UNIVERSITY OF CALIFORNIA,

IRVINE

Model fitting of a bilinear material with Genetic Algorithm

THESIS

Submitted in partial satisfaction of the requirements

for the degree of

MASTER OF SCIENCE

In Civil Engineering

by

Yair Marc Martínez Palomino

Thesis Committee:

Professor Farzin Zareian, Chair

Professor Ayman Mosallam

Pofessor Lizhi Sun

2011

TABLE OF CONTENTS

LIST OF TABLES iv
LIST OF GRAPHICS vi
LIST OF IMAGES vii
ACKNOWLEDGEMENTS viii
ABSTRACT OF THE THESIS ix
CHAPTER 1: INTRODUCTION 1
1.1 MOTIVATION AND BACKGROUND 1
1.2 OBJECTIVES 2
CHAPTER 2: OPENSEES 3
2.1 INTRODUCTION TO OPENSEES 3
2.2 OBJECTIVES OF THE FIRST PART OF THE STUDY 3
2.3 CANTILIVER DESIGN IN OPENSEES 3
2.3.1 STEEL 01 5
2.3.1 STEEL 02 6
2.4 FITTING CURVES 8
2.4.1 LEAST SQUARES 10
2.4.1 FIRST FITTING MODEL 10
2.4.2 CODE IMPROVAL FOR 1 PARAMETER 12
2.4.3 MULTIVARIABLE BISECTION METHOD 16
2.4.4 RESULTS OF THE MULTIVARIABLE PROGRAM WITH THREE PARAMETERS 26
CHAPTER 3: GENETIC ALGORITHM 31
3.1 INTRODUCTION TO GENETIC ALGORITHMS 31
3.2 APPLICATION TO THE FITTING PROBLEM 36
3.3 RESULTS 37
CHAPTER 4: UNIAXIAL BILINEAR MATERIAL 39
4.1 INTRODUCTION AND EXPLANATION OF OPENSEES UNIAXIAL BILINEAR MATERIAL 39
4.2 EXPERIMENTAL DATA USED TO CALIBRATE THE MODEL 42

4.3 DATA FILTERING 43
4.4 OPENSEES MODEL 45
4.5 PARAMETERS IN UNIAXIAL BILINEAR MATERIAL 46
4.6 STUDY OF THE PARAMETERS 50
4.7 FINAL PARAMETERS SUMMARY TABLE 60
4.8 GENETIC ALGORITHM FITTING FOR UNIAXIAL BILINEAR MATERIAL 61
4.9 RESULTS 62
4.9.1 EXPERIMENT #1 62
4.9.2 EXPERIMENT #2 65
4.9.3 EXPERIMENT #4 67
4.9.2 EXPERIMENT #4 68
CHAPTER 5: FINDINGS, CONCLUSIONS AND FUTURE WORK 70
BIBLIOGRAPHY 72
APPENDIX 73
1 Code for the cantilever designed with steel01 73
2 Code for the cantilever designed with steel02: 74
3 Matlab Code for the first fitting model: 76
4 Matlab code for the three parameters fitting problem: 77
5- Matlab code for the objective function of the 3 parameters fitting using GA: .. 99
6 Matlab code for the frequency analysis: 101
7 Matlab code that filters the data 101
8 OpenSees TCL code defining the model 102
9 OpenSees code for the load of the displacement control of the experimental data and the analysis 103
10 Matlab code to study the effect in the value of the parameters 104
11 Matlab code for the GA objective function after having studied the different parameters 106

LIST OF TABLES

TABLE 1: Comparison of the behavior of the cycle for different K0s. At the left K0=186 and at the right K0=46 52
TABLE 2: Difference in the hardening ratio. With a value of 0.001 at the left and a value of 0.3 at the right 52
TABLE 3: Comparison of hysteretic cycle with different My. At the left My=40 PSI. At the right My=200 PSI 54
TABLE 4: Comparison of cyclic deterioration parameter for strength deterioration. At the left, model with lambda S=100 and at the right lambda S=2000 54
TABLE 5: Comparison of cyclic deterioration parameter for post-capping strength deterioration. At the left lambda C=100 and the right Lambda C=1000 55
TABLE 6:Comparison of cyclic deterioration parameter for unloading stiffness deterioration. At the left, lambda k=100 at the right lambda K=1000 56
TABLE 7: Comparison of rate of strength deterioration. At the left C_S=0.6 at the right C_S=1.0 56
TABLE 8: Comparison of rate of post-capping strength deterioration.At the left c_C=0.99 at the right c_C=1 57
TABLE 9: Comparison of rate of unloading stiffness deterioration. At the left c_K=0.5 and at the right c_K=10. 57
TABLE 10: Comparison of pre-capping rotation. At the left θp=1.0 and at the right θp=5.0 58
TABLE 11: Comparison of post-capping rotation. At the left θpc=2.0 and at the right θpc=6.0 58
TABLE 12: Comparison of rate of cyclic deterioration. At the left D=1.0 at the right D=30.0 59
TABLE 13: Summary table of all variables of the model and it's characteristics .. 60
TABLE 14: Table that defines each component of the vector plugged into the GA…… 62

TABLE 15: Different moments representing different populations of the GA converging to the best solution..... 63
TABLE 16: Final results for experiment #1 after applying the GA..... 64
TABLE 17: Final results for experiment #2..... 67
TABLE 17: Final results for experiment #3..... 68
TABLE 17: Final results for experiment #4..... 69

LIST OF GRAPHICS

GRAPHIC 1: displacement control applied on the top of the cantiliver 4
GRAPHIC 2: definition of steel 01 from opensees website 5
GRAPHIC 3: graphic force-deformation for model designed with steel 01 6
GRAPHIC 4: definition of steel 02 by opensees website 7
GRAPHIC 5: displacement-force graphic for steel 02 8
GRAPHIC 6: steel 01 model behavior 20
GRAPHIC 7: tri-dimensional domain of the steel 01 parameters. 23
GRAPHIC 8: time that takes the program to run depending on the number of iterations 27
GRAPHIC 9: error in function of the number of iterations 28
GRAPHIC 10: total accumulated error. 29
GRAPHIC 11: fitting curve using multivariable bisection method. 30
GRAPHIC 12: fitting curve using ga 37
GRAPHIC 13: comparison of ga and multivariable methods. 38
GRAPHIC 14: bilin material behavior 40
GRAPHIC 15: initial experimental data 43
GRAPHIC 16: frequency analysis 44
GRAPHIC 17: filtered experimental data 45
GRAPHIC 18: displacement control in inches of the experiment 46
GRAPHIC 19: backbone curve 48
GRAPHIC 20: effect of the deterioration after cyclic loading 49
GRAPHIC 21: determination of elastic stiffness. 51
GRAPHIC 22: determination of my 53
GRAPHIC 23: final fitting to the experimental test after applying the ga 64
GRAPHIC 24: initial data of the second experiment 65
GRAPHIC 25: data of experiment #2 after filtering 66
GRAPHIC 26: fitting curve to experiment #2 67
GRAPHIC 27: fitting curve to experiment #3 68
GRAPHIC 28: fitting curve to experiment #4 69

LIST OF IMAGES

IMAGE 1: diagram of the modeled cantiliver……..…..4
IMAGE 2: diagram of the working of the code improval……..…..12
IMAGE 3: diagram of the initial status of the program……...……..……13
IMAGE 4: diagram of the next iteration……..……...…..14
IMAGE 5: boundaries of the different parameters……..…….....16
IMAGE 6: bidimensional diagram of the bisection method……..……..17
IMAGE 7: working diagram of the genetic algorithm…….....33

ACKNOWLEDGEMENTS

I was awarded with the Balsells Fellowship in the summer of 2010 in order to pursue a Master Science in Structural Engineering at University of California, Irvine. I would like to express my most sincere gratitude to Mr. Pere Balsells and the institutions involved in the Balsells Fellowship Program, such as La Generalitat de Catalunya and the University of California, Irvine for the economical support that has allowed me to study in this institution.

I am very grateful of having worked and having had for advisor the committee chair, Professor Farzin Zareian, for his quick help from the very beginning, even before I had arrived to this school, for his worthy advice and guidance and help to finish my thesis all this year long.

I would like to thank also the students of my laboratory, specially: Bahare, Peng, Pearson and Carmino for their help in any kind of problems I have found during the development of the thesis and their hardworking capacity and their companionship in the structures laboratory.

I would like to thank my friends from Barcelona and the ones I have spread all over the globe for their unconditional support and specially my new American friends that have become like a family to me during all this year.

Finally I would like to thank my parents and family for their unconditional help and enormous love.

ABSTRACT OF THE THESIS

Model fitting of a bilinear material with Genetic Algorithm

by

Yair Marc Martinez Palomino
Master Science of Civil Engineering
University of California, Irvine, 2011
Professor Farzin Zareian, Chair

This study is on the development of the technology that allows the estimation of the parameters that define the bilinear material defined in OpenSees software using a modern programming technique named Genetic Algorithm and also the structural program OpenSees.

In order to obtain the parameters of the modeled bilinear material, real experimental data provided by NEES (Network for Earthquake Engineering Simulation) has been used and then a model has been fitted to it using a Genetic Algorithm.

The thesis contains the treatment of the experimental data and the design of the procedure for the cancelation or at least diminishment of the noise of the set of data using Fourier Transformation and frequency analysis.

Finally the code that allows the Genetic Algorithm to find the parameters of the modeled bilinear material is programmed and successfully tested with the experimental results.

CHAPTER 1: INTRODUCTION

1.1 MOTIVATION AND BACKGROUND

Significant progress has been made in recent years in methods to predict collapse under earthquake loading (e.g., Ibarra et al 2002; Vamvatsikos and Cornell 2002; Ibarra and Krawinkler 2005; Haselton and Deirlein 2007; Zareian and Krawinkler 2009) and to develop engineering approaches for collapse protection (FEMA P695 2009, ATC-76-1 2009; Zareian et al. 2010). The collapse mode addressed in these studies is associated with sidesway instability in which P-Delta effects accelerated by component deterioration fully offset the first order story shear resistance and dynamic instability occurs. One of the main challenges has been, and still is, the ability to reliably predict deterioration properties of structural components and to incorporate these properties into analysis tools.

Reliable collapse assessment of structural systems under earthquake loading requires analytical models that are able to capture component deterioration in strength and stiffness of the material that is going to be used in the structure of the building, bridge or element. The quantification of important parameters that affect the cyclic force-displacement relationship in beams is crucial to the development of software capable to calculate structures under different loads scenarios such as earthquakes.

Experimental studies have shown that the hysteretic behavior of structural components depends upon numerous structural parameters that affect the deformation and energy dissipation characteristics, leading to the development of a wide range of versatile deterioration models. The most recently model, developed by Ibarra et al. (2005) is an energy based phenomenological deterioration model that captures most important modes of component deterioration.

Reliable deterioration modeling of structural components requires validation of analytical models described earlier with experimental data from components that have been subjected to various loading histories.

The purpose of the present thesis is to fit an already design model on the library of OpenSees software, developed at University of California, Berkeley. And obtain the values of the parameters, making the model to be the closest possible to reality.

1.2 OBJECTIVES

The first part of this thesis presents the learning project aiming to the development of the programming and use of software that allows the final fitting of the model of a bilinear material to the experimental results. Therefore this part consists in the understanding of Opensees software, the Matlab software, and the beginning of technics in order to adjust models to experimental curves and obtaining the parameters using other techniques different from the Genetic Algorithm

The second part of the thesis starts treating the bilinear material model understanding the different parameters of the model.

The third part treats the experimental data in order to be able to work and adjust a model. The treatment consists of a filtering of the noise of the data using Fast Fourier Transformation and a frequency analysis.

Once the experimental data is ready, the fourth part of the thesis consists on the development of the code that the Genetic Algorithm will use in order to find the parameters that adjust the model to the experimental curve and also the study of the behavior of the different parameters in how do they affect the model so that the calculation time is minimum.

The final part of the thesis consists in proving the code and adjusting the model to the filtered experimental data.

CHAPTER 2: OPENSEES

2.1 INTRODUCTION TO OPENSEES

OpenSees, the Open System for Earthquake Engineering Simulation, is an object-oriented, open source software framework created at the NSF and sponsored Pacific Earthquake Engineering (PEER) Center. It allows users to create finite element applications for simulating the response of structural and geotechnical systems subjected to earthquakes or other kinds of of loads. OpenSees is primarily written in C++ and uses several Fortran numerical libraries for linear equation solving.

Users of OpenSees create applications by writing scripts in the Tcl programming language with extended commands with direct application for OpenSees in element model building design and analysis.

The present chapter describes the first part of the thesis, the use of OpenSees and the first fitting model research in order to fit a model to an experiment.

2.2 OBJECTIVES OF THE FIRST PART OF THE STUDY

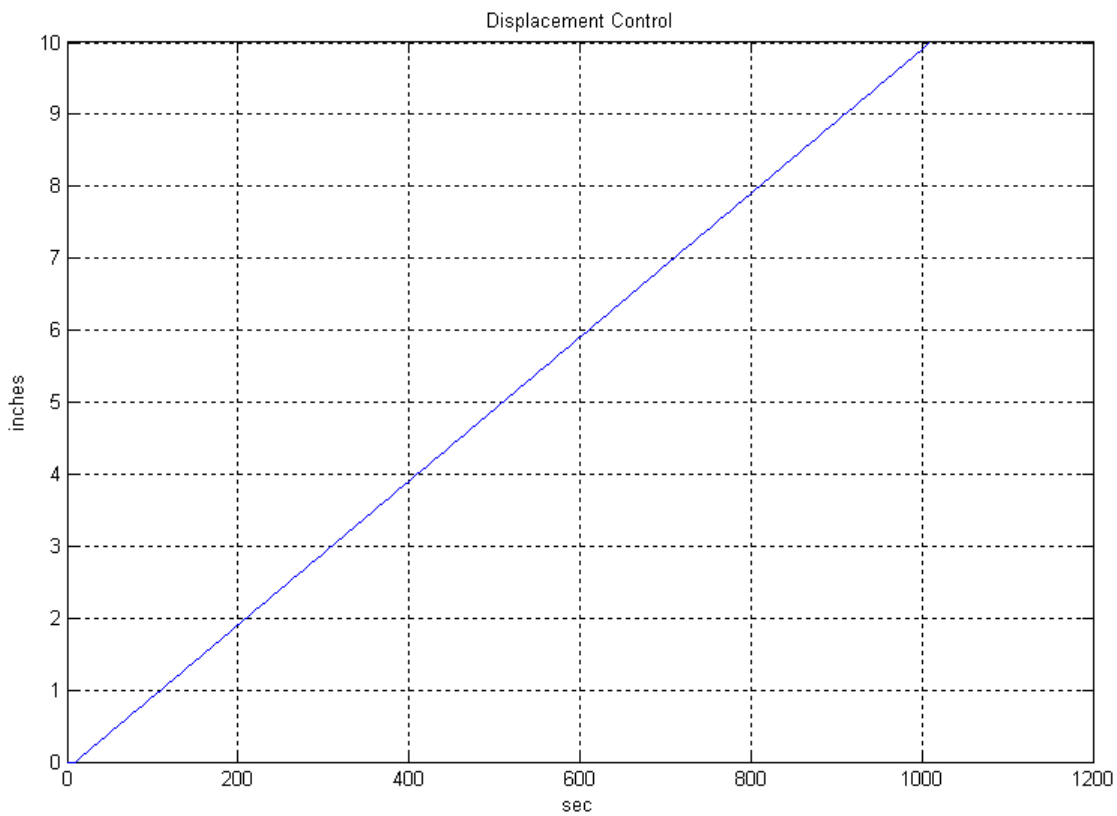
The main objective of the first part of the thesis is firstly to get familiarized with OpenSees software and secondly to start adjusting some easy models using at the beginning some coding that will be explained later on and afterwards Genetic Algorithm provided by Matlab.

2.3 CANTILIVER DESIGN IN OPENSEES

The first model that is going to be designed by OpenSees is going to be the behavior of a cantilever; a vertical column with one of the ends fixed and the other one free. The code for Opensees can be found in the appendix The model looks physically as follows:

Image 1: Diagram of the modeled cantiliver

Then, a linear displacement control is going to be applied at the top of the column, where the red arrow is located.

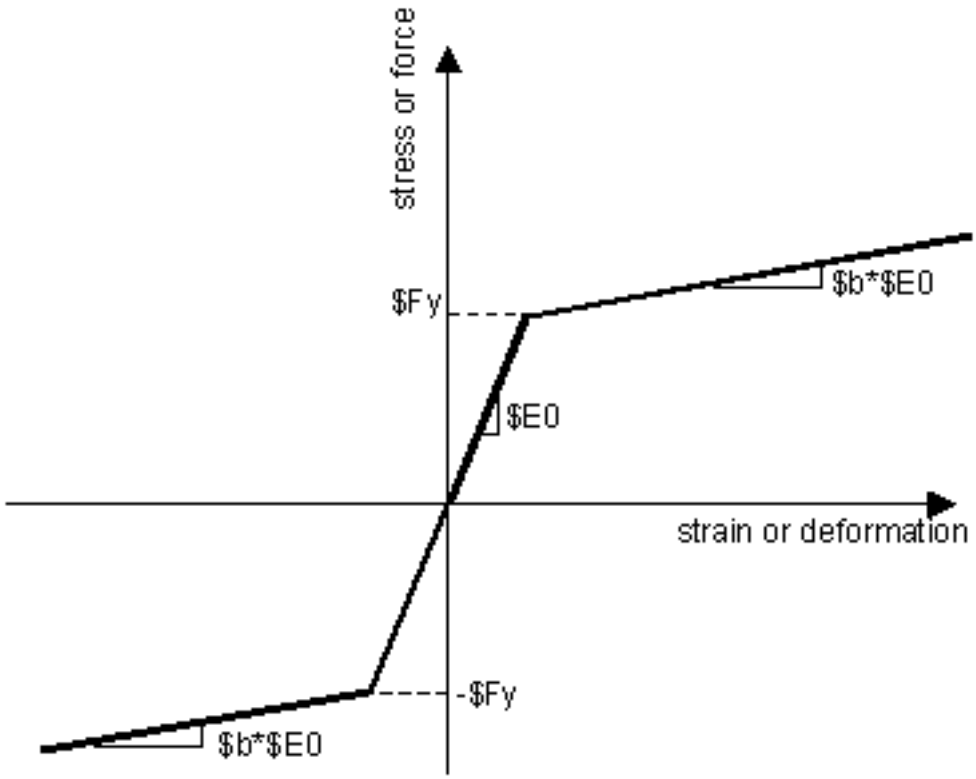


Graphic 1: Displacement Control applied on the top of the cantiliver

This model has been designed with uniaxial material steel 01 with a plastic behavior with hardening in the plastic part.

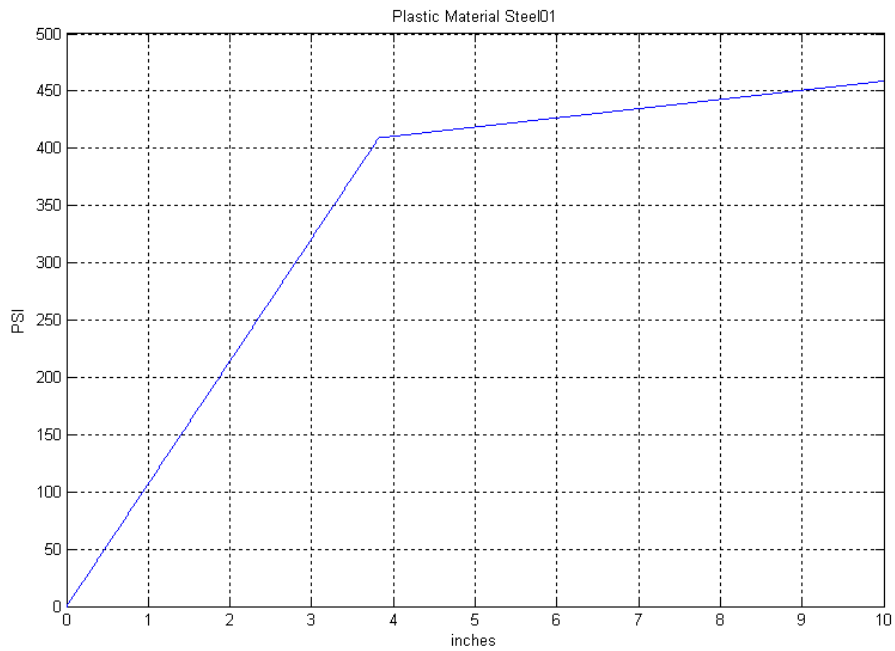
2.3.1 STEEL 01

Steel 01 looks like this:



Graphic 2: Definition of steel 01 from OpenSees website

The results after having applied with OpenSees the displacement control previously explained are:



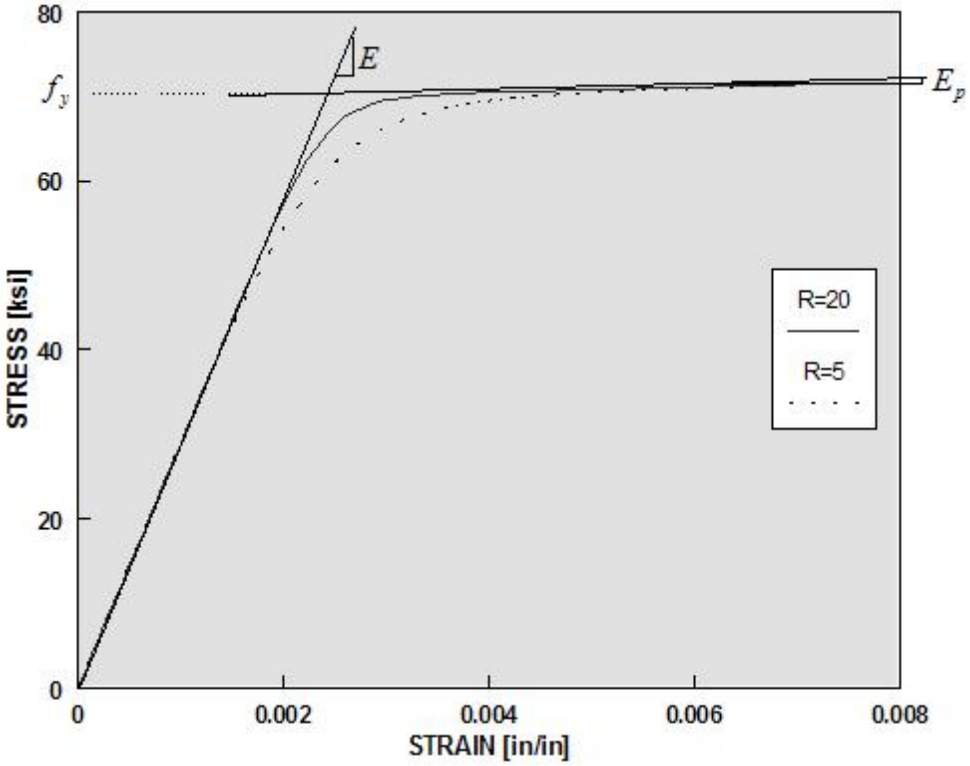
Graphic 3: Graphic force-deformation for model designed with steel 01

Now, this is going to be the reference model for this part of the thesis. The objective is going to be to fit this model to the real behavior of a pushover with the same characteristic in a real experiment.

2.3.1 STEEL 02

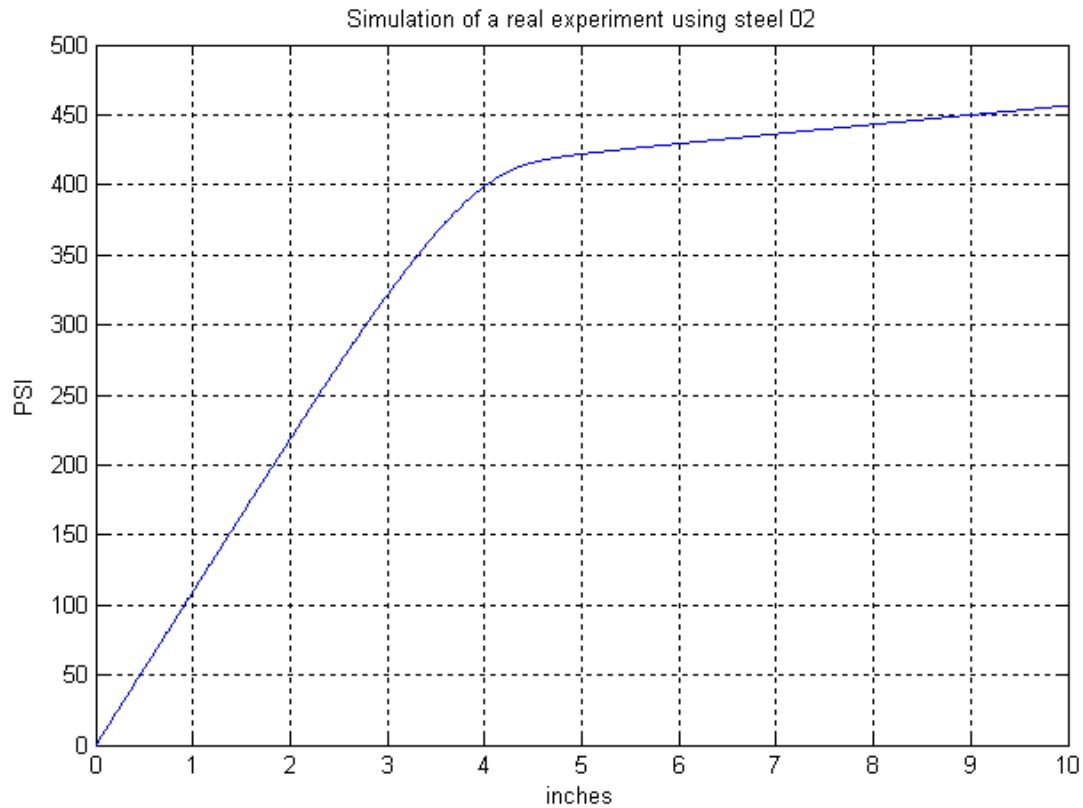
The lack of a real experiment with a pushover has been solved designing a second model closest to reality with a gradual change of slope. That has been possible by using uniaxial material steel 02 from the OpenSees materials library.

The aim is to fit the model made with steel 01 to the simulation of a real material made with steel 02.



Graphic 4: Definition of steel 02 by OpenSees website

Once applied the same pushover as in graphic 1 the results of OpenSees are:



Graphic 5: Displacement-force graphic for steel 02

2.4 FITTING CURVES

The first fitting model has been though to fit the model that uses steel 01, that depends in 3 parameters, to the simulation of a real experiment using steel 02.

The parameters that uses steel 01 are:

- K0: Stiffnes or Young Modulus
- Fy: Yield strenght
- b: Strain-hardening ratio

Therefore the fitting consists in adjusting a model that uses 3 variables into the simulated experiment.

The idea of curve fitting is to find a mathematical model that fits the experimental data. Assuming that there are theoretical reasons for picking a function of a certain form. The fitting curve finds the specific coefficients or parameters that make the function match the data as closely as possible.

Some people try to use curve fitting to find which of thousand of functions fit their data.

$y=ax+b$ will exactly connect 2 points
$y=ax^2+bx+c$ will exactly connect 3 points
...
$y=SUM(i=0,n)ai\cdot xi$ will exactly connect n+1 points

Generalizing instead of fitting points we could fit constraints, such as angle, curvature, etc.

i.e. $y=ax+b$ would fit 1 point and 1 angle.

If we have more than n+1 constraints, where n is the degree of the polynomial. There is no exact fit to all constraints, however there are methods to evaluate each approximation such as the least squares method.

Why would we want to approximate fit when we could just increase the degree of the polynomial equation and get an exact match?

1. Even if an exact match exists, it does not necessarily mean that we can find it. Depending on the algorithm used, we may encounter a divergent case, where the exact fit cannot be calculated, or it might take too much computer time to find the solution. Either way it is better to accept an approximate solution.

2. We may actually prefer the effect of the averaging out questionable data points in a

sample, rather than deforming the curve to fit them exactly

3. High order polynomials can be highly oscillatory. If we run a curve through points A and B, we would expect the curve to run somewhat near the midpoint A and B, as well. this may not happen with high order polynomial curves, they may even have values that are very large in magnitude. With low order polynomials, the curve is more likely to fall near the midpoint.

4. High order polynomials tend to be smooth and high order polynomial curves tend to be lumpy.

2.4.1 LEAST SQUARES

The methodology used in the process of fitting the curve is least squares. The method of least squares is a standard approach to the approximate solution of overdetermined systems, i.e. sets of equations in which there are more equations than unknowns. Least squares means that the overall solution minimizes the sum of the squares of the errors made in solving every single equation or in the case of the thesis, minimizes the sum from the first to the last iteration of the square of the distance between the model and the experimental data.

One of the most important applications of least squares is in data fitting. The best fit in the least-squares sense minimizes the sum of squared residuals, a residual being the difference between an observed value and the fitted value provided model as explained at the end of the last paragraph.

2.4.1 FIRST FITTING MODEL

The model that uses steel 01 has 3 unknowns: E, Fy, b.

A general idea of the values of the parameters is possible, so let's define some boundaries. Then a Matlab code has been defined runs a triple loop and calculates the error that corresponds to each combination of value of E(i) , b(j) and Fy(k). It saves the values of the errors in a matrix and then chooses the minimum one, obtaining the combination of values for which the error is minimum. An scheme of the code is:

```
Emin=1
Emax=10.000

bmin=0.001
bmax=0.99

Fymin=1
Fymax=10.000


err2=zeros(101,101,101)

For i=1:101
        For j=1:101
                For k=1:101


                        E = Emin + ((Emax-Emin)/100)*(i-1)
                        b = bmin + ((bmax-bmin)/100)*(j-1)
                        Fy = Fymin + ((Fymax-Fymin)/100)*(k-1)

                        make Opensees File with E, b, Fy
                        Run Opensees
                        Get the pushover curve

                        Let's assume that pushover curve has M points

                        For l=1:M

                                err2(i,j,k)=err2(i,j,k)+(estimated-target)^2
                        End

                End
        End
End

Find minimum of err2 --> i,j,k of the minimum --> E, b, Fy that best fits the curve.
```

However, this methodology is quite rough and very time consuming. The accuracy depends on the number of partitions (n) that are made in each of the parameters and the time that takes the program to calculate is is going to be of the order of $O(n^3)$. An improvement of the program is needed.

2.4.2 CODE IMPROVAL FOR 1 PARAMETER

Thinking in only one variable, let's say for example only parameter E needs to be calibrated to fit the curve. The different values of E that the program reaches are defined by:

$$E = \frac{Emax - Emin}{Nsteps} \cdot (i - 1) + Emin$$

Emax and Emin are defined by the user as boundaries of the parameter, i is the counter and Nstep is also chosen by the user depending on the accuracy he wants to obtain.

Then if we want to approach E to the experimental E (Eexp) the maximum error that we will have will be:

$$\in max = \frac{Emax(i) - Emin(i)}{Nsteps} \cdot \frac{1}{2}$$

Image 2: Diagram of the working of the code improval

If we make Emax and Emin to change to Emax2 and Emin2, within a distance from the Esol of $\pm\in max$ in a new loop of iterations for Ntimes we will approach the solution in a faster way.

The precision of the previous program was:

$$Prec = \frac{Emax - Emin}{Nsteps}$$

And the new program precision would be:

$$Prec = \frac{Emax - Emin}{Nsteps^{Ntimes}}$$

Splitting the domain of the variable into 2 steps instead of Nsteps we obtain the so called bisection method.

At the first iteration of the program the boundaries are defined and let's call Esol to the real solution.

Image 3: Diagram of the initial status of the program.

Let's calculate two different errors; the one corresponding to the distance from the E solution to the minimum value of the boundary (err1) and the one consisting on the distance between the solution and the maximum value of the boundary (err2).

In the example drawn above err1 will be smaller than err2, since the solution is closer to Emin. Also, the maximum will be half of the distance between Emax and Emin.

$$\in max = \frac{1}{2} \cdot (Emax - Emin)$$

In the following steps (Nsteps) of the loop a distance of $\in max$ expanding from the closest previous choice will be taken.

Leading to the following step:

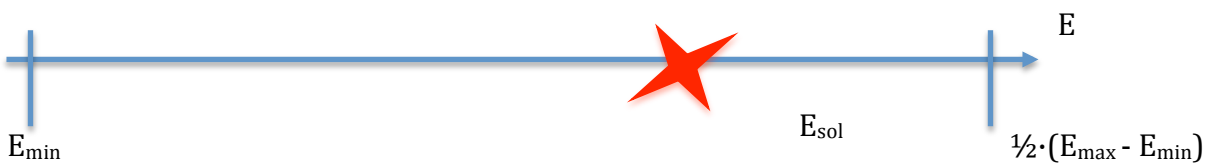


Image 4: Diagram of the next iteration.

Having a new value for the boundary that corresponds to the maximum value and two new values for err1 and err2.

The accuracy of the program is:

$$Prec = \frac{Emax1 - Emax2}{2^{Nsteps}}$$

And the code for a single parameter would be:

Program

Emin and Emax chosen

For i=1:n

```
Get err1 corresponding to Emin
Get err2 corresponding to Emax

        If err1<err2
                Emin=Emin
                Emax=Emin+1/2·(Emax-Emin)

        If err1>err2
                Emin=Emax-1/2·(Emax-Emin)
                Emax=Emax

        If err1=err2
                Esol=1/2·(Emax-Emin)
                End For

End For

If err1<err2
        Esol=err1
If err1>err2
        Esol=err2

End Program
```

Notice that to get err1 and err2 corresponding with each Emin and Emax, it is needed to run OpenSees and make the model with the corresponding E value. Then obtain an output file that will be compared with the experimental data file to obtain the corresponding error.

The issue is that not only the model consists in 1 parameter but it consists in 3.

Emin, Emax, Fymin, Fymax, bmin and bmax delimit our spatial boundary as follows:

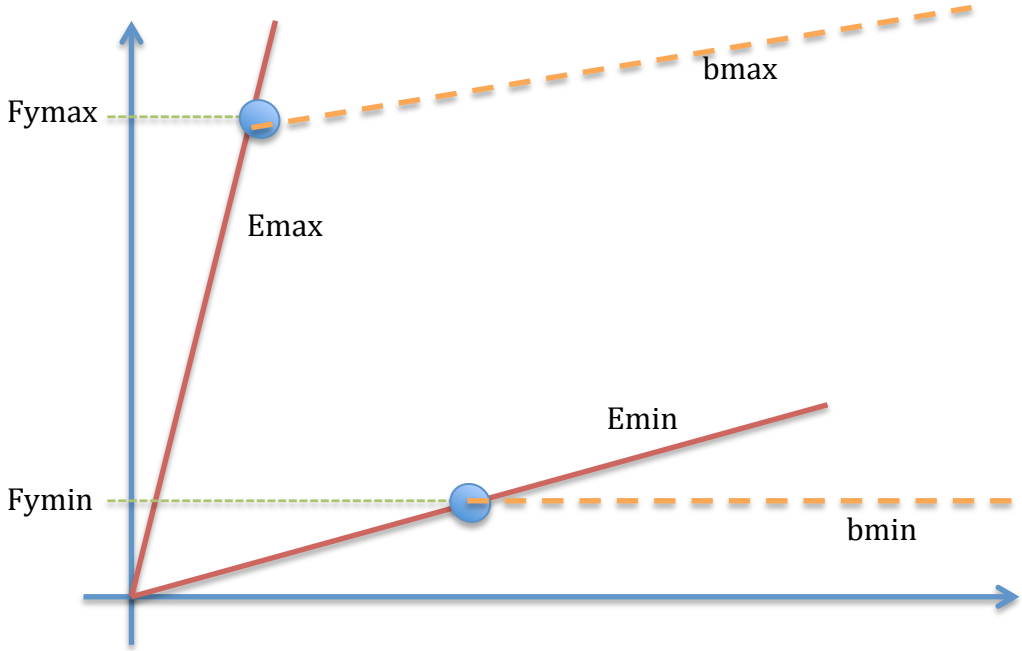


Image 5: Boundaries of each of the parameters

An extrapolation of the bisection method for a multidimension problem is needed.

2.4.3 MULTIVARIABLE BISECTION METHOD

First of all let's think of two parameters, lets say P1 and P2. For each pair of possible values of P1 and P2 there is going to be an error in the modeled beam, but one of these errors is going to be the minimum.

Se the drawn a grid of the values of P1 and P2, and then a possible approach to the solution like in the one variable method previously explained but in two dimensions.

The procedure is:

Start with an minimum and maximum value for each of the parameters' boundaries.

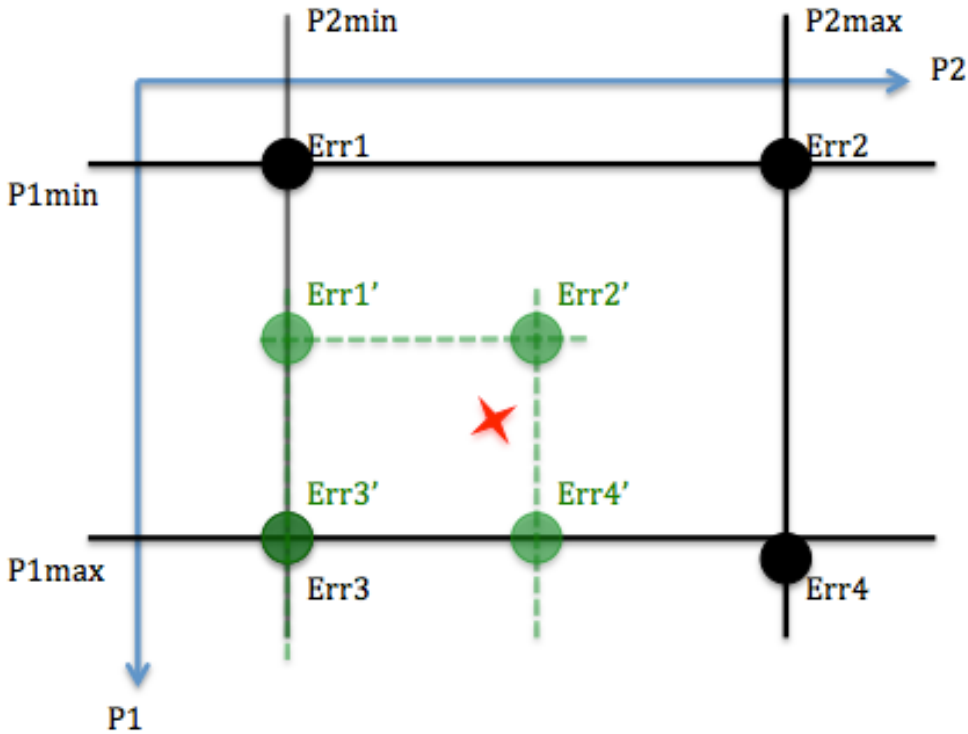


Image 6: Bi-dimensional diagram of the bisection method

The red start represents the location of the minimum error.

OpenSees needs to be runned 4 times in each iteration (2^{Np}; where Np equals the number of parameters, in this case 2)

1. One time for the parameters P1min and P2min.
2. A second time for the parameters P1min and P2max
3. A third time for the parameters P1max and P2min
4. A fourth time for the parameters P1max and P2max

For every time Opensees is runned Matlab obtains the error corresponding to each point. That is err1, err2, err3 and err4.

Then take the minimum of these 4 errors. (In this example it's err3)

After the first iteration a new domain is defined by 4 updated points: P1min', P1max', P2min' and P2max'. Surrounding the point corresponding to err3. (Green)

To select this domain, there will be an if function with 4 different cases (2^{N_p} different cases) depending where the minimum error is. (i.e):

If (minimum error corresponds to err1)

P1min'= P1min
P1max'= P1min+(P1max-P1min)·1/2
P2min'= P2min
P2max'= P2min+(P2max-P2min)·1/2

else (minimum error corresponds to err2)

P1min'= P1min
P1max'= P1min+(P1max-P1min)·1/2
P2min'= P2min+(P2max-P2min)·1/2
P2max'= P2min

else (minimum error corresponds to err3)

P1min'= P1min+(P1max-P1min)·1/2
P1max'= P1min
P2min'= P2min
P2max'= P2min+(P2max-P2min)·1/2

else (minimum error corresponds to err4)

P1min'= P1min+(P1max-P1min)·1/2

P1max'= P1min

P2min'= P2min

P2max'= P2min+(P2max-P2min)·1/2

End if

The new domain is defined, tis domain is smaller than the previous one, specifically ¼ smaller ($1/2^{Np}$ smaller).

This is the main core of the program. Now to converge to a solution a loop is needed (i=1:n) to decide the quality of the solution.

The maximum error in this method will be:

$$Emax = \frac{1}{2 \cdot n} \cdot \sqrt{(P1max - P1min)^2 + (P2max - P2min)^2}$$

All this theory and this code can be extrapolated to more than two variables.

Therefore, to develop a code for 3 parameters, instead of a quadrilater that defines the boundaries a cube will do so, and then 8 errors (2^3) will need to be checked instead of 4, that means running OpenSees 8 times in each iteration of the loop.

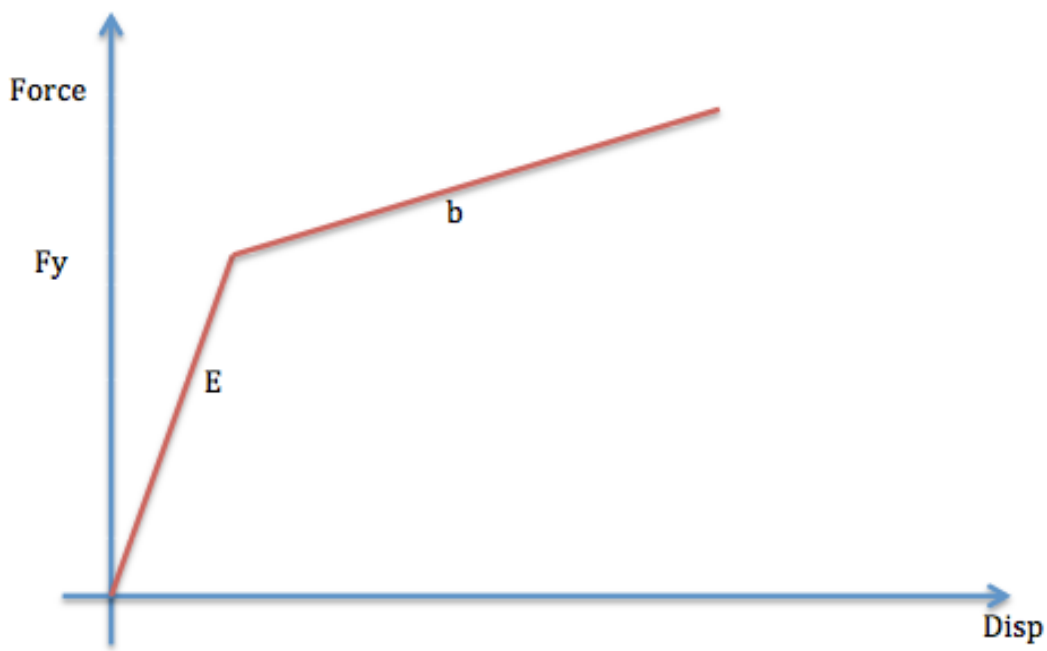
In general if we have P parameters, we will have a cube of P dimension, and therefore in each iteration of the loop we will need to run the Opensees 2^P times.

The if function will have also 2^P cases.

And the error will be:

$$Emax = \frac{1}{2 \cdot n} \cdot \sqrt{(P1max - P1min)^2 + \cdots + (Ppmax - Ppmin)^2}$$

However the model depends only on 3 variables. That is the slope of the elastic part (E), the slope of the plastic part (b), and the ordinate of the point where there is the slope change (Fy) or elastic strength.



Graphic 6: Steel 01 model behavior

Let's derive the multivariable code previously exposed but in more detail, now for three parameters.

To follow the same notation, let's call P1=E, P2=Fy and P3=b.

Program for three variables semi code

Firstly select an initial interval of minimum and maximum and give value to the following variables:

P1min, P1max, P2min, P2max, P3min and P3max

For i=1:n

FunctionGetTheErrors(input: P1min, P1max, P2min, P2max, P3min, P3max; Output: Err1, Err2, Err3, Err4, Err5, Err6, Err7, Err8);

Get the minimum error of the 8 values from Err1 to Err8→Errmin;

FunctionGetTheNewPoints(input: Errmin, Output: P1min, P1max, P2min, P2max, P3min, P3max)

n=n+1

End for

Write the P1, P2 and P3 corresponding to the minimum error.

End Program

DEVELOPMENT OF THE TWO FUNCTIONS

Function Get the Errors

What this function does is to obtain the errors for each of the combination of the three parameters, which results in 8 different combinations.

To do so, we have to execute the Opensees 8 different times.

For each time that we execute the Opensees we have to compare the obtained curve with the experimental one, and obtain an error.

FunctionGetTheErrors

Run Opensees for P1=P1min, P2=P2min and P3=P3min

Compare with the experimental Data

Get err1

.
.
.

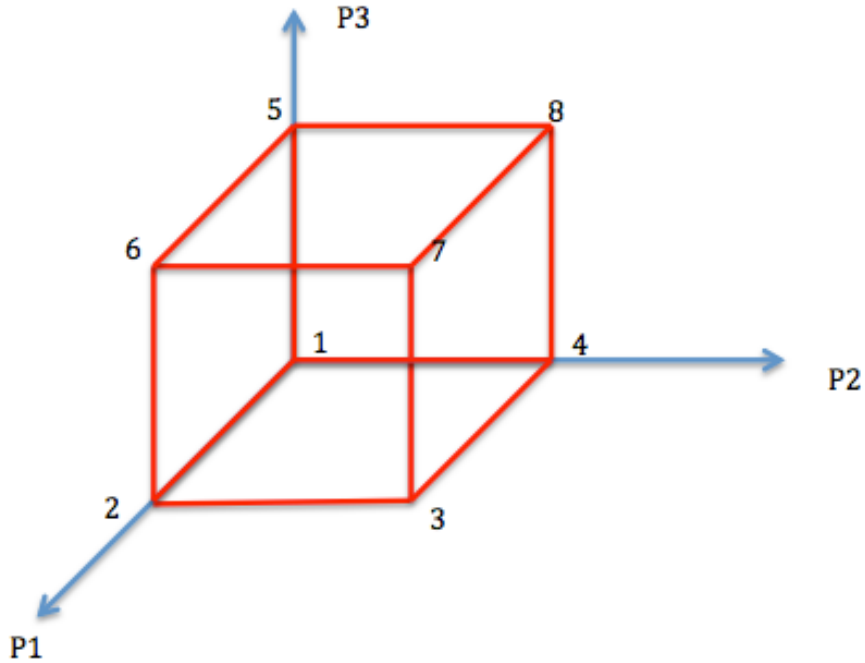
Run Opensees for P1=P1max, P2=P2max and P3=P3max

Compare with the experimental Data

Get err8

FunctionGetTheNewPoints

Eight new points will be selected depending on the location of the minimum error. In comparison to the two dimensional example imagine a cube now instead of a square, which is the two dimensional case. Therefore we will have 8 subspaces, instead of four.



Graphic 7: Tri-dimensional domain of the steel 01 parameters.

This function will be defined by 8 "if" sub functions. There is going to be one sub cube of ¼ of the volume of the bigger cube attached either to one of the 8 points.

If ErrMin=err1

P1min=P1min

P1max=P1min+(P1max-P1min)·1/2

P2min=P2min

P2max=P2min+(P2max-P2min)·1/2

P1min=P3min

P1max=P3min+(P3max-P3min)·1/2

Else Errmin=err2

P1min=P1min+(P1max-P1min)·1/2
P1max=P1max

P2min=P2min
P2max=P2min+(P2max-P2min)·1/2

P1min=P3min
P1max=P3min+(P3max-P3min)·1/2

Else Errmin=err3

P1min=P1min+(P1max-P1min)·1/2
P1max=P1max

P2min= P2min+(P2max-P2min)·1/2
P2max=P2max

P1min=P3min
P1max=P3min+(P3max-P3min)·1/2

Else Errmin=err4

P1min=P1min
P1max=P1min+(P1max-P1min)·1/2

P2min= P2min+(P2max-P2min)·1/2
P2max=P2max

P1min=P3min

P1max=P3min+(P3max-P3min)·1/2

Else Errmin=err5

P1min=P1min

P1max=P1min+(P1max-P1min)·1/2

P2min=P2min

P2max=P2min+(P2max-P2min)·1/2

P1min=P3min+(P3max-P3min)·1/2

P1max=P3max

Else Errmin=err6

P1min=P1min+(P1max-P1min)·1/2

P1max=P1max

P2min=P2min

P2max=P2min+(P2max-P2min)·1/2

P1min=P3min+(P3max-P3min)·1/2

P1max=P3max

Else Errmin=err7

P1min=P1min+(P1max-P1min)·1/2

P1max=P1max

P2min= P2min+(P2max-P2min)·1/2

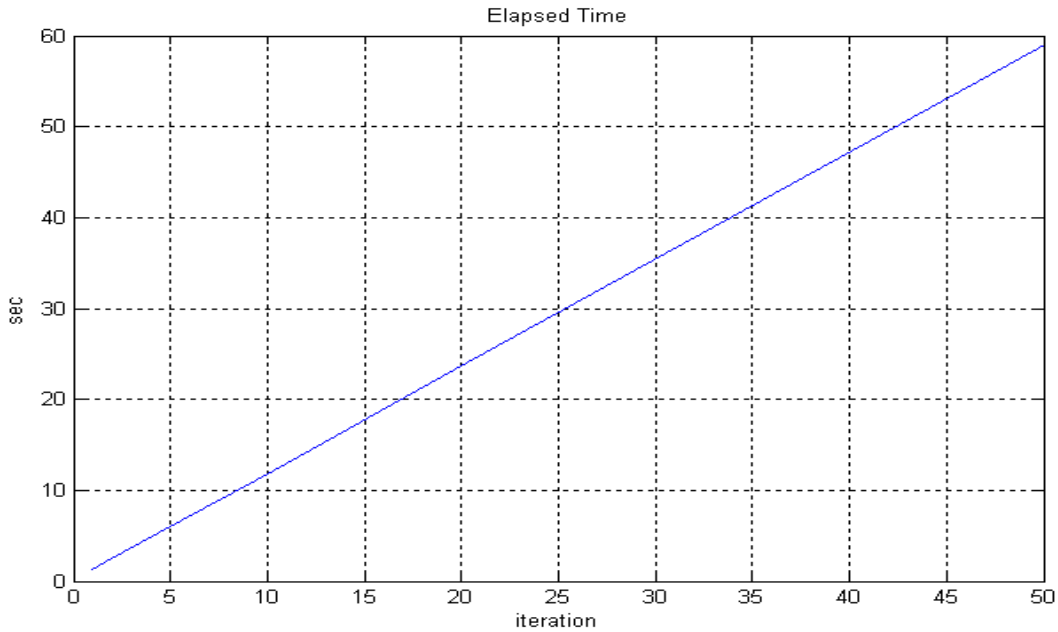
```
        P2max=P2max

        P1min=P3min+(P3max-P3min)·1/2
        P1max=P3max


Else Errmin=err8

        P1min=P1min
        P1max=P1min+(P1max-P1min)·1/2

        P2min= P2min+(P2max-P2min)·1/2
        P2max=P2max

        P1min=P3min+(P3max-P3min)·1/2
        P1max=P3max

End If
```

2.4.4 RESULTS OF THE MULTIVARIABLE PROGRAM WITH THREE PARAMETERS

The performance of the code works very well. The next two graphics show the time that it takes to run the program in function of the number of iterations, and a second graphic that shows the maximum error in function of the number of iterations too.

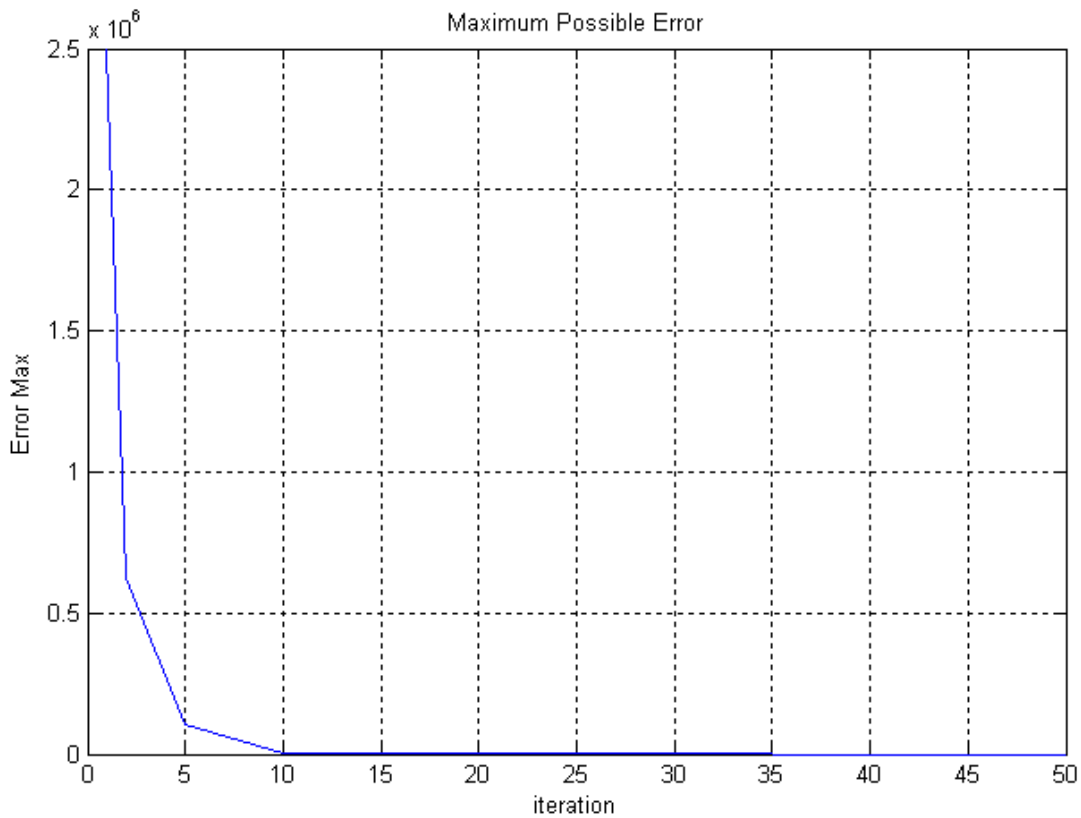
The maximum error that can be found in each iteration has been calculated as follows:

$$\epsilon_{max} = \frac{1}{2 \cdot n} \cdot \sqrt{(P1max - P1min)^2 + (P2max - P2min)^2 + (P3max - P3min)^2}$$



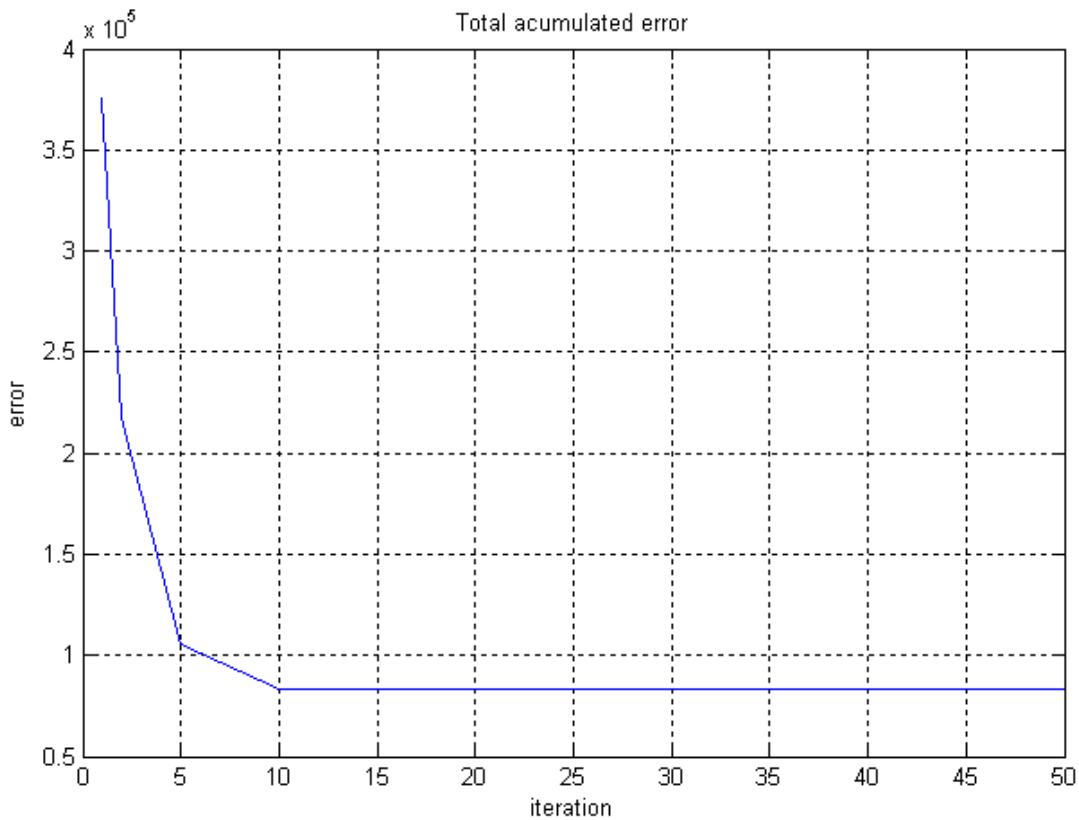
Graphic 8: Time that takes the program to run depending on the number of iterations

It is important to have clear that this ϵ_{max} is the maximum error in each of the iterations, not the accumulative one.



Graphic 9: Error in function of the number of iterations

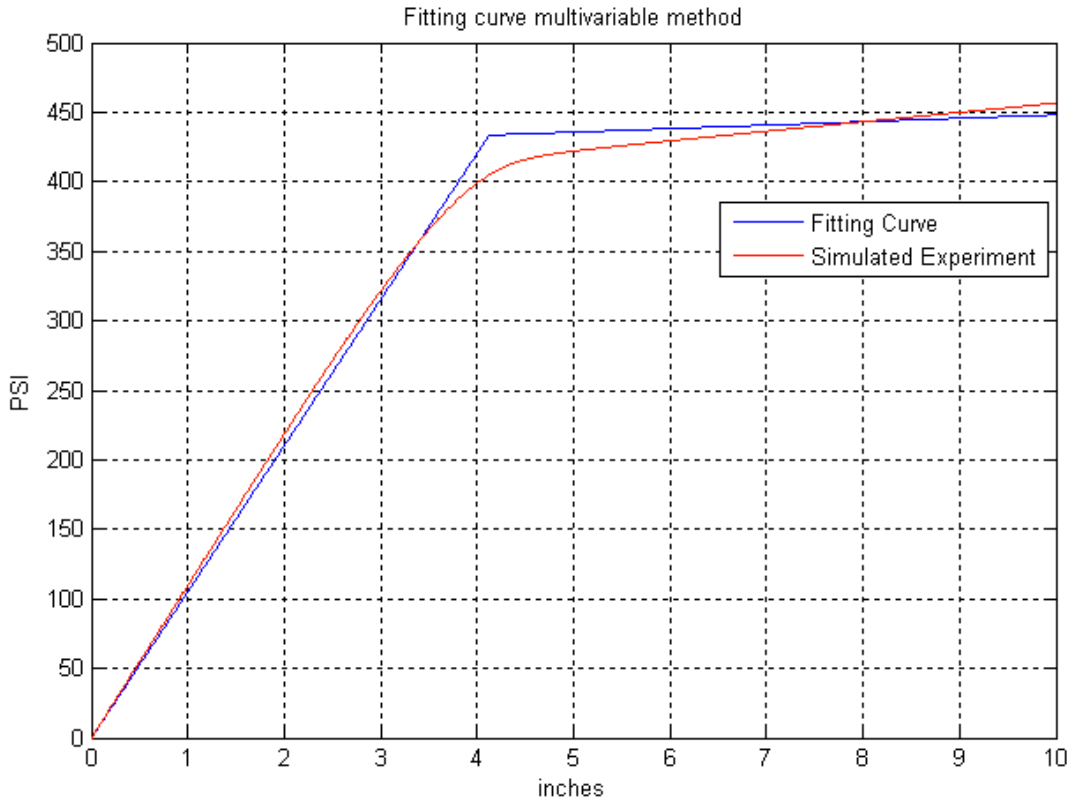
The total accumulated error is:



Graphic 10: Total accumulated error.

The graphic from above presents a minimum starting on 10 iterations that diminishes at a very slow rate.

The next curve has been obtained using 50 iterations of the bisection program method.



Graphic 11: Fitting curve using multivariable bisection method.

The program works, but as we can graphically see there is still an error that could be improved.

CHAPTER 3: GENETIC ALGORITHM

Later on, in the thesis, a fitting using a much more complex material and with more variables is going to be done. There are other methodologies used for finding the minimum of a function, specifically, there is a kind of algorithms called Genetic Algorithms, or GA that are very powerful in terms of calculating the minimum of a function.

3.1 INTRODUCTION TO GENETIC ALGORITHMS

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

In a genetic algorithm, a population of strings (alled chromosomes or genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based in their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generation has been produced, or a satisfactory fitness level has been reached for the population. If the

algorithm has terminated due to a maximum numbr of generations, a statisfactory solution may or may not have been reached.

A typical genetic algorithm requires:

1. A genetic representation of the solution domain
2. A fitness function to evaluate the solution domain.

A standard representation of the solution is as an array of bits. Arrays of ther types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used.

Once the genetic representation and the fitness function are defined, GA proceeds to initialize a population of solutions randomly, then improve it through repetitive application of mutation, crossover, inversion and selection operators.

The typical mechanism of this algorithms can be explain with the following graphic:

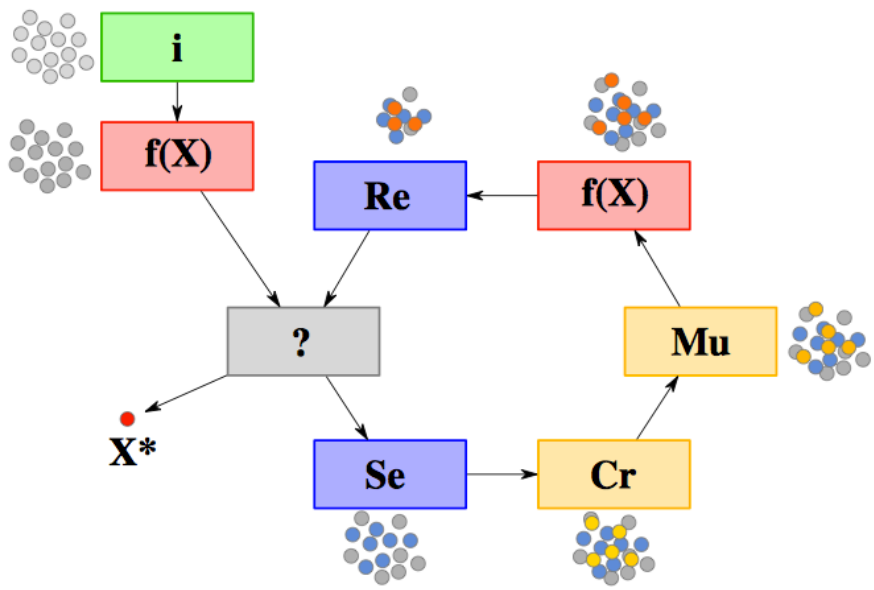


Image 7: Working diagram of a Genetic Algorithm

Where (i) means initialization, (f(X)) evaluation, (?) stopping criterion, (Se) selection, (Cr) cross over, (Mu) mutation, (Re) replacement and (X*) optimal.

The main steps are:

Initialization: Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be seeded in areas where optimal solutions are likely to be found.

Selection: During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are

typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time consuming. To each of the solutions of the population an evaluation function is applied to know how good is the solution that is being codified.

Reproduction: The next step is to generate a second generation population of solution from those selected through genetic operators: crossover (also called recombination) and/or mutation.

For each new solution to be produced a pair of "parent" solution is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristic of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research suggests more than two parents are better to be used to reproduce a good quality chromosome.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.

Although crossover and mutation are known as the main genetic operators, it is possible to use other operators such as regrouping, colonization-extinction, or migration in genetic algorithms.

Ending condition: This generational process is repeated until a termination condition has been reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- A fxed number of generations is reached
- An allocated budged (computation time/money) is reached
- The highes ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combination of the above

While the ending condition is not accomplished GA creates new generations by reproducing its population. Some of the reproduction methods are briefly explained:

Selection: After knowing the requirements or conditions for each possible solution, we proceed to select the next possible solutions in the next iterations.

Crossover: Works on two different solutions at the same time to generate two following solutions, where we combine the characteristics of the previous ones.

Mutation: Modifies the chance and it allows to get to solutions in the space of search which is not covered by the actual search.

Replacement: Once applied the operators, the best solution are selected to be the next best solution.

To sum up, a simple generational genetic algorithm procedure consists in:

1. Choose the initial population of individuals
2. Evaluate the fitness of each individual in the population
3. Repeat on this generation until termination
    a. Select the best fit individuals for reproduction

b. Breed new individuals through crossover and mutation operations to give birth to offspring
c. Evaluate the individual fitness of new individuals
d. Replace least-fit population with new individuals

3.2 APPLICATION TO THE FITTING PROBLEM

The objective is to apply GA technology to the fitting of the model using steel 01 with OpenSees to the simulated experimental behavior using steel 02.

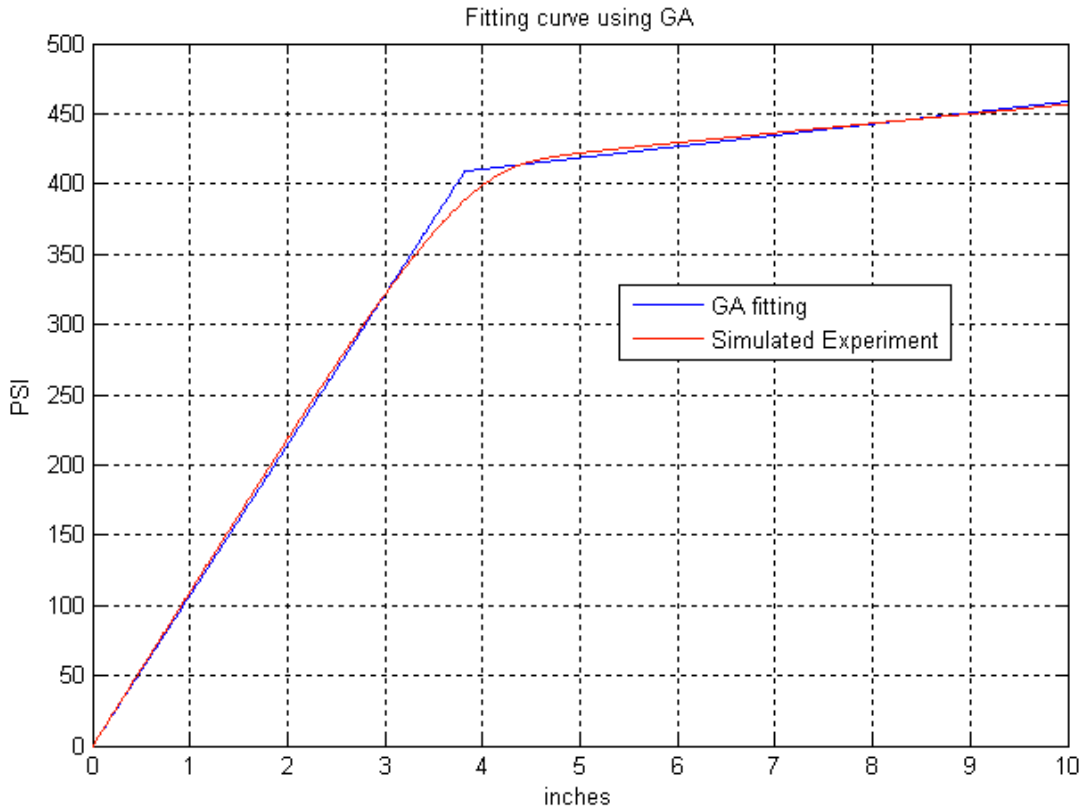
To do so, the optimization tool from Matlab will be used. It already has the implementation of the genetic algorithm code. What the user needs to define is the objective function, and choose the properties of the GA, such as number of variables, initial population, termination options, among others.

To use a genetic algorithm from the Matlab desktop type **optimtool('ga')** and select **gamultiobj** which is a multiobjective optimization using genetic algorithm for more than one variable. Indicate the number of variables to optimize, which is three in this particular case. Call the function to optimize using the sign @ followed by the name of the function. And finally indicate the lowest and the highest boundaries for each of the variable in form of a vector using [MinValue1 MinValue2 MinValue3] and the same for the upper boundary.

The objective function depends on the vector P, that contains the parameters P1, P2 and P3, corresponding to E, Fy and b. Therefore P=[P1,P2,P3]. Given the three parameters the function calls OpenSees and runs a simulation obtaining model defined by the values of the vector P. Then it calculates the error by least squares between the model and the simulated experiment. The GA will obtain the value of P that minimizes the error.

3.3 RESULTS

After running the GA for the three parameter fitting problem we obtain the following results shown in the next graphic.

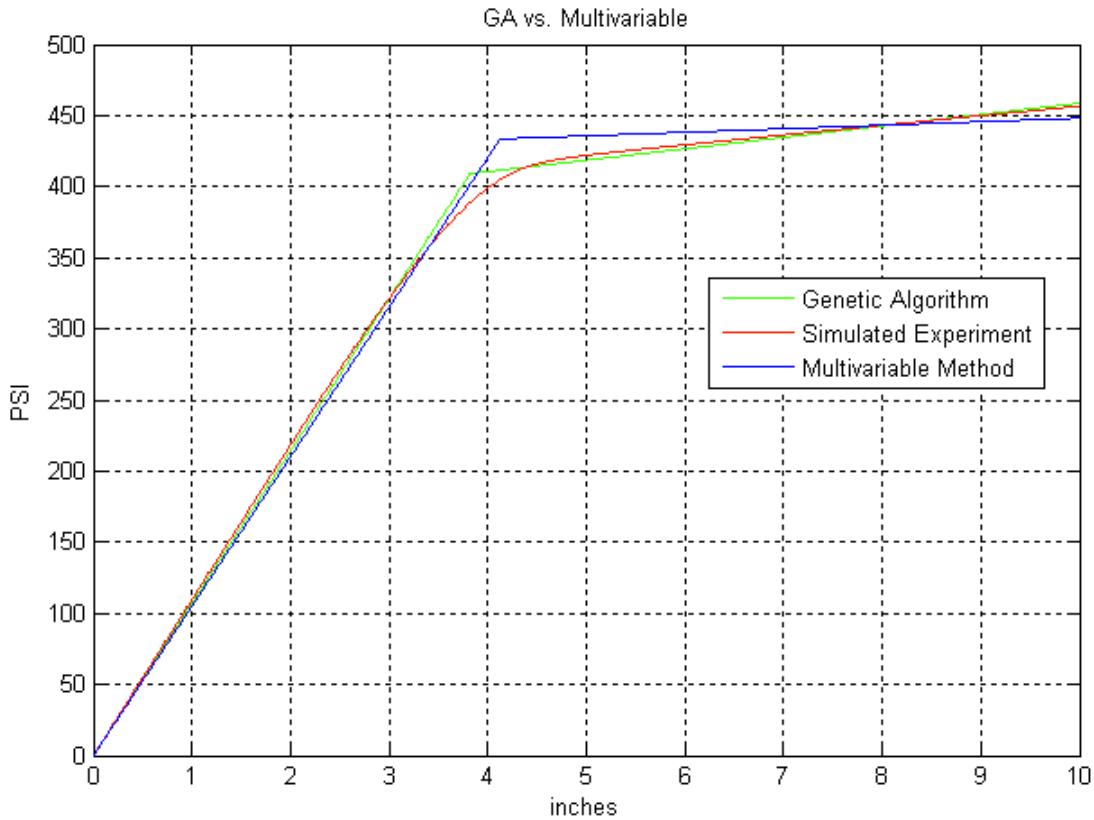


Graphic 12: Fitting curve using GA

The next table and graphic compares the multivariable bisection method of fitting curve with the genetic algorithm one. By comparing the values of each parameter it is easy to see that GA approaches better the solution. The corroboration of this fact can be seen by comparing the error, being the one from the GA around 4 times smaller.

	E	Fy	B	Error
Real experiment	29132476	66532	0.01032	0
Bisection program	27934570.312	69379.883	0.003	83535.0
GA	28461524.524	65464.425	0.012	19735.468

Graphically this fact is translated in a better approach to the experimental result, therefore being GA method the best option to fit a model to experimental results.



Graphic 13: Comparison of GA and Multivariable methods.

The only disadvantage is that while Multivariable Bisection method takes around 1 minute to calculate while GA takes twice as much time. However the benefits are much better, being therefore GA method more propitious for curve fitting calculation.

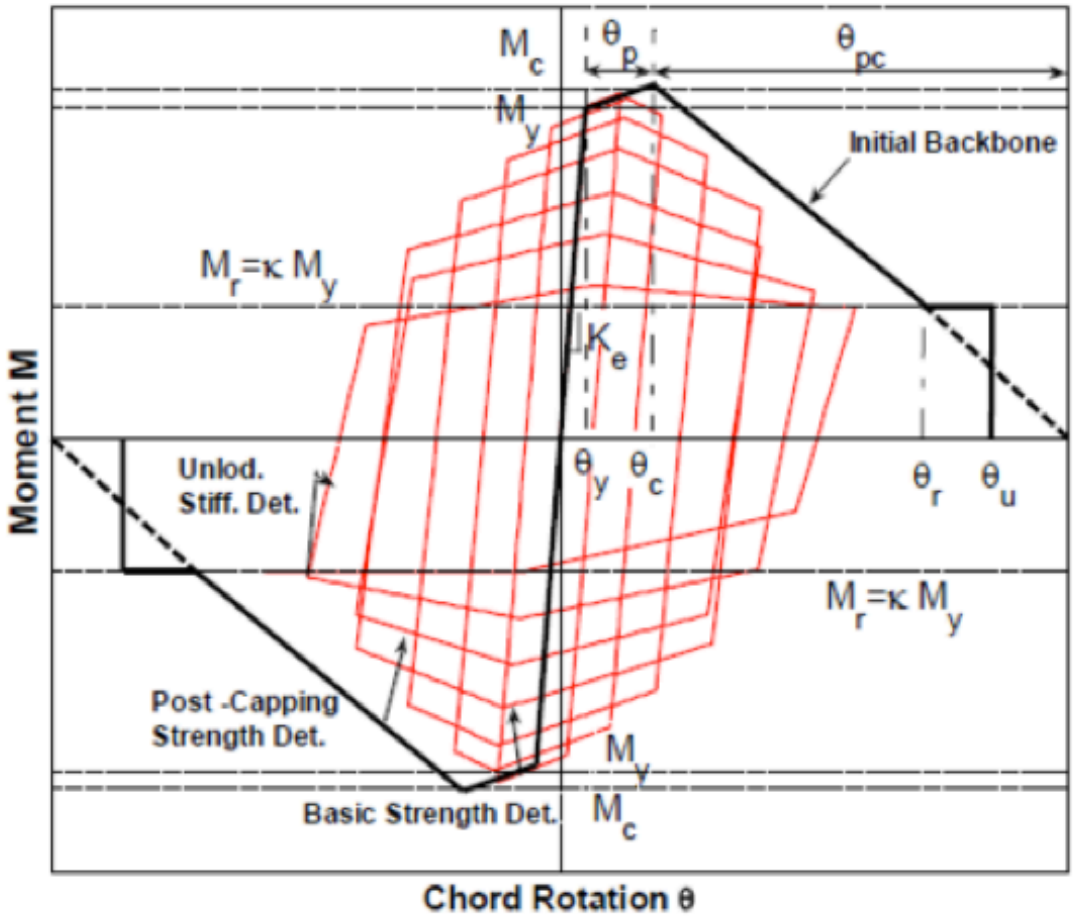
CHAPTER 4: UNIAXIAL BILINEAR MATERIAL

4.1 INTRODUCTION AND EXPLANATION OF OPENSEES UNIAXIAL BILINEAR MATERIAL

The main aim of the present thesis is to fit a simplified model to experimental data of reverse cycles. In order to do so, the model that will be used is going to be a material designed in OpenSees library called Uniaxial Bilinear Material.

The bilin material command is used to construct a bilinear material that simulates the modified Ibarra-Krawinkler deterioration model with bilinear hysteretic response.

The behavior of the bilin material defined by OpenSees is:



Graphic 14: Bilin material behavior

Where

- My and θy represents the yield strength and rotation
- K_e=My/ θy is the effective stiffness
- θp is the pre-capping rotation capacity for monotonic loading
- θpc is the post-capping rotation capacity
- Mr= κ My is the residual strength
- Θu is the ultimate rotation capacity

In order to define the material, the user needs to define the following command and parameters when programing in TCL code.

uniaxialMaterial Bilin $matTag $K0 $as_Plus $as_Neg $My_Plus $My_Neg $Lamda_S $Lamda_C $Lamda_A $Lamda_K $c_S $c_C $c_A $c_K $theta_p_Plus $theta_p_Neg $theta_pc_Plus $theta_pc_Neg $Res_Pos $Res_Neg $theta_u_Plus $theta_u_Neg $D_Plus $D_Neg

Where:

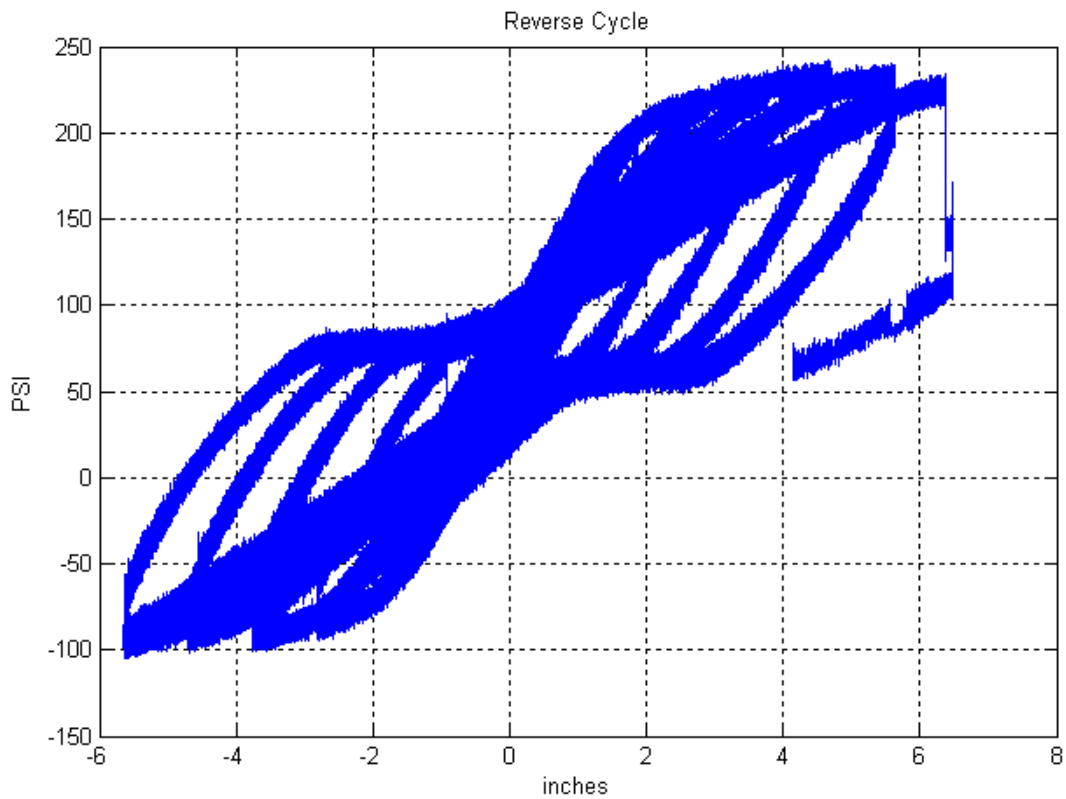
$matTag	integer tag identifying material
$K0	elastic stiffness
$as_Plus	strain hardening ratio for positive loading direction
$as_Neg	strain hardening ratio for negative loading direction
$My_Plus	effective yield strength for positive loading direction
$My_Neg	effective yield strength for negative loading direction (negative value)
$Lamda_S	Cyclic deterioration parameter for strength deterioration
$Lamda_C	Cyclic deterioration parameter for post-capping strength deterioration
$Lamda_A	Cyclic deterioration parameter for acceleration reloading stiffness deterioration (is not a deterioration mode for a component with Bilinear hysteretic response).
$Lamda_K	Cyclic deterioration parameter for unloading stiffness deterioration
$c_S	rate of strength deterioration. The default value is 1.0.
$c_C	rate of post-capping strength deterioration. The default value is 1.0.
$c_A	rate of accelerated reloading deterioration. The default value is 1.0.
$c_K	rate of unloading stiffness deterioration. The default value is 1.0.
$theta_p_Plus	pre-capping rotation for positive loading direction (often

	noted as plastic rotation capacity)
$theta_p_Neg	pre-capping rotation for negative loading direction (often noted as plastic rotation capacity) (positive value)
$theta_pc_Plus	post-capping rotation for positive loading direction
$theta_pc_Neg	post-capping rotation for negative loading direction (positive value)
$Res_Pos	residual strength ratio for positive loading direction
$Res_Neg	residual strength ratio for negative loading direction (positive value)
$theta_u_Plus	ultimate rotation capacity for positive loading direction
$theta_u_Neg	ultimate rotation capacity for negative loading direction (positive value)
$D_Plus	rate of cyclic deterioration in the positive loading direction (this parameter is used to create assymetric hysteretic behavior for the case of a composite beam). For symmetric hysteretic response use 1.0.
$D_Neg	rate of cyclic deterioration in the negative loading direction (this parameter is used to create assymetric hysteretic behavior for the case of a composite beam). For symmetric hysteretic response use 1.0.

4.2 EXPERIMENTAL DATA USED TO CALIBRATE THE MODEL

Experimental data from a reverse cycle pushover of a cantilever has been taken from the NEES website (Network for Earthquake Engineering Simulation). The objective is to be able to fit a model that uses OpenSees bilinear material by using genetic algorithm techniques.

The data consisted in 185,880 pair of points of force and displacement that looked as follows right after the download:



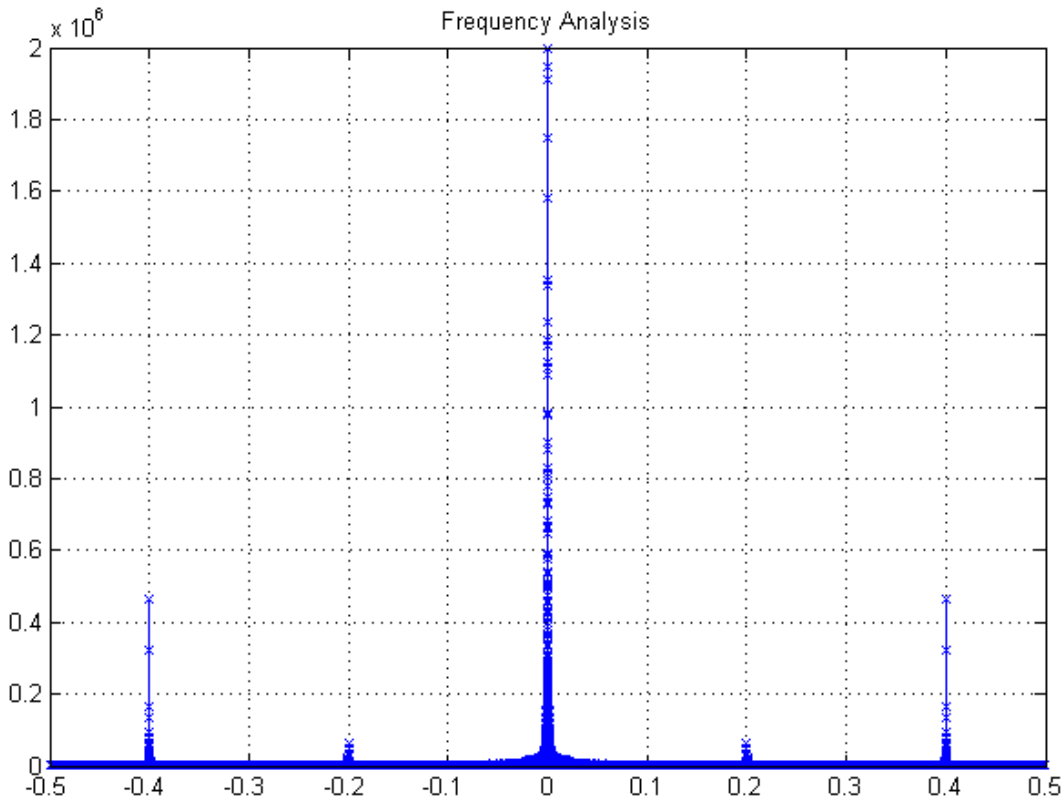
Graphic 15: Initial experimental data

As it is seen, the data presents a lot of vibration, or signal noise; this vibration should be reduced to adjust the model.

4.3 DATA FILTERING

In order to do the data filtering to reduce the noise an analysis of the frequency is required to analyze what other frequencies a part of the main one are presents on the data.

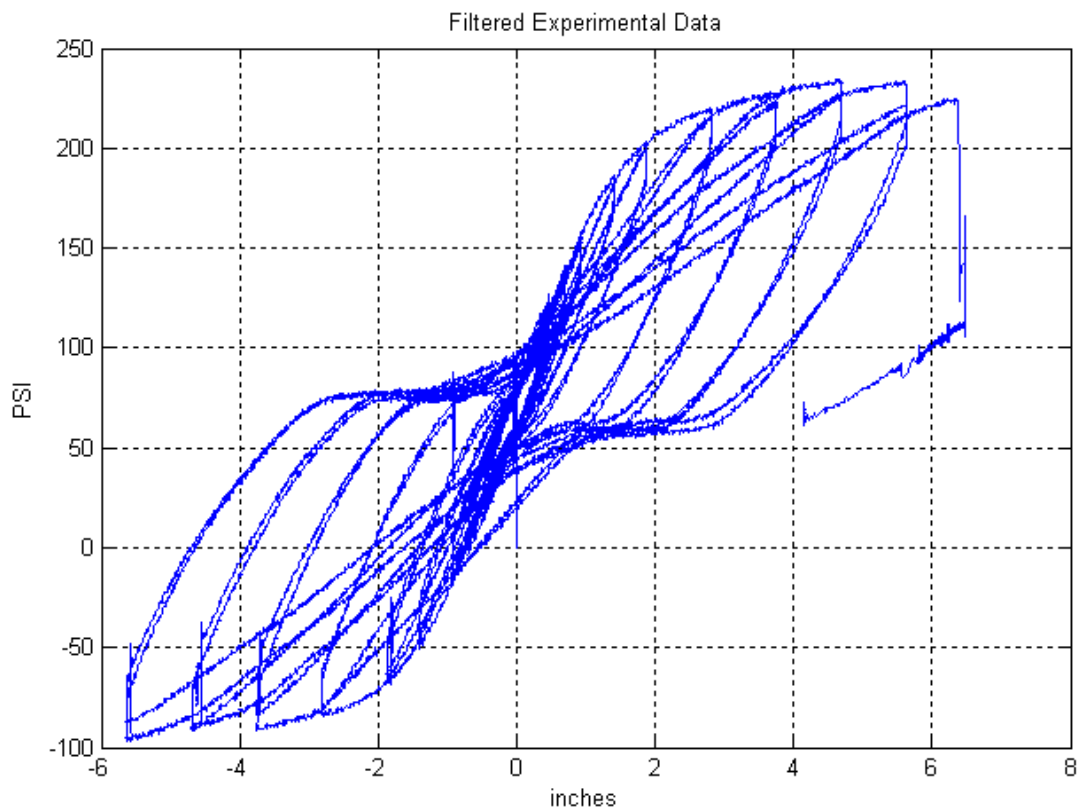
After applying a fast fourier transformation to the data, and normalize the points the frequency graphic obtained is:



Graphic 16: Frequency analysis

The main frequency is the highest peak located at the center of the graphic, and then there are two more peaks of lower amplitude that influence in the vibration of the signal. These other peaks have to be removed to avoid the noise, or diminish it. Therefore with the exception of the central part of the graphic, where the main peak is, the rest should be zero.

Substituting the values outside of the central peak by zero and applying the inverse of the fast Fourier transformation, the filtered data is obtained.



Graphic 17: Filtered experimental data

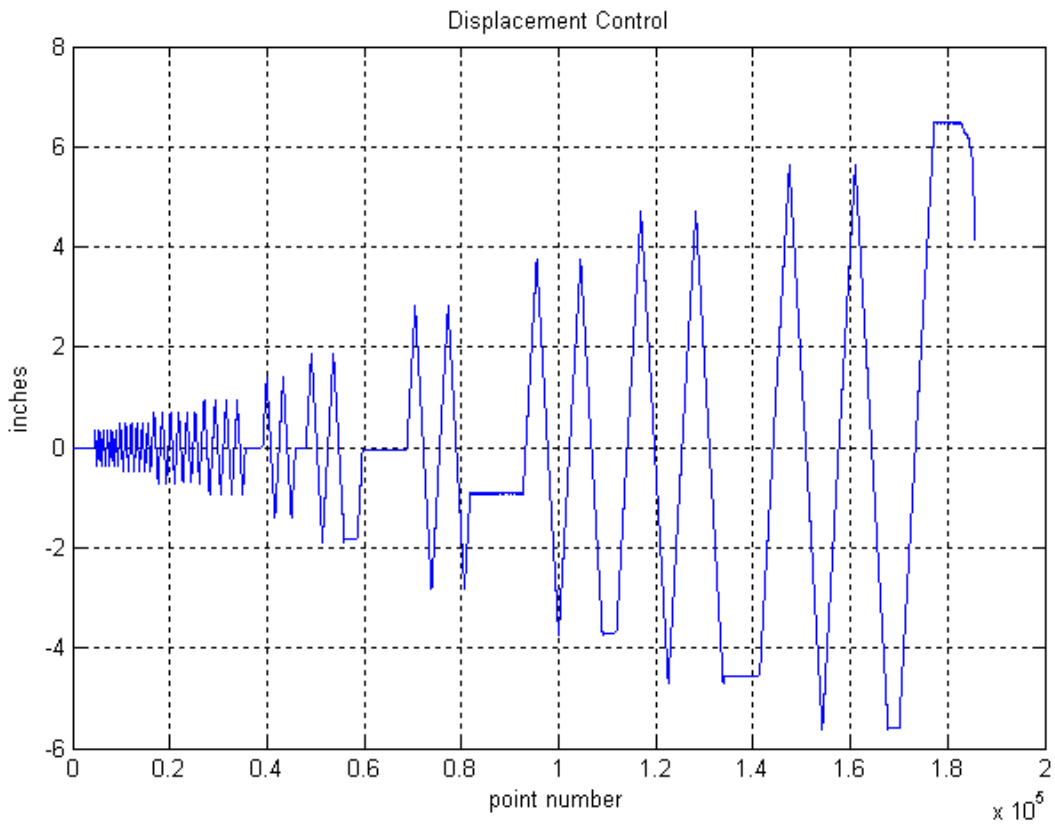
Once the data is ready to work with, is time to define the model by OpenSees.

4.4 OPENSEES MODEL

The objective is to model the behavior of the material using OpenSees. Therefore, instead of modeling the cantilever and the experiment, the simplest model is the best choice. This model consists in a one-dimension model, with a single degree of freedom and with no dimension. The OpenSees command used to build the model is called zero length element.

This command is used to construct an element defined by two nodes at the same location. The nodes are connected by multiple uniaxial material objects to represent the force deformation relationship for the element. In the case of this thesis, the uniaxial material will be the bilinear material previously explained.

An important thing in order to fit the model to the experimental data, is that the model needs to have the same displacement control as the real experiment. This displacement control is one with increasing amplitude as seen in the next graphic:



Graphic 18: Displacement control in inches of the experiment

The code corresponding to the displacement control load to the model can be seen in the appendix.

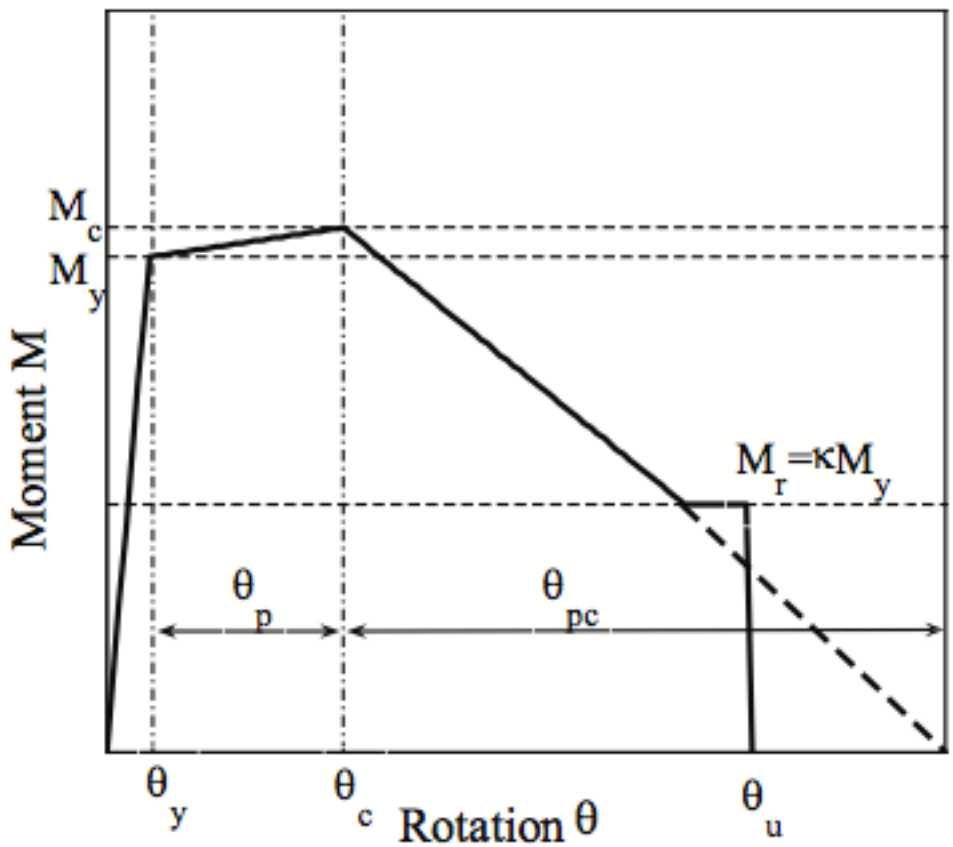
4.5 PARAMETERS IN UNIAXIAL BILINEAR MATERIAL

The modified Ibarra-Krawinkler model establishes strength bounds based on a monotonic backbone curve and a set of rules that define the characteristics of hysteretic behavior between the bounds. For a bilinear hysteretic response three

modes of cyclic deterioration are defined with respect to the backbone curve (basic strength, post-capping strength, and unloading/reloading stiffness deterioration). The model can be applied to any force-deformation relationship. The parameters are defined in terms of moment and rotation quantities, but they can be used in terms of force and displacement.

The backbone curve is defined by three strength parameters (My=effective yield moment, Mc=capping moment strength (or post yield strength ratio Mcc/My), and residual moment Mr= κ My) and four deformations parameters (θy=yield rotation, θp=pre-capping plastic rotation for monotonic loading (difference between yield rotation and rotation at maximum moment), θpc=post-capping plastic rotation (diference between rotation at maximum moment and rotation at complete loss of strength), and θu=ultimate rotation capacity).

The next graphic shows the defined parameters:



Graphic 19: Backbone curve

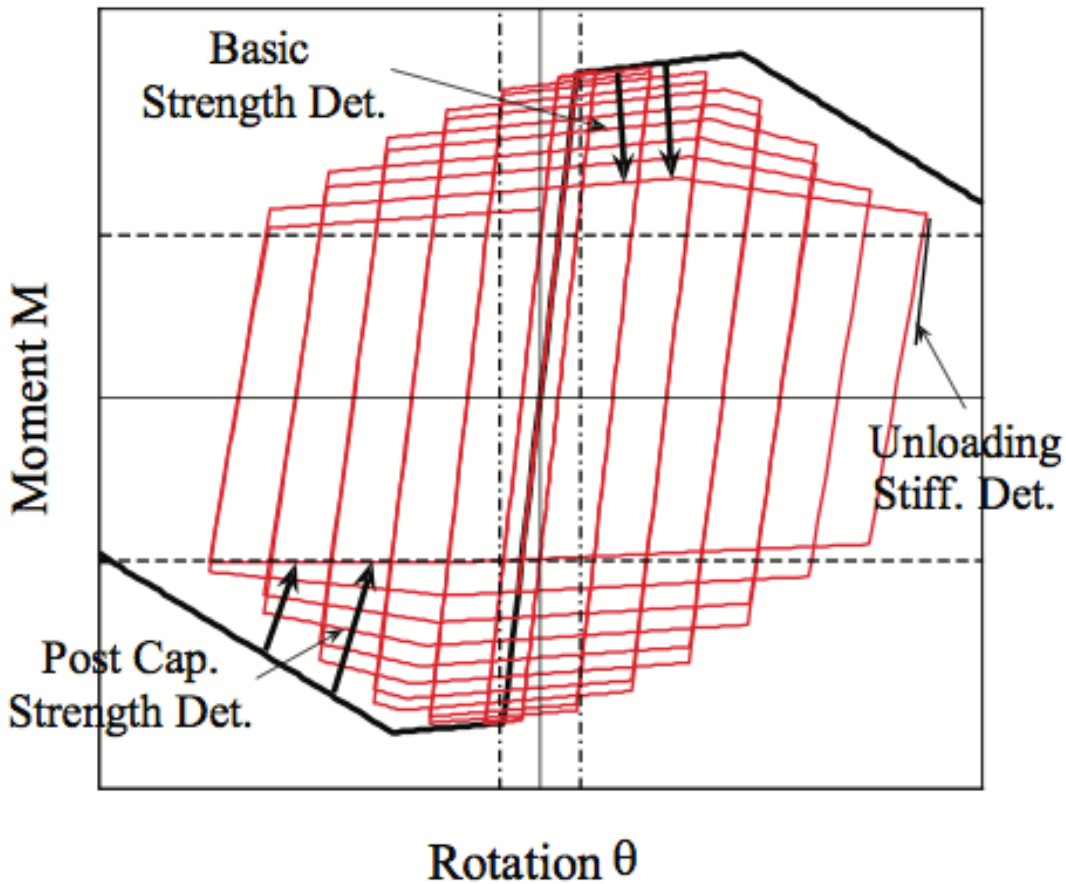
Reference cumulative plastic rotation ∧

The rates of cyclic deterioration are controlled by a rule developed by Rahnama and Krawinkler. This rule is based on the hysteretic energy dissipated when the component is subjected to cyclic loading. It is assumed that every component has a reference hysteretic energy dissipation capacity Et, which is an inherent property of the components regardless of the loading history applied to the component. The reference hysteretic energy dissipation capacity is expressed as a multiple of My· θp, i.e.,

$$Et = \lambda \cdot \theta p \cdot My \text{ or } Et = \wedge \cdot My$$

Where $\Lambda = \lambda \cdot \theta p$ is a reference cumulative rotation capacity, and θp and My are the pre-capping plastic rotation and effective yield strength of the component, respectively.

The effect of this deterioration after cyclic loading can be seen in the following figure:



Graphic 20: Effect of the deterioration after cyclic loading

Residual strength ratio κ

Low cycle fatige experimental studies indicate four ranges of cyclic deterioration: a range of negligible deterioration in which local instabilities have not yet occurred or are insignificant. The second range involves an almost constant rate of cyclic deterioration due to continuous growth of local buckles. In the third range,

deterioration proceeds at a very slow rate due to the stabilization in buckle size. This range is associated with the residual strength of a steel component. These ranges are followed by a range of very rapid deterioration, which is caused by crack propagation at local buckles (ductile tearing). A reasonable estimate of residual strength is 0.4 times the effective yield strength My.

Ultimate rotation capacity θu

At very large inelastic rotation cracks may develop in the steel base material close to the apex of the most sever local buckle, and rapid crack propagation will then occur followed by ductile tearing and essentially complete loss of strength. The modified IK deterioration model captures this failure mode with the ultimate rotation capacity θu. This rotation depends on the loading history and may be very large for cases in which only a few very large cycles are executed.

Other parameters

Other parameters such as λ_1, λ_2, λ_3, λ_4 and c_1, c_2, c_3 and c_4 will be determined by the use of a genetic algorithm among other parameters that are hard to determine.

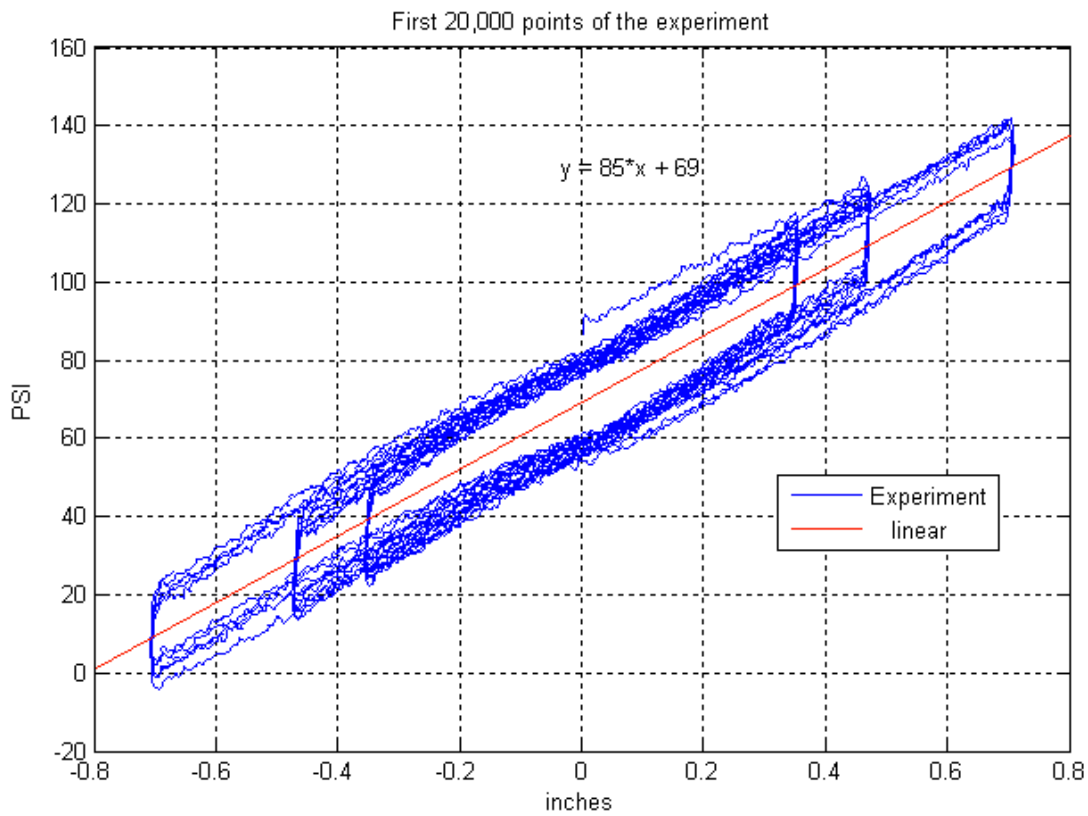
4.6 STUDY OF THE PARAMETERS

Once the model properties have been understood, let's see the effects of the values in the parameters in the model.

Considering a symmetric cycle, the model consists in 16 parameters. However, if it is possible to determine some of these parameters, or realize which ones do not interfere in the final hysteretic curve, the process of fitting the model to the experiment will be faster since less variables will be needed.

Elastic Stiffness K0

This parameter can be measured as the initial slope in the elastic part of the experimental cycle. In the following graphic the first 20,000 points out of 185,880 points have been drawn, and a linear fitting has been realized in order to determine K0.



Graphic 21: Determination of elastic stiffness.

Therefore the value for elastic stiffness K0=85 PSI/inch

A representation of what would represent a change of this parameter keeping the others constants is:

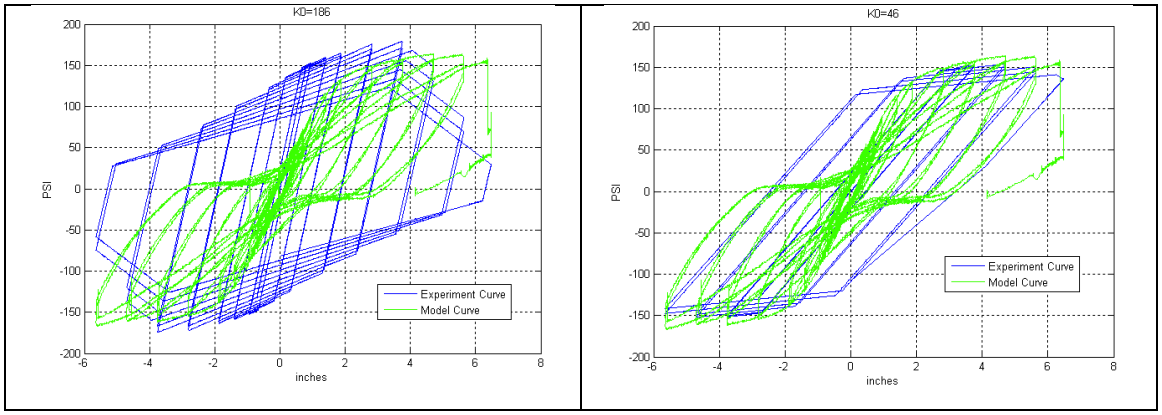


Table 1: Comparison of the behavior of the cycle for different K0s. At the left K0=186 and at the right K0=46

Strain hardening ratio α_s

This parameter has to be estimated at the fitting process of the curve by the GA. A difference in the value of this parameter implies a modification in the slope of the hardening plastic part of the material. For values lower than 0.001 the hardening is too flat, and values greater than 0.5 the slope is too high:

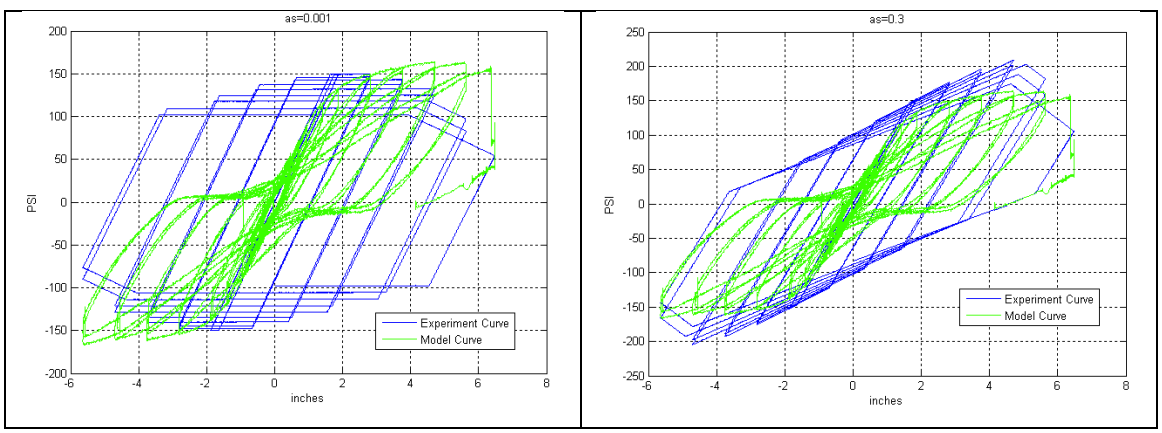
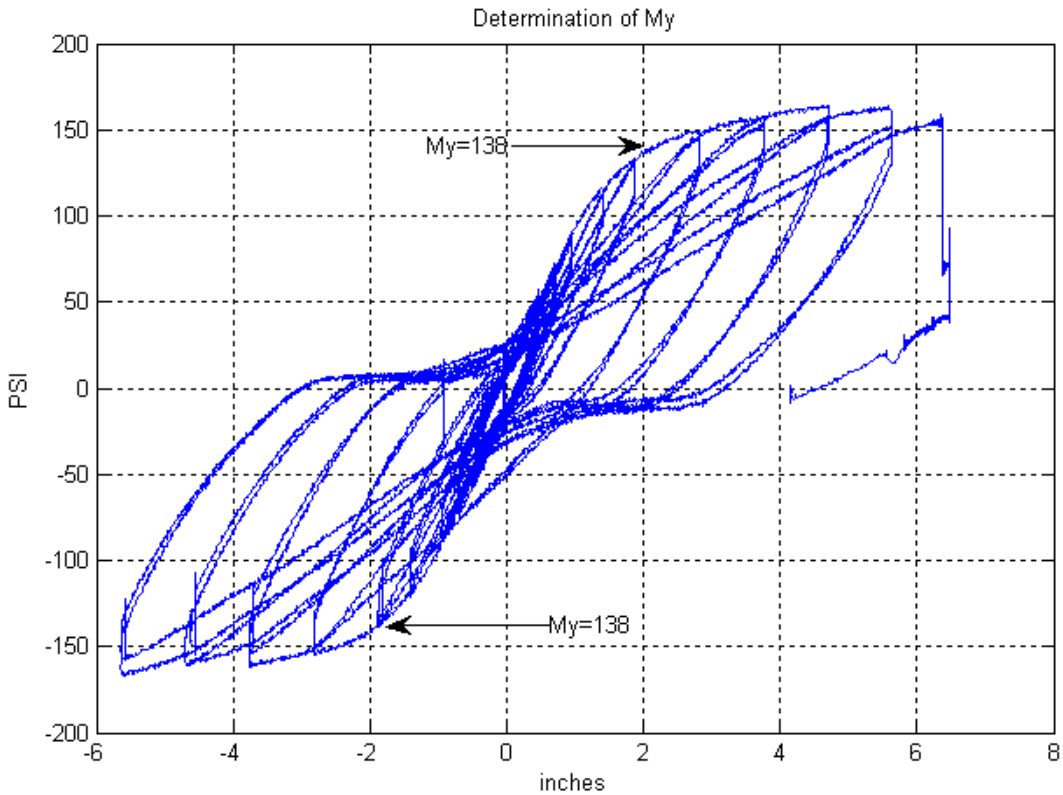


Table 2: Difference in the hardening ratio. With a value of 0.001 at the left and a value of 0.3 at the right

Effective yield strength My

OpenSees allows to create non symmetric cycles by defining a different value for the positive and the negative value of the yield strength. In this thesis, the cycles are considerd symmetric and therefore only one value of My needs to be determined.

This parameter can be determined from the experiment, using the available data. Therefore graphically one can determine the value of My=138 PSI:



Graphic 22: Determination of My

Notice that yield strength is a force and not a moment, but My has been chosen this way to follow the same nomenclature used in the description of the parameters and in the OpenSees instructions.

A change in the value of My implies the following effects in the reverse cycle graphic:

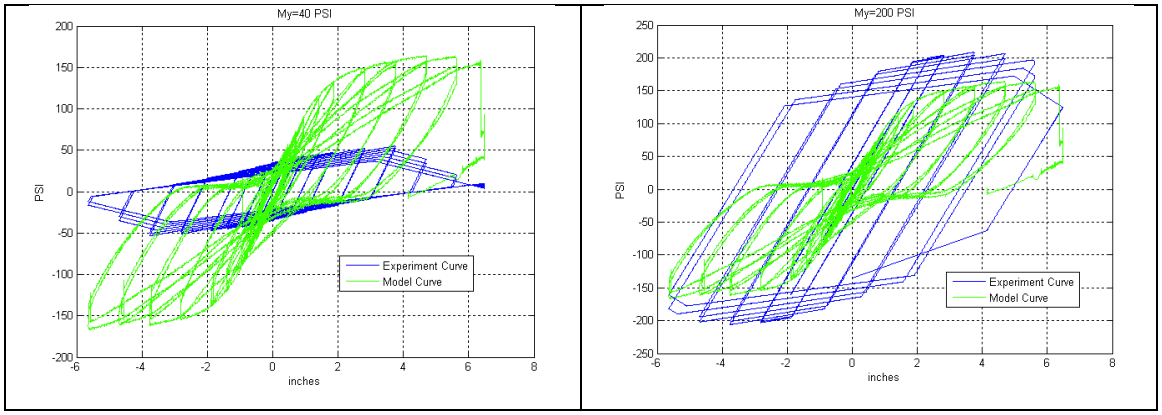


Table 3: Comparison of hysteretic cycle with different My. At the left My=40 PSI. At the right My=200 PSI

Cyclic deterioration parameter for strength deterioration. (Lamba S)

The variation in the value of this parameters causes a change in the cycle. Therefore this variable will be determined by the GA. However, values greater than 10,000 produces little or no changes in the cycle, and values less than 10 produces convergence problems.

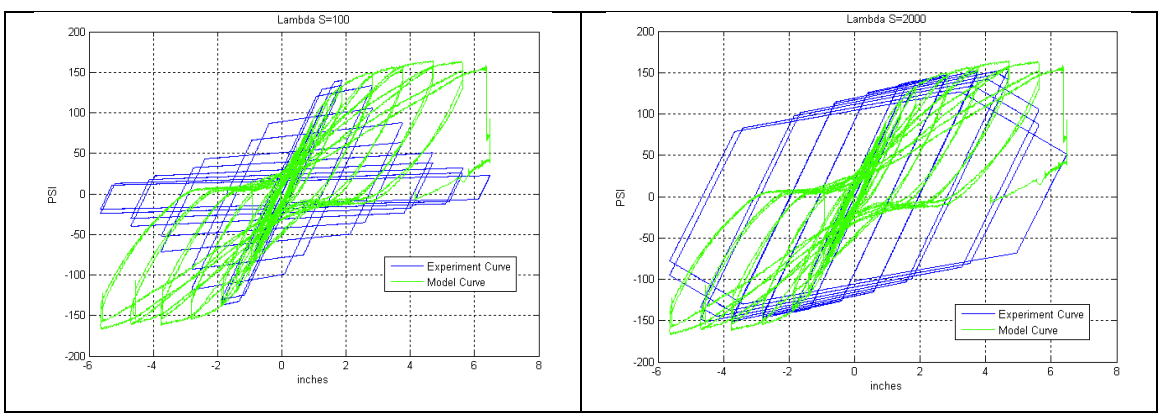


Table 4: Comparison of cyclic deterioration parameter for strength deterioration. At the left, model with lambda S=100 and at the right lambda S=2000

Cyclic deterioration parameter for post capping strength deterioration. (Lamba C)

Like the previous parameter, changes in lambda C also makes the cycle to vary, therefore this parameter will be considered as a variable in the GA. The variation of this parameter for values greater than 1000 produces very few or none modification in the cycle, and for values less than 10 produces convergence problems.

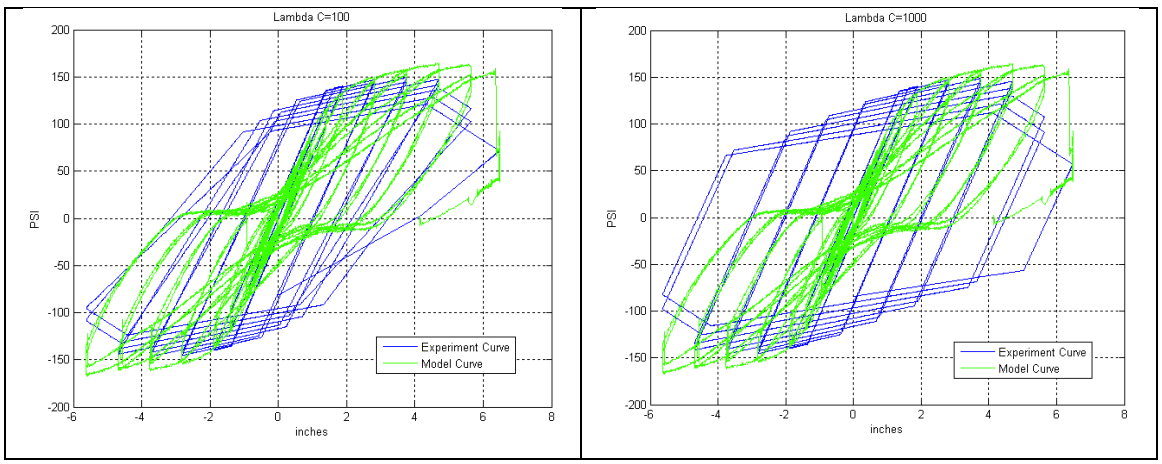


Table 5: Comparison of cyclic deterioration parameter for post-capping strength deterioration. At the left lambda C=100 and the right Lambda C=1000

Cyclic deterioration parameter for acceleration reloading stiffness deterioration. (Lamba A)

Big changes in this parameter produce no alteration of the cycle. Therefore, a random value will be assigned to this variable and it won't be a parameter for the GA

Cyclic deterioration parameter for unloading stiffness deterioration. (Lamba K)

This parameter also produces changes in the reverse cycle. It has been seen, anyway that for values higher than 10000, the changes are very small, and that for values lower than 100, the program starts having convergence problems. It will be a variable for the GA with determined boundaries.

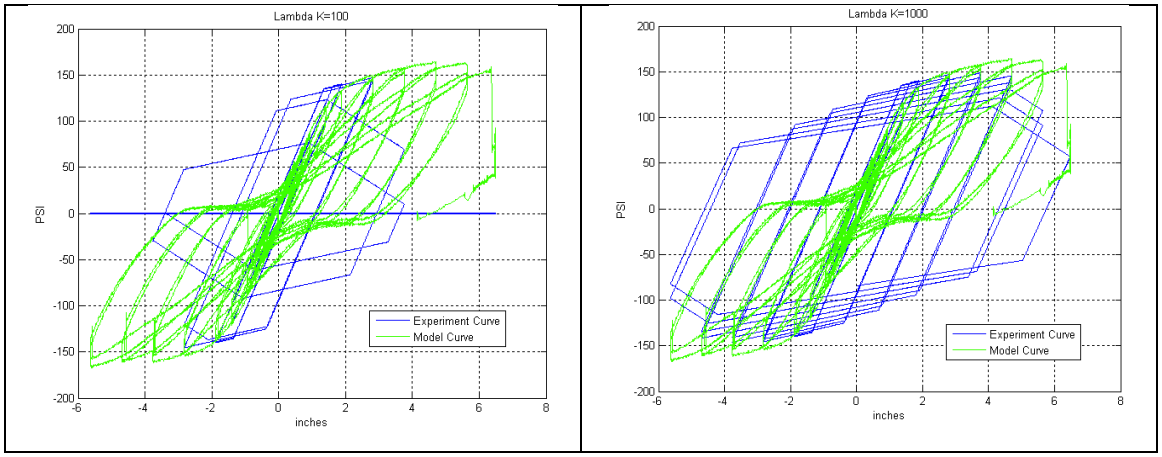


Table 6:Comparison of cyclic deterioration parameter for unloading stiffness deterioration. At the left, lambda k=100 at the right lambda K=1000

Rate of strength deterioration c_S

C_S presents important changes in the cycle, but for reasons of convergence its lowest value is 0.6 and because higher values do not produce important changes, the highest value given will be 2. However, when interacting with other variables, value of C_s lower than 1.0 makes the program not to converge.

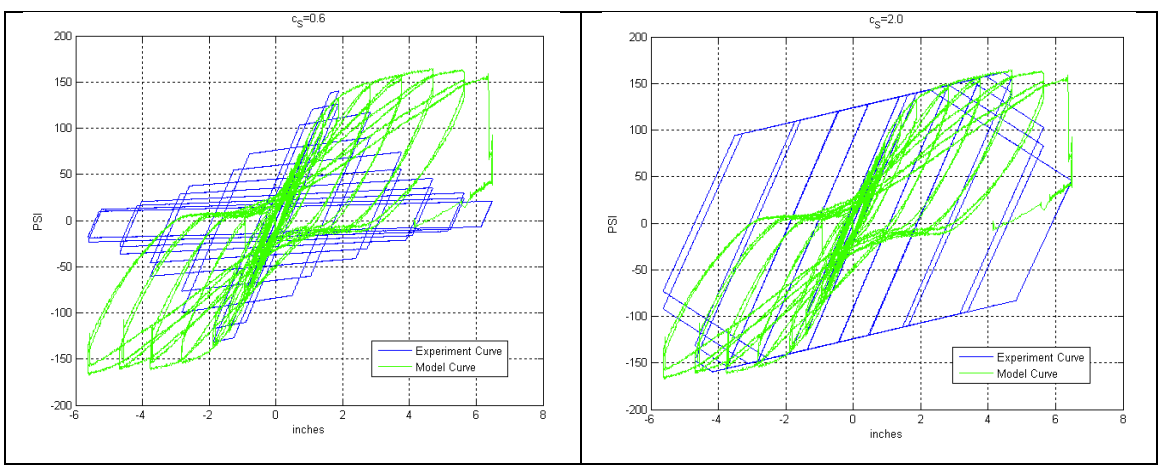


Table 7: Comparison of rate of strength deterioration. At the left C_S=0.6 at the right C_S=1.0

Rate of post-capping strength deterioration c_C

c_C makes the cycle to have no deterioration if the value is less than 1.0 and it affects almost nothing for values greater than 1.0. Therefore this value will be chosen 1.0 and will be a constant for the GA.

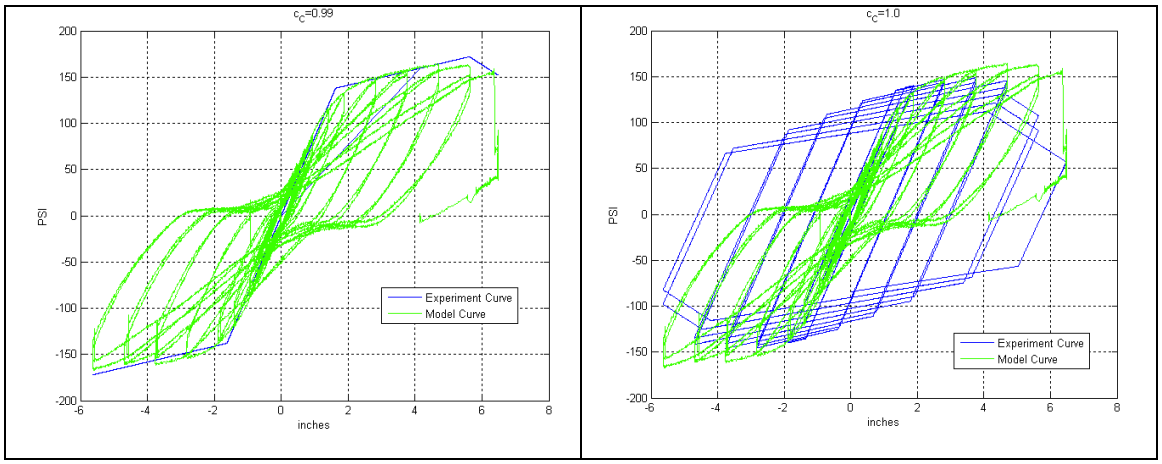


Table 8: Comparison of rate of post-capping strength deterioration.At the left c_C=0.99 at the right c_C=1

Rate of accelerated reloading deterioration c_A

Like Lambda A, this parameter does not produces any changes at all in the cycle, therefore a random value will be selected for the fitting.

Rate of unloading stiffness deterioration c_K

The variation of this parameter causes variations in the reverse cycle. Therefore it will be a variable to determine with the GA. It presents very few or no changes in the cycle after a value higher than 10. And presents convergence problems for values smaller than 0.5.

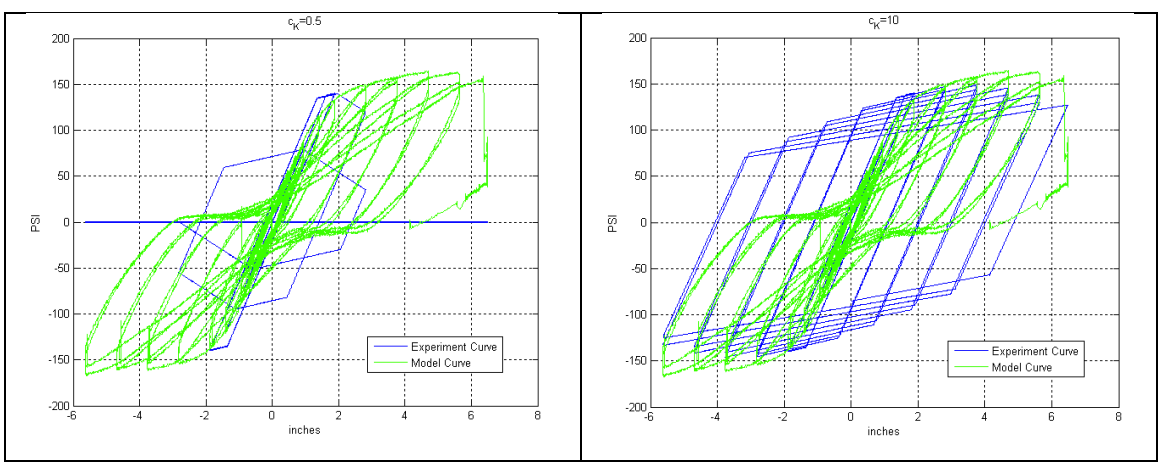


Table 9: Comparison of rate of unloading stiffness deterioration. At the left c_K=0.5 and at the right c_K=10.

Pre-capping rotation. θp

This will be a parameter to determine using the GA. Its upper limit will be the maximum deformation given in the experimental cycle. In the particular case used in this thesis this value is 6.0. The lower limit will be 0.1, since for lower values the changes produced in the cycle are very little.

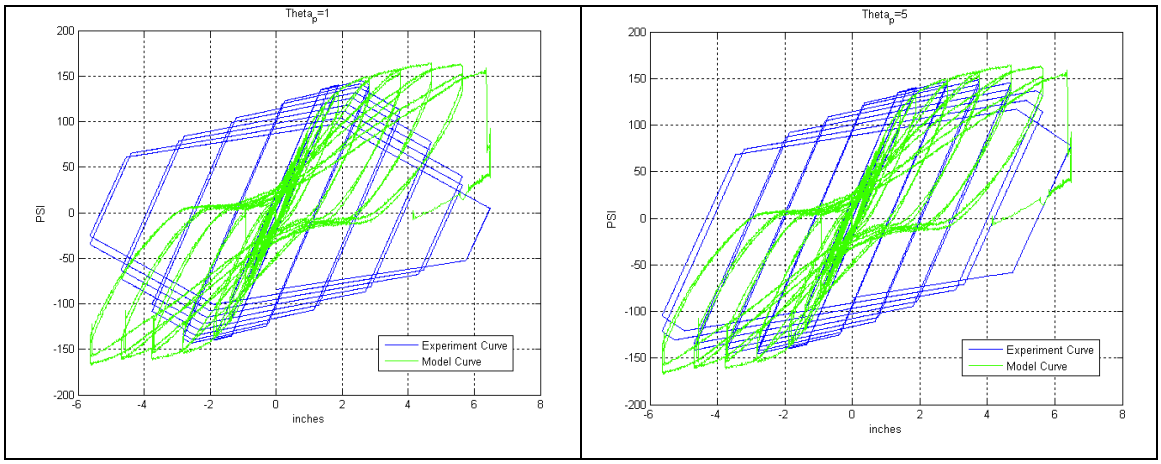


Table 10: Comparison of pre-capping rotation. At the left θp=1.0 and at the right θp=5.0

Post-capping rotation. θpc

This will be a value to determine with the GA. For values lower than 2 in this parameter, strange behavior of the cycle is found as seen in the left figure of the next table. And for values higher than 20 the cycle present almost no change.

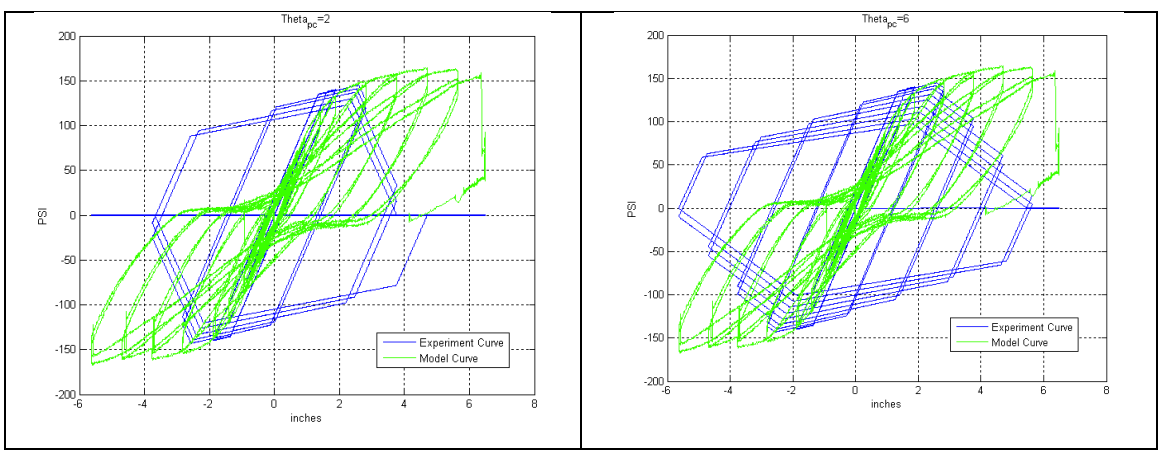


Table 11: Comparison of post-capping rotation. At the left θpc=2.0 and at the right θpc=6.0

Residual strength ratio. κ

Residual strength ratio will be a very small value since in the experiment we are not reaching ultimate rotation. Therefore this parameter will be assigned the value of 0.00001 and will be a constant value for the GA.

Ultimate rotation capacity. θu

The ultimate rotation capacity is not reached in the experiment, therefore it will be a constant parameter not to determine by using the GA. The value assigned has to be greater than the maximum deformation of the cycle. In this case a value of 8.0 has been chosen for it.

Rate of cyclic deterioration. D

Cyclic deterioration is one of the main parameters to be determined using the GA. For values less than 1.0 the deterioration is too small and for values greater than 60 it deforms the cycle making it have strange behavior.

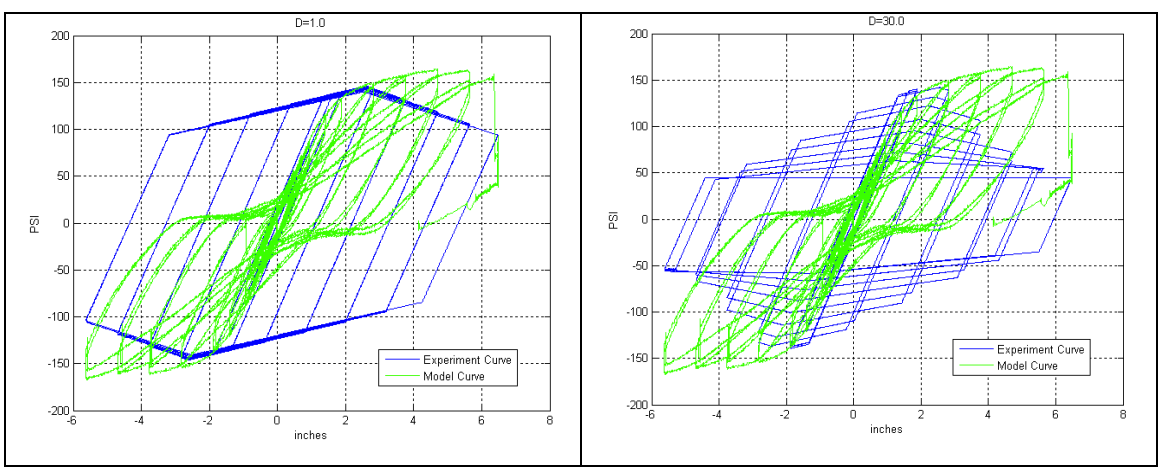


Table 12: Comparison of rate of cyclic deterioration. At the left D=1.0 at the right D=30.0

4.7 FINAL PARAMETERS SUMMARY TABLE

Once having analyzed the effect of the different parameters, we obtain the final summary table that tells which parameter is constant, and which are the boundaries for the variables ones.

Parameter	Type	Value	Boundaries
K0	Constant	85.0	n/a
α_s	Variable	n/a	[0.001 and 0.5]
My	Constant	138.0	n/a
λ_S	Variable	n/a	[10 and 10,000]
λ_C	Variable	n/a	[10 and 1,000]
λ_A	Constant	1,000	n/a
λ_K	Variable	n/a	[100 and 10,000]
C_S	Variable	n/a	[1.0 and 2.0]
C_C	Constant	1.0	n/a
C_A	Constant	1.0	n/a
C_K	Variable	n/a	[0.5 and 10.0]
θ_P	Variable	n/a	[0.1 and 6.0]
θ_{PC}	Variable	n/a	[2.0 and 20.0]
κ	Constant	0.00001	n/a
θ_U	Constant	8.0	n/a
D	Variable	n/a	[1 and 60.0]

Table 13: Summary table of all variables of the model and it's characteristics

4.8 GENETIC ALGORITHM FITTING FOR UNIAXIAL BILINEAR MATERIAL

The same idea developed in the three parameters GA fitting for a plastic material (steel 01) is going to be applied for the bilinear material.

Given initial values to the parameters to determine, the objective function will call OpenSees. OpenSees will load the same displacement protocol used in the experiment to the model defined by the given parameters, it will calculate the resultant forces after applying step by step in a static analysis the displacement control. Then the function will compare the modeled reverse cycle with the experimental one, and will obtain the difference using lest squares as:

$$\varepsilon = \sum_{i=1}^{n} (Yexp - Ymod)^2$$

Where Yexp and Ymod represents the force component of the experiment cycle in the case of Yexp and of the calculated response for the model in the case of Ymod.

Notice that to obtain the error, only the force coordinate has been used since the displacement coordinate is exactly the same for both the model and the experiment.

The objective function that the GA is going to use to minimize the error consists in the following steps once a set of variables in the form of a vector is given:

- First, it designs the zero length element writing in TCL language at a text file.
- Second, it calls the previously design code written in TCL language too, that plugs the displacements of the real experiment to the model.
- Third, it calls and runs the OpenSees, and the program calculates the forces.

- Finally, it compares the model and the experiment using least squares and gives a value of the accumulated error.

The code can be found in the appendix in #10

The variable vector is defined as P(P1,...P9):

P1	α_S
P2	λ_S
P3	λ_C
P4	λ_K
P5	C_S
P6	C_K
P7	θ_P
P8	θ_{PC}
P9	D

Table 14: Table that defines each component of the vector plugged into the GA

4.9 RESULTS

4.9.1 EXPERIMENT #1

Set the options in the optimization tool of Matlab, select, in order to arrive faster to the solution, a size of the initial population of 5 times the number of variables instead of the default option which is 20 times. Call the function to optimize, specify the number of variables and define the boundaries previously explained for each of the parameters.

In the following table 4 different iterations of the genetic algorithm are presented. The GA randomly chooses the value of the vector P, having in count the boundaries previously defined for each of the variables, and it generates the cycle. Then it

calculates the error in each iteration and mutates the values of the variables in order to obtain the minimum error and therefore, the value of the parameters of the models that bests simulate the real experiment.

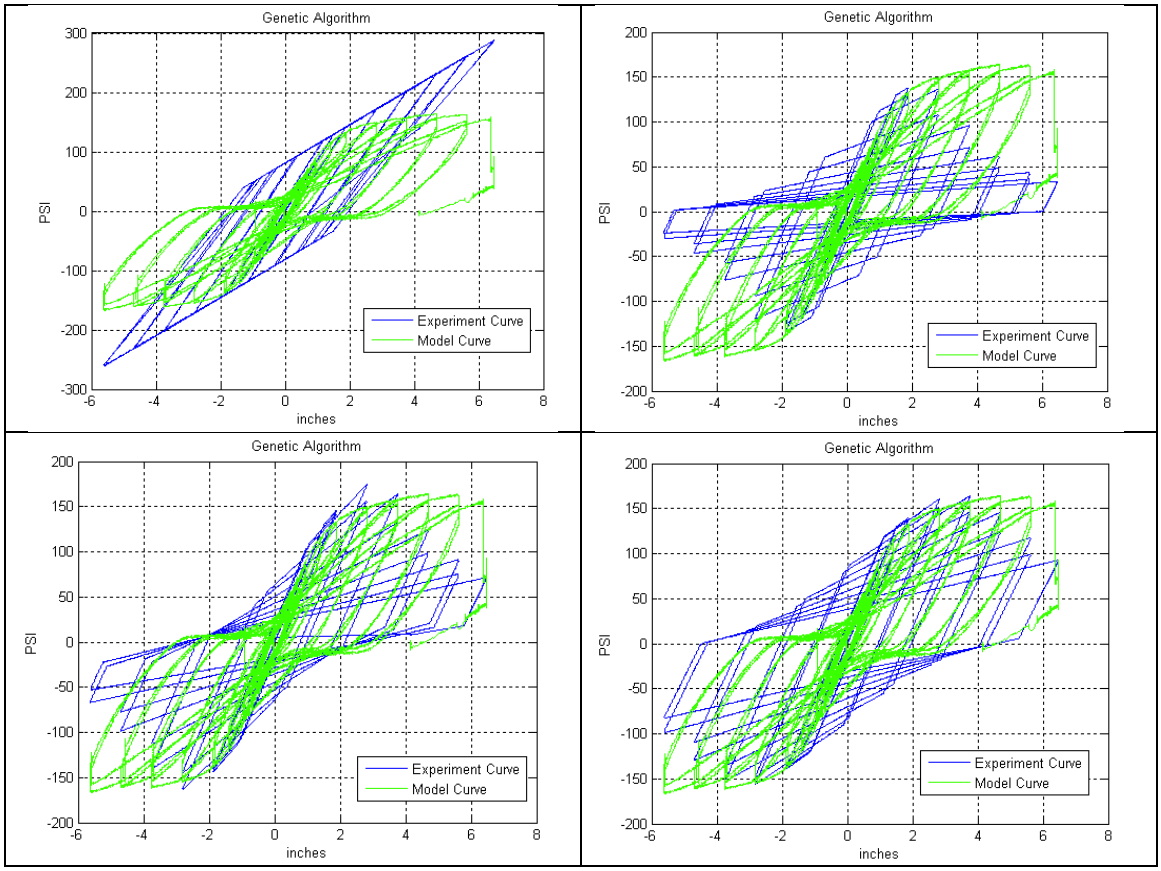


Table 15: Different moments representing different populations of the GA converging to the best solution

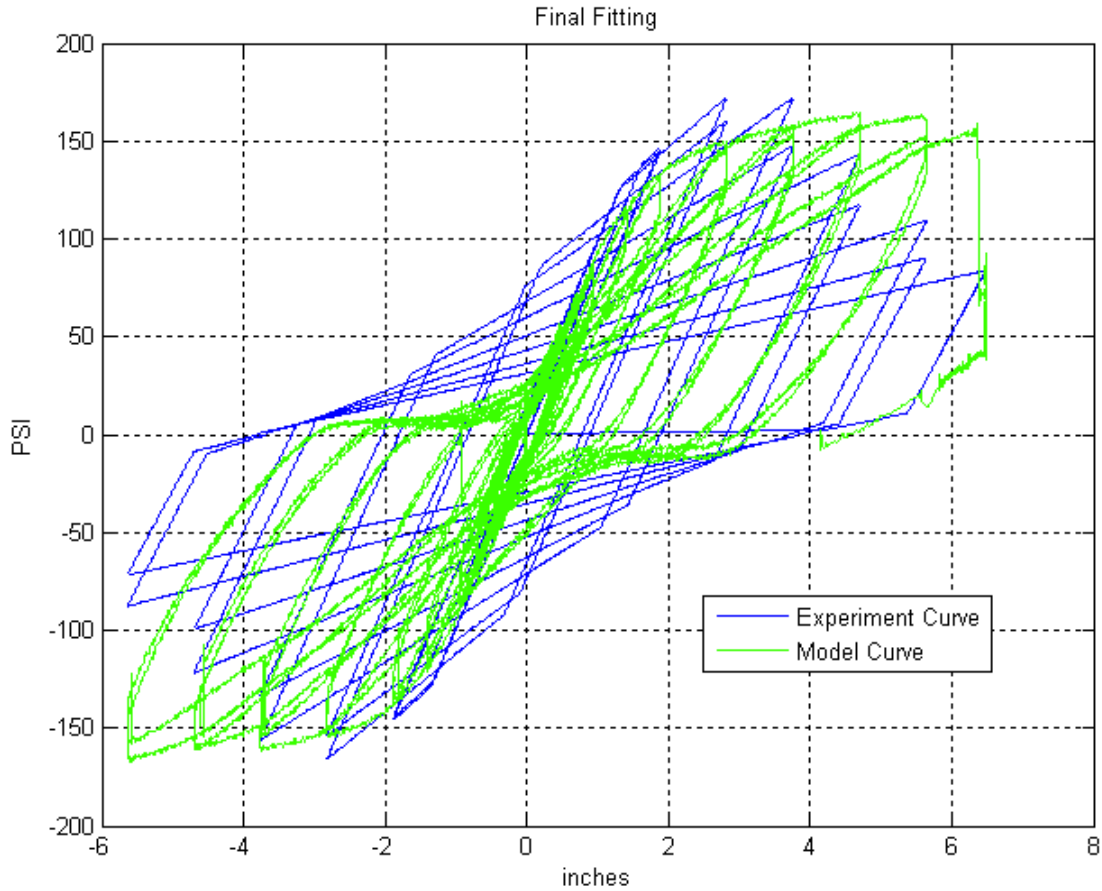
After 2 hours of calculations the results are:

P1	α_s	0.38
P2	λ_S	447.249
P3	λ_C	186.736
P4	λ_K	627.371
P5	C_S	1.098
P6	C_K	5.519
P7	θ_P	11.88
P8	θ_{PC}	20.704

P9	D	48.202

Table 16: Final results for experiment #1 after applying the GA

Graphically the fitting is:

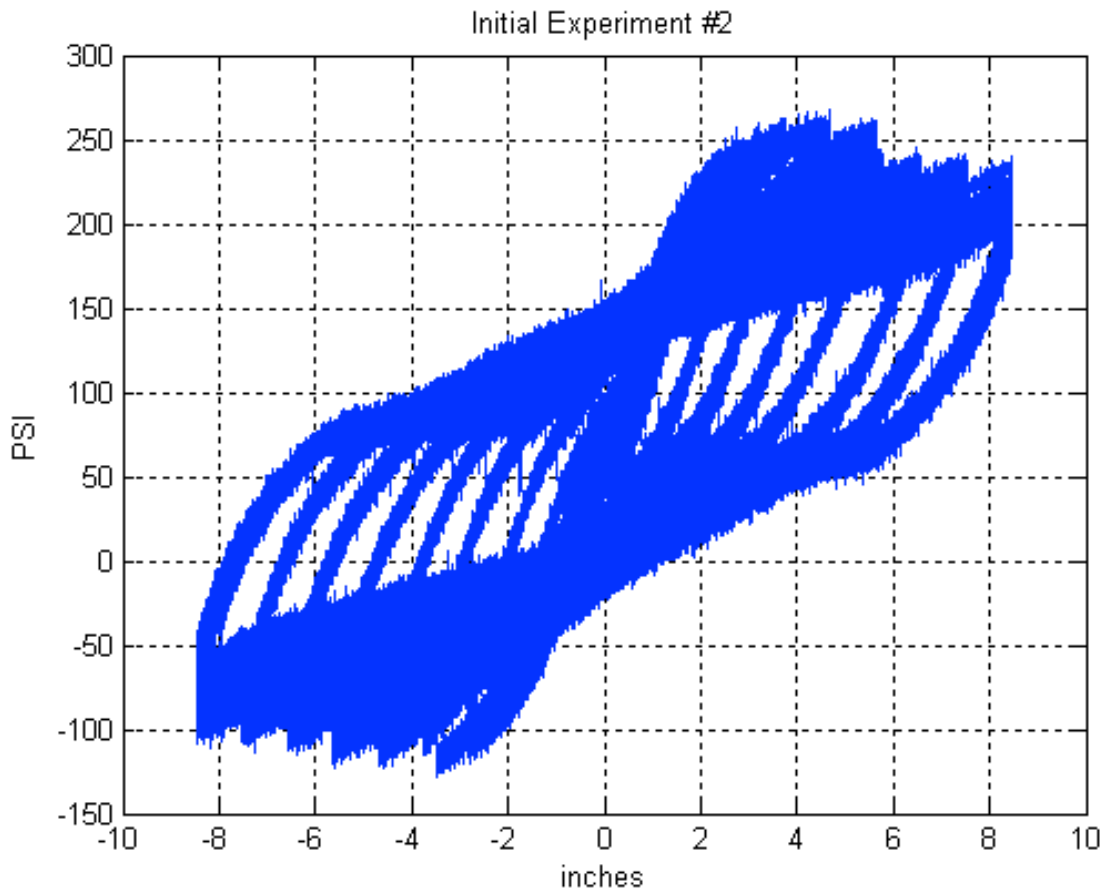


Graphic 23: Final fitting to the experimental test after applying the GA

4.9.2 EXPERIMENT #2

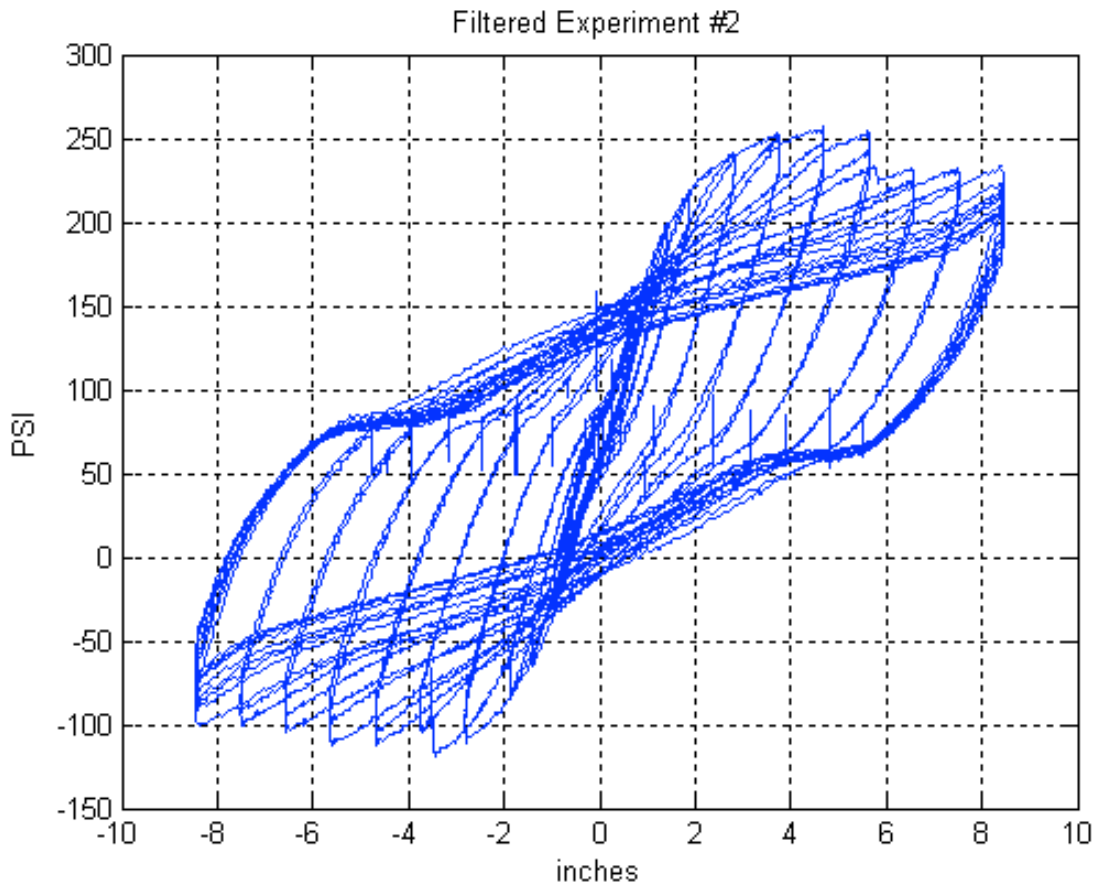
To corroborate the well running of the program, another three sets of data have been used and three new model curves have been calculated by using the same procedure and methodology as in experiment number 1. The second experiment is analyze in the following paragraphs.

The initial data before filtering looks like:



Graphic 24: Initial data of the second experiment

Applying the same algorithm used for experiment #1, and only changing the number of points, the filtered data is obtained:



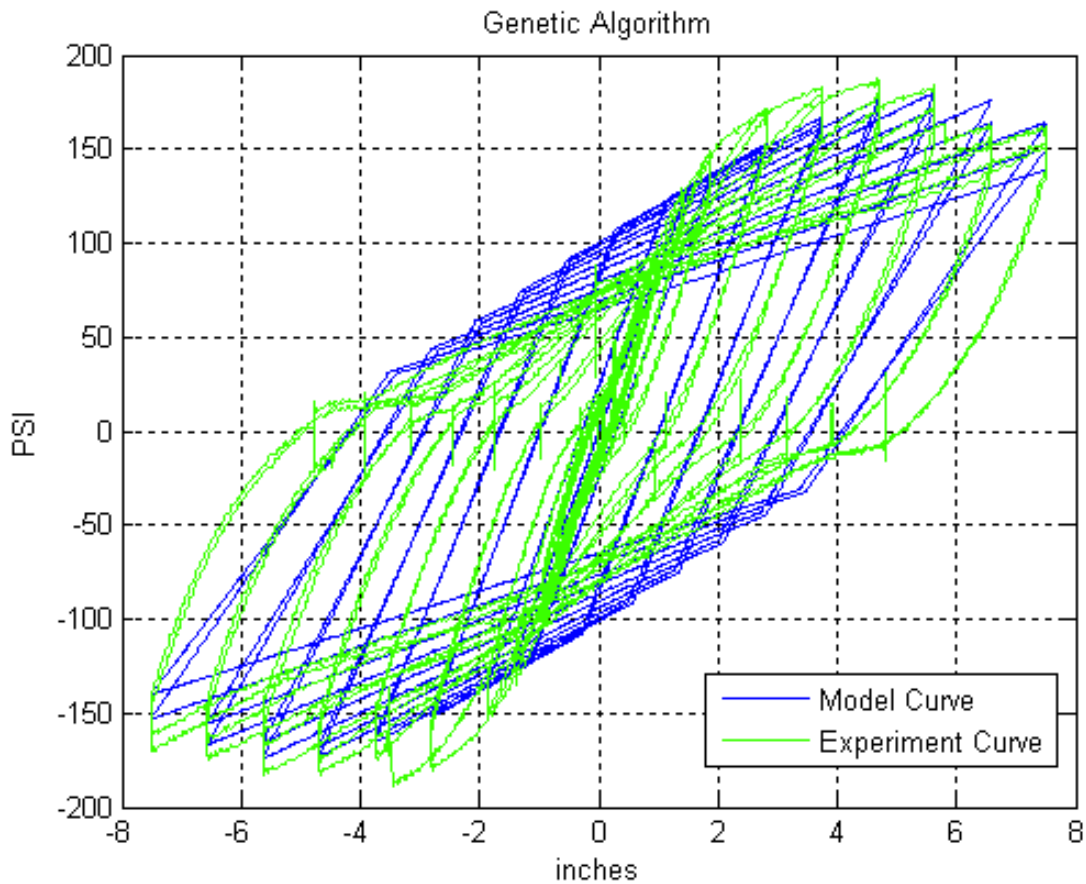
Graphic 25: Data of experiment #2 after filtering.

Once the data is ready, the code that uses the genethic algorithm technology is applied to the data. After 2 hours of calculations the result of the fitting is:

P1	α_s	0.207
P2	λ_S	335.724
P3	λ_C	210.39
P4	λ_K	579.472
P5	C_S	1.644
P6	C_K	3.515
P7	θ_P	11.542
P8	θ_{PC}	39.164

P9	D	40.875

Table 17: Final results for experiment #2.



Graphic 26: Fitting curve to experiment #2

4.9.3 EXPERIMENT #4

The third data set is used to make the third fitting of a model:

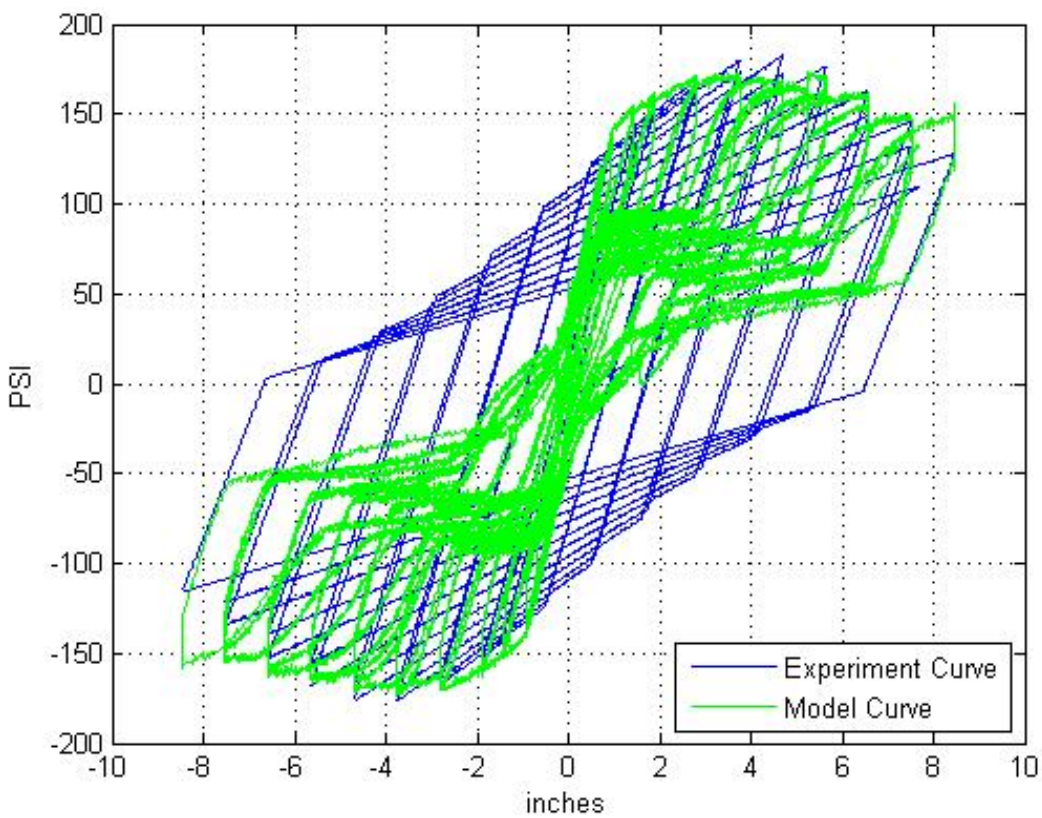
The value obtained for each parameter is:

P1	α_s	0.241
P2	λ_S	132.238
P3	λ_C	214.607
P4	λ_K	552.901
P5	C_S	1.3421

P6	C_K	5.180
P7	θ_P	4.668
P8	θ_{PC}	14.273
P9	D	21,8115

Table 18: Final results for experiment #3

The fitting is shown in the following graphic:



Graphic 27: Fitting curve to experiment #3

4.9.2 EXPERIMENT #4

The fourth data set is used to make the third fitting of a model:

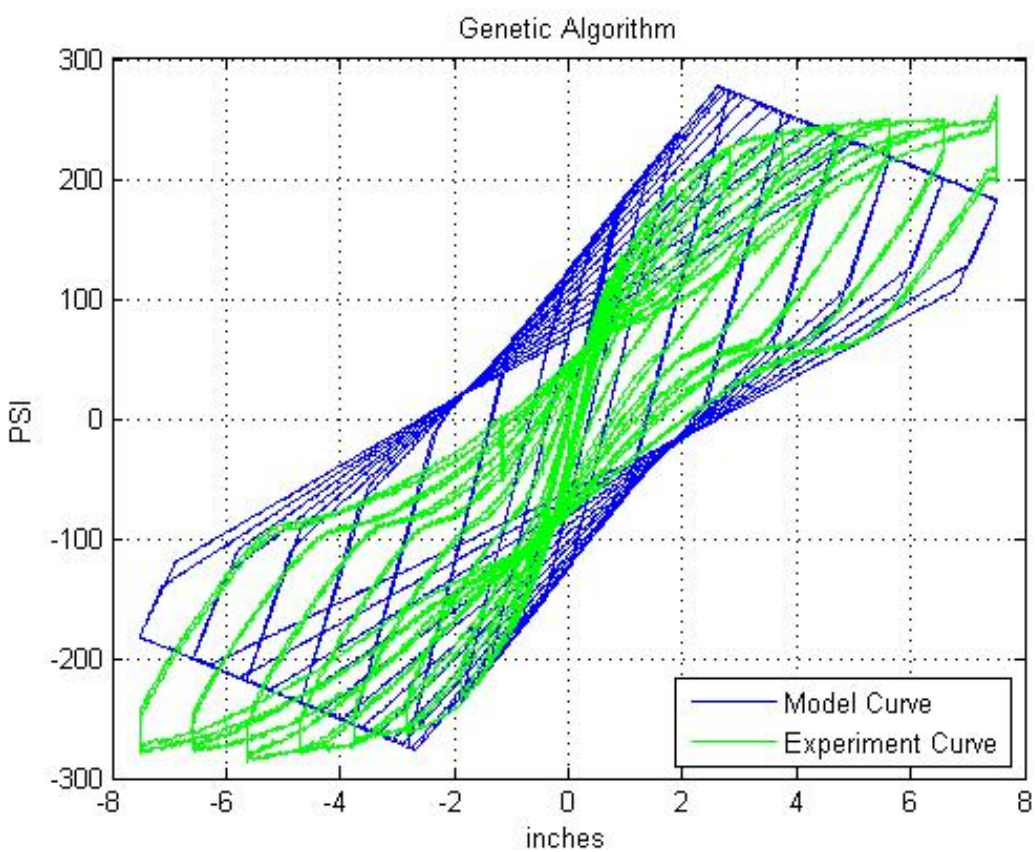
The value obtained for each parameter is:

P1	α_s	0.300

P2	λ_S	507.720
P3	λ_C	239.724
P4	λ_K	461.559
P5	C_S	1.513
P6	C_K	6.6488
P7	θ_P	1.6527
P8	θ_{PC}	17.0957
P9	D	70.4548

Table 19: Final results for experiment #3

The fitting is shown in the following graphic:



Graphic 28: Fitting curve to experiment #4

CHAPTER 5: FINDINGS, CONCLUSIONS AND FUTURE WORK

The fitting of the curves as can be seen at the end of chapter four is not perfect, but that is due that the model uses straight elements without curves or gradual changes of the slope. Therefore while using such a model, it will be not possible to achieve a perfect fitting.

However, the fitting represents fairly well the behavior of a real material. It achieves a very similar deterioration of the cycle, and it presents close values of strength and deformation.

The elastic parameters such as the Young modulus, the yielding strength, etc. can be easily determined graphically by the experimental results. Nonetheless, deteriorations parameters can't be determined graphically, but genetic algorithms help to find their values when properly applied in computational codes.

In order to run the program to generate a modeled curve fitting a real experiment it is necessary to open Matlab software. Copy in a column vector the displacements and strength of the real experiment that we want to fit the bilinear Opensees defined model. Apply the code for filtering the noise of the data. This code can be found in the appendix of the present thesis. Once the data is filtered, the values of the Young's modulus and the yielding strength can be obtained graphically. Then one can study the boundaries of the values of the different variable parameters that will be calculated with the genetic algorithm with a code that graphically overlaps the experimental graphic and the model in order to make the calculation time of the GA shorter. Finally open the multiobjective genetic algorithm function typing optimtool('ga') in Matlab. Select multiobjective GA. Call the function number 11 from the appendix with @Name_of_Function. Select the number of variables, in this case 9. Then select the lower and upper boundaries as [P1min P2min … P3min] and [P1max

P2max … P3max]. And everything is ready to calculate values of the deterioration parameters, lambdas and C's.

Some future work ideas are:

- The developing of a new model in OpenSees that has in consideration the non rectilinear elements of the real cycle.
- Application of the present work for calculation of real buildings or structures and the consideration of propitious security coefficients to be sure that the strength of the material is not overestimated.
- Study of the correlation between variables in order to adjust better boundaries and have a faster approach of the solution.

BIBLIOGRAPHY

Gomez, I., Kanvinde, A., Deierlein, G. (2009). Exposed column base conncections subjected to axial compression and flexure.

McKenna, F., Fenves, G.L. and Scott, M.H. (2000). Open System for Earthquake Engineering Simulation, University of California Berkeley, California, http://opensees.berkeley.edu.

University of Rhode Island Department of Electrical and Computer Engineering. ELE 436: Communication Systems. FFT Tutorial.

Liel, A. and Deierlein G.G. (2008). Assessing the collapse risk of California's existing reinforced concrete frame structures: Metrics for seismic safety decisions. The John A. Blume Earthquake Engineering Center.

NEES. (2011). Network for Earthquake Engineering Simulation. NEES Project Warehouse Database. Ultra-Low Cycle Fatigue and Fracture in Steel Structures.

TCL TK MANUAL. (2011). Tcl and tk manual pages. http://www.tcl.tk/man

Casado, M.C. (2008) Manual Básico de Matlab. Servicios Informaticos UCM. Apoyo a investigación y a docencia.

MATLAB Getting Started Guide. (2011) Copyright 1984-2011 by the MathWorks Inc.

Räihä, O. (2008). Applying Genetic Algorithms in Software Architecture Design.

MATHWORKS. (2011). Matlab Product Documentation. http://www.mathwork.com

APPENDIX

1 Code for the cantilever designed with steel01

```
# MODEL 1 TO FIT THE EXPERIMENT

# SET UP ------------------------------------------------------------

wipe;
model basic -ndm 2 -ndf 3;
file mkdir Data;

# DEFINE GEOMETRY---------------------------------------------------

set L 160.;    # Lenght of the column

node 1 0 0;
node 2 0 $L;

# BOUNDARY CONDITIONS-----------------------------------------------

fix 1 1 1 1;

# MATERIAL PROPERTIES-----------------------------------------------

set A 4.06;
set Iz 5.14;
set E 28560925.367;
set b 0.042;
set Fy 58475.141;
set Vy 58475.141;
set My 22500.;


uniaxialMaterial Steel01 1 $Fy [expr $A*$E] $b
uniaxialMaterial Steel01 2 $My [expr $E*$Iz] $b
uniaxialMaterial Steel01 3 $Vy [expr $E*$Iz] $b


# NODAL MASSES------------------------------------------------------

set WCol [expr $L * 119.6];

mass 1 [expr $WCol/2] 0. 0.
mass 2 [expr $WCol/2] 0. 0.


# DEFINE ELEMENTS---------------------------------------------------

set transfTag 1;
geomTransf Linear $transfTag;

# CONECTIVITY-------------------------------------------------------
```

```
# Define section

set secTag 1;
section Aggregator $secTag 1 P 2 Vy 3 Mz;

# Define elements

set numIntgrPts 5;
element nonlinearBeamColumn 1 1 2 $numIntgrPts $secTag $transfTag;


# DEFINE RECORDERS-----------------------------------------------------

recorder Node -file Data/RBase1.out -time -node 1 -dof 1 2 3 reaction;
recorder Node -file Data/Disp13.out -time -node 2 -dof 1 2 3 disp;


# DEFINE GRAVITY-------------------------------------------------------

pattern Plain 1 Linear {

      load 2 0. 2000. 0.
}

constraints Plain;
numberer Plain;
system BandGeneral;
test NormDispIncr 1.0e-8 6;
algorithm Newton;
integrator LoadControl 0.1;
analysis Static;
loadConst -time 0.0;


pattern Plain 2 Linear {

      load 2 1. 0. 0.
}

integrator DisplacementControl 2 1 0.01;
analyze 1000;

puts "Done!"
```

2 Code for the cantilever designed with steel02:

```
# MODEL OF AN EXPERIMENTAL COLUMN REPRESENTING A REAL MODEL WITH UKNOWN
PARAMETERS.

# SET UP ---------------------------------------------------------------

wipe;
model basic -ndm 2 -ndf 3;
file mkdir Data;
```

```
# DEFINE GEOMETRY-----------------------------------------------------

set L 160.;    # Lenght of the column

node 1 0 0;
node 2 0 $L;

# BOUNDARY CONDITIONS-------------------------------------------------

fix 1 1 1 1;

# MATERIAL PROPERTIES-------------------------------------------------

set A 4.06;
set Iz 5.14;
set E 29132476.;
set b 0.01032;
set Fy 67321.;
set Vy 66532.;
set My 12546.;
set R0 5;
set cR1 0.925;
set cR2 0.15;

uniaxialMaterial Steel02 1 $Fy [expr $A*$E] $b $R0 $cR1 $cR2
uniaxialMaterial Steel02 2 $My [expr $E*$Iz] $b $R0 $cR1 $cR2
uniaxialMaterial Steel02 3 $Vy [expr $E*$Iz] $b $R0 $cR1 $cR2


# NODAL MASSES--------------------------------------------------------

set WCol [expr $L * 119.6];

mass 1 [expr $WCol/2] 0. 0.
mass 2 [expr $WCol/2] 0. 0.


# DEFINE ELEMENTS-----------------------------------------------------

set transfTag 1;
geomTransf Linear $transfTag;

# CONECTIVITY---------------------------------------------------------

# Define section

set secTag 1;
section Aggregator $secTag 1 P 2 Vy 3 Mz;

# Define elements

set numIntgrPts 5;
element nonlinearBeamColumn 1 1 2 $numIntgrPts $secTag $transfTag;


# DEFINE RECORDERS----------------------------------------------------
```

```
recorder Node -file Data/RBase1.out -time -node 1 -dof 1 2 3 reaction;
recorder Node -file Data/Disp13.out -time -node 2 -dof 1 2 3 disp;


# DEFINE GRAVITY-----------------------------------------------------

pattern Plain 1 Linear {

      load 2 0. 2000. 0.
}


constraints Plain;
numberer Plain;
system BandGeneral;
test NormDispIncr 1.0e-8 6;
algorithm Newton;
integrator LoadControl 0.1;
analysis Static;
analyze 10;
loadConst -time 0.0;



#PUSHOVER-----------------------------------------------------------

pattern Plain 2 Linear {

      load 2 1. 0. 0.
}

integrator DisplacementControl 2 1 0.01;
analyze 1000;




puts "Done!"
```

3 Matlab Code for the first fitting model:

```
  fid = fopen('errors.txt', 'w+')
  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' )
  cd Data
  kk=0
  %Number or partitions n
  n=10
for i=1:(n+1)
    for j=1:(n+1)
        for k=1:(n+1)
            kk=kk+1;
    [tMod FxMod FyMod MMod ] = textread(['Reaction ',int2str(i),'
```

```
',int2str(j),' ',int2str(k),'.out'], '%f%f%f%f');
    dif=FxExp-FxMod;
    err=dif.*dif;

    sum=0;
            for m=1:1010
                 sum=sum+err(m);
            end
    vector(kk,:)=[sum i j k];



        end
    end
end
cd ..

 fclose(fid);

 a=find(vector(:,1)==min(vector(:,1)));
 vector(a,:);
 i=vector(a,2);
  j=vector(a,3);
   k=vector(a,4);
Emin=10000000;
Emax=30000000;
Vymin=50000;
Vymax=100000;
bmin=0.001;
bmax=0.1;

E=Emin+((Emax-Emin)/n)*(i-1)
Vy=Vymin+((Vymax-Vymin)/n)*(j-1)
b=bmin+((bmax-bmin)/n)*(k-1)
```

4 Matlab code for the three parameters fitting problem:

```
clear all

clc

% Code for a three parameter model using Opensees

%P1=E, P2=Vy and P3=b

P1min=25000000;
P1max=35000000;

P2min=60000;
```

```
P2max=70000;

P3min=0.001;
P3max=0.1;

% n is the number of iterations

tic

for n=1:50;

    %Function Get The Errors

    %Run Opensees for the eight different combos and get the
corresponding
    %

    %POINT 1

    E=P1min;
    Vy=P2min;
    b=P3min;


                     %Open TCL

                    fid=fopen('Model.tcl','w');

                % Write the code for Opensees in TCL language in the text
file.


                    fprintf(fid,'wipe;\n');
                    fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
                    fprintf(fid,'file mkdir Data;\n');

                 % Define the geometry of the column

                    fprintf(fid,'set L 160.;\n');
                    fprintf(fid,'node 1 0 0;\n');
                    fprintf(fid,'node 2 0 $L;\n');

                 % Define Boundary Conditions

                    fprintf(fid,'fix 1 1 1 1;\n');

                 % Define Material Properties

                    fprintf(fid,'set A 4.06;\n');
                    fprintf(fid,'set Iz 5.14;\n');
                    fprintf(fid,['set E ' num2str(P1min) ';\n']);
                    fprintf(fid,'set Fy 65000;\n');
                    fprintf(fid,['set Vy ' num2str(P2min) ';\n']);
```

```
                    fprintf(fid,'set My 12500;\n');
                    fprintf(fid,['set b ' num2str(P3min) ';\n']);


                    fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');

                    % Define Nodal Masses

                    fprintf(fid,'set WCol [expr $L * 119.6];\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');

                  % Define Elements and Conectivity



                    fprintf(fid,'set transfTag 1;\n');
                    fprintf(fid,'geomTransf Linear $transfTag;\n');

                    fprintf(fid,'set secTag 1;\n');
                    fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                    fprintf(fid,'set numIntgrPts 5;\n');
                    fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders


                    fprintf(fid,'recorder Node -file Data/Reaction.out -
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');

                  % Define Gravity

                    fprintf(fid,'pattern Plain 1 Linear {\n');
                    fprintf(fid,'   load 2 0. 2000. 0.;\n');
                    fprintf(fid,'}\n');

                  % Define Analysis Parameters (PUSHOVER)

                    fprintf(fid,'constraints Plain;\n');
                    fprintf(fid,'numberer Plain;;\n');
                    fprintf(fid,'system BandGeneral;\n');
                    fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                    fprintf(fid,'algorithm Newton;\n');
                    fprintf(fid,'integrator LoadControl 0.1;\n');
```

```
                    fprintf(fid,'analysis Static;\n');
                    fprintf(fid,'analyze 10;\n');
                    fprintf(fid,'loadConst -time 0.0;\n');

                    fprintf(fid,'pattern Plain 2 Linear {\n');
                    fprintf(fid,'   load 2 1. 0. 0.\n');
                    fprintf(fid,'}\n');


                    fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
                    fprintf(fid,'analyze 1000;\n');


                    fclose(fid);
                    dos('opensees model.tcl');



          % GET THE ERROR COMPARING TO THE SOLUTION


  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
  cd Data
  [tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
  dif=FxExp-FxMod;
  err=dif.*dif;
  err1=sum(err);
  cd ..

  err1



    % POINT 2

    E=P1max;
    Vy=P2min;
    b=P3min;


                %Open TCL

                   fid=fopen('Model.tcl','w');

                % Write the code for Opensees in TCL language in the text
file.


                    fprintf(fid,'wipe;\n');
                    fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
                    fprintf(fid,'file mkdir Data;\n');
```

```
                % Define the geometry of the column

                   fprintf(fid,'set L 160.;\n');
                   fprintf(fid,'node 1 0 0;\n');
                   fprintf(fid,'node 2 0 $L;\n');

                % Define Boundary Conditions

                   fprintf(fid,'fix 1 1 1 1;\n');

                % Define Material Properties

                   fprintf(fid,'set A 4.06;\n');
                   fprintf(fid,'set Iz 5.14;\n');
                   fprintf(fid,['set E ' num2str(P1max) ';\n']);
                   fprintf(fid,'set Fy 65000;\n');
                   fprintf(fid,['set Vy ' num2str(P2min) ';\n']);
                   fprintf(fid,'set My 12500;\n');
                   fprintf(fid,['set b ' num2str(P3min) ';\n']);


                   fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
                   fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
                   fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');

                   % Define Nodal Masses

                   fprintf(fid,'set WCol [expr $L * 119.6];\n');
                   fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
                   fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');

                  % Define Elements and Conectivity



                   fprintf(fid,'set transfTag 1;\n');
                   fprintf(fid,'geomTransf Linear $transfTag;\n');

                   fprintf(fid,'set secTag 1;\n');
                   fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                   fprintf(fid,'set numIntgrPts 5;\n');
                   fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders


                   fprintf(fid,'recorder Node -file Data/Reaction.out -
```

```
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');

                  % Define Gravity

                    fprintf(fid,'pattern Plain 1 Linear {\n');
                    fprintf(fid,'   load 2 0. 2000. 0.;\n');
                    fprintf(fid,'}\n');

                  % Define Analysis Parameters (PUSHOVER)

                    fprintf(fid,'constraints Plain;\n');
                    fprintf(fid,'numberer Plain;;\n');
                    fprintf(fid,'system BandGeneral;\n');
                    fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                    fprintf(fid,'algorithm Newton;\n');
                    fprintf(fid,'integrator LoadControl 0.1;\n');
                    fprintf(fid,'analysis Static;\n');
                    fprintf(fid,'analyze 10;\n');
                    fprintf(fid,'loadConst -time 0.0;\n');

                    fprintf(fid,'pattern Plain 2 Linear {\n');
                    fprintf(fid,'   load 2 1. 0. 0.\n');
                    fprintf(fid,'}\n');


                    fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
                    fprintf(fid,'analyze 1000;\n');


                    fclose(fid);
                    dos('opensees model.tcl');



          % GET THE ERROR COMPARING TO THE SOLUTION


  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
  cd Data
  [tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
  dif=FxExp-FxMod;
  err=dif.*dif;
  err2=sum(err);
  cd ..

  err2



      % POINT 3
```

```
    E=P1max;
    Vy=P2max;
    b=P3min;


                %Open TCL

                   fid=fopen('Model.tcl','w');

                % Write the code for Opensees in TCL language in the text
file.


                    fprintf(fid,'wipe;\n');
                    fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
                    fprintf(fid,'file mkdir Data;\n');

                 % Define the geometry of the column

                    fprintf(fid,'set L 160.;\n');
                    fprintf(fid,'node 1 0 0;\n');
                    fprintf(fid,'node 2 0 $L;\n');

                 % Define Boundary Conditions

                    fprintf(fid,'fix 1 1 1 1;\n');

                 % Define Material Properties

                    fprintf(fid,'set A 4.06;\n');
                    fprintf(fid,'set Iz 5.14;\n');
                    fprintf(fid,['set E ' num2str(P1max) ';\n']);
                    fprintf(fid,'set Fy 65000;\n');
                    fprintf(fid,['set Vy ' num2str(P2max) ';\n']);
                    fprintf(fid,'set My 12500;\n');
                    fprintf(fid,['set b ' num2str(P3min) ';\n']);


                    fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');

                    % Define Nodal Masses

                    fprintf(fid,'set WCol [expr $L * 119.6];\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');

                  % Define Elements and Conectivity
```

```
                    fprintf(fid,'set transfTag 1;\n');
                    fprintf(fid,'geomTransf Linear $transfTag;\n');

                    fprintf(fid,'set secTag 1;\n');
                    fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                    fprintf(fid,'set numIntgrPts 5;\n');
                    fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders


                    fprintf(fid,'recorder Node -file Data/Reaction.out -
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');

                  % Define Gravity

                    fprintf(fid,'pattern Plain 1 Linear {\n');
                    fprintf(fid,'   load 2 0. 2000. 0.;\n');
                    fprintf(fid,'}\n');

                  % Define Analysis Parameters (PUSHOVER)

                    fprintf(fid,'constraints Plain;\n');
                    fprintf(fid,'numberer Plain;;\n');
                    fprintf(fid,'system BandGeneral;\n');
                    fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                    fprintf(fid,'algorithm Newton;\n');
                    fprintf(fid,'integrator LoadControl 0.1;\n');
                    fprintf(fid,'analysis Static;\n');
                    fprintf(fid,'analyze 10;\n');
                    fprintf(fid,'loadConst -time 0.0;\n');

                    fprintf(fid,'pattern Plain 2 Linear {\n');
                    fprintf(fid,'   load 2 1. 0. 0.\n');
                    fprintf(fid,'}\n');


                    fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
                    fprintf(fid,'analyze 1000;\n');


                    fclose(fid);
                    dos('opensees model.tcl');
```

```
        % GET THE ERROR COMPARING TO THE SOLUTION


[tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
cd Data
[tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
dif=FxExp-FxMod;
err=dif.*dif;
err3=sum(err);
cd ..

err3









%POINT 4

  E=P1min;
  Vy=P2max;
  b=P3min;


               %Open TCL

                fid=fopen('Model.tcl','w');

              % Write the code for Opensees in TCL language in the text
file.


                 fprintf(fid,'wipe;\n');
                 fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
                 fprintf(fid,'file mkdir Data;\n');

               % Define the geometry of the column

                 fprintf(fid,'set L 160.;\n');
                 fprintf(fid,'node 1 0 0;\n');
                 fprintf(fid,'node 2 0 $L;\n');

               % Define Boundary Conditions

                 fprintf(fid,'fix 1 1 1 1;\n');

               % Define Material Properties

                 fprintf(fid,'set A 4.06;\n');
```

```
                    fprintf(fid,'set Iz 5.14;\n');
                    fprintf(fid,['set E ' num2str(P1min) ';\n']);
                    fprintf(fid,'set Fy 65000;\n');
                    fprintf(fid,['set Vy ' num2str(P2max) ';\n']);
                    fprintf(fid,'set My 12500;\n');
                    fprintf(fid,['set b ' num2str(P3min) ';\n']);


                    fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');

                    % Define Nodal Masses

                    fprintf(fid,'set WCol [expr $L * 119.6];\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');

                  % Define Elements and Conectivity



                    fprintf(fid,'set transfTag 1;\n');
                    fprintf(fid,'geomTransf Linear $transfTag;\n');

                    fprintf(fid,'set secTag 1;\n');
                    fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                    fprintf(fid,'set numIntgrPts 5;\n');
                    fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders


                    fprintf(fid,'recorder Node -file Data/Reaction.out -
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');

                  % Define Gravity

                    fprintf(fid,'pattern Plain 1 Linear {\n');
                    fprintf(fid,'   load 2 0. 2000. 0.;\n');
                    fprintf(fid,'}\n');

                  % Define Analysis Parameters (PUSHOVER)

                    fprintf(fid,'constraints Plain;\n');
                    fprintf(fid,'numberer Plain;;\n');
```

```
                    fprintf(fid,'system BandGeneral;\n');
                    fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                    fprintf(fid,'algorithm Newton;\n');
                    fprintf(fid,'integrator LoadControl 0.1;\n');
                    fprintf(fid,'analysis Static;\n');
                    fprintf(fid,'analyze 10;\n');
                    fprintf(fid,'loadConst -time 0.0;\n');

                    fprintf(fid,'pattern Plain 2 Linear {\n');
                    fprintf(fid,'   load 2 1. 0. 0.\n');
                    fprintf(fid,'}\n');


                    fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
                    fprintf(fid,'analyze 1000;\n');


                    fclose(fid);
                    dos('opensees model.tcl');



          % GET THE ERROR COMPARING TO THE SOLUTION


  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
  cd Data
  [tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
  dif=FxExp-FxMod;
  err=dif.*dif;
  err4=sum(err);
  cd ..

  err4


    % POINT 5

    E=P1min;
    Vy=P2min;
    b=P3max;


                   %Open TCL

                   fid=fopen('Model.tcl','w');

                % Write the code for Opensees in TCL language in the text
file.


                    fprintf(fid,'wipe;\n');
```

```
                    fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
                    fprintf(fid,'file mkdir Data;\n');

                 % Define the geometry of the column

                    fprintf(fid,'set L 160.;\n');
                    fprintf(fid,'node 1 0 0;\n');
                    fprintf(fid,'node 2 0 $L;\n');

                 % Define Boundary Conditions

                    fprintf(fid,'fix 1 1 1 1;\n');

                 % Define Material Properties

                    fprintf(fid,'set A 4.06;\n');
                    fprintf(fid,'set Iz 5.14;\n');
                    fprintf(fid,['set E ' num2str(P1min) ';\n']);
                    fprintf(fid,'set Fy 65000;\n');
                    fprintf(fid,['set Vy ' num2str(P2min) ';\n']);
                    fprintf(fid,'set My 12500;\n');
                    fprintf(fid,['set b ' num2str(P3max) ';\n']);


                    fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');

                    % Define Nodal Masses

                    fprintf(fid,'set WCol [expr $L * 119.6];\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');

                  % Define Elements and Conectivity



                    fprintf(fid,'set transfTag 1;\n');
                    fprintf(fid,'geomTransf Linear $transfTag;\n');

                    fprintf(fid,'set secTag 1;\n');
                    fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                    fprintf(fid,'set numIntgrPts 5;\n');
                    fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders
```

```
                    fprintf(fid,'recorder Node -file Data/Reaction.out -
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');

                  % Define Gravity

                    fprintf(fid,'pattern Plain 1 Linear {\n');
                    fprintf(fid,'   load 2 0. 2000. 0.;\n');
                    fprintf(fid,'}\n');

                  % Define Analysis Parameters (PUSHOVER)

                    fprintf(fid,'constraints Plain;\n');
                    fprintf(fid,'numberer Plain;;\n');
                    fprintf(fid,'system BandGeneral;\n');
                    fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                    fprintf(fid,'algorithm Newton;\n');
                    fprintf(fid,'integrator LoadControl 0.1;\n');
                    fprintf(fid,'analysis Static;\n');
                    fprintf(fid,'analyze 10;\n');
                    fprintf(fid,'loadConst -time 0.0;\n');

                    fprintf(fid,'pattern Plain 2 Linear {\n');
                    fprintf(fid,'   load 2 1. 0. 0.\n');
                    fprintf(fid,'}\n');


                    fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
                    fprintf(fid,'analyze 1000;\n');


                    fclose(fid);
                    dos('opensees model.tcl');



          % GET THE ERROR COMPARING TO THE SOLUTION


  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
  cd Data
  [tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
  dif=FxExp-FxMod;
  err=dif.*dif;
  err5=sum(err);
  cd ..

  err5
```

```
% POINT 6

E=P1max;
Vy=P2min;
b=P3max;


%Open TCL

fid=fopen('Model.tcl','w');

% Write the code for Opensees in TCL language in the text
file.


fprintf(fid,'wipe;\n');
fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
fprintf(fid,'file mkdir Data;\n');

% Define the geometry of the column

fprintf(fid,'set L 160.;\n');
fprintf(fid,'node 1 0 0;\n');
fprintf(fid,'node 2 0 $L;\n');

% Define Boundary Conditions

fprintf(fid,'fix 1 1 1 1;\n');

% Define Material Properties

fprintf(fid,'set A 4.06;\n');
fprintf(fid,'set Iz 5.14;\n');
fprintf(fid,['set E ' num2str(P1max) ';\n']);
fprintf(fid,'set Fy 65000;\n');
fprintf(fid,['set Vy ' num2str(P2min) ';\n']);
fprintf(fid,'set My 12500;\n');
fprintf(fid,['set b ' num2str(P3max) ';\n']);


fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');

% Define Nodal Masses

fprintf(fid,'set WCol [expr $L * 119.6];\n');
fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
```

```
                  % Define Elements and Conectivity



                    fprintf(fid,'set transfTag 1;\n');
                    fprintf(fid,'geomTransf Linear $transfTag;\n');

                    fprintf(fid,'set secTag 1;\n');
                    fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                    fprintf(fid,'set numIntgrPts 5;\n');
                    fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders


                    fprintf(fid,'recorder Node -file Data/Reaction.out -
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');

                  % Define Gravity

                    fprintf(fid,'pattern Plain 1 Linear {\n');
                    fprintf(fid,'   load 2 0. 2000. 0.;\n');
                    fprintf(fid,'}\n');

                  % Define Analysis Parameters (PUSHOVER)

                    fprintf(fid,'constraints Plain;\n');
                    fprintf(fid,'numberer Plain;;\n');
                    fprintf(fid,'system BandGeneral;\n');
                    fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                    fprintf(fid,'algorithm Newton;\n');
                    fprintf(fid,'integrator LoadControl 0.1;\n');
                    fprintf(fid,'analysis Static;\n');
                    fprintf(fid,'analyze 10;\n');
                    fprintf(fid,'loadConst -time 0.0;\n');

                    fprintf(fid,'pattern Plain 2 Linear {\n');
                    fprintf(fid,'   load 2 1. 0. 0.\n');
                    fprintf(fid,'}\n');


                    fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
                    fprintf(fid,'analyze 1000;\n');


                    fclose(fid);
                    dos('opensees model.tcl');
```

```
        % GET THE ERROR COMPARING TO THE SOLUTION


  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
  cd Data
  [tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
  dif=FxExp-FxMod;
  err=dif.*dif;
  err6=sum(err);
  cd ..

  err6

    % POINT 7

    E=P1max;
    Vy=P2max;
    b=P3max;


                    %Open TCL

                    fid=fopen('Model.tcl','w');

                  % Write the code for Opensees in TCL language in the text
file.


                     fprintf(fid,'wipe;\n');
                     fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
                     fprintf(fid,'file mkdir Data;\n');

                  % Define the geometry of the column

                     fprintf(fid,'set L 160.;\n');
                     fprintf(fid,'node 1 0 0;\n');
                     fprintf(fid,'node 2 0 $L;\n');

                  % Define Boundary Conditions

                     fprintf(fid,'fix 1 1 1 1;\n');

                  % Define Material Properties

                     fprintf(fid,'set A 4.06;\n');
                     fprintf(fid,'set Iz 5.14;\n');
                     fprintf(fid,['set E ' num2str(P1max) ';\n']);
                     fprintf(fid,'set Fy 65000;\n');
                     fprintf(fid,['set Vy ' num2str(P2max) ';\n']);
                     fprintf(fid,'set My 12500;\n');
```

```
                    fprintf(fid,['set b ' num2str(P3max) ';\n']);


                    fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');

                    % Define Nodal Masses

                    fprintf(fid,'set WCol [expr $L * 119.6];\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');

                  % Define Elements and Conectivity



                    fprintf(fid,'set transfTag 1;\n');
                    fprintf(fid,'geomTransf Linear $transfTag;\n');

                    fprintf(fid,'set secTag 1;\n');
                    fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                    fprintf(fid,'set numIntgrPts 5;\n');
                    fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders


                    fprintf(fid,'recorder Node -file Data/Reaction.out -
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');

                  % Define Gravity

                    fprintf(fid,'pattern Plain 1 Linear {\n');
                    fprintf(fid,'   load 2 0. 2000. 0.;\n');
                    fprintf(fid,'}\n');

                  % Define Analysis Parameters (PUSHOVER)

                    fprintf(fid,'constraints Plain;\n');
                    fprintf(fid,'numberer Plain;;\n');
                    fprintf(fid,'system BandGeneral;\n');
                    fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                    fprintf(fid,'algorithm Newton;\n');
                    fprintf(fid,'integrator LoadControl 0.1;\n');
                    fprintf(fid,'analysis Static;\n');
```

```
                    fprintf(fid,'analyze 10;\n');
                    fprintf(fid,'loadConst -time 0.0;\n');

                    fprintf(fid,'pattern Plain 2 Linear {\n');
                    fprintf(fid,'   load 2 1. 0. 0.\n');
                    fprintf(fid,'}\n');


                    fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
                    fprintf(fid,'analyze 1000;\n');


                    fclose(fid);
                    dos('opensees model.tcl');



          % GET THE ERROR COMPARING TO THE SOLUTION


  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
  cd Data
  [tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
  dif=FxExp-FxMod;
  err=dif.*dif;
  err7=sum(err);
  cd ..

  err7


    % POINT 8

    E=P1min;
    Vy=P2max;
    b=P3max;

                   %Open TCL

                   fid=fopen('Model.tcl','w');

                % Write the code for Opensees in TCL language in the text
file.


                    fprintf(fid,'wipe;\n');
                    fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
                    fprintf(fid,'file mkdir Data;\n');

                 % Define the geometry of the column

                    fprintf(fid,'set L 160.;\n');
```

```
                    fprintf(fid,'node 1 0 0;\n');
                    fprintf(fid,'node 2 0 $L;\n');

                 % Define Boundary Conditions

                    fprintf(fid,'fix 1 1 1 1;\n');

                 % Define Material Properties

                    fprintf(fid,'set A 4.06;\n');
                    fprintf(fid,'set Iz 5.14;\n');
                    fprintf(fid,['set E ' num2str(P1min) ';\n']);
                    fprintf(fid,'set Fy 65000;\n');
                    fprintf(fid,['set Vy ' num2str(P2max) ';\n']);
                    fprintf(fid,'set My 12500;\n');
                    fprintf(fid,['set b ' num2str(P3max) ';\n']);


                    fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');

                    % Define Nodal Masses

                    fprintf(fid,'set WCol [expr $L * 119.6];\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');

                  % Define Elements and Conectivity



                    fprintf(fid,'set transfTag 1;\n');
                    fprintf(fid,'geomTransf Linear $transfTag;\n');

                    fprintf(fid,'set secTag 1;\n');
                    fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                    fprintf(fid,'set numIntgrPts 5;\n');
                    fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders


                    fprintf(fid,'recorder Node -file Data/Reaction.out -
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');
```

```
                    % Define Gravity

                      fprintf(fid,'pattern Plain 1 Linear {\n');
                      fprintf(fid,'   load 2 0. 2000. 0.;\n');
                      fprintf(fid,'}\n');

                    % Define Analysis Parameters (PUSHOVER)

                      fprintf(fid,'constraints Plain;\n');
                      fprintf(fid,'numberer Plain;;\n');
                      fprintf(fid,'system BandGeneral;\n');
                      fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                      fprintf(fid,'algorithm Newton;\n');
                      fprintf(fid,'integrator LoadControl 0.1;\n');
                      fprintf(fid,'analysis Static;\n');
                      fprintf(fid,'analyze 10;\n');
                      fprintf(fid,'loadConst -time 0.0;\n');

                      fprintf(fid,'pattern Plain 2 Linear {\n');
                      fprintf(fid,'   load 2 1. 0. 0.\n');
                      fprintf(fid,'}\n');


                      fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
                      fprintf(fid,'analyze 1000;\n');


                      fclose(fid);
                      dos('opensees model.tcl');



          % GET THE ERROR COMPARING TO THE SOLUTION


  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
  cd Data
  [tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
  dif=FxExp-FxMod;
  err=dif.*dif;
  err8=sum(err);
  cd ..

  err8



    %Function Get the New Points

    err_vector=[err1, err2, err3, err4, err5, err6, err7, err8]
    errmin=min(err_vector)
```

```
  if errmin==err1

  P1min=P1min
  P1max=P1min+(P1max-P1min)*1/2

  P2min=P2min
  P2max=P2min+(P2max-P2min)*1/2

  P3min=P3min
  P3max=P3min+(P3max-P3min)*1/2


elseif errmin==err2

  P1min=P1min+(P1max-P1min)*1/2
  P1max=P1max

  P2min=P2min
  P2max=P2min+(P2max-P2min)*1/2

  P3min=P3min
  P3max=P3min+(P3max-P3min)*1/2


elseif errmin==err3

  P1min=P1min+(P1max-P1min)*1/2
  P1max=P1max

  P2min= P2min+(P2max-P2min)*1/2
  P2max=P2max

  P3min=P3min
  P3max=P3min+(P3max-P3min)*1/2

elseif errmin==err4

  P1min=P1min
  P1max=P1min+(P1max-P1min)*1/2

  P2min= P2min+(P2max-P2min)*1/2
  P2max=P2max

  P3min=P3min
  P3max=P3min+(P3max-P3min)*1/2

elseif errmin==err5

  P1min=P1min
  P1max=P1min+(P1max-P1min)*1/2
```

```
    P2min=P2min
    P2max=P2min+(P2max-P2min)*1/2

    P3min=P3min+(P3max-P3min)*1/2
    P3max=P3max

  elseif errmin==err6

    P1min=P1min+(P1max-P1min)*1/2
    P1max=P1max

    P2min=P2min
    P2max=P2min+(P2max-P2min)*1/2

    P3min=P3min+(P3max-P3min)*1/2
    P3max=P3max

  elseif errmin==err7

    P1min=P1min+(P1max-P1min)*1/2
    P1max=P1max

    P2min= P2min+(P2max-P2min)*1/2
    P2max=P2max

    P3min=P3min+(P3max-P3min)*1/2
    P3max=P3max


  elseif errmin==err8

    P1min=P1min
    P1max=P1min+(P1max-P1min)*1/2

    P2min= P2min+(P2max-P2min)*1/2
    P2max=P2max

    P3min=P3min+(P3max-P3min)*1/2
    P3max=P3max

    end



  n=n+1;



end


P1final=0.5*(P1min+P1max)
```

```
P2final=0.5*(P2min+P2max)
P3final=0.5*(P3min+P3max)

ERROR_MAX=(1/(2*(n-1)))*((P1max-P1min)^2+(P2max-P2min)^2+(P3max-
P3min)^2)^0.5

toc
```

5- Matlab code for the objective function of the 3 parameters fitting using GA:

```
function err1 = myfun(P)

         %Open TCL

                    fid=fopen('Model.tcl','w');

                % Write the code for Opensees in TCL language in the text
file.


                    fprintf(fid,'wipe;\n');
                    fprintf(fid,'model basic -ndm 2 -ndf 3;\n');
                    fprintf(fid,'file mkdir Data;\n');

                 % Define the geometry of the column

                    fprintf(fid,'set L 160.;\n');
                    fprintf(fid,'node 1 0 0;\n');
                    fprintf(fid,'node 2 0 $L;\n');

                 % Define Boundary Conditions

                    fprintf(fid,'fix 1 1 1 1;\n');

                 % Define Material Properties

                    fprintf(fid,'set A 4.06;\n');
                    fprintf(fid,'set Iz 5.14;\n');
                    fprintf(fid,['set E ' num2str(P(1)) ';\n']);
                    fprintf(fid,'set Fy 65000;\n');
                    fprintf(fid,['set Vy ' num2str(P(2)) ';\n']);
                    fprintf(fid,'set My 12500;\n');
                    fprintf(fid,['set b ' num2str(P(3)) ';\n']);


                    fprintf(fid,'uniaxialMaterial Steel01 1 $Fy [expr
$A*$E] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 2 $My [expr
$E*$Iz] $b;\n');
                    fprintf(fid,'uniaxialMaterial Steel01 3 $Vy [expr
$E*$Iz] $b;\n');
```

```
                    % Define Nodal Masses

                    fprintf(fid,'set WCol [expr $L * 119.6];\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');
                    fprintf(fid,'mass 1 [expr $WCol/2] 0. 0.;\n');

                  % Define Elements and Conectivity



                    fprintf(fid,'set transfTag 1;\n');
                    fprintf(fid,'geomTransf Linear $transfTag;\n');

                    fprintf(fid,'set secTag 1;\n');
                    fprintf(fid,'section Aggregator $secTag 1 P 2 Vy 3
Mz;\n');

                    fprintf(fid,'set numIntgrPts 5;\n');
                    fprintf(fid,'element nonlinearBeamColumn 1 1 2
$numIntgrPts $secTag $transfTag;\n');

                  % Define Recorders


                    fprintf(fid,'recorder Node -file Data/Reaction.out -
time -node 1 -dof 1 2 3 reaction;\n');
                    fprintf(fid,'recorder Node -file
Data/Displacement.out -time -node 2 -dof 1 2 3 disp;\n');

                  % Define Gravity

                    fprintf(fid,'pattern Plain 1 Linear {\n');
                    fprintf(fid,'   load 2 0. 2000. 0.;\n');
                    fprintf(fid,'}\n');

                  % Define Analysis Parameters (PUSHOVER)

                    fprintf(fid,'constraints Plain;\n');
                    fprintf(fid,'numberer Plain;;\n');
                    fprintf(fid,'system BandGeneral;\n');
                    fprintf(fid,'test NormDispIncr 1.0e-8 6;\n');
                    fprintf(fid,'algorithm Newton;\n');
                    fprintf(fid,'integrator LoadControl 0.1;\n');
                    fprintf(fid,'analysis Static;\n');
                    fprintf(fid,'analyze 10;\n');
                    fprintf(fid,'loadConst -time 0.0;\n');

                    fprintf(fid,'pattern Plain 2 Linear {\n');
                    fprintf(fid,'   load 2 1. 0. 0.\n');
                    fprintf(fid,'}\n');


                    fprintf(fid,'integrator DisplacementControl 2 1
0.01;\n');
```

```
                    fprintf(fid,'analyze 1000;\n');


                    fclose(fid);
                    dos('opensees model.tcl');



          % GET THE ERROR COMPARING TO THE SOLUTION


  [tExp FxExp FyExp MExp ] = textread('experiment.out', '%f%f%f%f' );
  cd Data
  [tMod FxMod FyMod MMod ] = textread(['Reaction.out'],'%f%f%f%f');
  dif=FxExp-FxMod;
  err=dif.*dif;
  err1=sum(err);
  cd ..
  err1

end
```

6 Matlab code for the frequency analysis:

```
% obtains the analysis of frequencies

N=length(y);

F1=[-N/2:N/2-1]/N;

X1=fftshift(X);


plot(F1,X1,'-x'),title('N=185880'),axis([-0.5 0.5 0 2000000])

grid on
title('Frequency Analysis')
```

7 Matlab code that filters the data

```
% filter the data

load stressStrain.mat
N=length(y);
figure
plot(x,y);
```

```
X=fft(y,N);
X1=fftshift(X);
X1(1:65000)=0
X1(105000:end)=0


Xt=ifftshift(X1);
Xt2=ifft(Xt);

figure
plot(x,Xt2)
```

8 OpenSees TCL code defining the model

```
# ------------------------------------------------------------
# zeroLength element with Bilin Materail
#------------------------------------------------------------

# SET UP ---------------------------------------------------------------
------------
wipe; # clear memory of all past model definitions
model BasicBuilder -ndm 2 -ndf 3; # Define the model builder,
ndm=#dimension, ndf=#dofs

# nodal coordinates:
node 1 0 0; # node#, X, Y
node 2 0 0

# Single point constraints -- Boundary Conditions
fix 1 1 1 1; # node DX DY RZ
fix 2 0 1 1; # node DX DY RZ

# Define ELEMENTS ------------------------------------------------------
------
# Material parameters
set matTagB 1

set K0 100000.
set as_Plus 0.1
set My_Plus 200000.
set Lamda_S 10000.
set Lamda_C 1000.
set Lamda_A 1000.
set Lamda_K 1000.
set c_S 1.0
set c_C 1.0
set c_A 1.0
set c_K 1.0
set theta_p_Plus 2.
set theta_pc_Plus 2.0
set Res_Pos 0.4
set theta_u_Plus 4.
set D_Plus 50;
```

```
uniaxialMaterial Bilin $matTagB $K0 $as_Plus $as_Plus $My_Plus [expr -
$My_Plus] $Lamda_S $Lamda_C $Lamda_A $Lamda_K $c_S $c_C $c_A $c_K \
$theta_p_Plus $theta_p_Plus $theta_pc_Plus $theta_pc_Plus $Res_Pos
$Res_Pos $theta_u_Plus $theta_u_Plus $D_Plus $D_Plus



element zeroLength 1 1 2 -mat $matTagB -dir 1

# Create recorder
recorder Node -file Displacement.out -time -node 2 -dof 1 disp;
recorder Element -file Reaction.out -time -ele 1 force;
source analizar.tcl
```

9 OpenSees code for the load of the displacement control of the experimental data and the analysis

```
# INPUT OF TXT DISPLACEMENTS FILE AND ANALYSIS


set SupportNode 2;
set GMdirection "1"



set History "Series -dt 0.05 -filePath DispControl.out"
pattern Plain 10 $History {
      sp 2 1 1
}



constraints Penalty 1e15 1e15;
# how it handles boundary conditions
numberer Plain;
# renumber dof's to minimize band-width (optimization), if you want to
system BandGeneral;
# how to store and solve the system of equations in the analysis
test NormDispIncr 1.0e-8 6 ;
# determine if convergence has been achieved at the end of an iteration
step
algorithm Newton;
# use Newton's solution algorithm: updates tangent stiffness at every
iteration
integrator LoadControl 0.05
analysis Static
# define type of analysis static or transient
analyze 181465 0.05;
# perform gravity analysis
loadConst -time 0.0;
# hold gravity constant and restart time



puts "Displacements input Done. End Time: [getTime]"
```

10 Matlab code to study the effect in the value of the parameters

```
% Code to study the different effect in the value of the parameters

P=[1:16];

%K0
P(1)=86.;
%as_Plus
P(2)=0.1;
%My_Plus
P(3)=150.;
%Lamda_S This has to be variable in GA from 10 to 10000
P(4)=1000.;
%Lamda_C; This has to be a variable too from 10 to 1000
P(5)=1000.;
%Lamda_A; Constant value, no changes no matter what number it is
P(6)=1000;
%Lamda_K; Variable in GA from 100 to 10000
P(7)=1000.;
%c_S; Variable from 0.6 to 2
P(8)=1.0;
%c_C This value can't be less than 1 and doesn't affect the cycle
P(9)=1.0;
%c_A This value doesnt affect the cycle
P(10)=1.0;
%c_K vary between 0.1 and 10
P(11)=1.0;
%theta_p_Plus vary between 1 and 10
P(12)=4.0;
%theta_pc_Plus vary between 2 and 20
P(13)=8.0;
%Res_Pos. It is a constant with a small value
P(14)=0.00001;
%theta_u_Plus It is a constant greater than 8.0
P(15)=8.;
%D_Plus. variable in GA from 1 to 50.
P(16)=10.;


fid=fopen('bilin.tcl','w');

% Write the code for Opensees in TCL language in the text file.


fprintf(fid,'wipe;\n');
fprintf(fid,'model BasicBuilder -ndm 2 -ndf 3;\n');

% nodal coordinates

fprintf(fid,'node 1 0 0;\n');
fprintf(fid,'node 2 0 0;\n');
```

```
%single point constraints - Boundary Conditions

fprintf(fid,'fix 1 1 1 1;\n');
fprintf(fid,'fix 2 0 1 1;\n');

%Define ELEMENTS
%Material Parameters

fprintf(fid,'set matTagB 1;\n');


fprintf(fid,['set K0 ' num2str(P(1)) ';\n']);
fprintf(fid,['set as_Plus ' num2str(P(2)) ';\n']);
fprintf(fid,['set My_Plus ' num2str(P(3)) ';\n']);
fprintf(fid,['set Lamda_S ' num2str(P(4)) ';\n']);
fprintf(fid,['set Lamda_C ' num2str(P(5)) ';\n']);
fprintf(fid,['set Lamda_A ' num2str(P(6)) ';\n']);
fprintf(fid,['set Lamda_K ' num2str(P(7)) ';\n']);
fprintf(fid,['set c_S ' num2str(P(8)) ';\n']);
fprintf(fid,['set c_C ' num2str(P(9)) ';\n']);
fprintf(fid,['set c_A ' num2str(P(10)) ';\n']);
fprintf(fid,['set c_K ' num2str(P(11)) ';\n']);
fprintf(fid,['set theta_p_Plus ' num2str(P(12)) ';\n']);
fprintf(fid,['set theta_pc_Plus ' num2str(P(13)) ';\n']);
fprintf(fid,['set Res_Pos ' num2str(P(14)) ';\n']);
fprintf(fid,['set theta_u_Plus ' num2str(P(15)) ';\n']);
fprintf(fid,['set D_Plus ' num2str(P(16)) ';\n']);

fprintf(fid,'uniaxialMaterial Bilin $matTagB $K0 $as_Plus $as_Plus
$My_Plus [expr -$My_Plus] $Lamda_S $Lamda_C $Lamda_A $Lamda_K $c_S $c_C
$c_A $c_K $theta_p_Plus $theta_p_Plus $theta_pc_Plus $theta_pc_Plus
$Res_Pos $Res_Pos $theta_u_Plus $theta_u_Plus $D_Plus $D_Plus;\n');

%Rotational hinge:
fprintf(fid,'element zeroLength 1 1 2 -mat $matTagB -dir 1;\n');

%Create recorder
fprintf(fid,'recorder Node -file Displacement.out -time -node 2 -dof 1
disp;\n');
fprintf(fid,'recorder Element -file Reaction.out -time -ele 1 force;\n');

%Apply CYCLIC load

fprintf(fid,'source analizar.tcl;\n');

% Function to minimize is err1

fclose(fid)

! Opensees bilin.tcl

[tdMod dMod] = textread('Displacement.out', '%f%f' );
[td FMod z2 z4 z5 z6 M] = textread('Reaction.out', '%f%f%f%f%f%f%f' );
```

```
[dExp FExp] = textread('experiment.out', '%f%f' );
dif=FExp-(-FMod);
err=dif.*dif;
err1=sum(err);
err1

plot(dMod,-FMod)
```

11 Matlab code for the GA objective function after having studied the different parameters.

```
function err1=bilfun2(P)


fid=fopen('bilin.tcl','w');

% Write the code for Opensees in TCL language in the text file.


fprintf(fid,'wipe;\n');
fprintf(fid,'model BasicBuilder -ndm 2 -ndf 3;\n');

% nodal coordinates

fprintf(fid,'node 1 0 0;\n');
fprintf(fid,'node 2 0 0;\n');

%single point constraints - Boundary Conditions

fprintf(fid,'fix 1 1 1 1;\n');
fprintf(fid,'fix 2 0 1 1;\n');

%Define ELEMENTS
%Material Parameters

fprintf(fid,'set matTagB 1;\n');


fprintf(fid,['set K0 86;\n']);
fprintf(fid,['set as_Plus ' num2str(P(1)) ';\n']);
fprintf(fid,['set My_Plus 130;\n']);
fprintf(fid,['set Lamda_S ' num2str(P(2)) ';\n']);
fprintf(fid,['set Lamda_C ' num2str(P(3)) ';\n']);
fprintf(fid,['set Lamda_A 1000;\n']);
fprintf(fid,['set Lamda_K ' num2str(P(4)) ';\n']);
fprintf(fid,['set c_S ' num2str(P(5)) ';\n']);
fprintf(fid,['set c_C 1;\n']);
fprintf(fid,['set c_A 1;\n']);
fprintf(fid,['set c_K ' num2str(P(6)) ';\n']);
fprintf(fid,['set theta_p_Plus ' num2str(P(7)) ';\n']);
fprintf(fid,['set theta_pc_Plus ' num2str(P(8)) ';\n']);
```

```
fprintf(fid,['set Res_Pos 0.00001;\n']);
fprintf(fid,['set theta_u_Plus 8.;\n']);
fprintf(fid,['set D_Plus ' num2str(P(9)) ';\n']);

fprintf(fid,'uniaxialMaterial Bilin $matTagB $K0 $as_Plus $as_Plus
$My_Plus [expr -$My_Plus] $Lamda_S $Lamda_C $Lamda_A $Lamda_K $c_S $c_C
$c_A $c_K $theta_p_Plus $theta_p_Plus $theta_pc_Plus $theta_pc_Plus
$Res_Pos $Res_Pos $theta_u_Plus $theta_u_Plus $D_Plus $D_Plus;\n');

%Rotational hinge:
fprintf(fid,'element zeroLength 1 1 2 -mat $matTagB -dir 1;\n');

%Create recorder
fprintf(fid,'recorder Node -file Displacement.out -time -node 2 -dof 1
disp;\n');
fprintf(fid,'recorder Element -file Reaction.out -time -ele 1 force;\n');

%Apply CYCLIC load

fprintf(fid,'source analizar.tcl;\n');

% Function to minimize is err1

fclose(fid)

! Opensees bilin.tcl

[tdMod dMod] = textread('Displacement.out', '%f%f' );
[td FMod z2 z4 z5 z6 M] = textread('Reaction.out', '%f%f%f%f%f%f%f' );
[dExp FExp] = textread('experiment.out', '%f%f' );
dif=FExp-(-FMod);
err=dif.*dif;
err1=sum(err);
err1

plot(dMod,-FMod)
hold on
plot(dExp,FExp-70,'g')
hold off

end
```