1.589e+108
E12	g6	0.77	yes	0.768	1
E13	g6	0.05	yes	0.230	1
E14	g6	0.3	yes	0.5	0.476
E15	g6	0.76	yes	0.5	0.476
E16	f1	(1, 2)	yes	(0.0176, 0.0352)	2.002
E17	f2	(1, 2)	yes	(0.018, 1.583e-18)	0.0003
E18	f3	(1, 2)	yes-but-diverge	(0.446, 5.731e+15)	3.9412e+33
E19	f4	(1, 2)	yes-but-diverge	(0.902, 4.273e+16)	2.192e+36

- (b) E3 and E4:

  ANSWER: Similar to (a), we can see if difference in global minima found with different w^0 inputs.

- (c) E5 and E6:

  ANSWER: Again, we see a difference in minima as a result of different w^0. This time we can see that the two different minima are actually sign-flipped versions of each other, mirroring their sign-flipped w^0.

- (d) E7:

  ANSWER: A divergence! The first that we've seen. Given that g4 represents x^3, it appears that starting in the negatives causes the function to dive downwards toward a minimum that does not exist. If allowed to continue endlessly, this would likeley continue to infinity.

- (e) E8:

  ANSWER: Here we see a positive input leading to a convergence and an answer that is close, but not equal, to 0. This is likely because, starting in the positives, the formula is navigating the portion of the curve where the slope itself is approaching 0. Interestingly, 0 is not actually a minimum, it just appears to be one as the formula appraoches it.

- (f) E9 and E10:

  ANSWER: Here we see the negative input causing convergence while the positive input does not. The negative input is closer to the true minimum, possible contributing to this discrepancy. Meanwhile, the positive input is navigating a less steep slope than the negative input, likely contributing to it's slower progress.

- (g) E11:

  ANSWER: Here we see our second yes-but-diverge. Looking at the output file, it seems that the input of -1.8 produces a huge output that causes a massive step to the right. This is then compensated by another step to the left. Both of these steps skip over the minimum, and after the second step we can see that the slope seems to be simply too steep for the method to find a proper minimum.

- (h) E12 and E13:

  ANSWER: In both cases, we see a small, positive input leads to convergence and a found minimum. Interestingly, both of these cases seem to be finding the inflection points on either side of the minimum rather than the true minimum itself. This seems to be because the starting points themselves are close to said minima.

- (i) E14:

  ANSWER: This reaches the true minimum! This seems to be because this input is, in fact, very close to the true minimum.

- (j) E12 and E15:

  ANSWER: The difference of 0.01 here appears to affect the direction that method uses to search. 0.77 prompts the algorithm to find the inflection point, while 0.76 directs it towards the true minimum.

- (k) E11-E15:

  ANSWER: These results demonstrate the volatile nature of an algorithm with precipitous slopes and multiple inflection points. One can see how very small changes in the input can dramatically affect the algorithm's ability to find the minimum.

- (l) E16-E19:

  ANSWER: In these four, we can see that the first two functions are able to converge and find their minimums, while the latter to do not. This seems to be related to the fact that the latter two functions have much steeper slopes than the first two, complicating the algorithms computations in the same way that we see in some of the earlier calculations.

- (m) Summarize your findings from E1-E19:

  ANSWER: The slope of the functions being analyzed has a very direct affect on our algorithm's ability to find the local minimum. Also, in functions with multiple minima or inflection points, the starting point of our algorithm is very important, as it could direct towards something that is not actually the minimum we are seeking.

Q6 (15 points): Effect of the learning rate.

- When running the gradient descent algorithm, set initial point to 100, and iternation number to 200.
- Save the output files to q6/L_i and submit them.
- (12 points) Fill out Table 3, which is the same as Table 2, except that here we fix the initial point w^0 and experiment with different learning rates α.
- (3 points) What conclusion can you draw from Expt L1-L6?

  ANSWER: The learning rate is most effectively able to find the true minimum when between 0.1 and 0.6, exceeding that boundary in either direction prevents convergence. Looking at the output files, one can see that a value ≤ 0.01 is too small, slowing the rate of convergence. A value ≥ 1.0, however, is too large, causing the algorithm to skip over the minimum, preventing convergence entirely.

Table 3: Gradient Descent Results with $w^0 = 100$ and $N = 200$, but different learning rate α

Expt id	func name	α	converge?	w^{200}	func(w^{200})
L1	g3	0.01	no	1.759	3.293
L2	g3	0.1	yes	4.162e-15	0.2
L3	g3	0.3	yes	1.917e-15	0.2
L4	g3	0.6	yes	1.921e-15	0.2
L5	g3	1.0	no	100	10000
L6	g3	1.01	no	5248.5	2.755e+07

Submission: Submit the following to Canvas:

- Your note file *readme.(txt | pdf)* that includes Tables 1-3, answers to Q2, Q5-Q6, and any notes that you want the TA to read.
- hw.tar.gz that includes all the files specified in dropbox/24-25/572/hw5-hw6/submit-file-list:
  - Source code and shell scripts for Q1 and Q3.
  - The output files produced in Q2: q2/Z_i (i=1-11)
  - The output files produced in Q4: q4/E_i (i=1-19)
  - The output files produced in Q6: q6/L_i (i=1-6)
- Make sure that you run **check_hw5_hw6.sh** before submitting your hw.tar.gz.