ENGINEERING OPTIMIZATION PROJECT

MECH 580 – SUMMER TERM

Date

10TH Aug 2022.

Submitted by

Ogunfowora Oluwaseyi

VOO994397

TABLE OF CONTENTS

1.0 SUMMARY.

"Optimization is a process of finding the conditions that give the maximum or minimum value of a function. (Sohouli, n.d., slide 9) The purpose of this project is to help students gain a better understanding of the concepts of optimum design, develop optimization algorithms codes using the MATLAB programming language and apply optimization techniques that were taught in the MECH 580 Engineering Optimization course to solving real-world optimization problems. This project was organized as a requirement towards the completion of the MECH 580 Engineering Optimization course work.

The project **scope** covers software programming with the MATLAB-programming language of a Gradient-based constrained optimization algorithm, a Global optimization Algorithm and hyperparameter selection and tuning to get optimum design values for the given problems. **Out of the scope** of this project is the five-step optimization problem formulation procedure. The objective function, constraint function and design variables were given.

This report describes the project in great detail, it outlines the project aims and objectives, project execution procedures and results discussions. It is divided into 6 main sections: The Summary, Introduction, Project Objectives, Procedures for the two sections of the project, Conclusion and References. The aim of this project is to develop robust and efficient high performance, multi-parameter optimization algorithms to solve the design of a pressure vessel problem using a gradient-based optimization algorithm and the design of a welded beam using the particle swarm optimization algorithm with the objective of reducing production costs. To use the developed programs, the F_main file for each algorithm is the main file that needs to be executed or run; it calls all underlying subroutines needed to execute the algorithms.

In this project, the gradient-based optimization algorithm selected and used was the Augmented Lagrangian Multiplier (ALM) algorithm. For clarity of this report, sections 3 and 4 focus on each developed program, the Augmented Lagrangian Multiplier (ALM) and Particle Swarm Optimization (PSO) respectively. In these sections, the justification for the chosen algorithms, the development, the validation and the testing procedures were explained in detail, how to use the programs and when to take caution were also discussed, the results and graphical representations for each design problem were highlighted and discussed as well. In the body of the report, the references to the codes/subroutines/programs/stored values are highlighted in blue for easy identification, these references point to the folders and MATLAB M-files where the codes are written. There are also references to MATLAB MAT-files, excel and pdf files that were used to store relevant variables in the programs.

2.0 INTRODUCTION.

2.1 Project Description.

The Engineering Optimization course project is in two (2) parts. It involves the application of one of the Gradient-based optimization algorithms discussed in class and a global optimization algorithm; the Particle Swarm Optimization Algorithm (PSO), to optimize the design of a cylindrical pressure vessel and a welded beam respectively. This project was carried out using the MATLAB programming language to programme the above mentioned algorithms with the objective of minimizing the total production cost of the design items.

The Engineering Optimization design process begins with the problem formulation. The optimum design problem formulation can be defined as the translation of a descriptive statement of the design problem to a mathematical statement that can be optimized (Arora, 2004). It should be noted that in this project, the proper definition and formulation of the design problem was already conducted and the requirement for project completion is to write the codes for the chosen optimization algorithms in MATLAB and use them to find the optimum design variables for the given problems. The outputs of the problem formulation stage includes:

1. The design variables.
2. The Objective Function.
3. The constraint functions.

The Design Variables: These are the entities that describe a system. These variables are referred to as optimization variables and regarded as free because any value can be assigned to them; they are also required to be independent of one another.

The Objective Function: This is a function of the design variables which needs to be maximized or minimized depending on the problem requirements. For this project, the cost function(s), functions to be minimized were given.

The Constraint Function(s): In engineering applications or most real systems, the systems have to be designed within given resource limits and performance requirements; these factors impose constraints on the design variables and are translated to mathematical constraint functions that have to be satisfied for a design to be accepted. These constraints also depend on the design variables and they are simply the restrictions placed on the system design.

In this report, the design variables, objective function and constraint functions for each of the optimization problems are presented and discussed in sections 3 and 4 respectively, they are divided into numbered and headed sections and the different main ideas are presented in a logical order.

2.2 Project Objectives.

The purpose of this project is to consolidate the concept of engineering design and optimization techniques discussed in the MECH 580 course through a practical approach. It focuses on providing a platform for the implementation of the engineering design trial and error procedure to estimate a system design and analyze it to see if it performs according to given specifications and the use of MATLAB programming language to apply optimization algorithms to select optimum design variables.The **goal** is to code a gradient-based and a global optimization algorithm for constrained optimization problems.

The objectives of the project are:

1. Complete a high-performance optimum design of a pressure vessel and documentation.
2. Complete a high-performance optimum design of a welded beam and documentation.
3. Demonstrate the quality of the results of the project quantitatively and graphically.

Some other requirements or project specifications include:

1. Transcribing the design problem into the standard form.
2. Trying different initial designs and discussing their results.
3. Verify the solution(s) graphically and trace the history of the iterative process graphically.

3.0 DESIGN OF PRESSURE VESSEL.

3.1 The Pressure Vessel Design Problem Formulation.

3.1.1 The Design Variables.

There are four(4) design variables:

1. The thickness of the shell (T_s).
2. The thickness of the head (T_h).
3. The inner radius (R).
4. The length of the cylindrical section of the vessel, not including the head (L).

3.1.2 The Objective Function.

Minimize $f(R, L, T_s, T_h) = 0.6224T_sRL + 1.7781T_hR^2 + 3.1661(T_s)^2L + 19.84(T_s)^2R$

3.1.3 The Constraint Functions.

Subject to: $0.0193R \leq T_s$

$0.00954R \leq T_h$

$\pi R^2L + 4/3\pi R^3 \geq 1296000$

$L \leq 240$

$0.1 \leq T_s \leq 99$

$0.1 \leq T_h \leq 99$

$10 \leq R \leq 200$

$10 \leq L \leq 200$

3.2 Problem in Standard form.

In a typical optimization problem, the standard form is defined as minimization of a function with **"≤"** type constraints.

3.2.1 The Standard Objective Function.

For the pressure vessel design problem, the given function is the same as the standard form function.

Minimize $f(x_1, x_2, x_3, x_4) = 0.6224x_3x_1x_2 + 1.7781X4x_1^2 + 3.1661(x_3)^2x_2 + 19.84(x_3)^2x_1$

3.2.3 The Standard Constraint Functions.

The standard constraint functions, however, need to be written as "≤" type constraints.

Subject to: $0.0193x_1 - x_3 \leq 0$

$0.00954x_1 - x_4 \leq 0$

$1296000 - \pi x_1^2 x_2 - 4/3\pi x_1^3 \leq 0$

$x_2 - 240 \leq 0$

$x_3 - 99 \leq 0$

$0.1 - x_3 \leq 0$

$x_4 - 99 \leq 0$

$0.1 - x_4 \leq 0$

$x_1 - 200 \leq 0$

$10 - x_1 \leq 0$

$x_2 - 200 \leq 0$

$10 - x_2 \leq 0$

3.3 Procedure and Code References.

The procedure for solving the pressure vessel design problem in terms of programming the MATLAB codes involves the development, testing and graphical analysis of the subroutine algorithms that make up the Gradient-based algorithm. For this project, the gradient-based constrained optimization algorithm I used was the Augmented Lagrangian Multiplier algorithm (ALM).

3.3.1 The Augmented Lagrangian Multiplier Algorithm And Reason For Selection.

The Augmented Lagrangian Multiplier Algorithm is a gradient-based solution for constrained optimization problems using unconstrained optimization methods and these methods are referred to as indirect methods. Generally, there are two (2) main methods of solving constrained optimization problems: The direct and Indirect Methods.

In Indirect Methods, The basic idea is to construct a composite function using the cost and constraint functions, it also contains certain parameters referred to as the penalty parameters that are responsible for penalizing the composite function for violation of constraints. They are termed "Transformed methods" because these methods solve the constrained optimization problem by transforming them into one/more unconstrained problems. They are generally characterized under SUMT (Sequential Unconstrained Minimization Techniques) and broadly classified into; penalty and barrier functions.

The Sequential Unconstrained Minimization Techniques have certain weaknesses; the penalty and barrier functions tend to be ill-behaved near the boundary of the feasible set where the optimum lies when the penalty parameters go to infinity. (Arora, 2004) To cater for this weakness, the Augmented Lagrangian Multiplier Method was introduced, the basic idea of penalty/barrier methods is that for convergence to be guaranteed, the penalty parameters goes to infinity which in turn causes many irregularities so, instead of trying to make the penalty parameters go to infinity, the Augmented Lagrangian Multiplier Method constructs the composite function in such a way that the penalty parameters are kept finite while minimizing the transformation function and that is why I have selected this optimization method the solve this problem.

Also, the Augmented Lagrangian Multiplier Method (ALM) is a robust method because it has been

proven to converge starting from any arbitrary point. The transformed or composite unconstrained function is solved using a Quasi-Newton Method, Davidon-Fletcher-Powell (DFP) method. This method was chosen because it is one of the most powerful methods for minimization of a function due to its fast rate of convergence compared to the ordinary steepest-descent method, its robustness and computational efficiency. The steepest-descent method has a poor rate of convergence because only first-order information is used, The Newton methods were introduced to correct these flaws; they use second-order derivatives of the function and they were able to achieve good convergence properties but they are computationally inefficient because they require calculation of n(n+1)/2 second-order derivatives to generate the Hessian matrix (where n is the number of design variables). Newton's method also runs into difficulties if the Hessian of the function is singular at any iteration that is it is not continuous so the quasi-newton methods were developed to overcome these drawbacks of Newton's method by generating an approximation for the Hessian matrix or its inverse at each iteration and only the first derivatives of the function are used to generate these approximations which makes them robust, they possess the desirable features of both the steepest-descent and the Newton's methods (Sohouli, n.d., slide 8)

Basically, in optimum design applications because it is a trial and error process and sometimes might contain many hyperparameters to be tuned, designers focus their efforts on algorithms that are robust, guaranteed to converge, applicable and reliable because in practical applications; the failure of the algorithms can cause disastrous effect, it can make the codes very hard to debug, difficult to implement and in return kill the morale of the designers. Sometimes in the case of this project/course it might even require individuals to be well grounded in optimization techniques before they can apply them to solve problems which cannot be learnt in three (3) months of taking a course. It also makes numerical implementation of algorithms in general purpose design optimization softwares such as MATLAB less reliable so a robust and generally functional algorithm such as DFP is better suited and that is why I have chosen it as the minimization algorithm. The ALM algorithm is composed of 3 subroutine programs and one main program to call the other subroutine functions.

3.3.2 The Subroutines, Testing and Validation.

3.3.2.1 The Golden Search Algorithm and Reason for Selection.

Numerical methods for solving unconstrained problems have gained considerable importance over time and substantial effort has been expended in developing efficient algorithms for these problems. These methods were all developed on the general idea of initiating an estimate for the optimum solution and this estimate is improved through an iterative process till the optimal conditions are satisfied or stopping criteria is reached. It involves updating the design in the direction of descent. The change in design is composed of two (2) subproblems: Direction of descent finding and Step-size determination subproblems. The Golden search method is a one-directional search algorithm, it is a method that searches for the minimum of a given function within a chosen interval, it basically determines the interval in which the minimum of a function lies and it is used to solve the step-size determination subproblem in the DFP subroutine. The Golden Search method is an improvement over the alternate equal interval line search and one of the better methods in the class of interval reducing methods and that is why I chose it for this project. It is more computationally efficient. The number of function evaluations is a measure of efficiency of an algorithm and compared to other line 1-D/Line search algorithms like the alternate equal interval line search, the golden search is more computationally efficient because for the same problem/function, lesser function evaluations are needed to obtain the same solution compared to other line search methods which makes the golden search a better method.

The golden search program was developed using the MATLAB programming language.

See MECH 580 project folder - GoldenSearch_1D Algorithms folder - GoldSection_1Var for golden search subroutine code.

3.3.2.2 The Golden Search Algorithm Testing.

In software development projects, it is important to test each subroutine or developed unit of a program before using it in another function. It makes the program easy to debug and integrate with other programs. To test the developed golden search Algorithm code, I used the already solved example 8.3 in The Optimum Design Textbook (Arora, 2004). This problem has already been solved in the textbook so it can be considered a reliable source for cross-validation of the result of my

algorithm.

See MECH 580 project folder - GoldenSearch_1D Algorithms folder - Test_Example for function code.

See MECH 580 project folder - GoldenSearch_1D Algorithms folder - GoldenSection_Test_Example for code to call the golden section algorithm function.

3.3.2.3 The Golden Search Algorithm Validation and Results.

Here is the result: 1.3859073551243202, 0.45482285502908049.

See MECH 580 project folder - GoldenSearch_1D Algorithms folder - GoldenSection_Test_Example_Result for results.

The above results are approximately the same as the result in example 8.3 in The Optimum Design Textbook which are:1.386511, 0.454823 where the values represent where the minimum of the function is obtained and the values of the function at the given point respectively. Below is a graphical representation of the function showing the minimum point that was found which is exactly the same as the values in the textbook.

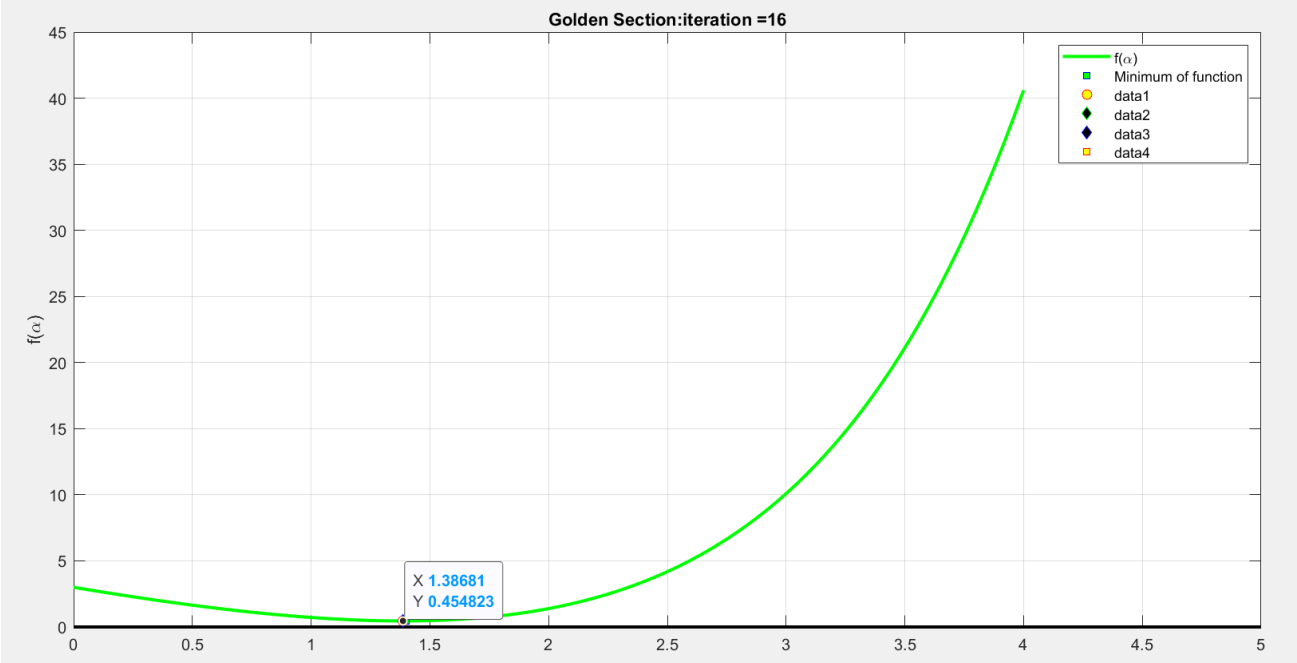

Figure 1

Golden Search algorithm: Test example function plot.

3.3.3.1 The DFP Algorithm.

As earlier mentioned, DFP is a quasi-newton method, it is a search technique or direct method of optimization which uses an iterative process representing an organized search through the design space to find points that represent the local minimum of the cost function.

The DFP algorithm was developed using the MATLAB programming language.

See MECH 580 project folder - DFP_Algorithms folder - DFP for subroutine.

3.3.3.2 The DFP Algorithm Testing.

To test the developed DFP Algorithm code, I used the already solved Assignment 2: question 1 for this course. This problem has already been solved so it can be considered a reliable source for cross-validation of the result of my algorithm.

See MECH 580 project folder - DFP_Algorithms folder - Test_ExampleDFP for function code.

See MECH 580 project folder - DFP_Algorithms folder - DFP_Test_Example for code to call the DFP algorithm function.

3.3.3.3 The DFP Algorithm Validation and Results.

Here is the result: -0.428838902926593, 0.142589037909284, 1.71428585787757

See MECH 580 project folder - DFP_Algorithms folder - DFP_Test_Example_Result for results.

The above results are approximately the same as the result of the already solved Assignment 2: question 1 for this course which are: -0.4286, 0.1429, 1.7143 where the values represent design variable points (x1 & x2) where the minimum of the function is obtained and the value of the function at the given point respectively. Below is a graphical representation of the function showing the iterative progress of the DFP function from the initial point (1,1) to the minimum point that was found.

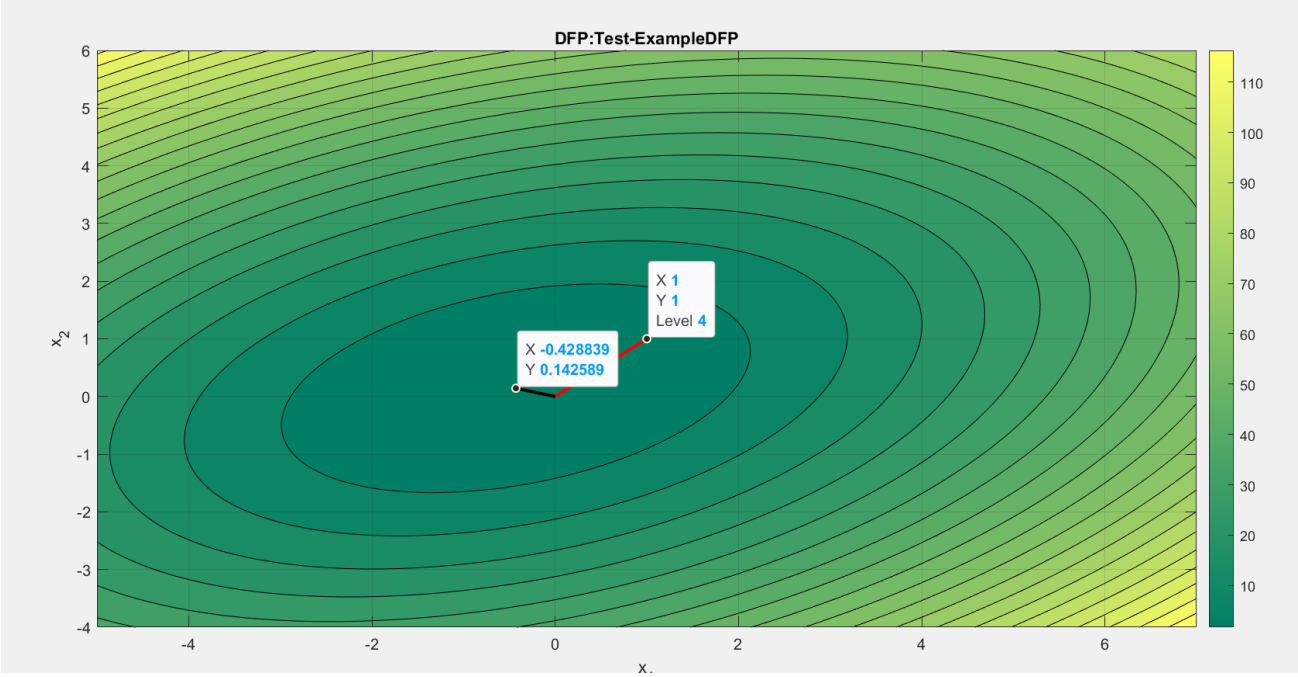

Figure 2

DFP algorithm: Test Example function contour plot and change in design plot.

3.3.4.1 The ALM Algorithm.

The ALM algorithm was developed using the MATLAB programming language.

See MECH 580 project folder - ALM_Algorithms folder - ALM for subroutines.

3.3.4.2 The ALM Algorithm Testing.

To test the developed ALM Algorithm code, I used the already solved example 7.1 in the Numerical Techniques for Applied Optimization with MATLAB programming textbook (Venkataraman, 2008). This problem has already been solved so it can be considered a reliable source for cross-validation of the result of my algorithm.

See MECH 580 project folder - ALM_Algorithms folder - Ofun_Test_Example, Hfun_Test_Example and Gfun_Test_Example for function code and equality and inequality constraints codes respectively.

See MECH 580 project folder - ALM_Algorithms folder - ALM_Test_Example for code to call the ALM algorithm function.

3.3.4.3 The ALM Algorithm Validation and Results.

Here is the result: 0.99940470233208223 1.0002023256159265.

See MECH 580 project folder - ALM_Algorithms folder - ALM_Test_Example_XResult for results.

The above results are approximately the same as the result of the example 7.1 in the Numerical Techniques for Applied Optimization with MATLAB programming textbook where the values represent design variable points (x1 & x2) where the minimum of the function is obtained. Below is a graphical representation of the function showing a combination of the cost function contour plots, the constraint functions and feasible regions, the DFP iterative steps towards the minimum starting from point (3,2). For more results of the constraint values after ALM algorithm optimization, *See MECH 580 project folder - ALM_Algorithms folder - ALM_Test_Example_RgResult for results and ALM_Test_Example_RhResults.*

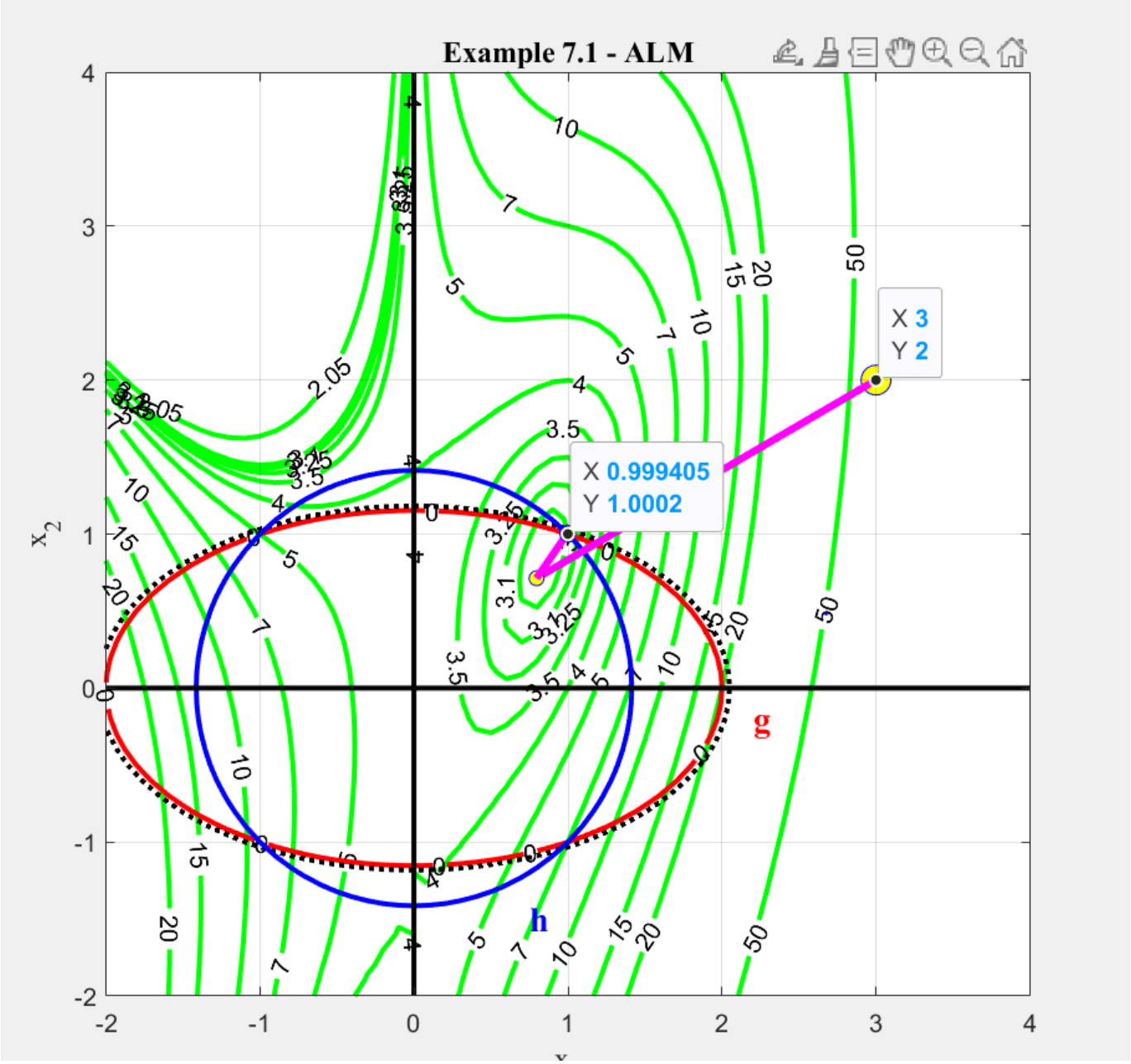

Figure 3

ALM algorithm: Test Example objective function contour plot, constraint function plots and change in design direction.

3.4 The Main Algorithm.

The main ALM algorithm used to solve the pressure vessel design problem is a combination of all the above-mentioned subroutines and additional programs to initialize design variables from varying randomly chosen starting points. The main Algorithm is in the MECH580_project - Engr_Optimization_Project_1 folder.

These files have in them comments on the definition of the functions, what the functions do, their input arguments, required subroutines and outputs. Each file has a thorough guide to gain an understanding on how to use it and when to take caution.

3.4.1 Implementation Notes.

1. **Using Different Initial Designs**: One of the major drawbacks of search methods is the problem of getting stuck in a local minimum. It is possible for the algorithms to get stuck in a local minimum while searching through the variable space which makes it impossible to get to the point where the function has the least value. To cater for this, it is good practice to start the algorithm at different initial designs with the hope that one or more good starting points will help to get to the global minimum or as close to it as possible.

   Also, from observation, the ALM algorithm converged at some local minima where all the constraints were not satisfied but the change in design was constant so no more improvements were made on the design points; to deal with these issues, it is recommended to start at random initial values; this was adopted in this project.
2. **Random Initial Values**: In order to accurately capture the entire feasible design space and move randomly in it, the initial values were chosen at random using the MATLAB rand function while ensuring that the values are within the explicit constraint limits for each design variable.

   ```
   % init_x1 = Xlow(:,1) + (Xhigh(:,1)-Xlow(:,1)).*rand(1);
   % init_x2 = Xlow(:,2) + (Xhigh(:,2)-Xlow(:,2)).*rand(1);
   % init_x3 = Xlow(:,3) + (Xhigh(:,3)-Xlow(:,3)).*rand(1);
   % init_x4 = Xlow(:,4) + (Xhigh(:,4)-Xlow(:,4)).*rand(1);
   % initialdesign = [init_x1, init_x2, init_x3, init_x4];
   ```

See the *MECH580_project - Engr_Optimization_Project_1 folder - ALM_Main* file for implementation of this. Kindly note that the ALM_Main file is also a function that calls the ALM primary algorithm; the ALM_Main file is called by the F_Main file.

3. **How to Use (The F_Main file)**: To use the developed program, the F_main file is the main file to run/execute. It calls the ALM_Main function 200 times while the ALM_Main file passes different random initial values to the ALM primary algorithm. This makes it possible to have **200 random initial starting designs** that move through the variable space to find a good minimum. So to run the program, just run the F_Main file.
    a. **Note**: For this project, the F_Main file has been used to get 200 random initial starting designs, the results have been analyzed and the initial design points that result in the lowest function value and that satisfies all constraints have been chosen and used to replace the random value generators. Here is the chosen initial design: initialdesign = [186.3794818, 120.2171695, 1.779612602, 12.05301158]
    b. The random value generators were not deleted, just commented out, to use them for verification purposes, uncomment them and comment out the already chosen starting point for this project.
    c. Result comparison: If the random value generators are activated, the collated results will not be the same as mine and cannot be directly compared because the values are generated randomly so it wouldn't be the same initial design values as mine.
    d. Storage: For better analysis, the F_main file is also equipped with functions to store the resulting, initial designs for each of the 200 iterations, stores the updated xvalues, the constraint values using the updated xvalues, the composite function values using the updated xvalues and the Rg parameter values.

3.4.2 Results and Discussions.

From the F_Main files, necessary parameters and values for every iteration using different initial values were collected and stored in MATLAB mat files. These results can be found in the *MECH580_project - Engr_Optimization_Project_1 folder - Results folder.* The results folder contains the final results from using random generators to find the best initial starting design. Below

is the definition of all the results present in the Final_Results1 MAT files.

Final-xstore: This file contains the learnt parameter values from various randomly chosen initial designs.

Final_initialdesign: Contains the randomly chosen initial values.

Final_rgstore: Contains the final rg parameters updated at every ALM iteration.

Final_gAstore: Contains the constraint values using learned parameters.

Final_fVstore: It contains the values of the composite function after every iteration of randomly chosen initial values.

1. **Result Tables:** For better visualization of the stored values, an excel sheet was drawn to collate all these results. See *MECH580_project - Engr_Optimization_Project_1 folder - Results folder - Analysis_table* excel file to see the table of results showing the 200 random initial parameter values, optimized parameter values after ALM, final constraint values at ALM convergence and function values at ALM convergence.
2. **Reduced Result Table**: The large result table was sorted for optimized parameter values that resulted in the values of all constraints being satisfied. The reduced result table contains stored initial design values, function values, optimized parameter values and constraint values of the few select values. *See MECH580_project - Engr_Optimization_Project_1 folder - Results folder - Data_SatisfiedConstraints pdf file for table.*

   Also see table(s) below. The row highlighted in dark teal color is the corresponding initial design values, optimized parameter values, constraint values and function values of the selected initial design for this project.
3. **Final Results**.
   a. Initial Design [X1, X2, X3, X4] = [186.3794818, 120.2171695, 1.779612602, 12.05301158]
   b. Optimized Parameter Values = [49.7935860289998, 99.9913823007038, 0.961732643764591, 15.3367501818286]
   c. Constraint Values After Optimization = [-0.000716433404894024, -14.8617193711120, -0.560178874060512, -140.008617699296, -98.0382673562354, -0.861732643764591 -83.6632498181714,

-15.2367501818286, -150.20641397100, -39.7935860289998

-100.008617699296, -89.9913823007038]

d. Function Value = [71800.8127810083]

	INITIAL PARAMETER VALUES				OPTIMIZED PARAMETER VALUES			
	x1	x2	x3	x4	x1	x2	x3	x4
33	59.37537654	162.0130112	42.76682754	90.16304709	43.49685145	160.0504253	2.663794058	88.49404876
60	66.36841591	70.56787737	42.05009254	50.32718435	54.11383291	68.72608623	8.794991516	48.36065603
63	55.08388016	97.18127735	95.34945654	54.17908558	50.24770053	96.39341324	88.70273945	54.04647468
129	113.3383275	146.9988579	51.77478574	98.37738733	44.90690993	144.689887	7.015099891	69.57065398
131	86.87019921	95.19085329	36.27921989	75.61060898	50.60794531	93.59434212	1.687635316	60.79163551
159	186.3794818	120.2171695	1.779612602	12.05301158	49.79358603	99.9913823	0.961732644	15.33675018
162	78.85817783	19.41119002	48.51847193	19.13927817	63.73319328	16.58641213	2.924188879	15.9632643

CONSTRAINT VALUES AFTER OPTIMIZATION												FUNCTION VALUES AFTER OPTIMIZATION
g1	g2	g3	g4	g5	g6	g7	g8	g9	g10	g11	g12	FVALUES
-1.824304825	-88.0790888	-27.12746539	-79.94957473	-96.33620594	-2.563794058	-10.50595124	-88.39404876	-156.5031486	-33.49685145	-39.94957473	-150.0504253	318966.1468
-7.750594541	-47.84441006	-12.59397747	-171.2739138	-90.20500848	-8.694991516	-50.63934397	-48.26065603	-145.8861671	-44.11383291	-131.2739138	-58.72608623	372041.0224
-87.73295883	-53.56711162	-10.89570958	-143.6065868	-10.29726055	-88.60273945	-44.95352532	-53.94647468	-149.7522995	-40.24770053	-103.6065868	-86.39341324	10755237.52
-6.14839653	-69.14224206	-12.45450075	-95.31011299	-91.98490011	-6.915099891	-29.42934602	-69.47065398	-155.0930901	-34.90690993	-55.31011299	-134.689887	344223.35
-0.710901972	-60.30883572	-3.763731208	-146.4056579	-97.31236468	-1.587635316	-38.20836449	-60.69163551	-149.3920547	-40.60794531	-106.4056579	-83.59434212	285524.3967
-0.000716433	-14.86171937	-0.560178874	-140.0086177	-98.03826736	-0.861732644	-83.66324982	-15.23675018	-150.206414	-39.79358603	-100.0086177	-89.9913823	71800.81278
-1.694138248	-15.35524964	-47.86439233	-223.4135879	-96.07581112	-2.824188879	-83.0367357	-15.8632643	-136.2668067	-53.73319328	-183.4135879	-6.586412126	128479.9662

Table 1

ALM Algorithm: Result objective function value, constraint function values and initial and optimized parameter values table

4.0 DESIGN OF WELDED BEAM.

4.1 The Welded Beam Design Problem Formulation.

4.1.1 The Design Variables.

There are four(4) design variables:

1. The Height of weld (h).
2. The Length of weld (L).
3. The Height of the beam (t).
4. The Width of the beam (b).

4.1.2 The Objective Function.

Minimize $f(\mathbf{h,L,t,b}) = 1.10471h^2L + 0.04811tb\,(14.0 + L)$

4.1.3 The Constraint Functions.

Subject to: $\tau \leq \tau_{max}$

$\sigma \leq \sigma_{max}$

$h \leq b$

$0.10471h^2 + 0.04811\,t\,b\,(14.0 + L) \leq 5$

$0.125 \leq h$

$\delta \leq \delta_{max}$

$P \leq P_c$

4.2 Problem in Standard form.

In a typical optimization problem, the standard form is defined as minimization of a function with **"≤"** type constraints.

4.2.1 The Standard Objective Function.

For the beam design problem, the given function is the same as the standard form function.

Minimize $f(\mathbf{h,L,t,b}) = 1.10471x_1^2x_2 + 0.04811x_3x_4\,(14.0 + x_2)$

4.2.3 The Standard Constraint Functions.

The standard constraint functions, however, need to be written as **"≤"** type constraints.

Subject to: $\tau - \tau_{max} \leq 0$

$\sigma - \sigma_{max} \leq 0$

$x_1 - x_4 \leq 0$

$0.10471x_1^2 + 0.04811\,x_3\,x_4\,(14.0 + x_2) - 5 \leq 0$

$0.125 - x_1 \leq 0$

$\delta - \delta_{max} \leq 0$

$P - P_c \leq 0$

4.3 Particle Swarm Optimization: Algorithm.

Particle swarm optimization (PSO) is inspired by social and cooperative behavior displayed by various species to fulfill their needs in the search space. The algorithm is guided by personal experience (Pbest), overall experience (Gbest) and the present movement of the particles to decide their next positions in the search space. For constrained optimization problems, just like indirect search methods, an equivalent unconstrained function is constructed using a penalty function for the

constraints and the PSO algorithm is applied. In literature, the stationary penalty parameters have proved to work better than the non-stationary penalty, so it was applied in this project. Also, the most commonly used hyperparameters for the PSO (not mathematically proven but a rule-of-thumb), these hyperparameters have worked well with the PSO algorithm and they were adopted in this project.

For improvement of the particle swarm optimization algorithm, an inertia weight and constriction factor was added to reduce the velocity value because the particles' velocities tend to build up so fast which might lead to skipping the optimum point. Finally, boundary conditions were introduced to the problem, this is to confine the search space and prevent the particle from going to a position that will result in invalid solution.

4.4 Procedure and Code References.

The procedure for solving the welded beam design problem is to develop the Particle Swarm Optimization algorithm with the MATLAB programming software. To ensure that the developed solution for this project is reliable; a code verification procedure was conducted which involved the use of a constrained three variable optimization problem as a use case to validate the results from the PSO algorithm. The results were compared with the results from a verified, general-purpose, built-in MATLAB software algorithm (fmincon) function. After verification, the developed algorithm was used to solve the beam design optimization problem, performance analysis of the algorithm based on the changing hyperparameters was carried out and graphical analysis was used to draw convincing inferences.

4.4.1 The PSO Code Verification.

4.4.1.1 The PSO Algorithm.

The PSO algorithm was developed using the MATLAB programming language. The same algorithm used for the test was used for the main project solution.

See MECH 580 project folder - PSO_Algorithms folder - PSO for the PSO primary algorithm code.

Three MATLAB live script files are needed to fully execute the PSO programme. In the first file, the transformed objective function is defined, the second file is the primary PSO algorithm that was

developed and the third file contains codes that call the PSO algorithm. The third file does more than just calling the PSO program and it is explained in detail in section 3.4.2.

4.4.1.2 The PSO Algorithm Testing.

To test the developed PSO Algorithm code, I used the example below. I solved the example with my PSO algorithm and the MATLAB **"fmincon"**function which is a general purpose design optimization algorithm so it is a reliable algorithm for cross-validation of my results.

Objective function = min [$10 (1 - x(1))^2 + 20 (2 - x(2))^2 + 30 (3 - x(3))^2$]

Subject to: $x(1) + x(2) + x(3) <= 5$

$x(1)^2 + 2 x(2) <= x(3)$

See MECH 580 project folder - PSO_Algorithms folder - Code Verification folder - fmincon_Ver folder - Verification_Func & nlcon for the implementation of the fmincon algorithm.

4.4.1.3 The PSO Algorithm Validation and Results.

Using the PSO algorithm, here are the values of X: 0.418344296902070, 1.46158365760199, 3.09901684296184

Function Value = 9.4725

See MECH 580 project folder - PSO_Algorithms folder - Code Verification folder - PSO_Ver - PSO_TestResult MAT file for Xvalues and function value.

The above results are approximately the same as the results from the fmincon MATLAB function which are:

X: 0.438277951180446 1.45654478489865 3.10517713228823

Function Value = 9.3941.

See MECH 580 project folder - PSO_Algorithms folder - Code Verification folder - fmincon_Ver folder - Xvalues to verify results.

Below is the plot showing the convergence of the PSO algorithm after 10 iterations to get the best cost value of 9.4725.

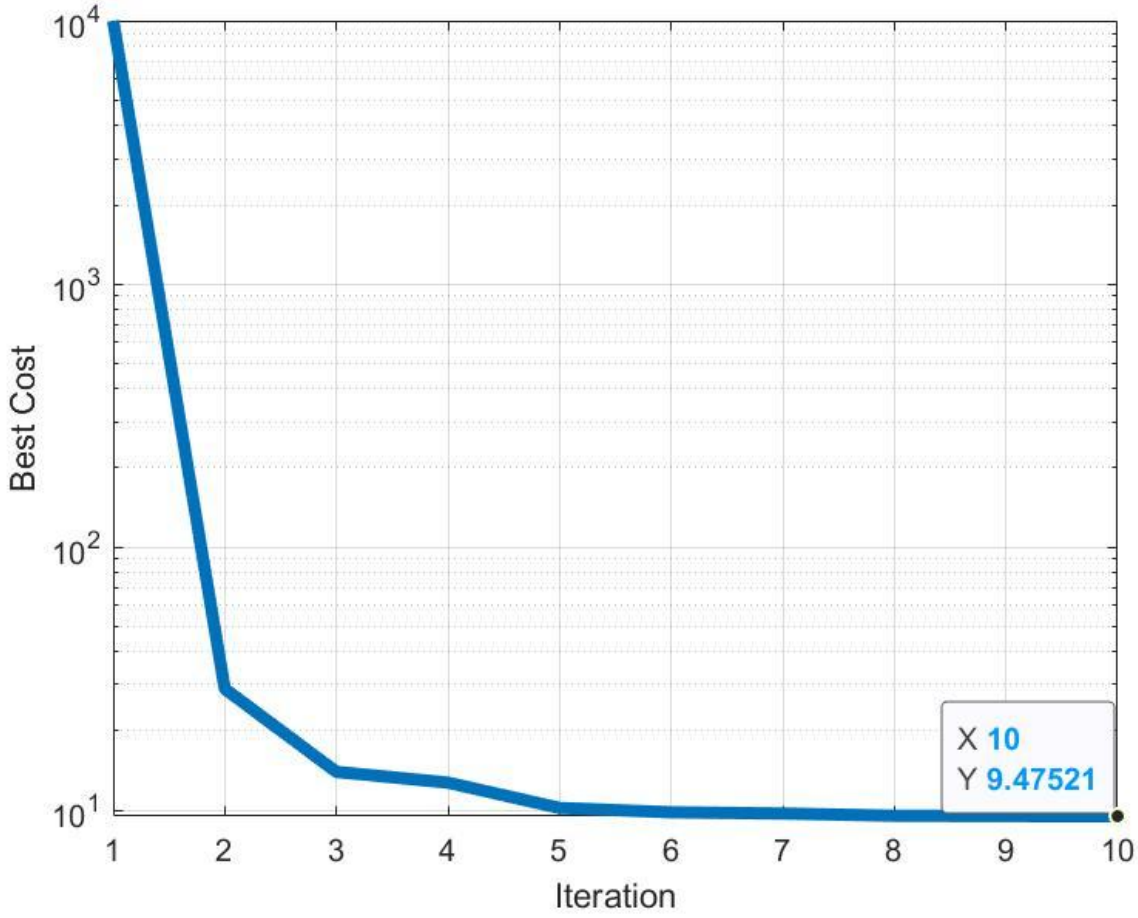

Figure 4

PSO Algorithm: Test example objective function values vs No of iterations plot

4.4.2 The PSO Main Algorithm.

The main PSO algorithm used to solve the welded beam design problem contains lines of codes that execute two main functions.

1. It is used to call/execute the Particle Swarm Optimization (PSO) primary function.
2. It is used to analyze the effect of population size on the performance of the algorithm.
3. It is used to select and store relevant parameters and information of iteration with the best cost.

4.4.2.1 How to use.

As mentioned above, the PSO main algorithm is used to call the PSO primary function so to execute the PSO function on a given problem, a MATLAB file containing the function is created, the main PSO function references both the function and primary PSO file to solve the problem. So to use the program, simply run the PSO main file.

See MECH 580 project folder - PSO_Algorithms folder - PSO_main for the PSO main mlx. file.

4.4.2.2 Analyzing the Effect of Population size.

To see the effect of different PSO population sizes on the behavior or performance of the algorithm while keeping the other hyperparameters fixed after adequately tuning them to get good algorithmic performance for the given optimization problem, the population sizes were varied using a for loop. The effect of the population sizes ranging from a of size 50 to a size 450 with an increment of 50 sizes at every iteration was used for testing; in order to be able to make reasonable and fairly accurate inferences from the statistical data, the average and medians of a large sample size was taken for every population size by creating an inner for loop of 100 iterations. For instance, at population size 50, the best costs (i.e. the lowest cost for a complete PSO optimization) over 100 iterations were collected and averaged and the medians were also collected.

The median is a preferred measure for central tendency for this kind of problem because it was observed that maybe due to the effect of factors like randomized starting points of the algorithm, a very few times , the PSO algorithm converges to very high cost values which is triggered by the penalty imposed on the cost function when constraints are violated. These unusual, occasional higher values are outliers and the mean of a distribution is greatly affected by outliers as seen in figure 5 below, so median is a better representation of central tendency for this analysis because it isn't greatly affected by outliers, figure 6 shows the graphical representation of how the performance of the PSO algorithm improves as the population size increases. From the plot, we can confidently infer that the performance of the PSO algorithm as a measure of the fitness function (*because this is a minimization problem, the lower the fitness function, the better the performance of the algorithm*) increases as the population size increases. The cost function value reduces as the population size increases until it plateaus out where any increase in population size while keeping other

hyperparameters constant wouldn't cause any increase in the performance of the algorithm. From figure 6, you can observe that it plateaus around population size 6 which is equivalent to population size 6*50=300 (*coding rep*).

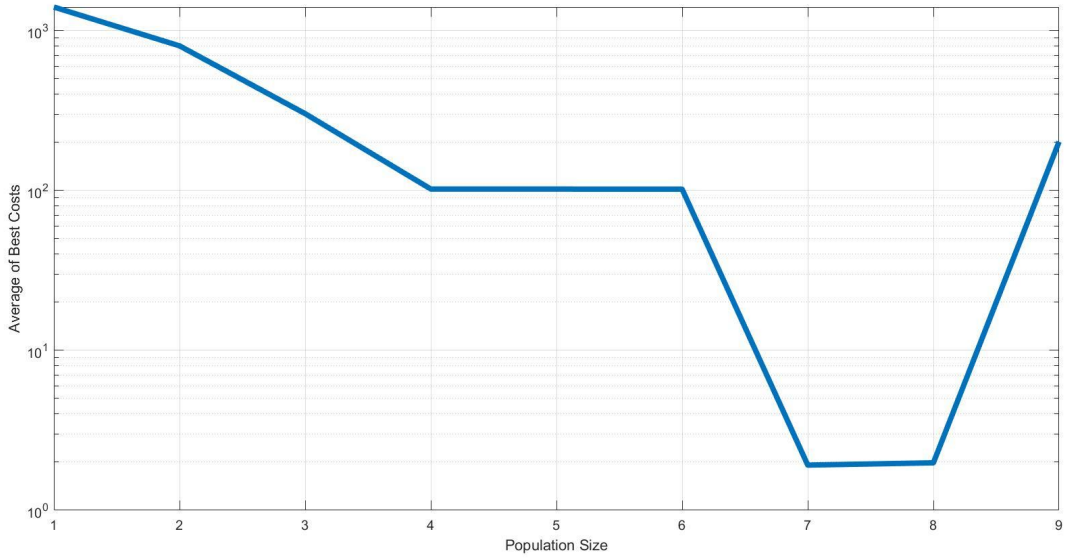

Figure 5

PSO Algorithm: Test example average objective function values vs Population size plot

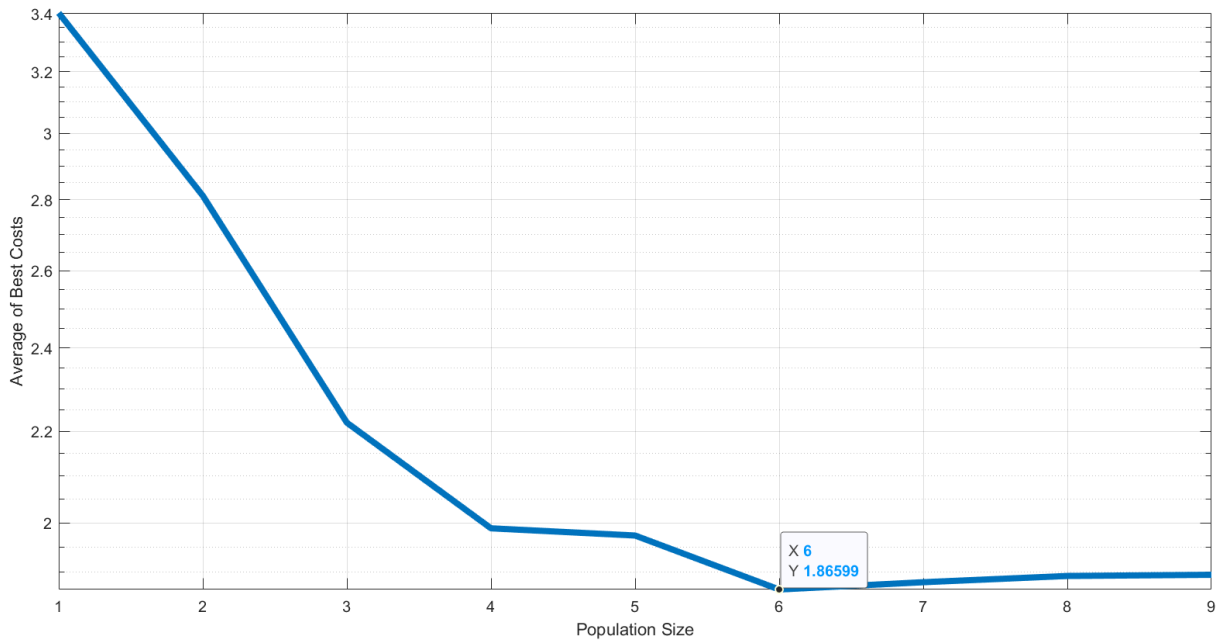

Figure 6

PSO Algorithm: Test example median objective function values vs Population size plot

4.4.2.3 Results and Discussions.

The third function of the PSO_main MATLAB file was to record the hyperparameters, population size, swarm positions, cost function value, constraint values and constraint value index, which ensures all constraints are satisfied while going through almost a thousand iterations to analyze the effect of change in population size. It tracks the iterative process and stores the necessary information about the lowest cost function value seen; which is referred to as the best cost so far in the code. These results were stored in the LeastCost_info mat file.

See MECH 580 project folder - PSO_Algorithms folder - PSO_Results - LeastCost_Info MAT file for stored values.

For a better understanding of the variables stored in the LeastCost_info mat file, below is a definition of the variables.

1. Best_CostSoFar - As the name implies is the lowest value of the cost function for this

minimization problem and the value was **1.7249**.

2. BestSol - This is a struct file with 2 fields. The first field BestSol.position contains the position or optimized values of parameters x resulting in the lowest cost value. The second field contains the best cost value which has been captured above.
   Parameter Values/Position = [0.205729639674165 3.47048866755492 9.03662391366253 0.205729639776849]
3. cO - This is a vector containing all constraint function values for the chosen best position. These values show that all the constraints are satisfied.
   Constraint Values = [-2.47179013967980e-06 -2.05973519769032e-05 -1.02684388769703e-10 -3.43298420794021 -0.0807296396741648 -0.235540322599970 -6.35089236311615e-07]
4. Params - This field is storage for relevant hyperparameter values. From the result, it can be seen that the best cost value was found at a population size of 450.

5.0 CONCLUSION

The purpose of this project was to help students gain practical skills in the field of engineering optimization using defined algorithms. The goal of the project was to develop robust optimization algorithms using the MATLAB programming language to solve real problems. The Gradient-based and PSO algorithms were developed and successfully applied to solve the pressure vessel and welded beam design problems.

The following conclusions can also be drawn from this project:

1. The performance of the Particle Swarm Optimization Algorithm is greatly influenced by the chosen swarm population size, from the analysis of the effect of population size on PSO performance, it is evident that while keeping other hyperparameters constant and increasing the population size the performance of the algorithm gets better till it levels out and causes no evident increase in the performance of the PSO algorithm. So it is recommended to increase the population size till you find the one that works best for the problem at hand.
2. The convergence of the gradient-based algorithm (ALM) or in general search methods/algorithms are greatly affected by the chosen starting point, depending on the starting point, the algorithm might get stuck at a local minimum thereby causing it to converge to suboptimal points. To combat this, it is recommended to start at several random initial positions till we find points that give better cost function values.
3. Lastly, hyperparameter tuning is a key factor in solving engineering optimization problems; a well tuned algorithm gives better performances and results.

6.0 REFERENCES.

1. Sohouli, A. (n.d.). *Engineering Optimization* [PowerPoint presentation]. Retrieved from University of Victoria Brightspace Website: Lecture 1_Before Class - Summer 2022 MECH 580 A03 (30637) (uvic.ca)
2. Sohouli, A (n.d.). *Nonlinear Programming: Constrained Optimization* [PowerPoint presentation]. Retrieved from University of Victoria Brightspace Website: Lecture 10_Before Class - Summer 2022 MECH 580 A03 (30637) (uvic.ca)
3. Arora, J., ed. (2004) *Introduction to Optimum Design*. 2nd ed. Elsevier Science & Technology.
4. Sohouli, A (n.d.). *Engineering Optimization: Assignment 2, Question 1* [PowerPoint presentation]. Retrieved from University of Victoria Brightspace Website:Ass. 2_Mech 450D - Summer 2022 MECH 580 A03 (30637) (uvic.ca)
5. Venkataraman, P., ed. *(2008) Applied Optimization with MATLAB Programming*. 2nd ed. John Wiley.