2014

# Four-bar Three-position Mechanism Synthesis Using the Principles of Fuzzy Logic Mathematics

Dana Reuther
*Lehigh University*

# THREE-POSITION FOUR-BAR LINKAGE MECHANISM SYNTHESIS USING THE PRINCIPLES OF FUZZY LOGIC MATHEMATICS

By
Dana Reuther

A Thesis
Presented to the Graduate and Research Committee
Of Lehigh University
In Candidacy for the Degree of
Master of Science
In
Mechanical Engineering and Mechanics

Lehigh University
Fall 2014 Semester
January 2015

## Certification of Approval

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

Date

Thesis Advisor, Dr. Meng-Sang Chew

Chairperson of Department, Dr. Gary Harlow

## ACKNOWLEDGEMENTS

There are many people who have helped me accomplish my goal of achieving my Master's Degree in Mechanical Engineering, to whom I will forever be indebted. First, I would like to thank my advisor, Professor Meng-Sang Chew, who has shown me that failure is never an option, for there is always another solution. I would also like to acknowledge Dr. Vallorie Peridier, who knew I would succeed at Lehigh University and encouraged me throughout my years at Temple University and then onto Lehigh University. Another person, who has attempted to teach me advanced MATLAB coding and stuck by my side throughout my entire time at Lehigh, Brian Wisniewski, will always know how much he has taught me in many walks of life. Lastly, I would like to recognize the dedication to my education that my parents, Mary and Tom Reuther, have always shown. They have supported me in my dreams every step of the way. Without all of those people mentioned above, I would not have been as successful throughout my Master's program as I am now.

## TABLE OF CONTENTS

## List of Tables

## LIST OF FIGURES

## ABSTRACT

The human mind is very imprecise and uncertain most of the time. We are allowed to think as we wish, as things really are. This, however, is not how computers are designed to allow us to solve realistic problems. Now the question arises: how do we represent uncertainty and impreciseness using very accurate devices? In order to add flexibility into mechanical design, fuzzy logic can be used to represent the imprecise data, such as how humans think. Fuzzy logic allows the use of representing more than one value for a specific variable; it also allows the user to specify the shape the data takes (the membership function) which could be a function of the problem at hand. While incorporating fuzzy logic into design, the design becomes more imprecise in the sense that its variables represent more than one value, yet the design becomes more accurate when considering how the human brain interprets these variables.

# Chapter 1: Introduction

Within the field of kinematics and dynamics, mechanism synthesis is used to determine the motion of a linkage system to aid in machine design. Machine design consists of mechanism synthesis, which is the process of defining specific values of lengths and angles of a four-bar linkage, in this case. Sometimes, the values of the used lengths are either imprecise or assumed, which must be represented within the problem analysis. Antonsson and Otto stated in their published ASME article that these types of uncertainties or impreciseness must be represented throughout the problem in order to replicate real world engineering. They also suggest the use of several methods which accommodate these uncertainties, such as the utility theory, fuzzy sets, optimization, matrix methods, probability methods, necessity methods, and fuzzy design methods. The desired method taken in this specific approach is the use of fuzzy logic in conjunction with standard mechanism design.

The use of fuzzy logic mimics the human mind in the sense that it can process data that is not precise and then make sense of that data. Incorporating fuzzy logic into an algebraic procedure and evaluate areas instead of points, which is the goal of this project entailed below. Integrating fuzzy logic into four-bar linkage synthesis will allow the definition of an area, rather than a fixed point. Instead of allowing a point of interest to be fixed at one specific location, the point will now be able to be defined as an area, which more adequately represents the possible region in which the point may be located.

## 1.1 Description of Project

Three-position four-bar linkage synthesis has conventionally been thought of as a method to design a mechanism with specific locations for the mechanism to pass through. If the mechanism is to pass through a specific region, as to not interfere with obstructions, this region would conventionally be represented as a specific point within the region; however, representing a region as a specific point limits the number of possible solutions, potentially depriving the user of the optimal configuration of their mechanism. This project is meant to address this situation.

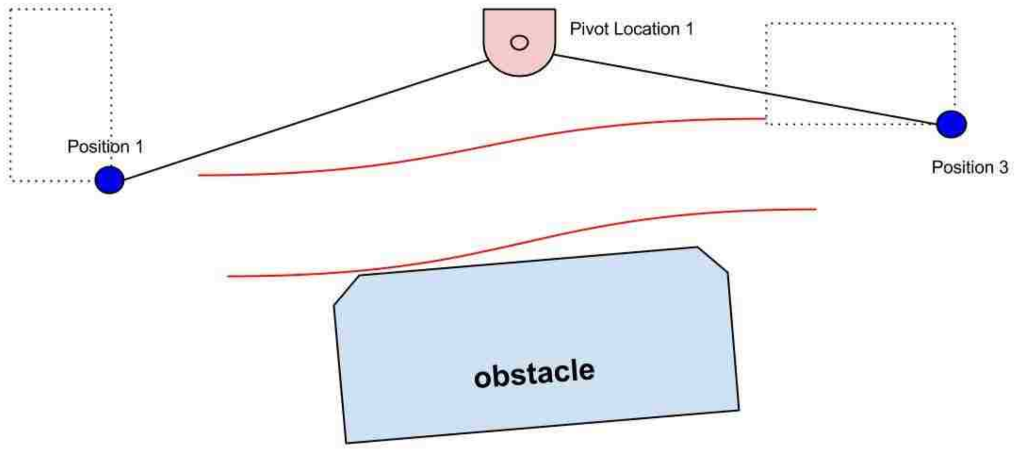

*Figure 1-1. Mechanism Synthesis*

To represent the area the mechanism must pass through as a region instead of a specific point, the principles of fuzzy logic must be employed to determine the appropriate position locations. In Figure 1.1, shown above, the above scenario is depicted. It is important to note that in Figure 1.1 and Figure 1.2, discussed later, only part of the four-bar linkage is shown for simplicity. In Figure 1.1, there is an obstacle inhibiting some movements of the mechanism from Point 1 to Point 3. Before now, there would be another point, Point 2, located within the optimal region of movement, outlined by the red curves, to ensure the movement of the mechanism would not interact with the obstacle. Figure 1.2 displays the same scenario, but the second position is now depicted as a trapezoidal area within the optimal region of movement.

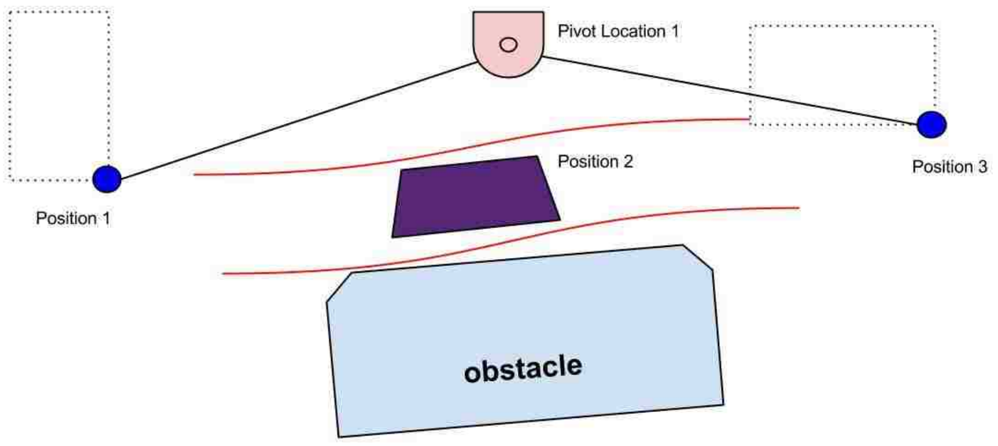

*Figure 1-2. Fuzzy Position specified in purple.*

The process of representing the second position as an area creates more possibilities for the end result of the mechanism synthesis, fulfilling the user's needs.

# Chapter 2: Relative Background Information

## 2.1 Mathematical Logic

Logic is known as the study of reasoning. There are two main types of reasoning: classical logic and multi-valued logic. Classical logic deals with certainty, while multi-valued logic, namely fuzzy logic, deals with certainty versus uncertainty.

### 2.1.1 Classical Logic

Classical logic refers to a type of logic that consists of concrete facts that are either true or false. Elements in the universe of consideration are considered to be distinct, discrete, and countable. An element of consideration either is or is-not within the universe of consideration. This type of logic provides no room for uncertainty, ultimately reducing accuracy (Ross 26).

### 2.1.2 Fuzzy Logic

Unlike classical logic, fuzzy logic allows uncertainty and inexactness. Mostly, this uncertainty may arise from the complexity of a particular system or device. Zadeh first introduced fuzzy logic in 1973 to express vagueness throughout the world ("Multi-Valued Logic"). Fuzzy logic occurs in daily life, when speaking to others and within most applications, where there is not a true or false response (Ross, 2010, p. 13). "Our understanding of most physical processes is largely based on imprecise human reasoning" (Ross, 2010, pg. 2). The ability to process this imprecise reasoning is known as fuzzy logic.

An example of a situation that utilizes fuzzy logic is as follows: A doctor asks how much an ailment hurts. The patient responds with, 'a lot'. The term 'a lot' here is deterministic of the patient, and it is relative to pain they have experienced previously. The question by the doctor here allowed for the patient to answer the question in a “fuzzy” way. If the doctor had asked, “are you in pain?”, the patient could have responded with, ‘yes’. This here would be the crisp answer to the same question, where ‘yes’ is a definite answer.

Fuzzy logic is a type of multi-valued logic, which indicates that there are more than two “truth” values (“Multi-Valued Logic”). Fuzzy logic is a theory that is related to the idea that sets of data do not have sharp boundaries, rather they are determined by a membership value within the set (Mathworks, 2014, pp. 1-3). Fuzzy logic is used for this application because of the ability it has to allow for imprecision.

## 2.2 Fuzzification and Defuzzification of Fuzzy Logic

When using fuzzy logic, it is imperative to properly fuzzify and defuzzify the data in question in order to achieve accurate results. To fuzzify a crisp number to transform it into a fuzzy number, the uncertainty of a variable must be known. Once this is known, the membership function can be used to represent the variable(s) (Ross, 2010, p. 115). Defuzzification can be thought of as reversing fuzzification; it is the process of approximating one value to represent an entire fuzzy set. Both fuzzification and defuzzification are unique properties of fuzzy sets that can greater approximate and understand uncertainty.

### 2.2.1 Fuzzification

There are six methods of assigning a membership function to a set of variables: intuition, inference, rank ordering, neural networks, genetic algorithms, and inductive reasoning. Intuition requires the observer/user to assign membership values to the variables in the function based on the context of the problem along with the amount of uncertainty in the values belonging to the membership function (Ross, 2010, p. 175). The interference method to fuzzify variables is based on deductive reasoning, where the user predicts the outcome based on facts and previous knowledge (p. 176). Rank ordering is the process of comparing each variable in the set to each other and assigning membership values to each based on that of the other variables (p. 178). The most common method to fuzzify a variable that has been presented in literature, which utilizes the basic principles of fuzzy logic, is the neural networks. Neural networks are considered an intelligent system, where the program learns the behavior of the model, simulating the functionality of the human brain (pg. 179). Like neural networks, genetic algorithms are programs that utilize the information from the system to create the best model. To fuzzify a system using genetic algorithms, the algorithm will process through a vast amount of potential solutions, comparing each solution to the next to find the best answer, mimicking evolution (pg. 189). Each fuzzification method has specific benefits; however, these benefits may not prove useful for every system.

The type of fuzzification used within the scope of this project is intuition. Here within, the user must define the membership function based on the specifics of each presented problem in order to obtain the best solution for the intended purposes. This system, although it may seem lacking some of the features of other fuzzification methods, is problem specific and user specific; therefore, it will be the best method because of the flexibility.

### 2.2.2 Defuzzification

Like the fuzzification process, there are many different methods that may be used in order to obtain a crisp answer from a fuzzy result. The method chosen to defuzzify a fuzzy variable is situation-based, in which the user has a preference as to which method to use. There are four main methods that are used to defuzzify a variable: max-membership principle, centroid method, weighted average method, and mean-max membership method. Each of these four methods have their advantages and disadvantages.

The max-membership method is a simple method that takes the peak of the membership function, the number in the fuzzy set that has the highest associated grade. The formula used to represent this is the following:

$$\mu_{\tilde{C}}(z^*) \geq \mu_{\tilde{C}}(z),\ for\ all\ z \in Z,$$

where $z^*$ is the defuzzified value, and z is every value in the fuzzy set, $\tilde{C},$ in the universe of Z (Ross, 2010, pg. 99). From this expression, it is clear that the grade value, μ, of the defuzzified value is the greatest of the entire fuzzy set. The max-membership function can be seen in Figure 2-1 below.

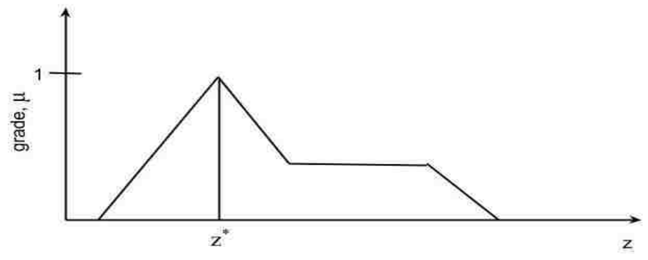

*Figure 2-1. Max-Membership Method*

The centroid method, which may also be referred to as the center of area method or the center of gravity method, defines the defuzzified value of a fuzzy set as:

$$z^* = \frac{\int \mu_{\tilde{C}}(z) \cdot z\, dz}{\int \mu_{\tilde{C}}(z)\, dz}$$

where, again, $z^*$ is the crisp value from the defuzzified fuzzy set, $\tilde{C},$ where $z \in Z$. The centroid method can be seen in Figure 2-2 below.

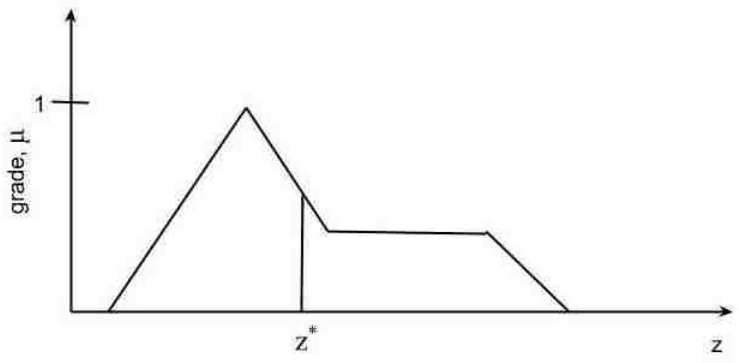

*Figure 2-2. Centroid Method Defuzzification*

Weighted average method is the third type of defuzzification method that is most often used when dealing with fuzzy applications because of its efficiency; however this method is extremely limited by the shape of the functions (Ross, 2010, pg. 99). To calculate the defuzzified value of the fuzzy set using this method:

$$z^* = \frac{\sum \mu_{\tilde{C}}(\bar{z}) \cdot \bar{z}}{\sum \mu_{\tilde{C}}(\bar{z})}$$

The figure below, Figure 2-3, displays this method.

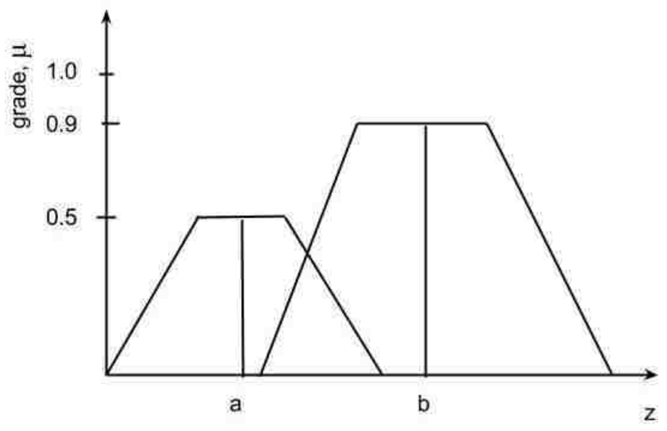

*Figure 2-3. Weighted Average Method*

As seen by the image, this method is useful when there are two distinct membership functions within the fuzzy set. The maximum of each subsection of this membership function will be weighted according to its respective grade in order to find the defuzzified value of the set (pg. 100).

The final method that will be discussed here is the mean max membership method, which may also be referred to as the middle-of-maxima method. This method is best used, when the maximum grade is associated with more than one value within the fuzzy set and the maxima is in the form of a plateau rather than a point. This defuzzification can be calculated by simply calculating the average of the plateau:

$$z^* = \frac{a+b}{2}$$

where a and be are the values represented by Figure 2-4 below.

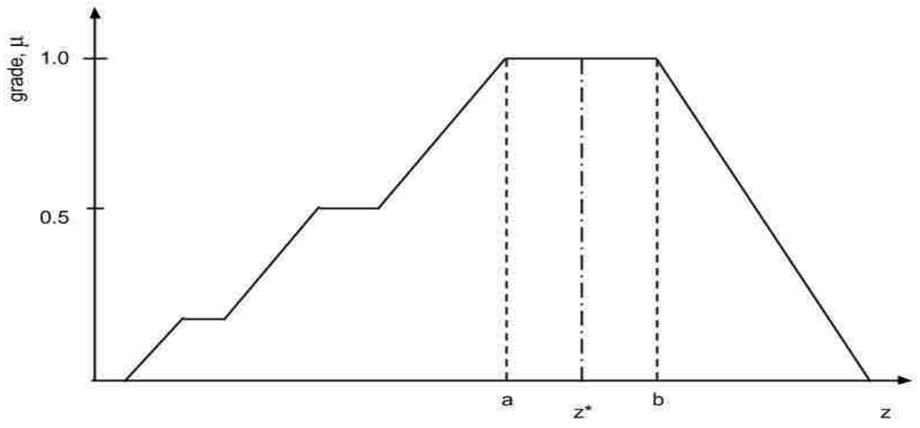

*Figure 2-4. Mean Max-Membership*

The method of defuzzification that is chosen is primary based on the shape of the fuzzy membership function as well as the usage of the fuzzy set.

## 2.3 Mechanism Synthesis

In his text, Norton expresses machine design as the following: "QUALITATIVE SYNTHESIS means the creation of potential solutions in the absence of a well-defined algorithm which configures or predicts the solution" (Norton, 1999, p. 76). There are several types of mechanism synthesis, such as analytical linkage synthesis, graphical linkage synthesis, and position analysis.

### 2.3.1 Analytical Linkage Synthesis

Analytical synthesis is an algebraic method, making it the simplest method to use with the aid of technology. This method of mechanism synthesis originated with Sandor, and has continued to be perfected throughout the years by a few of his students (Norton, 1999, p. 188).

Within analytical linkage synthesis, there are three different types of synthesis, namely, function, path, and motion generations. These various types of synthesis produce results that are differentiated by the initial request of the user. Function generation creates an output motion for a given input motion, such as a crank-rocker, where there is a rotation input that produces a rotation output. The second type of synthesis, path generation, allows for a specific point of the mechanism to move along a desired path. Path generation is usually used with a four-bar crank-rocker (Norton, 1999, p. 188). The final type of analytical synthesis, motion generation, assumes a set of positions for a given curve inside of a plane.

Here within, motion generation analytical synthesis will be used to determine the needed values of the problem from the given or assumed values of the problem. This type of synthesis allows for the four-bar linkage to move in a way such that a desired point will pass through two or three precise positions. This type of synthesis is the best way to approximate the given positions of a specific point of a four-bar linkage.

### 2.3.2 Position Analysis

The goal of position analysis is to determine the forces that the four-bar linkage can withstand to ensure the mechanism will not fail. This is done by using position analysis to calculate the acceleration of the four-bar linkage as it is moving from one position to the next under the given lengths and angles. The acceleration comes from the twice-differentiated position equations. Because this method does not determine the lengths and angles within the four-bar linkage that will result in the linkage moving from one desired point to the next, this is not the optimal analysis type that will be used. Function, path, and motion generations are used to complete a given task.

### 2.3.3 Graphical Linkage Synthesis

Like analytical synthesis, there are three basic types of synthesis, which accomplish different goals set forth by the problem requirements. The main difference between graphical synthesis and analytical synthesis is that graphical synthesis uses non-algebraic techniques to solve for a solution (Norton, 1999, pg. 78). The main type of graphical synthesis is dimensional synthesis.

Dimensional synthesis is “the determination of the proportions (lengths) of the links necessary to accomplish the desired motions” (Norton, 1999, pg. 78). The assumption made with dimensional analysis is that the determination between cam and linkage for the problem is set. Dimensional analysis can be either quantitative or qualitative, where quantitative analysis is mostly used for cams, while qualitative analysis is used for linkages (p. 78). Graphical analysis is much less complicated than analytical analysis; however, it can produce inaccurate results.

## 2.4 Importance of Mechanism Synthesis using the Principles of Fuzzy Logic

Mechanism synthesis, such as analytical synthesis of a four-bar three-position linkage, is not guaranteed to work for a specific set of given and known values of each variable. With this in mind, it is noticeable that a range of values around a specific number for a given variable, may suit our purposes more accurately.  Representing a variable by a set of numbers, rather than a specific number, increases the chances of the synthesis producing desired results.  One of the best ways to represent a set of numbers around a specific location is to use fuzzy logic mathematics.  With fuzzy logic, specific variables, such as those that are assumed or non-exact, can be represented as a fuzzy variable, meaning a variable that represents a universe of discourse.  This idea would allow the user to be provided with a variety of results along with corresponding grades; therefore, the user would be able to choose the result which works best for their purpose.

# Chapter 3: Prior Works

Fuzzy logic is a relatively new area of study, only being introduced in the 1960s. There is, however, many different fields of study in which fuzzy logic has been used. Two of the main uses of fuzzy logic have been in terms of mathematics and control theory. The following sections outline previous research that has been conducted which are similar to, yet not the same as, the procedures outlined here within this report.

## 3.1 Kumar and Schuhmacher, 2005

Vikas Kumar and Marta Schuhmacher discuss the use of fuzzy logic in system modeling in their article, “Fuzzy uncertainty analysis in system modelling”, written in 2005. This article focuses on the fuzzy analysis of ground water pollutant transport. Clearly, this is not similar to the situation this paper has addressed. This article does, however, address uncertainty and within mechanical design, which is the basis of most uses of fuzzy logic, as it is within this project.

## 3.2 Dhingra and Rao, 1991

Dhingra and Rao have also conducted work that is similar to that mentioned above. The premise of this work was to optimize high speed mechanisms whose parameters are for a planar four bar linkage. This work is similar to but not the same as the work presented here. The authors, Dhingra and Rao, have brought up significant ideas that can be considered for future progress of this project, such as

> (a) multiple objectives in the nonlinear programming formulation, (b) ensuring that the resulting mechanism is free of branch, order, and Grashof defects, (c) addition of counterweights to moving links to improve dynamics performance measures of

the resulting mechanism, and (d) incorporating techniques to model vague and imprecise statements in mechanism design problems (Dhingra and Rao, 1991).

The last suggestions, (d), can be taken into consideration for this project, since the use of fuzzy logic is considered a technique that models impreciseness. However, Dhingra and Rao's general ideas do not correlate completely with the topic of this project.

## 3.3 Laribi, Mlika, Romdhane, and Zeghloul, 2004

Another similar process that uses algorithms in combination with mechanism synthesis is that which is introduced by Laribi, Mlika, Romdhane, and Zeghloul in their journal article, "A combined genetic algorithm – fuzzy logic method (GA-FL) in mechanisms synthesis", 2004. This process uses fuzzy logic in conjunction with mechanism synthesis, similar to the project mentioned here within, but fuzzy logic is not solely used; genetic algorithms are also used to accomplish the optimized mechanism synthesis.

## 3.4 Chen, Tzeng, Hsu, and Chen, 2010

Chen, Tzeng, Hsu, and Chen have also done work using both fuzzy logic and linkage synthesis simultaneously; however, their work also incorporated the theories of the Taguchi Method and Principle Component Analysis. Their work also differs from the work stated here due to the fact that they used a six-bar linkage, whereas a four-bar linkage was used here.

## 3.5 Previous Works Conclusion

There have been many reports of researchers combining the principles of fuzzy logic with engineering design; however, there are significant differences with the research they conducted and the work specified in this project.

# Chapter 4: Application

The goal of this project was to complete a mechanism synthesis, where uncertainty in some of the input values may be present. In this particular case, the uncertainty is placed in the location of the second position as well as the pivot location. To accomplish this task, the use of fuzzy logic was utilized throughout the mechanism synthesis to represent the uncertainty in the necessary variables. To simplify the amount of work, only the left hand side of the linkage was computed, dyad WZ, for most of the examples; however, the same process would be repeated for dyad US to complete the full synthesis.

## 4.1 Fuzzification Process

The equation used to fuzzify values is:

$$trapezoid(x; a, b, c, d) = \begin{cases} \dfrac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \dfrac{d-x}{d-c}, & c \leq x \leq d \end{cases}$$

and all other values within the universe of discourse are zero.

To fuzzify these variables, a built-in MATLAB function, trapmf([a b c d]), was used to create a trapezoidal membership function. The values of a, b, c, and d can be found in Figure 4-1, and the user specifies the desired values, where the mid-way between b and c would be the specific point location of $P_{21}$, if $P_{21}$ were specified as a point.

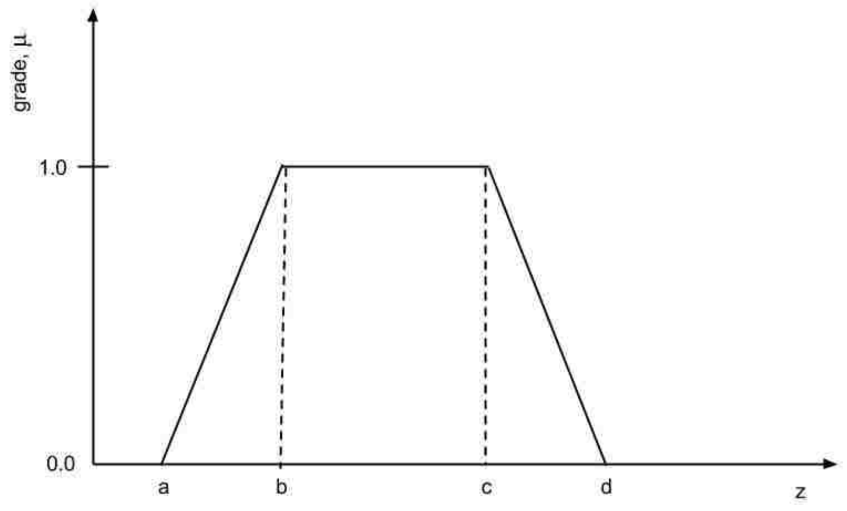

*Figure 4-1. Trapezoidal Membership Function*

This fuzzification method is used for all of the fuzzifications in this work, regardless of the desired variable or result. If the input fuzzy number were to be fuzzified as a triangular membership function, the values of b and c would be equal.

## 4.2 Defuzzification Process

Of the defuzzification processes that were detailed in Section 2.2.2, the method of fuzzification used to defuzzify this work was the centroid method. This method was chosen because it represents the mean of all the numbers within the fuzzy result. Another method that could have been chosen to defuzzify these values could have been the mean-maxima method, where the middle of the plateau of the fuzzy result would have been the crisp, defuzzified value of the answer. The defuzzified values are shown in the results tables of each fuzzy synthesis as “actual answer” and the value that this defuzzified answer is being compared to is the result of the crisp synthesis.

## 4.3 Mechanism Synthesis with a fuzzy $P_2$

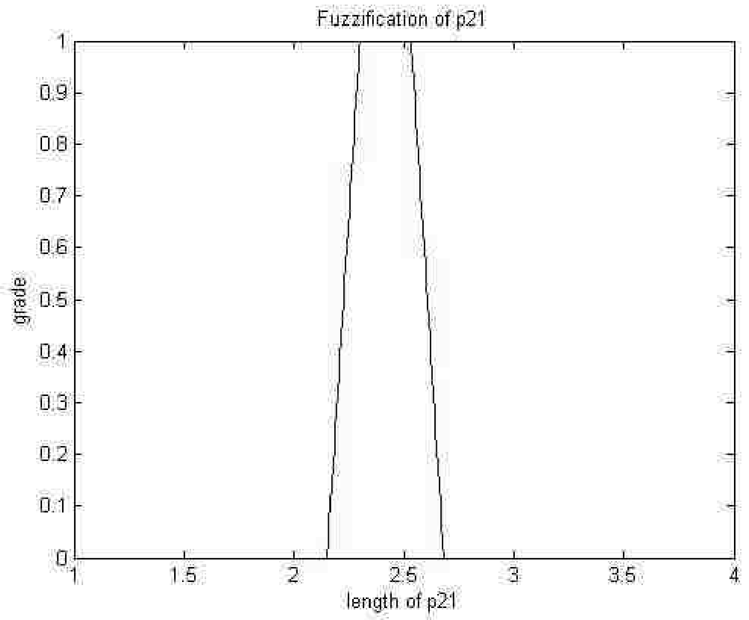

*Figure 4-2. Fuzzification of P21*

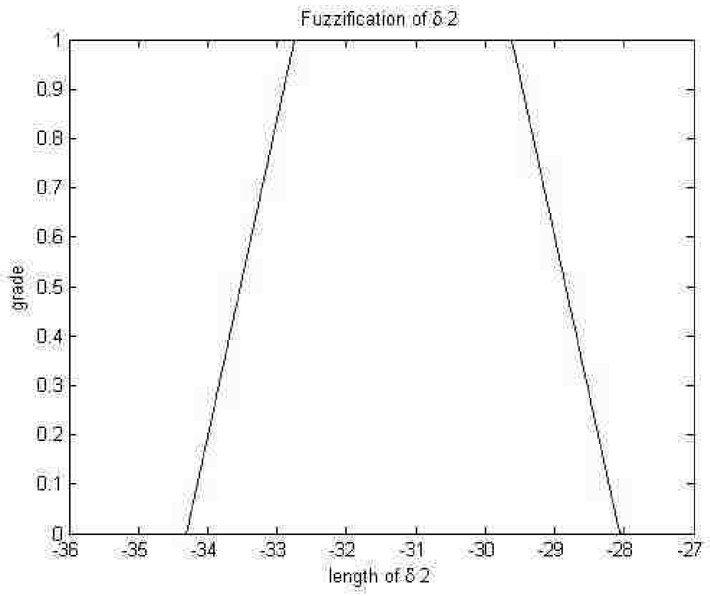

*Figure 4-3. Fuzzification of δ2*

The fuzzification of the second position, $P_2$, consists of fuzzifying the length of $P_{21}$ and $\delta_2$. The fuzzified values of $P_{21}$ and $\delta_2$ can be found in Figures 4-2 and 4-3, respectively.

The shaded region in the figure below, Figure 4-4, displays the area in which the length of $P_{21}$ can expand. This area was found using the fuzziness of the length of $P_{21}$ and the fuzziness of $\delta_2$. The area represented is shaded such that, the darker the color, the closer the fuzziness of $P_{21}$ and $\delta_2$ are to 1. From this figure, it is easily seen that there are many combination of lengths and angles that form the location of $P_2$.

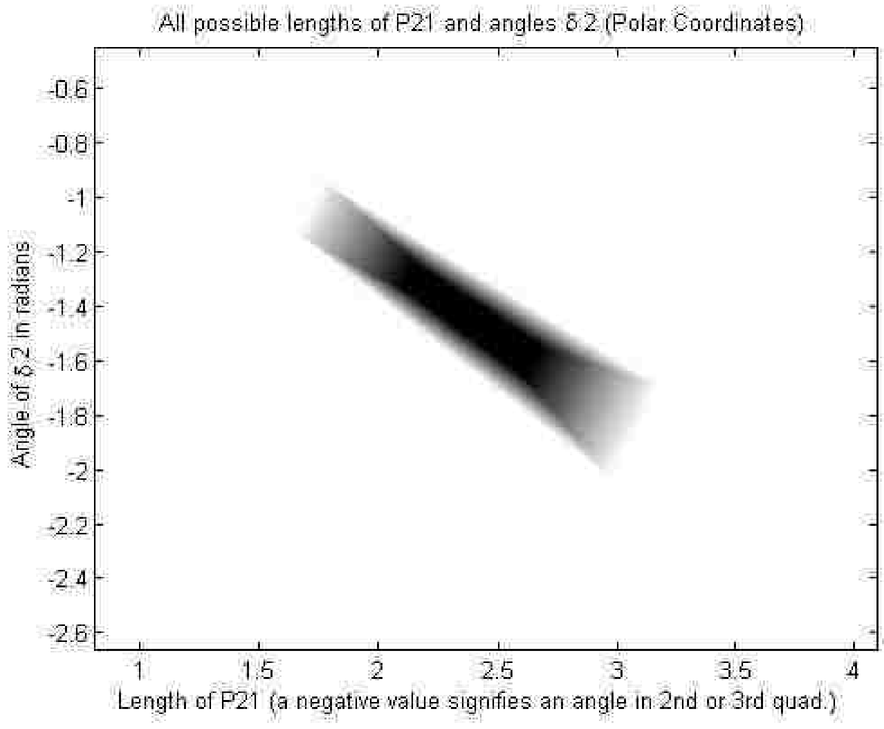

*Figure 4-4. Locations of fuzzy P2*

### 4.3.1 Two-position synthesis, fuzzification of $P_2$

The first mechanism synthesis that was conducted incorporating the use of fuzzy logic was a two-position four-bar linkage motion generation linkage synthesis. Two position synthesis was completed to confirm the feasibility of allowing the use of fuzzy logic simultaneously with linkage synthesis. Figure 4-5 displays a two-position four-bar linkage, where $P_1$ denotes the first position of the specified point, $\delta_2$ denotes the angle the point moves from the horizontal to arrive at the second position, $P_2$. The second position of the entire linkage is shown in the figure with a subscript 2.

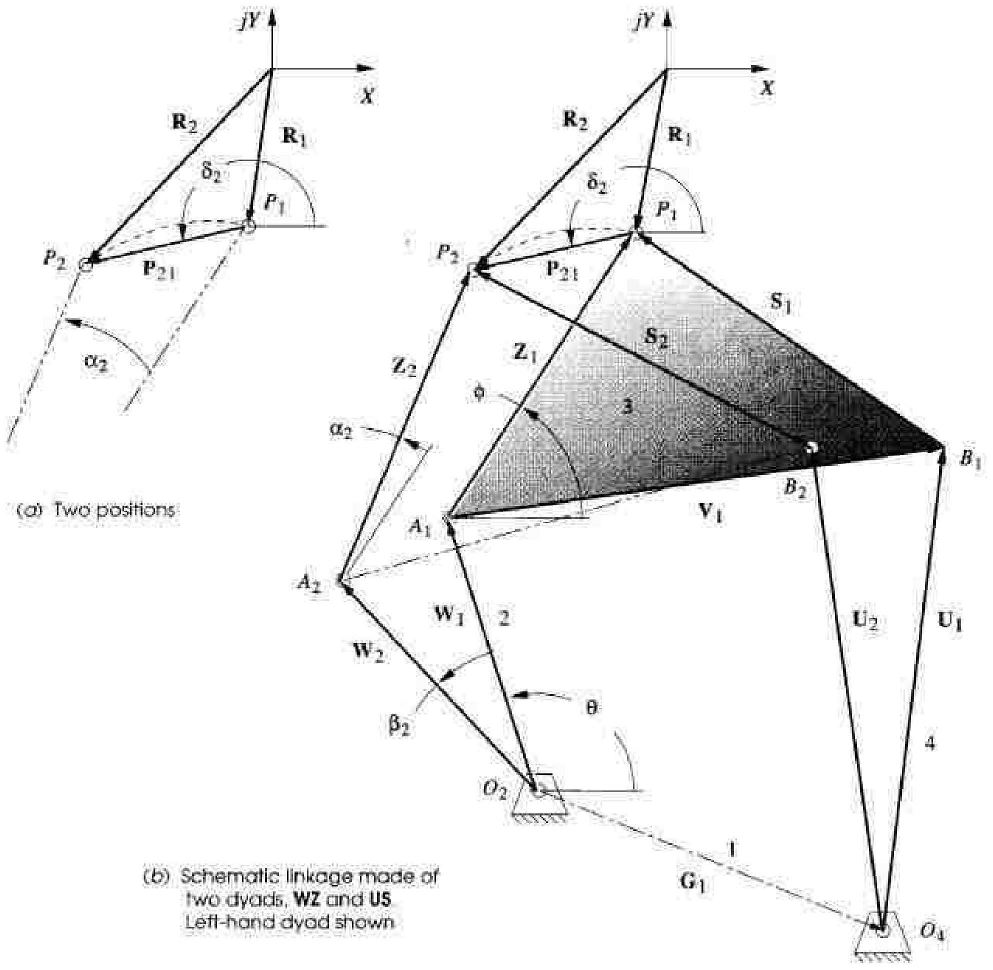

*Figure 4-5. Two-position Four-bar Linkage (Norton, Fig 2-1)*

Figure 4-6, below, shows the two-position, four-bar linkage from Figure 4-5, except that $P_2$ is denoted with a red trapezoid. The red outlined trapezoid represents the potential locations of $P_2$. Fuzzy logic was used to represent the potential positions of $P_2$ mathematically.

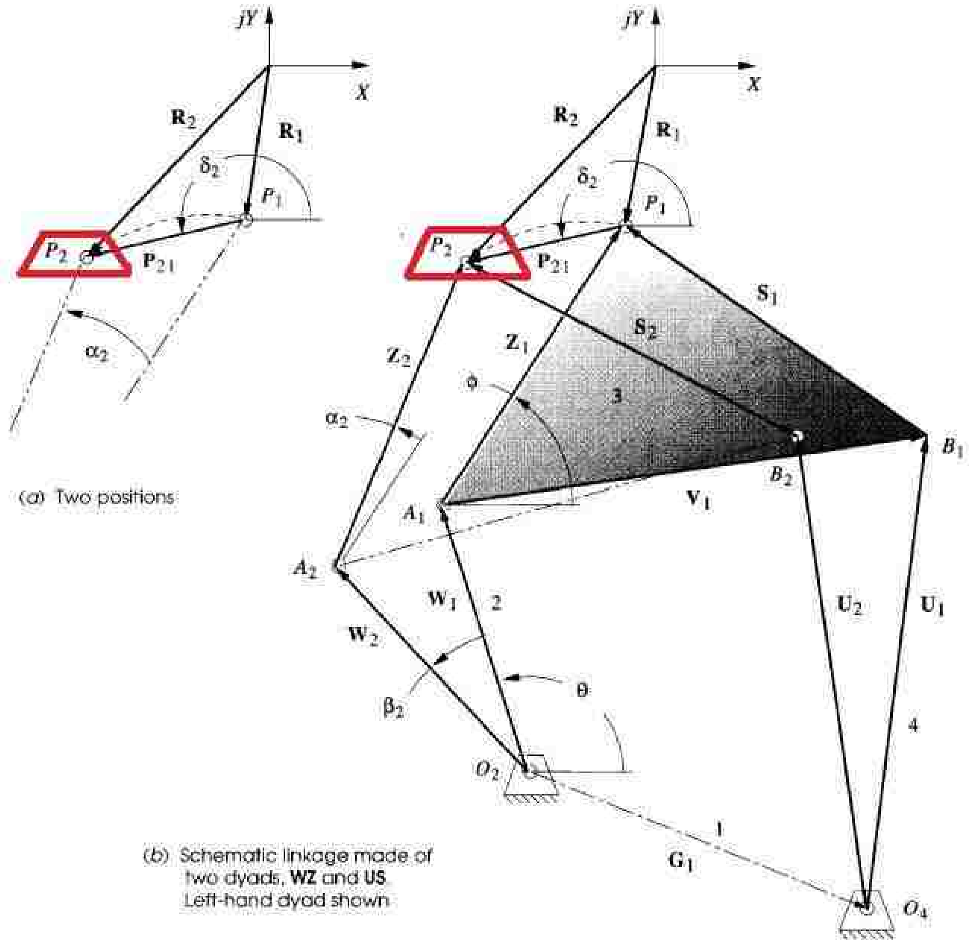

*Figure 4-6. Two-position Four-bar Linkage, Modified (Norton, Fig 5-1, modified)*

The vector loop equation for the two-position four-bar linkage synthesis is (Norton, 1999, p. 191):

$$W_2 + Z_2 - P_{21} - W_1 - W_2 = 0$$

Which can be manipulated to produce the resulting equations (p. 193):

$$A = \cos(\beta_2) - 1; \qquad B = \sin(\beta_2)\,; \qquad C = \cos(\alpha_2) - 1;$$

$$D = \sin(\alpha_2)\,; \qquad E = p_{21}\cos(\delta_2)\,; \qquad F = p_{21}\sin(\delta_2)$$

Substituting,

$$AW_x - BW_y + CZ_x - DZ_y = E$$

$$AW_y + BW_x + CZ_y + DZ_x = F$$

It can be seen from these equations that only E and F are considered to be fuzzy variables (recall only $P_{21}$ and $\delta_2$ are fuzzy). These above two equations were solved using matrix multiplication using custom MATLAB functions, shown in the appendix. The matrix multiplication was done explicitly, so the fuzzy arithmetic could be manipulated properly. The following figures are the results of the fuzzy mathematics and the solution to the two-position four-bar linkage synthesis with a fuzzy $P_{21}$ position, using values found in Norton's textbook. These values, extracted from Example 5-1 in Norton's text can be found in the Table 4-1.

*Table 4-1. Values extracted from Norton Ex. 5-1*

| Variable | Magnitude of Value |
|---|---|
| $P_{21}$ | 2.416 |
| $\delta_2$ | 165.2 |
| $\alpha_2$ | 43.3 |
| $\beta_2$ | 38.4 |
| $\varphi$ | 26.4 |
| $\theta$ | 71.6 |

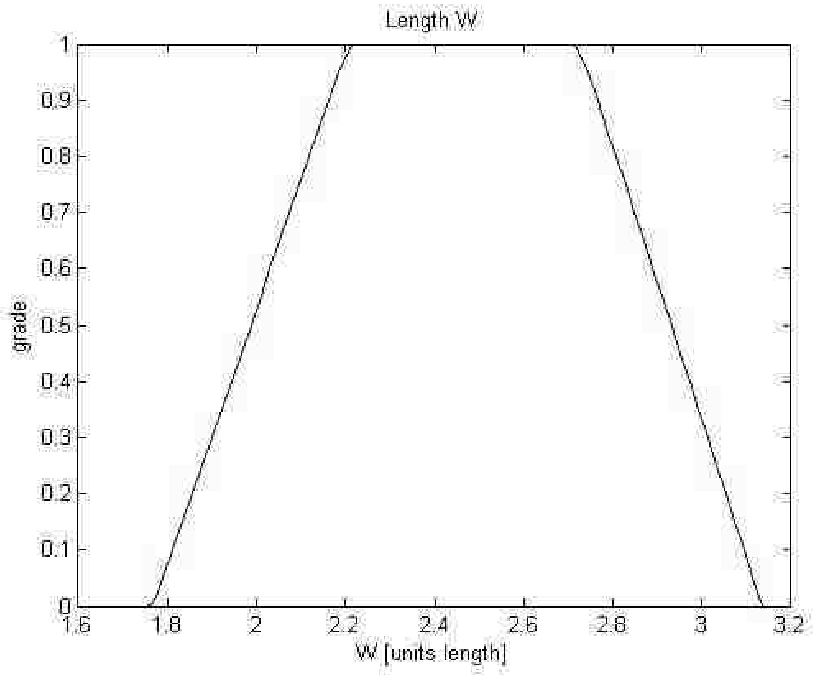

*Figure 4-7. Fuzzy Length of Linkage W*

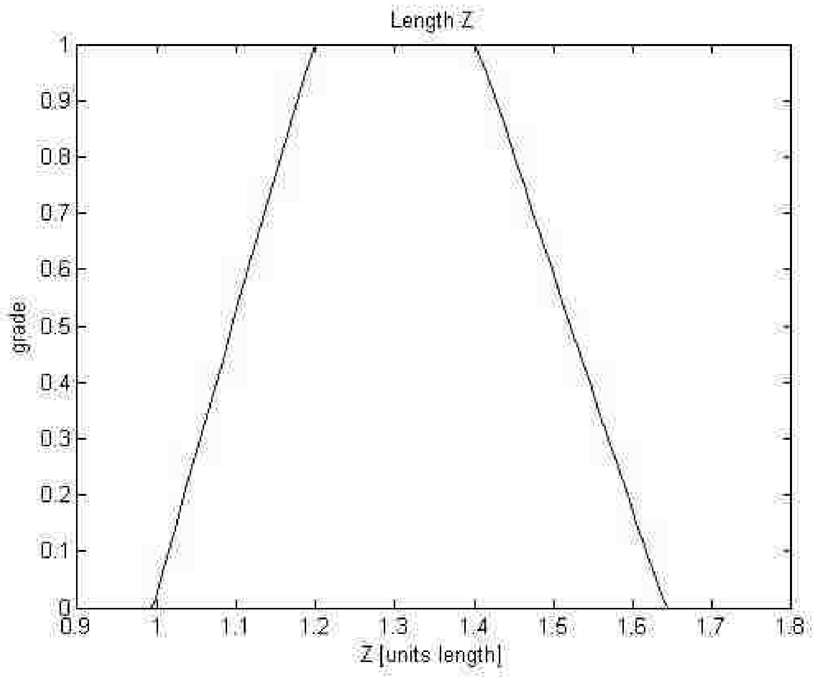

*Figure 4-8. Fuzzy Length of Linkage Z*

It can be seen from Figure 4-7 and 4-8 that the solutions to the linkage synthesis have a general trapezoidal fuzzy shape, but there are a few areas that are not perfectly straight. This is due to the manipulation of the data throughout the fuzzy arithmetic. Also, the data was manipulated using the vertex method, with the goal of streamlining the results.

To calculate the optimum value of the lengths of W and Z, the data was defuzzified, using the centroid “defuzz” command in MATLAB. These defuzzified values were calculated to be 2.4612 and 1.3056 units for W and Z, respectively. The error percentages from the exact values listed in Norton’s text are 1.871% and 0.586%, respectively, which can be attributed to fuzzifying the values of $P_{21}$ and $\delta_2$.

The uniqueness of the presented results is that the lengths of W and Z can be a multitude of values that will create the desired linkage at the desired locations. The result isn’t limiting, like it would be if the location of $P_{21}$ was a specific point.

## 4.3.2 Three-position fuzzification of P$_2$

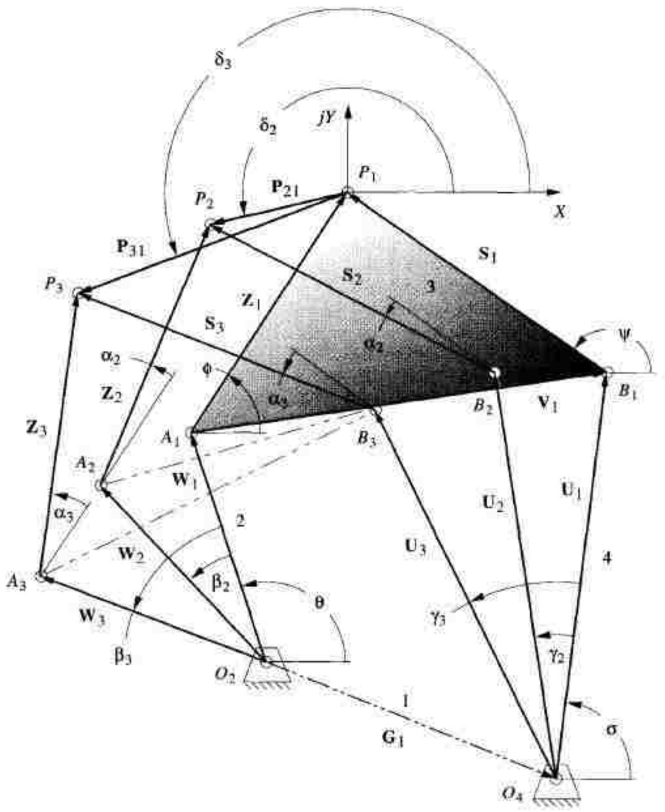

*Figure 4-9. Three-position Four-bar Linkage (Norton, Fig. 5-4)*

As seen above in Figure 4-9, three-position four-bar linkage synthesis follows the same process as the two-position four-bar linkage synthesis, except the mathematical equations are different. The vector loop equation will incorporate the addition of P$_{31}$, which is the distance from P$_1$ to P$_3$, as seen in Figure 4-9. The vector loop equations now becomes,

$$W_2 + Z_2 - P_{21} - W_1 - Z_1 = 0$$

$$W_3 + Z_3 - P_{31} - W_1 - Z_1 = 0$$

Using, again, the values in Norton's textbook, seen below, to solve for a specific linkage, the following graphs were produced to show the fuzzification of the $P_{21}$ and $\delta_2$.

*Table 4-2. Values extracted from Norton Example 5-2*

| Variables | Magnitude of Value |
|---|---|
| $P_{21}$ | 2.798 |
| $\delta_2$ | -31.19 |
| $P_{31}$ | 3.919 |
| $\delta_3$ | -16.34 |
| $\alpha_2$ | -45.0 |
| $\alpha_3$ | 9.3 |
| $\beta_2$ | 342.3 |
| $\beta_3$ | 324.8 |

Again, $P_{21}$ and $\delta_2$ are fuzzy variables, and the "fuzzyfun" custom function fuzzifies these variables. Figures 4-10 and 4-11 show the fuzzification graphs of $P_{21}$ and $\delta_2$.

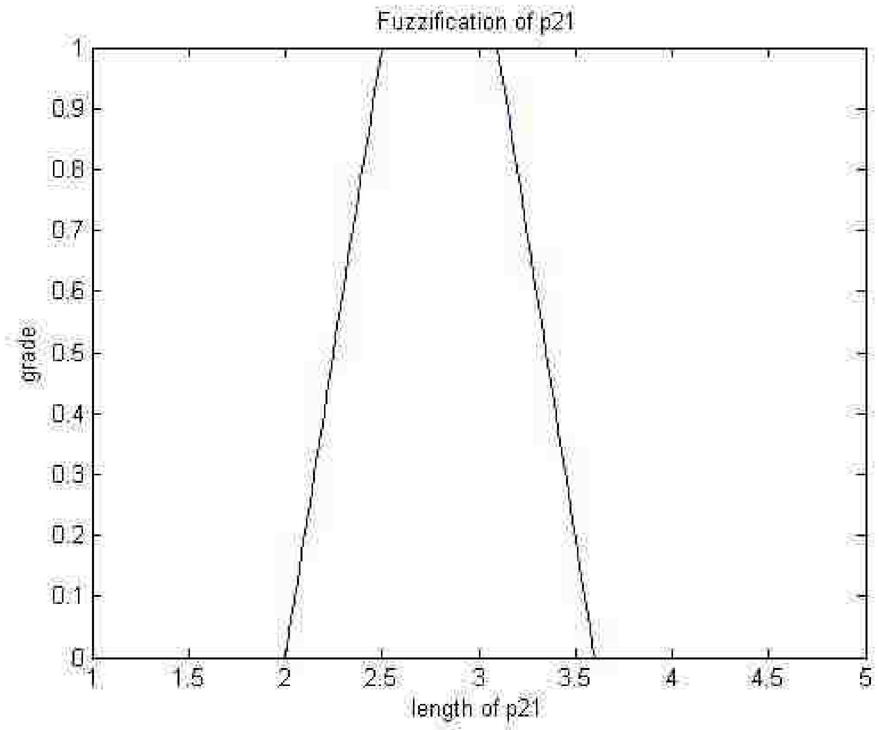

*Figure 4-10. Fuzzification of P21*

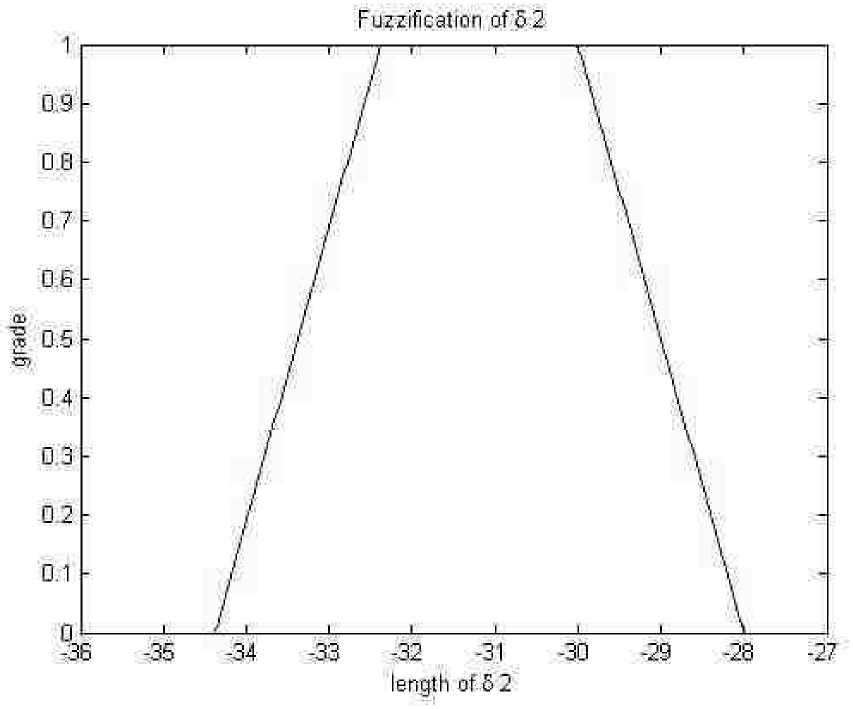

*Figure 4-11. Fuzzification of δ2*

The following is the equation and matrix forms of the simplified vector loop equation for the three-position four-bar linkage synthesis. Using the information in the equations below, it can be seen that, again, only E and M are fuzzy; therefore, the solution of this matrix multiplication was done by hand to account for the fuzzy variables. Using numbers provided in Norton's textbook (see above) alongside the custom MATLAB codes (provided in the appendix), the following output graphs were computed.

$$A = \cos(\beta_2) - 1; \qquad B = \sin(\beta_2); \qquad C = \cos(\alpha_2) - 1;$$

$$D = \sin(\alpha_2); \qquad E = p_{21}\cos(\delta_2); \qquad F = \cos(\beta_3) - 1;$$

$$G = \sin(\beta_3); \qquad H = \cos(\alpha_3) - 1; \qquad K = \sin(\alpha_3);$$

$$L = p_{31}\cos(\delta_3); \qquad M = p_{21}\sin(\delta_2); \qquad N = p_{31}\sin(\delta_3)$$

The system can then be put into the simplified system of equations as:

$$AW_x - BW_y + CZ_x - DZ_y = E$$
$$FW_x - GW_y + HZ_x - KZ_y = L$$
$$BW_x + AW_y + DZ_x + CZ_y = M$$
$$GW_x + FW_y + KZ_x + HZ_y = N$$

Putting this system into matrix form:

$$\begin{bmatrix} A & -B & C & -D \\ F & -G & H & -K \\ B & A & D & C \\ G & F & K & H \end{bmatrix} \cdot \begin{bmatrix} W_x \\ W_y \\ Z_x \\ Z_y \end{bmatrix} = \begin{bmatrix} E \\ L \\ M \\ N \end{bmatrix}$$

Figures 4-12 and 4-13 shows the fuzzy lengths of W and Z using the vertex method to allow for ease of use of the graphs. To achieve the optimal length of W and Z, the data for W and Z was defuzzified using MATLAB's centroid defuzz function. To compare the results with the text's results, the values that were defuzzified were the x and y components of both W and Z, noted as Wx, Wy, Zx, and Zy, respectively.

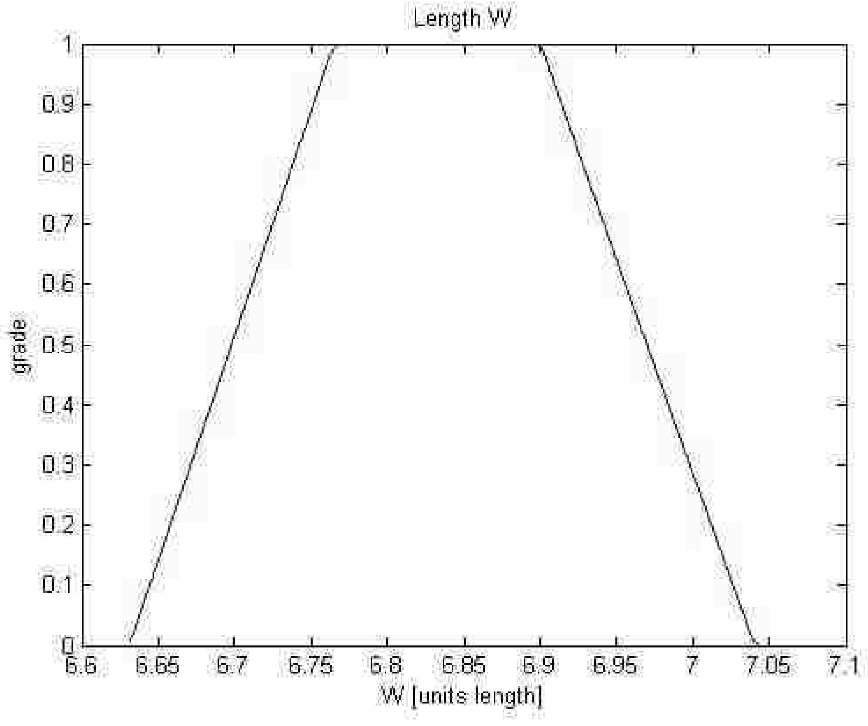

*Figure 4-12. Fuzzy Synthesis Solution of Length W*

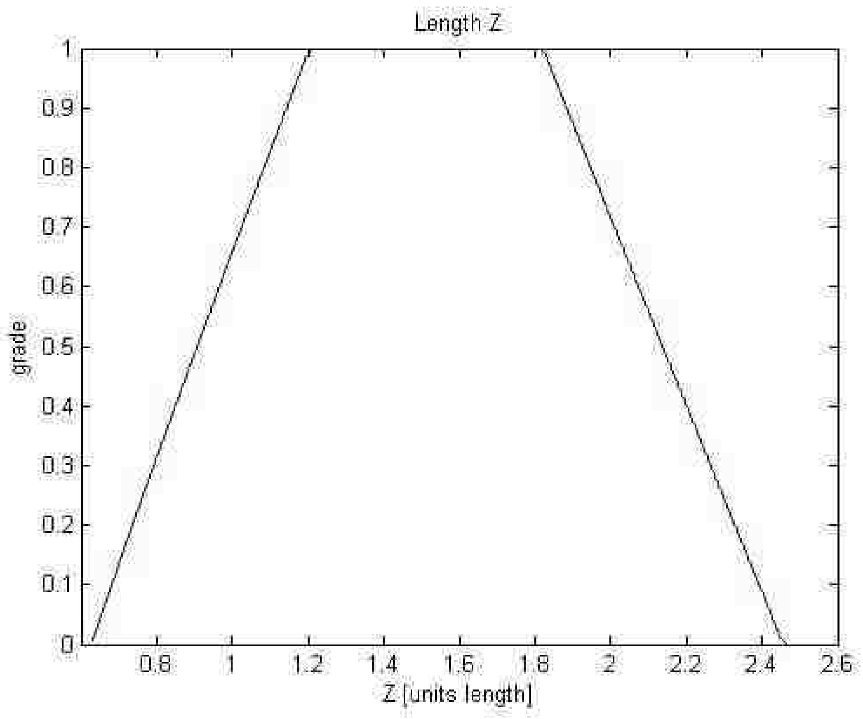

*Figure 4-13. Fuzzy Synthesis Solution of Length Z*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | 0.0558 | 1.45% |
| Wy | 6.832 | 6.8329 | 0.013% |
| Zx | 1.179 | 1.1847 | 0.483% |
| Zy | 0.940 | 0.9420 | 0.213% |

*Table 4-3. Results of Three-Position Four-bar Linkage Synthesis.*

Again, this error is attributed to rounding error throughout the mathematical processes, as well as how the fuzziness of $P_{21}$ and $\delta_2$ are defined.

Three-position fuzzy synthesis is useful if there is a mechanism that must pass through a specific region going from point one to point three because it will interfere with an obstacle if it is allowed to go move without restriction.

## 5.4 Mechanism Synthesis with Fuzzy Fixed Pivot Points

Instead of fuzzifying one of the positions of the four-bar linkage, this section will detail the fuzzification of the fixed pivot. This method is done much differently than the fuzzification of any of the positions of the linkage. The analytical method to synthesize a mechanism which has a fixed pivot is the following:

Vector loop equations for all three positions of the mechanism:

$$we^{j\theta} + ze^{j\emptyset} = \boldsymbol{R}_1$$

$$we^{j(\theta+\beta_2)} + ze^{j(\emptyset+\alpha_2)} = \boldsymbol{R_2}$$

$$we^{j(\theta+\beta_3)} + ze^{j(\emptyset+\alpha_3)} = \boldsymbol{R_3}$$

If $\boldsymbol{W} = we^{j\theta}$ and $\boldsymbol{Z} = ze^{j\emptyset}$ then the above equations can be simplified to:

$$\boldsymbol{W} + \boldsymbol{Z} = \boldsymbol{R_1}$$

$$\boldsymbol{W}e^{j\beta_2} + \boldsymbol{Z}e^{j\alpha_2} = \boldsymbol{R_2}$$

$$\boldsymbol{W}e^{j\beta_3} + \boldsymbol{Z}e^{j\alpha_3} = \boldsymbol{R_3}$$

There is only a solution to the above system of equations if the determinant is equal to zero, so this provides an equation to solve for the values of $\beta_2$ and $\beta_3$.

$$\left(\boldsymbol{R}_3e^{j\alpha_2} - \boldsymbol{R}_2e^{j\alpha_3}\right) + e^{j\beta_2}\left(\boldsymbol{R}_1e^{j\alpha_3} - \boldsymbol{R}_3\right) + e^{j\beta_3}\left(\boldsymbol{R}_2 - \boldsymbol{R}_1e^{j\alpha_2}\right) = 0$$

And this can be further simplified by stating:

$$A = \boldsymbol{R}_3e^{j\alpha_2} - \boldsymbol{R}_2e^{j\alpha_3}$$
$$B = \boldsymbol{R}_1e^{j\alpha_3} - \boldsymbol{R}_3$$
$$C = \boldsymbol{R}_2 - \boldsymbol{R}_1e^{j\alpha_2}$$

So,

$$A + Be^{j\beta_2} + Ce^{j\beta_3} = 0$$

After accounting for the real and imaginary components of the above equation, the solution becomes:

$$\beta_3 = 2\tan^{-1}\left(\frac{K_2 \pm \sqrt{{K_1}^2 + {K_2}^2 - {K_3}^2}}{K_1 + K_3}\right); \quad \beta_2 = \cos^{-1}\left(\frac{A_5 \sin\beta_3 + A_3 \cos\beta_3 + A_6}{A_1}\right)$$

Where,

$$K_1 = A_2A_4 + A_3A_6$$
$$K_2 = A_3A_4 + A_5A_6$$
$$K_3 = \frac{{A_1}^2 - {A_2}^2 - {A_3}^2 - {A_4}^2 - {A_6}^2}{2}$$

And,

$$A_1 = -{C_3}^2 - {C_4}^2 \qquad A_4 = C_2C_3 + C_1C_4$$
$$A_2 = C_3C_6 - C_4C_5 \qquad A_5 = C_4C_5 - C_3C_6$$
$$A_3 = -C_4C_6 - C_3C_5 \qquad A_6 = C_1C_3 - C_2C_4$$

And,

$$C_1 = R_3\cos(\alpha_2 + \zeta_3) - R_2\cos(\alpha_3 + \zeta_2)$$
$$C_2 = R_3\sin(\alpha_2 + \zeta_3) - R_2\sin(\alpha_3 + \zeta_2)$$

$$C_3 = R_1 \cos(\alpha_3 + \zeta_1) - R_3 \cos \zeta_3$$

$$C_4 = -R_1 \sin(\alpha_3 + \zeta_1) + R_3 \sin \zeta_3$$
$$C_5 = R_1 \cos(\alpha_2 + \zeta_1) - R_2 \sin \zeta_2$$
$$C_6 = -R_1 \sin(\alpha_2 + \zeta_1) + R_2 \sin \zeta_2$$

As it can be seen, this procedure is cumbersome, and there is much room for error. The first time this method was used to calculate the betas for the fuzzy fixed pivot mechanism synthesis, the solution was never able to reach a stable answer. Because of this, a new method was chosen to compute the betas for the fuzzy fixed pivot mechanism synthesis.

The alternative method chosen to achieve the fuzzy betas was an iterative method. The Newton-Raphson method in conjunction with the vertex method was used to formulate the results of the synthesis. The Newton-Raphson method was used to find the best value of beta based on the proper parts of the equations.

### 4.4.1 The Use of the Newton-Raphson Method for Mechanism Synthesis with a Fuzzy Fixed Pivot

The Newton Raphson Method is used to obtain the solution to a single non-linear equation as well as a system of equations using iterative schemes. The Newton-Raphson method is derived from the Taylor-series expansion, and the resulting equation becomes:

$$x_{i+1} = x_i - f(x_0)/f'(x_0)$$

The modified version of this equation, which is utilized for a system of non-linear equations is

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \boldsymbol{J}^{-1} \cdot \boldsymbol{f}(\boldsymbol{x}_i)$$

Where here

$$\left(\boldsymbol{R}_3 e^{j\alpha_2} - \boldsymbol{R}_2 e^{j\alpha_3}\right) + e^{j\beta_2}\left(\boldsymbol{R}_1 e^{j\alpha_3} - \boldsymbol{R}_3\right) + e^{j\beta_3}\left(\boldsymbol{R}_2 - \boldsymbol{R}_1 e^{j\alpha_2}\right) = 0$$

is written in terms of its real and imaginary components to compose $\boldsymbol{f}(x_i) = 0$.

Therefore,

Real: $f_1 = D_{1x} + \left[D_{2x}\cos\beta_2 - D_{2y}\sin\beta_2\right] + \left[D_{3x}\cos\beta_3 - D_{3y}\sin\beta_3\right] = 0$

Imaginary: $f_2 = D_{1y} + \left[D_{2y}\cos\beta_2 + D_{2x}\sin\beta_2\right] + \left[D_{3y}\cos\beta_3 - D_{3x}\sin\beta_3\right] = 0$

And $D_1 = R_3 e^{j\alpha_2} - R_2 e^{j\alpha_3}$; $D_2 = R_1 e^{j\alpha_3} - R_3$; $D_3 = R_2 - R_1 e^{j\alpha_2}$.

Now, the Jacobian must also be calculated and used to use the Newtown-Raphson method, and the Jacobian is:

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial \beta_2} & \frac{\partial f_1}{\partial \beta_3} \\ \frac{\partial f_2}{\partial \beta_2} & \frac{\partial f_2}{\partial \beta_3} \end{bmatrix}$$

And,

$$\frac{\partial f_1}{\partial \beta_2} = -D_{2x}\sin\beta_2 - D_{2y}\cos\beta_2$$

$$\frac{\partial f_1}{\partial \beta_3} = -D_{3x}\sin\beta_3 - D_{3y}\cos\beta_3$$

$$\frac{\partial f_2}{\partial \beta_2} = D_{2x}\cos\beta_2 - D_{2y}\sin\beta_2$$

$$\frac{\partial f_2}{\partial \beta_3} = D_{3x}\cos\beta_3 - D_{3y}\sin\beta_3$$

Now, [**J**] and [**f** ] can be used in the Newtown-Raphson equation to solve for the optimal value of $\beta_2$ and $\beta_3$. In order to account for the different alpha-cuts of the input, each of the betas is calculated based on its grade. Once the betas are determined for each of the two R's in each alpha-cut, the maximum and minimum betas are determined for the corresponding grade. This method is used for each alpha-cut from 0.0 to 1.0.

Once the betas are calculated, the lengths of the linkages, W and Z, are calculated using the following system of equations:

$$\boldsymbol{W}e^{j\beta_2} + \boldsymbol{Z}e^{j\alpha_2} = \boldsymbol{R}_2$$

$$\boldsymbol{W}e^{j\beta_3} + \boldsymbol{Z}e^{j\alpha_3} = \boldsymbol{R}_3$$

This method can be repeated to calculate the lengths of the other dyad of the four-bar linkage, US.

### 4.4.2 Fuzzy Mechanism Synthesis for Fuzzy Fixed Pivot

The previous section outlined the procedure for how to complete a mechanism synthesis for a fuzzy fixed pivot. Chapter 5 contains the tape unit example which displays this method. It is important to note, however, that the input, and therefore the output, of the fuzzy synthesis using the fuzzy fixed pivot is a triangular membership function. Chapter 8 uses the Newton-Raphson scheme to achieve the results of the lengths of W and Z will again be used to complete an analysis using a trapezoidal membership function.

# Chapter 5: Tape Unit Example of Using Fuzzy Logic Mathematics throughout Mechanism Synthesis

To illustrate the significance of the use of fuzzy logic within mechanism synthesis, the following example will demonstrate how these two subjects work, simultaneously. This example, extracted from Erdman and Sandor's text, involves a tape unit, which rotates into the dashboard, out of the line of visibility, to deter thieves from breaking into a car to steal the tape unit.

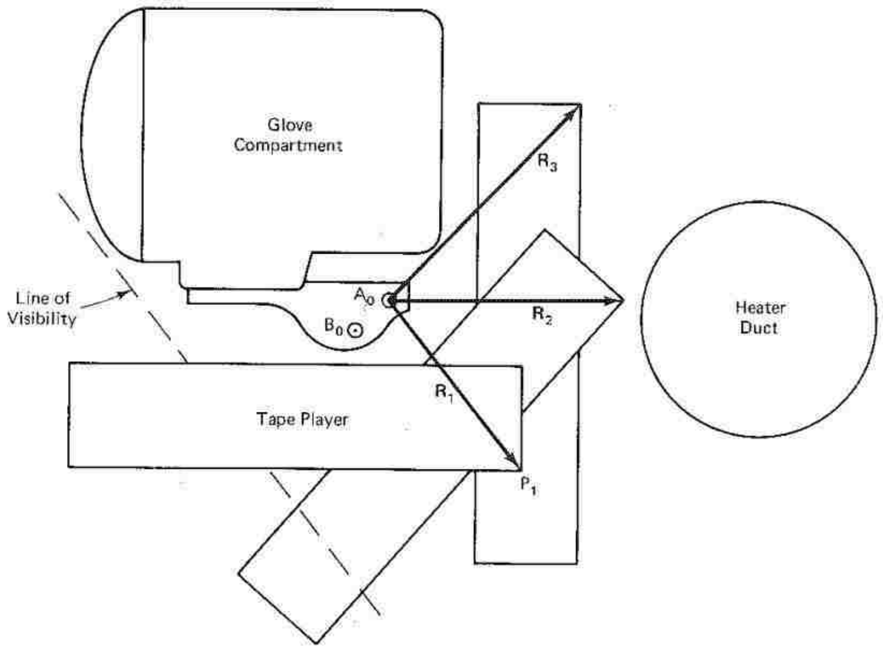

*Figure 5-1. Tape Unit Mechanism Synthesis Example.*
(Erdman, A. G.; Sandor, G. N. *Advanced Mechanism Design: Analysis and Synthesis, Vol. II. "*Figure 2.76: Four-bar motion-generation synthesis with prescribed ground pivots". 1984. Prentice Hall, Upper Saddle River, New Jersey.)

Figure 5-1 displays the tape unit in three different positions, designated by the vector $R_i$, where $i$=1, 2, 3. With this example, the tape unit must be located at the precise location for the starting position, $P_1$, as well as in the final position, $P_3$; however, the tape unit does not need to be located at exactly $P_2$, as long as the tape unit does not conflict with the heater duct. $P_2$ can now be represented as a fuzzy variable.

*Table 5-1. Variables of Crisp Tape Unit Example*

| Variables | Magnitude of Value |
|---|---|
| $P_{21}$ | 3.8321 |
| $\delta_2$ | 52.7414 |
| $P_{31}$ | 7.1730 |
| $\delta_3$ | 74.1424 |
| $\alpha_2$ | 50.7 |
| $\alpha_3$ | 91.9 |
| $\beta_2$ | 58.09 |
| $\beta_3$ | 122.70 |

## 5.1 Fuzzification of $P_2$ of Tape Unit

To fuzzify the location of $P_2$, both $P_{21}$ and $\delta_2$ must become fuzzy variables. In order to do this, $P_{21}$ was specified as [3.0, 3.5, 4.1642, 4.66] a length between 3.0 units and 4.66 units, where the interval of 3.5 units to 4.1642 units were set with a grade of 1. $\delta_2$ was specified as an angle between 45 and 60.5 degrees, with the interval of 50 to 55.4828 degrees having a grade of 1. From these intervals, a custom-made function, "fuzzyfun", which utilized MATLAB's "trapmf" function to create a trapezoidal membership function, was created to fuzzify these variables. Fuzzified $P_{21}$ and $\delta_2$ represent the area in which the second position of the tape unit could be located. The following graph displays the results of this fuzzification:

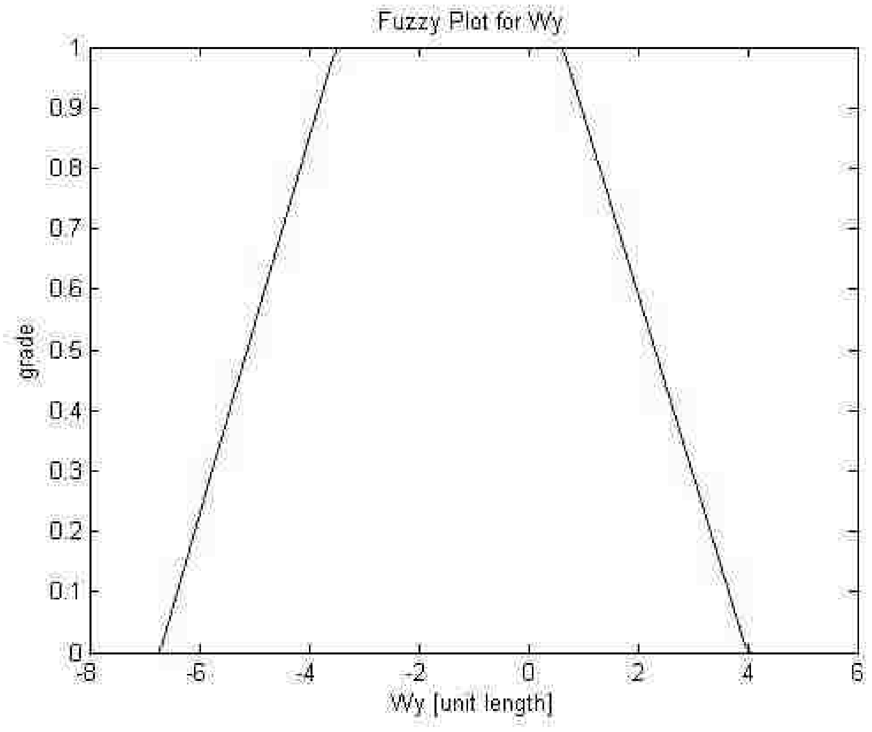

*Figure 5-2. Results for Wy from fuzzy synthesis of tape unit*

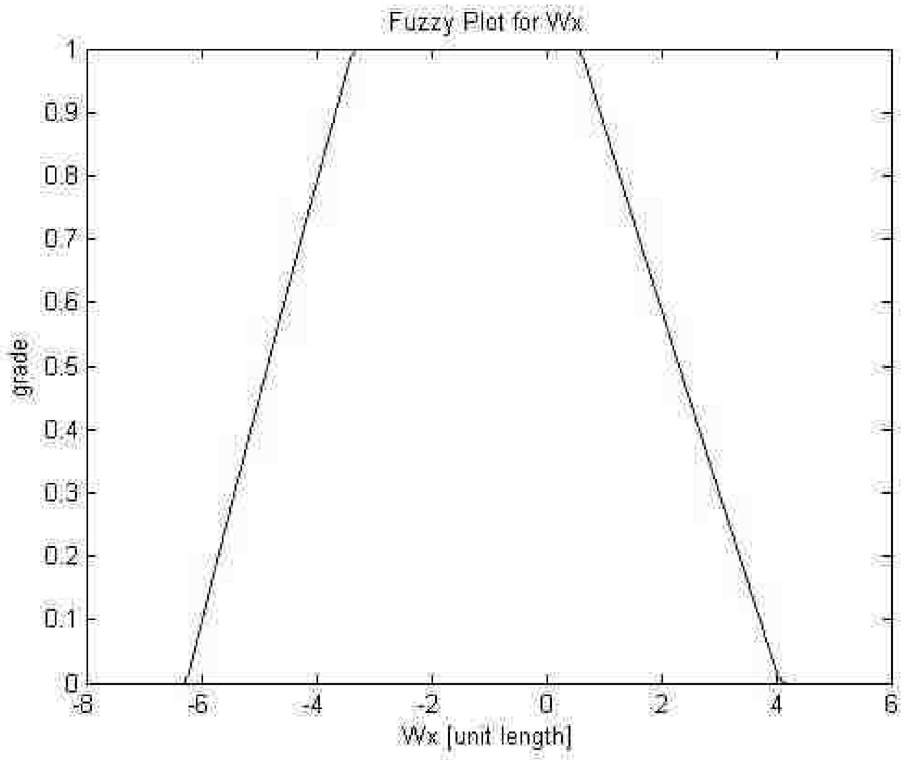

*Figure 5-3. Results for Wx from fuzzy synthesis of tape unit*

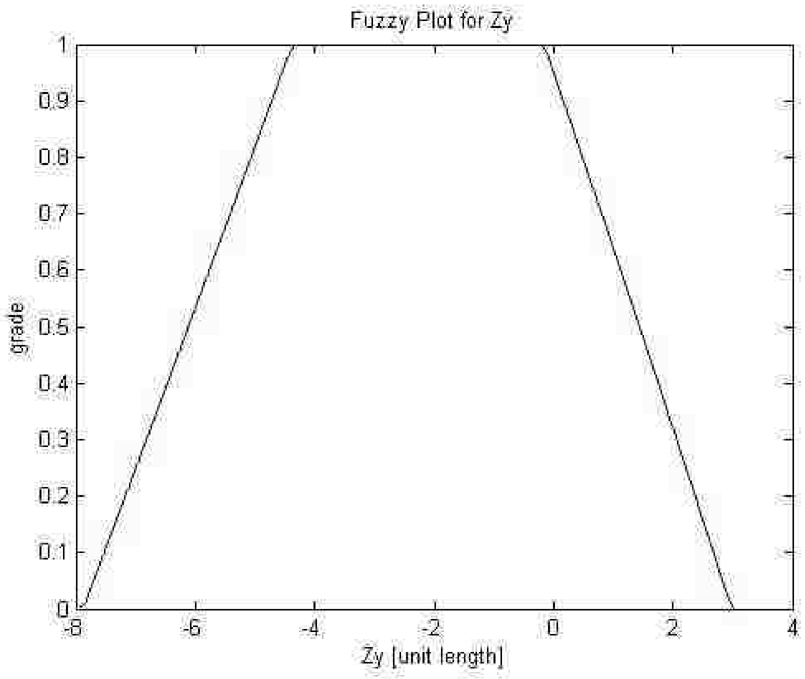

*Figure 5-4. Results for Zy from fuzzy synthesis of tape unit*

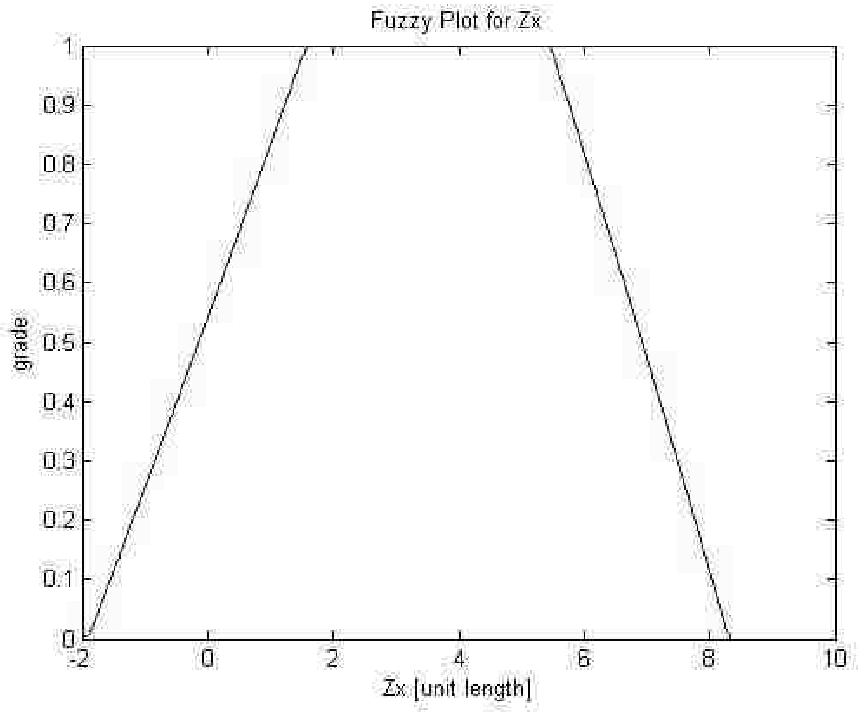

*Figure 5-5. Results for Zx from fuzzy synthesis of tape unit*

Table 5-2 compares the results of the crisp synthesis to the fuzzy synthesis. The results of the fuzzy synthesis are the defuzzified answers.

*Table 5-2. Comparison of Results for Fuzzy P2 Tape Unit*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | -1.42 | -1.3026 | 8.267% |
| Wy | -1.45 | -1.4126 | 2.579% |
| Zx | 3.56 | 3.4328 | 3.573% |
| Zy | -2.23 | -2.3128 | 3.713% |

The discrepancies in the results in Table 5-2 are due to the way the fuzziness of $P_{21}$ and $\delta_2$ were defined.

To complete the right hand side of the dyad, linkages U and S, the synthesis process explained above must be used to solve for the lengths of U and S. To do this, it can be assumed that $\gamma_2$=30.9 and $\gamma_3$=80.6. This produces the following lengths of S and U as those defined below in Figures 5.6 and 5.7.

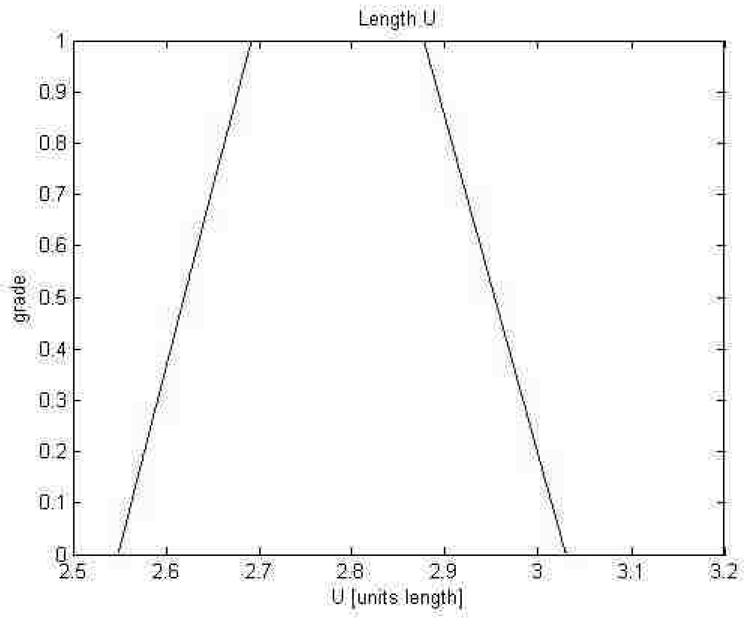

*Figure 5-6. Fuzzy Result of Length U.*

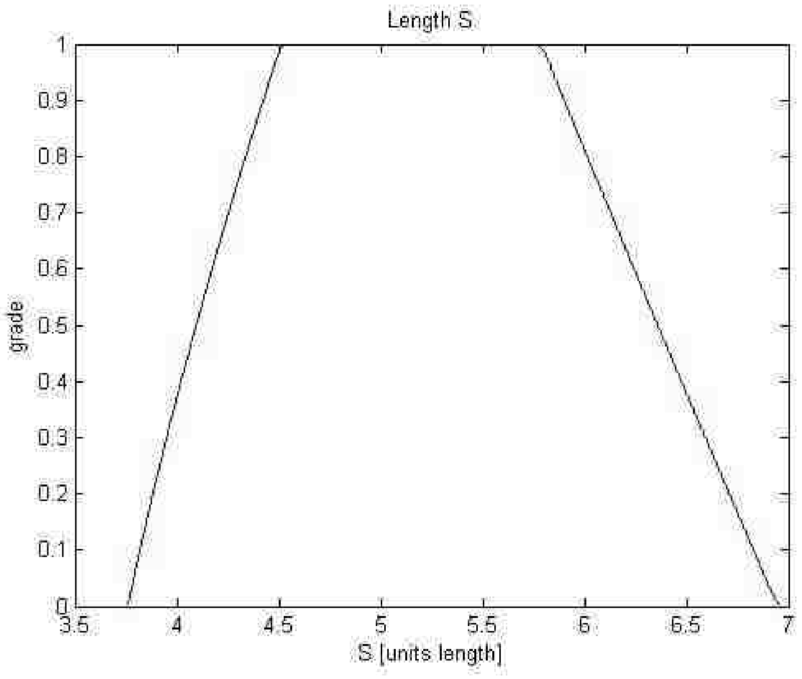

*Figure 5-7. Fuzzy Result of Length S.*

The defuzzified values are shown in the table below.

*Table 5-3. Results of RHS dyad fuzzy synthesis*

| Variable | Defuzzified Result [units] |
|---|---|
| $U_x$ | -1.9647 |
| $U_y$ | -1.9749 |
| $S_x$ | -5.1044 |
| $S_y$ | -0.6274 |

The results shown in Table 5-3, in conjunction with the results from Table 5-2 can be used to design the four-bar, three-position linkage desired.

## 5.2 Fuzzification of the Fixed Pivots

In the previous example, the position of $P_2$ was fuzzified and the mechanism synthesis was conducted using fuzzy logic mathematics. This next example, again using the tape unit scenario, will focus on the scenario as if the location of the fixed pivots was imprecise. This scenario is relatable because it describes a bolt or screw that may be placed a bit off from its exact specified location, which is likely to occur if the product is not manufactured by a machine, which has impeccable preciseness.

Fuzzifying both the x-location and the y-location of the fixed pivot with a triangular membership function will create a range of locations surrounding the specified exact location. Figure 5-8 displays this concept, where the ring around the colored pivot location are also possible locations of the pivot locations.

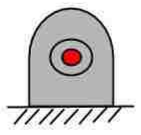
*Figure 5-8. Fuzzy Pivot Schematic*

The following figures, Figure 5-9 and Figure 5-10, display the resulting fuzzification of the x and y lengths of the vector R1. This fuzzification is the result of the fuzzy fixed pivot.

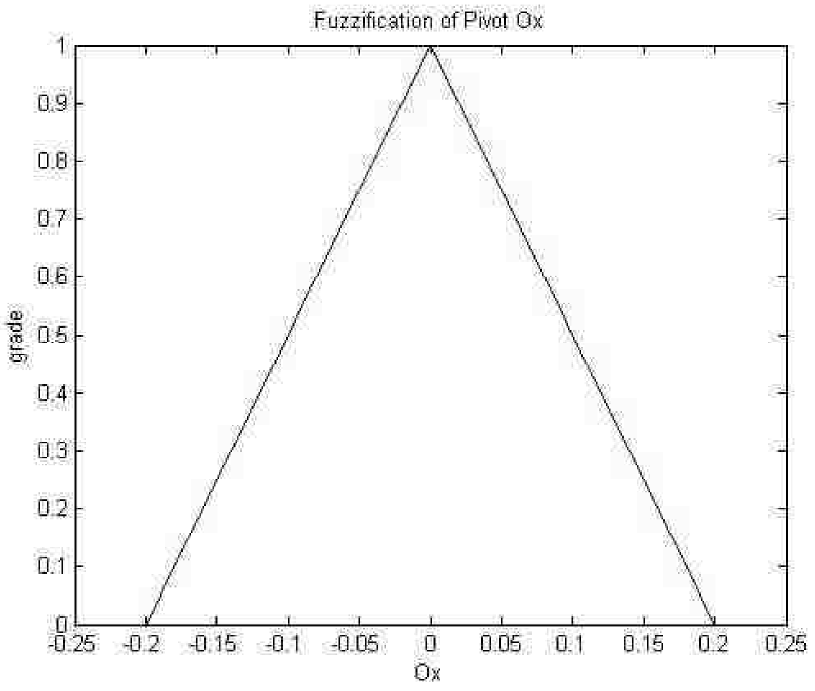

*Figure 5-9. Fuzzification of Pivot's x-location.*

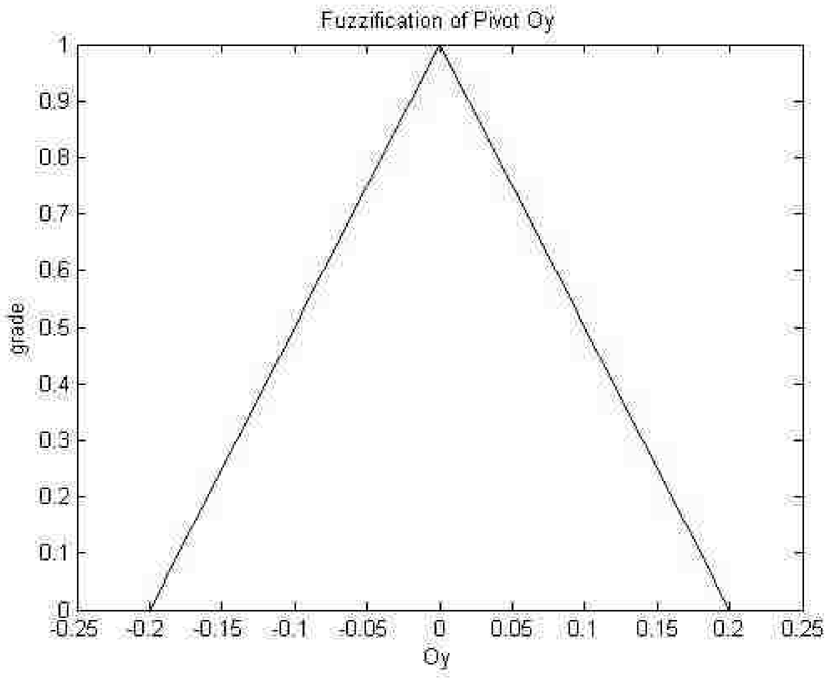

*Figure 5-10. Fuzzification of pivot's y-location*

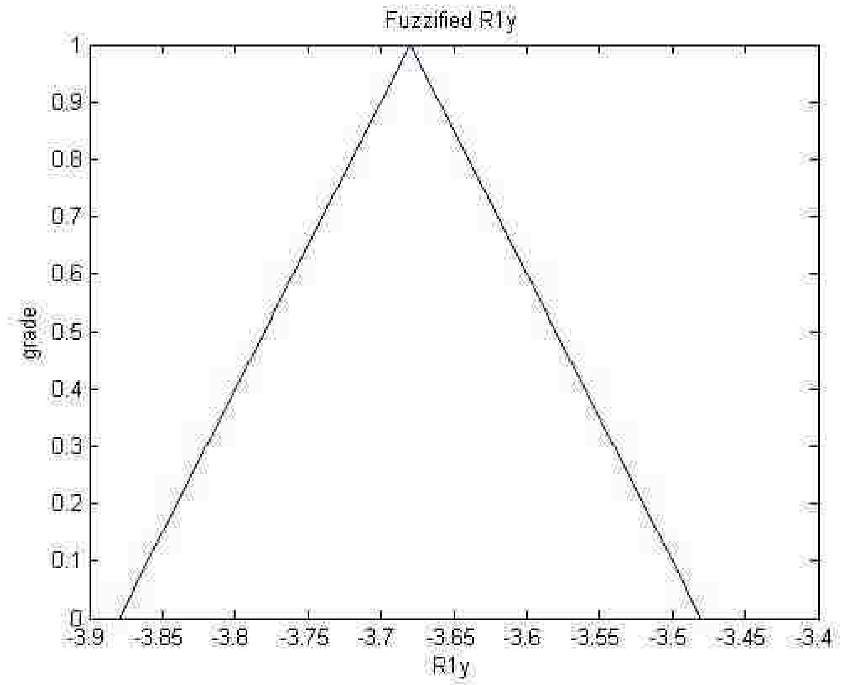

*Figure 5-11. Fuzzification of R1y.*

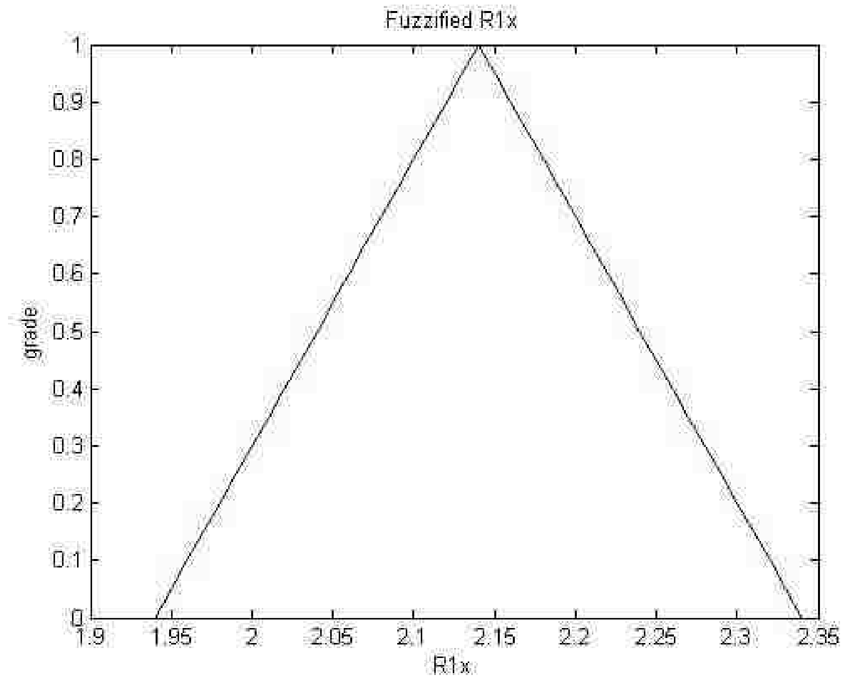

*Figure 5-12. Fuzzification of R1x.*

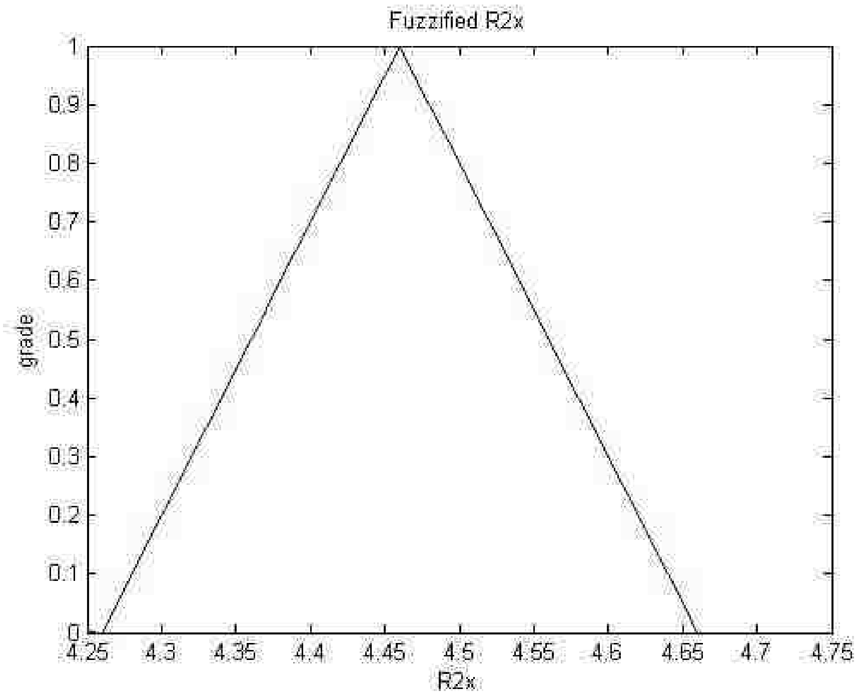

*Figure 5-13. Fuzzification of R2x*

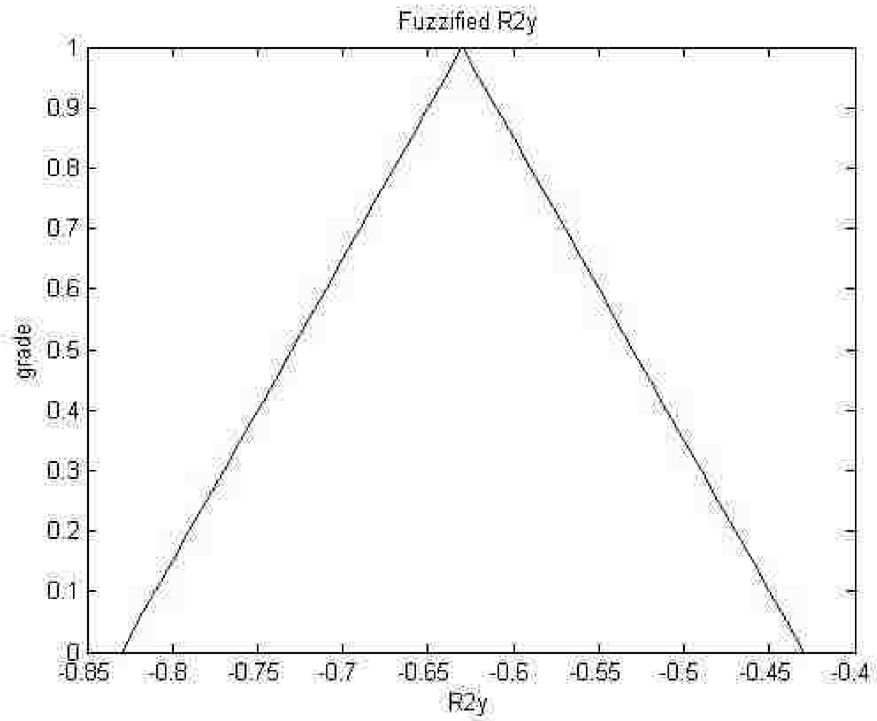

*Figure 5-14. Fuzzification of R2y*

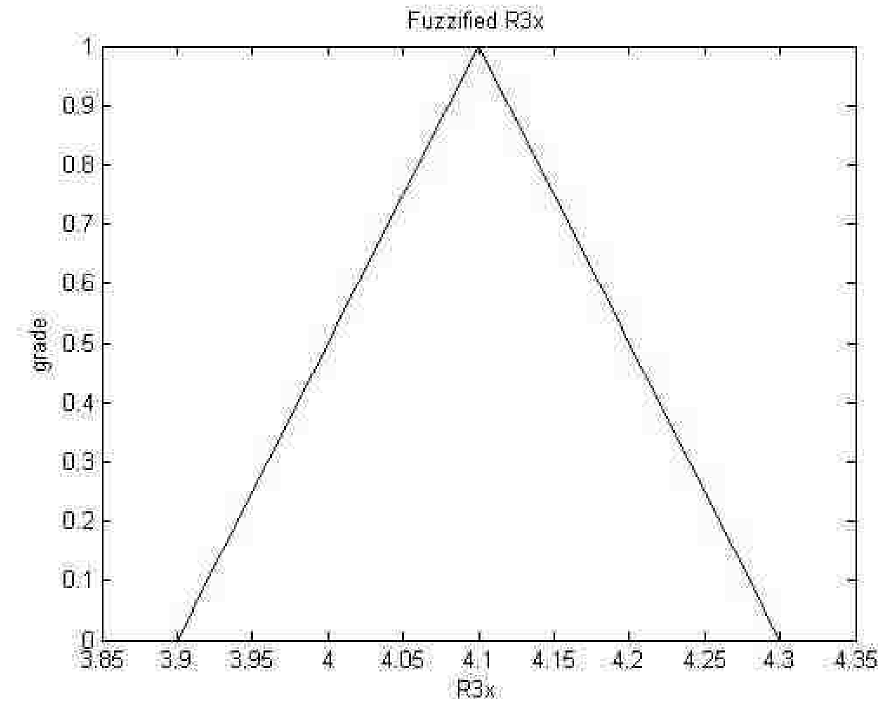

*Figure 5-15. Fuzzification of R3x*

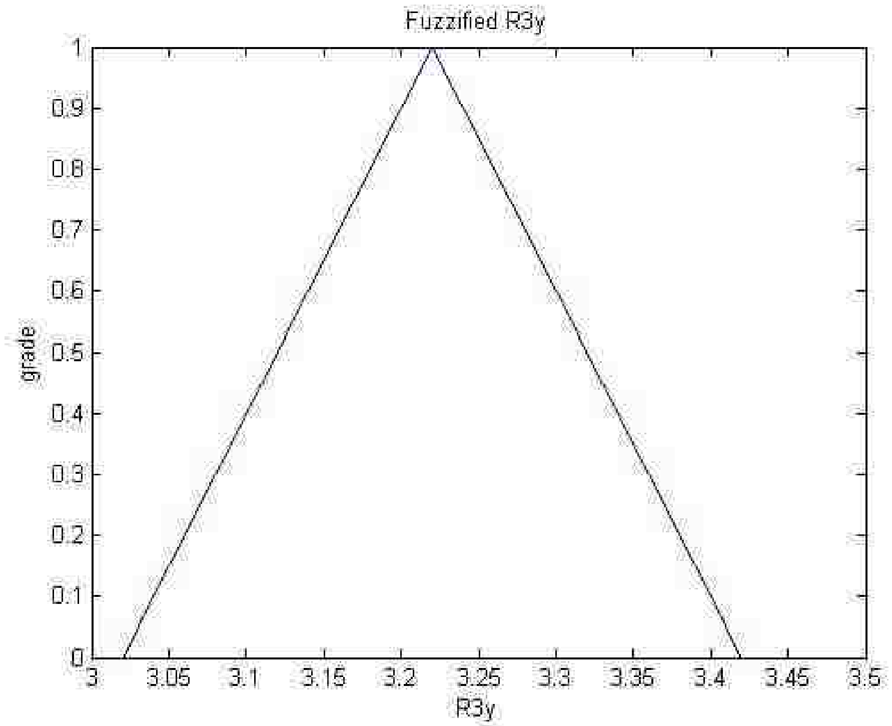

*Figure 5-16. Fuzzification of R3y*

The following four figures are the results of the mechanism synthesis with a fuzzy fixed pivot.

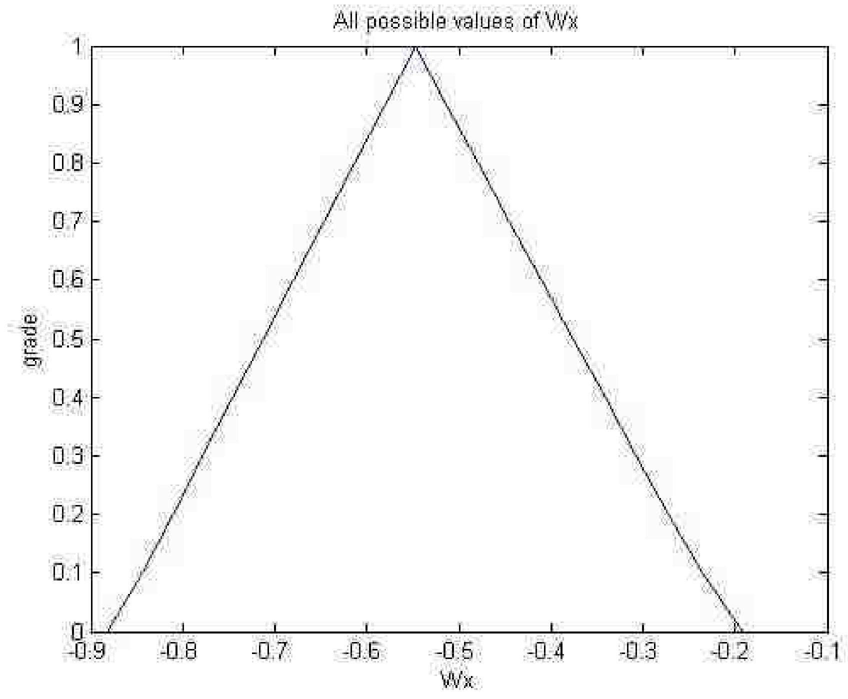

*Figure 5-17. Results of fuzzy fixed pivot synthesis: Wx*

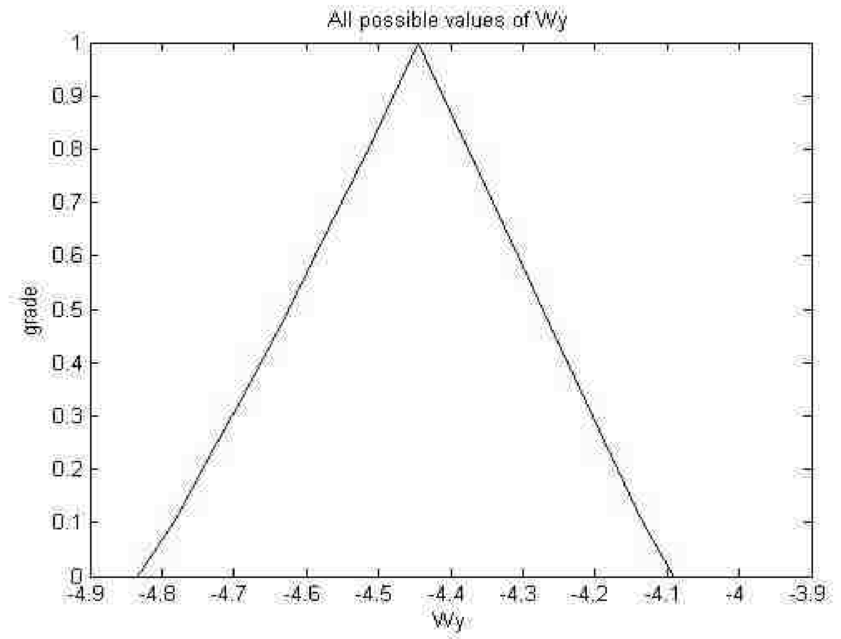

*Figure 5-18. Results of fuzzy fixed pivot synthesis: Wy*

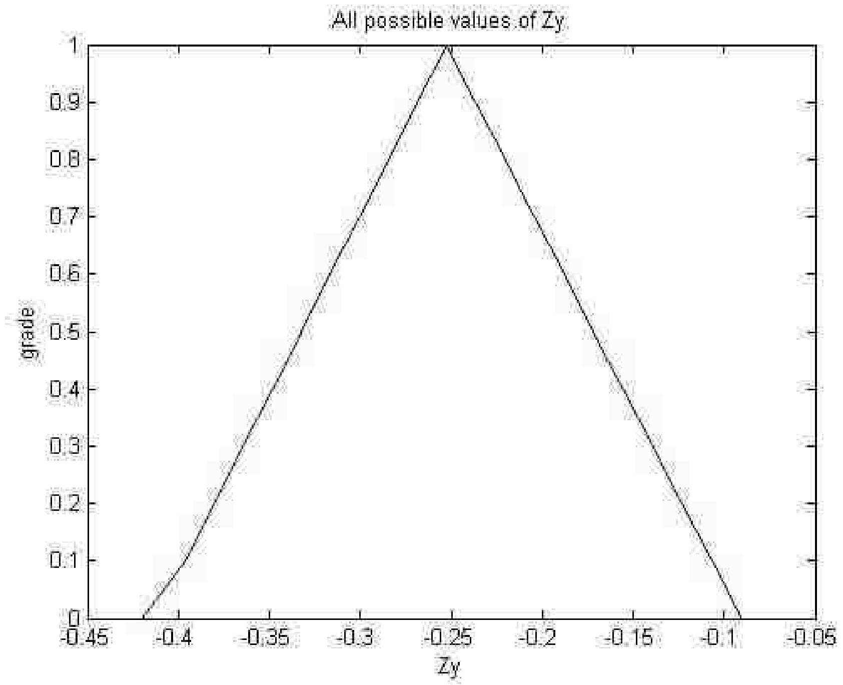

*Figure 5-19 Results of fuzzy fixed pivot synthesis: Zy.*

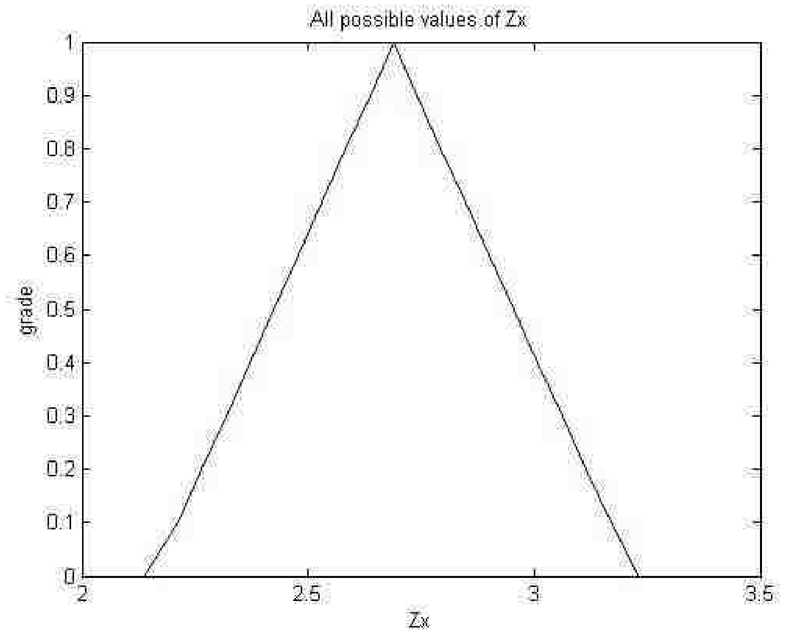

*Figure 5-20. Results of fuzzy fixed pivot synthesis: Zy*

*Table 5-4. Results of fuzzy fixed pivot synthesis*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | -0.5467 | -0.5454 | 0.2377% |
| Wy | -4.4443 | -4.4468 | 0.0562% |
| Zx | 2.6889 | 2.6892 | 0.0112% |
| Zy | -0.2525 | -0.2523 | 0.0792% |

Comparing the results of the crisp synthesis (desired results) to the actual results, the defuzzified results acquired from the synthesis using fuzzy logic mathematics, it can be seen that there is little error in performing fuzzy synthesis, yet it accounts for uncertainty and impreciseness in the input values.

If it is assumed that the tape unit is 2 units high in its initial position. The same procedure can be used to find the right-hand dyad linkage lengths. Since the specific values for **R**'s are not given, we can assume realistic values in order to complete the mechanism synthesis.

$$\overline{R}_{1x} = 2.50 \qquad \overline{R}_{1y} = -3.75$$

$$\overline{R}_{2x} = 4.5 \qquad \overline{R}_{2y} = -0.5$$

$$\overline{R}_{3x} = 4.12 \qquad \overline{R}_{3y} = 3.25$$

$$\alpha_2 = 50.7 \qquad \alpha_3 = 91.9$$

With these new values for the lengths of the vector from the new pivot, specified as $B_0$, to the bottom point on the tape unit, the following figure are the new fuzzy R vectors.

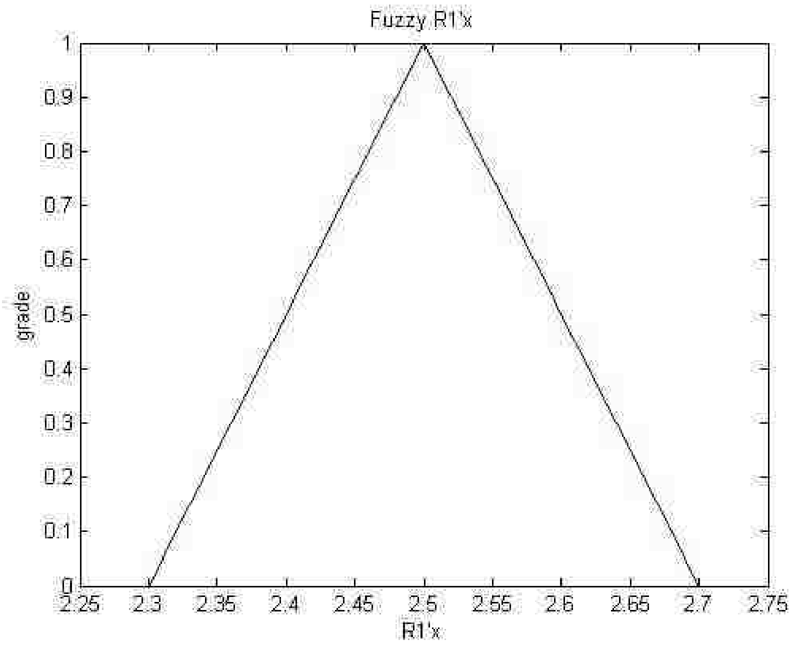

*Figure 5-21. Fuzzification of length R1'x.*

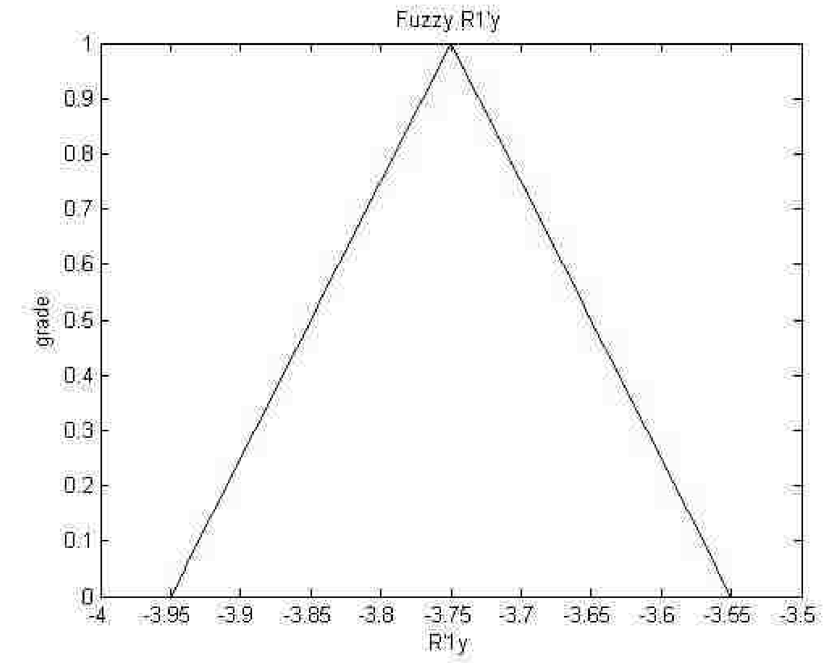

*Figure 5-22. Fuzzification of R1'y.*

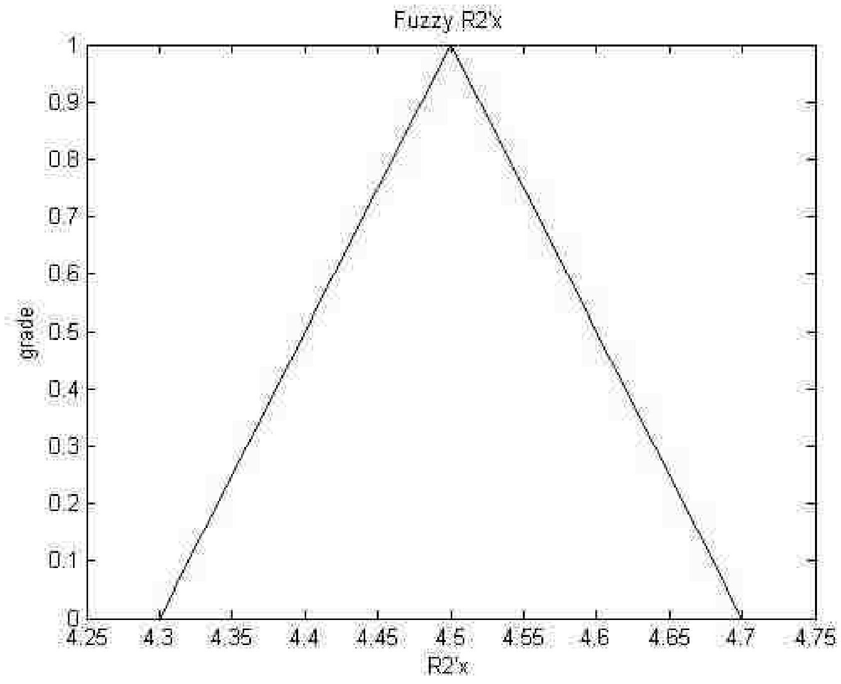

*Figure 5-23. Fuzzification of R2'x.*

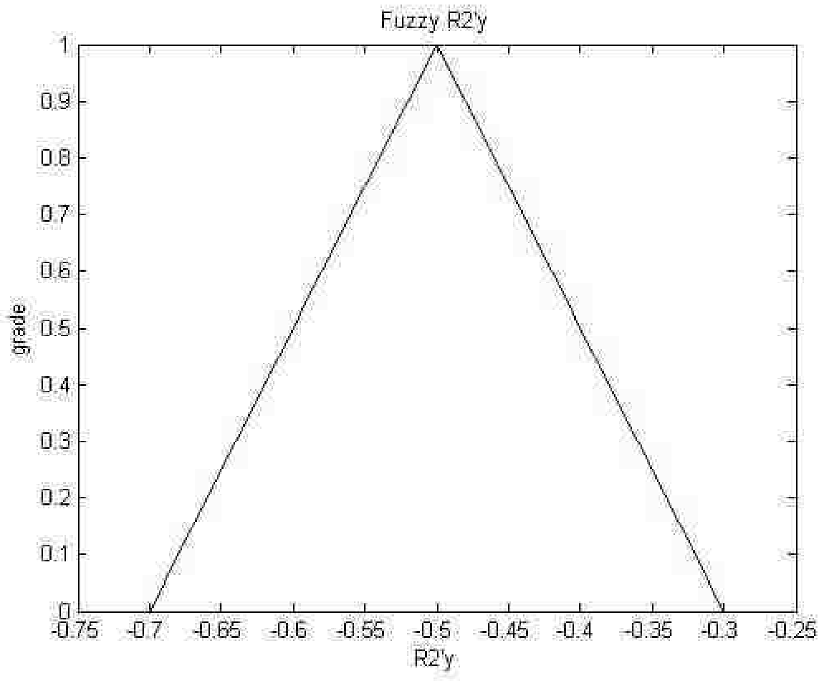

*Figure 5-24. Fuzzification of R2'y.*

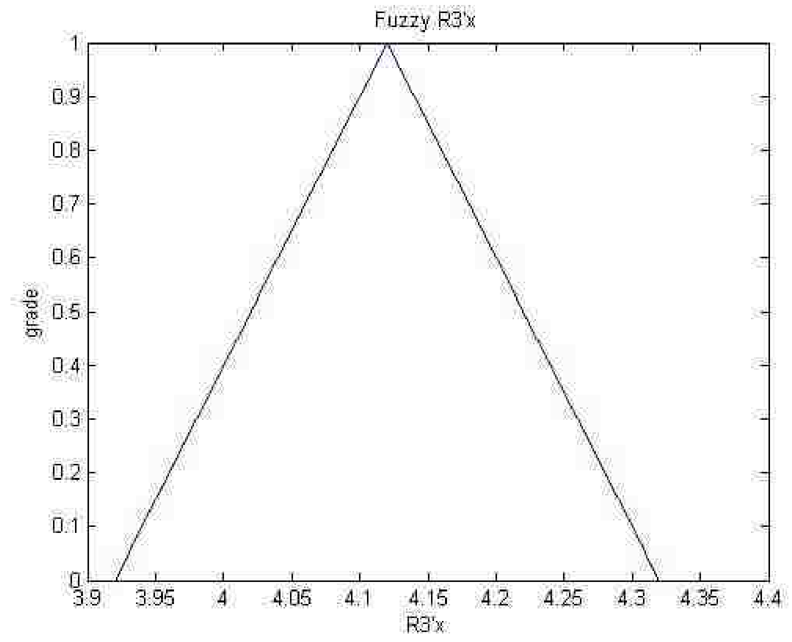

*Figure 5-25. Fuzzification of R3'x.*

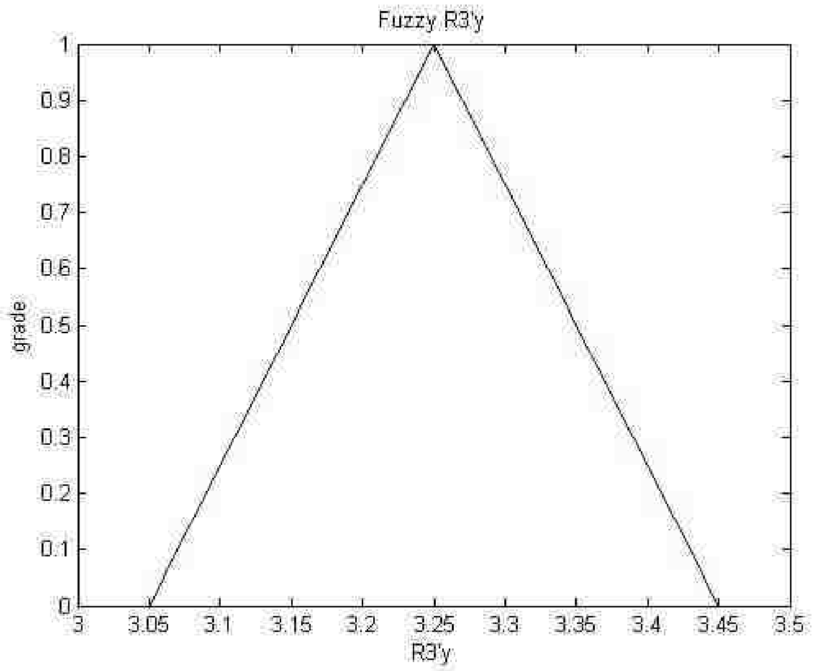

*Figure 5-26. Fuzzification of R3'y.*

The following table displays the results of the fuzzy fixed pivot mechanism synthesis for the right-hand dyad, US, of the four-bar linkage.

*Table 5-5. Results for RHS dyad of Fuzzy Fixed Pivot Synthesis*

| Variable | Results of Fuzzy Mechanism Synthesis |
|---|---|
| Ux | -0.5385 |
| Uy | -4.3951 |
| Sx | 3.0431 |
| Sy | -0.1142 |

Because there is no results of this synthesis in the text, from which this example was extracted, there are no values that we can compare the results to. It can be assumed, however, that these results exhibit little error based on the results concluded from the WZ dyad, discussed above. These results can be seen in the following four figures of the fuzzy outputs of the synthesis.

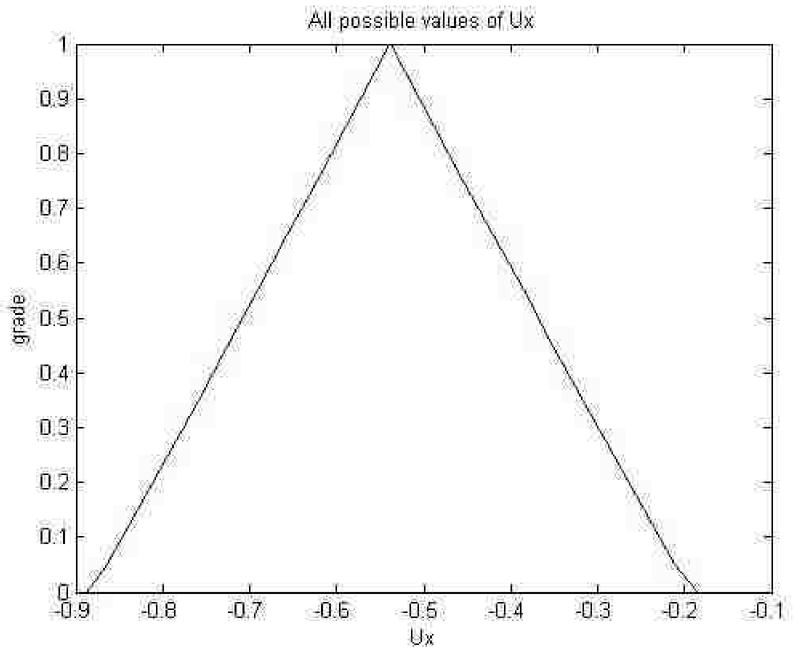

*Figure 5-27. Results for Fuzzy Ux*

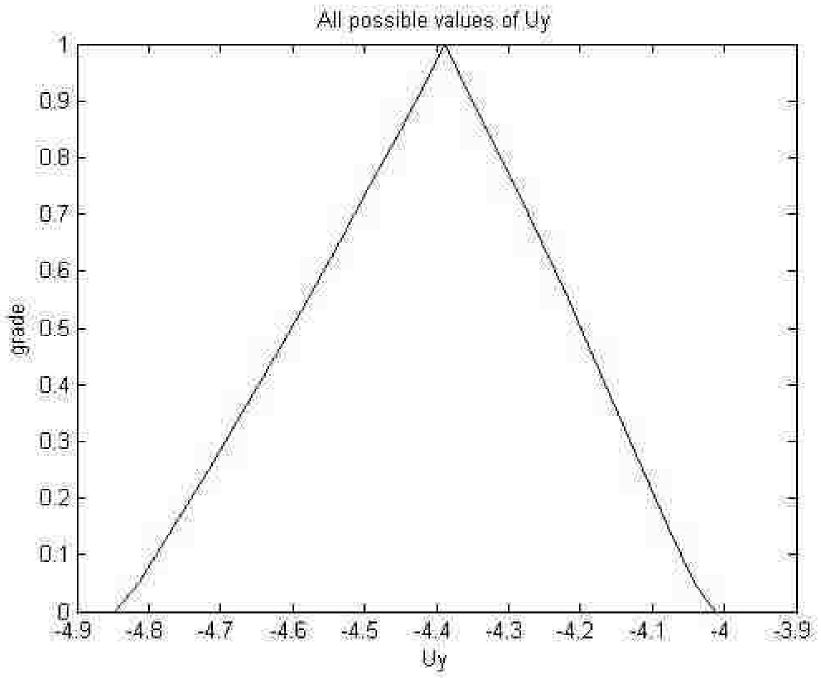

*Figure 5-28. Results for Fuzzy Uy.*

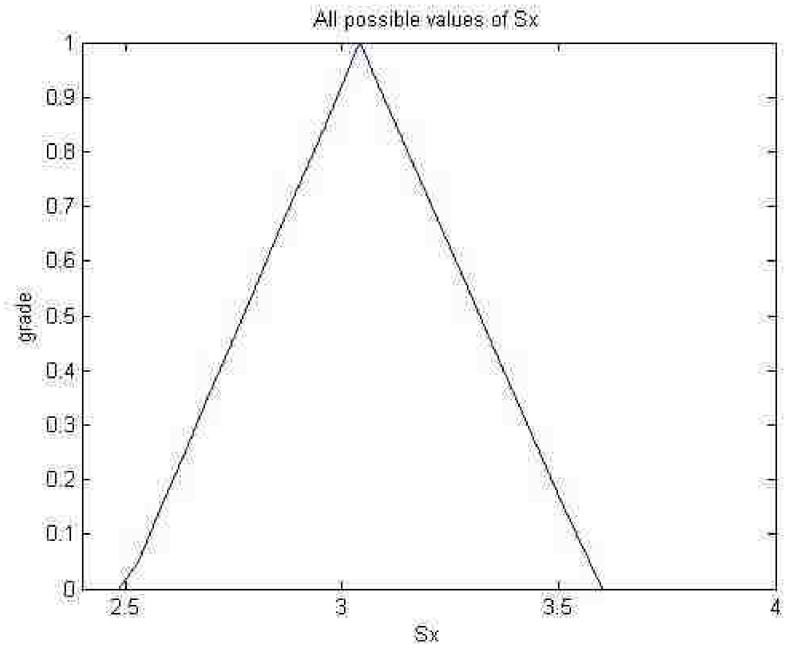

*Figure 5-29. Results for Fuzzy Sx.*

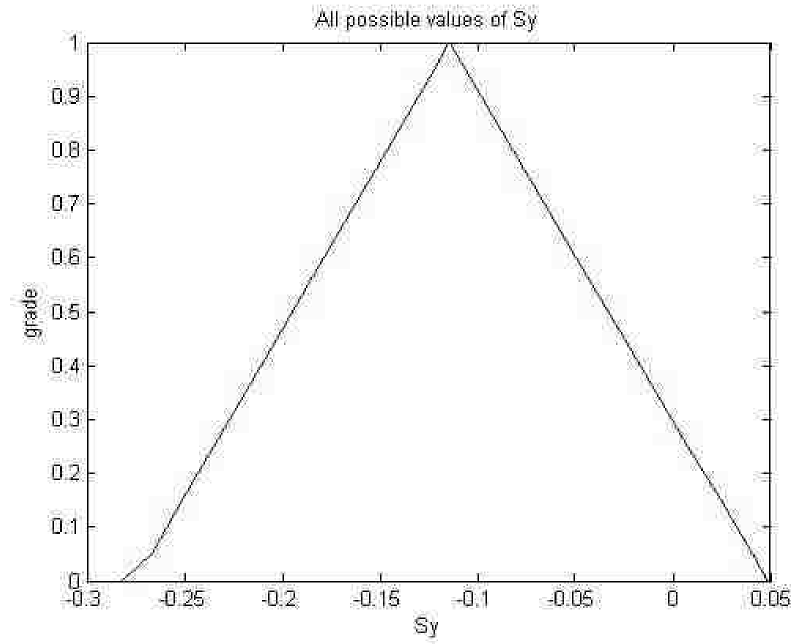

*Figure 5-30. Results for Fuzzy Sy.*

It can be seen, like the syntheses before, that the results of the fuzzy synthesis take on the basic shape of the membership function as that of the fuzzy inputs into the synthesis.

Now that there are results for both dyads of the mechanism, the WZ dyad and the US dyad, the entire mechanism has been designed to accommodate the three positions specified, Point 1, Area 2, and Point 3.

## 5.3 Example Conclusion

Fuzzifying variables of the mechanism synthesis, or any other aspect of mechanical design, will result in the fuzzification of other variables within the situation. The results of the synthesis or design will also be fuzzy, so there will be multiple answers the user can choose from, based on their needs and conditions.

# Chapter 6: Comparing Fuzzy Spans

Increasing or decreasing the range of fuzziness of a number can affect the final results of the mechanism synthesis. The following analysis displays the results of increasing fuzziness, where the fuzziness of $P_{21}$ and $\delta_2$ increase or decrease by the specified percentage between the crisp number and the points c and b, respectively, according to Figure 4-1. The values of a and d, according to Figure 4-1, are calculated by respectively decreasing or increasing the variable by twice the specified percentage. With this alteration, each crisp number is fuzzified according to the value of the specific variable. This can be shown in Figure 6-1.

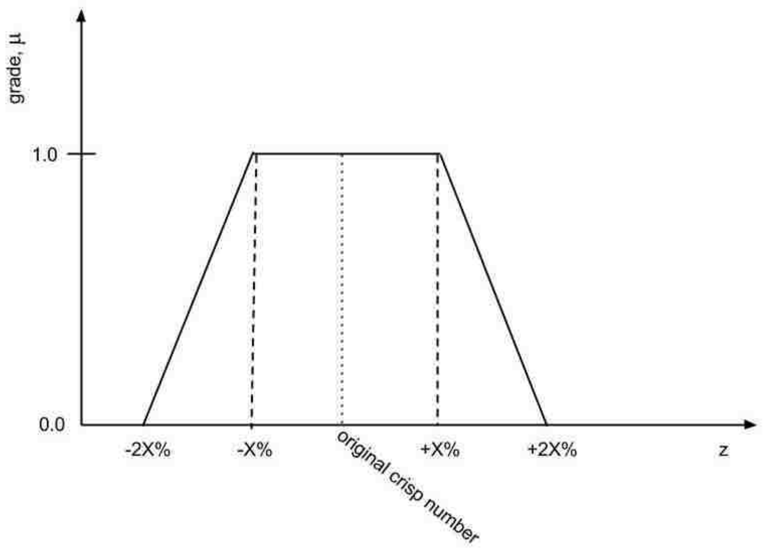

*Figure 6-1. Definition of spans*

The following results are computed from the example in Section 5.2.2, where only the location of $P_2$ is considered fuzzy.

## 6.1 Fuzzy Variables Span 5%

Figures 6-2 and 6-3 display the fuzzification of $P_{21}$ and $\delta_2$, which are the variables that must be fuzzified in order to represent the location of $P_2$ as an area. The fuzzy lengths are represented with a 5% span about the crisp value of $P_{21}$ and $\delta_2$, respectively.

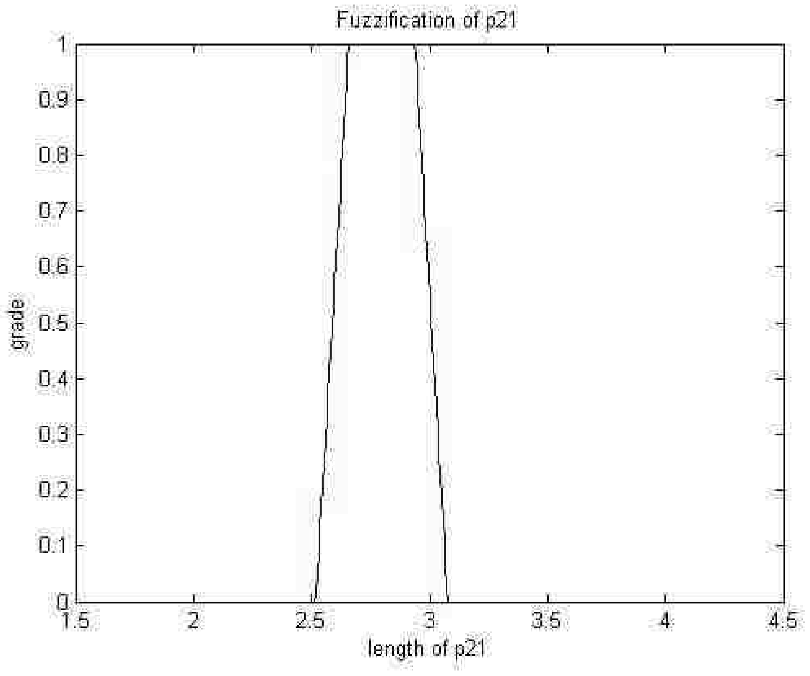

*Figure 6-2. Fuzzy length of P21 with 5% span*

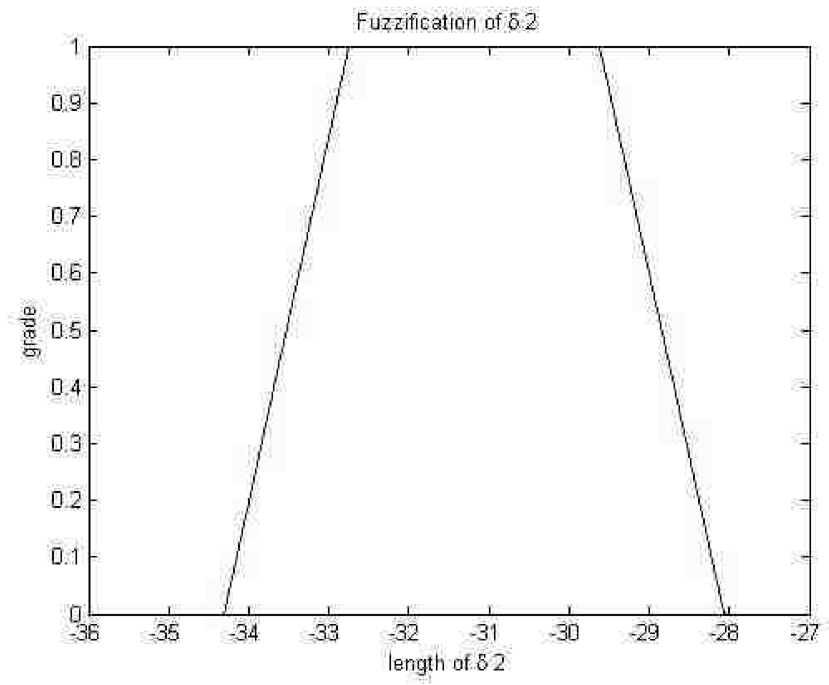

*Figure 6-3. Fuzzy length of δ2 with 5% span.*

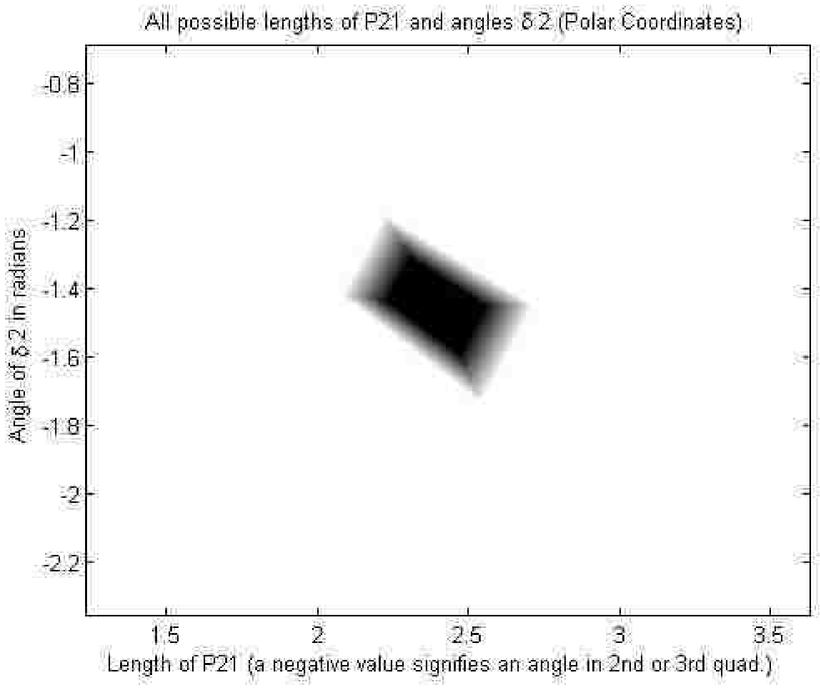

*Figure 6-4. Fuzzy locations of P2*

Figure 6-3 displays the possible locations of $P_2$, when the lengths of $P_{21}$ and $\delta_2$ are fuzzified as seen above. The following table is the result of the mechanism synthesis with the above fuzzifications. It can be seen that the results from the synthesis produce very little error.

*Table 6-1. Results from 5% span.*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | 0.0555 | 0.909% |
| Wy | 6.832 | 6.8333 | 0.019% |
| Zx | 1.179 | 1.1840 | 0.424% |
| Zy | 0.940 | 0.9439 | 0.415% |

The following figures, Figures 6-5 and 6-6 display the results from above, the graphs of the fuzzy length of W and the fuzzy length of Z, respectively.

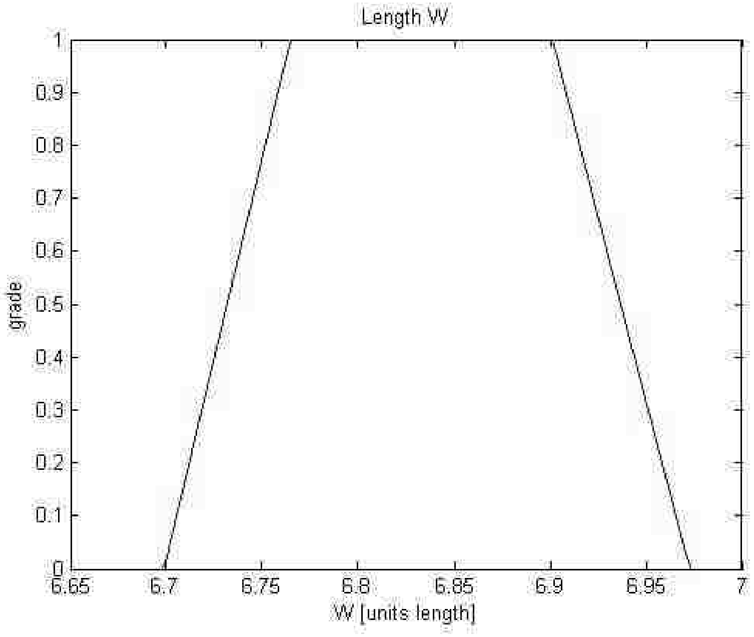

*Figure 6-5. Fuzzy Length of W (5% span).*

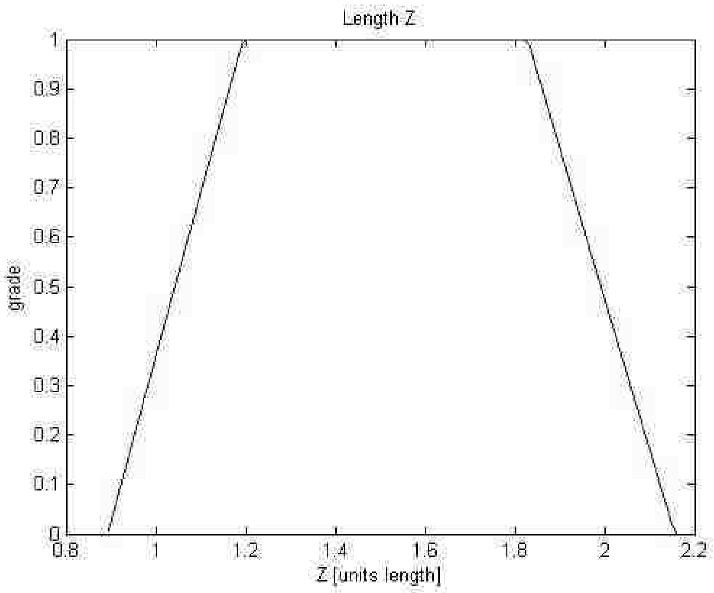

*Figure 6-6. Fuzzy Length of Z (5% span).*

## 6.2 Fuzzy variables span 10%

Figures 6-7 and 6-8 display the fuzzification of $P_{21}$ and $\delta_2$, which are the variables that must be fuzzified in order to represent the location of $P_2$ as an area. The fuzzy lengths are represented with a 10% span about the crisp value of $P_{21}$ and $\delta_2$, respectively.

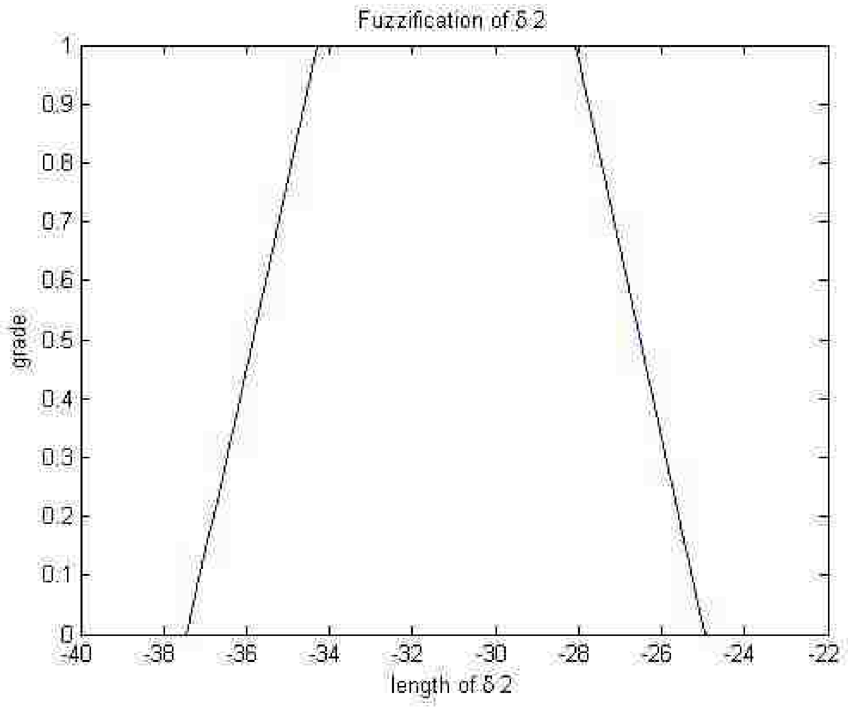

*Figure 6-7. Fuzzy length of δ2 with 10% span.*

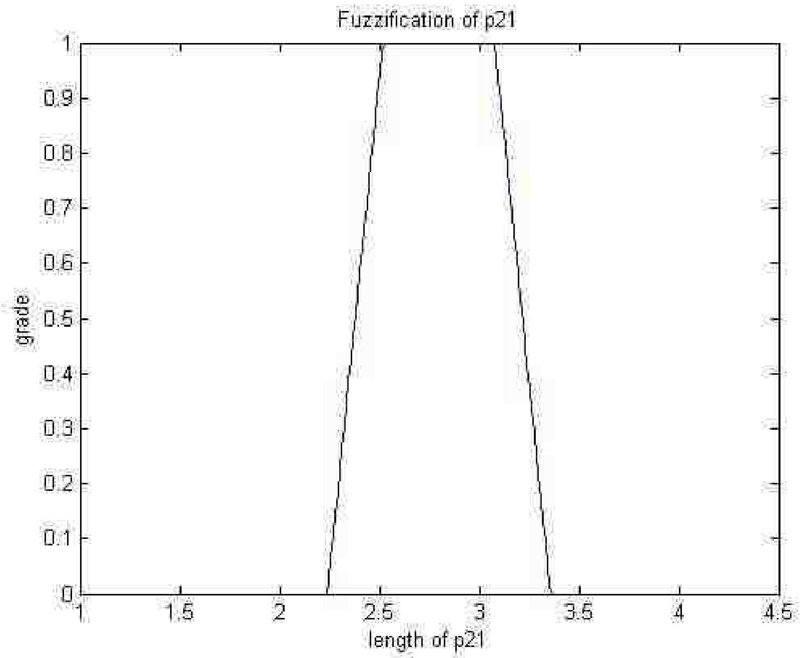

*Figure 6-8. Fuzzy Length of P21 with 10% span.*

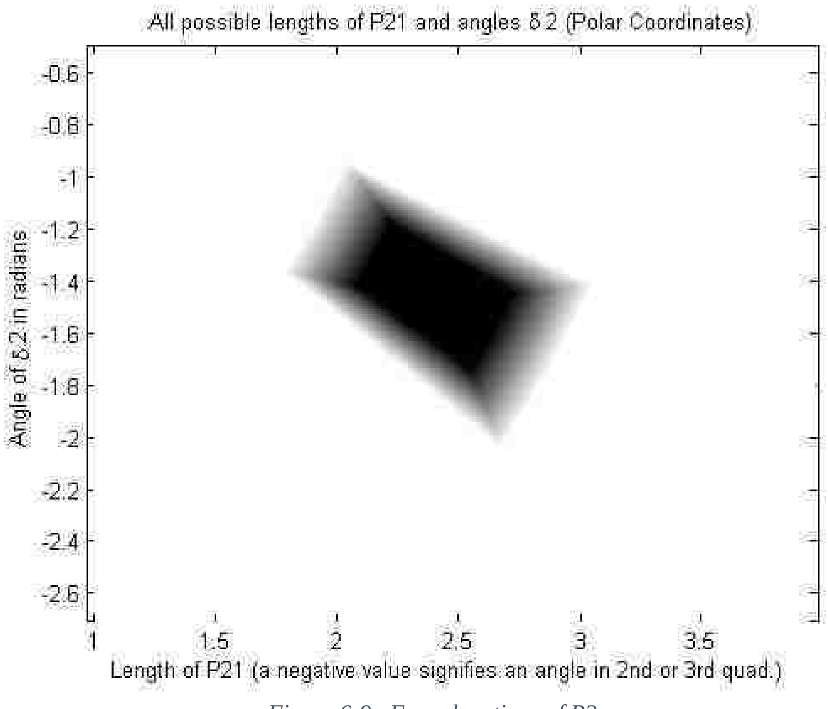

*Figure 6-9. Fuzzy locations of P2.*

Figure 6-9 displays the possible locations of $P_2$, when the lengths of $P_{21}$ and $\delta_2$ are fuzzified as seen above. The following table is the result of the mechanism synthesis with the above fuzzifications. It can be seen that the results from the synthesis produce more error compared to the 5% span.

*Table 6-2. Results from 10% span fuzzy synthesis*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | 0.0578 | 5.091% |
| Wy | 6.832 | 6.8379 | 0.0864% |
| Zx | 1.179 | 1.1986 | 1.662% |
| Zy | 0.940 | 0.9565 | 1.755% |

The following figures, Figures 6-10 and 6-11 display the results from above, the graphs of the fuzzy length of W and the fuzzy length of Z, respectively.

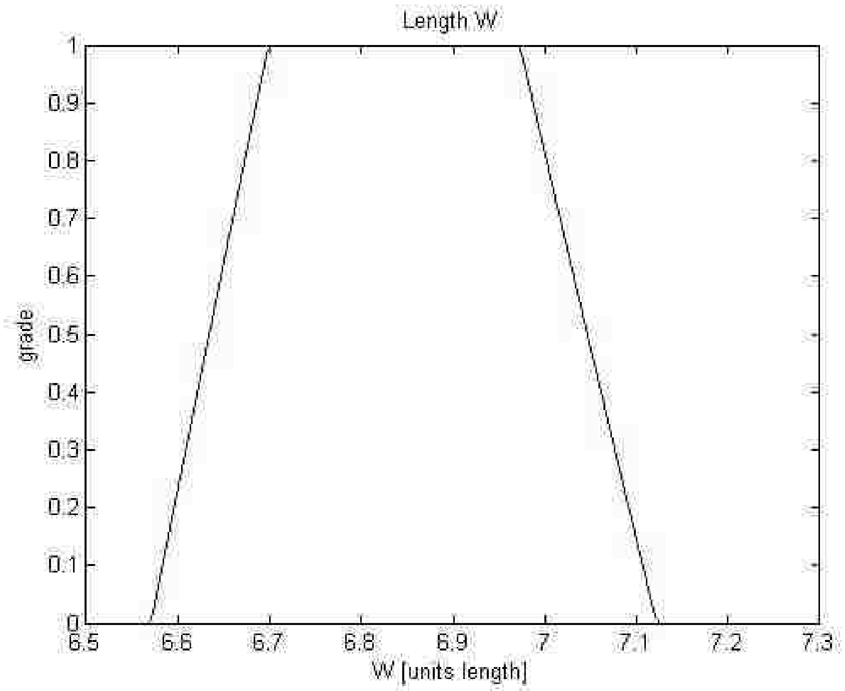

*Figure 6-10. Fuzzy length of W (10% span).*

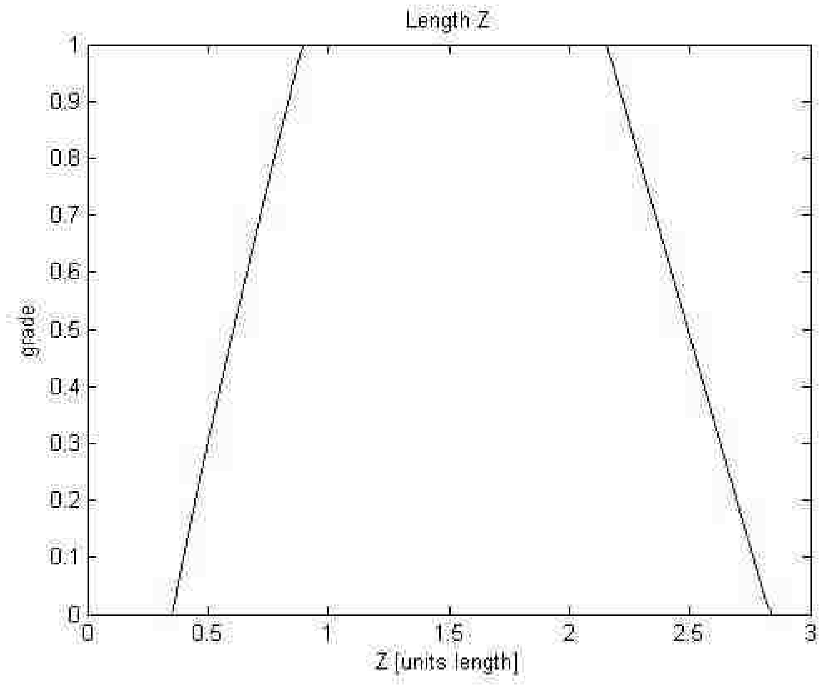

*Figure 6-11. Fuzzy length of Z (10% span).*

## 6.3 Fuzzy Variables span 20%

Figures 6-12 and 6-13 display the fuzzification of $P_{21}$ and $\delta_2$, which are the variables that must be fuzzified in order to represent the location of $P_2$ as an area. The fuzzy lengths are represented with a 20% span about the crisp value of $P_{21}$ and $\delta_2$, respectively.

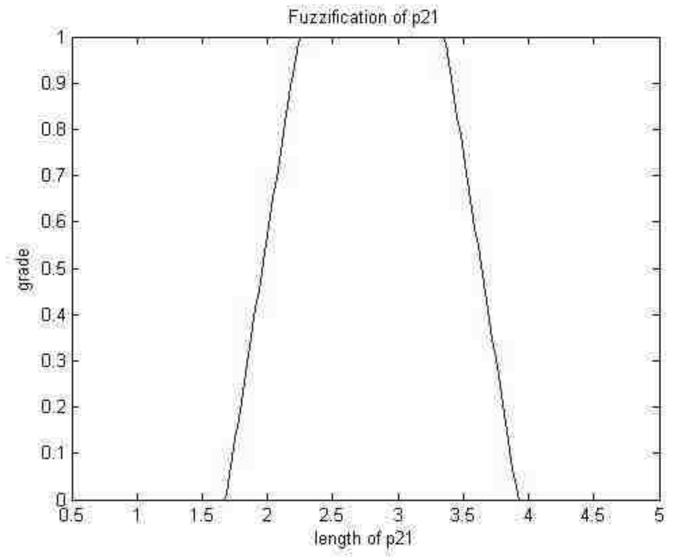

*Figure 6-12. Fuzzy length of P21 with 20% span.*

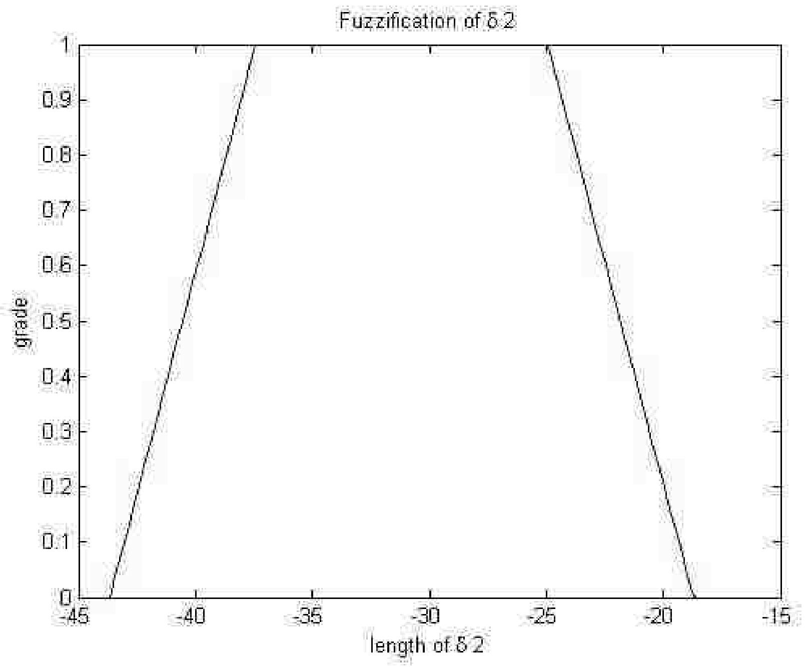

*Figure 6-13. Fuzzy length of δ2 with 20% span.*

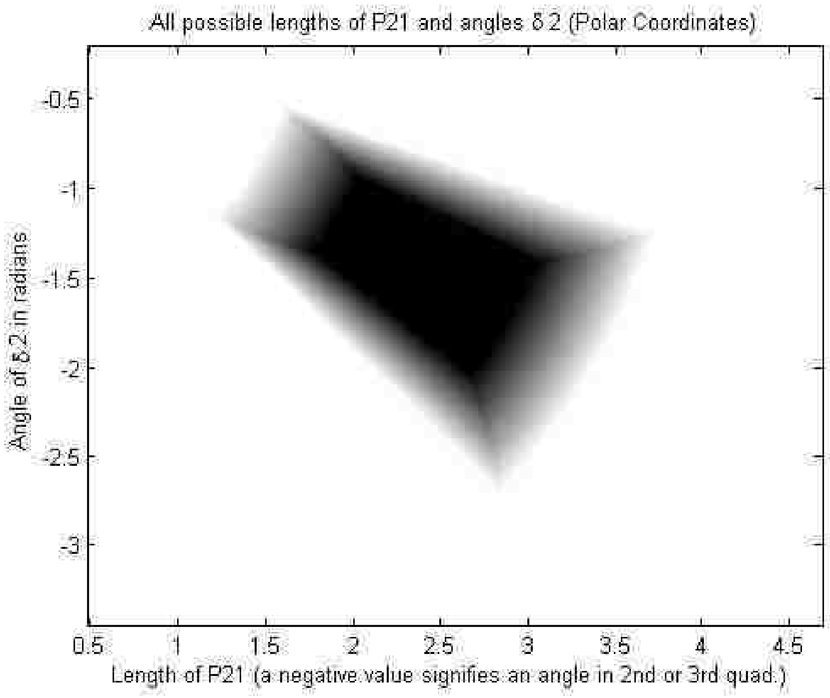

*Figure 6-14. Possible locations of P2.*

Figure 6-14 displays the possible locations of $P_2$, when the lengths of $P_{21}$ and $\delta_2$ are fuzzified as seen above. The following table is the result of the mechanism synthesis with the above fuzzifications. It can be seen that the results from the synthesis produce the most error of the three cases shown, 5%, 10%, and 20%.

*Table 6-3. Results from 20% span fuzzy synthesis*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | 0.0671 | 22.0% |
| Wy | 6.832 | 6.8563 | 0.356% |
| Zx | 1.179 | 1.2566 | 6.582% |
| Zy | 0.940 | 1.0067 | 7.096% |

The following figures, Figures 6-15 and 6-16 display the results from above, the graphs of the fuzzy length of W and the fuzzy length of Z, respectively.

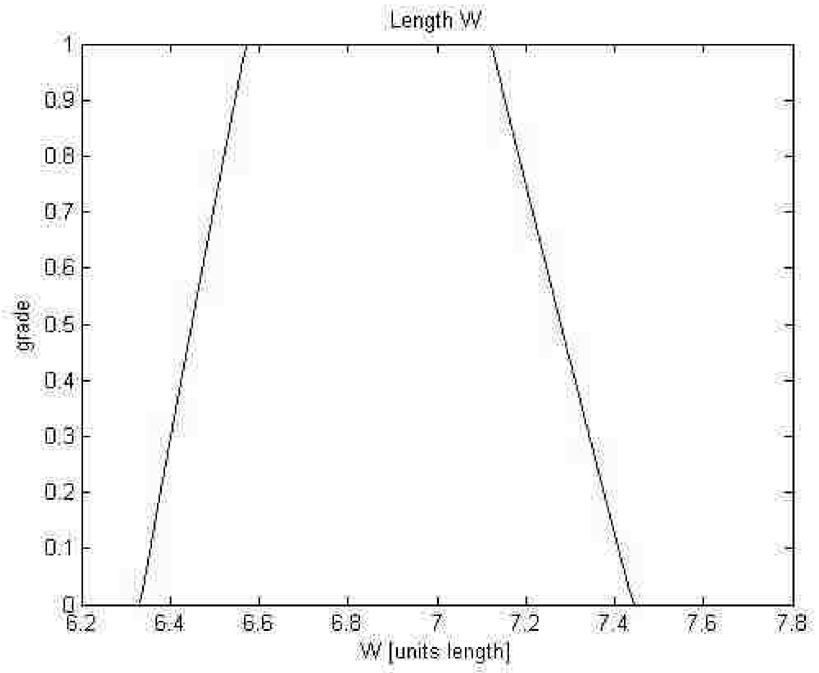

*Figure 6-15. Fuzzy length of W (20% span).*

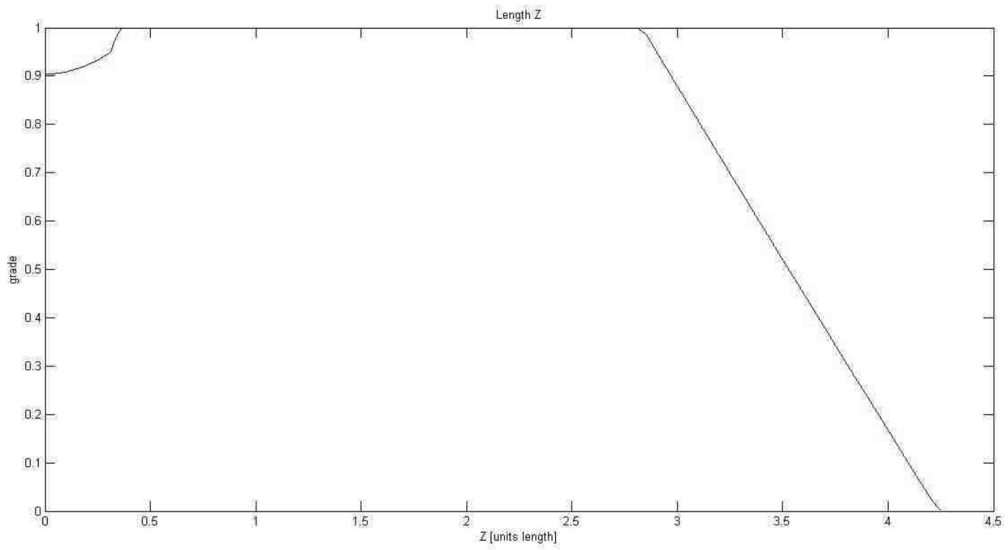

*Figure 6-16. Fuzzy length of Z (20% span).*

## 6.4 Analysis of Variance

It can be seen from the three different analyses above, the larger the span of fuzziness about the crisp value of $P_{21}$ and $\delta_2$, the larger the error in the results become. This can be assumed for all types of fuzzy four-bar linkage syntheses. In the case presented above, a 10% span is reasonable, meaning that any fuzziness that spans within 10% of the crisp value of $P_{21}$ and $\delta_2$ is acceptable to use within a fuzzy synthesis. It is true, however, that the smaller the span, the more accurate the results of the fuzzy synthesis will be. Intuitively, this makes sense because, the smaller the span, the closer the fuzzy numbers are to the actual values of $P_{21}$ and $\delta_2$, creating more accurate results.

The defuzzification method used throughout the entirety of this analysis is the centroid method, also known as the center of gravity method, which is the most common method used to defuzzify a fuzzy set into a scalar (Ross, 2010, 99). Other defuzzification methods may produce different error percentages, which will be discussed below.

# Chapter 7: Difference in Defuzzification Methods

As expected, the different methods of defuzzification produce different defuzzified results. This chapter will demonstrate this theory by comparing the results of the 10% span from above using the centroid method, bisector method, middle of maximum (MOM), smallest of maximum (SOM), and largest of maximum (LOM). These are the methods chosen to compare because they are all easily represented in MATLAB.

## 7.1 Comparison of Defuzzification Methods

The following tables show the results of fuzzy synthesis using the 10% span from above. The results are found using different defuzzification methods, and the error percentage is calculated to allow for a comparison.

*Table 7-1. Results of 10% span using Centroid Method*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | 0.0578 | 5.091% |
| Wy | 6.832 | 6.8379 | 0.0864% |
| Zx | 1.179 | 1.1986 | 1.662% |
| Zy | 0.940 | 0.9565 | 1.755% |

*Table 7-2. Results of 10% span using Bisector Method*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | 0.0586 | 6.545% |
| Wy | 6.832 | 6.9678 | 1.988% |
| Zx | 1.179 | 1.5952 | 35.301% |
| Zy | 0.940 | 1.4103 | 50.032% |

*Table 7-3. Results of 10% span using MOM.*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | 0.0553 | 0.545% |
| Wy | 6.832 | 6.8354 | 0.050% |
| Zx | 1.179 | 1.1863 | 0.619% |
| Zy | 0.940 | 0.9512 | 1.191% |

*Table 7-4. Results of 10% span using SOM.*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | -0.0759 | 238% |
| Wy | 6.832 | 6.7029 | 1.890% |
| Zx | 1.179 | 0.7584 | 35.674% |
| Zy | 0.940 | 0.4920 | 47.660% |

*Table 7-5. Results of 10% span using LOM.*

| Variables | Desired Results | Actual results | Percent Error |
|---|---|---|---|
| Wx | 0.055 | 0.1866 | 239.273% |
| Wy | 6.832 | 6.9678 | 1.988% |
| Zx | 1.179 | 1.6142 | 36.913% |
| Zy | 0.940 | 1.4103 | 50.032% |

### 7.1.1 Defuzzification Method Comparison Analysis

The two defuzzification methods used above that display the least error between the desired results (results from the crisp synthesis) and the actual results (defuzzified results of the fuzzy synthesis) are the centroid method and the middle of maximum (MOM) method. MOM method is most likely the best method to use for this purpose because most of the final results resemble perfect trapezoidal membership functions and the middle of the maximum for these results is the middle all the results which had a grade of 1. The MOM method overall produces the least percentage error for the case of fuzzy mechanism synthesis using a trapezoidal membership, while the centroid method is the most commonly used method.

It is important to note that the centroid method was the method used to defuzzify the fuzzy synthesis results throughout this study.

# Chapter 8: Difference in Fuzzification Methods

As discussed earlier, there are a few different types of membership functions that can be chosen to fuzzify a set of data. The main membership functions that are commonly used are piece-wise linear functions, Gaussian distribution functions, sigmoid curves, and quadratic and cubic polynomial curves. Within this project, piece-wise linear functions are primarily used; such membership functions are triangular and trapezoidal. The main difference between triangular and trapezoidal membership functions is the amount of variables that are completely within the set, meaning their respective grade is one. With a triangular membership function, there is only one value that has a grade of one. The following analysis will display the difference between the results of a fuzzy mechanism synthesis done with trapezoidal membership function inputs and triangular membership function inputs.

To begin this analysis, it will be best to use the tape unit example from chapter 4 in order to allow the reader to more fully understand the importance of this analysis. As was seen previously, there were two different analyses done using the tape unit example, one being for a fuzzy $P_2$, and the other being a fuzzy fixed pivot. The fuzzy $P_2$ analysis was done using a trapezoidal membership function for its input, while the fuzzy fixed pivot input was triangular in nature. Here, the a trapezoidal membership function will be used for the fuzzy fixed pivot example in order to compare the two types of membership functions and how this affect the output.

## 8.1 Fuzzy Fixed Pivot of tape unit

Again, since the synthesis completed in Chapter 5 was using triangular inputs, this section will highlight the synthesis using a trapezoidal membership function to represent the inputs.

The following figures will display the trapezoidal inputs for the fuzzy fixed pivot mechanism synthesis.

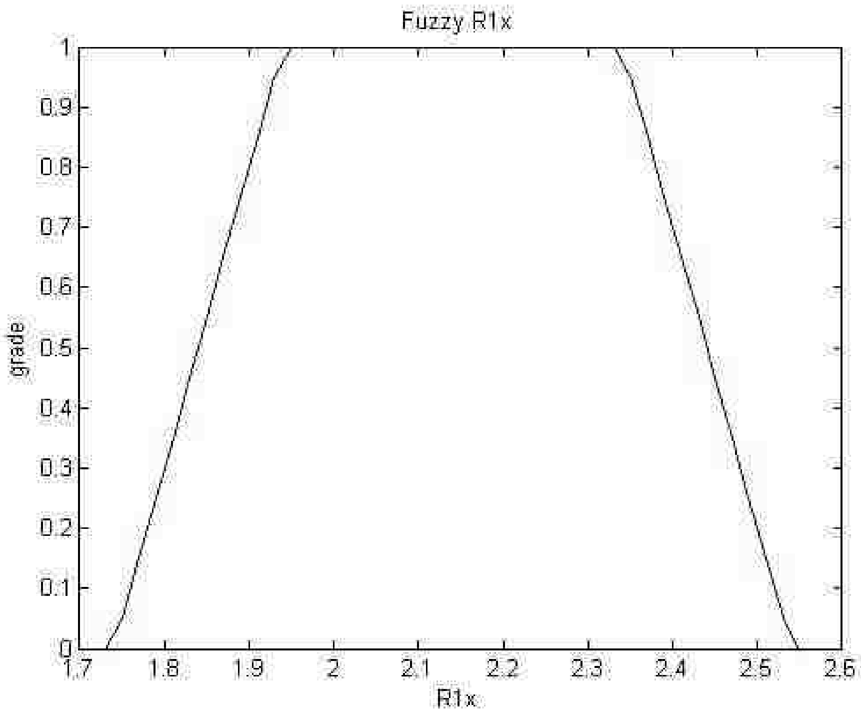

*Figure 8-1. Trapezoidal MF for R1x.*

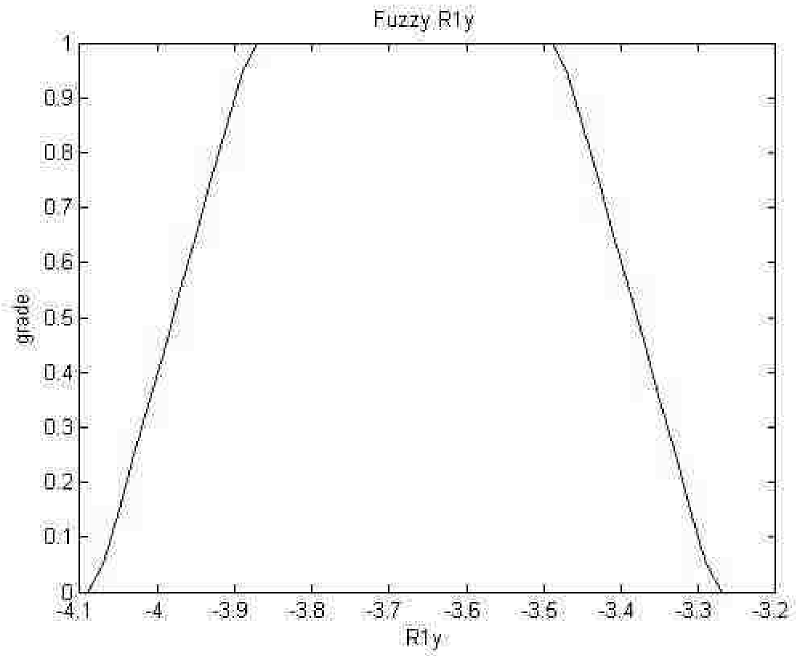

*Figure 8-2. Trapezoidal MF for R1y.*

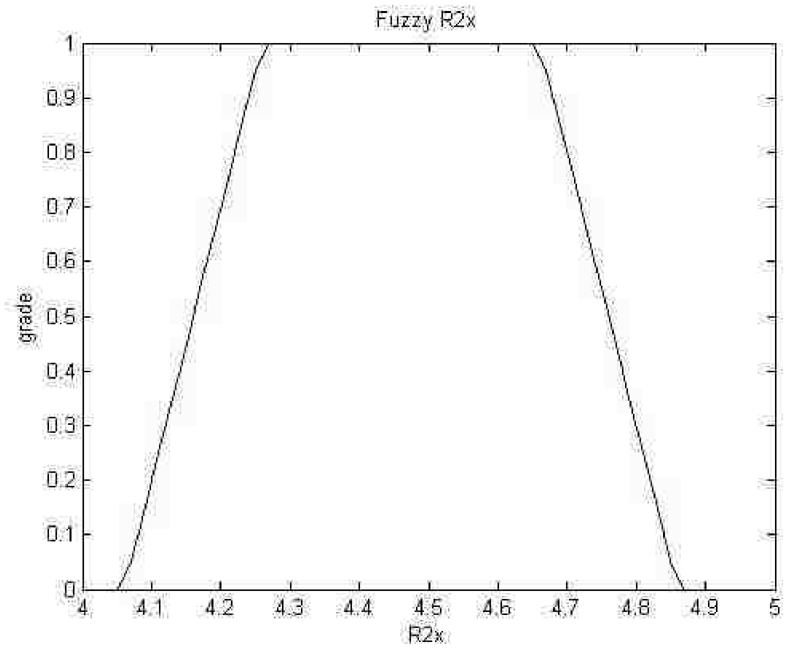

*Figure 8-3. Trapezoidal MF for R2x.*

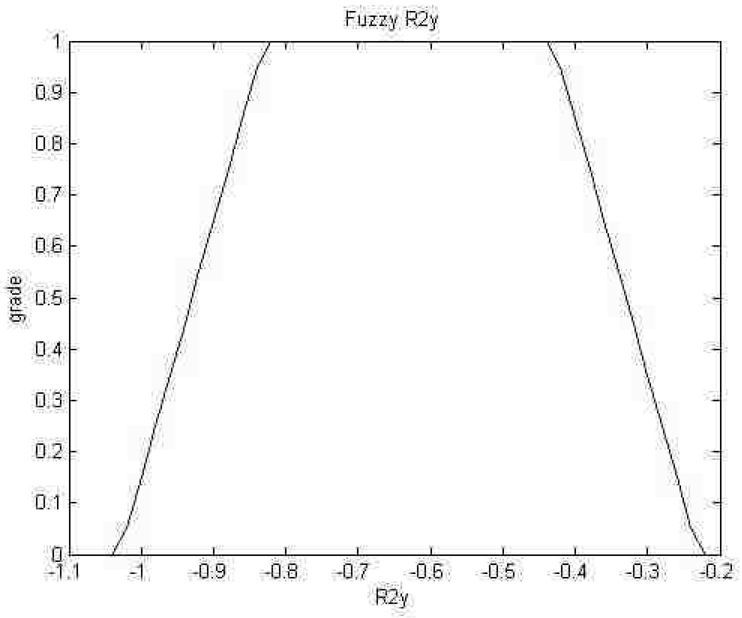

*Figure 8-4. Trapezoidal MF for R2y.*

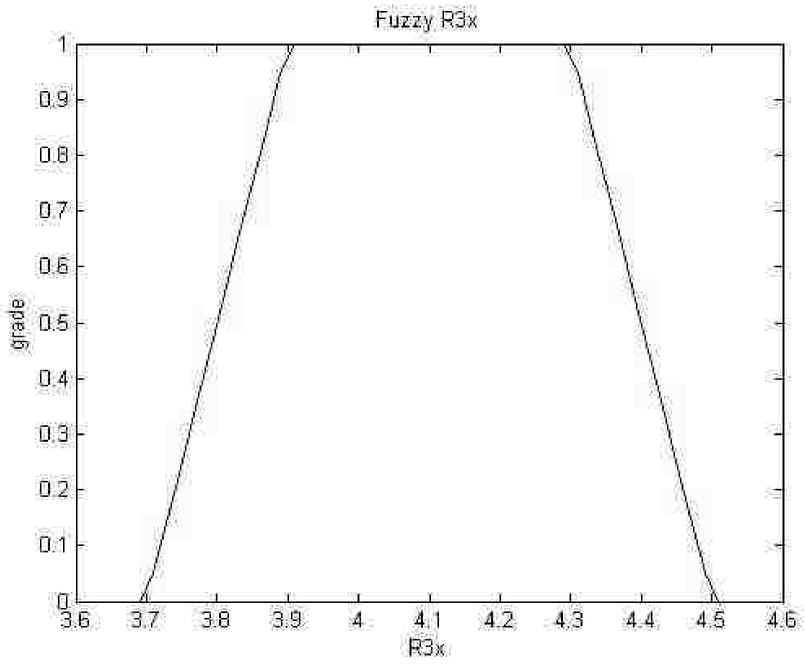

*Figure 8-5. Trapezoidal MF for R3x.*

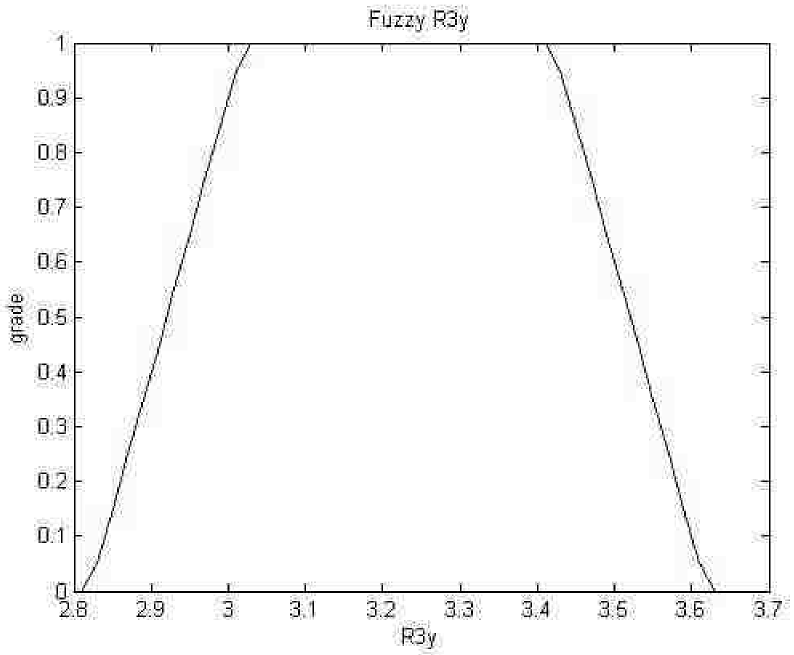

*Figure 8-6. Trapezoidal MF for R3y.*

The following four figures, Figures 8-7 to 8-10, display the results of the trapezoidal fuzzy fixed pivot mechanism synthesis, the fuzzy lengths of linkages W and Z.

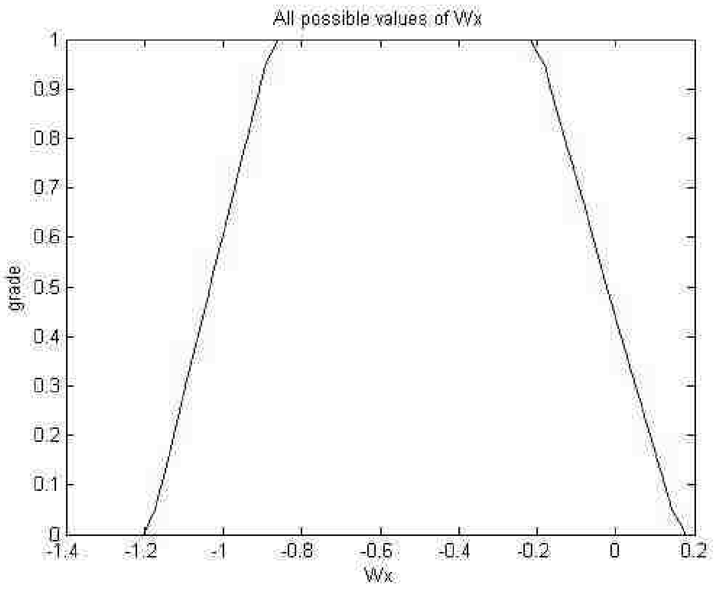

*Figure 8-7. Results for trapezoidal fuzzy Wx.*

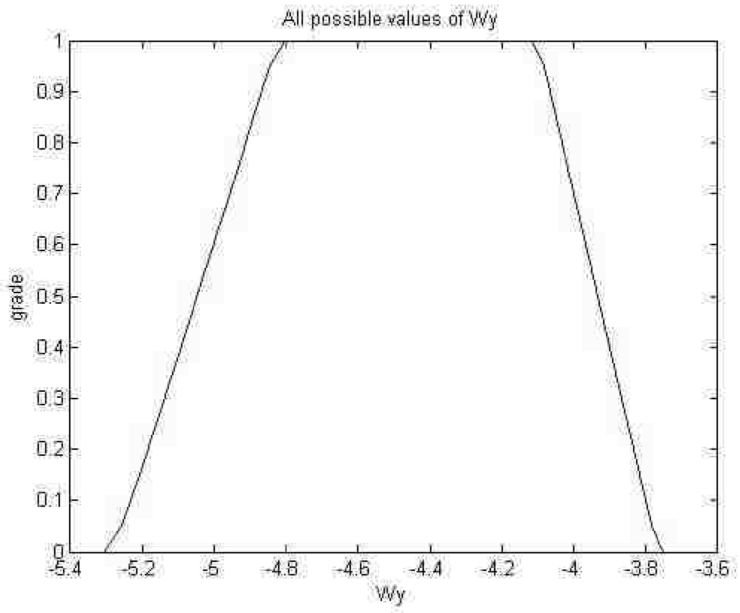

*Figure 8-8. Results for trapezoidal fuzzy Wy.*

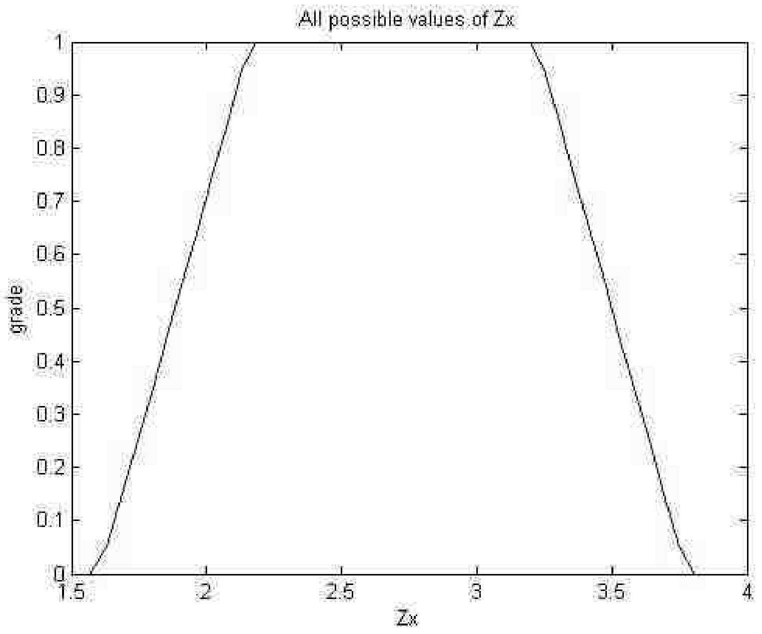

*Figure 8-9. Results for trapezoidal fuzzy Zx.*

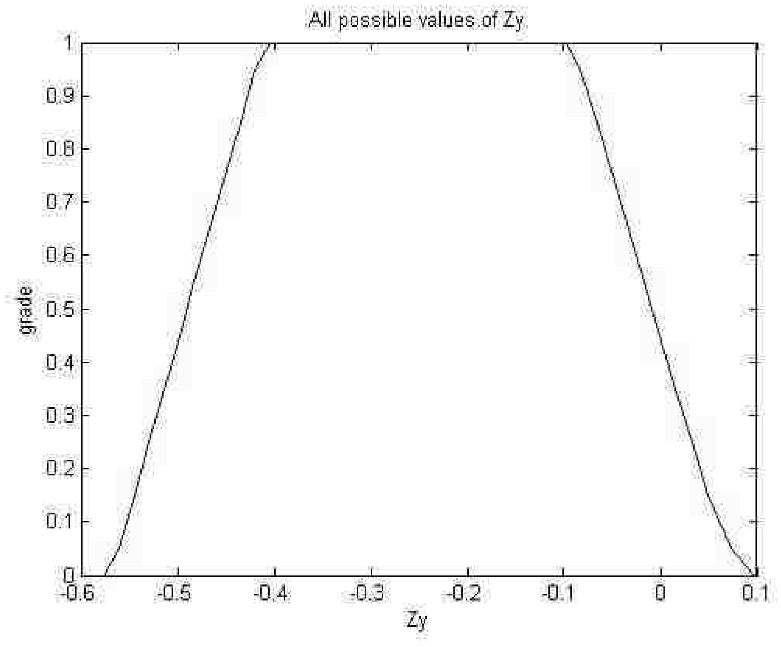

*Figure 8-10. Results for trapezoidal fuzzy Zy.*

Based on the previously specified inputs, the outputs of the trapezoidal fuzzy fixed pivot synthesis are shown below in Table 8-1 alongside the results from the triangular fuzzy fixed pivot synthesis.

*Table 8-1. Results Comparison for Fuzzy Fixed Pivot Synthesis*

| Variable | Trapezoidal MF | Triangular MF | % Difference |
|---|---|---|---|
| Wx | -0.5324 | -0.5467 | 2.616% |
| Wy | -4.4774 | -4.4443 | 0.745% |
| Zx | 2.6911 | 2.6889 | 0.082% |
| Zy | -0.2502 | -0.2525 | 0.911% |

## 8.2 Differences in Membership Functions Conclusion

It can be seen from Table 8-2 that there is some difference between the defuzzified results of the synthesis using the trapezoidal results and the triangular results. This difference could be the result of the larger range of more possible results with the trapezoidal results.  Since the difference between the actual crisp results and the defuzzified triangular results is extremely small, it can be assumed that the difference between the actual crisp answer and the trapezoidal answer is greater than that of the difference between the actual crisp answer and the triangular results. This analysis shows that inputs with triangular membership functions produce more accurate results when compared to the results using trapezoidal membership functions.

# Chapter 9: Conclusion and Potential Future Work

## 9.1 Conclusion

As previously mentioned, using fuzzy logic in mechanical design and synthesis allows the user to incorporate the impreciseness within their inputs into the respective procedures. This concept will increase necessary flexibility in mechanical design, specifically mechanism synthesis. There are many different types of membership functions that can represent many types of data sets, or the user can make their own membership function to map their data to a specific value. Inserting human reasoning into design is important, especially if the user wishes for the output to reflect this type of reasoning.

The different types of fuzzification and defuzzification of a data set does have an effect on the outcome of the synthesis; however, it is up to the user to determine which method for both the fuzzification and defuzzification will best suit the specific needs. In this case, the best type of fuzzification, which is shown within, is the use of the trapezoidal membership function. The trapezoidal membership function assigns more than one value fully to the data set. In this case, it is important to distinguish the entire area in which the second position of the mechanism is represented or the area representing the location of the fixed pivot. The fuzzification of the data is an important process which begins to represent the uncertainty in the set.

After the process of fuzzification occurs and the data is mathematically manipulated, the data can be defuzzified in order to determine the best number in the set. Like the fuzzification methods, it is best to choose a defuzzification method that is best for the given scenario. Here within, the method of defuzzification used was the centroid method. This is because the centroid method accounts for the distortion of the results due to the fuzzy arithmetic the data must undergo. Another appropriate defuzzification method for this project would be the middle of maxima method; however, this method does not account for the distortion the entire data set undergoes, making this not the best method to use for this project. The choice of defuzzification method is dependent upon the user's needs, and it is just as important as the fuzzification methods.

Throughout the scope of this project, the main objective was to represent uncertainty within mechanism synthesis, specifically three-position four-bar mechanism synthesis, by representing the uncertain or imprecise variables as fuzzy sets, then conducting the mathematical procedures necessary. After the result is obtain, which should be fuzzy sets, the results can be defuzzified to obtain the optimal value within the fuzzy sets. This process can guarantee that all possibilities are considered, while only requiring one fuzzy synthesis.

## 9.2 Future Work

This research was conducted using a four-bar linkage; therefore, there is room and need to complete mechanism synthesis using fuzzy variables for other linkages with a different amount of links. Research could be altered to accommodate trusts, five-bar linkages, etc. The basic idea of the procedure conducted here within would remain the same; however, the vector loop equation would differ, changing the entire process.

Another area of study for further analysis would be to increase the amount of positions the linkage moves through, such as a four-position or five-position analysis. With the increase in the amount of positions, the complexity of the problem increases, but the general idea of how to go about solving the problem would be the same. To increase the amount of positions would allow for a more real world analysis of a mechanism design.

Also in terms of mechanism design, fuzzy logic can introduce uncertainty and impreciseness into function generation and path generation synthesis.  The type of synthesis conducted here within was motion generation mechanism synthesis, and there are many different types of mechanism synthesis that can be implemented alongside fuzzy logic.

# Chapter 10: References

Antonsson, E. K., and Otto, K. N. (1995). Imprecision in Engineering Design. *ASME Journal of Mechanical Design*, 117(B), 25-32.

Chen, F., Tzeng, Y., Hsu, M., & Chen, W. (2010). Combining Taguchi Method, Principal Component Analysis and Fuzzy Logic to the Tolerance Design of a Dual-Purpose Six-bar Mechanism. *Transactions of the Canadian Society for Mechanical Design,* 34 (2), 277-293.

Dhingra, A. K., & Rao, S. S. (1991). An Integrated Kinematic-Kinetostatic Approach to Optimal Design of Planar Mechanisms Using Fuzzy Theories. *Journal of Mechanical Design*, 113, 306-311.

Kumar, V. & Schuhmacher, M. (2005). Fuzzy Uncertainty Analysis in System Modeling. *European Symposium on Computer Aided Process Engineering* ,15.

Laribi, M. A., Mlika, A., Romdhane, L., Zeghloul, S. (2004). A combined genetic algorithm-fuzzy logic method (GA-FL) in mechanism synthesis. *Mechanism and Machine Theory,* 39, 717-735.

MathWorks. (2014). *Fuzzy Logic Toolbox.* Natick: MathWorks.

*Multi-valued Logic*. Princeton University. Retrieved from https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Multi-valued_logic.html

Nariman-Zadeh, N., Felezi, M., Jamali, A., & Ganji, M. (2009). Pareto optimal synthesis of four-bar mechanisms for path generation. *Mechanism and Machine Theory*, 44, 180-191.

Norton, R. L. (1999). *An Introduction to the Synthesis and Analysis of Mechanism and Machines: Second Edition*. Worcester: WCB/McGraw-Hill.

Ross, T. J. (2010). *Fuzzy Logic with Engineering Applications: Third Edition*. West Sussex, UK: Wiley.

Sandor, G. N. & Erdman, A. G. (1984). *Advanced Mechanism Design: Analysis and Synthesis Volume 2*. Englewood Cliffs: Prentice-Hall, Inc.

Urroz, G. E. (2004). “Solution of Non-Linear Equations”. Lecture Notes. Logan, UT: Utah State University.

# Chapter 11: Appendices

The following sections outline the basic knowledge of mechanism synthesis.

## Appendix A: Example of Crisp Three-Position Mechanism Synthesis

As noted before:

$$A = \cos(\beta_2) - 1; \qquad B = \sin(\beta_2)\,; \qquad C = \cos(\alpha_2) - 1;$$

$$D = \sin(\alpha_2)\,; \qquad E = p_{21}\cos(\delta_2)\,; \qquad F = \cos(\beta_3) - 1;$$

$$G = \sin(\beta_3)\,; \qquad H = \cos(\alpha_3) - 1; \qquad K = \sin(\alpha_3)\,;$$

$$L = p_{31}\cos(\delta_3)\,; \qquad M = p_{21}\sin(\delta_2)\,; \qquad N = p_{31}\sin(\delta_3)$$

The system can then be put into the simplified system of equations as:

$$AW_x - BW_y + CZ_x - DZ_y = E$$
$$FW_x - GW_y + HZ_x - KZ_y = L$$
$$BW_x + AW_y + DZ_x + CZ_y = M$$
$$GW_x + FW_y + KZ_x + HZ_y = N$$

Putting this system into matrix form:

$$\begin{bmatrix} A & -B & C & -D \\ F & -G & H & -K \\ B & A & D & C \\ G & F & K & H \end{bmatrix} \cdot \begin{bmatrix} W_x \\ W_y \\ Z_x \\ Z_y \end{bmatrix} = \begin{bmatrix} E \\ L \\ M \\ N \end{bmatrix}$$

Using the following values:

| Variables | Magnitude of Value |
|---|---|
| $P_{21}$ | 2.798 |
| $\delta_2$ | -31.19 |
| $P_{31}$ | 3.919 |
| $\delta_3$ | -16.34 |
| $\alpha_2$ | -45.0 |
| $\alpha_3$ | 9.3 |
| $\beta_2$ | 342.3 |
| $\beta_3$ | 324.8 |

These values can now be substituted into the above matrix in order to solve for the values of W and Z.

$$A = -0.0473; \quad B = -0.3040; \quad C = -0.2929; \quad D = -0.7071;$$

$$E = 2.3936; \quad F = -0.1829; \quad G = -0.5764; \quad H = -0.0131;$$

$$K = 0.1616; \quad L = 3.7607; \quad M = -1.4490; \quad N = -1.1026$$

In matrix form,

$$\begin{bmatrix} -0.0473 & 0.3040 & -0.2929 & 0.7071 \\ -0.1829 & 0.5764 & -0.0131 & -0.1616 \\ -0.3040 & -0.0473 & -0.7071 & -0.2929 \\ -0.5764 & -0.1829 & 0.1616 & -0.0131 \end{bmatrix} \cdot \begin{bmatrix} W_x \\ W_y \\ Z_x \\ Z_y \end{bmatrix} = \begin{bmatrix} 2.3936 \\ 3.7607 \\ -1.4490 \\ -1.1026 \end{bmatrix}$$

$$\begin{bmatrix} W_x \\ W_y \\ Z_x \\ Z_y \end{bmatrix} = \begin{bmatrix} 0.055 \\ 6.832 \\ 1.179 \\ 0.940 \end{bmatrix}$$

## Appendix B: Two-Position Fuzzy Mechanism Synthesis MATLAB Code

```
function [] = TwoDimFuzzySyn_v9(theta, phi, beta2,alpha2,
p21, delta2)
% -----  THIS CODE ASSUMES THE USER IS SOLVING FOR VALUES
W AND Z (THE
% -----  LENGTHS OF THE ROTATING LINKAGES)AS OPPOSED TO
THE LENGTH AND ANGLE OF
% -----  ONE OF THE ROTATING LINKAGES---


% Assigning values to variables (Norton page 192)
% These variables are directly from Norton's book.  They
significantly
% simplify the calculations of w and z.
A=cosd(theta)*(cosd(beta2)-1)-sind(theta)*sind(beta2);
B=cosd(phi)*(cosd(alpha2)-1)-sind(phi)*sind(alpha2);
C=sind(theta)*(cosd(beta2)-1)+cosd(theta)*sind(beta2);
D=sind(phi)*(cosd(alpha2)-1)+cosd(phi)*sind(alpha2);


% Defining the trap. fuzzy membership function
% Sent to fuzzyfun function to have the trapezoidal
membership function
% calculated for the 4 number vectors specified above.
[p21new, delta2new]=fuzzyfun(p21,delta2);


% calculating the sine and cosine of the angle delta2
% The membership function will remain the same (second
column) because of
% the extention principle
delta2cos=[cosd(delta2new(1,:)); delta2new(2,:)];
delta2sin=[sind(delta2new(1,:)); delta2new(2,:)];


% multiplication of two numbers, both of which have an
assigned fuzzy
% membership function, sent to fuzzy mult function,
variables defined by
% Norton
E=(fuzzymultvertex(p21new, delta2cos));
F=(fuzzymultvertex(p21new, delta2sin));
```

```
% Matrix multiplication and re-assigning fuzzy membership
% In order to simplify the calculation of the membership
function, the
% matrix multiplication was done "by hand"
det=A*D-B*C;
W1=[D*E(:,1),E(:,2)];
W2=[-B*F(:,1),F(:,2)];%separation of variables with fuzzy
numbers
Z1=[A*F(:,1),F(:,2)];
Z2=[-C*E(:,1),E(:,2)];
W=sortrows(fuzzyaddvertex(W1',W2'));  % adding the two
numbers with attached membership functions in fuzzy add
function

W=[W(:,1)/det,W(:,2)];  % dividing by the determinate
left over from the inverse of W

Z=sortrows(fuzzyaddvertex(Z1', Z2'));
Z=[Z(:,1)/det, Z(:,2)];

%Defuzzifying using built in Matlab function defuzz
defuzzW=defuzz(W(:,1),W(:,2),'centroid')
defuzzZ=defuzz(Z(:,1),Z(:,2),'centroid')



%plotting the possible lengths and angles of p21 and
delta 2
for ii=1:length(p21new) %for loops that iterate through
combinations of p21 and delta2
    for jj=1:length(delta2new)

        X(jj,ii)=p21new(1,ii)*cosd(delta2new(1,jj));
%sets X and Y as polar coordinates
        Y(jj,ii)=p21new(1,ii)*sind(delta2new(1,jj));
        C(jj,ii)=1-minfun;  %ensures darker to lighter
color when membership function goes from 1 to 0
    end
end
```

```
figure(1); pcolor(X,Y,C)    %pseudocolor (contour) plot -
- the darker the color, the greater the grade
shading interp; colormap(gray); %removes gridlines and
uses grayscale colors
title('All possible lengths of P21 and angles \delta 2
(Polar Coordinates)');
xlabel('Length of P21 (a negative value signifies an
angle in 2nd or 3rd quad.) ');
ylabel('Angle of \delta 2 in radians')

% Displaying the length of linkages W and Z against their
fuzziness
figure(2);
plot(W(:,1),W(:,2)); title('Length W');
xlabel('W [units length]'); ylabel('grade');
figure(3);
plot(Z(:,1),Z(:,2)); title('Length Z');
xlabel('Z [units length]'); ylabel('grade');

end
```

The MATLAB code shown above will complete a mechanism synthesis for two-positions.  The inputs for this function are the variables for the synthesis, (theta, phi, beta2, alpha2, p21, delta2).  Here, since this program was designed for a fuzzy $P_2$, p21 and delta2 must be represented as fuzzy numbers, as described within the content of this thesis.  The results will be fuzzy because the inputs are fuzzy.

## Appendix C: Three-Position Fuzzy Mechanism Synthesis with a Fuzzy $P_2$

```
function [] = ThreeDimFuzzySyn_v1(beta2, beta3, alpha2,
alpha3, p31, delta3, p21, delta2)
% THREE POSITION, FUZZY P2
% -----  THIS CODE ASSUMES THE USER IS SOLVING FOR VALUES
W AND Z (THE
% -----  LENGTHS OF THE ROTATING LINKAGES)AS OPPOSED TO
THE LENGTH AND ANGLE OF
% -----  ONE OF THE ROTATING LINKAGES---

% Assigning values to variables (Norton page 204)
% These variables are directly from Norton's book.  They
significantly
% simplify the calculations of w and z.
A=cosd(beta2)-1;        G=sind(beta3);
B=sind(beta2);          H=cosd(alpha3)-1;
C=cosd(alpha2)-1;       K=sind(alpha3);
D=sind(alpha2);         L=p31*cosd(delta3);
F=cosd(beta3)-1;        N=p31*sind(delta3);

% Defining the trap. fuzzy membership function
% Sent to fuzzyfun function to have the trapezoidal
membership function
% calculated for the 4 number vectors specified above.
[p21new, delta2new]=fuzzyfun(p21,delta2);


% calculating the sine and cosine of the angle delta2
% The membership function will remain the same (second
column) because of
% the extention principle
delta2cos=[cosd(delta2new(1,:)); delta2new(2,:)];
delta2sin=[sind(delta2new(1,:)); delta2new(2,:)];

% multiplication of two numbers, both of which have an
assigned fuzzy
% membership function, sent to fuzzy mult function,
variables defined by
% Norton
E=(fuzzymultvertex(p21new, delta2cos));
M=(fuzzymultvertex(p21new, delta2sin));
```

```
% Matrix multiplication and re-assigning fuzzy membership
% In order to simplify the calculation of the membership
function, the
% matrix multiplication was done "by hand"
P=[A -B C -D; F -G H -K; B A D C; G F K H];
Y=inv(P);

Y1=[Y(1,1).*E(:,1),E(:,2)];   Y2=[Y(1,3).*M(:,1),M(:,2)];
Ywx=(fuzzyaddvertex(Y1',Y2'));
Wx=[Ywx(:,1)+Y(1,2)*L+Y(1,4)*N, Ywx(:,2)];
Wxdefuzz=defuzz(Wx(:,1), Wx(:,2),'centroid')


Y3=[Y(2,1)*E(:,1),E(:,2)];   Y4=([Y(2,3)*M(:,1),M(:,2)]);
Ywy=sortrows(fuzzyaddvertex(Y3',Y4'));
Wy=([Ywy(:,1)+Y(2,2)*L+Y(2,4)*N, Ywy(:,2)]);
Wydefuzz=defuzz(Wy(:,1), Wy(:,2), 'centroid')


Y5=[Y(3,1)*E(:,1),E(:,2)];   Y6=[Y(3,3)*M(:,1),M(:,2)];
Yzx=(fuzzyaddvertex(Y5',Y6'));
Zx=[Yzx(:,1)+Y(3,2)*L+Y(3,4)*N, Yzx(:,2)];
Zxdefuzz=defuzz(Zx(:,1), Zx(:,2), 'centroid')


Y7=[Y(4,1)*E(:,1),E(:,2)];   Y8=[Y(4,3)*M(:,1),M(:,2)];
Yzy=(fuzzyaddvertex(Y7',Y8'));
Zy=[Yzy(:,1)+Y(4,2)*L+Y(4,4)*N, Yzy(:,2)];
Zydefuzz=defuzz(Zy(:,1), Zy(:,2), 'centroid')


%Find full lengths of W and Z linkages

Wy2=[Wy(:,1).^2, Wy(:,2)];
Wx2=[Wx(:,1).^2, Wx(:,2)];
Wfuzz1=sortrows(fuzzyaddvertex(Wy2', Wx2'));
Wfuzz=sortrows([(Wfuzz1(:,1)).^(1/2), Wfuzz1(:,2)]);
Zy2=[Zy(:,1).^2, Zy(:,2)];
Zx2=[Zx(:,1).^2, Zx(:,2)];
Zfuzz1=(fuzzyaddvertex(Zy2', Zx2'));
Zfuzz=sortrows([(Zfuzz1(:,1)).^(1/2), Zfuzz1(:,2)]);
```

```
%plotting the possible lengths and angles of p21 and
delta 2
for ii=1:length(p21new) %for loops that iterate through
combinations of p21 and delta2
    for jj=1:length(delta2new)
        minfun=min(p21new(2,ii),delta2new(2,jj));
%defines the minimum of the two fuzzinesses as the
respective fuzziness of the point in question
        X(jj,ii)=p21new(1,ii)*cosd(delta2new(1,jj));
%sets X and Y as polar coordinates
        Y(jj,ii)=p21new(1,ii)*sind(delta2new(1,jj));
        C(jj,ii)=1-minfun;  %ensures darker to lighter
color when membership function goes from 1 to 0
    end
end

figure(3); pcolor(X,Y,C)    %pseudocolor (contour) plot -
- the darker the color, the greater the grade
shading interp; colormap(gray); %removes gridlines and
uses grayscale colors
hold on; figure (3); scatter(0,0, 'filled', 'b')
scatter(p31*cosd(delta3), p31*sind(delta3), 'filled')
axis([0 5 0 10]);
title('Three-Position Fuzzy Synthesis');
xlabel('X-length');
ylabel('Y-length')
%gtext('Position 1'); gtext('Position 2');
gtext('Position 3');
hold off;

% Displaying the length of linkages W and Z against their
fuzziness
figure(4);
plot(Wfuzz(:,1),Wfuzz(:,2)); title('Length U');
xlabel('U [units length]'); ylabel('grade');

figure(5);
plot(Zfuzz(:,1),Zfuzz(:,2)); title('Length S');
xlabel('S [units length]'); ylabel('grade');

end
```

Like the MATLAB code for the two-position synthesis, this code will require the inputs of p21 and delta2 to be fuzzy inputs.  Because of this, the results will be fuzzy outputs.  The other inputs for this function are not to be fuzzy numbers, but they are required to complete the synthesis.

## Appendix D: Three-position Fuzzy Synthesis for Fuzzy Fixed Pivot MATLAB Code

Unlike the previous two MATLAB codes, seen in Appendices B and C, this code completes a fuzzy mechanism synthesis for a fuzzy fixed pivot. This code is written as a script, so there are no inputs necessary; however, to utilize this code, the user will have to be sure the variables specified in the code itself as correct for their intended problem.

```
% fuzzify pivot point - O2x and O2y
% assume pivot point is origin (0,0)
% Tape Unit Ex - crisp: beta2= -48.2609 and beta3= -89.7104
clear all; close all; clc;
format compact;
digits(3)

delx = 0.2;  dely = 0.2;    %wideness of fixed pivot
R1x=2.14;   R1y=-3.68;
R2x=4.46;   R2y=-0.63;
R3x=4.10;   R3y=3.22;
a2=50.7;    a3=91.9;    %alphas

%fuzzifying the length of the vector from the pivot point
to the positions
[R1x, R1y]=fuzzyfun([R1x-delx R1x R1x R1x+delx], [R1y-dely
R1y R1y R1y+dely]);
R1x=round(R1x'*1000)/1000;    R1y=round(R1y'*1000)/1000;
%this a makes sure there is no floating numbers.

[R2x, R2y]=fuzzyfun([R2x-delx R2x R2x R2x+delx], [R2y-dely
R2y R2y R2y+dely]);
R2x=round(R2x'*1000)/1000;    R2y=round(R2y'*1000)/1000;
[R3x, R3y]=fuzzyfun([R3x-delx R3x R3x R3x+delx], [R3y-dely
R3y R3y R3y+dely]);
R3x=round(R3x'*1000)/1000;  R3y=round(R3y'*1000)/1000;

M=1; %Matrix of beta2 and beta3 index
XX=[0, 0.05:0.1:0.95, 1.0]; %range of fuzziness
for pp=1:length (XX);
    XXX=round(XX(pp)*1000)/1000;
    %finds the location within the fuzzy vector of the
specified fuzziness
    [r1,c1]=find(R1x(:,2)==XXX);
    [r2,c2]=find(R1y(:,2)==XXX);
    [r3,c3]=find(R2x(:,2)==XXX);
    [r4,c4]=find(R2y(:,2)==XXX);
    [r5,c5]=find(R3x(:,2)==XXX);
    [r6,c6]=find(R3y(:,2)==XXX);


    for ii=1:2
        for jj=1:2
            for kk=1:2
                  %account for the single value of grade 1
                 if XXX==1 && ii==2
                     break; %doesn't allow search for second
                     value
                 end
                 if XXX==1 && jj==2
                     break; %doesn't allow search for second
                value
                 end
                 if XXX==1 && kk==2
                     break; %doesn't allow search for second
                     value
                 end

                %finds crisp R values of specified fuzziness
                 R1xnew=R1x(r1(ii)); R1ynew=R1y(r2(ii));
                 R2xnew=R2x(r3(jj)); R2ynew=R2y(r4(jj));
                 R3xnew=R3x(r5(kk)); R3ynew=R3y(r6(kk));



                %account for the multiple zeros in fuzzy
                vectors (must
                %check that there are two zeros at each end
                of vector.)
                 if XXX==0 && ii==2
                     R1xnew=R1x(r1(3)); R1ynew=R1y(r2(3));
                 end
```

```
             if XXX==0 && jj==2
                R2xnew=R2x(r3(3)); R2ynew=R2y(r4(3));
            end
            if XXX==0 && kk==2
                R3xnew=R3x(r5(3)); R3ynew=R3y(r6(3));
            end


            %calculates the D values from Erdmand and
           Sandor page 123
            D1x=R3xnew*cosd(a2)+R3ynew*sind(a2)-
           R2xnew*cosd(a3)-R2ynew*sind(a3);
            D1y=R3ynew*cosd(a2)+R3xnew*sind(a2)-
           R2ynew*cosd(a3)-R2xnew*sind(a3);
            D2x=R1xnew*cosd(a3)+R1ynew*sind(a3)-R3xnew;
            D2y=R1ynew*cosd(a3)+R1xnew*sind(a3)-R3ynew;
            D3x=R2xnew-R1xnew*cosd(a2)-R1ynew*sind(a2);
            D3y=R2ynew-R1ynew*cosd(a2)-R1xnew*sind(a2);


            %Jacobian
            J11=@(x) -D2x*sind(x)-D2y*cosd(x);
            J12=@(x) -D3x*sind(x)-D3y*cosd(x);
            J21=@(x) D2x*cosd(x)-D2y*sind(x);
            J22=@(x) D3x*cosd(x)-D3y*sind(x);


           %calculate f1 and f2 (real and imaginary
           functions of
            %vector loop analysis)
            f1=@(x1,x2) D1x+D2x*cosd(x1)-
      D2y*sind(x1)+D3x*cosd(x2)-D3y*sind(x2);
            f2=@(x1,x2)
      D1y+D2x*sind(x1)+D2y*cosd(x1)+D3x*sind(x2)+D3y*co
      sd(x2);
            func=@(x1,x2) [f1;f2]; %func. concatination
```

```
        %-- Newton Raphson Method--%
 % NR method is used to calculate each beta and then length
               % of W and Z for the

         x0=[-45;-85];    %initial guess of beta2 and beta3
         N = 1000; % define max. number of iterations
         epsilon = 1e-5; % define tolerance
         maxval = 10000.0; % define value for divergence
                xx = x0; % load initial guess

                while N>0
                    %evaluate Jacobian
                    J1= J11(xx(1));
                    J2= J12(xx(2));
                    J3= J21(xx(1));
                    J4= J22(xx(2));
                    JJ=[J1 J2; J3 J4];

                    if abs(det(JJ))<epsilon      %singular?
error('newtonm - Jacobian is singular - try new x0');
                        N=0; %ends function
                    end;

                    %evaluates function
                    func=[f1(xx(1), xx(2));
                    f2(xx(1),xx(2))];
                    xn = xx - inv(JJ)*func;  %NR method to
find the best value of beta 2 and 3
                    if abs(func)<epsilon
                        x=xn;
                        iter = 1000-N;  %number of iter
                        N=0;            %ends function
                    end;

                    if abs(func)>maxval     %stops if too
               many iterations have passed
                        iter = 1000-N;  %number of iter
                       disp(['iterations =
                    ',num2str(iter)]);
                        error('Solution diverges');
                        N=0;        %ends function
                    end;

                    N = N - 1;      %iterations counter
                    xx = xn;        %sets next guess value
                end; %while loop
```

```
%calculate beta2 and beta3 and assigns them to new matrix
                %with corresponding grade
                allbeta2(M,2)=xn(1);
                allbeta2(M,1)=XXX;
                allbeta3(M,2)=xn(2);
                allbeta3(M,1)=XXX;
                %find the lengths and W and Z for the value
of beta
                B=[cosd(xn(1)), sind(xn(1)), cosd(a2),
sind(a2);...
                    sind(xn(1)), cosd(xn(1)), sind(a2),
cosd(a2);...
                    cosd(xn(2)), sind(xn(2)), cosd(a3),
sind(a3);...
                    sind(xn(2)), cosd(xn(2)), sind(a3),
cosd(a3)];

                WandZ(:,M)=inv(B)*[R2xnew; R2ynew; R3xnew;
R3ynew];
                %assign values of W and Z to matrix with
corresponding
                %grade
                Wx(M,2)=WandZ(1,M)';  Wx(M,1)=XXX;
                Wy(M,2)=WandZ(2,M)';  Wy(M,1)=XXX;
                Zx(M,2)=WandZ(3,M)';  Zx(M,1)=XXX;
                Zy(M,2)=WandZ(4,M)';  Zy(M,1)=XXX;
                M=M+1;
            end

        end
    end
end

%Stripped down version of Vertex Method
%finding min and max of Wx
Wxsort=sortrows(Wx);
kk=1;
for ii=2:length(Wxsort)
    if  Wxsort(ii,1)>Wxsort(ii-1,1)
        Wxnew(kk,:)=Wxsort(ii-1,:);
        kk=kk+1;
        Wxnew(kk,:)=Wxsort(ii,:);
        kk=kk+1;
    end

end
```

```
%plotting Wx
Wxnew(kk,:)=Wxsort(end,:);
Wxnew(kk+1,:)=Wxsort(1, :);
Wx=sortrows([Wxnew(:,2),Wxnew(:,1)]);
figure(1); plot(Wx(:,1), Wx(:,2));
title('All possible values of Ux');
xlabel('Ux'); ylabel('grade')

%finding min and max of Wy
Wysort=sortrows(Wy);
kk=1;
for ii=2:length(Wysort)
    if  Wysort(ii,1)>Wysort(ii-1,1)
        Wynew(kk,:)=Wysort(ii-1,:);
        kk=kk+1;
        Wynew(kk,:)=Wysort(ii,:);
        kk=kk+1;
    end

end

%plotting Wy
Wynew(kk,:)=Wysort(end,:);
Wynew(kk+1,:)=Wysort(1, :);
Wy=sortrows([Wynew(:,2),Wynew(:,1)]);
figure(2); plot(Wy(:,1), Wy(:,2));
title('All possible values of Uy');
xlabel('Uy'); ylabel('grade')


%finding min and max of Zx
Zxsort=sortrows(Zx);
kk=1;
for ii=2:length(Zxsort)
    if  Zxsort(ii,1)>Zxsort(ii-1,1)
        Zxnew(kk,:)=Zxsort(ii-1,:);
        kk=kk+1;
        Zxnew(kk,:)=Zxsort(ii,:);
        kk=kk+1;
    end

end
```

```
%plotting Zx
Zxnew(kk,:)=Zxsort(end,:);
Zxnew(kk+1,:)=Zxsort(1, :);
Zx=sortrows([Zxnew(:,2),Zxnew(:,1)]);
figure(3); plot(Zx(:,1), Zx(:,2));
title('All possible values of Sx');
xlabel('Sx'); ylabel('grade')

%finding min and max of Zy
Zysort=sortrows(Zy);
kk=1;
for ii=2:length(Zysort)
    if  Zysort(ii,1)>Zysort(ii-1,1)
        Zynew(kk,:)=Zysort(ii-1,:);
        kk=kk+1;
        Zynew(kk,:)=Zysort(ii,:);
        kk=kk+1;
    end

end

%plotting Zy
Zynew(kk,:)=Zysort(end,:);
Zynew(kk+1,:)=Zysort(1, :);
Zy=sortrows([Zynew(:,2),Zynew(:,1)]);
figure(4); plot(Zy(:,1), Zy(:,2));
title('All possible values of Sy');
xlabel('Sy'); ylabel('grade')


%defuzzified answers using centroid method and built-in
matlab function
%"defuzz"
defuzzWx=defuzz(Wx(:,1), Wx(:,2), 'centroid')
defuzzWy=defuzz(Wy(:,1), Wy(:,2), 'centroid')
defuzzZx=defuzz(Zx(:,1), Zx(:,2), 'centroid')
defuzzZy=defuzz(Zy(:,1), Zy(:,2), 'centroid')
```

I would like to credit By Gilberto E. Urroz for the basic structure of the Newton Raphson Method for this code. This portion of the code was extracted from his lecture notes, “Solution of Non-Linear Equations”, September 2004.

## Appendix E: Additional Custom MATLAB Codes

The following MATLAB code was written to transform a set of numbers to a membership function, more specifically a trapezoidal membership function. The input of this function is a vector of four numbers, [a b c d], where a and d are the outer limits, and b and c are the inner limits, where the grade is 1.0 between b and c.

```
function [p21new, delta2new]= fuzzyfun(p21,delta2)
%THIS FUNCTION CREATES THE TRAPEZOIDAL MEMBERSHIP
FUNCTION FROM THE GIVEN
%VECTOR SPECIFIED FOR P21 AND DELTA2

        n=400;

        p21step=(p21(4)-p21(1))/n; %divides the length of
the vector into n divisions
        x1=p21(1)-0.01:p21step:p21(4)+0.01;    %creates a
vector to span the entire length of specified p21
        p21mf=trapmf(x1, p21);      %uses built-in matlab
function to the membership function for p21 (fuzziness of
p21)

        %Repeat of p21 steps but for delta 2
        delstep=(delta2(4)-delta2(1))/n;
        x2=delta2(1)-0.01:delstep:delta2(4)+0.01;
        delta2mf=trapmf(x2,delta2);

        figure(1); plot(x1, p21mf);
        title('Fuzzification of O2x');
        xlabel('O2x location'); ylabel('grade');
        figure(2); plot(x2, delta2mf);
        title('Fuzzification of O2y');
        xlabel('O2y location'); ylabel('grade');


        p21new=[x1; p21mf];    delta2new=[x2; delta2mf];
%creates new matrices that attaches the variables and
their fuzziness

 end
```

The next MATLAB code that is displayed is the function that computes the addition of two fuzzy sets. It is important to note that addition between fuzzy sets in similar to but not the same as crisp addition. To add fuzzy sets, the vertex method was used.

```
function [P]=fuzzyaddvertex(A,B)
% -- This function multiplies the two fuzzy numbers using
the vertex
% method in combination with interpolation to ensure both
functions being
% multipled have the same membership functions


%===============INTERPOLATION===========================
Anew=zeros(1,2);
k=0;
X=0:0.001:1;
for jj=1:length(X)
    for ii=2:length(A)

          if (A(2,ii)>X(jj) && A(2,ii-1)<X(jj)) ||
     (A(2,ii)<X(jj) && A(2,ii-1)>X(jj))
            k=k+1;
            Anew(1)=(((X(jj)-A(2,ii-1))*(A(1,ii)-A(1,ii-
1)))/(A(2,ii)-A(2,ii-1)))+A(1,ii-1);   %interpolates to
find the value related to mu=[X]

            Anew(2)=X(jj); %relates the fuzziness with
the previous step

            M(k,:)=Anew; %sends to a new matrix outside
of for loop to ease calculation of vertex function

          elseif A(2,ii)==X(jj)
            k=k+1;
            Anew=[A(1,ii),A(2,ii)];
            M(k,:)=Anew;
        end
    end

end
```

```
k=0;
for jj=1:length(X)
    for ii=2:length(B)
        if (B(2,ii)>X(jj) && B(2,ii-1)<X(jj)) ||
(B(2,ii)<X(jj) && B(2,ii-1)>X(jj))
            k=k+1;
            Bnew(1)=(((X(jj)-B(2,ii-1))*(B(1,ii)-B(1,ii-
1)))/(B(2,ii)-B(2,ii-1)))+B(1,ii-1);   %interpolates to
find the value related to mu=[X]
            Bnew(2)=X(jj); %relates the fuzziness with
the previous step
            N(k,:)=Bnew; %sends to a new matrix outside
of for loop to ease calculation of vertex function
        elseif B(2,ii)==X(jj)
            k=k+1;
            Bnew=[B(1,ii),B(2,ii)];
            N(k,:)=Bnew;
        end
    end
end

kk=0;
for ii=1:length(N)
    for jj=1:length(M)
        if M(jj,2)==N(ii,2)
            kk=kk+1;
            Z(kk,2)=M(jj,1)+N(ii,1);
            Z(kk,1)=M(jj,2);
        end
    end
end


Z=sortrows(Z);
kk=1;
for ii=2:length(Z)
   if  Z(ii,1)>Z(ii-1,1)
       W(kk,:)=Z(ii-1,:);
       kk=kk+1;
       W(kk,:)=Z(ii,:);
       kk=kk+1;
   end
end
   W(kk,:)=Z(end,:);
   P=[W(:,2),W(:,1)];
end
```

The MATLAB code for the subtraction of fuzzy sets, except where Z(kk,2)=M(jj,1)+N(ii,1); would now become Z(kk,2)=M(jj,1)-N(ii,1); To save room, this code is not displayed. This rule also applies for the multiplication of fuzzy sets, where Z(kk,2)=M(jj,1)+N(ii,1); now becomes Z(kk,2)=M(jj,1)*N(ii,1).

As previously mentioned, all the fuzzy arithmetic MATLAB codes utilize the vertex method to compute the addition, subtraction, or multiplication.

## Vita

Dana Reuther was born on November 9, 1990 to Mary and Tom Reuther in Bryn Mawr, PA. She has lived in Ridley Township, PA, with her parents and attended Ridley High School in Folsom, PA.

In August 2010, Dana began attending Temple University to pursue her Bachelor's degree in Mechanical Engineering, during which time she was involved in many hands-on projects. Starting in November 2011, she was a research assistant in a biofluidics lab, where she designed and conducted experiments to investigate properties of a common biopolymer. During the same time, Dana was involved with the Temple Lunabotics Team, where she assisted with the design and prototyping of a lunar mining vehicle for the NASA Lunabotics Competition, held at Kennedy Space Center in FL. Her final year at Temple University, Dana took on the lunar mining vehicle project (LMV) as her senior design project. She then became responsible for the design and prototyping of the LMV. Dana graduated from Temple University in May 2013 with a B.S. in M.E.

After receiving her Bachelor's degree, Dana transferred to Lehigh University, where she began pursuing her Master's degree in Mechanical Engineering in the Department of Mechanical Engineering and Mechanics. She immediately came into contact with Professor Meng-Sang Chew, who she began working with towards writing this thesis. Dana finished her Master's program in three semesters and ended her time at Lehigh University in December 2014, where her final graduation date is January 18, 2015.

Dana is looking forward to a career in a research and development department in a company that will appreciate her work and contributions to the engineering world.