Parametric Design of a MEMS Accelerometer

ME 128 – Project 2

Professor Liwei Lin
Spring 2006

Scott Moura
SID 15905638

April 5, 2006

ME128 – Computer-Aided Mechanical Design

Spring 2006

Name: **Scott Moura**

Project: #2 **Design of a MEMS Accelerometer**

Introduction:	10	____________
Theory:	20	____________
Code Verification:	10	____________
Summary of Results:	10	____________
Results:	10	____________
FEM Results:	10	____________
Conclusions:	20	____________
Computer Source Code	10	____________
Total:	100	____________

April 5, 2006

Table of Contents

Introduction

The objective of this investigation is to find the optimum design for a typical MEMS (Micro Electro-Mechanical Systems) accelerometer, which satisfies a set of given constraints. Due to the complex nature of the problem, a genetic algorithm (GA) is developed for optimization. The GA attempts to minimize the die area while satisfying all other engineering goals. Four major dimensions (L_1, L_2, L_3, y_m) are determined from this optimization. The optimal design from the theoretically derived genetic algorithm is compared to hand derived calculations and finite element analysis in order to ascertain its accuracy and verify the results.

The genetic algorithm, developed in MATLAB, utilizes concepts from evolution in order to ascertain the best performing design. Although computationally intense, this method is very useful for complex problems in which system of differential equations are difficult to write. All the details for implementing this method are provided in the optimization method section.

The software used for finite element analysis (FEA) is SolidWorks with the built-in COSMOSWorks finite element method (FEM) analysis package. A three dimensional model of the best design is created and analyzed under three different loading conditions. These cases represent the limiting conditions of operation, and are therefore of interest for evaluating resolution and survival. Unlike the theoretical calculations, this method considers the mass of the beams, whereas the MATLAB computations assume the beam masses are negligible compared to the proof mass size.

Nomenclature

Variables

A	Die Area, [μm^2]
b	Beam Depth, [μm]
E	Young's modulus, [psi]
F	Force [N]
h	Beam Width [μm]
I	Moment of Inertia about x-axis, [in^4]
k	Spring Constant for entire system [N/m]
L	Beam length, [in]
m	Proof Mass size [kg]
M	Moment, [in-lb]
x	Position along length of beam, [in]
λ	Children Design Variables
Λ	Design Variables, Parent
Π	Objective Function
θ	Angle of deflection, [rad]
Φ	Random Value
ρ	Density [kg/m^3]
σ	Stress, [GPa]

Subscripts

a	Anchor
m	Proof Mass
x	x-direction
y	y-direction

Background

MEMS accelerometers are used for a variety of applications, namely automobile airbag systems. Consumer products, such as computer games, cell phones, pagers, PDAs, advanced robotics, laptop computers, computer input devices, camcorders, digital cameras, and after-market SD card accessories are also common applications[1]. In each of these applications, size and accuracy are the most critical characteristics of the sensor's performance. As such, these factors will be considered throughout this investigation.

The MEMS accelerometer under study operates under the same principles of a spring-mass system, shown schematically in Figure 1. However, instead of springs, the accelerometer employs a double folded beam flexure system. The mass being displaced is the proof mass. To measure displacement, one capacitive sensor exists on each side of the proof mass. The sensitivity of these sensors is proportional to the size length of the mass. These sensors send back a voltage signal proportional to the displacement measured. By equating Hooke's Law to Newton's second law, $kx = ma$, the acceleration experienced by the mass can easily be determined.

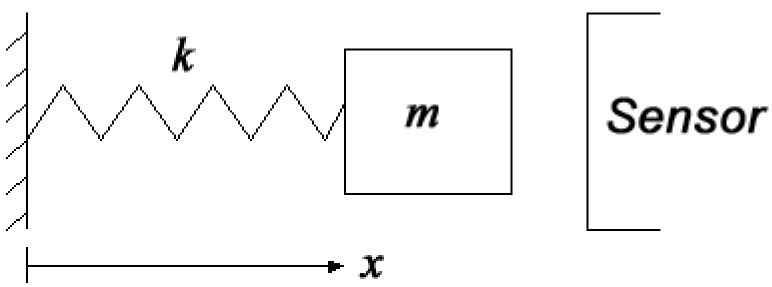


Figure 1: Schemetic Diagram of Spring-Mass System, with sensor measuring displacement.

A representative diagram of the MEMS accelerometer is provided in Figure 2. The narrow beams of the assembly make up the equivalent spring for the entire system. The proof mass is shown in the middle and the anchors are restrained

[1] Kionix. *MEMS Accelerometers-Inertial Sensors.* Accessed April 2, 2006 <http://www.kionix.com/Accelerometers/accelerometers.htm>

to the MEMS substrate. Ideally, the legs (beams) of the structure would be made as long as possible for softer springs and the mass as large as possible. This would result in the greatest displacement and thus maximize acceleration resolution. However, a maximum die area constrains the size of the accelerometer, which we seek to minimize in order to produce more devices per wafer. As a result, these two factors fight each other when finding the optimal design.

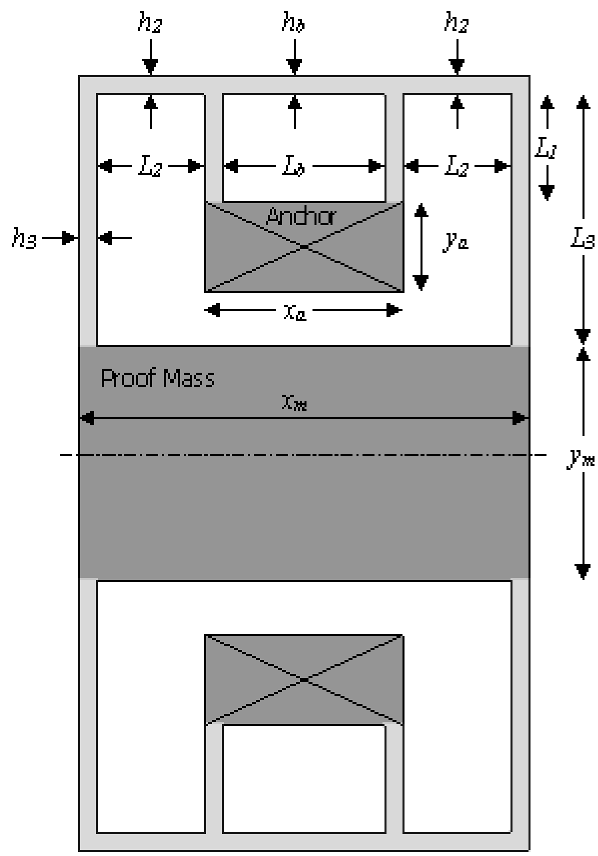


Figure 2: MEMS Accelerometer Diagram

Theory

The governing differential equations for elastic beam bending serve as the basis for the theoretical analysis on the device. Since the focus of this project is parametric optimization and not an in-depth review of beam force analysis, all relevant equations and derivations are provided and shown below. They utilize the concepts of superposition to sum forces and moments at each beam's intersection. The results include the forces and moments at the points of maximum stress (A and H) and the total effective spring constant for the entire system. These equations are later used to analyze the beams properties for a given set of design parameters.

Relevant Equations

$$F_{xA} = 6dEI\left(6L_1L_2^2 + L_BL_2^2 + 4L_BL_1L_2 + 24L_1L_2L_3 + 12L_BL_1L_3 + 4L_BL_2L_3\right) \Bigg/ \left(\begin{array}{l} 3L_1^4L_2^2 + 12L_1L_2^2L_3^3 + 2L_BL_1^3L_2^2 + 2L_BL_1^4L_2 + 12L_1^4L_2L_3 \\ +12L_1L_2L_3^4 + 6L_BL_1^4L_3 + 6L_BL_1L_3^4 + 2L_BL_2^2L_3^3 \\ +6L_BL_2L_1^2L_3^2 + 8L_BL_1L_2L_3^3 + 2L_BL_2L_3^4 + 8L_BL_1^3L_2L_3 \end{array}\right)$$

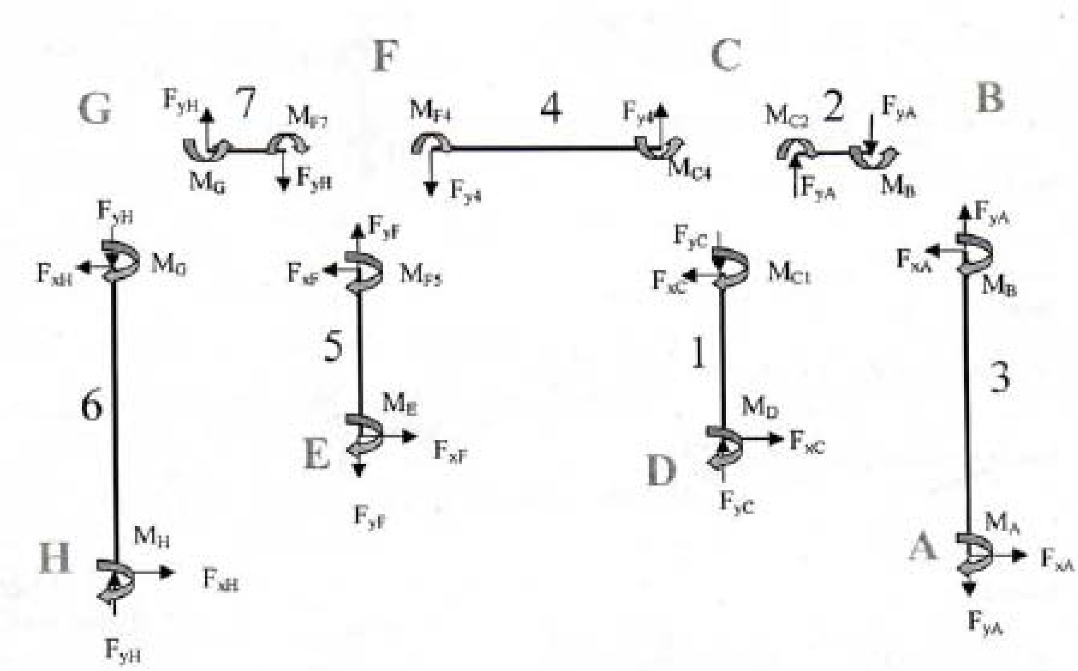
$$F_{yA} = \frac{18dEI}{L_2}\left(-L_BL_1^2L_2 + 2L_BL_1L_3^2 - 2L_BL_1^2L_3 + 6L_1L_2L_3^2 + L_BL_2L_3^2\right) \Bigg/ \left(\begin{array}{l} 3L_1^4L_2^2 + 12L_1L_2^2L_3^3 + 2L_BL_1^3L_2^2 + 2L_BL_1^4L_2 + 12L_1^4L_2L_3 \\ +12L_1L_2L_3^4 + 6L_BL_1^4L_3 + 6L_BL_1L_3^4 + 2L_BL_2^2L_3^3 \\ +6L_BL_1^2L_2L_3^2 + 8L_BL_1L_2L_3^3 + 2L_BL_2L_3^4 + 8L_BL_1^3L_2L_3 \end{array}\right)$$

$$M_A = 6dEI\left(\begin{array}{l} 6L_BL_1L_3^2 + 12L_1L_2L_3^2 + 2L_BL_2L_3^2 + 6L_1L_2^2L_3 \\ +L_BL_2^2L_3 + 4L_BL_1L_2L_3 + L_BL_1^2L_2 \end{array}\right) \Bigg/ \left(\begin{array}{l} 3L_1^4L_2^2 + 12L_1L_2^2L_3^3 + 2L_BL_1^3L_2^2 + 2L_BL_1^4L_2 + 12L_1^4L_2L_3 \\ +12L_1L_2L_3^4 + 6L_BL_1^4L_3 + 6L_BL_1L_3^4 + 2L_BL_2^2L_3^3 \\ +6L_BL_2L_1^2L_3^2 + 8L_BL_1L_2L_3^3 + 2L_BL_2L_3^4 + 8L_BL_1^3L_2L_3 \end{array}\right)$$

$$F_{xH} = F_{xA} \qquad F_{yH} = F_{yA} \qquad M_H = M_A$$

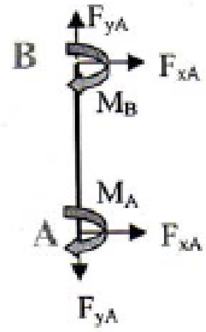
$$k_x = \frac{4F_{xA}}{d} = 24EI\left(6L_1L_2^2 + L_BL_2^2 + 4L_BL_1L_2 + 24L_1L_2L_3 + 12L_BL_1L_3 + 4L_BL_2L_3\right) \Bigg/ \left(\begin{array}{l} 3L_1^4L_2^2 + 12L_1L_2^2L_3^3 + 2L_BL_1^3L_2^2 + 2L_BL_1^4L_2 + 12L_1^4L_2L_3 \\ +12L_1L_2L_3^4 + 6L_BL_1^4L_3 + 6L_BL_1L_3^4 + 2L_BL_2^2L_3^3 \\ +6L_BL_2L_1^2L_3^2 + 8L_BL_1L_2L_3^3 + 2L_BL_2L_3^4 + 8L_BL_1^3L_2L_3 \end{array}\right)$$

Provided Derivations



Known: d

Unknowns: $F_{xA} \quad F_{yA} \quad M_A \quad F_{xH} \quad F_{yH} \quad M_H$

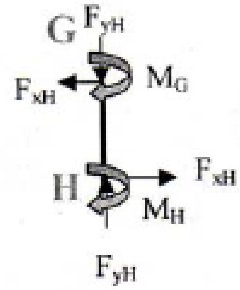


Beam 3:

$M_B = F_{xA}L_3 - M_A \qquad \theta_B = -\dfrac{F_{xA}L_3^2}{2EI} + \dfrac{M_A L_3}{EI}$

$x_A = d\,, \qquad -x_B = -d - \dfrac{F_{xA}L_3^3}{6EI} + \dfrac{M_A L_3^2}{2EI}$

Beam 6:



$M_G = F_{xH}L_3 - M_H \qquad \theta_G = -\dfrac{F_{xH}L_3^2}{2EI} + \dfrac{M_H L_3}{EI}$

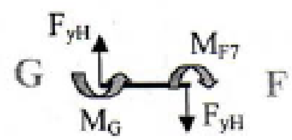
$-x_G = -d - \dfrac{F_{xH}L_3^3}{6EI} + \dfrac{M_H L_3^2}{2EI}$

Eqn 1. $x_B = x_G$

Beam 7:

$M_{F7} = -F_{yH}L_2 + M_G \qquad \theta_F = \theta_G + \dfrac{F_{yH}L_2^2}{2EI} - \dfrac{M_G L_2}{EI}$

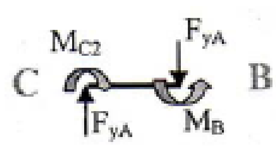
Eqn 2. $y_F = 0 = \theta_G L_2 + \dfrac{F_{yH}L_2^3}{6EI} - \dfrac{M_G L_2^2}{2EI}$



Beam 2:

$M_{C2} = -F_{yA}L_2 + M_B \qquad \theta_C = \theta_B + \dfrac{F_{yA}L_2^2}{2EI} - \dfrac{M_B L_2}{EI}$

Eqn 3. $y_C = 0 = \theta_B L_2 + \dfrac{F_{yA}L_2^3}{6EI} - \dfrac{M_B L_2^2}{2EI}$

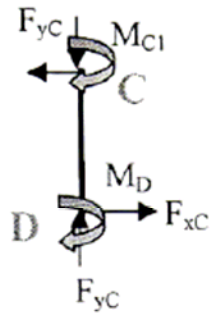


Beam 1:

$$\theta_D = 0 = \theta_C - \frac{F_{xC}L_1^2}{2EI} + \frac{M_{C1}L_1}{EI}$$

$$x_D = 0 = x_B + \theta_C L_1 - \frac{F_{xC}L_1^3}{6EI} + \frac{M_{C1}L_1^2}{2EI}$$

So $M_{C1} = f_1(\theta_C, x_B)$ $\qquad F_{xC} = F_{xD} = f_2(\theta_C, x_B)$



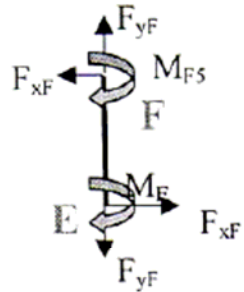
Beam 5:

$$\theta_E = 0 = \theta_F - \frac{F_{xF}L_1^2}{2EI} + \frac{M_{F5}L_1}{EI}$$

$$x_E = 0 = x_B + \theta_F L_1 - \frac{F_{xF}L_1^3}{6EI} + \frac{M_{F5}L_1^2}{2EI}$$

So $M_{xF} = f_1(\theta_F, x_B)$ $\qquad F_{xE} = F_{xF} = f_2(\theta_F, x_B)$

Eqn 4. $F_{xA} + F_{xH} + F_{xE} + F_{xD} = 0$



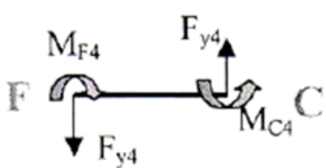
Beam 4:

$M_{C4} = M_{C1} + M_{C2}$ $\qquad M_{F4} = M_{F5} + M_{F7}$

$F_{y4} = F_{yD} - F_{yA} = F_{yE} - F_{yH}$ $\qquad M_{C4} + M_{F4} + F_{y4}L_B = 0$

Eqn 5. $$\theta_C = \theta_F - \frac{F_{y4}L_B^2}{2EI} - \frac{M_{F4}L_B}{EI} = \theta_F + \frac{L_B}{2EI}(M_{C1} + M_{C2} - M_{F7} - M_{F5})$$

Eqn 6. $$y_C = 0 = \theta_F L_B - \frac{F_{y4}L_B^3}{6EI} - \frac{M_{F4}L_B^2}{EI} = \theta_F L_B + \frac{L_B^2}{EI}(M_{C1} + M_{C9} - 2M_{F5} - 2M_{F7})$$



Constraints & Engineering Goals

Coupled with the equations shown above are both equality and inequality constraints. These parameters are requirements determined by the manufacturing process, cost constrictions, material properties, and design goals. They are given by the following equations:

Equality Constraints

$$h_1 = h_2 = h_3 = h_b = 1.8\mu m$$
$$b_1 = b_2 = b_3 = b_m = 1.8\mu m$$
$$L_b = 150\mu m$$
$$E = 160GPa$$
$$\rho = 2.33\frac{g}{cm^3}$$
$$y_a = 100\mu m$$
$$x_a = L_b + 2h_1$$
$$x_m = L_b + 2L_2 + 2h_1 + 2h_3$$

Inequality Constraints

$$L_1 \geq 20\mu m$$
$$L_2 \geq 20\mu m$$
$$L_3 \geq 20\mu m$$
$$L_3 \geq h_2 + y_a + L_1$$

With these constraints in mind, this investigation seeks to accomplish certain engineering goals. The goals provided by the project descriptions are the following:

Minimum DC acceleration resolution: $a_r \leq 0.0005g$

Maximum DC acceleration survival: $a_{max} \geq 2000g$

Maximum die area: $A_d = x_m\left(y_m + 2L_3 + 2h_2\right) \leq 90{,}000\mu m^2$

Maximum stress in suspension: $\sigma_{max} \leq 1.6GPa$

Motion Resolution: $x_r = \dfrac{\left(10^{-1}\mu m\right)^2}{y_m}$

After a thorough analysis of the various designs, it becomes clear that the maximum die area becomes the most significant limiting constraint. In fact, it is impossible to satisfy all of the constraints with a design that is less than 90,000µm^2 in area. As a result, this constraint is relaxed and the optimum design is found. Detailed descriptions of these findings are discussed in this report.

Genetic Algorithm Optimization Method

In many cases where parametric optimization is desired, methods such as the Lagrange Multiplier, First-order Necessary Conditions (FONC), and Second-order Necessary Conditions (SONC) are preferred for analytical solutions. However, this design problem contains features too complicated to represent in a system of differential equations. Moreover, determining inactive constraints through monotonicity is nearly impossible using logic tables. As a result, a genetic algorithm is developed and employed. Although computationally intensive, the basic concept behind the genetic algorithm is simple. This method also allows the freedom to change any of the constraints with relative ease.

The employment of the genetic algorithm follows a very simple iterative technique to minimize an objective function, given by Π. The details of this objective function will be explained later. The design variables are represented by Λ.

$$\Lambda = \{L_1, L_2, L_3, y_m\}$$

To apply the GA we, we take random values for the design variables within the following ranges,

$$\begin{aligned} 20\mu m \le L_1 \le 500\mu m \\ 20\mu m \le L_2 \le 100\mu m \\ 100\mu m \le L_3 \le 500\mu m \\ 100\mu m \le y_m \le 500\mu m \end{aligned}$$

These ranges were chosen based on the minimum size constraints and maximum area constraints, in addition to general observation and intuition about the final design's optimal geometry.

The algorithm begins with populations of 100 random strings within the aforementioned ranges. For each string, the following objective function is evaluated.

$$\Pi = A$$

where A represents the die area. Clearly, this method will find the beam lengths that minimize the total die area, which is only one of the design goals. However, it can be argued that die area is the most significant goal, since it directly relates to cost. By minimizing the die area we maximize the number of accelerometers that can be manufactured on the same silicon wafer. This means more accelerometers for the same price. Also initial calculations have shown that this constraint is the most difficult to satisfy. Therefore, this simple objective function is chosen. The other four design goals are evaluated for each design and thrown out if they do not meet the inequality constraints. This is done by giving Π a value of infinity.

Once the design goals are checked, the values of Π are sorted to determine the top ten performing designs (10 smallest Π's). These top ten 'parents' are then 'mated' to produce ten 'offspring' or 'children' design values. The following scheme is used to perform this 'mating' task,

$$\lambda^{i} = \Phi^{(I)}\overline{\Lambda}^{i} + \left(1 - \Phi^{(I)}\right)\overline{\Lambda}^{i+1}$$
$$\lambda^{i+1} = \Phi^{(II)}\overline{\Lambda}^{i} + \left(1 - \Phi^{(II)}\right)\overline{\Lambda}^{i+1}$$

where λ^{i} are the children design values, $\overline{\Lambda}^{i}$ are the parent design values, and $\Phi^{(I)}, \Phi^{(II)}$ are random values between zero and one.

At this point, we take the ten parents and ten children and combine them with 70 new random design variables to create a second generation of 100 strings. With these 100 new strings, the objective function is calculated again, checked with the constraints, sorted, and so forth. This process is run for 50 generations (iterations). At the end of the 50 generations, the top ranked design variables give the values of L_1, L_2, L_3, and y_m. On top of performing 50 iterations of this genetic algorithm, we perform this entire scheme 1000 times, thus allowing 1000 different starting populations. As a result, 1000 different top performing design values are found for 1000 different starting populations. The top value represents the design variables used for the optimum design, recommended by this report.

Code Verification

The genetic algorithm written in MATLAB is compared to hand derived calculations to prove accuracy within the code. The results for the most significant accelerometer properties are shown in Table 1 for the optimum design variables. The values match perfectly, thus ensuring the program's precision.

Table 1: Code Verification Table.

	Die Dimensions	Die Area	Proof Mass Size	Effective Spring Constant
MATLAB Calculations	197.2 µm x 785.9 µm	154,979.48 μm^2	0.23629 µg	0.32911 N/m
Hand Calculations	197.2 µm x 785.9 µm	155,000 μm^2	0.2363 µg	0.33 N/m

Summary of Results

The genetic algorithm developed finds the optimum design that minimizes die area while just satisfying all the other design constraints. The results are given in Table 2.

Table 2: Results of the Genetic Algorithm Optimization Method.

Optimum Design Property	Value
L_1	146.0 µm
L_2	20 µm
L_3	248.3 µm
y_m	285.7 µm
Die Area	154979.48 μm^2
Motion Resolution	3.5002 x 10^{-5} µm
Acceleration Resolution	0.0049747g
Acceleration Survival	20998.1619g
Max Stress in Suspension	0.2166 GPa
Die Dimensions	197.2 µm x 785.9 µm
Proof Mass Size	0.23629 µg
Effective Spring Constant	0.32911 N/m

Initial calculations have shown that it is impossible to satisfy all of the constraints, as they are given in the project description. Specifically, the 90,000 μm^2 maximum die area and 0.0005g acceleration resolution are the limiting constraints. As a result, the die area constraint is relaxed and minimized for values below 160,000 μm^2 while the minimum acceleration resolution is increased by a factor of ten. Although it is possible to satisfy the minimum acceleration resolution given, it is found that changing this constraint can significantly minimize die area. The cost savings are chosen in favor of this engineering goal. Additionally, an acceleration resolution of 0.005g still satisfies most applications. The engineering goals and their corresponding constraints are shown below. The necessary modifications for the design are shown in the right-most column.

Table 3: Design Goal Metric Table. The die area and acceleration resolution constraints are relaxed in order to minimize cost and satisfy the other constraints.

Design Goals	Optimal Design	Initial Constraint	Modified Constraint
L_1	146.0 μm	**> 20 μm**	**-**
L_2	20 μm	**> 20 μm**	**-**
L_3	248.3 μm	**> 20 μm** **> $h_2 + y_a + L_1$**	**-**
Die Area	154979.48 μm^2	**< 90,000 μm^2**	**< 160,000 μm^2**
Motion Resolution	3.5002 x 10^{-5} μm	**< 3.518 x 10^{-5} μm**	**-**
Acceleration Resolution	0.0049747g	**< 0.0005g**	**< 0.005g**
Acceleration Survival	20998.1619g	**2,000g**	**-**
Max Stress in Suspension	0.2166 GPa	**1.6 GPa**	**-**

Theoretical Results

The theory-based genetic algorithm is designed using the MATLAB programming environment. The iterative genetic process minimizes the die area, represented by the objective function, Π , while satisfying all other design criteria. The best performing design is saved for each successive starting population to converge on the optimum values. Figure 3 illustrates this fact by displaying the minimum value of the objective function for the first 100 starting populations. Clearly, the genetic algorithm succeeds in progressively finding designs with smaller design areas. Additionally, the genetic algorithm appears to converge to the best design asymptotically, as the starting population count increases.

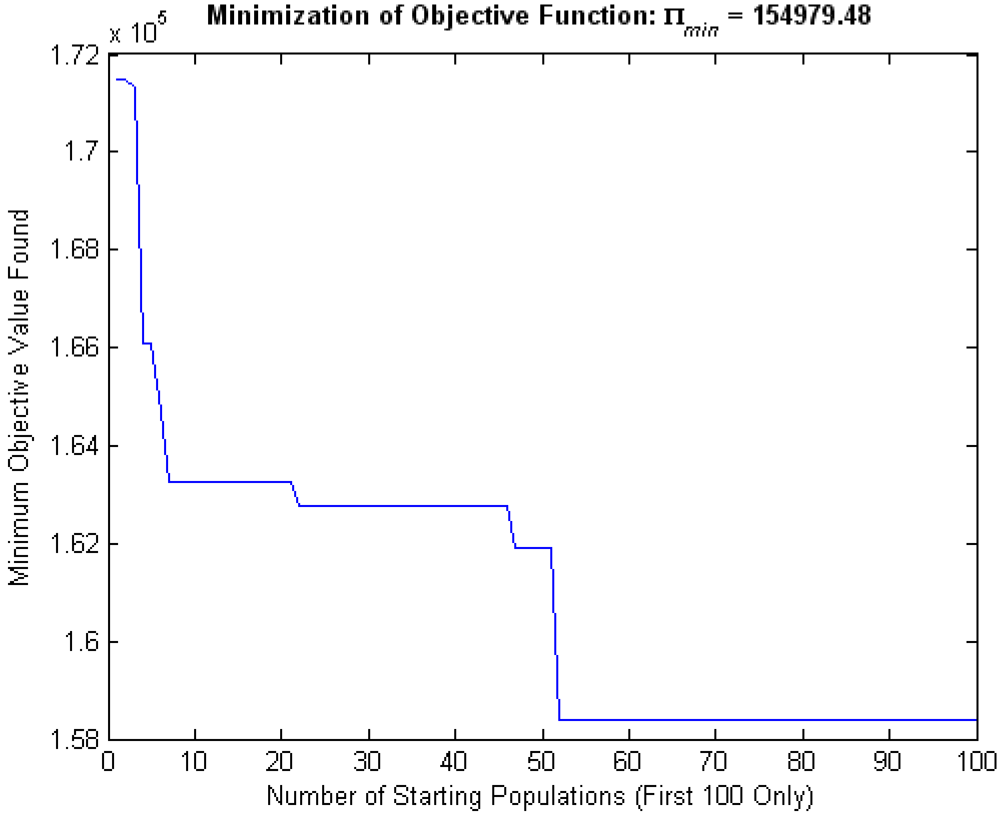


Figure 3: Minimization of Objective Function for first 100 starting populations. The value of Π decreases as better performing designs are found.

After 1,000 starting populations of 50 generations (iterations) have been computed, the five best performing designs are output to the user. The final results are shown in Table 4. Note that L_2 converges to 20 µm, the minimum value possible. Although 20 µm is never achieved exactly (due to the exclusive nature of the random generator function) it can be assumed that the optimum design has L_2 = 20 µm. Conversely, L_1, L_2, and y_m do not appear to converge to a value. This must imply that there is a range of optimum values that can be used to achieve the best design. As a result, the optimum dimensions presented here are only one set of the possible values.

Table 4: Five Best Performing Designs computed using the Genetic Algorithm.

Rank	L_1 (µm)	L_2 (µm)	L_3 (µm)	y_m (µm)	$\Pi = A$ (µm^2)
1	146.0	20.62	248.3	285.7	154979.48639
2	142.6	20.48	253.0	281.6	156774.86310
3	128.8	21.16	235.7	312.3	157088.24299
4	117.7	20.57	232.8	324.2	157360.54186
5	159.7	20.43	267.1	259.8	157965.94329

The first ranked design is the one evaluated and recommended in this report. In order to analysis this design, three different operating conditions are considered. They include the sensor motion resolution (~0.005g), acceleration survival (2000g), and maximum displacement due to geometry constraints (20 µm). For each condition, the displacement of the proof and mass and maximum stress is determined. The first test analyzes the accelerometer when it experiences a load just large enough to register a signal from the sensor. The second test considers the accelerometer's reaction to experiencing a load of 2000g, which is the minimum value for survival. The final test investigates the maximum stress in suspension of the accelerometer at the maximum displacement physically possible. Any displacement greater than 20 µm would cause the beams to crash into the proof mass. The theoretical values for the displacement of the proof mass and maximum stress are shown in Table 5. Theses values are validated through finite element analysis.

Table 5: Theoretical Values of Proof Mass Displacement and Maximum Stress.

Case	Condition	Displacement (µm)	Stress (GPa)
1	Sensor Motion Resolution (~0.005g)	3.5002×10^{-5}	3.81×10^{-7}
2	Acceleration Survival (2000g)	14.072	0.1524
3	Maximum Displacement	20	0.2166

FEM Results

The finite element analysis was performed within SolidWorks using the COSMOSWorks package. This program allows the user to select a custom material for the part by inputting user-defined values for the material properties. The material properties used for this analysis are given by the equality constraints. Additionally, a value of zero was assumed for the Poisson's ratio. Together, all of the material properties are

$$E = 160GPa$$
$$\rho = 2330\frac{kg}{m^3}$$
$$\nu = 0$$

A few simple techniques are applied to simulate the accelerometer under various loads (Figure 4). The two anchors are constrained to the substrate by placing fixed restraints on their bottom surfaces, shown by the green arrows. This process eliminates all six degrees of freedom (DOFs). Simulating acceleration is an equally simple process: define the magnitude of gravity in the x-direction equal to the acceleration in question (shown schematically by the red arrow). Using these techniques, the three conditions are analyzed. The mesh size used for all case studies is equal to 6.6 µm.

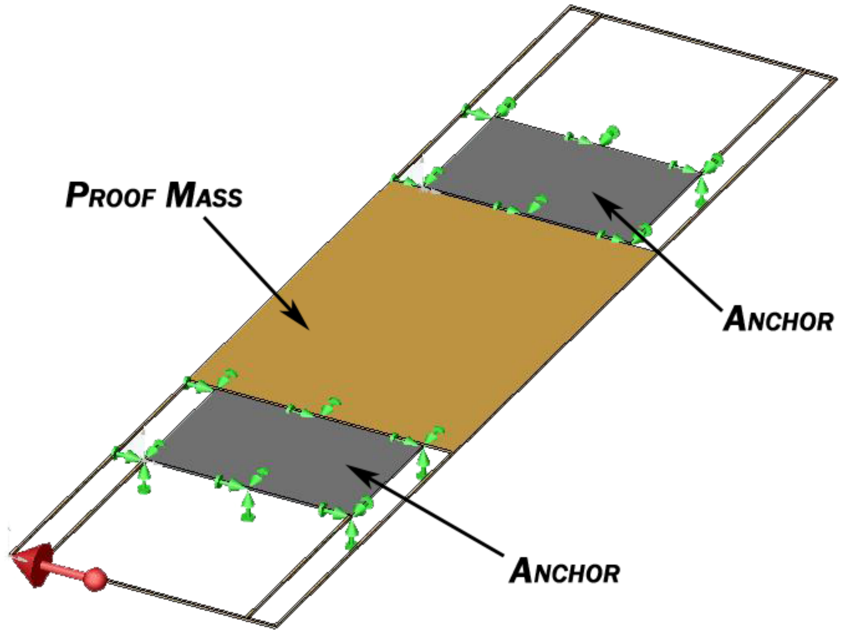


Figure 4: Accelerometer Model is SolidWorks COSMOS simulation study.

Case 1 – Sensor Motion Resolution

In the first case, the proof mass undergoes an acceleration of 0.005g, which roughly generates a displacement equal to the sensor's motion resolution. Figures 5 and 6 respectively show the maximum displacement and stress experienced by the device. The values computed by the COSMOS FEA are 3.546 x 10^{-5} µm for the displacement and 3.766 x 10^{-7} GPa for the maximum stress.

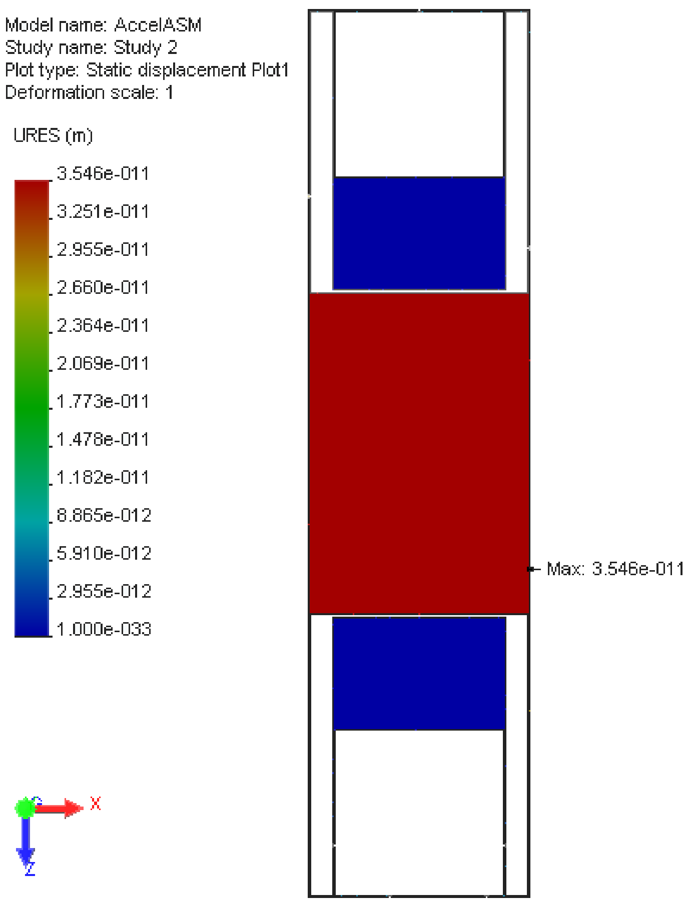


Figure 5: Proof Mass Displacement at 0.005g.

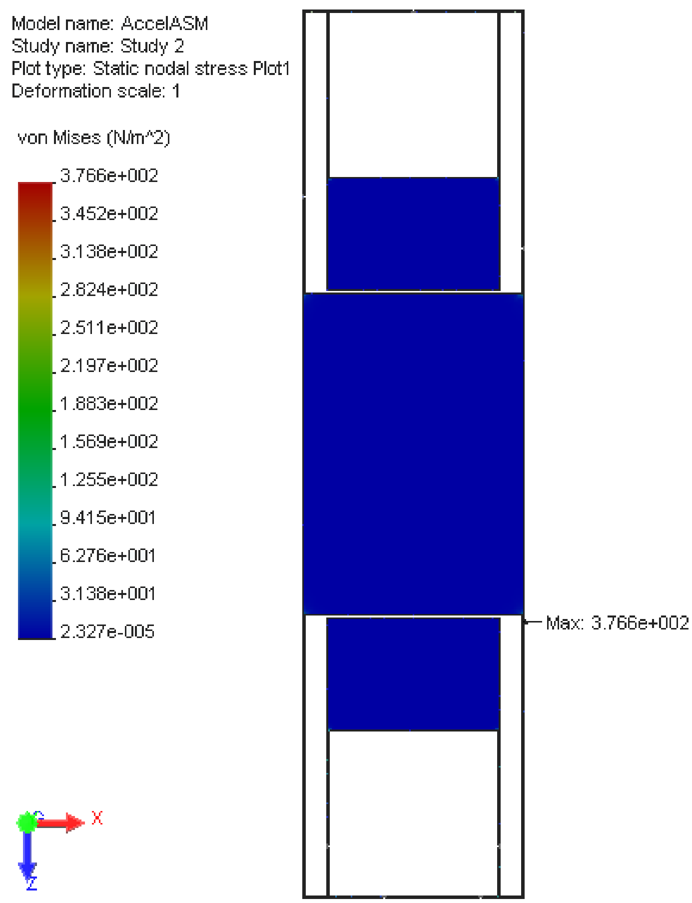


Figure 6: Maximum Stress at 0.005g.

A more detailed view of the area under maximum stress is shown in Figure 7. Note that this area is the same area for which we assumed maximum stress would occur.

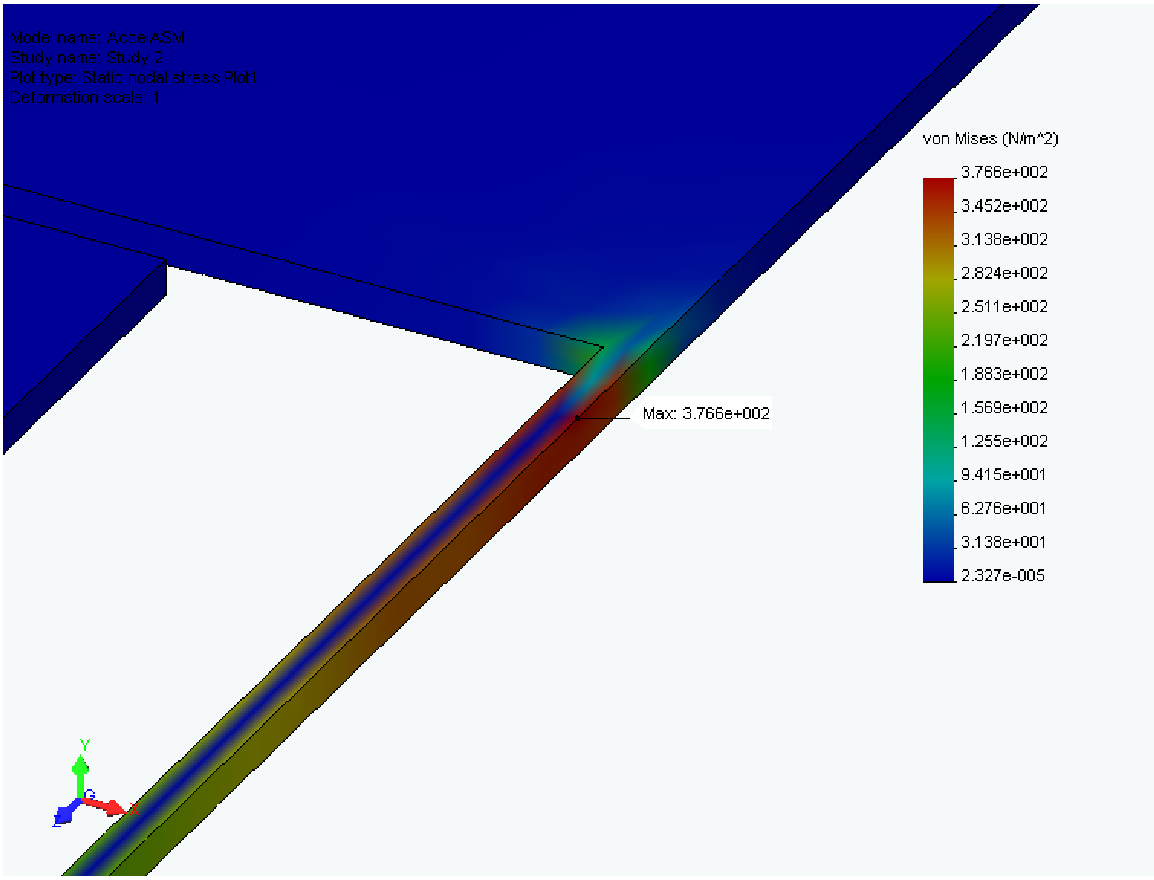


Figure 7: Detailed view of Maximum Stress at 0.005g.

Case 2 – Acceleration Survival

One of the engineering goals is to ensure that the device material will not fail for accelerations greater than 2000g. The theoretical calculations performed by MATLAB state that failure occurs at an acceleration of about 21,000g, over 10 times greater than the minimum required value. Nonetheless, we wish to verify the theoretical calculations for stress and displacement at 2000g by performing FEA. The results from this analysis are shown in Figures 8 and 9.

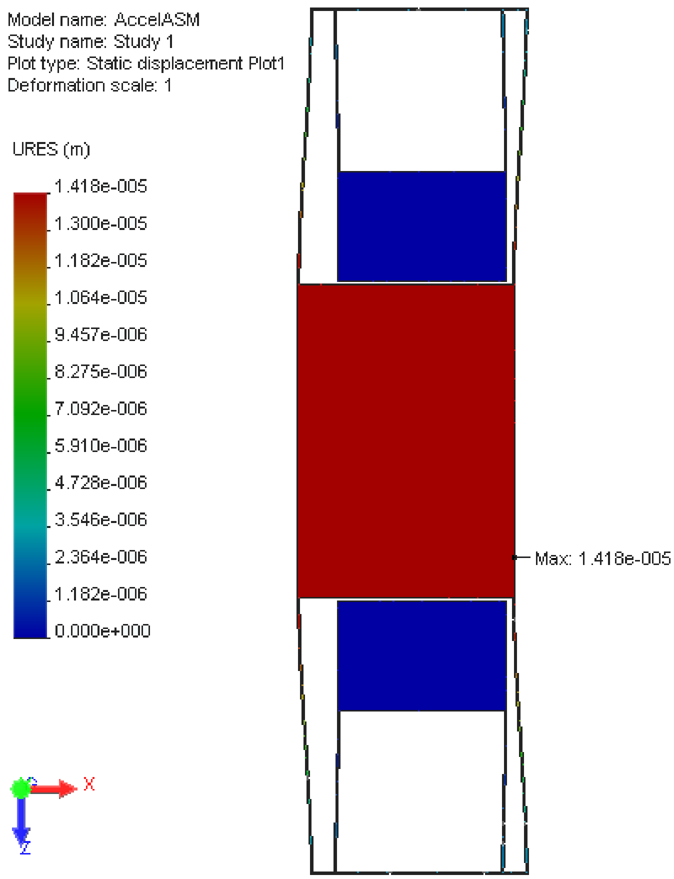


Figure 8: Proof Mass Displacement at 2000g.

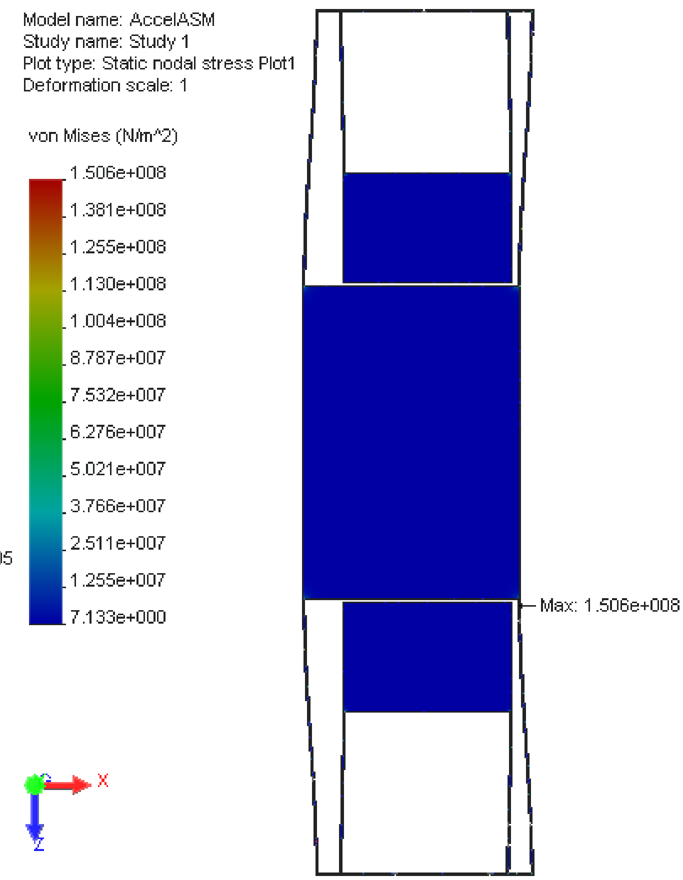


Figure 9: Maximum Stress at 2000g.

Figure 10 shows a detailed view of the area of maximum stress. It is evident from this test that the suggested accelerometer design meets the required acceleration survival constraint by over a factor of ten.

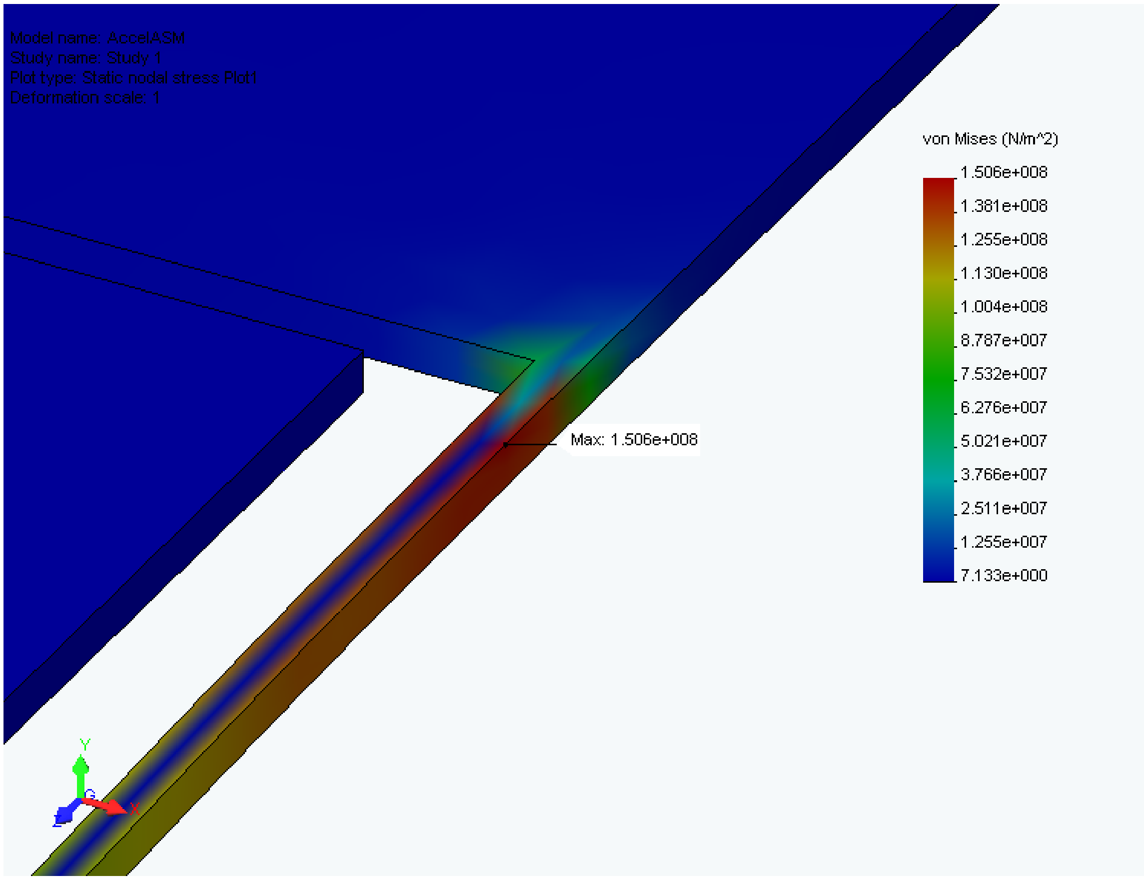


Figure 10: Detailed view of Maximum Stress at 2000g.

Case 3 – Maximum Displacement

As discussed previously, the proof mass may experience a displacement up to 20 µm before the beams crash into the anchors. This limiting case provides useful information about the maximum possible stress. In order to model a displacement of 20 µm, MATLAB computations are performed to find the corresponding acceleration. This acceleration (2842.5279g) is input as the magnitude of gravity in order to model a displacement of 20 µm. Figure 11 exposes the approximate nature of this test case, since the FEA displacement result is slightly larger than 20 µm at about 20.16 µm. Note also how the beams come into contact with the anchors, thus confirming our theoretical computations.

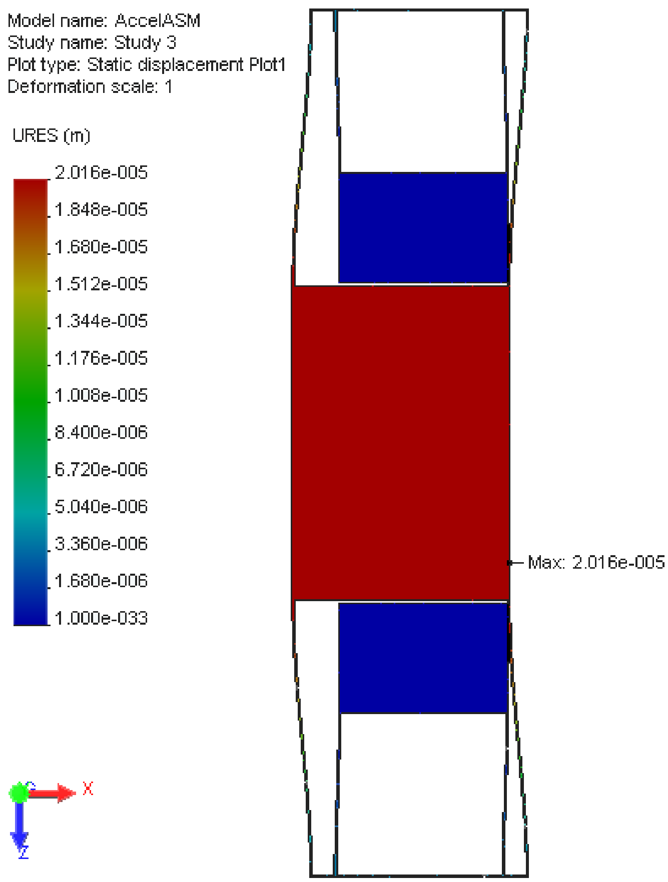


Figure 11: Proof Mass Displacement at Maximum Possible Displacement.

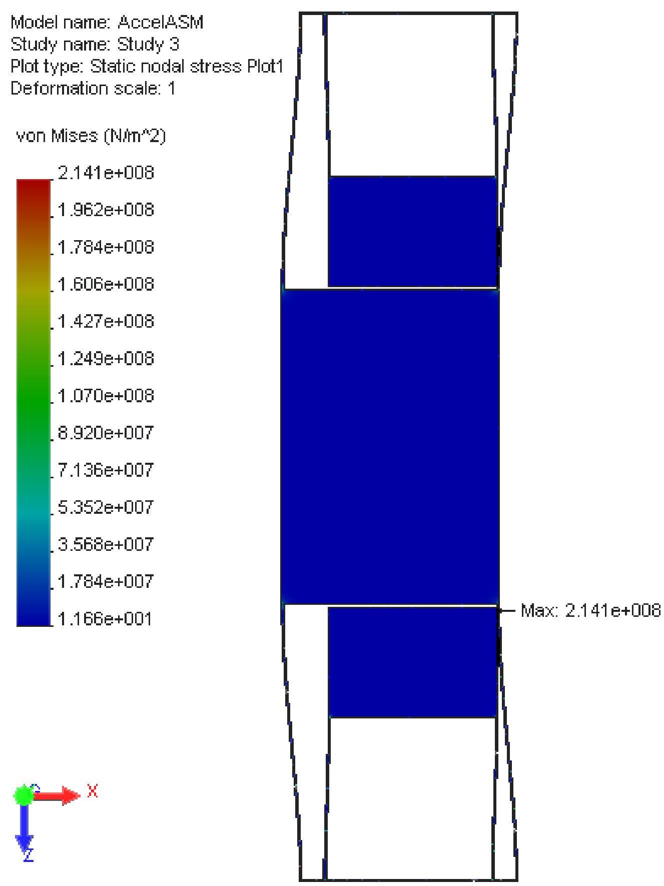


Figure 12: Maximum Stress at Maximum Possible Displacement.

Shown in Figure 13 is a detailed view of the Von Mises near the proof mass corner. This stress is the maximum stress that the accelerometer can possibly experience, which easily satisfies the maximum stress constraint of 1.6 GPa by a factor of 7.5.

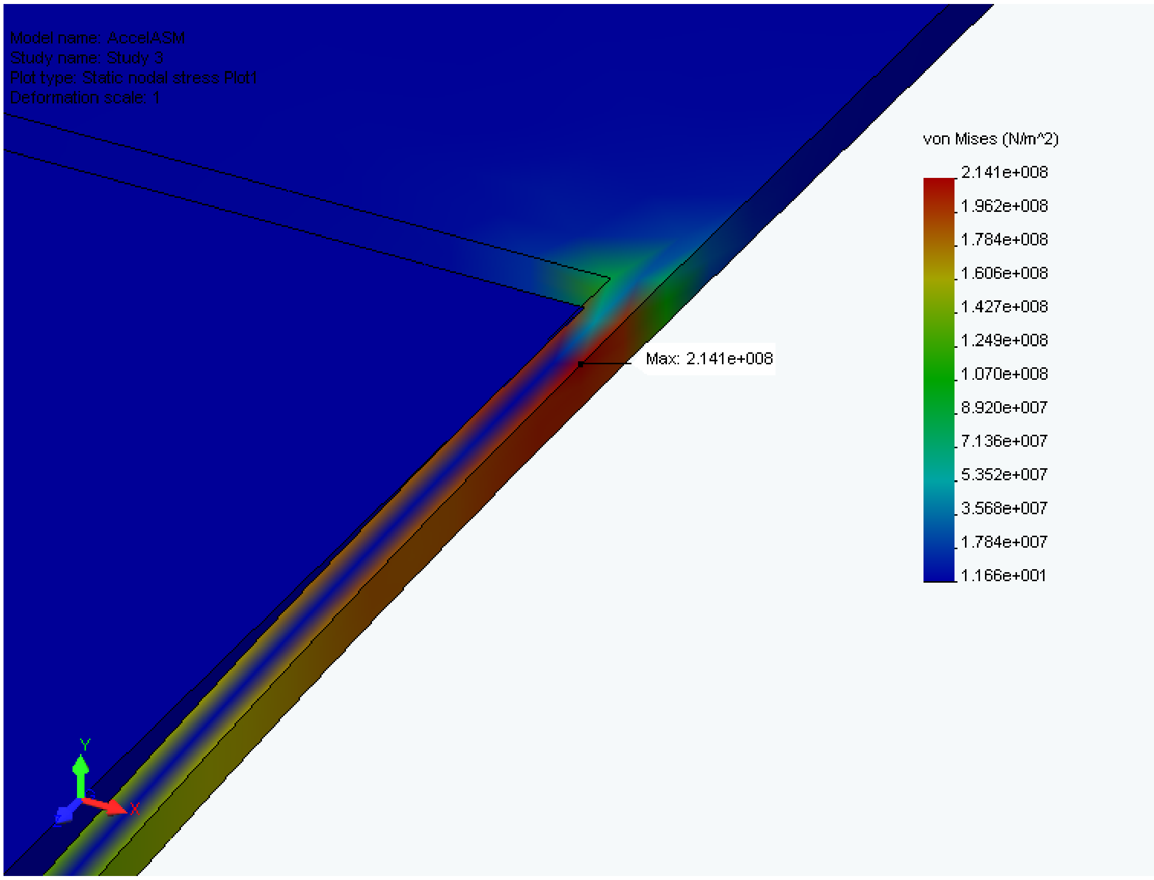


Figure 13: Detailed view of Maximum Stress at Maximum Possible Displacement.

Discussion & Conclusions

Theoretical Analysis vs. FEA

As described in the results, the MATLAB program agrees very well with the hand derived calculations. Both methods also agree with FEA for displacement and stress distributions. The results from both the theoretically driven genetic algorithm and finite element analyses are provided in Table 6. Clearly, both methods provide very close results that help verify each method and satisfy the engineering constraints and goals.

It should be noted that the FEM analysis was performed with the default tetrahedral global mesh size of 0.6 µm. This value is the maximum size able to accurately mesh the smallest feature of the device. As a result, controls meshes and/or finer meshes are unnecessary as the standard 0.6 µm proves to be more than adequate.

Table 6: GA and FEM Results Comparison.

Analysis Method	GA	FEM
Case 1 – Sensor Motion Resolution		
Displacement (µm)	3.518×10^{-5}	3.546×10^{-5}
Stress (GPa)	3.81×10^{-7}	3.766×10^{-7}
Case 2 – Acceleration Survival		
Displacement (µm)	14.072	14.18
Stress (GPa)	0.1524	0.1506
Case 3 – Maximum Displacement		
Displacement (µm)	20 µm	20.16 µm
Stress(GPa)	0.2166 GPa	0.2141 GPa

Another notable difference between the two methods is that the theoretical method ignored the mass of the beams, whereas the FEA does account for this additional mass. Although this is a critical assumption for developing the Euler-Beam equations, it appears that the proof mass dwarfs the mass of the beams. As a result, assuming that the beams' masses are negligible is verified.

For additional quantifiable verification of the error between the two methods, Table 7 is provided below. For all cases the error difference never exceeds ±1.2%, a remarkably accurate result.

Table 7: GA and FEM Results Comparison with Percentage Difference

	Theoretical Value	Percent Difference
Analysis Method	**GA**	**FEM**
Case 1 – Sensor Motion Resolution		
Displacement	3.518×10^{-5} µm	0.7959 %
Stress	3.81×10^{-7} GPa	-1.155 %
Case 2 – Acceleration Survival		
Displacement	14.072 µm	0.7675 %
Stress	0.1524 GPa	-1.181 %
Case 3 – Maximum Displacement		
Displacement	20 µm	0.8 %
Stress	0.2166 GPa	-1.154 %

Error due to Dimension Magnitude

Another minor source of error in both methods is round-off and truncation error. The theoretical calculations were all performed in MKS units and therefore variable values existed between a range of 10^9 to 10^{-9}. This broad range of numbers could easily lead to calculation errors. In contrast, COSMOS calculations were performed in the µMKS unit system. These calculations were characterized by a much smaller range of values, so round-off and truncation error is less likely.

Anchor/Proof Mass Clearance

In an effort to minimize the die area and maximize the resolution, the genetic algorithm suggested legs as long as possible (softest springs), and a mass as large as possible. This fact leads to a minimal clearance between the anchor and mass. Although a structure characterized by these dimensions will produce the most sensitive accelerometer possible, this small value may be an unacceptable distance for the clearance between a moving and stationary part. Although the analysis performed for this project only considered beam bending, the beam legs may also be susceptible to buckling, thus allowing the mass to crash into the anchor. This situation could occur if the device was dropped, thus creating damage. However, if the accelerometer is guaranteed to only encounter accelerations in one-direction, this is of no concern. The design proposed in this report has a clearance of only 2.3 µm between the anchor and mass. However, if a greater clearance is required, the value of L_1 can be decreased without a significant difference in resolution and no difference in die size.

Die Area and Acceleration Resolution

Initial optimization attempts revealed that satisfying all of the constraints and goals is nearly impossible with a maximum die size of only 90,000 μm^2. As a result, this constraint was relaxed and the optimization method attempted to minimize the area. Without modifying the acceleration resolution, it is possible to satisfy all of the constraints and engineering goals. However, the resulting design will produce a die area of at least 240,000 μm^2. In comparison, increasing the acceleration resolution by a factor of ten produced a minimum die area of 154,979.48 μm^2. This is a decrease in die area by a factor of over 1.5, which is directly related to a decrease in cost by a factor of 1.5. Additionally, background research for most accelerometer applications proved that a resolution of 0.005g was more than adequate. Therefore, it appears preferable to relax the resolution constraint in favor of increasing cost savings, which should more than offset the increased resolution in terms of significance to the manufacturer/business.

ANSYS Complications

It was my original intention to provide a detailed analysis of the accelerometer design using ANSYS to augment the existing FEA performed with COSMOS. In fact, I spent a significant portion of my Spring Break in Etcheverry working with

ANSYS. However, I was not successful in obtaining meaningful results that approached values obtained theoretically and through COSMOS. I believe my problem had to do with the units input for the material properties and accelerations. First, I created a model in SolidWorks with the µMKS units as suggested by the professor's e-mail. After importing the IGES file into ANSYS and using µMKS units for the material properties and accelerations, I received results that were off by 6 orders of magnitude. The GSI had told me that there were errors for exporting µMKS models and suggested that I use MMKS. I did as she said and used MMKS units for the material properties and accelerations, but still obtain unsatisfactory results. The plots for displacement and stress for an acceleration of 2000g are shown in Figures 14 and 15, respectively. These results are the best I could obtain with the limited ANSYS training we received. As a result, it is my recommendation that more lectures are dedicated to ANSYS training for future classes, as I obviously did not have the skills required to perform analysis on a MEMS scale device.

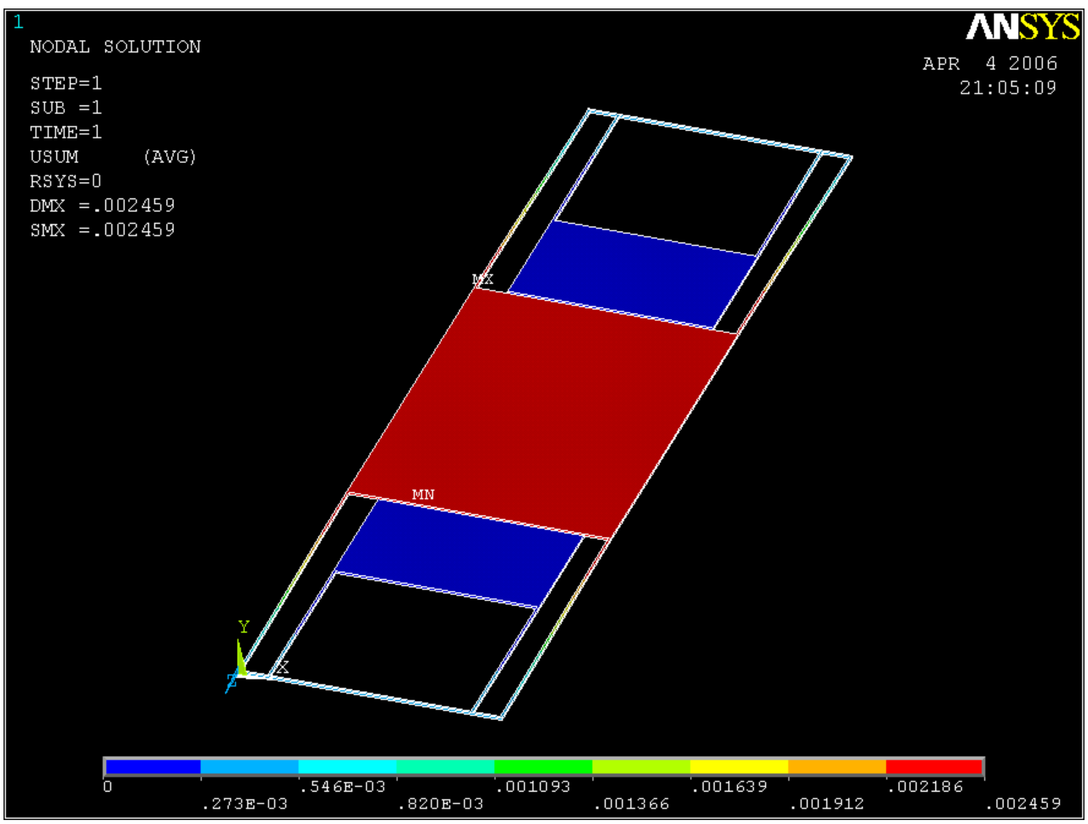


Figure 14: ANSYS Results for Displacement at 2000g.

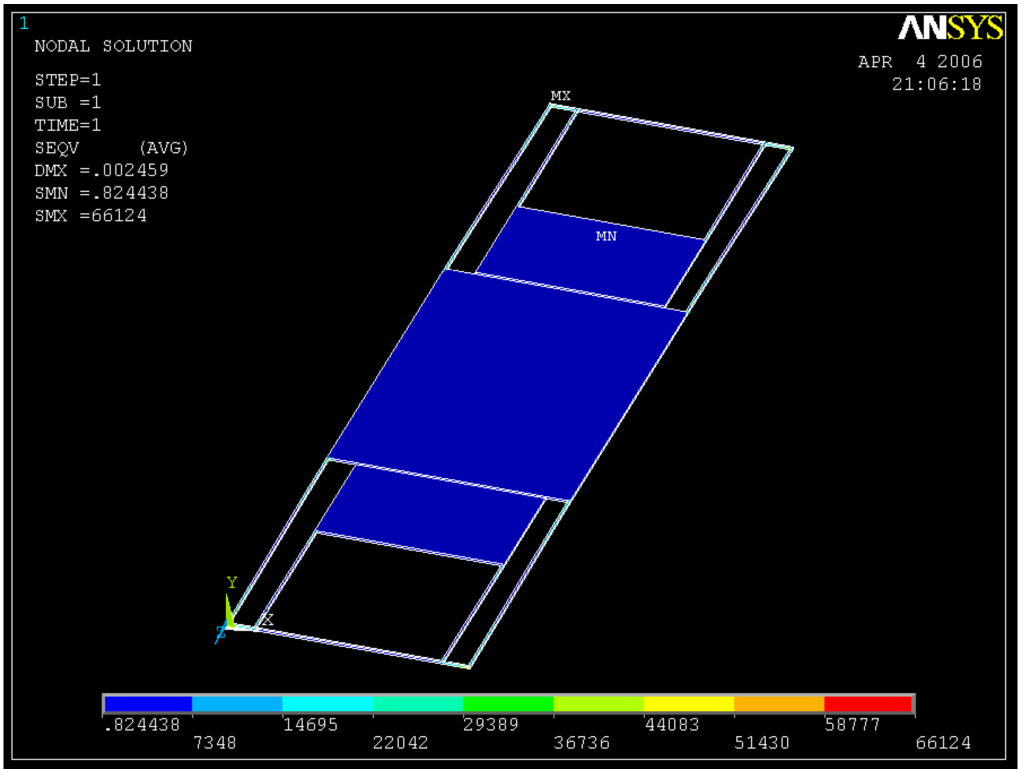


Figure 15: ANSYS Results for Stress at 2000g.

Summary Conclusions

Using a theoretically derived genetic algorithm for optimization and FEM analysis, it is concluded that satisfying the die area requirement is impossible. Therefore, this constraint is relaxed and minimized by the software. In addition, the acceleration resolution is relaxed in favor of decreasing the die area, which improves the cost per unit ratio. As a result of the theoretical and FEM analyses, an accelerometer with the following dimensions is recommended:

$$L_1 = 146.0\ \mu m$$

$$L_2 = 20.0\ \mu m$$

$$L_3 = 248.3\ \mu m$$

$$y_m = 285.7\ \mu m$$

This design meets the resolution criteria and exceeds the maximum stress criteria. In addition, the die area is exactly 154,979.48 μm^2, which is a 1.722 times increase above the maximum die area given. Both methods prove that this design is the most cost efficient while simultaneously satisfying all of the required engineering goals and constraints.

```
%% Project #2 - Parametric Design of a MEMS Accelerometer
% Scott Moura
% SID 15905638
% ME 128, Prof. Lin
% Due Wed April 5, 2006

%% Genetic Algorithm
clear

%% Define Equality Constraints
h_1 = 1.8e-6;           % Beam Width [m]
h_2 = 1.8e-6;           % Minimum Feature Length of Device
h_3 = 1.8e-6;
h_b = 1.8e-6;

b_1 = 1.8e-6;           % Beam Thickness [m]
b_2 = 1.8e-6;           % Minimum Feature Length of Device
b_3 = 1.8e-6;
b_m = 1.8e-6;

L_b = 150e-6;           % Central Beam Width [m]

E = 160e9;              % Elastic Modulus [Pa]
I = h_1*b_1^3/12;       % Area Moment of Inertia [m^4]
rho = 2.33*100^3/1000;  % Density [kg/m^3] (converted from gm/cm^3)
g = 9.8;                % Gravitational Acceleration [m/s^2]

y_a = 100e-6;           % Anchor Height [m]
x_a = L_b + 2*h_1;      % Anchor Width [m]

A_max = 250000/(1e6)^2; % Maximum Die Area [m^2]

%% Define Inequality Constraint Limits
L_min = 20e-6;         % Minimum beam length
a_r_min = 0.005*g;     % Accerlation at minimum required resolution [m/s^2]
a_stress = 2000*g;     % Acceleration at maximum required stress [m/s^2]
sigma_max = 1.6e9;     % Maximum stress [Pa]

%% Define boundary limits
L_1_min = L_min;
L_1_max = 500e-6;
L_2_min = L_min;
L_2_max = 100e-6;
L_3_min = 100e-6;
L_3_max = 500e-6;
y_m_min = 100e-6;
y_m_max = 500e-6;

% Initialize best1 & best2
start_pops = 1000;
iters = 50;
best1 = zeros(iters,start_pops);
best2 = zeros(1,iters);

% Loop for 1000 different starting populations
for idx1 = 1:start_pops
```

```
    % Define Set of Random Strings
    d = 100;                  % Number of strings
    clear Lambda
    random = rand(d,4);
    Lambda(:,1) = L_1_min + random(:,1) * (L_1_max - L_1_min);  % L_1
    Lambda(:,2) = L_2_min + random(:,2) * (L_2_max - L_2_min);  % L_2
    Lambda(:,3) = L_3_min + random(:,3) * (L_3_max - L_3_min);  % L_3
    Lambda(:,4) = y_m_min + random(:,4) * (y_m_max - y_m_min);  % y_m

    % Repeat Genetic Algorithm for 50 iterations
    for idx2 = 1:iters

        L_1 = Lambda(:,1);
        L_2 = Lambda(:,2);
        L_3 = Lambda(:,3);
        y_m = Lambda(:,4);

        % Calculate design variable dependent parameters
        x_m = L_b + 2*L_2 + 2*h_1 + 2*h_3;   % Proof mass width [m]
        A = x_m .* (y_m + 2*L_3 + 2*h_2);    % Die Area [m^2]
        m = rho * x_m .* y_m * b_m;          % Mass of system (ignore beams) [kg]
        s_res = (1e-7)^2 ./ y_m;             % Motion resolution [m]

        % System Spring Constant [N/m]
        k = 24*E*I * (6*L_1.*L_2.^2 + L_b*L_2.^2 + 4*L_b*L_1.*L_2 + 24*L_1.*L_2.*L_3 + ...
            12*L_b*L_1.*L_3 + 4*L_b*L_2.*L_3) ./ LOCALden(L_1,L_2,L_3,L_b);

        a_res = (k ./ m) .* s_res;           % Minimum resolvable acceleration [m/s^2]
        s_stress = (m ./ k) .* a_stress;     % Displacement at maximum survival accel [m]

        % Moment at max survival accel [N*m]
        M = 6*s_stress*E*I .* (6*L_b*L_1.*L_3.^2 + 12*L_1.*L_2.*L_3.^2 + ...
            2*L_b*L_2.*L_3.^2 + 6*L_1.*L_2.^2.*L_3 + L_b*L_2.^2.*L_3 + ...
            4*L_b*L_1.*L_2.*L_3 + L_b*L_1.^2.*L_2) ./ LOCALden(L_1,L_2,L_3,L_b);

        % Forces at max survival accel [N]
        F_x = (k .* s_stress) / 4;
        F_y = 18*s_stress*E*I ./ L_2 .* (-L_b*L_1.^2.*L_2 + 2*L_b*L_1.*L_3.^2 - ...
            2*L_b*L_1.^2.*L_3 + 6*L_1.*L_2.*L_3.^2 + L_b*L_2.*L_3.^2) ./ ...
            LOCALden(L_1,L_2,L_3,L_b);

        % Stresses at max survival accel [N/m^2]
        sigma_x = F_x / (h_1*b_1);
        sigma_y = F_y / (h_1*b_1);
        sigma_bend = M * (b_1/2) / I;

        % Calculate the Fitness of the Objective Function
        Pi = A*1e12;

        % Check Constraints
        for idx3 = 1:d

            % Anchor cannot petrude through proof mass
            if (L_3(idx3) < h_2 + y_a + L_1(idx3))
                Pi(idx3) = inf;
            end
```

```
            % Check Die Area
            if (A(idx3) > A_max)
                Pi(idx3) = inf;
            end

            % Check minimum motion resolution
            if (a_res(idx3) > a_r_min)
                Pi(idx3) = inf;
            end

            % Check maximum stress
            sigma_norm = norm([sigma_x(idx3) sigma_y(idx3) sigma_bend(idx3)]);
            if (sigma_norm > sigma_max)
                Pi(idx3) = inf;
            end

            % Check if beams crash into anchor
            if (s_stress(idx3) > L_2(idx2))
                %disp(s_stress(idx3))
                Pi(idx3) = inf;
            end

        end

        % Rank the Genetic Strings output by Pi and save the best
        [sorted_Pi, order_Pi] = sort(Pi);
        if (sorted_Pi(1) == inf)
            %disp('INFINITY')
        end
        best1(idx2,idx1) = sorted_Pi(1);

        % Keep the Ten Best Parents
        nps = 10; % Number of Parents
        for i = 1:nps
            parent(i,:) = Lambda(order_Pi(i),:);
        end

        % Mate the Top Ten Parents to Generate Ten Children
        random2 = rand(nps,1);
        for j = 1:2:nps-1
            child(j,:) = random2(j,1) * parent(j,:) + (1 - random2(j,1)) * parent(j+1,:);
            child(j+1,:) = random2(j+1,1) * parent(j,:) + (1 - random2(j+1,1)) * parent(j ↙
+1,:);
        end

        % Generate New Random Strings to Combine with Parents and Children
        clear Lambda_new
        random3 = rand(d-2*nps, 4);

        Lambda_newrand(:,1) = L_1_min + random3(:,1) * (L_1_max - L_1_min);  % new L_1
        Lambda_newrand(:,2) = L_2_min + random3(:,2) * (L_2_max - L_2_min);  % new L_2
        Lambda_newrand(:,3) = L_3_min + random3(:,3) * (L_3_max - L_3_min);  % new L_3
        Lambda_newrand(:,4) = y_m_min + random3(:,4) * (y_m_max - y_m_min);  % new y_m

        Lambda_new = vertcat(parent, child, Lambda_newrand);

        % Repeat Genetic Algorithm
```

```
        Lambda = Lambda_new;

    end

    % Save the best lambda values
    Lambda_final(idx1,:) = Lambda(1,:);

end

% Determine the runs that had the six best performing
best2 = best1(iters,:);
[sorted_best2, order_best2] = sort(best2);

% Output the design variable values for top five
for k = 1:5
    best_Lambda(k,:) = Lambda_final(order_best2(k),:);
    fprintf('Rank: %2.0f   L_1 = %6.3e   L_2 = %6.3e   L_3 = %6.3e   y_m = %6.3e   Pi = % ↙
6.5f\n',...
        k,best_Lambda(k,1),best_Lambda(k,2),best_Lambda(k,3),best_Lambda(k,4),sorted_best ↙
2(k));
end

% Plot the minimum objective value as a function of generations
for idx4 = 1:100
    if (idx4 == 1)
        Pi_min(idx4) = best2(idx4);
    elseif (best2(idx4) <= Pi_min(idx4-1))
        Pi_min(idx4) = best2(idx4);
    else
        Pi_min(idx4) = Pi_min(idx4-1);
    end
end

plot(1:100,Pi_min(1:100))
title(['\bfMinimization of Objective Function: \Pi_{\it{min}} = ' num2str(sorted_best2(1) ↙
)])
xlabel('Number of Starting Populations (First 100 Only)')
ylabel('Minimum Objective Value Found')
```

```
function output = enginprop(L_1,L_2,L_3,y_m)

%% Project #2 - Parametric Design of a MEMS Accelerometer
% Scott Moura
% SID 15905638
% ME 128, Prof. Lin
% Due Wed April 5, 2006

%% Calculate Engineering Properties

%% Define Equality Constraints
h_1 = 1.8e-6;           % Beam Width [m]
h_2 = 1.8e-6;           % Minimum Feature Length of Device
h_3 = 1.8e-6;
h_b = 1.8e-6;

b_1 = 1.8e-6;           % Beam Thickness [m]
b_2 = 1.8e-6;           % Minimum Feature Length of Device
b_3 = 1.8e-6;
b_m = 1.8e-6;

L_b = 150e-6;           % Central Beam Width [m]

E = 160e9;              % Elastic Modulus [Pa]
I = h_1*b_1^3/12;       % Area Moment of Inertia [m^4]
rho = 2.33*100^3/1000;  % Density [kg/m^3] (converted from gm/cm^3)
g = 9.8;                % Gravitational Acceleration [m/s^2]

y_a = 100e-6;           % Anchor Height [m]
x_a = L_b + 2*h_1;      % Anchor Width [m]

%% Define Inequality Constraint Limits
L_min = 20e-6;         % Minimum beam length
a_r_min = 0.005*g;     % Accerlation at minimum required resolution [m/s^2]
a_stress = 2000*g;     % Acceleration at maximum required stress [m/s^2]
sigma_max = 1.6e9;     % Maximum stress [Pa]

% Calculate design variable dependent parameters
x_m = L_b + 2*L_2 + 2*h_1 + 2*h_3;   % Proof mass width [m]
A = x_m .* (y_m + 2*L_3 + 2*h_2);    % Die Area [m^2]
zz = y_m + 2*L_3 + 2*h_2;            % Die length [m]
m = rho * x_m .* y_m * b_m;          % Mass of system (ignore beams) [kg]
s_res = (1e-7)^2 ./ y_m;             % Motion resolution [m]

% System Spring Constant [N/m]
k = 24*E*I * (6*L_1.*L_2.^2 + L_b*L_2.^2 + 4*L_b*L_1.*L_2 + 24*L_1.*L_2.*L_3 + ...
    12*L_b*L_1.*L_3 + 4*L_b*L_2.*L_3) ./ LOCALden(L_1,L_2,L_3,L_b);

a_res = (k ./ m) .* s_res;           % Minimum resolvable acceleration [m/s^2]
s_stress = (m ./ k) .* a_stress;     % Displacement at maximum survival accel [m]

% Moment at max survival accel [N*m]
M = 6*s_stress*E*I .* (6*L_b*L_1.*L_3.^2 + 12*L_1.*L_2.*L_3.^2 + ...
    2*L_b*L_2.*L_3.^2 + 6*L_1.*L_2.^2.*L_3 + L_b*L_2.^2.*L_3 + ...
    4*L_b*L_1.*L_2.*L_3 + L_b*L_1.^2.*L_2) ./ LOCALden(L_1,L_2,L_3,L_b);

% Forces at max survival accel [N]
```

```
F_x = (k .* s_stress) / 4;
F_y = 18*s_stress*E*I ./ L_2 .* (-L_b*L_1.^2.*L_2 + 2*L_b*L_1.*L_3.^2 - ...
    2*L_b*L_1.^2.*L_3 + 6*L_1.*L_2.*L_3.^2 + L_b*L_2.*L_3.^2) ./ ...
    LOCALden(L_1,L_2,L_3,L_b);

% Stresses at max survival accel [N/m^2]
sigma_x = F_x / (h_1*b_1);
sigma_y = F_y / (h_1*b_1);
sigma_bend = M * (b_1/2) / I;
sigma = norm([sigma_x sigma_y sigma_bend]);

% Find maximum survivable acceleration (assume only bending stress)
M_surv = sigma_max * I / (b_1/2);
s_surv = M_surv / (6*E*I .* (6*L_b*L_1.*L_3.^2 + 12*L_1.*L_2.*L_3.^2 + ...
    2*L_b*L_2.*L_3.^2 + 6*L_1.*L_2.^2.*L_3 + L_b*L_2.^2.*L_3 + ...
    4*L_b*L_1.*L_2.*L_3 + L_b*L_1.^2.*L_2) ./ LOCALden(L_1,L_2,L_3,L_b));
a_surv = (k ./ m) .* s_surv;        % Maximum survivable acceleration

% Find displacements at 0.005g and 2000g
s_sm = m/k * 0.005*g;
s_lg = m/k * 2000*g;

% Find stress at displacement of 20 um
s_20 = 20e-6;
a_20 = (k ./ m) .* s_20;
M_20 = 6*s_20*E*I .* (6*L_b*L_1.*L_3.^2 + 12*L_1.*L_2.*L_3.^2 + ...
    2*L_b*L_2.*L_3.^2 + 6*L_1.*L_2.^2.*L_3 + L_b*L_2.^2.*L_3 + ...
    4*L_b*L_1.*L_2.*L_3 + L_b*L_1.^2.*L_2) ./ LOCALden(L_1,L_2,L_3,L_b);
F_x_20 = (k .* s_20) / 4;
F_y_20 = 18*s_20*E*I ./ L_2 .* (-L_b*L_1.^2.*L_2 + 2*L_b*L_1.*L_3.^2 - ...
    2*L_b*L_1.^2.*L_3 + 6*L_1.*L_2.*L_3.^2 + L_b*L_2.*L_3.^2) ./ ...
    LOCALden(L_1,L_2,L_3,L_b);
sigma_x_20 = F_x_20 / (h_1*b_1);
sigma_y_20 = F_y_20 / (h_1*b_1);
sigma_bend_20 = M_20 * (b_1/2) / I;
sigma_20 = norm([sigma_x_20 sigma_y_20 sigma_bend_20]);

% Find stress at 0.005g
M_sm = 6*s_sm*E*I .* (6*L_b*L_1.*L_3.^2 + 12*L_1.*L_2.*L_3.^2 + ...
    2*L_b*L_2.*L_3.^2 + 6*L_1.*L_2.^2.*L_3 + L_b*L_2.^2.*L_3 + ...
    4*L_b*L_1.*L_2.*L_3 + L_b*L_1.^2.*L_2) ./ LOCALden(L_1,L_2,L_3,L_b);
F_x_sm = (k .* s_sm) / 4;
F_y_sm = 18*s_sm*E*I ./ L_2 .* (-L_b*L_1.^2.*L_2 + 2*L_b*L_1.*L_3.^2 - ...
    2*L_b*L_1.^2.*L_3 + 6*L_1.*L_2.*L_3.^2 + L_b*L_2.*L_3.^2) ./ ...
    LOCALden(L_1,L_2,L_3,L_b);
sigma_x_sm = F_x_sm / (h_1*b_1);
sigma_y_sm = F_y_sm / (h_1*b_1);
sigma_bend_sm = M_sm * (b_1/2) / I;
sigma_sm = norm([sigma_x_sm sigma_y_sm sigma_bend_sm]);

%% Display Engineering Properties
disp(['L_1:                 ' num2str(L_1*1e6) ' um'])
disp(['L_2:                 ' num2str(L_2*1e6) ' um'])
disp(['L_3:                 ' num2str(L_3*1e6) ' um'])
disp(['y_m:                 ' num2str(y_m*1e6) ' um'])
disp(['Die Area:            ' num2str(A*1e12) ' um^2'])
disp(['Die Dimensions:      ' num2str(x_m*1e6) ' um x ' num2str(zz*1e6) ' um'])
```

```
disp(['Proof Mass:           ' num2str(m*1e9) ' ug'])
disp(['Spring Constant:      ' num2str(k) ' N/m'])
disp(['Min Accel Resolution: ' num2str(a_res/g) 'g'])
disp(['Motion Resolution:    ' num2str(s_res*1e6) ' um'])
disp(['Disp @ 0.005g:        ' num2str(s_sm*1e6) ' um'])
disp(['Stress @ 0.005g:      ' num2str(sigma_sm/1e9) ' GPa'])
disp(['Maximum Stress:       ' num2str(sigma/1e9) ' GPA'])
disp(['Disp @ 2000g:         ' num2str(s_lg*1e6) ' um'])
disp(['Max Survival Accel:   ' num2str(a_surv/g) 'g'])
disp(['Disp @ ' num2str(a_surv/g) 'g:     ' num2str(s_surv*1e6) ' um'])
disp(['Accel. @ 20 um disp:  ' num2str(a_20/g) 'g'])
disp(['Stress @ 20 um disp:  ' num2str(sigma_20/1e9) ' GPa'])
```

```
function output = LOCALden(L_1,L_2,L_3,L_b)
output = (3*L_1.^4.*L_2.^2 + 12*L_1.*L_2.^2.*L_3.^3 + ...
        2*L_b*L_1.^3.*L_2.^2 + 2*L_b*L_1.^4.*L_2 + 12*L_1.^4.*L_2.*L_3 + ...
        12*L_1.*L_2.*L_3.^4 + 6*L_b.*L_1.^4.*L_3 + 6*L_b*L_1.*L_3.^4 + ...
        2*L_b*L_2.^2.*L_3.^3 + 6*L_b.*L_2.*L_1.^2.*L_3.^2 + 8*L_b*L_1.*L_2.*L_3.^3 + ...
        2*L_b*L_2.*L_3.^4 + 8*L_b*L_1.^3.*L_2.*L_3);
```