

OPTIMAL DESIGN OF HELICAL COMPRESSION SPRINGS

By

Muhammad Hakim Bin Mat Tasir

16418

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
Mechanical Engineering

JANUARY 2016

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

OPTIMAL DESIGN OF HELICAL COMPRESSION SPRINGS

By

Muhammad Hakim Bin Mat Tasir

16418

A project dissertation submitted to the
Mechanical Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
MECHANICAL ENGINEERING

Approved by,

(DR. DEREJE ENGIDA WOLDEMICHAEL)

UNIVERSITI TEKNOLOGI PETRONAS
BANDAR SERI ISKANDAR, PERAK

January 2016

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

MUHAMMAD HAKIM BIN MAT TASIR

ABSTRACT

This paper is aimed to present a calculation process to optimize the design of helical compression springs with regard to the criterion of minimum mass. By using conventional design a helical spring may be under-designed or over-designed. This process can be used in the early design stage or to re-design the spring. For this purpose, it revolves around the deflection limit, shear stress, frequency of surge waves, diameter constraint and minimum and maximum limits on design variables. This model of helical springs is translated into programming language and it is able to automatically solve the set of constraints. The optimal design of helical spring is generated in term of spring coil diameter, spring wire diameter and number of coil. It is resolved by using the Python Software and focus primarily on utilizing the PyOpt and NumPy package. The validation test of the software was done by sample calculation and also bench marking with other researcher's work.

ACKNOWLEDGEMENTS

First and foremost, First and foremost, I would like to extend my gratitude to Allah for his guidance and blessing for me to complete the final year project in the duration time given. Deepest appreciation to UTP Mechanical Engineering department who have provided and assist me directly and indirectly during the progress of this project being carried out. Other than that I would also to thank my family for giving me moral support throughout this final year, thus giving me strength in order to finish this final year project successfully. My acknowledgement of thankfulness to my supervisor, Dr. Dereje Engida Woldemichael for guiding and supervising me in completing the project. Without guidance from him, I may not be able to complete this project. I owe gratitude to him for his unwavering commitment and support for this final year project. Thank you for the supportive nature and for the concern of my professional and personal well-being development. Not to forget to all my friends that assists me in this project and for their unstoppable support to me for every problem that I faced throughout this project. Lastly, thank you very much to whoever involves directly and indirectly in this project. Without all of them, I may not be able to accomplish my final year project entitled “Optimal Design of Helical Compression Springs”.

TABLE OF CONTENT

CERTIFICATION OF APPROVAL	i
CERTIFICATION OF ORIGINALITY	ii
ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
CHAPTER 1: INTRODUCTION	1
1.1 Background of study.....	1
1.2 Problem Statement.....	2
1.3 Objectives.....	3
1.4 Scope of Study.....	3
CHAPTER 2: LITERATURE REVIEW	4
2.1 Design of Helical Springs.....	4
2.2 Optimal Design of Helical Springs.....	5
2.3 Composite Material Spring.....	6
2.4 Python Programming and PyOPT.....	7
CHAPTER 3: METHODOLOGY	8
3.1 Process Flow Chart.....	8
3.2 Materials, Tools and Equipment.....	10
3.3 Project Timeline.....	11
CHAPTER 4: RESULTS AND DISCUSSIONS	12
CHAPTER 5: CONCLUSION AND RECOMMENDATION	22
REFERENCES	23
APPENDICES	26

LIST OF FIGURE

Figure 1: Process Flow Chart of Project Development.....	8
Figure 2: Flow Chart of modelling methodology	9
Figure 3: Design of Tension/Compression Spring.....	13
Figure 4: Specification Sheet	19
Figure 5 Screen Shot of Program Developed.....	21

LIST OF TABLE

Table 1: List of equipment with its function.....	10
Table 2: Gantt Chart for FYP	11
Table 3: Information needed to design helical springs	12
Table 4: Design Variables values subjected to a set of constraint.	15
Table 5: Best Solution Iterated by Optimizers	21
Table 6: Constraints values for the proposed model of helical springs:	26

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Design is the engineering process in which involves imagination, creativity as well as the application and knowledge of technical and scientific skills with the use of material [1]. Designing process of a system includes trial and error procedure where the designer estimates and analyses it to check whether it behaves according to the subjected requirement. It is depending solely on designer's acumen, experience and intuition.

On the other hand, optimal design process is a step in which it maximizing the reliability, efficiency, durability and cost-effectiveness of a design without jeopardizing the ability to carry out the required task. Optimal design are iterative process where the trial design is analysed to determine if it is the best [2]. This project is expected to calculate and model the optimal design of helical spring from the given requirements according to certain constraints.

Springs can be found in most of our daily applications. In automotive industry, springs are crucial suspension elements to lower down the vertical vibrations, impacts and bumps due to road irregularities [3]. Helical springs are one of the common component used in the industry. There are many types of helical springs such as; compression, extension, torsion, conical, spiral, and disc spring.

1.2 Problem Statement

Main industrial program for spring design do not make the most of the powerful capabilities for optimization currently available. The current software only provides validation tool for the design of spring. On the other hand, this project is expected to introduce an advance optimization tool at any design stage and involves specifications for interval analysis and optimization process to provide the best design as output.

Helical Springs are used in various practical applications. The in-depth analysis and designing such components are developing over the years. There are certain available software in the market which calculate the optimal design of springs such as FED1/FED1+ calculates cylindrical helical compression springs in accordance with EN 13906-1. This software is developed by HEXAGON Software, Berlin. The total price for the software is EUR1,186.00 [4]. Apart from that, there is another software developed by Spring Manufacturers Institute that is charging 750\$ for licence and 350\$ for annual subscription [5]. Hence, the cost for getting of the software is not possible subjected to the budget constraint.

One example for usage of helical springs from automotive applications can be seen in damping systems of vehicle suspension. This have to be optimized under consideration of mechanical aspects, electrical aspects, electronical aspects and software-related to ensure a sufficient vibration absorption and a smooth ride. Without an optimized spring design, the vehicle damping system might be inadequate or overdesigned.

The purpose of this project is to develop the optimal design of helical spring. The designed spring are expected to carry a given axial load (tension-compression spring) without material failure. This tool will assist designers in modelling and developing the optimal design of helical spring based on the given requirements and constrains.

1.3 Objectives

There are three objectives to achieve in the project, which are:

- To develop models for the optimal design of helical springs.
- To develop a user friendly software for the optimal design of helical spring using the proposed model.
- To conduct validation test on software.

1.4 Scope of Study

Optimal Design of helical springs' scope of study includes the following; compression spring, extension spring, and torsional spring. The design process of the helical springs includes modelling, programming, debugging and validation of the program. In modelling, appropriate algorithm and calculation technics are done and validated before proceeding to further step. Some of the area of applications are automotive, industrial, goods, motorsports, aerospace, and oil & gas. Springs are very common in these area and the operation is more or less the same. The outcome of optimal design of helical spring will be validated and compared to conventional design by testing the program with existing product and by sample calculations.

CHAPTER 2

LITERATURE REVIEW

There are literature reviews that meet the objective of the project. Below shows some directly related literature reviews for understanding the purpose of the project.

2.1 Design of Helical Springs

A helical spring is usually coiled and has a uniform diameter cylinder and often manufactured in round wire. A conventional design of compression helical spring normally governs the dimensional limit that are solid height, outside diameter and inside diameter with regard to the load and deflection requirement. Compression helical springs end feature four types of ends; closed, open, ground and ungrounded. Combination of both either closed or open with either ground or ungrounded depends on the nature of application. When two compression springs are put in series, the deflection for the same load will be doubled and three springs with the same applied load with tripled the deflection. Whereas two compression springs in parallel can carry out the same deflection with double load applied and three springs are able to carry out triple amount of load with the same deflection [6].

Extension helical springs are able to store and absorb energy by resistance to a pulling force. There are many types of ends depending on the location or source of force. A conventional design of extension springs can be done when the amount of force at a length, number of cycle, and the length between attachment location. From the information given, the suitable spring wire diameter, mean diameter and number of coils can be determined by using the appropriate calculation and analysis [6].

Torsional helical springs' ends are rotated in angular deflection to exert a torsional moment. The design of torsion springs are always close-wounded whereas the end of the coil tends to be extended to provide levers to carry out the torque. The way it operates is when torque is applied, it always increase in body length and reduce in

coil diameter as it is deflected. In contrast to compression and extension helical springs, the wire is in bending stresses. The fatigue factor, inside coil diameter and pin diameter clearance when subjected to torque, maximum operating torque and corresponding rotation for static loading can be calculated when given the diameter of music wire, number of coil/turns, coil outside diameter and pin diameter [7].

2.2 Optimal Design of Helical Springs

On the other hand, optimal design differs from conventional design in term of maximizing the usage of material (e.g. minimum mass) and the highest fatigue life under given operation requirement. When conventional design are done, the product may be inadequate or overdesigned. [2] defined the optimization of coil spring with minimum-mass spring while complying two constraints. The deflection of the spring at the least Δ (in) and the frequency of the surge waves must not be less than ω_0 .

Another research on optimal design of helical springs [8] concluded that the choice of a large spring diameter has a result of large value of the spring mass and the reduction of the spring mass by optimal design may be of 16%. This was done by nonlinear programming model with given spring rate, the fatigue stress, buckling stability condition, torsional stress corresponding to the maximum force applied, and constraints of outer coil diameter and spring index.

M. Paredes, M. Sartor, and C. Masclet approached the optimization process by linking both industrial and mathematical knowledge. This method is done by determining the optimal extension spring design from a specification sheet where data are set with interval values. The optimization problem is solved using branch-and-bound process and added to a Generalized Reduced Gradient solution process using excel solver [9]. The outcome is that the spring with biggest fatigue life and efficient design for mass production are obtained.

Spring mass optimization can be done in many methods. Different methods are used to get different related outcome. Minimization of the helical spring mass and its first natural frequency as objective function is developed as a method of dynamic optimization for helical spring. The sensitivity of the spring to its first natural frequencies where the phenomenon of resonance appears with large displacements is

taken into account to establish a complex helical spring optimization problem. The results show that by moving away the helical spring first natural frequency from working zone can greatly reduce the spring mass and increase the design quality [10].

Another optimal design mathematical model of helical spring is established by theory of Particle Swarm Optimization (PSO) algorithm. With fourteen inequality constraints conditions and three design variables, the complex optimal design problem is simulated and thus the optimal values of the variables and minimal weight of the helical spring can be obtained. The result of simulation shows that by using PSO algorithm the design quality and efficiency can be improved greatly and weight of the helical spring can be greatly reduced [11].

Y. Takao T. Takeaki and G. Mitsuo in their paper formulate an optimal weight design problem of helical spring for a constrained allowable shearing stress, number of active coils and coil's average radius as a nonlinear integer programming problem and solve it directly by keeping the nonlinear constraint by using improved genetic algorithm. As a result, the number of decision (design) variables did not increase, the best compromised solution is generated [12].

This approach can be extended to the other types of springs such as extension, torsion or conical springs. More generally, as this approach can be time consuming, it could be of major interest to help designers when analytical formulae can be exploited.

2.3 Composite Material Spring

Studies on optimal design of composite material tubular compression spring with given the stiffness, the maximum deflection and the spring material properties show that the mass of tubular metal spring can be calculated as at least one half of the mass of common bar springs. By adopting materials with regard to the steel springs, the outcome show that the mass of helical springs can dramatically be reduced [13].

The results of analysis and optimization of a composite leaf spring [3] close that the composite springs has lower stresses, higher natural frequency whilst reducing the spring weight by 80%. The optimum spring width decrease hyperbolically and the thickness increase linearly from the spring eye towards the axle seat.

To reduce the weight of automotive coil springs, the use of carbon fiber/epoxy composite materials was considered. The verification of the static spring rates, as well as material and design parameters are determined. In order to get the properties of matrix and fiber, theoretical equations with modified correction coefficient were used and the results agreed with carbon fiber/epoxy composite material experimental results. From the inverse relationship between the twist angle and the shear modulus, the equivalent shear modulus of the composite was predicted. Weight reduction in excess of 55% could be obtained by applying the finite element analysis[14].

In one of the study done by Mehdi Bakhshesh and Majid Bakhshesh [15], where three composite materials; Carbon-Epoxy, Kevlar-Epoxy and Glass-Epoxy, shows that numerical results have been compared with theoretical results and found to be in good agreement. When fiber position has been considered to be in direction of the loading, the composite helical springs has been found to have lesser stress and has the most value compared to steel spring. When changing the percentage of fiber, Weight of spring has been reduced and has been shown that changing percentage of fiber does not affect the spring weight. Composite helical spring longitudinal displacement is more than steel composite helical spring is more than that of steel helical spring.

2.3 Python Programming and PyOpt

Python is an open source, free, high-level programming language that has a large following in the scientific computing community and supports object-oriented programming. Python is able to provide a readable and clear syntax in line with an intuitive object oriented with large number of data structures and types[16]. It is highly stable and run in an interactive mode. Thus, this make Python an easy to debug and learn programming language.

Python Optimization Framework or known as PyOpt is a Python-based package for solving nonlinear optimization. The goal of the optimization process can either be to maximize or minimize the objective function subjected to a set of constraints. There are several types of optimization solver available in the PyOpt package; SNOPT, NLPQL, NLPQLP, FSQP, SLSQP, PSQP, ALGENCAN, FILTERSD, COBYLA, SDPEN, SOLVOPT, ALPSO, ALSHO, MMA, GCMMA, CONMIN, MMFD, KSOPT, NSGA-II, and MIDACO [16]. However, only a few is selected for ease of analysis.

CHAPTER 3

METHODOLOGY

In order to achieve the objectives, this project focused on three types of helical springs which are compression spring, extension spring and torsional spring. For each type of spring, different objective function will be developed and further analysed to obtain different criteria which are either fatigue life or minimum weight. The result of optimal design of helical springs are then exported to Computer Aided Drawing (CAD) format for modelling.

3.1 Process Flow Chart

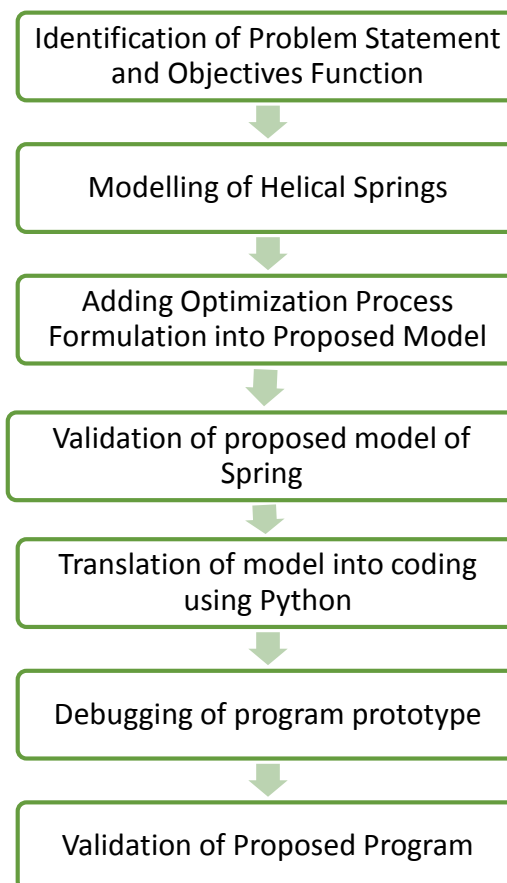


Figure 1: Process Flow Chart of Project Development

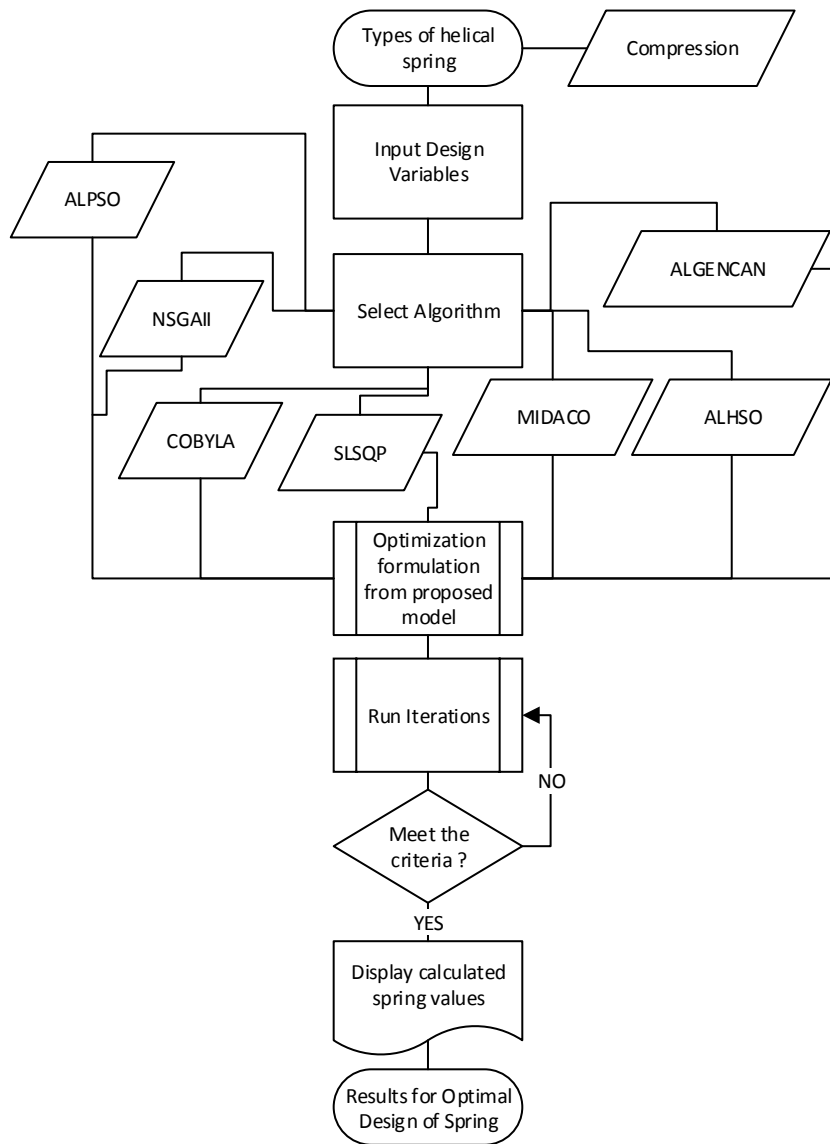


Figure 2: Flow Chart of modelling methodology

3.2 Materials, Tools and Equipment

Availability of resources specifically on materials, tools and equipment are one of the important parameters in executing this project. This inclusive of software (computer programs for project preparation, documentation and project execution), and project tools i.e. hardware. The items are listed in table 1.

Table 1: List of equipment with its function

No	Items	Function
1	Microsoft Office Excel 2013 Software	To carry out preliminary calculations and formulation.
2	Programming Software (Python, C++, C)	To develop equations and performing calculations.
3	GUI Development Software (tkInter, wxPython, PyGtk)	To prepare the friendly Graphical User Interface.
4	Scientific and Numeric Software (SciPy, Pandas, IPython)	To calculate and solve the formulation/algorithm.
5	CAD Software (freeCAD, TinkerCAD, Cura)	To develop 3D model of the design.

3.3 Project Timeline

Table 2: Gantt Chart for FYP

No	Activity	Final Year Project 1														Final Year Project 2														
		Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Modelling of Helical Spring																													
1.1	Identifying Parameters																													
1.2	Identifying Variables																													
1.3	Defining Constrains																													
1.4	Verification of modelling process																													
1.5	Complete model of spring																													
2	Optimization Process Formulation																													
2.1	Problem Statement																													
2.2	Data and information collection																													
2.3	Identification of design variables																													
2.4	Criterion to be optimized																													
2.5	Formulation of constraints																													
3	Programming / Coding																													
3.1	Identifying suitable program																													
3.2	Defining the algorithm																													
3.3	Converting model into programming																													
3.4	Running and debugging of program																													
3.5	Developing Graphical User Interface																													
4	Validation Process																													
4.1	Comparison with conventional design																													
4.2	Sample calculation for existing product																													

CHAPTER 4

RESULTS AND DISCUSSION

The informations needed to design a coil spring is tabulated as below:

Table 3: Information needed to design helical springs

DESCRIPTION	SYMBOL	UNIT
Deflection along the axis of spring	δ	m
Mean coil diameter	D	m
Wire diameter	d	m
Number of active coils	N	
Gravitational constant	g	$\frac{m}{s^2}$
Frequency of surge waves	ω	Hz
Weight Density of spring material	γ	$\frac{g}{m^3}$
Shear Modulus	G	$\frac{kg}{ms^2}$
Mass Density of Material	ρ	$\frac{kg s^2}{m^4}$
Allowable Shear Stress	τ_a	$\frac{kg}{ms^2}$
Number of inactive coils	Q	
Applied load	P	kg
Minimum Spring Deflection	Δ	m
Lower limit on surge wave frequency	ω_0	Hz
Limit on outer diameter of coil	D_0	m

The design equation for the spring are as follows:

Load deflection equation: $P = K\delta$ (a)

Spring Constant:
$$K = \frac{d^4 G}{8D^3 N} \quad (b)$$

Shear Stress:
$$\tau = \frac{8kPD}{\pi d^3} \quad (c)$$

Wahl Stress Concentration Factor:
$$k = \frac{4D-d}{4(D-d)} + \frac{0.615d}{D} \quad (d)$$

Frequency of surge waves:
$$\omega = \frac{d}{2\pi ND^3} \sqrt{\frac{G}{2\rho}} \quad (e)$$

PyOpt

PyOpt is a package designed for formulating and solving optimization problems[16]. It requires the user to be familiar with Python Programming software or any other similar programming language (C++, C, etc.). Basic structures required in solving optimization problem by using PyOpt are:

1. Objective Function (Minimum or maximum)
2. Constraints (Inequality or equality equations)
3. Design Variables (Continuous, Integer, or Discrete)
4. Optimizers (Algorithm Solver)

Spring Design Case Study

The optimization problem is described by Garg, H. [17] and it consist of minimizing the weight of a compression/tension spring as in Figure 5 subjected to constraints on limits on outside diameter, design variables, surge frequency, shear stress, and minimum deflection. The equations on objective functions and the respective constraints can be summarize as follow:

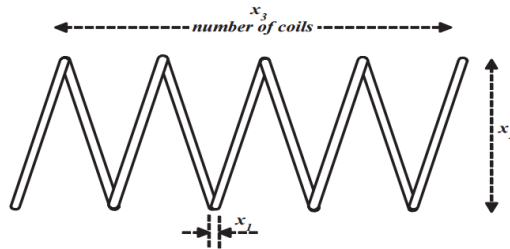


Figure 3: Design of Tension/Compression Spring

$$\begin{aligned}
\text{Minimize} \quad & f(X) = (x_3 + 2)x_2x_1^2 \\
\text{s.t} \quad & g(X) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0 \\
& g(X) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
& g(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \\
& g(X) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \\
& 0.05 \leq x_1 \leq 2; \quad 0.25 \leq x_2 \leq 1.3; \quad 2 \leq x_3 \leq 15
\end{aligned}$$

This problem has been solved using geometric programming and the solution was found iteratively as shown in Table 4 below. For different solver used, it can be seen that the values are different from each other. This is because each of them has a unique way of formulating the optimization problem to be solved.

In order to achieve the objectives, this project focused on three types of helical springs which are compression spring, extension spring and torsional spring. For each type of spring, different objective function will be developed and further analyzed to obtain different criterions which are either fatigue life or minimum weight. The result of optimal design of helical springs can later be used for desired application.

Therefore, to generate the possible configuration of helical compression springs, there are 7 algorithm techniques were selected namely; Sequential Least Squares Programming (SLSQP), Augmented Lagrangian with GENCAN (ALGENCAN, Constrained Optimization BY Linear Approximation (COBYLA), Augmented Lagrangian Particle Swarm Optimizer (ALPSO), Non-Sorting Genetic Algorithm II (NSGA2), Augmented Lagrangian Harmony Search Optimizer (ALHSO), and Mixed Integer Distributed Ant Colony Optimizer (MIDACO). To verify the applicability of this algorithm in generating the optimal result, one model of helical compression spring was taken into test.

Table 4: Design Variables values subjected to a set of constraint.

<i>Method</i>	<i>Design Variables</i>			<i>f(X)</i>
	x_1	x_2	x_3	
Harish [17]	0.0516891	0.3567200	11.288831	0.0126652
Akay and Karaboga[18]	0.051749	0.358179	11.203763	0.012665
Coelho[19]	0.051515	0.352529	11.538862	0.012665
Keveh and Talatahari [20]	0.051865	0.361500	11.00000	0.012643
Omran and Salman [21]	0.0516837	0.3565898	11.296471	0.0126652
Montes and Coelho [22]	0.051643	0.355360	11.397926	0.012698
PRESENT STUDY				
<i>SLSQP</i>	0.051718	0.357416	11.243917	0.0126612
<i>COBYLA</i>	0.051687	0.356668	11.28766	0.0126612
<i>NSGA – II</i>	0.054379	0.424763	8.190675	0.0128002
<i>ALGENCAN</i>	0.051687	0.356666	11.287729	0.0126612
<i>ALPSO</i>	0.050141	0.320925	13.725588	0.0126881
<i>MIDACO</i>	0.051353	0.356283	11.282812	0.0126261
<i>ALHSO</i>	0.068999	0.933431	2	0.0177758

The results obtained were subjected to initial value of $x_1 = 1.473727$, $x_2 = 1.93532$, and $x_3 = 9.918556$.

Modelling of helical springs.

The approach used in modelling the helical spring is by Geometric Programming.

1. Objective Function – to minimize the mass of spring for production.

$$f(x) = \text{Mass} = \frac{1}{4}(N + Q)\pi^2 D d^2 \rho \quad (\text{f})$$

2. Constraints – 4 design constraints are formulated in inequality form

Deflection of spring to be at least Δ

$$g(X) = \frac{8PD^3N}{d^4G} \geq \Delta \quad (\text{g})$$

Shear Stress in wire must be no greater than τ_a

$$g(X) = \frac{8PD}{\pi d^3} \left[\frac{(4D-d)}{4(D-d)} + \frac{0.615d}{D} \right] \leq \tau_a \quad (h)$$

Frequency of surge waves to be as great as possible

$$g(X) = \frac{d}{2\pi ND^2} \sqrt{\frac{G}{2\rho}} \geq \omega_0 \quad (i)$$

Summation of coil and wire diameter must be less than limit on outer diameter

$$g(X) = D + d \leq D_0 \quad (j)$$

3. Design Variables:

$$d_{min} \leq d \leq d_{max}$$

$$\text{Min and Max limits :} \quad D_{min} \leq D \leq D_{max} \quad (k)$$

$$N_{min} \leq N \leq N_{max}$$

- a. d = wire diameter, mm
- b. D = coil diameter, mm
- c. N = number of active coils

4. Optimizers

- i. SLSQP
- ii. COBYLA
- iii. MIDACO
- iv. ALHSO
- v. NSGA-II
- vi. ALGENCAN
- vii. ALPSO

Before programming of the proposed model into Python, each of the equation from (f) to (j) must be modified to ensure that it is calculated correctly.

The steps taken to convert into programming codes is shown as below:

From equation (f),

$$\begin{aligned} f(x) &= \frac{1}{4}(N + Q)\pi^2 D d^2 \rho \\ &= \frac{\pi^2 \rho}{4}(N + Q) D d^2 \end{aligned}$$

This equation is coded as below:

$$\begin{aligned} q &= ((\pi^{**2}) * D_) / 4 \\ f &= q * (x[2] + 2) * x[1] * x[0]^{**2} \end{aligned}$$

From equation (g),

$$\begin{aligned} g(X) &= \frac{8PD^3N}{d^4G} \geq \Delta \\ &= 1 - \frac{8PD^3N}{d^4G\Delta} \leq 0 \\ &= 1 - \frac{8P}{G\Delta} \times \frac{D^3N}{d^4} \leq 0 \end{aligned}$$

This equation is coded as below:

$$\begin{aligned} y &= (8 * P) / (G * K) \\ k &= 1 / y \\ g[0] &= 1 - (((x[1]^{**3}) * x[2])) / (k * (x[0]^{**4})) \end{aligned}$$

From equation (h),

$$\begin{aligned} \frac{8PD}{\pi d^3} \left[\frac{4D-d}{4(D-d)} + \frac{0.615d}{D} \right] &\leq \tau_a \\ \frac{8PD}{\tau_a \pi d^3} \left[\frac{4D-d}{4(D-d)} + \frac{0.615d}{D} \right] - 1 &\leq 0 \\ \frac{8P}{\tau_a \pi} \times \frac{D}{d^3} \left[\frac{4D-d}{4(D-d)} + \frac{0.615d}{D} \right] - 1 &\leq 0 \\ \frac{8P}{\tau_a \pi} \times \left[\frac{4D^2 - Dd}{4Dd^3 - d^4} + \frac{0.615}{d^2} \right] - 1 &\leq 0 \end{aligned}$$

$$\frac{8P}{\tau_a \pi} \times \left[\frac{4D^2 - Dd}{4Dd^3 - d^4} + \frac{0.615}{d^2} \right] - 1 \leq 0$$

$$\left[\left(\frac{8P}{\tau_a \pi} \right) \frac{4D^2 - Dd}{4Dd^3 - 4d^4} + \left(\frac{8P}{\tau_a \pi} \right) \frac{0.615}{d^2} \right] - 1 \leq 0$$

This equation is coded as below:

$$m = ((8 * P) / (\pi * T))$$

$$z = 1 / (m / 4)$$

$$g[1] = (((4 * (x[1]**2)) - (x[0] * x[1])) / (z * (((x[1] * (x[0]**3)) - (x[0]**4)))) + (1 / (u * x[0]**2)) - 1$$

From equation (i),

$$\begin{aligned} g(X) &= \frac{d}{2\pi ND^2} \sqrt{\frac{G}{2\rho}} \geq \omega_0 \\ &= 1 - \frac{d}{2\pi ND^2 \omega_0} \sqrt{\frac{G}{2\rho}} \leq 0 \\ &= 1 - \frac{d}{ND^2} \times \frac{1}{2\pi \omega_0} \times \sqrt{\frac{G}{2\rho}} \leq 0 \end{aligned}$$

This equation is coded as below:

$$r = (G / (2 * D)) ** 0.5$$

$$t = r / (2 * \pi * W)$$

$$g[2] = 1 - ((t * x[0]) / ((x[1]**2) * x[2]))$$

From equation (j),

$$\begin{aligned} g(X) &= D + d \leq D_0 \\ &= \frac{d + D}{D_0} - 1 \leq 0 \end{aligned}$$

This equation is coded as below:

$$g[3] = ((x[0] + x[1]) / N) - 1$$

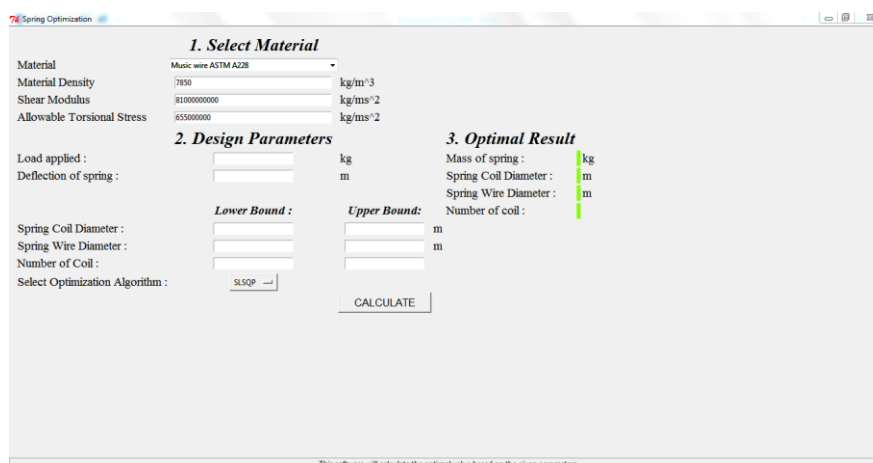
The Optimization Program.

The goal here is to minimize the weight of the spring with regards to the value given by the user. The program is divided into two parts in which the first part specifies the material used. The common type of spring is included in form of list that can be choose from in either Hard drawn steel wire ASTM A227, Music wire ASTM A228, Oil-tempered steel wire ASTM A229, Chrome-vanadium alloy steel wire ASTM A231, Stainless steel wire ASTM A313-302 or a custom material defined by the user itself.

The second part of the program is the design parameters which define the helical spring geometry which are:

- i. Load Applied (kg)
- ii. Deflection of spring (m)
- iii. Spring Coil Diameter (m)
- iv. Spring Wire Diameter (m)
- v. Number of Coil

These parameters are required and have to be specified in order to calculate the other. Only two operating parameters is considered in this study that is load applied and the deflection of the springs. The key here is to get the best possible configuration to define the spring. For that reason, parameters (iii) to (iv) are defined with interval values. This can be set by giving their bounds (lower and upper limits).



The screenshot displays the 'Spring Optimization' software interface, organized into three main sections:

- 1. Select Material:** A dropdown menu is set to 'Music wire ASTM A228'. Below it, four input fields show material properties: Material Density (7850 kg/m³), Shear Modulus (8100000000 kg/ms²), and Allowable Torsional Stress (655000000 kg/ms²).
- 2. Design Parameters:** This section includes input fields for 'Load applied' (kg) and 'Deflection of spring' (m). Below these are 'Lower Bound' and 'Upper Bound' fields for 'Spring Coil Diameter' (m) and 'Spring Wire Diameter' (m). A 'Number of Coil' field is also present. At the bottom, the 'Select Optimization Algorithm' is set to 'SLSQP', and a 'CALCULATE' button is visible.
- 3. Optimal Result:** This section displays the calculated results: 'Mass of spring' (kg), 'Spring Coil Diameter' (m), 'Spring Wire Diameter' (m), and 'Number of coil' (m). Each result is accompanied by a vertical green bar indicating its value.

A footer note at the bottom of the window states: 'This software will calculate the optimal value based on the given parameters'.

Figure 4: Specification Sheet

The data set in this way for stock compression springs has been tested by Peredes [23] where is provides an efficient and powerful means of expression. The

resolution circumstances requires a starting point close to the solution area. So, a random number is generated from the interval by implementing a function. This is done by importing the `random` module from python standard library where the program is able to generate random point from the given interval. This function is written as follow;

```
import random
A_ = random.uniform(A1,A2)
B_ = random.uniform(B1,B2)
C_ = random.uniform(C1,C2)
```

where $A_$ is a random number generated from the given interval $A1$ and $A2$. Before running the program, this kind of process needs the variables to be initialized, defining the calculation starting point. The choice of the starting point is very important as the algorithm is most likely to converge towards the optimal solution as closer this point is to the final solution.

Calculation Examples.

After successful modelling and programming, a program is developed to validate whether the proposed model is able to carry out the optimization problem. In order to validate this, a sample calculation has been done referencing to the materials and design parameter values provided by Harish Garg[17].

$$\text{Weight Density} = 11215.334213 \text{ kg/m}^3$$

$$\text{Shear Modulus} = 79289108835 \text{ Pa}$$

$$\text{Allowable Torsional Stress} = 79289708835 \text{ Pa}$$

$$\text{Maximum Deflection} = 0.0127 \text{ m}$$

$$\text{Applied Load} = 4.5359 \text{ kg}$$

$$\text{Coil Diameter Range} = 0.00127 \text{ m} \leq x_1 \leq 0.0508 \text{ m}$$

$$\text{Wire Diameter Range} = 0.00635 \text{ m} \leq x_2 \leq 0.00635 \text{ m}$$

$$\text{Number of Coil} = 2 \leq x_3 \leq 15$$

Table 5: Best Solution Iterated by Optimizers

PRESENT STUDY				
Method	Design Variables			$f(X)$: mass
	x_1 : coil diameter	x_2 : wire diameter	x_3 : no. of coil	
SLSQP	0.001270	0.028371	3.164777	0.0006562
COBYLA	0.001270	0.030718	2.490710	0.0006178
NSGA – II	0.001270	0.033020	2.005172	0.0005923
ALGENCAN	0.001270	0.033020	2.005168	0.0005922
ALPSO	0.001270	0.033020	2.005125	0.0005923
MIDACO	0.001270	0.033020	2.003163	0.000598
ALSHO	0.001270	0.029889	2.976400	0.0006661

All of the optimizers are able to get the optimal solution for the springs from the given values. It can be seen that ALGENCAN method gives the least weight and ALHSO method gives the highest weight. This is again because different method are using different method in formulating the constraints set.

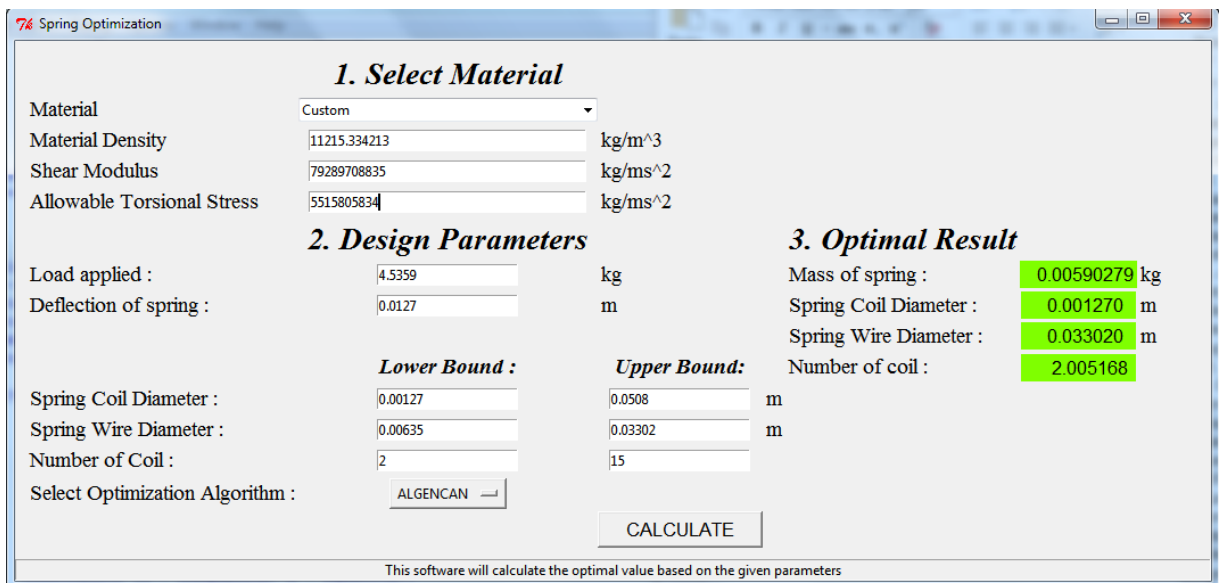


Figure 5 Screen Shot of Program Developed

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

The model used to develop into programming was based on Aurora[2] and Harish Garg[17]. It can be concluded that modelling of helical springs on compression/tension spring was successful and future works will include another type of springs that is torsional spring. This paper presents the Optimal Design of helical springs. Modelling of Helical Springs. The objective is to minimize the weight of the spring so that the cost of the design will also decrease. To evaluate the performance of PyOpt package, a case study was conducted and compared to other optimization method done by other researchers. The solutions found by each of the method is comparable with the other algorithm-based optimization methods. To ease the utilization of the package, a friendly Graphical User Interface (GUI) has been developed. All the results are tabulated and compared accordingly. Thus, it is concluded that PyOpt with Python Programming software is able to carry out the process of getting the Optimal Design of Helical Springs based on the proposed helical springs model.

There are plenty of improvement can be taken into consideration. This optimization problem can be more fine-tuned by adding more constraints into consideration. Sets of constraints that can be considered is the range for length of spring, range of operating load, and frequency of natural frequency. Adding more constraints will lead to a more accurate result. Addition of options in getting the objective in designing such as maximum fatigue life can also be taken into recommendation. From the outcome of the optimal design of helical spring calculation, the result can be transformed into CAD springs drawing which can later be used for actual prototype and also simulations. For distribution of software, this program can be simplified to executable files (.exe) where user without Python software can use it easily.

REFERENCES

- [1] J. Armstrong, *Design Matters*. London: Springer, 2008.
- [2] J. S. Arora, *Introduction to Optimum Design*: Academic Press, 2011.
- [3] M. M. Shokrieh and D. Rezaei, "Analysis and optimization of a composite leaf spring," *Composite Structures*, vol. 60, pp. 317-325, 5// 2003.
- [4] H. I. GmbH. (2015, 20 October). *Mechanical Engineering Software*. Available: <http://www.hexagon.de/>
- [5] S. M. Institute. (2015, 20 October 2015). *Advanced Spring Design Software (ASDS)*. Available: <http://www.smihq.org/>
- [6] P. R. N. Childs, "Chapter 15 - Springs," in *Mechanical Design Engineering Handbook*, P. R. N. Childs, Ed., ed Oxford: Butterworth-Heinemann, 2014, pp. 625-675.
- [7] R. G. Budynas, J. K. Nisbett, and J. E. Shigley, *Shigley's mechanical engineering design*: McGraw-Hill, 2008.
- [8] P. S. Aurel, "ON THE OPTIMAL DESIGN OF HELICAL SPRINGS OF AN AUTOMOBILE SUSPENSION," 2009.
- [9] M. Paredes, M. Sartor, and C. Masclet, "An optimization process for extension spring design," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 783-797, 12/21/ 2001.
- [10] M. Taktak, K. Omheni, A. Aloui, F. Dammak, and M. Haddar, "Dynamic optimization design of a cylindrical helical spring," *Applied Acoustics*, vol. 77, pp. 178-183, 3// 2014.

- [11] X. Qimin, L. Liwei, and X. Qili, "The optimal design and simulation of helical spring based on particle swarm algorithm and MATLAB," *WSEAS Transactions on Circuits and Systems*, vol. 8, pp. 84-93, 2009.
- [12] T. Yokota, T. Taguchi, and M. Gen, "A solution method for optimal weight design problem of helical spring using genetic algorithms," *Computers & industrial engineering*, vol. 33, pp. 71-76, 1997.
- [13] G. Mastinu, M. Gobbi, and C. Miano, "Optimal Design of Helical Spring," in *Optimal Design of Complex Mechanical Systems*, ed: Springer Berlin Heidelberg, 2006, pp. 303-330.
- [14] B.-L. Choi and B.-H. Choi, "Numerical method for optimizing design variables of carbon-fiber-reinforced epoxy composite coil springs," *Composites Part B: Engineering*, vol. 82, pp. 42-49, 12/1/ 2015.
- [15] M. Bakhshesh and M. Bakhshesh, "Optimization of steel helical spring by composite spring," *International journal of multidisciplinary science and engineering*, vol. 3, 2012.
- [16] R. E. Perez, P. W. Jansen, and J. R. R. A. Martins, "pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization," *Structural and Multidisciplinary Optimization*, vol. 45, pp. 101-118, 2011.
- [17] H. Garg, "Solving structural engineering design optimization problems using an artificial bee colony algorithm," 2014.
- [18] B. A. a. D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *Journal of Intelligent Manufacturing*, vol. 23, pp. 1001-1014, 2012.
- [19] L. S. Coelho, "Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems," *Expert Systems with Applications*, , vol. 37, pp. 1676-1683, 2010.

- [20] A. Kaveh and S. Talatahari, " An improved ant colony optimization for constrained engineering design problems," *Engineering Computations*, vol. 27, pp. 155-182, 2010.

- [21] M. G. H. Omran and A. Salman, "Constrained optimization using CODEQ," *Chaos, Solitons & Fractals*, vol. 42, 2009.

- [22] E. M. M. a. C. A. C. Coello, "An empirical study about the usefulness of evolution strategies to solve constrained optimization problems," *International Journal of General Systems*, vol. 37, pp. 443-473, 2008.

- [23] M. S. M. Paredes, C. Masclet, "Stock Spring Selection Tool," *SPRINGS*, 2000.

APPENDICES

Table 6: Constraints values for the proposed model of helical springs:

DESCRIPTION	SYMBOL	VALUE	UNIT
Deflection along the axis of spring	δ	<i>INPUT</i>	<i>m</i>
Mean coil diameter	<i>D</i>	<i>INPUT</i>	<i>m</i>
Wire diameter	<i>d</i>	<i>INPUT</i>	<i>m</i>
Number of active coils	<i>N</i>	<i>INPUT</i>	
Gravitational constant	<i>g</i>	9.81	$\frac{m}{s^2}$
Weight Density of spring material	γ	<i>INPUT</i>	$\frac{g}{m^3}$
Shear Modulus	<i>G</i>	<i>INPUT</i>	$\frac{kg}{ms^2}$
Mass Density of Material	ρ	$\gamma/9.81$	$\frac{kgs^2}{m^4}$
Allowable Shear Stress	τ_a	<i>INPUT</i>	$\frac{kg}{ms^2}$
Number of inactive coils	<i>Q</i>	2	
Applied load	<i>P</i>	<i>INPUT</i>	<i>kg</i>
Minimum Spring Deflection	Δ	<i>INPUT</i>	<i>m</i>
Lower limit on surge wave frequency	ω_0	100	Hz
Limit on outer diameter of coil	<i>D₀</i>	<i>INPUT</i>	<i>m</i>

Python Code for the Program Developed.

```
import os, sys, time
import pdb
import math
import linecache
import ctypes
from numpy import *
from pyOpt import Optimization
from pyOpt import SLSQP
from pyOpt import COBYLA
from pyOpt import NSGA2
from pyOpt import ALGENCAN
from pyOpt import ALPSO
from pyOpt import MIDACO
from pyOpt import ALHSO
from Tkinter import *
import ttk

master = Tk()
master.wm_title("Spring Optimization")
master.minsize(500,400)
master.maxsize(1366,689)

class MyListbox:
    def __init__(self, parent, title):
        self.parent = parent
        self.parent.title(title)
        self.parent.protocol("WM_DELETE_WINDOW", self.closes)

        self.myData= (
            ["1", "Hard drawn steel wire ASTM A227", "7850", "79500000000",
"59000000"],
            ["2", "Music wire ASTM A228", "7850", "81000000000", "655000000"],
            ["3", "Oil-tempered steel wire ASTM A229", "7850", "77000000000",
"658000000"],
            ["4", "Chrome-vanadium alloy steel wire ASTM A231", "7850", "77000000000",
"769000000"],
            ["5", "Stainless steel wire ASTM A313 - 302", "7900", "69000000000",
"483000000"],["6", "Custom", "", "", ""])

        self.establishment()

    def combobox_handler(self, event):
        current = self.combobox.current()
        self.MatDen.delete(0, END)
        self.MatMod.delete(0, END)
        self.MatTor.delete(0, END)

        self.MatDen.insert(END, self.myData[current][2])
        self.MatMod.insert(END, self.myData[current][3])
        self.MatTor.insert(END, self.myData[current][4])

    def establishment(self):
        mainFrame = Frame(self.parent)
        mainFrame.pack(fill=BOTH, expand=YES)

        self.statusBar = Label(mainFrame, text="This software will calculate the
optimal value based on the given parameters",relief=SUNKEN, bd=1)
```

```

self.statusBar.pack(side=BOTTOM, fill=X)

fr_left = Frame(mainFrame, bd=10)
fr_left.pack(fill=BOTH, expand=YES, side=LEFT)

Materials = [Types[1] for Types in self.myData]
self.combobox = ttk.Combobox(fr_left, values=Materials, width=40)
self.combobox.bind('<<ComboboxSelected>>', self.combobox_handler)
self.combobox.grid(row=1, column=1)

Label(fr_left, font = "Times 20 bold italic", text='1. Select
Material').grid(row=0, column=1)
Label(fr_left, font = "Times 14 ", text='Material').grid(row=1, column=0,
sticky=W)

Label(fr_left, font = "Times 14 ", text='Material Density').grid(row=2,
column=0, sticky=W)
self.MatDen = Entry(fr_left, width=40)
self.MatDen.grid(row=2, column=1)
Label(fr_left, font = "Times 14 ", text='kg/m^3').grid(row=2, column=2,
sticky=W)

Label(fr_left, font = "Times 14 ", text='Shear Modulus').grid(row=3, column=0,
sticky=W)
self.MatMod = Entry(fr_left, width=40)
self.MatMod.grid(row=3, column=1)
Label(fr_left, font = "Times 14 ", text='kg/m^2').grid(row=3, column=2,
sticky=W)

Label(fr_left, font = "Times 14 ", text='Allowable Torsional
Stress').grid(row=4, column=0, sticky=W)
self.MatTor = Entry(fr_left, width=40)
self.MatTor.grid(row=4, column=1)
Label(fr_left, font = "Times 14 ", text='kg/m^2').grid(row=4, column=2,
sticky=W)

Label(fr_left, font = "Times 20 bold italic", text="2. Design
Parameters").grid(row=7, column=1)

Label(fr_left, font = "Times 14 ", text="Load applied :").grid(row=8,
sticky=W)
e1 = Entry(fr_left)
e1.grid(row=8, column=1)
Label(fr_left, font = "Times 14 ", text='kg').grid(row=8, column=2, sticky=W)

Label(fr_left, font = "Times 14 ", text="Deflection of spring :").grid(row=9,
sticky=W)
e2 = Entry(fr_left)
e2.grid(row=9, column=1)
Label(fr_left, font = "Times 14 ", text='m').grid(row=9, column=2, sticky=W)

Label(fr_left, font = "Times 14 bold italic", text="Lower Bound
:").grid(row=11, column=1)
Label(fr_left, font = "Times 14 bold italic", text="Upper
Bound:").grid(row=11, column=2)

Label(fr_left, font = "Times 14 ", text="Spring Coil Diameter :").grid(row=12,
sticky=W)
e3 = Entry(fr_left)

```

```

e3.grid(row=12, column=1)
e4 = Entry(fr_left)
e4.grid(row=12, column=2)
Label(fr_left, font = "Times 14 ", text='m').grid(row=12, column=3, sticky=W)

Label(fr_left, font = "Times 14 ", text="Spring Wire Diameter :").grid(row=13,
sticky=W)
e5 = Entry(fr_left)
e5.grid(row=13, column=1)
e6 = Entry(fr_left)
e6.grid(row=13, column=2)
Label(fr_left, font = "Times 14 ", text='m').grid(row=13, column=3, sticky=W)

Label(fr_left, font = "Times 14 ", text="Number of Coil :").grid(row=14,
sticky=W)
e7 = Entry(fr_left)
e7.grid(row=14, column=1)
e8 = Entry(fr_left)
e8.grid(row=14, column=2)

Label(fr_left, font = "Times 14 ", text="Select Optimization Algorithm
:").grid(row=16,column=0, sticky=W)

Label(fr_left, font = "Times 20 bold italic", text="3. Optimal
Result").grid(row=7, column=5)
Label(fr_left, font = "Times 14 ", text="Mass of spring
:").grid(row=8,column=5, sticky=W)
Label(fr_left, font = "Times 14 ", text='kg').grid(row=8, column=7, sticky=W)
Label(fr_left, font = "Times 14 ", text="Spring Coil Diameter
:").grid(row=9,column=5, sticky=W)
Label(fr_left, font = "Times 14 ", text='m').grid(row=9, column=7, sticky=W)
Label(fr_left, font = "Times 14 ", text="Spring Wire Diameter
:").grid(row=10,column=5, sticky=W)
Label(fr_left, font = "Times 14 ", text='m').grid(row=10, column=7, sticky=W)
Label(fr_left, font = "Times 14 ", text="Number of coil
:").grid(row=11,column=5, sticky=W)

def callback():

    if not e1.get():
        def Mbox(title, text, style):
            ctypes.windll.user32.MessageBoxA(0, text, title, style)
            Mbox('ALERT', 'Please provide values before pressing calculate', 0)

    elif not e2.get():
        def Mbox(title, text, style):
            ctypes.windll.user32.MessageBoxA(0, text, title, style)
            Mbox('ALERT', 'Please provide values before pressing calculate', 0)
    elif not e3.get():
        def Mbox(title, text, style):
            ctypes.windll.user32.MessageBoxA(0, text, title, style)
            Mbox('ALERT', 'Please provide values before pressing calculate', 0)
    elif not e4.get():
        def Mbox(title, text, style):
            ctypes.windll.user32.MessageBoxA(0, text, title, style)
            Mbox('ALERT', 'Please provide values before pressing calculate', 0)
    elif not e5.get():
        def Mbox(title, text, style):
            ctypes.windll.user32.MessageBoxA(0, text, title, style)

```

```

        Mbox('ALERT', 'Please provide values before pressing calculate', 0)
elif not e6.get():
    def Mbox(title, text, style):
        ctypes.windll.user32.MessageBoxA(0, text, title, style)
    Mbox('ALERT', 'Please provide values before pressing calculate', 0)
elif not e7.get():
    def Mbox(title, text, style):
        ctypes.windll.user32.MessageBoxA(0, text, title, style)
    Mbox('ALERT', 'Please provide values before pressing calculate', 0)
elif not e8.get():
    def Mbox(title, text, style):
        ctypes.windll.user32.MessageBoxA(0, text, title, style)
    Mbox('ALERT', 'Please provide values before pressing calculate', 0)
elif not self.MatDen.get():
    def Mbox(title, text, style):
        ctypes.windll.user32.MessageBoxA(0, text, title, style)
    Mbox('ALERT', 'Please provide values before pressing calculate', 0)
elif not self.MatMod.get():
    def Mbox(title, text, style):
        ctypes.windll.user32.MessageBoxA(0, text, title, style)
    Mbox('ALERT', 'Please provide values before pressing calculate', 0)
elif not self.MatTor.get():
    def Mbox(title, text, style):
        ctypes.windll.user32.MessageBoxA(0, text, title, style)
    Mbox('ALERT', 'Please provide values before pressing calculate', 0)

Q = float(2)
D_ = float(self.MatDen.get())
D = float(float(self.MatDen.get())/9.81)
P = float(e1.get())
G = float(self.MatMod.get())
K = float(e2.get())
T = float(self.MatTor.get())
W = float(100)
N = float(e4.get())

A1 = float(e3.get())
B1 = float(e5.get())
C1 = float(e7.get())
A2 = float(e4.get())
B2 = float(e6.get())
C2 = float(e8.get())

q=((pi**2)*D_)/4

y=(8*P)/(G*K)
k=1/y

m=((8*P)/(pi*T))
z=1/(m/4)

i=(0.615*m)
u=1/i

r=(G/(2*D))**0.5
t=r/(2*pi*W)

def objfunc(x):

```

```

    f = q*(x[2]+2)*x[1]*x[0]**2
    g = [0.0]*4
    g[0] = 1-(((x[1]**3)*x[2]))/(k*(x[0]**4))
    g[1] = (((4*(x[1]**2))-(x[0]*x[1]))/(z*((x[1]*(x[0]**3))-
(x[0]**4))))+(1/(u*x[0]**2))-1
    g[2] = 1-((t*x[0])/((x[1]**2)*x[2]))
    g[3] = ((x[0]+x[1])/1.5)-1

    fail = 0
    return f,g, fail

import random
A_ = random.uniform(A1,A2)
B_ = random.uniform(B1,B2)
C_ = random.uniform(C1,C2)

opt_prob = Optimization('SPRING OPTIMIZATION',objfunc)
opt_prob.addVar('coil_diameter','c',lower=A1,upper=A2,value=A_)
opt_prob.addVar('wire_diameter','c',lower=B1,upper=B2,value=B_)
opt_prob.addVar('number_of_coil','c',lower=C1,upper=C2,value=C_)
opt_prob.addObj('f')
opt_prob.addCon('g1','i')
opt_prob.addCon('g2','i')
opt_prob.addCon('g3','i')
opt_prob.addCon('g4','i')
print opt_prob

if v1.get()=="SLSQP":
    slsqp = SLSQP()
    slsqp.setOption('IPRINT',-1)
    slsqp(opt_prob,sens_type='FD')
    print opt_prob.solution(0)
    opt_prob.write2file(outfile='springEX.txt', disp_sols=True)
    file = open("springEX.txt","r+")
    selectline = file.readlines()
    data = selectline[37]
    bodo = data[10:23]
    selectline2 = file.readlines()
    data2 = selectline[41]
    bodo2 = data2[22:37]
    selectline3 = file.readlines()
    data3 = selectline[42]
    bodo3 = data3[22:37]
    selectline4 = file.readlines()
    data4 = selectline[43]
    bodo4 = data4[22:37]
    file.close()
    l6['text'] = bodo
    l7['text'] = bodo2
    l8['text'] = bodo3
    l9['text'] = bodo4

elif v1.get()=="COBYLA":
    cobyala = COBYLA()
    cobyala.setOption('IPRINT',0)
    cobyala(opt_prob)
    print opt_prob.solution(0)
    opt_prob.write2file(outfile='springEX.txt', disp_sols=True)

```

```

file = open("springEX.txt","r+")
selectline = file.readlines()
data = selectline[36]
bodo = data[10:23]
selectline2 = file.readlines()
data2 = selectline[40]
bodo2 = data2[22:37]
selectline3 = file.readlines()
data3 = selectline[41]
bodo3 = data3[22:37]
selectline4 = file.readlines()
data4 = selectline[42]
bodo4 = data4[22:37]
file.close()
l6['text'] = bodo
l7['text'] = bodo2
l8['text'] = bodo3
l9['text'] = bodo4

elif v1.get()=="NSGA2":
    nsga2 = NSGA2()
    nsga2.setOption('PrintOut',0)
    nsga2(opt_prob)
    print opt_prob.solution(0)
    opt_prob.write2file(outfile='springEX.txt', disp_sols=True)
    file = open("springEX.txt","r+")
    selectline = file.readlines()
    data = selectline[36]
    bodo = data[10:23]
    selectline2 = file.readlines()
    data2 = selectline[40]
    bodo2 = data2[22:37]
    selectline3 = file.readlines()
    data3 = selectline[41]
    bodo3 = data3[22:37]
    selectline4 = file.readlines()
    data4 = selectline[42]
    bodo4 = data4[22:37]
    file.close()
    l6['text'] = bodo
    l7['text'] = bodo2
    l8['text'] = bodo3
    l9['text'] = bodo4

elif v1.get()=="ALGENCAN":
    algencan = ALGENCAN()
    algencan.setOption('iprint',0)
    algencan(opt_prob)
    print opt_prob.solution(0)
    opt_prob.write2file(outfile='springEX.txt', disp_sols=True)
    file = open("springEX.txt","r+")
    selectline = file.readlines()
    data = selectline[38]
    bodo = data[10:23]
    selectline2 = file.readlines()
    data2 = selectline[42]
    bodo2 = data2[22:37]
    selectline3 = file.readlines()
    data3 = selectline[43]

```

```

bodo3 = data3[22:37]
selectline4 = file.readlines()
data4 = selectline[44]
bodo4 = data4[22:37]
file.close()
l6['text'] = bodo
l7['text'] = bodo2
l8['text'] = bodo3
l9['text'] = bodo4

elif v1.get()=="ALPSO":
    alpsone = ALPSO()
    alpsone.setOption('fileout',0)
    alpsone(opt_prob)
    print opt_prob.solution(0)
    opt_prob.write2file(outfile='springEX.txt', disp_sols=True)
    file = open("springEX.txt","r+")
    selectline = file.readlines()
    data = selectline[38]
    bodo = data[10:23]
    selectline2 = file.readlines()
    data2 = selectline[42]
    bodo2 = data2[22:37]
    selectline3 = file.readlines()
    data3 = selectline[43]
    bodo3 = data3[22:37]
    selectline4 = file.readlines()
    data4 = selectline[44]
    bodo4 = data4[22:37]
    file.close()
    l6['text'] = bodo
    l7['text'] = bodo2
    l8['text'] = bodo3
    l9['text'] = bodo4

elif v1.get()=="MIDACO":
    midaco_none = MIDACO()
    midaco_none.setOption('IPRINT',-1)
    midaco_none.setOption('MAXEVAL',50000)
    midaco_none(opt_prob)
    print opt_prob.solution(0)
    opt_prob.write2file(outfile='springEX.txt', disp_sols=True)
    file = open("springEX.txt","r+")
    selectline = file.readlines()
    data = selectline[36]
    bodo = data[10:23]
    selectline2 = file.readlines()
    data2 = selectline[40]
    bodo2 = data2[22:37]
    selectline3 = file.readlines()
    data3 = selectline[41]
    bodo3 = data3[22:37]
    selectline4 = file.readlines()
    data4 = selectline[42]
    bodo4 = data4[22:37]
    file.close()
    l6['text'] = bodo
    l7['text'] = bodo2
    l8['text'] = bodo3

```



```

        19['text'] = bodo4

    elif v1.get()=="ALHSO":
        alhso_none = ALHSO()
        alhso_none.setOption('fileout',0)
        alhso_none(opt_prob)
        print opt_prob.solution(0)
        opt_prob.write2file(outfile='springEX.txt', disp_sols=True)
        file = open("springEX.txt","r+")
        selectline = file.readlines()
        data = selectline[38]
        bodo = data[10:23]
        selectline2 = file.readlines()
        data2 = selectline[42]
        bodo2 = data2[22:37]
        selectline3 = file.readlines()
        data3 = selectline[43]
        bodo3 = data3[22:37]
        selectline4 = file.readlines()
        data4 = selectline[44]
        bodo4 = data4[22:37]
        file.close()
        16['text'] = bodo
        17['text'] = bodo2
        18['text'] = bodo3
        19['text'] = bodo4

    else:
        print "Please Select Algorithm"

    b1 = Button(fr_left, font="bold", text="CALCULATE", width=15,
command=callback)
    b1.grid(row=17, column=2)

    v1 = StringVar()
    v1.set("SLSQP")

    option = OptionMenu(fr_left, v1,
"SLSQP", "COBYLA", "NSGA2", "ALGENCAN", "ALPSO", "MIDACO", "ALHSO")
    option.grid(row=16, column=1)

    16 = Label(fr_left, bg='chartreuse', font="bold", text='')
    16.grid(row=8, column=6)
    17 = Label(fr_left,bg='chartreuse', font="bold", text='')
    17.grid(row=9, column=6)
    18 = Label(fr_left,bg='chartreuse', font="bold", text='')
    18.grid(row=10, column=6)
    19 = Label(fr_left,bg='chartreuse', font="bold", text='')
    19.grid(row=11, column=6)

    def closes(self, event=None):
        self.parent.destroy()

if __name__ == '__main__':
    app = MyListbox(master, "Spring Optimization")
    master.mainloop()

```