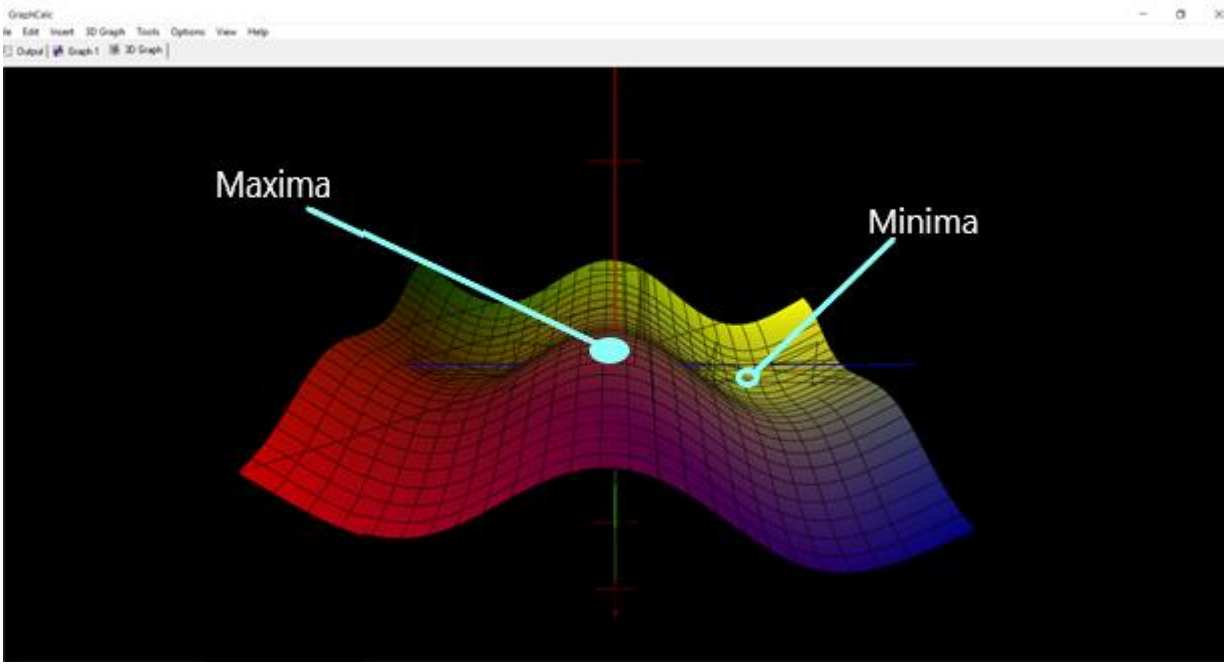
Project 4 - Evolutionary Computation

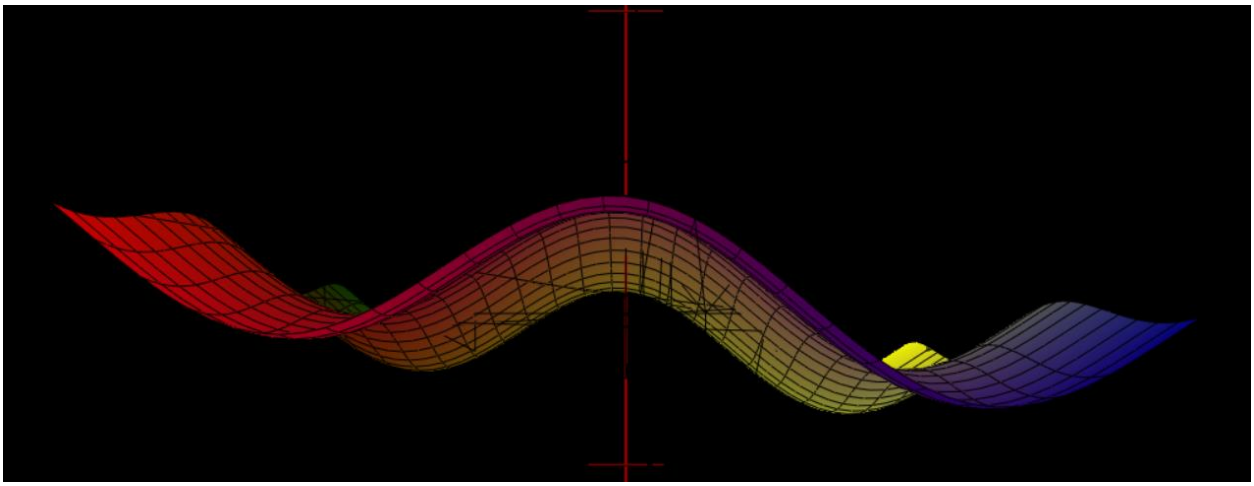
Section A:

The function I chose is, $f(x, y) = 0.75 * \cos(x) - 0.25 * \sin(y) - 0.1 * x - 0.1 * y$
I chose this function because it has a clearly defined minima as well as maxima. Having my function set up in this way allowed me to check my SGA code solutions to ensure my results were correct. To view the graph of my function, I used the graphing software, "GraphCalc", to examine the graph and visually find the approximate maxima and minima locations within my specified range of -5 to +5.

Below is an image from GraphCalc of my function. I have labeled the maxima and minima spots on the graph to show where I can visually see each of them.



As well, here is a side view of the graph to help better understand the orientation of the graph.



Using GraphCalc, there was no access to a tool to pinpoint the minima or maxima. And, not having a graphing calculator capable of rendering this, I made approximation of the minima and maxima from examining the graph and the tick marks labeled with it. I was able to come up with the following approximations;

$$Minima \approx (+3, +2)$$

$$Maxima \approx (-0.3, -2.1)$$

Section B:

I was able to successfully get the Genetic Algorithm working with the appropriate parameters of X and Y, for my function , $f(x, y) = 0.75 * \cos(x) - 0.25 * \sin(y) - 0.1 * x - 0.1 * y$. When running the program to find the minima, it produced the results $(3.2766, 1.9830)$ with a fitness value of -1.4982. This matches up nicely with the prediction made above, corresponding to the graph of the function. Having the fitness value shown as -1.4982, tells us that when looking for the minima, it wants the lowest fitness value. Below is an image of the results for running minima.

```
==========================================================
Best result over all generations:
11010011111000011011001011000011
Decoded valueX = 3.2766  Decoded valueY = 1.9830  Fitness = -1.4982
```

When solving for the maxima, it produced the results $(-0.1380, -1.9531)$ with the fitness 1.1839. This also matches up with the prediction made above, corresponding to the graph of the function. With the fitness value shown as 1.1839, when looking for maxima, this tells us that it wants the highest fitness value. Below is an image of the results of the minima.

```
=========================================================
Best result over all generations:
01111100011101110100111000000000
Decoded valueX = -0.1380  Decoded valueY = -1.9531  Fitness = 1.1839
```

Overall it can be seen that this program is generating the proper results and is able to find the optimum.

When experimenting with different parameters, I found that when adjusting PMUT (Probability to flip each bit) it performed better when I had its value set at 0.03. With this setting, the population converged to an answer quicker and had a majority of the population towards the end generations with close values to one another, which were also close the result. This is probably because, with the size and scale I'm using, flipping a bit randomly with more frequency could have negative effects instead of positive.

When it came to making adjustments to the population size and the number of generations, I found that adjusting the population size did not really help too much in this scenario. I could imagine when solving a more complicated problem that having a larger population size on top of more generations, than this can help produce better results. Even just comparing this to nature. When you have a decent sized colony of animals, not too many and not too few, and you wait a long enough time, then evolution will be able to work its magic by reinforcing the traits passed along by the idea of survival of the fittest. Especially when making these adjustments on the sin-bowl function. Since, this function is rather simple for the algorithm. These modifications have small effects.

Section C:

The new genetic operator that I ended up creating was a modification of the crossover function. After going over the different functionalities of each operator in the genetic algorithm, I noticed that the crossover function was rather simple. It seemed some modifications could be made to this function to make it more sophisticated. My thought process behind this was that, in nature when traits are being selected between parents for their offspring, the selection process is much more complex than just swapping over a chunk of information up until a random point. The selection process will pick and choose different pieces at different intervals, causing much more diversity. So I tried at first playing around with the segment used to indicate the site at which the crossover would take place. I decided to add another random segment, so that the crossover would happen twice instead of just once.

After having success with this first attempt I made, I decided to research more on crossover operators and what other improvements can be made. I found that the operator I had created was known as Two-Point crossover. I also discovered a more unique operator call Uniform Crossover.

Although I did not fully think of Uniform Crossover, the premise for it, is what I was intrigued with originally, so I wanted to implement this idea as well in order to try and see how much

better this could perform. So, describing this process, each time the crossover moves through the 16-bit string, it will randomly decide at each bit position, whether that bit should be swapped over or stay. I was able to implement this idea by removing the random segment variable and replacing it with a variable that, upon each iteration, would randomly choose 0 or 1. If the value is 0, then no swap for that bit would take place, if the value is 1, then that bit would be swapped. Examining this operator, it's evident that more selection and diversity will be passed onto the child between the two parents. This going back to my original thoughts and inspiration from nature. Children have complex variants of traits from both the Mother and the Father. So, going from One-Point crossover, this looks much more sophisticated.

Section D:

For my first operator which is now labeled, Two-Point Crossover, it was able to improve the Genetic Algorithm slightly. When running tests from both the sin-bowl function and the function I created, I saw that the values seemed to converge to a correct answer slightly faster. As well the population seemed to be more within an isolated range, closer to the result later in the generations.

As far as the improvements that Uniform Crossover had on the f(x,y) Genetic Algorithm, it ended up working rather well. It was able to find the optimum for the given problem in less generations. Comparing this to One-Point, after 100 generations, the population's values were more uniform. However, although hard to tell, there is a slight difference between the two, as well as comparing both their finesses. And especially the fact that the optimum for Uniform is closer to what we are looking for.

Below are 100 generations of One-Point Crossover with my function;

```
   3.2765       1.7163  -1.4898 1101001111100000101010111101111        3.2765  1.7163
   3.2765       0.6146  -1.2764 1101001111100000100011111011011        3.2765  0.6146
   3.5893       1.8746  -1.4610 1101101111100010101011111111101        3.5893  1.8746
   3.2670       1.8646  -1.4966 1101001110100010101011111011011        3.2670  1.8646
   3.2765       1.8307  -1.4955 1101001111100000101011011011101        3.2765  1.8307
   3.2777       1.8640  -1.4966 1101001111101000101011111011011        3.2777  1.8640
   3.4340       1.8652  -1.4873 1101011111101000101011111011111        3.4340  1.8652
   3.2670       1.8356  -1.4957 1101001110100010101011011111101        3.2670  1.8356
   3.1935       1.8750  -1.4944 1101000111000000101011111111111        3.1935  1.8750
   4.4484       1.8750  -1.0666 1111000111100000101011111111111        4.4484  1.8750
  -1.7236       4.3750  -0.1434 0101001111100000111011111111111        -1.7236 4.3750
   3.2765       1.8725  -1.4968 1101001111100000101011111101111        3.2765  1.8725
   3.2765       0.6225  -1.2788 1101001111100000100011111101111        3.2765  0.6225
   3.2643       1.8750  -1.4968 1101001110010000101011111111111        3.2643  1.8750
   3.1837       1.8307  -1.4924 1101000110000000101011011011101        3.1837  1.8307
=====================================================
Best result over all generations:
11010011111000001010111111111111
Decoded valueX = 3.2765  Decoded valueY = 1.8750  Fitness = -1.4969
```

Below are 100 generations of Uniform Crossover with my function.

```
   3.1254       1.9374  -1.4896 11010000000000101011000110011000        3.1254  1.9374
   3.1254       1.9375  -1.4896 11010000000000101011000110011001        3.1254  1.9375
   3.2817       1.9374  -1.4979 11010100000000101011000110011000        3.2817  1.9374
   -1.8551      1.9375  -0.4520 01010000100000101011000110011001        -1.8551 1.9375
   3.3012       2.0144  -1.4978 11010100100000101011001110010001        3.3012  2.0144
   3.2817       1.9363  -1.4979 11010100000000101011000110010001        3.2817  1.9363
   3.1450       1.9361  -1.4916 11010000100000101011000110010000        3.1450  1.9361
   3.1254       0.8436  -1.3336 11010000000000101001010110011000        3.1254  0.8436
   3.2817       1.9375  -1.4979 11010100000000101011000110011001        3.2817  1.9375
   3.1450       1.9351  -1.4916 11010000100000101011000110001001        3.1450  1.9351
   3.6137       2.2500  -1.4488 11011100100000101011100110011001        3.6137  2.2500
   3.1450       1.9374  -1.4916 11010000100000101011000110011000        3.1450  1.9374
   3.1254       1.9363  -1.4896 11010000000000101011000110010001        3.1254  1.9363
=======================================================
Best result over all generations:
11010100000000101011001011001000
Decoded valueX = 3.2817  Decoded valueY = 1.9837  Fitness = -1.4982
```

All in all, I have to say that I think overall, the Uniform Crossover is more advantageous especially working with larger more complex problems. However, after running through several different tests, it seems the Two-Point crossover can offer somewhat better improvements with the problems being addressed in this assignment here, being less generations and a smaller population size to handle the scope.