MSE 426:

Introduction to Engineering Design Optimization

Lab 4 – Part 1

OASIS - Simulation-Based Optimization

Selikem K

Selikem Kwadzovia | 301303898 |

Sepehr Rezvani | 301291960 |

Tuesday, March 9th, 2021

Table of Contents

Introduction

The main purpose of lab 4 is to compare the Optimization Assisted System Integration Software (OASIS) tools and MATLAB when used to solve various optimization problem. This report deals with black-box global optimization, and the next will deal with simulation-based optimization. Black-box method are favored over gradient-based algorithm for two reasons: Firstly, relying heavily on gradient information can be time consuming and cost inefficient. In addition, real world problems do not always have access to previous history. Gradient-based algorithms are used for solving local optimization problems, which are not as accurate as global optimization methods. Unlike these algorithms, OASIS automatically solves optimization problem and does not require manual implementation and is more user friendly. However, we will look deeper into their similarities and differences by finding the optimum of "Keane's Bump" and "Shubert" functions.

Results and Analysis

As mentioned in the introduction, the optimization methods do not rely on gradient data for first part of lab 4. In this section, OASIS and different optimization methods in MATLAB are used to find the optima of “Keane’s Bump” and “Shubert” functions. In MATLAB, each function will be optimized by fmincon (local optimizer) and ga (global optimizer).

Keane’s Bump function is written in equation (1), and its constraints are in equations (2), (3) and (4). Similarly, Shubert function is written in equation (5), with its constraint in equation (6).

$$f(x) = -\left|\left[\sum_{i=1}^{d}\cos^4(x_i) - 2\prod_{i=1}^{d}\cos^2(x_i)\right] / \left(\sum_{i=1}^{d} i x_i^2\right)^{0.5}\right| \tag{1}$$

$$0.75 - \prod_{i=1}^{d} x_i < 0 \tag{2}$$

$$\sum_{i=1}^{d} x_i - 7.5d < 0 \tag{3}$$

$$x \in [0, 10]; d = \#\ variables = 5 \tag{4}$$

$$f(\boldsymbol{x}) = \prod_{i=1}^{d}\sum_{j=1}^{5} j\cos\left((j+1)x_i + j\right) \tag{5}$$

$$x \in [-5.12, 5.12]; d = \#\ variables = 10 \tag{6}$$

Keane’s Bump Function

This function is optimized by fmincon optimizer in MATLAB, and results for 10 different runs are shown on Table 1. Similarly, 10 runs for ga optimizer are shown in Table 2.

The average minimum function values of 10 runs for fmincon and ga method are calculated and written in the last row of tables 1 and 2. The average absolute value of the minimum function value by the ga method is 12.9% lower than fmincon optimizer. Note that maximum function evaluations for fmincon was set on 1000, but was not possible to run for ga. This is due to the inability to set the maximum evaluations. The maximum generations were set to 1000, but the number of function evaluations for in Table 2 and Table 5 are above 1000.

In order to understand how to setup OASIS, refer Appendices section of this report. To optimize this function, the instructions in the appendices were followed, and optimizations were run for 2 periods limited to 1000 function evaluations. Results are shown on Table 3. Considering that OASIS is more accurate, we must compare its results with the last row of tables 1 and 2. We can see optimal value in Table 3 is -0.63, and it is closer to MATLAB’s fmincon method.

Table 1 – MATLAB fmincon

Run	Minimum Function Value	Location (5 variables)	Number of Iterations	Number of Function Evaluations
1	-5.284441e-01	3.100017e+00,1.542767e+00,2.341016e-01,2.921774e+00,2.292695e-01	64	475
2	-3.829219e-01	5.484439e-01,5.355001e-01,1.668246e+00,3.029608e+00,5.052710e-01	47	307
3	-5.998955e-01	3.082969e+00,3.006695e+00,2.410614e-01,2.377583e-01,1.411691e+00	34	264
4	-4.570255e-01	1.757276e+00,5.460697e-01,3.018521e+00,5.150327e-01,5.027426e-01	27	170
5	-3.724744e-01	6.228002e+00,3.086907e+00,8.568915e-03,1.515918e+00,3.004923e+00	41	410
6	-3.073934e-01	6.246068e+00,3.106074e+00,9.213173e-02,9.184681e-02,4.569013e+00	34	297
7	-3.406663e-01	6.237175e+00,3.095920e+00,3.073289e+00,3.050644e+00,1.514059e+00	33	256

8	-5.998952e-01	3.082968e+00,3.006694e+00,2.410625e-01,2.377594e-01,1.411690e+00	35	278
9	-4.515378e-01	4.636511e+00,3.842772e-01,3.033959e+00,3.747093e-01,3.702739e-01	59	398
10	-6.221505e-01	3.097128e+00,1.608065e+00,5.554811e-01,5.306992e-01,5.108351e-01	42	291
Average	-0.41339605	-	-	-

Table 2 – MATLAB ga

Run	Minimum Function Value	Location (5 variables)	Number of Iterations	Number of Function Evaluations
1	-3.733386e-01	-3.114670e+00,3.087282e+00,3.059364e+00,-1.516483e+00,3.004591e+00	3	8998
2	-5.998743e-01	-3.072236e+00,3.007532e+00,2.416648e-01,2.359279e-01,-1.421753e+00	3	7421
3	-3.186063e-01	4.701601e+00,3.128856e+00,3.119125e+00,-3.092797e+00,-3.076654e+00	2	5608
4	-3.517176e-01	-3.119675e+00,-1.551147e+00,3.071172e+00,-3.047566e+00,-3.022071e+00	3	9045
5	-3.193468e-01	4.682000e+00,3.101774e+00,-3.082020e+00,-3.061932e+00,3.042234e+00	3	8293
6	-3.074591e-01	3.123591e+00,4.655552e+00,3.084008e+00,3.067769e+00,3.049656e+00	3	8904
7	-3.193463e-01	4.682148e+00,-3.102553e+00,3.082395e+00,3.062375e+00,-3.042710e+00	3	8528
8	-3.733385e-01	-3.114653e+00,-3.085906e+00,-3.059272e+00,-1.516537e+00,3.004825e+00	3	9327
9	-3.193466e-01	4.681930e+00,-3.102538e+00,3.081585e+00,3.062158e+00,-3.041256e+00	3	8058
10	-3.193467e-01	4.681737e+00,3.101927e+00,3.082102e+00,-3.062363e+00,-3.042377e+00	3	8857
Average	-0.36017208	-	-	-

Table 3 – OASIS Results

Problem Summary	
Problem type	Optimization
Number of Inputs	5
Number of Objectives	1
Number of Constraints	2

Run Summary	
Iteration Count	1000
Run Time	0:16:36.619
Simulation Time	0:00:00.000
Session Time	0:37:03.758

Result Summary

Reference Source: Worst point from Current Run

Objective Performance

Objective Name	Weight	Reference Value	Best Value	Improvement	Difference
f1	1	-0.03333249277	-0.6343678255	1803.1515%	0.6010353327

Best Band

Fitness: N/A

Size: 148

Ranges

Symbol Name	Type	Lower Bound	Upper Bound
x1	Input	2.820668486	3.232588524
x2	Input	2.639186094	3.127598119
x3	Input	1.374153614	1.756943981
x4	Input	0.1266847182	0.3118118674
x5	Input	0.1925052965	0.4793669939
f1	Objective	-0.6343678255	-0.5494252053
c1	Constraint	-1.025236238	-0.00001722260543
c2	Constraint	-29.77935857	-29.1895921

Shubert Function

This function is optimized by fmincon optimizer in MATLAB, and results for 10 different runs are shown on Table 4. Similarly, 10 runs for ga optimizer are shown in Table 5.

The average Minimum Function Values of 10 runs for fmincon and ga method are calculated and written in last row of tables 4 and 5. Average absolute value of Minimum Function Value by fmincon method is 97.4% lower than ga optimizer.

Similar to the other function, results are shown on Table 6. We can see best value in Table 6 is -1.88E10, and it is closer to MATLAB's fmincon method. Although MATLAB is much simpler and less time consuming for this lab, for more complex optimization problems OASIS is preferred.

Table 4 – MATLAB fmincon

Run	Minimum Function Value	Location (10 variables)	Number of Iterations	Number of Function Evaluations
1	-1.381507e+10	-4.963491e+00,5.094295e+00,4.276042e+00,1.320140e+00,4.275846e+00,2.299233e+00,-4.963475e+00,3.280027e+00,-4.964017e+00,4.275983e+00	83	980
2	-7.049742e+13	1.026120e+00,1.320004e+00,-8.003211e-01,-8.003211e-01,-8.003211e-01,1.320004e+00,-8.003211e-01,1.320004e+00,-8.003211e-01,-8.003211e-01	66	911
3	-1.398916e+15	5.106149e+00,-8.003967e-01,-8.004598e-01,-8.003153e-01,-3.003977e+00,5.119947e+00,-2.007312e+00,-8.000615e-01,-8.007874e-01,-8.021318e-01	75	1004
4	-3.006812e+07	5.033273e+00,-3.003076e+00,-2.004022e+00,-3.497241e+00,3.277401e+00,1.315640e+00,1.320470e+00,-5.076864e+00,1.320004e+00,-8.003210e-01	83	1002
5	-4.176259e+12	8.217839e-01,2.785934e+00,4.679043e-01,-8.003211e-01,-8.003211e-01,3.342439e-01,-8.003211e-01,3.342439e-01,3.342439e-01,3.342439e-01	50	771
6	-5.052740e+12	-3.215320e+00,-3.497252e+00,4.858056e+00,4.858057e+00,8.217881e-01,-2.510877e+00,-1.953638e-01,-1.953864e-01,5.119901e+00,8.217861e-01	81	1000

7	-1.315991e+14	3.408611e-02,-8.003207e-01,-8.003211e-01,3.342474e-01,4.276007e+00,2.299222e+00,5.119906e+00,1.320181e+00,3.344687e-01,-8.036173e-01	73	997
8	-2.323383e+12	-1.425128e+00,3.772308e+00,4.858056e+00,3.772308e+00,-1.112649e+00,4.275983e+00,-2.007203e+00,2.299229e+00,-2.007203e+00,-8.003211e-01	72	995
9	-3.725390e+11	-5.117133e+00,4.858238e+00,-5.119691e+00,-5.119788e+00,4.858118e+00,-4.482551e+00,-5.117783e+00,-5.119995e+00,4.858043e+00,-5.109617e+00	83	1000
10	-1.307099e+12	2.785934e+00,5.930184e-02,5.120000e+00,-8.003211e-01,-8.003211e-01,2.299229e+00,4.275983e+00,-2.007203e+00,-8.003212e-01,-8.003210e-01	55	794
Average	-1.61426E+14	-	-	-

Table 5 – MATLAB ga

Run	Minimum Function Value	Location (10 variables)	Number of Iterations	Number of Function Evaluations
1	-5352428145109474	3.648224e-01,-1.425126e+00,-1.425098e+00,4.858099e+00,4.858060e+00,-1.425127e+00,4.858082e+00,4.858073e+00,4.858060e+00,4.858089e+00	5	49700
2	-2.502098e+15	-2.968573e+00,4.858060e+00,-1.953789e-01,-1.425124e+00,4.858066e+00,4.858052e+00,4.858066e+00,-1.425101e+00,4.858053e+00,-1.953729e-01	5	49700
3	-5352428337842454	3.648542e-01,-1.425109e+00,-1.425127e+00,-1.425120e+00,4.858053e+00,-1.425111e+00,4.858078e+00,4.858056e+00,4.858052e+00,-1.425114e+00	5	49700
4	-7990397624235936	-1.690639e+00,-8.003226e-01,-8.003206e-01,-8.003172e-01,4.276010e+00,-8.003356e-01,-8.003130e-01,-8.002828e-01,4.275980e+00,-8.003338e-01	5	49700
5	-5352428343859024	3.648652e-01,-1.425112e+00,4.858061e+00,-1.425134e+00,4.858073e+00,4.858051e+00,-1.425127e+00,4.858085e+00,4.858080e+00,-1.425117e+00	5	49700
6	-4076419225746123	-3.003125e+00,4.858069e+00,4.858068e+00,4.858061e+00,4.8	5	49700

		58062e+00,-1.953786e-01,-1.425128e+00,-1.425132e+00,-1.425133e+00,-1.425123e+00		
7	-5347066721512331	3.280038e+00,-1.425115e+00,-1.425106e+00,4.858063e+00,-1.425123e+00,-1.425099e+00,-1.425127e+00,4.858055e+00,4.858079e+00,-1.425118e+00	5	49700
8	-17442786608826540	3.772382e+00,-8.002647e-01,-8.003223e-01,-8.002892e-01,-8.003175e-01,-8.002973e-01,-8.003262e-01,-8.002793e-01,-8.003225e-01,-8.003555e-01	5	49700
9	-5347066804394502	3.280053e+00,4.858057e+00,4.858057e+00,4.858056e+00,-1.425128e+00,4.858090e+00,-1.425125e+00,4.858088e+00,4.858058e+00,-1.425124e+00	5	49700
10	-4.391113e+15	3.280062e+00,-1.425126e+00,-1.425125e+00,4.858065e+00,4.858055e+00,4.858077e+00,-1.953740e-01,-1.425128e+00,-1.425108e+00,-1.425117e+00	5	49700
Average	-6.31542E+15	-	-	-

Table 6 – OASIS Results

Problem Summary	
Problem type	Optimization
Number of Inputs	10
Number of Objectives	1
Number of Constraints	0

Run Summary	
Iteration Count	1000
Run Time	0:04:56.091
Simulation Time	0:00:00.000
Session Time	0:12:52.326

Result Summary

Reference Source: Worst point from Current Run

Objective Performance

Objective Name	Weight	Reference Value	Best Value	Improvement	Difference
f1	1	5504147309	-18780631208	441.2087%	24284778517

Best Band

Fitness	N/A
Size	278

Ranges

Symbol Name	Type	Lower Bound	Upper Bound
x1	Input	4.78112538	4.91773338
x2	Input	-1.50139996	-1.365499472
x3	Input	4.835803774	4.941667392
x4	Input	-1.496259775	-1.349808906
x5	Input	-1.490586862	-1.324028637
x6	Input	1.228073147	1.397356377
x7	Input	4.801614973	4.918156892
x8	Input	-0.2124831283	-0.06543106577
x9	Input	4.762417221	4.894041219
x10	Input	4.779235831	4.868006023
f1	Objective	-18780631208	-14100136290

Conclusion

In part 1 of this lab, we compared the optima found using fmincon, genetic algorithms, and the OASIS toolbox. What we found is that while being user friendly and quick to setup, OASIS found lower minima than both ga and fmincon for both keane's bump and shuberts function. OASIS is a powerful tool that can be used for black-box optimization, while ga and fmincon ran faster and retained a higher degree of control over individual parameters. Each tool has their pros and cons, which must be considered when deciding which to use.

Appendices

This section includes a brief instruction on how to setup OASIS model. For simplicity, we consider Keane's Bump function as reference. First, introduce design variables in MODEL → INPUTS (Figure 1). Then select OUTPUTS to register objective function and its constraints. Select Math Constraint or Math Objective (Figure 2) to write constraints or objective function, respectively. Once either selection is made, type in objective function (Figure 3) or constraint(s) (Figure 4), if they exist. Note that by default, objective functions start with letter 'f' and constraints start with letter 'c'. Once problem's knowns are defined, select RUN → OPTIMIZER (Figure 5), to set optimization criteria and select Optimize. By default, Run Count is set on 1 and no additional information is necessary.

Once optimization start, navigate to VISUALIZE → RESULTS SHEET (Figure 6). Here we find some important information. For instance, under constraint column(s), if the value for each iteration is negative it indicated the constraint is satisfied – otherwise not satisfied. You can also sort different columns. GRAPHS (Figure 7) has a live optimization status under Run Status. Under Parallel Coordinate Plot, vertical axes is an input or objective. Colors indicate quality of values: Red, yellow and green lines indicate bad, average and good values, respectively.

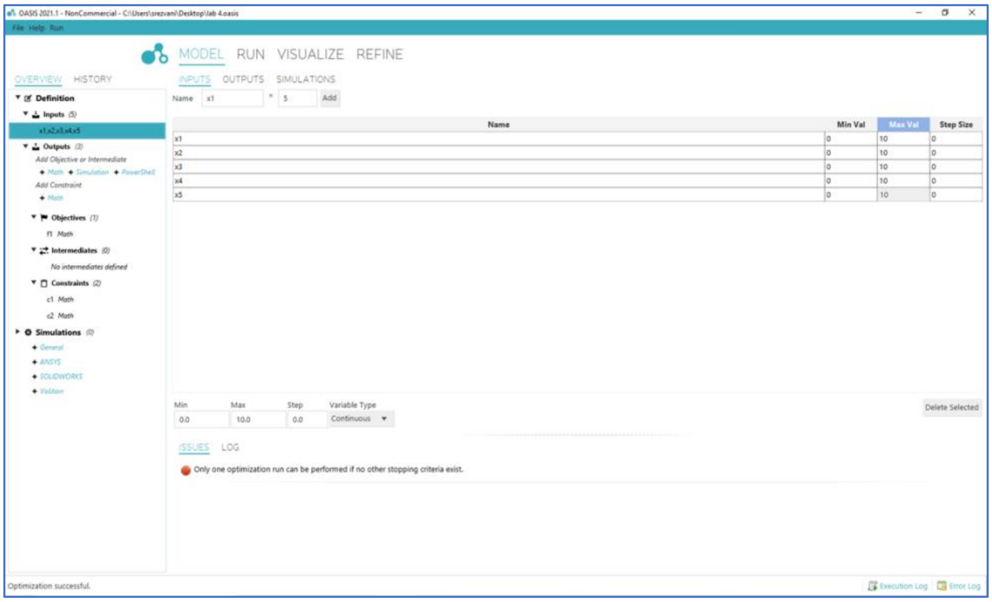

Figure 1 - Inputs

Figure 2 – Outputs

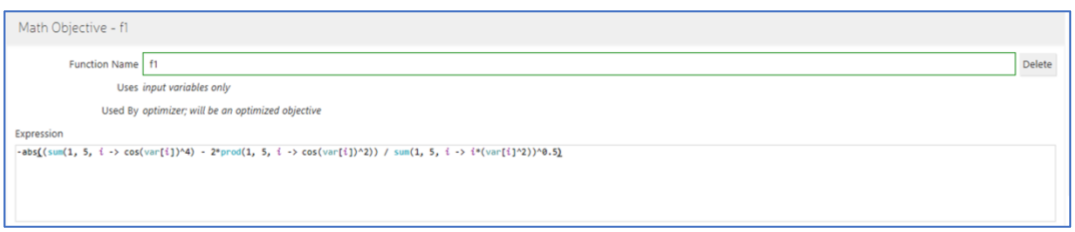

Figure 3 – Math Objectives

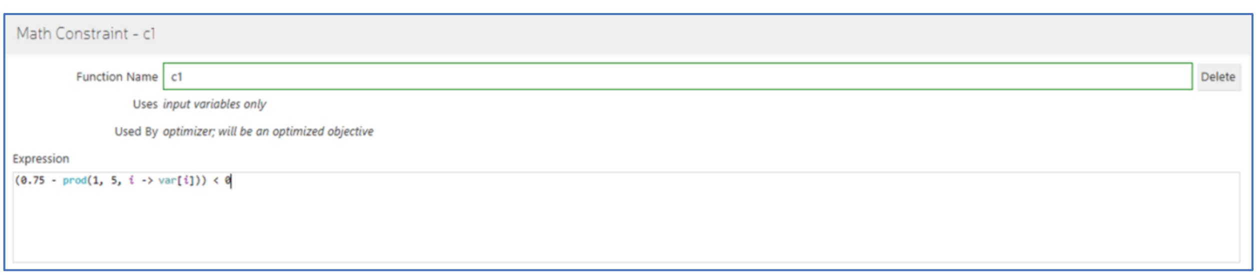

Figure 4 – Math Constraints

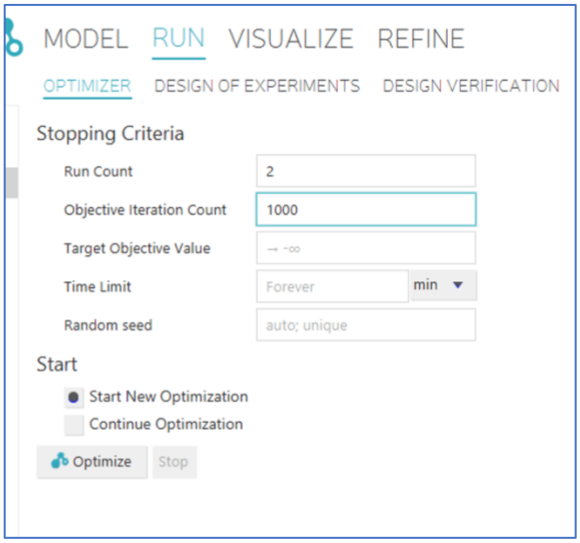

Figure 5 – Run

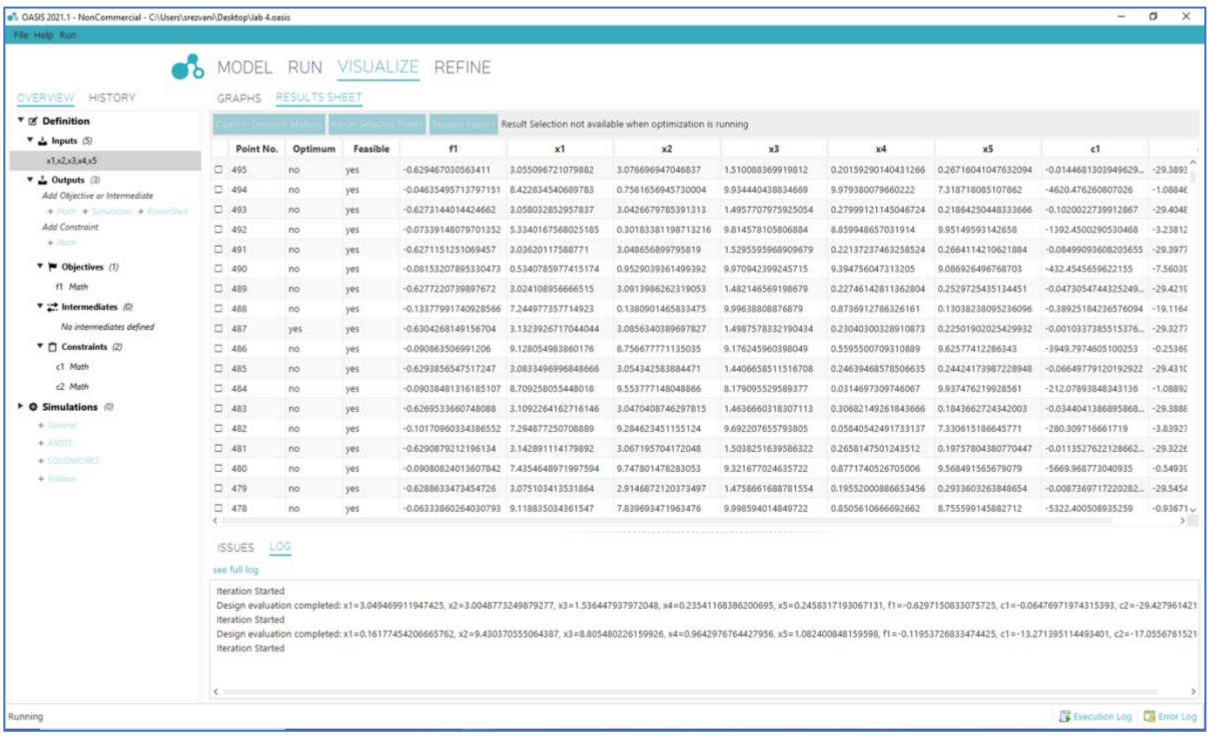

Figure 6 – Results Sheet

Figure 7 - Graphs

Code

Main

```
clc
clear all
close
global problem_number
problem_number = 4;
%1 = fmincon Keane
%2 = fmincon Shubert
%3 = Genetic Keane
%4 = Genetic Shubert

fminOptions = optimoptions('fmincon');
fminOptions.MaxFunctionEvaluations = 1000;
% gaOptions = optimoptions('ga');
% gaOptions.MaxGenerations = 1000;


if problem_number == 1          % fmincon Keane
    %fmincon Keane
    for i=1:10                  % 10 runs
        fprintf('FOR RUN %d WE HAVE:\n',i)
        x0 = rand([1 5]);
        [minPos,minVal,exitflag,output] =
fmincon(@keaneFunc,x0,[],[],[],[],[0,0,0,0,0],[10,10,10,10,10],@lab4NLCon,
fminOptions);
        fprintf('Minimum value found is %d \nfound at location x =
%d,%d,%d,%d,%d\n',minVal,minPos(1:5))
        fprintf('Number of iterations = %d \n',output.iterations)
        fprintf('Number of function evaluations = %d \n',output.funcCount)

fprintf('============================================================\n')
    end

elseif problem_number == 2      % fmincon Shubert
```

```
    %fmincon Shubert
    for i=1:10                 % 10 runs
        fprintf('FOR RUN %d WE HAVE:\n',i)
        x0 = rand([1,10]);
        lb = -5.12;
        ub = 5.12;
        [minPos,minVal,exitflag,output] =
fmincon(@shubertFunc,x0,[],[],[],[],[lb,lb,lb,lb,lb,lb,lb,lb,lb,lb],[ub,ub
,ub,ub,ub,ub,ub,ub,ub,ub],@lab4NLCon,fminOptions);
        fprintf('Minimum value found is %d \nfound at location x =
%d,%d,%d,%d,%d,%d,%d,%d,%d,%d \n',minVal,minPos(1:10))
        fprintf('Number of iterations = %d \n',output.iterations)
        fprintf('Number of function evaluations = %d \n',output.funcCount)

fprintf('============================================================\n')
    end

elseif problem_number == 3       % Genetic Keane
    %Genetic Algorithm Keane
    for i=1:10                 % 10 runs
        fprintf('FOR RUN %d WE HAVE:\n',i)
        lb = -5.12;
        ub = 5.12;
        [minPos,minVal,exitflag,output] =
ga(@keaneFunc,5,[],[],[],[],[lb,lb,lb,lb,lb],[ub,ub,ub,ub,ub],@lab4NLCon);
        fprintf('Minimum value found is %d \nfound at location x =
%d,%d,%d,%d,%d\n',minVal,minPos(1:5))
        fprintf('Number of Generations = %d \n',output.generations)
        fprintf('Number of function evaluations = %d \n',output.funccount)

fprintf('============================================================\n')
    end

elseif problem_number == 4        % Genetic Shubert
    %Genetic Algorithm Shubert
    for i=1:10                 % 10 runs
```

```
        fprintf('FOR RUN %d WE HAVE:\n',i)
        lb = -5.12;
        ub = 5.12;
        [minPos,minVal,exitflag,output] =
ga(@shubertFunc,10,[],[],[],[],[lb,lb,lb,lb,lb,lb,lb,lb,lb,lb],[ub,ub,ub,u
b,ub,ub,ub,ub,ub,ub],@lab4NLCon);
        fprintf('Minimum value found is %d \nfound at location x =
%d,%d,%d,%d,%d,%d,%d,%d,%d,%d \n',minVal,minPos(1:10))
        fprintf('Number of Generations = %d \n',output.generations)
        fprintf('Number of function evaluations = %d \n',output.funccount)

fprintf('============================================================\n')
    end
end
```

keaneFunc

```
function [output] = keaneFunc(x)

sum1 = 0;
product1 = 1;
sum2 = 0;

for i = 1:5
sum1 = sum1 + cos(x(i))^4;
product1 = product1*cos(x(i))^2;
sum2 = sum2 + (i*x(i)^2);
end


output = -((sum1-2*product1)/(sum2)^0.5);

end
```

shubertFunc

```
function [output] = shubertFunc(x)
```

```
sum1 = 0;
product1 = 1;

for i = 1:10

    for j = 1:5
        sum1 = sum1 + j*cos((j+1)*x(i)+j);
    end
    product1 = product1*sum1;
end

output = product1;

end
```

lab4NLCon

```
function [c,ceq] = lab4NLCon(x)
global problem_number

prod = 1;
sum = 0;
if problem_number == 1 || problem_number == 3
    for k = 1:5
        prod = prod*x(k);
        sum = sum + x(k);
    end

    c = [0.75-prod;sum-7.5*5];
    ceq = [];
elseif problem_number == 2 || problem_number == 4
    c = [];
    ceq = [];
else
    c = [];
    ceq = [];
end
end
```