Glen Browne

EE 476 - Professor Tsang

April 30, 2018

Fuzzy Logic Project Proposal



**Introduction: The Theory of Fuzzy Logic:**

Fuzzy logic arose out of engineers’ need to design systems that utilize variable levels of truth with continuous values from 0 to 1, in contrast to traditional (boolean) logic where logical variables can only possess one of two values: 0 or 1. Application of fuzzy logic is useful in many fields that handle inputs with ambiguous or “fuzzy” values, which, with fuzzy logic, can be successfully used to run a system accurately and efficiently.

A fuzzy logic-based process follows the following flow chart:

**Crisp inputs -> Fuzzification -> Inference Rules -> Fuzzy Output - Defuzzification -> Crisp Outputs**

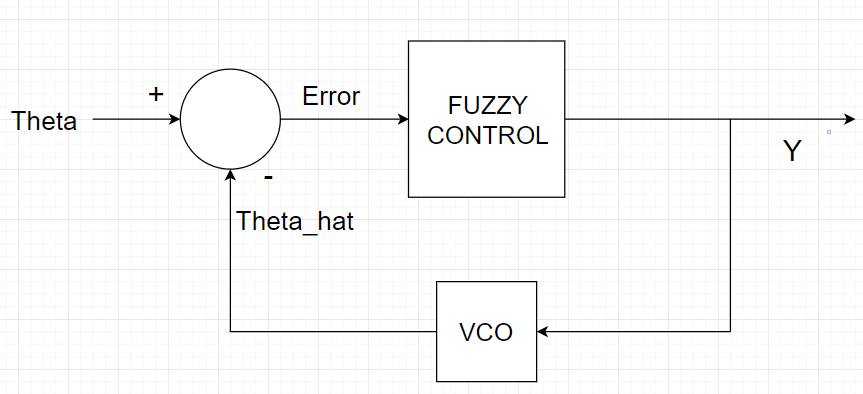
In words, crisp inputs are single-valued inputs, and crisp outputs are single-valued outputs. The fuzzification process maps each crisp input to a number of membership functions in the input’s universe of discourse or range. Each input may have a non-zero membership value (ranging from 0 to 1 continuously) in one or more of that input’s membership functions. After fuzzification, inference rules are used to map the combination of fuzzified inputs to the membership function in the control or output space. The fuzzy output or control value is then defuzzified into a crisp output. In many systems, such as control systems, the crisp outputs are used to determine the crisp inputs for the next cycle of the fuzzy logic system.

Applications of fuzzy logic are wide in range, from extremely accurate and efficient control systems, to artificial intelligence, to medical diagnosis and other classification systems. Fuzzy logic systems are particularly useful for engineering consumer products because the inputs can be entered in human language, with qualifiers such as heavy, light, medium, very, extremely, mild, etc.

**The Problem: Creating a Phase-Locked Loop with Fuzzy Logic**

The problem to be solved by this project is to implement a phase-locked loop using fuzzy logic. A phase locked loop is a negative-feedback control system that adjusts its output to match the phase of the input. In this fuzzy-rule based system, varying levels of error, and vary rates of the change in error produce varying levels of correction to the output phase of the system. The objective of the system is to keep the output “in lock” with the input by synchronizing the phases of the input and output. The error signal of a PLL is also useful in many applications such as FM demodulation.

The fuzzy controller is only part of the PLL:



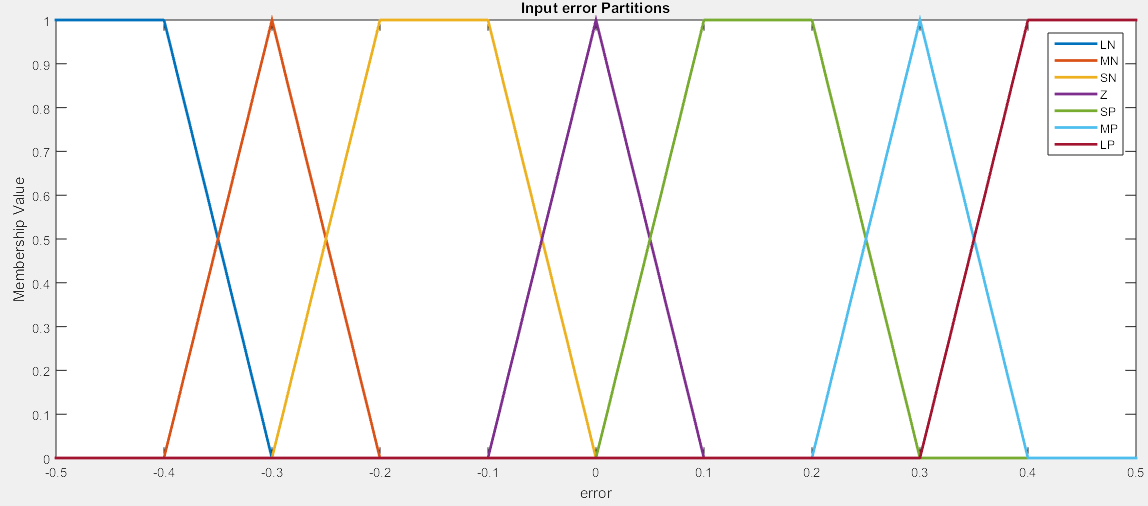
The entire system will take an input of a sine wave with angle theta and frequency fc. The fuzzy controller will take as inputs the error of the angle (theta - theta\_hat) and the rate of change of the error. The fuzzy controller will use these inputs to determine how to correct the output to produce an output that is synchronized with the input.

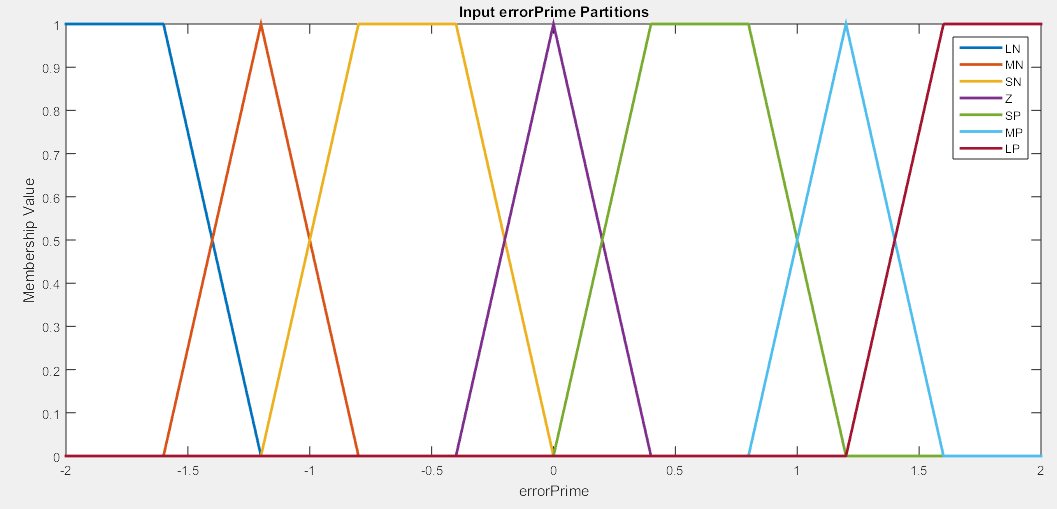
**Approaches for Solving the Problem and Different Designs Considered:**

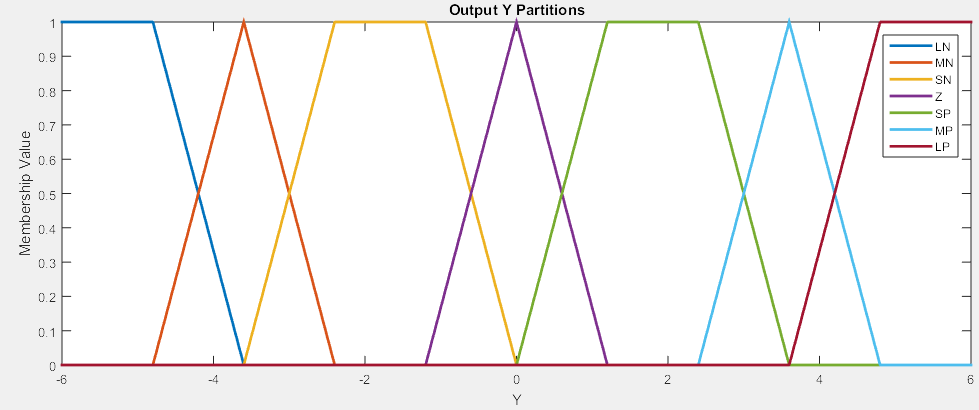
Fuzzy logic systems can be designed in a number of different ways. Different universes of discourse, numbers of membership functions, and membership function shapes can be used for each input and output. Also, different Inference rules can be used to determine the fuzzy output given fuzzy inputs.

My approach for solving this phase-locked loop problem was to use error and the derivative of error as the two crisp inputs to the fuzzy system, and a variable called y as the fuzzy output. I decided on using a 7-membership function input space to fuzzify each crisp input, and a 7-membership function output space to create the fuzzy output.

The fuzzy input and output spaces are shown in the following three figures:

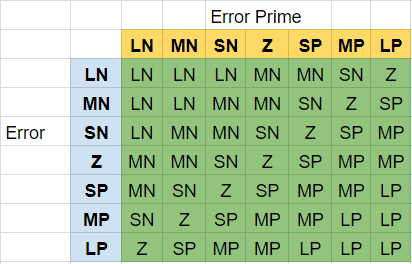






I am proposing 7-membership function spaces because I think that will allow for more accuracy and gradual correction than a lower number of membership functions, although 21 membership functions will take a lot of coding.

My proposed inference rules are summarized in the following fuzzy associative memory (FAM) table:

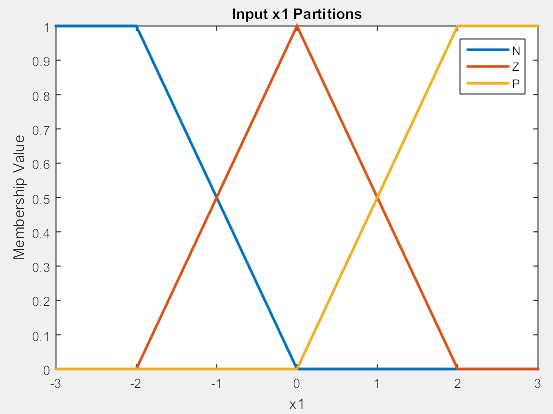


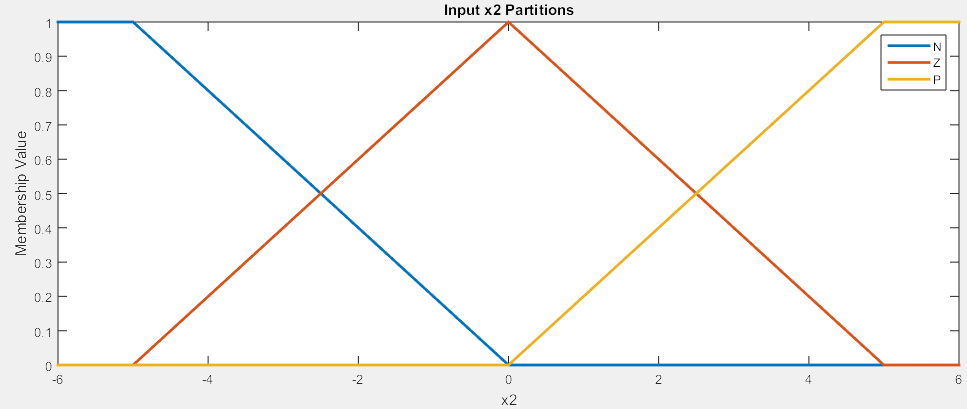
I am choosing to use the max-min inference method to implement the fuzzification process, since it seemed to work well with my sample inverted pendulum problem. The max-min method uses the minimum membership value of the inputs if each input has a nonzero value on the same membership function, then takes the maximum membership value of the output membership functions if there are more than one non-zero membership values of each respective function. I could have mapped the table differently, such as saying that a LN and a MP produce a Z, instead of an SN, but I erred on the high side in each case, so that the correction will be fast. Defuzzification will be done by the centroid method (demonstrated in the code for the inverted pendulum simulation).

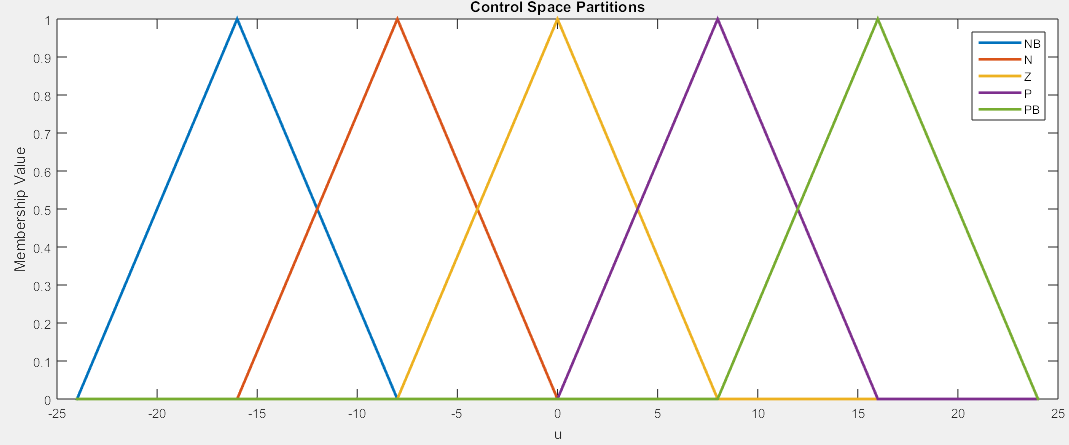
**Demonstration of Feasibility of Design (Fuzzy Logic Control of an Inverted Pendulum)**

In order to practice implementing a fuzzy-rule based system before undertaking the comparatively large and complex PLL project, I designed a fuzzy-rule based inverted pendulum control system. The following graphics and MATLAB code describe the fuzzy logic control process I used and demonstrate the feasibility of this method. The final graph entitled “Inverted Pendulum Behavior vs Iteration” demonstrates that the pendulum does alternate between small negative and small positive values indefinitely, which is the supposed outcome of the system.

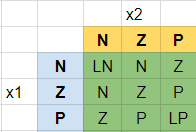
**Input/Control Space Partitions and Corresponding Membership Functions**

****

****

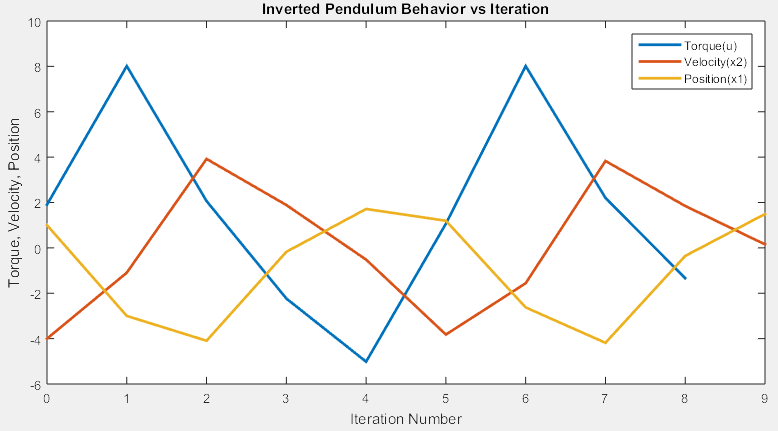
****

**FAM Table**

****

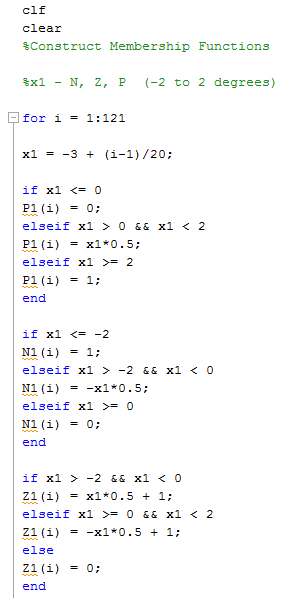
**Variable Values vs Iteration Number**

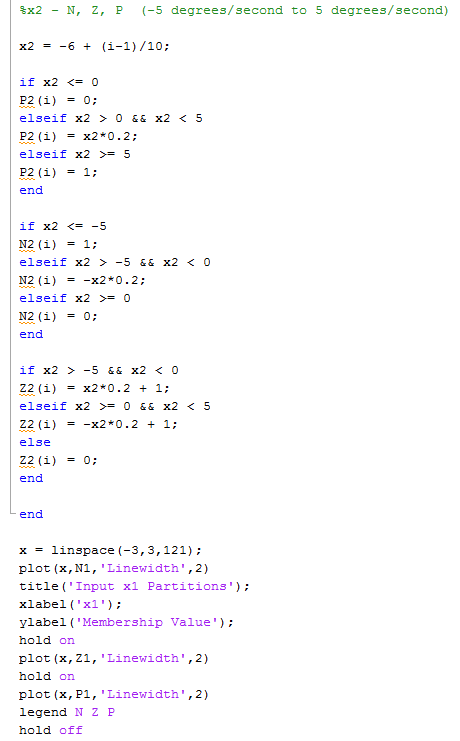


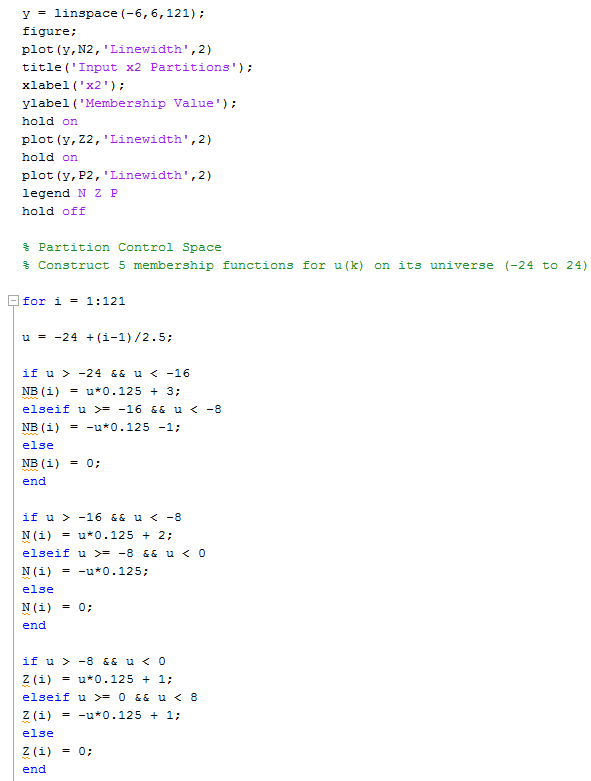


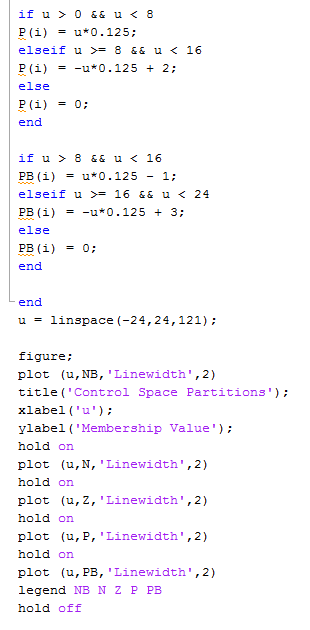
MATLAB Code:

File 1: Defining and Checking Membership Functions for Correctness



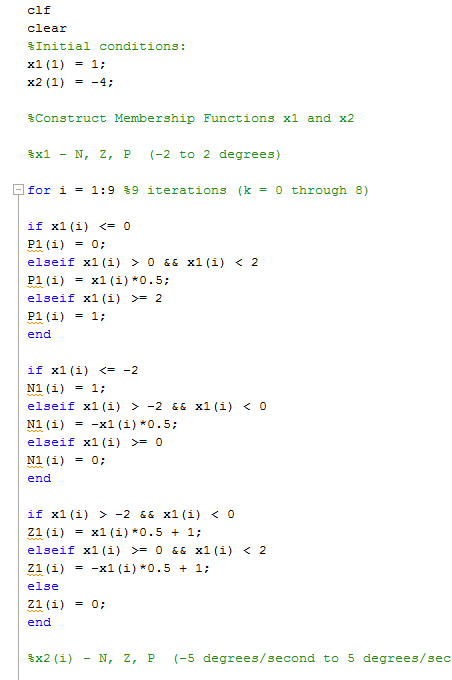


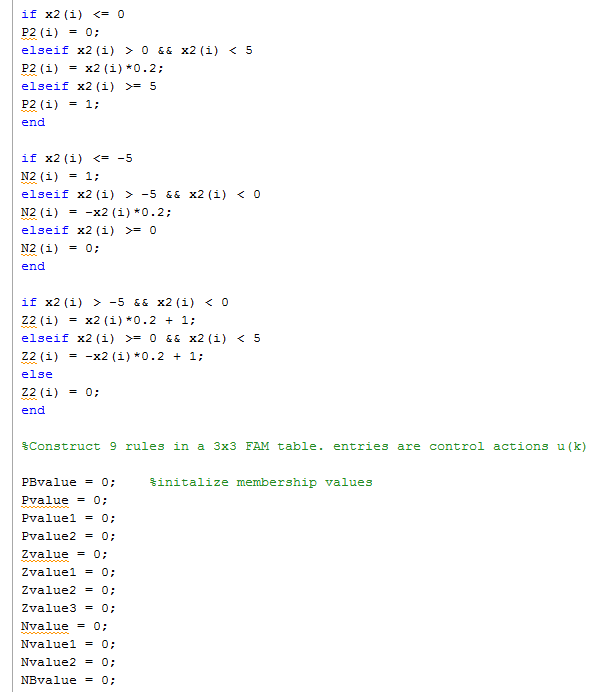


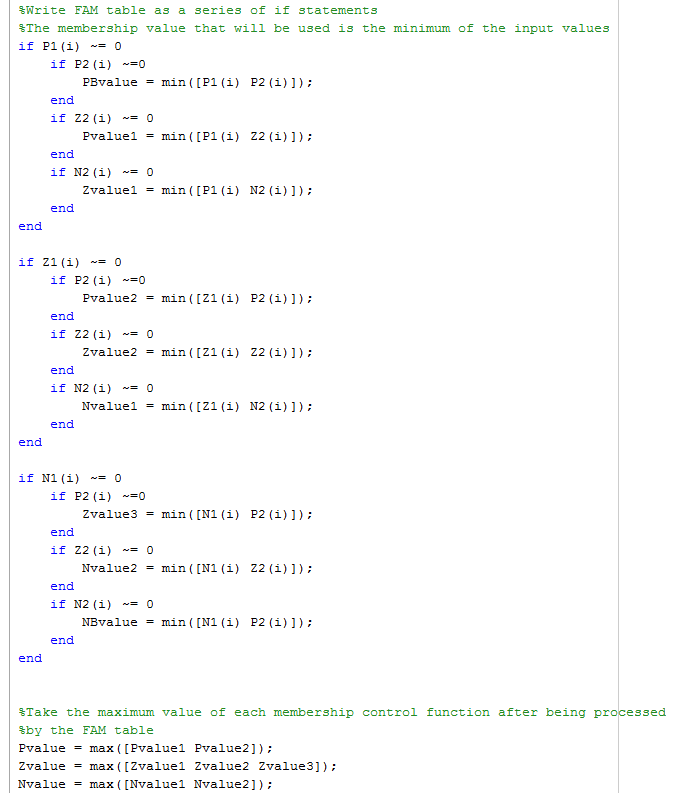


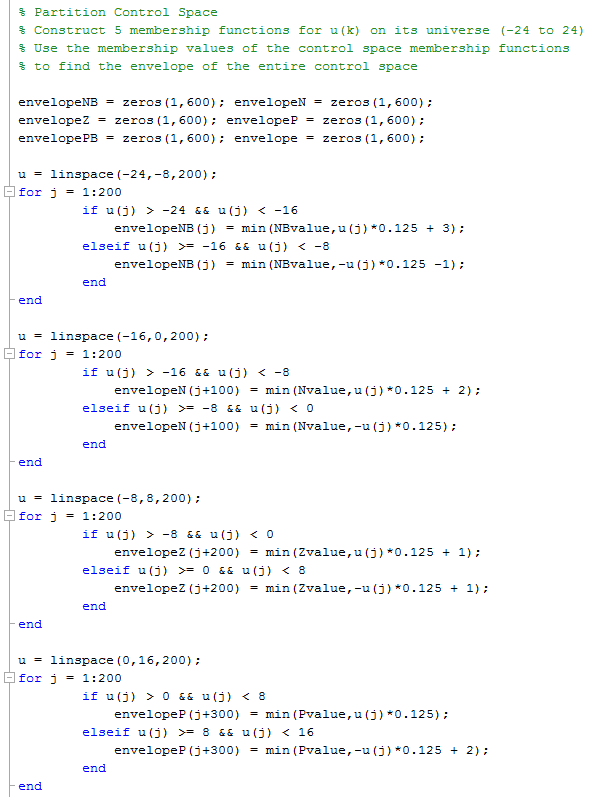
END OF FILE 1

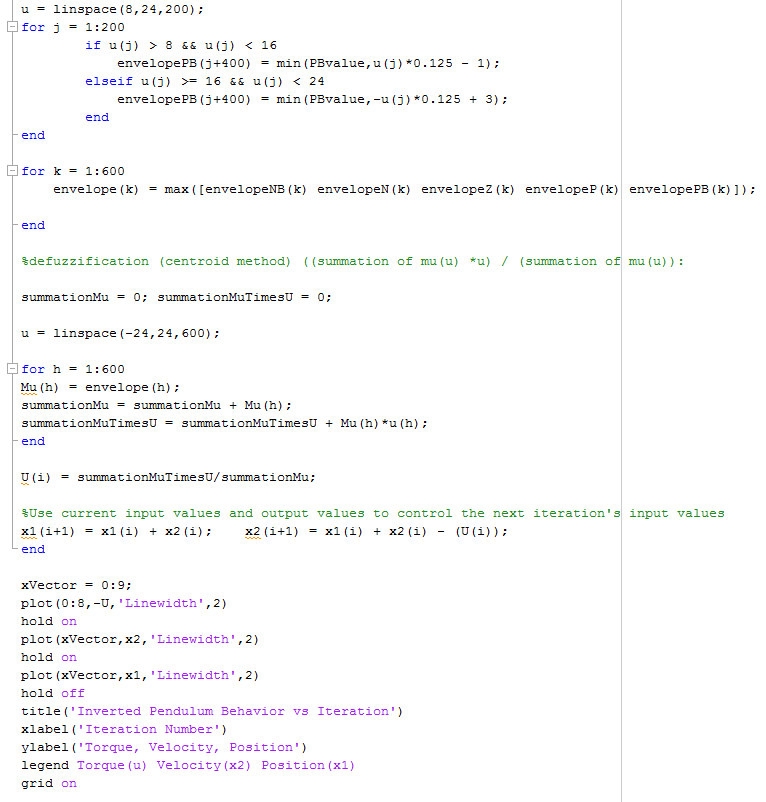
File 2: Simulating the Inverted Pendulum











END OF FILE 2