Νευρο-ασαφής Έλεγχος

9ο Εξάμηνο 2023 – 2024

Assignment 1 – Solutions

Ζαρίφης Στέλιος – el20435
Email: el20435@mail.ntua.gr

Contents

1 Ασαφής Ελεγκτής για Σύστημα Ηλεκτρικού Σιδηροδρόμου

1.1 Μοντέλο

Θεωρούμε τις εξισώσεις του τραίνου

$$m\frac{d^2x}{dt^2} = -\left(a_1 + ma_2\right)v - a_3 v|v| + bu$$

Και τις ονομαστικές τιμές $m = 100, a_1 = 1, a_2 = 0.05, a_3 = 0.1$ και $b = 200$

Χρησιμοποιούμε την εφαρμογή Simulink του MATLAB για να προσομοιώσουμε το σύστημα. Αρχικά ορίζουμε το block diagram που κωδικοποιεί τις εξισώσεις

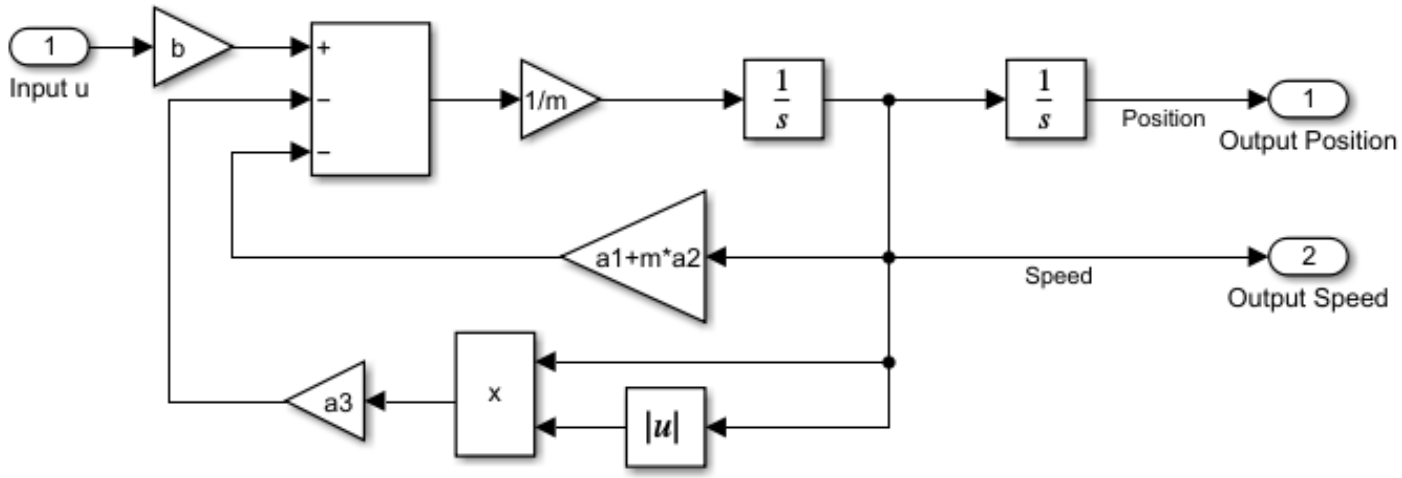


Figure 1: Train Equations Block Diagram

Ακόμα, ορίζουμε μία διάταξη η οποία θέτει την είσοδο του συστήματος του τραίνου σε μηδενική, εφόσον προσπαθεί το τραίνο να επιβραδύνει με αρνητική ταχύτητα.

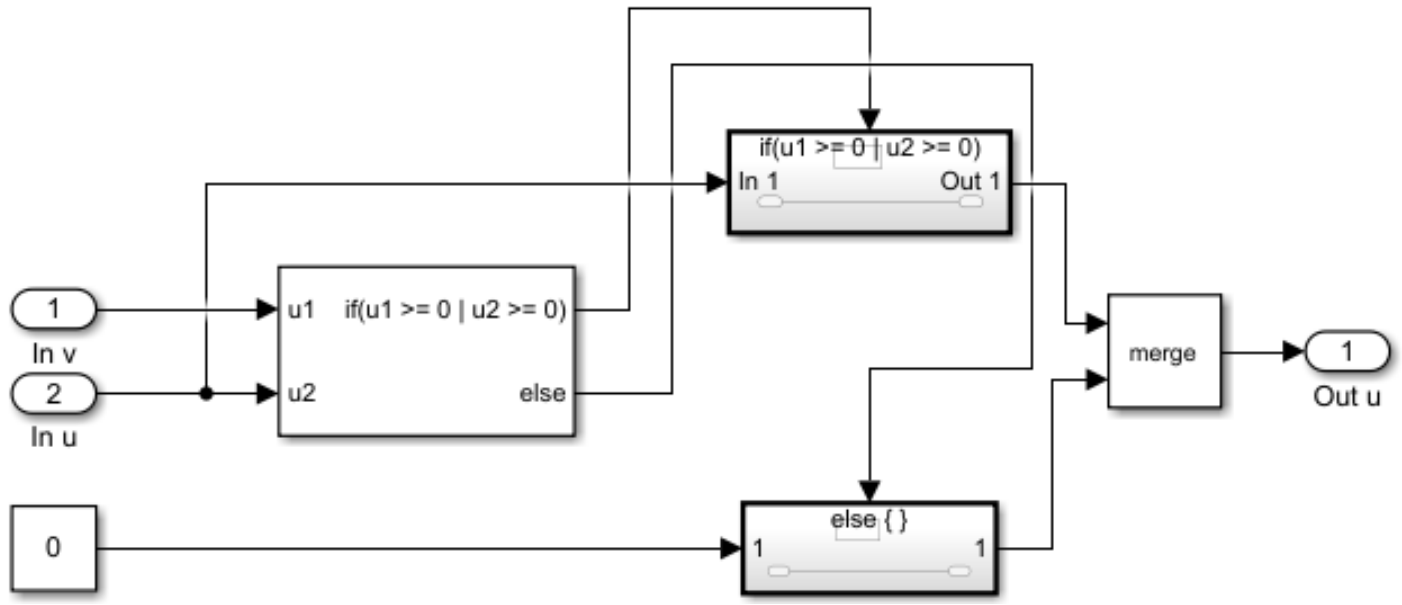


Figure 2: Layout to Prevent Deceleration at Negative Speed.

Οι 2 αυτές διατάξεις, μαζί με ένα παρεμβαλόμενο saturation block που εμποδίζει την είσοδο στο σύστημα να λάβει τιμές εκτός του διαστήματος $[-1, 1]$, συγκροτούν το ολικό σύστημα προς έλεγχο, όπως φαίνεται και στην επόμενη εικόνα.

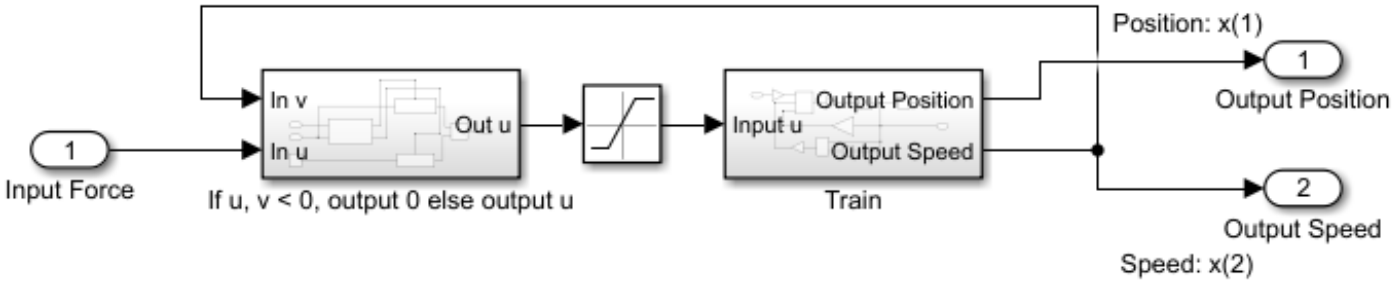


Figure 3: System Block Diagram

1.2 Προδιαγραφές

Οι 3 προδιαγραφές είναι οι εξής:

1. Μεταξύ των σταθμών το τρένο να αναπτύσσει μια ταχύτητα αναφοράς.
2. Να φτάνει στον προορισμό του γρήγορα και χωρίς υπερύψωση.
3. Η είσοδος να ικανοποιεί $u \in [-1, 1]$.

1.3 Σύντομη Επισκόπηση Κλασσικών Μεθόδων Ελέγχου

Υλοποιήσαμε 6 παραδείγματα κλασσικού ελέγχου για το σύστημα. Περιγράφουμε σύντομα παρακάτω.

1.3.1 P Control with Speed Feedback

Εφαρμόζουμε P Control με feedback την ταχύτητα.

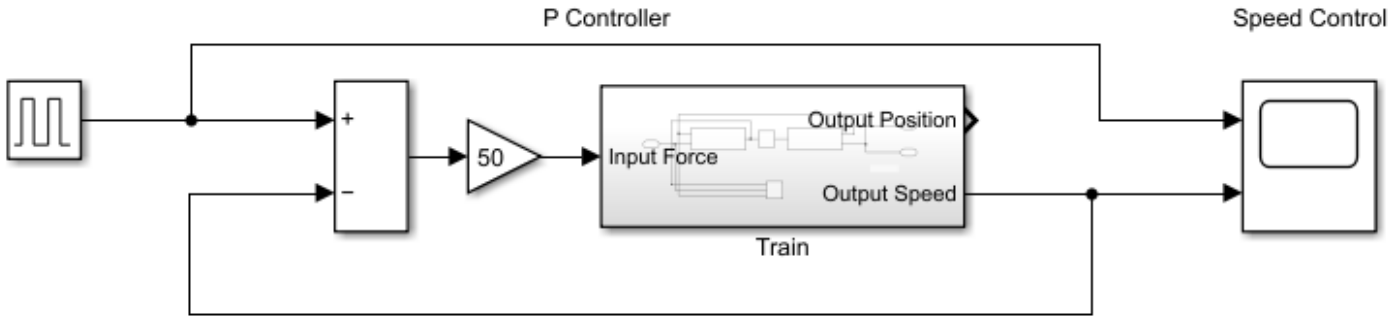


Figure 4: Speed P Control Block Diagram

Κάνουμε plot τις μετρήσεις και βλέπουμε πως υπάρχει error στην επιθυμητή ταχύτητα.

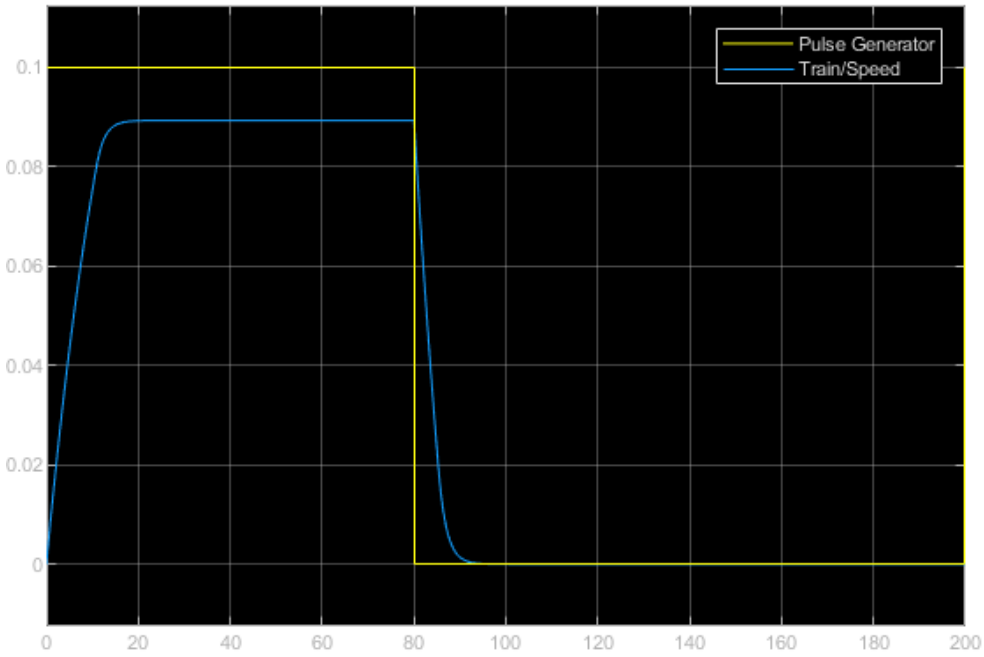


Figure 5: Speed P Control

1.3.2 PI Control with Speed Feedback

Εφαρμόζουμε PI Control με feedback την ταχύτητα.

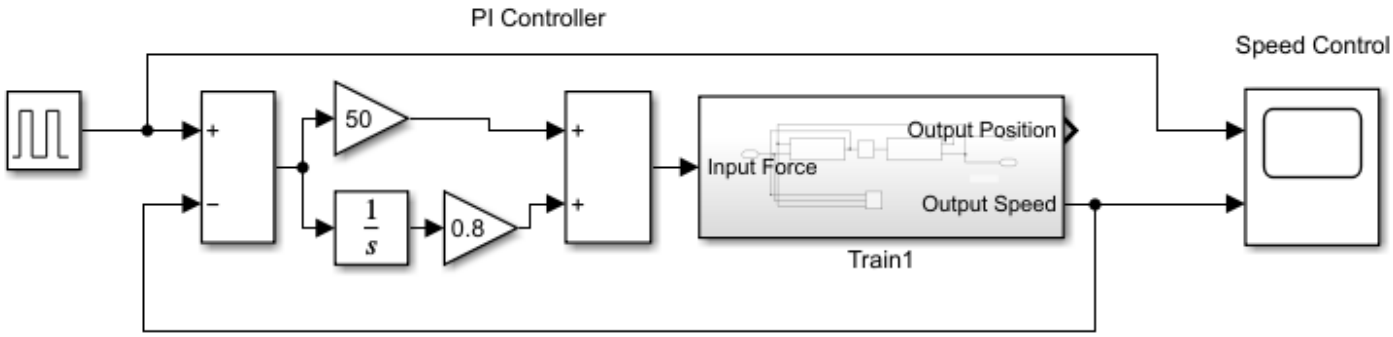


Figure 6: Speed PI Control Block Diagram

Κάνουμε plot τις μετρήσεις και βλέπουμε πως το error της ταχύτητας έχει μειωθεί σημαντικά.

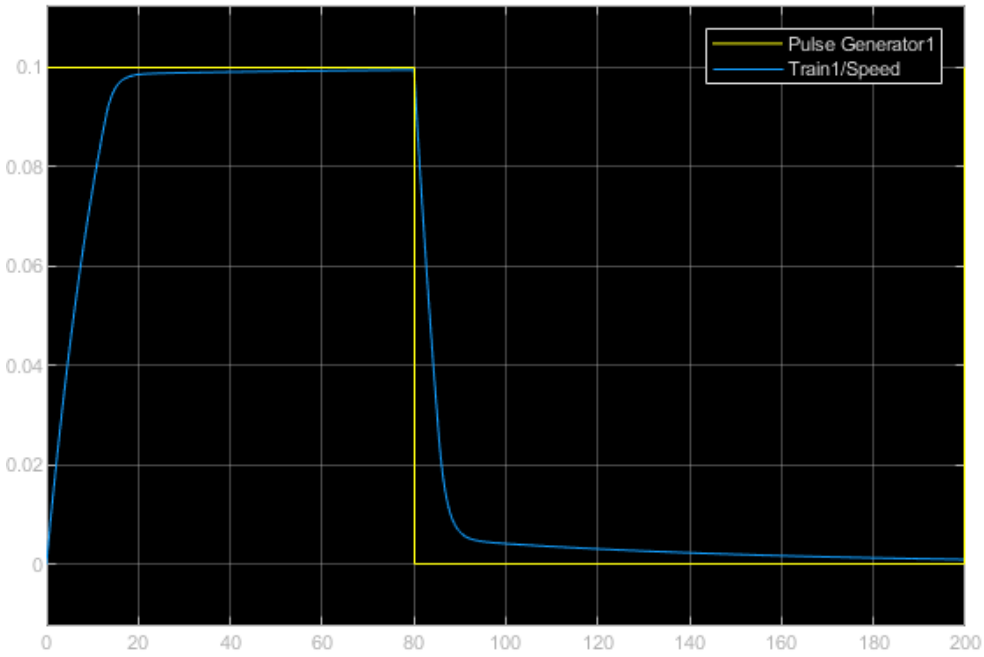


Figure 7: Speed PI Control

1.3.3 P Control with Position Feedback

Εφαρμόζουμε P Control με feedback τη θέση.

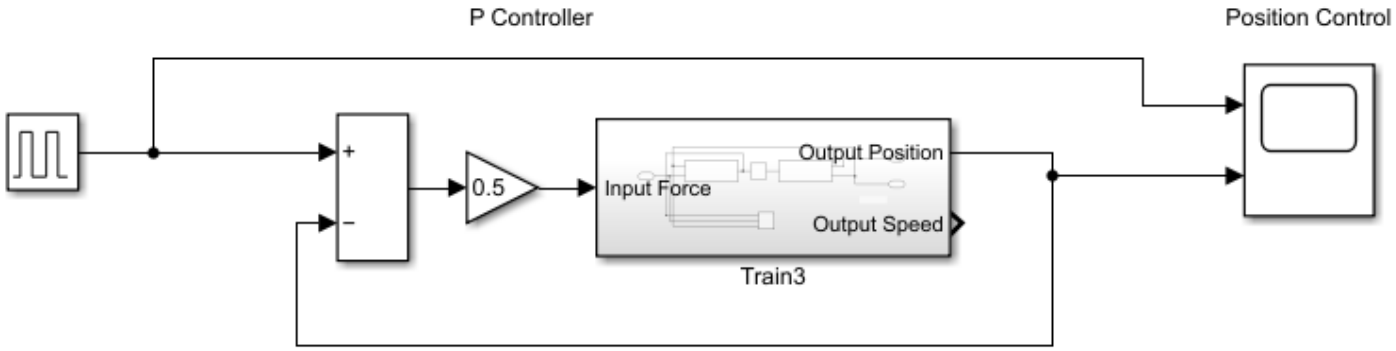


Figure 8: Position P Control Block Diagram

Κάνουμε plot τις μετρήσεις και βλέπουμε πως υπάρχει error στην επιθυμητή θέση.

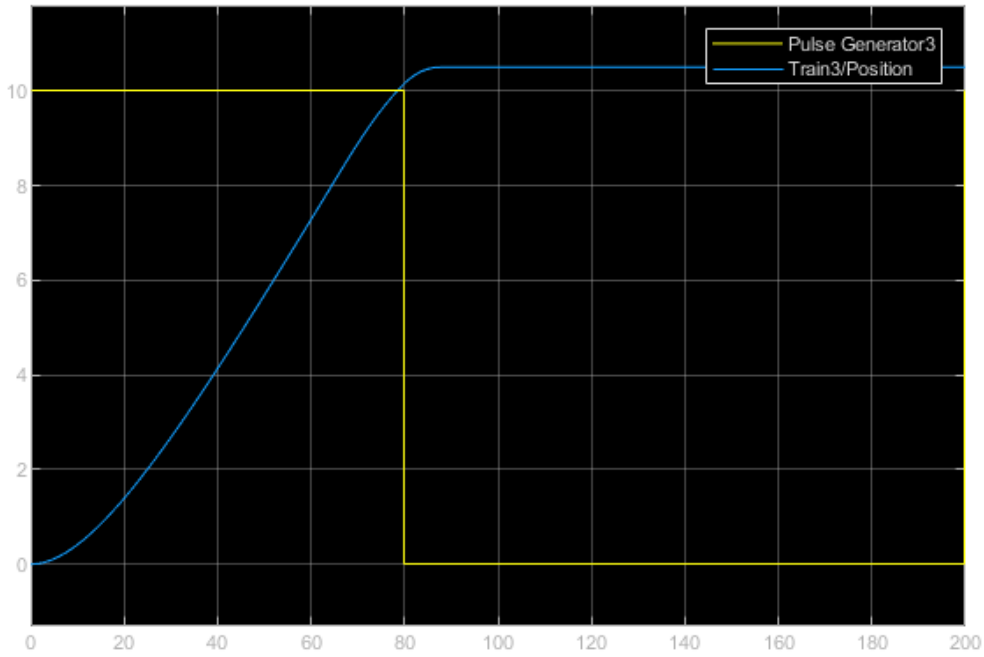


Figure 9: Position PI Control

1.3.4 PI Control with position Feedback

Εφαρμόζουμε PI Control με feedback τη θέση.

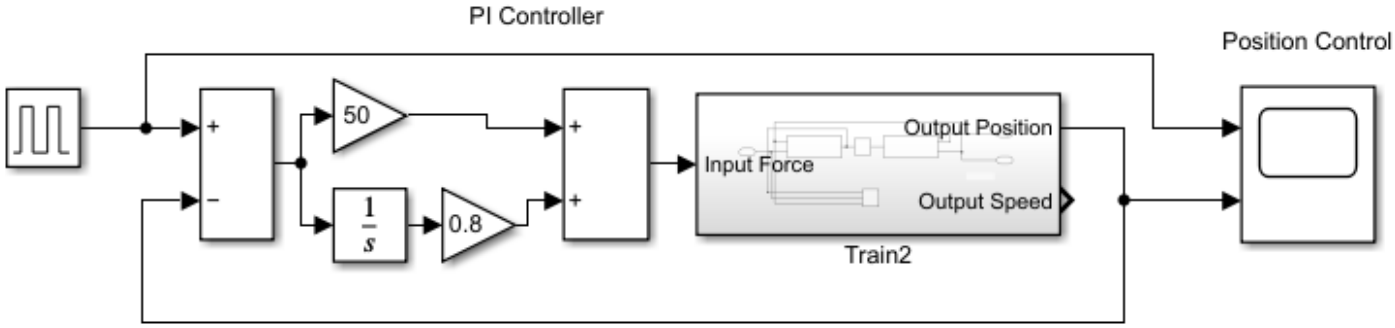


Figure 10: Position P Control Block Diagram

Κάνουμε plot τις μετρήσεις και βλέπουμε πως το error της θέσης δεν έχει μειωθεί.

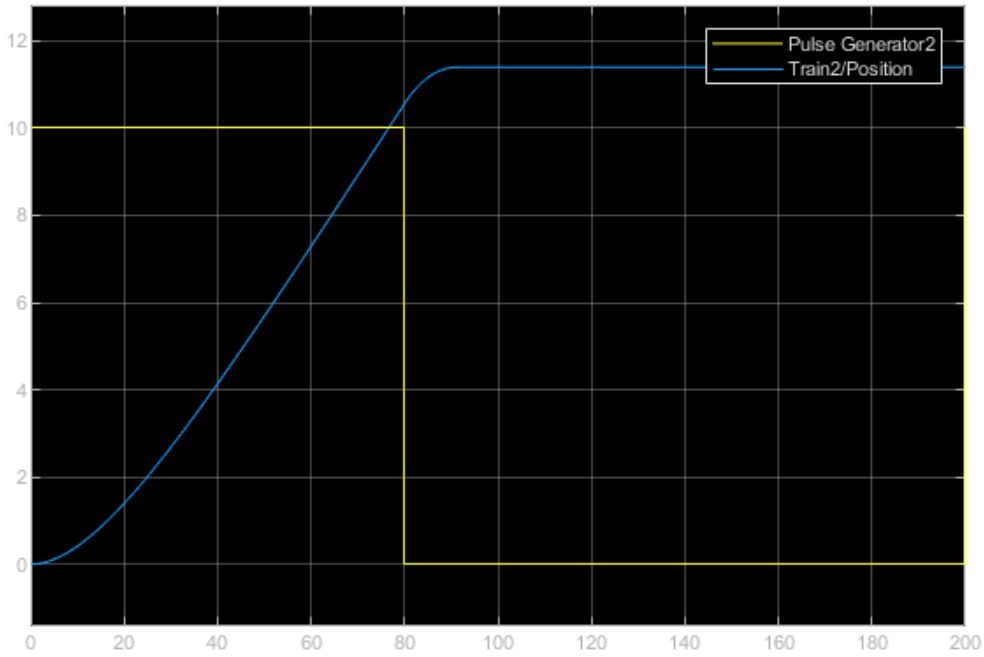


Figure 11: Position PI Control

1.3.5 P Control without Preventing Deceleration at Negative Speed

Παραθέτουμε και τις παρατηρήσεις μας αν αφαιρέσουμε τη διάταξη που αποτρέπει την επιβράδυνση με αρνητική ταχύτητα.

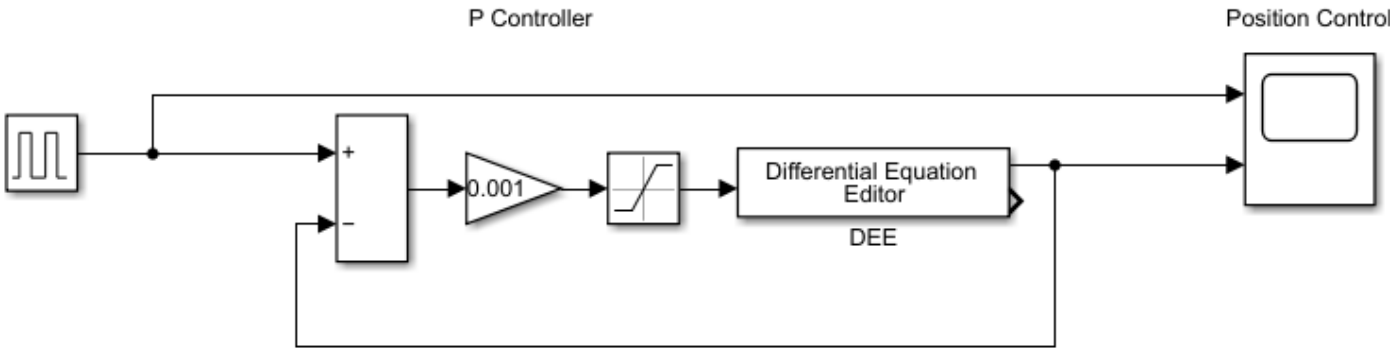


Figure 12: Position P Control Block Diagram Without the Layout

Κάνουμε plot τις μετρήσεις και βλέπουμε πως έχουμε υπερύψωση.

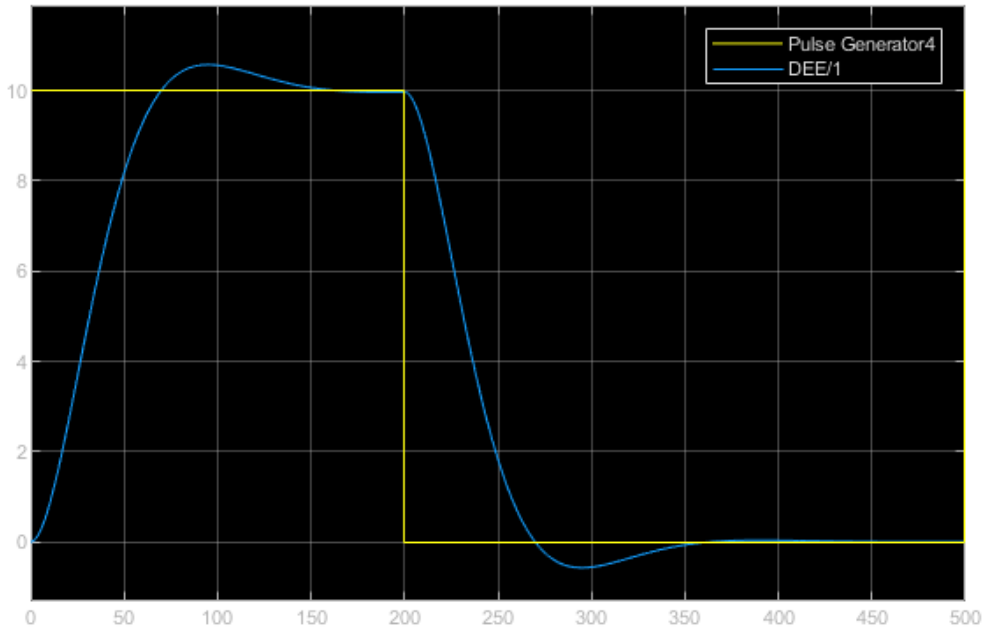


Figure 13: Position P Control Without the Layout

1.3.6 PI Control without Preventing Deceleration at Negative Speed

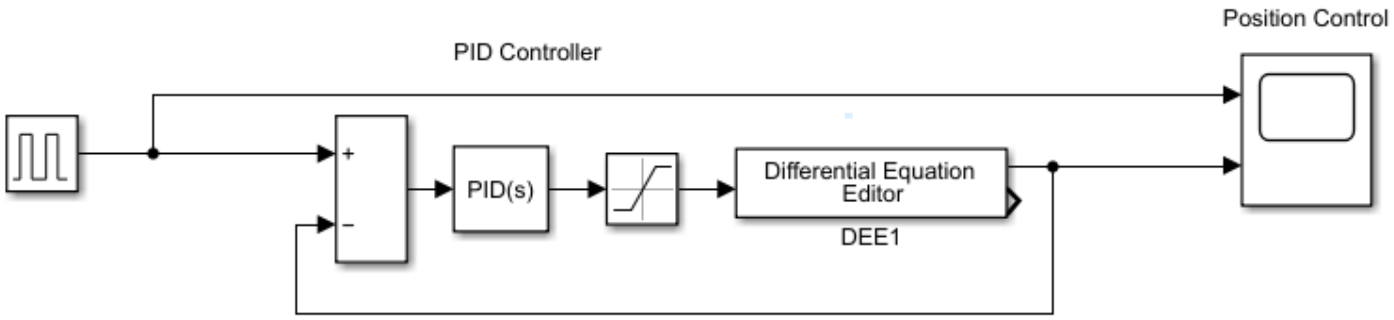


Figure 14: Position PI Control Block Diagram Without the Layout

Κάνουμε plot τις μετρήσεις και βλέπουμε πως καταλήξαμε σε έναν καλό ελεγκτή!

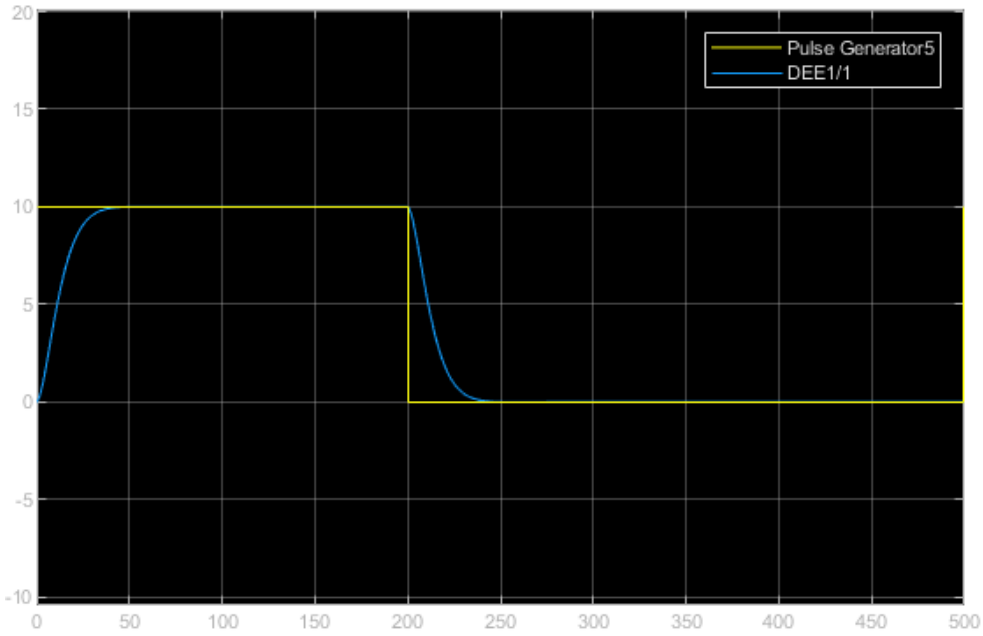


Figure 15: Position PI Control Without the Layout

1.4 Fuzzy Ελεγκτής

Το tuning, ήταν γενικά δύσκολο με τις κλασσικές μεθόδους και σε κάποια παραδείγματα δεν καταφέραμε να έχουμε το επιθυμητό αποτέλεσμα. Θα δούμε πώς η χρήση ενός Fuzzy Controller καθιστά τη διαδικασία του tuning πιο απλή και φυσική.

1.4.1 Position Membership Functions

Ορίζουμε τις συναρτήσεις συμμετοχής του ελεγκτή για θέση, ταχύτητα και δύναμη (είσοδο στο σύστημα του τραίνου). Θεωρούμε το στόχο να βρίσκεται στα 1000 μέτρα από την εκκίνηση. Για τη θέση θεωρούμε τα εξής:

1. Station: Αν η απόσταση από το στόχο είναι $0\ m$, έχουμε φτάσει στο στόχο.
2. Near: Αν η απόσταση από το στόχο είναι $0 - 140\ m$, βρισκόμαστε κοντά στο στόχο.
3. Far: Αν η απόσταση από το στόχο είναι $100 - 1000\ m$, βρισκόμαστε μακριά από το στόχο.

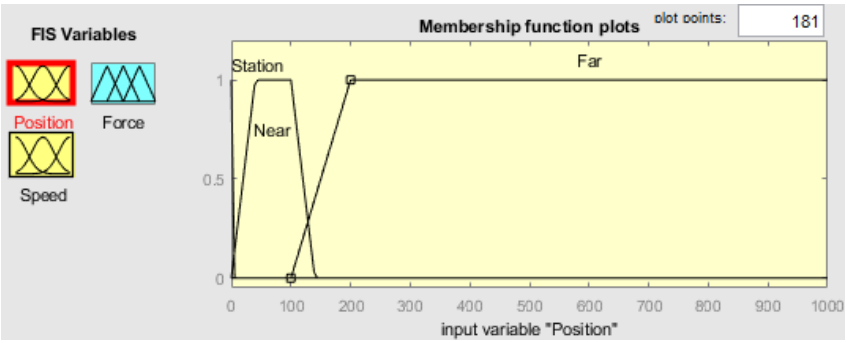


Figure 16: Position Membership Functions

1.4.2 Speed Membership Functions

Για την ταχύτητα θεωρούμε τα εξής:

1. Zero: Αν η ταχύτητα είναι $0\ m/s$, έχουμε μηδενική ταχύτητα.
2. Slow: Αν η ταχύτητα είναι $0 - 20\ m/s$, έχουμε χαμηλή ταχύτητα.
3. ref_speed: Αν η ταχύτητα είναι $20\ m/s$, έχουμε ταχύτητα αναφοράς.
4. High: Αν η ταχύτητα είναι $20 +\ m/s$, έχουμε υψηλή ταχύτητα.

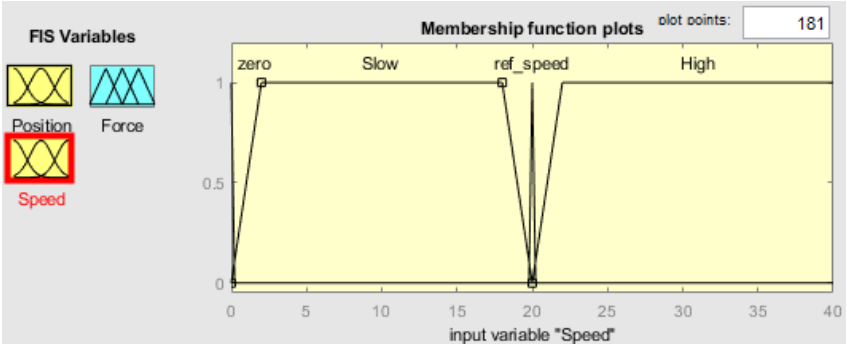


Figure 17: Speed Membership Functions

1.4.3 Force Membership Functions

Για την έξοδο ελεγκτή (είσοδος του συστήματος του τραίνου) θεωρούμε τα εξής:

1. High_Negative: Η δύναμη στο διάστημα $[-1, -0.5]$ θεωρείται υψηλή αρνητική.
2. Low_Negative: Η δύναμη στο διάστημα $[-0.7, 0]$ θεωρείται χαμηλή αρνητική.
3. zero: Η δύναμη 0 θεωρείται μηδενική (προφανώς :)).
4. Low_Positive: Η δύναμη στο διάστημα $[0, 0.7]$ θεωρείται χαμηλή θετική.
5. High_Positive: Η δύναμη στο διάστημα $[0.5, 1]$ θεωρείται υψηλή θετική.

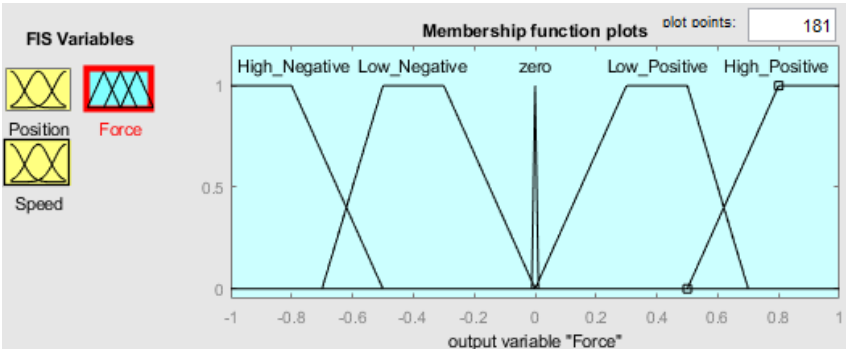


Figure 18: Force Membership Functions

1.4.4 Fuzzy Controller Rules

Τέλος, ορίζουμε τους κανόνες που θα ακολουθήσει ο ελεγκτής οι οποίοι προέκυψαν από πολύ απλούς συλλογισμούς, όπως φαίνεται παρακάτω:

1. If Position is Far and Speed is zero then Force is High Positive (1)
2. If Position is Far and Speed is Slow then Force is High Positive (1)
3. If Position is Far and Speed is ref speed then Force is zero (1)
4. If Position is Far and Speed is High then Force is Low Negative (1)
5. If Position is Near and Speed is High then Force is High Negative (1)
6. If Position is Near and Speed is ref speed then Force is High Negative (1)
7. If Position is Near and Speed is Slow then Force is Low Negative (1)
8. If Position is Station and Speed is High then Force is High Negative (1)
9. If Position is Station and Speed is Slow then Force is Low Negative (1)
10. If Position is Station and Speed is ref speed then Force is High Negative (1)
11. If Position is Station and Speed is zero then Force is zero (1)

Τα ανωτέρω μας δίνουν την επόμενη Control Surface:

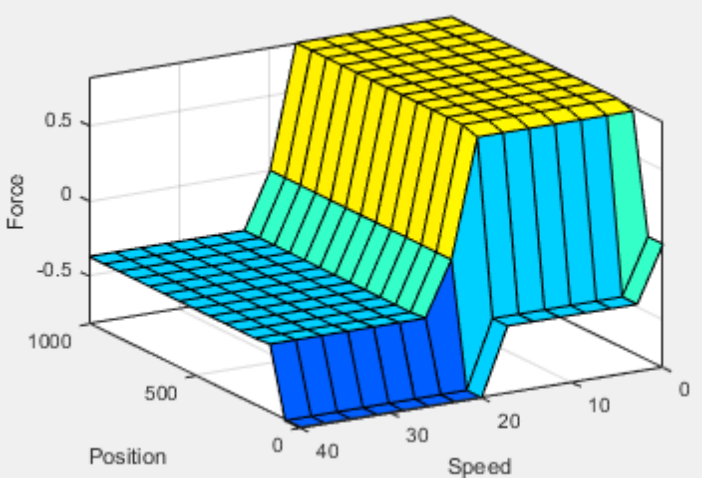


Figure 19: Control Surface

1.4.5 Controlled System

Παραθέτουμε το block diagram του συστήματος με fuzzy ελεγκτή:

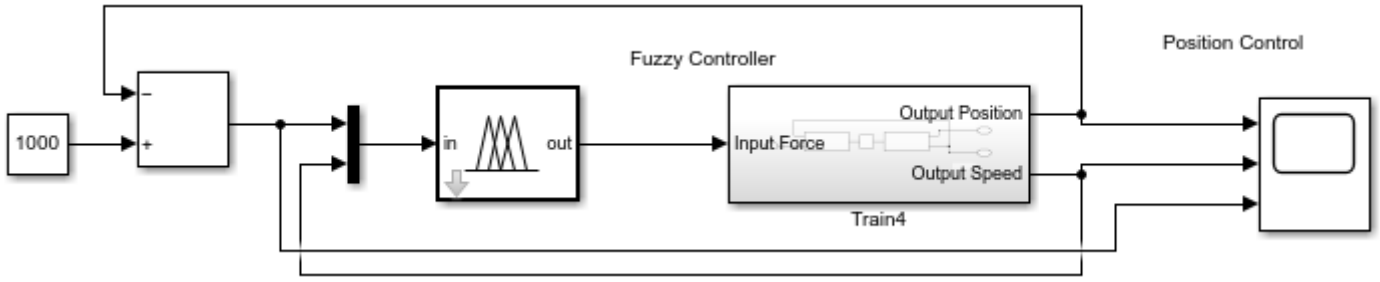


Figure 20: Position Fuzzy Control Block Diagram

Και βλέπουμε πως τα αποτελέσματα είναι πολύ καλα: Το τραίνο φτάνει στον προορισμό χωρίς υπερύψωση, αφού πρώτα έχει αναπτύξει την ταχύτητα αναφοράς και η είσοδος βρίσκεται πάντα στο επιτρεπτό διάστημα.

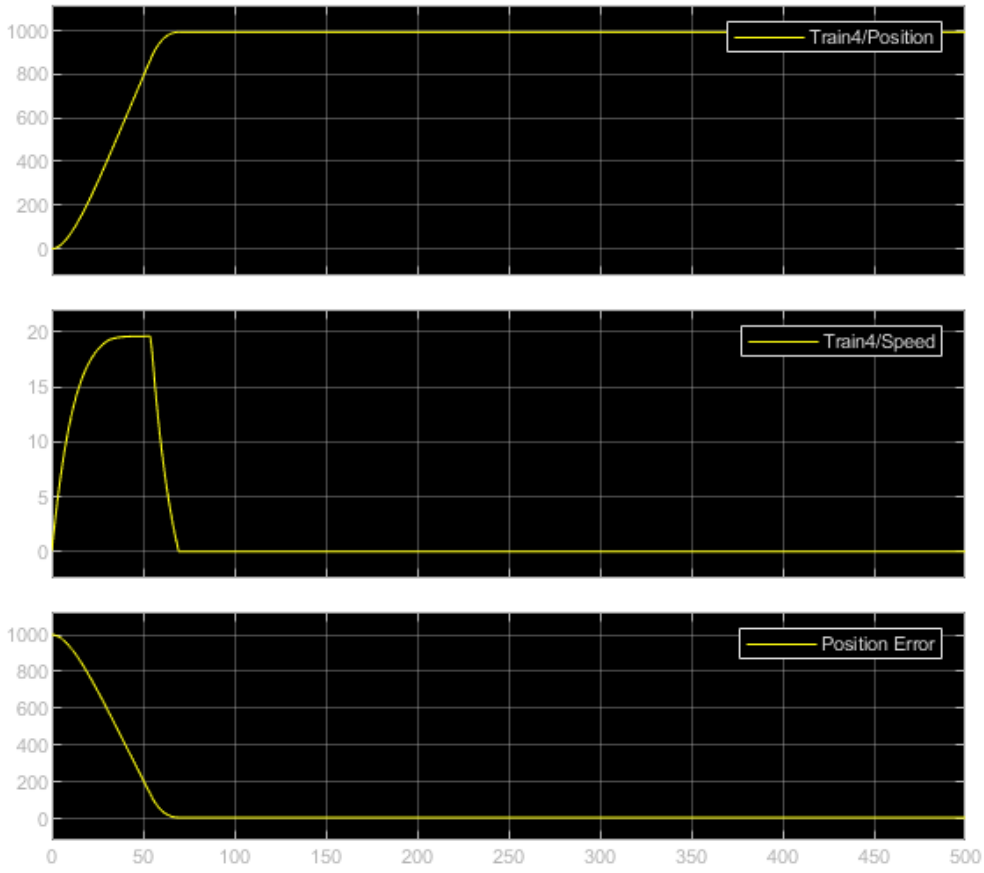
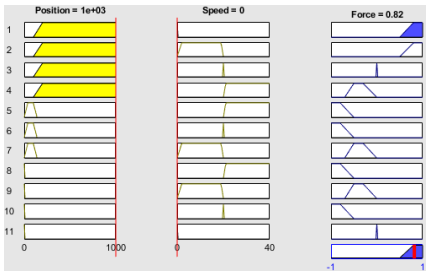
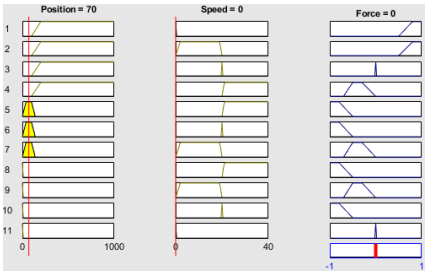


Figure 21: Position Fuzzy Control

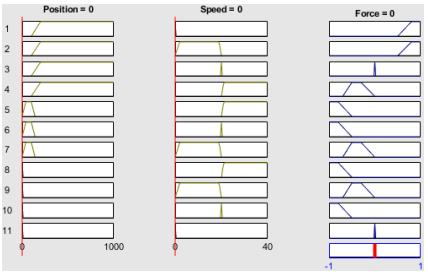
Ακόμα, παραθέτουμε ποιοι κανόνες και κατά πόσο είναι σε ισχύ για διάφορες τιμές των εισόδων. Οι στήλες είναι για αποστάσεις που θεωρούμε Far, Near, Station και οι γραμμές για ταχύτητες που θεωρούμε Zero, Slow, Reference Speed, High.



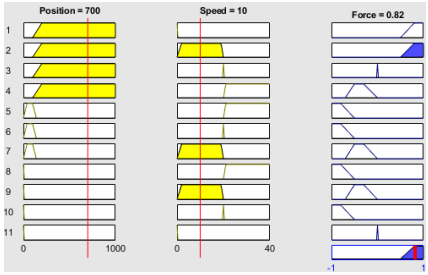
(α΄) Far and Zero speed



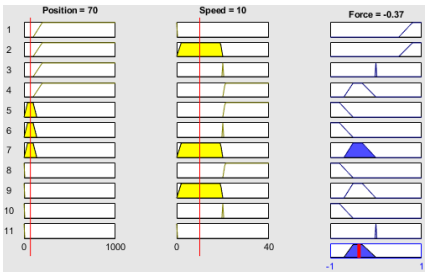
(β΄) Near and Zero speed



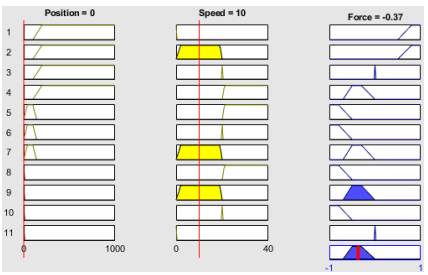
(γ΄) Station and Zero speed



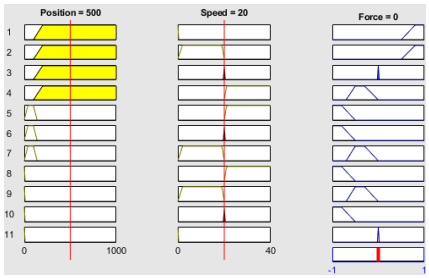
(δ΄) Far and Slow speed



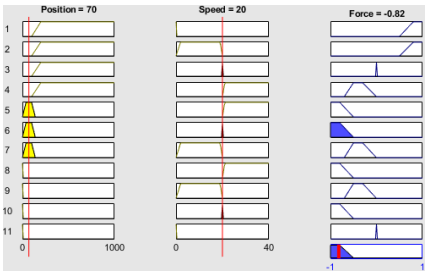
(ε΄) Near and Slow speed



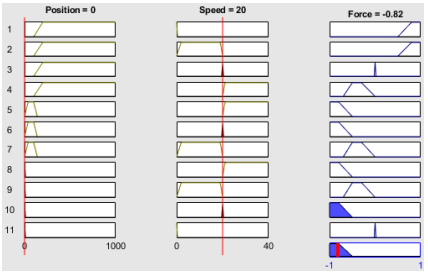
(στ΄) Station and Slow speed



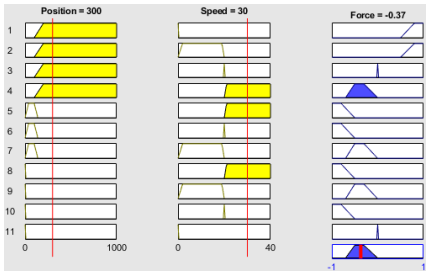
(ζ΄) Far and Reference speed



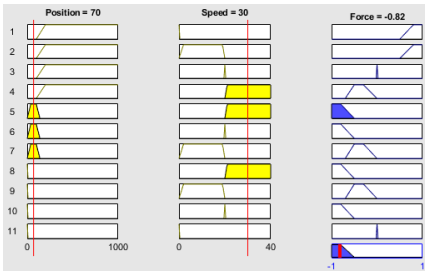
(η΄) Near and Reference speed



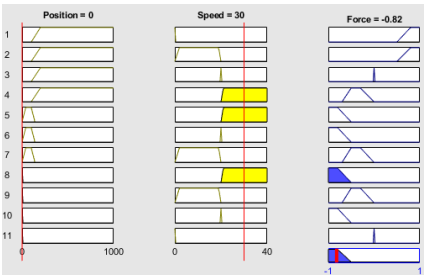
(θ΄) Station and Reference speed



(ι΄) Far and High speed



(ια΄) Near and High speed



(ιβ΄) Station and High speed

Figure 22: 12 Combinations of Inputs and Their Outputs.

1.4.6 Modified System

Αυξάνουμε τη μάζα του τραίνου κατά 10% και προσθέτουμε μια κλίση στις ράγες, έστω προς τα πίσω (προσθήκη μίας αρνητικής σταθεράς στο δεξιό μέλος της διαφορικής εξίσωσης που περιγράφει το σύστημα).

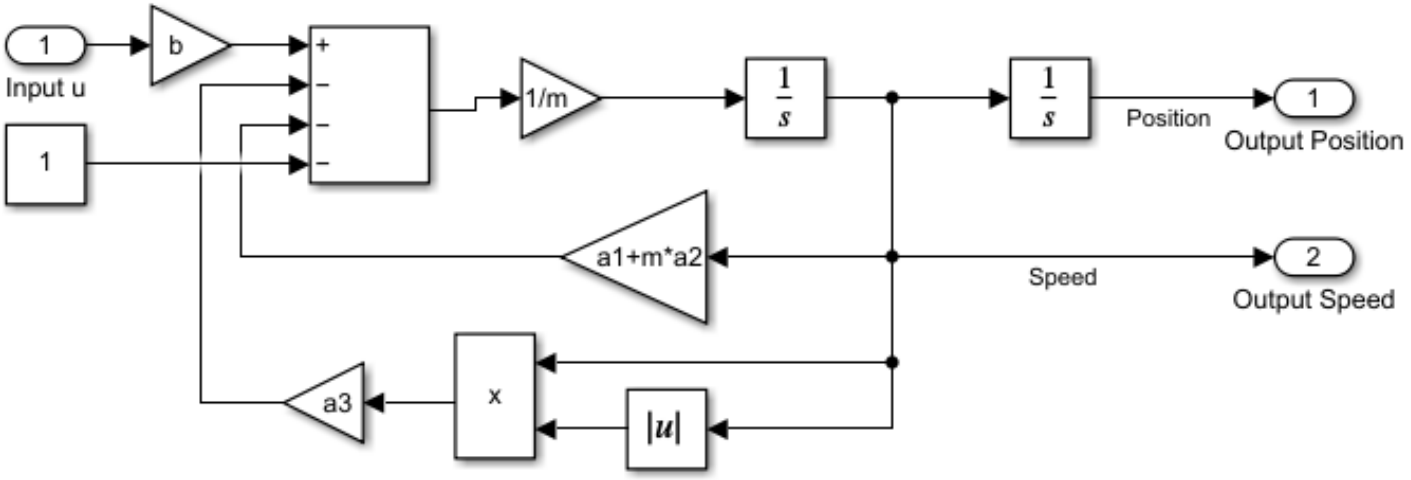


Figure 23: Position Fuzzy Control Block Diagram

Παρατηρούμε πως ο ίδιος ελεγκτής δεν είναι κατάλληλος για το νέο σύστημα. Φαίνεται πως η επιπλέον μάζα δεν επηρρεάζει πολύ το αποτέλεσμα, αλλά η κλίση κάνει το τραίνο να απομακρύνεται από το σταθμό.

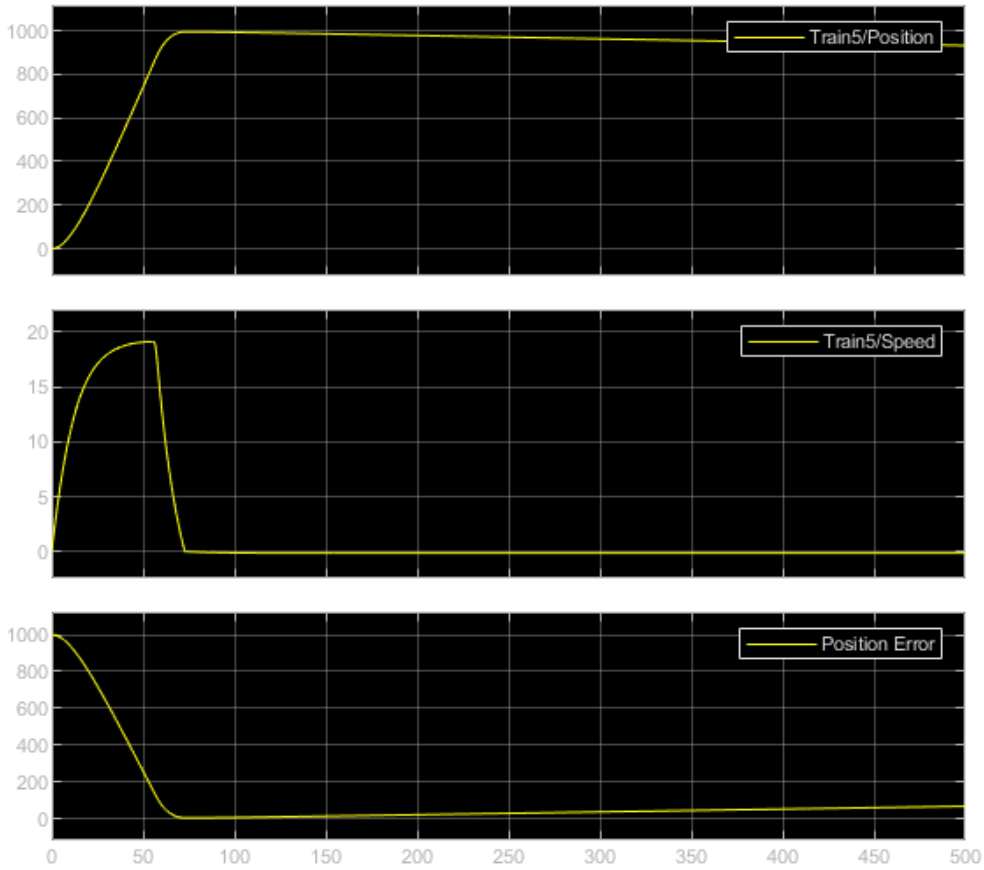


Figure 24: Position Fuzzy Control

2 Διερεύνηση Ευστάθειας σε Takagi-Sugeno Σύστημα

Σύστημα Θεωρούμε το εξής Takagi-Sugeno σύστημα

$$x_{k+1} = h_1(x)A_1x_k + h_2(x)A_2x_k, A_1 = \begin{bmatrix} 0.9 & a \\ 0 & 0.8 \end{bmatrix}, A_2 = \begin{bmatrix} 0.9 & 0 \\ a & 0.8 \end{bmatrix}$$

2.1 Ευστάθεια του Ασαφούς Συστήματος με Τετραγωνική Συνάρτηση Lyapunov

Στο ερώτημα αυτό θα διερευνήσουμε τις τιμές της παραμέτρου a για τις οποίες υπάρχει τετραγωνική συνάρτηση Lyapunov $V = x^TPx$ που αποδεικνύει την ευστάθεια του συστήματος. Χρησιμοποιούμε το εξής θεώρημα:

Το σημείο ισορροπίας ενός ασαφούς συστήματος είναι globally & asymptotically stable αν υπάρχει ένας common positive definite πίνακας P, για κάθε υποσύστημα, τέτοιος ώστε $A_i^TPA_i - P < 0, \forall i$

Χρησιμοποιούμε το εργαλείο cvx του MATLAB και η βασική διαδικασία είναι:

```
cvx_begin sdp quiet
variable P(n, n) symmetric
A1'*P*A1 - P <= -eye(n)
A2'*P*A2 - P <= -eye(n)
P >= eye(n)
cvx_end
```

Το cvx ορίζει έναν μεταβλητό πίνακα P και αριθμητικά προσπαθεί να αποφανθεί αν υπάρχει τέτοιος P που ικανοποιεί τις προϋποθέσεις του θεωρήματος. Ο υπόλοιπος κώδικας είναι απλά ένα iteration της μεταβλητής a σε ένα διάστημα με κάποιο βήμα και αποθηκεύονται οι τιμές του a για τις οποίες ξεκινούν και σταματούν να ικανοποιούνται οι προϋποθέσεις του θεωρήματος. Αυτές οι τιμές ορίζουν το ζητούμενο διάστημα της μεταβλητής a. Μια τροποποίηση που κάνουμε μετά είναι το refining του διαστήματος. Αφού έχουμε πάρει μια αρχική εκτίμηση για το διάστημα, ταλαντώνουμε λίγο τα άκρα του και -με μικρότερο βήμα- αναζητούμε πιο ακριβή άκρα. Τέλος τυπώνουμε το διάστημα

```
n = 2;
low_bound = -1;
high_bound = 1;
points = 100;
a_values = linspace(low_bound, high_bound, points);
a_min = -inf;
a_max = inf;
flag = false;

h = waitbar(0,'Please␣␣wait...');
for idx = 1:numel(a_values)
        waitbar(idx / numel(a_values))
        a = a_values(idx);
        A1 = [0.9 a; 0 0.8];
        A2 = A1';

        cvx_begin sdp quiet
        variable P(n, n) symmetric
        A1'*P*A1 - P <= -eye(n)
```

```
        A2'*P*A2 - P <= -eye(n)
        P >= eye(n)
        cvx_end

        % Check if the solution is feasible (system is stable)
        if flag == false && strcmp(cvx_status, 'Solved')
                a_min = a;
                flag = true;
                % fprintf('Common Lyapunov function exists for a = %f\n', a);
        elseif flag == true && ~ strcmp(cvx_status, 'Solved')
                a_max = a - (high_bound - low_bound)/points;
                flag = false;
                % fprintf('Common Lyapunov function does not exist for a = %f\n
    ', a);
        end
end
close(h)
fprintf('Range of a values is [%f, %f]\n', a_min, a_max);

%% First Refining
fprintf('Refining...\n')

% Refinement step with smaller steps in both directions
refinement_steps = 100;
step_size = (high_bound - low_bound) / points / refinement_steps;

h = waitbar(0,'Refining, please wait...');
for step = 1:refinement_steps
        waitbar(step / refinement_steps)
        a_min_temp = a_min - step * step_size;
        a_max_temp = a_max + step * step_size;

        % Check if changing a_min_temp affects stability
        A1 = [0.9 a_min_temp; 0 0.8];
        A2 = A1';
        cvx_begin sdp quiet
        variable P(n, n) symmetric
        A1'*P*A1 - P <= -eye(n)
        A2'*P*A2 - P <= -eye(n)
        P >= eye(n)
        cvx_end
        if strcmp(cvx_status, 'Solved') && ~flag
                a_min = a_min_temp;
                flag = true;
                break
        end
end
close(h)

h = waitbar(0,'Refining, please wait...');
for step = 1:refinement_steps
        waitbar(step / refinement_steps)
```

```
        a_min_temp = a_min - step * step_size;
        a_max_temp = a_max + step * step_size;

        % Check if changing a_max_temp affects stability
        A1 = [0.9 a_max_temp; 0 0.8];
        A2 = A1';
        cvx_begin sdp quiet
        variable P(n, n) symmetric
        A1'*P*A1 - P <= -eye(n)
        A2'*P*A2 - P <= -eye(n)
        P >= eye(n)
        cvx_end
        if ~strcmp(cvx_status, 'Solved') && flag
                a_max = a_max_temp;
                flag = false;
                break
        end
end
close(h)

fprintf('+-------------------------------------------------+\n')
fprintf('|Refined range of a values is [%f, %f]|\n', a_min, a_max);
fprintf('+-------------------------------------------------+\n')
%% Second Refining
fprintf('Refining again ...\n')

% Refinement step with smaller steps in both directions
refinement_steps = 100;
step_size = (high_bound - low_bound) / points / refinement_steps;

h = waitbar(0,'Refining for low bound, please wait...');
for step = 1:refinement_steps
        waitbar(step / refinement_steps)
        a_min_temp = a_min - step * step_size;
        a_max_temp = a_max + step * step_size;

        % Check if changing a_min_temp affects stability
        A1 = [0.9 a_min_temp; 0 0.8];
        A2 = A1';
        cvx_begin sdp quiet
        variable P(n, n) symmetric
        A1'*P*A1 - P <= -eye(n)
        A2'*P*A2 - P <= -eye(n)
        P >= eye(n)
        cvx_end
        if strcmp(cvx_status, 'Solved') && ~flag
                a_min = a_min_temp;
                flag = true;
                break
        end
end
close(h)
```

```
123
124 h = waitbar(0,'Refining␣for␣high␣bound,␣please␣wait...');
125 for step = 1:refinement_steps
126         waitbar(step / refinement_steps)
127         a_min_temp = a_min - step * step_size;
128         a_max_temp = a_max + step * step_size;
129
130         % Check if changing a_max_temp affects stability
131         A1 = [0.9 a_max_temp; 0 0.8];
132         A2 = A1';
133         cvx_begin sdp quiet
134         variable P(n, n) symmetric
135         A1'*P*A1 - P <= -eye(n)
136         A2'*P*A2 - P <= -eye(n)
137         P >= eye(n)
138         cvx_end
139         if ~strcmp(cvx_status, 'Solved') && flag
140                 a_max = a_max_temp;
141                 flag = false;
142                 break
143         end
144 end
145 close(h)
146
147 fprintf('+-----------------------------------------------------+\n')
148 fprintf('|Refined␣range␣of␣a␣values␣is␣[%f,␣%f]|\n', a_min, a_max);
149 fprintf('+-----------------------------------------------------+\n')
150 %% Third Refining
151 fprintf('Refining␣again␣:D␣...\n')
152
153 % Refinement step with smaller steps in both directions
154 refinement_steps = 100;
155 step_size = (high_bound - low_bound) / points / refinement_steps;
156
157 h = waitbar(0,'Refining␣for␣low␣bound,␣please␣wait...');
158 for step = 1:refinement_steps
159         waitbar(step / refinement_steps)
160         a_min_temp = a_min - step * step_size;
161         a_max_temp = a_max + step * step_size;
162
163         % Check if changing a_min_temp affects stability
164         A1 = [0.9 a_min_temp; 0 0.8];
165         A2 = A1';
166         cvx_begin sdp quiet
167         variable P(n, n) symmetric
168         A1'*P*A1 - P <= -eye(n)
169         A2'*P*A2 - P <= -eye(n)
170         P >= eye(n)
171         cvx_end
172         if strcmp(cvx_status, 'Solved') && ~flag
173                 a_min = a_min_temp;
174                 flag = true;
```

```
                break
        end
end
close(h)

h = waitbar(0,'Refining␣for␣high␣bound,␣please␣wait...');
for step = 1:refinement_steps
        waitbar(step / refinement_steps)
        a_min_temp = a_min - step * step_size;
        a_max_temp = a_max + step * step_size;

        % Check if changing a_max_temp affects stability
        A1 = [0.9 a_max_temp; 0 0.8];
        A2 = A1';
        cvx_begin sdp quiet
        variable P(n, n) symmetric
        A1'*P*A1 - P <= -eye(n)
        A2'*P*A2 - P <= -eye(n)
        P >= eye(n)
        cvx_end
        if ~strcmp(cvx_status, 'Solved') && flag
                a_max = a_max_temp;
                flag = false;
                break
        end
end
close(h)

fprintf('+---------------------------------------------------+\n')
fprintf('|Refined␣range␣of␣a␣values␣is␣[%f,␣%f]|\n', a_min, a_max);
fprintf('+---------------------------------------------------+\n')
```

Το αποτέλεσμα φαίνεται εδώ:

```
Range of a values is [-0.252525, 0.252727]
+---------------------------------------------------+
|Refining...                                        |
+---------------------------------------------------+
|Refined Range of a values is [-0.252725, 0.261727]|
+---------------------------------------------------+
|Refining again ...                                 |
+---------------------------------------------------+
|Refined Range of a values is [-0.252925, 0.261927]|
+---------------------------------------------------+
|Refining again :D ...                              |
+---------------------------------------------------+
|Refined Range of a values is [-0.253125, 0.262127]|
+---------------------------------------------------+
```

Η τελική εκτίμηση για τις τιμές του a είναι $a \in [-0.253125, 0.262127]$

2.2 Ασυμπτωτική Ευστάθεια

Θεωρούμε ότι $h_1\left(\min\left(\max\left(x(2)-0.5\right),0\right),1\right)$. Θα επιδιώξουμε με python να αποφανθούμε για ποιες τιμές του a το σύστημα είναι ασυμπτωτικά ευσταθές. Συγκεκριμένα, ορίζουμε ένα grid αρχικών θέσεων κοντά στο origin ($x, y \in [-2, 2]$ με βήμα 0.1) και για διάφορες τιμές του a (από -1 ως 1 με βήμα 0.01) αφήνουμε το σύστημα να εξελιχθεί από όλες τις αρχικές θέσεις και εξετάζουμε αν μετά από "πολλά" βήματα (100 βήματα) το σύστημα συγκλίνει στην αρχή των αξόνων (δηλαδή οι συντεταγμένες x, y είναι μικρότερες από ένα μικρό κατώφλι). Παραθέτουμε τον κώδικα που χρησιμοποιήσαμε και τα αποτελέσματα:

```python
import numpy as np

def system_update(x, a):
    h1 = min(max(x[1] - 0.5, 0), 1)
    h2 = 1 - h1
    A1 = np.array([[0.9, a], [0, 0.8]])
    A2 = np.transpose(A1)
    return h1 * np.dot(A1, x) + h2 * np.dot(A2, x)

# Define the range of initial conditions
x_min, x_max = -2, 2
y_min, y_max = -2, 2

# Define the step size for the grid
step = 0.1

# Create a grid of initial conditions
x_grid = np.arange(x_min, x_max + step, step)
y_grid = np.arange(y_min, y_max + step, step)

print("Searching for grid (x, y), -2 <= x, y <=2")

# Initialize a dictionary to store starting points for each 'a' value
starting_points_dict = {}

# Iterate over different values of 'a'
for a in np.arange(-1, 1.01, 0.01):
    # Initialize an empty array to store stability results
    stability_results = []

    # Iterate over initial conditions in the grid
    for x0 in x_grid:
        for y0 in y_grid:
            # Set the initial state vector
            x = np.array([x0, y0])

            # Simulate the system for 100 iterations
            for _ in range(100):
                x = system_update(x, a)

            # Check if the state vector is close to the origin
```

```
            if abs(x[0]) < 0.01 and abs(x[1]) < 0.01:
                stability_results.append(True)
            else:
                stability_results.append(False)

    # Determine the overall stability for this value of 'a'
    if sum(stability_results) / len(stability_results) >= 0.9:
    print(f"System is asymptotically stable   for a = {a:.2f}")
    else:
    print(f"System is asymptotically unstable for a = {a:.2f}")
```

Παραλείπουμε πολλές γραμμές που είναι ίδιες για οικονομία χώρου

```
Searching for grid (x, y), -2 <= x, y <=2
System is asymptotically unstable for a = -1.00
System is asymptotically unstable for a = -0.99
System is asymptotically unstable for a = -0.98
System is asymptotically unstable for a = -0.97
System is asymptotically unstable for a = -0.96
...
...
...
System is asymptotically unstable for a = -0.31
System is asymptotically unstable for a = -0.30
System is asymptotically unstable for a = -0.29
System is asymptotically unstable for a = -0.28
System is asymptotically unstable for a = -0.27
System is asymptotically stable   for a = -0.26
System is asymptotically stable   for a = -0.25
System is asymptotically stable   for a = -0.24
System is asymptotically stable   for a = -0.23
System is asymptotically stable   for a = -0.22
...
...
...
System is asymptotically stable   for a = 0.22
System is asymptotically stable   for a = 0.23
System is asymptotically stable   for a = 0.24
System is asymptotically stable   for a = 0.25
System is asymptotically stable   for a = 0.26
System is asymptotically unstable for a = 0.27
System is asymptotically unstable for a = 0.28
System is asymptotically unstable for a = 0.29
System is asymptotically unstable for a = 0.30
System is asymptotically unstable for a = 0.31
...
...
...
System is asymptotically unstable for a = 0.95
System is asymptotically unstable for a = 0.96
```

```
System is asymptotically unstable for a = 0.97
System is asymptotically unstable for a = 0.98
System is asymptotically unstable for a = 0.99
System is asymptotically unstable for a = 1.00
```

Τελικά βλέπουμε ότι για τιμές $a \in [-0.26, 0.26]$ το σύστημα είναι ασυμπτωτικά ευσταθές.