CSS 322: Scientific Computing
Project #2: Optimization
Due Date: December 1st, 2021 @17:00
This is an individual project.

Preparation Step

For this homework, we will generate some random numbers. However, for ease of grading, you will need to run the same seed values. Later on, you will need to restore your random number sequence, so that you can use your random number generator normally. Note that, it will unlikely affect your computer if you do not do these steps. But the different results may cause the score to be lowered.

In Matlab, every time you are running the homework, before running anything run:

s = rng;

Note that you should not use s variable until the end of the session. At the end of the session run:

rng(s);

Introduction:
For this project, you will experiment with finding the optimal values using the various method. The function that we will play with is the following

$$f(x) = \sin(x_1) - \cos(x_2)$$

Part I: Newton's method
For this part, you will implement Newton's method for unconstraint optimization in handout #12: optimization. For this part, before calling any random value, set rng using the command: rng(100);

Let x = rand(2,1), be the initial guess for finding the minimum value of f(x).

You are to hand in all your Matlab code for this part, including Newton's method implementation for this particular f(x), and report x and the corresponding f(x) at k = 0, 5, 10, and 15.

Part II:
For this part, you will implement a naïve random search algorithm in handout #18: global optimization. For this part, before calling any random value, set rng using the command: rng(1000); We will use f(x) but we will perturb it with some random value. You are to create the mesh and the perturbed f(x) for the search using the following functions.

[X,Y] = meshgrid(0:0.01:2*pi,0:0.01:2*pi);

```
ZZ = rand(size(X));
Z = sin(X)-cos(Y) + ZZ;
```

(You can try to see what the surface of the perturbed f(x) looks like using the command "surf(X,Y,Z)".)

Then, you are to find the optimal value in Z using the naïve random search method by randomly picking $x^{(0)}$ using the command x = randi(629,2,1). For the neighborhood of x, you can use the 5x5 grid surrounding x = (x_1,x_2), shown below) for as long as it does not fall within the boundaries.

x_1-2,x_2-2	x_1-2,x_2-1	x_1-2,x_2	x_1-2,x_2+1	x_1-2,x_2+2
x_1-1,x_2-2	x_1-1,x_2-1	x_1-1, x_2	x_1-1,x_2+1	x_1-1,x_2+2
x_1,x_2-2	x_1,x_2-1	x_1,x_2	x_1,x_2+1	x_1,x_2+2
x_1+1,x_2-2	x_1+1,x_2-1	x_1+1,x_2	x_1+1,x_2+1	x_1+1,x_2+2
x_1+2,x_2-2	x_1+2,x_2-1	x_1+2,x_2	x_1+2,x_2+1	x_1+2,x_2+2

1	2	3	4	5
6	7	8	9	10
11	12	x_1,x_2	13	14
15	16	17	18	19
20	21	22	23	24

You are to randomly pick one of these points for the candidate, by calling the i = randi(24). Then, you can find the values of the neighbor by following the corresponding table cell above. For example, if i = 14, then your next candidate is x_1,x_2+2. You should check if the neighbor exists before continuing. If the neighbor does not exist (falls outside the matrix), simply generate the i value again using randi(24).

For this part, you are hand in all your Matlab code including the x value before picking the candidate its corresponding Z[x] at the iteration k = 0, 25, 50, 75, 100.

Part III [Extra Credits]:
For this part, we will search the same Z matrix from part II. However, you will implement the simulated annealing algorithm for this part. For this part, before calling any random value, set rng using the command: rng(2000); Use the acceptance probability function stated on page 7th of the Global optimization handout. For the cooling temperature, use the Hajek function stated on page 9th of the Global optimization handout. Set the $\gamma = 2$. For the coin toss, simply use the function rand(). Specifically, if rand() < p(k; f ($z^{(k)}$); f ($x^{(k)}$)) consider that you get HEAD. For every other parameter setting, use those specified in part II.

For this part, you are hand in all your Matlab code including the x value before picking the candidate its corresponding Z[x] at the iteration k = 0, 25, 50, 75, 100.