ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Υπολογιστική Νοημοσύνη
2ο Παραδοτέο

Παναγιώτης Καρβουνάρης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Contents

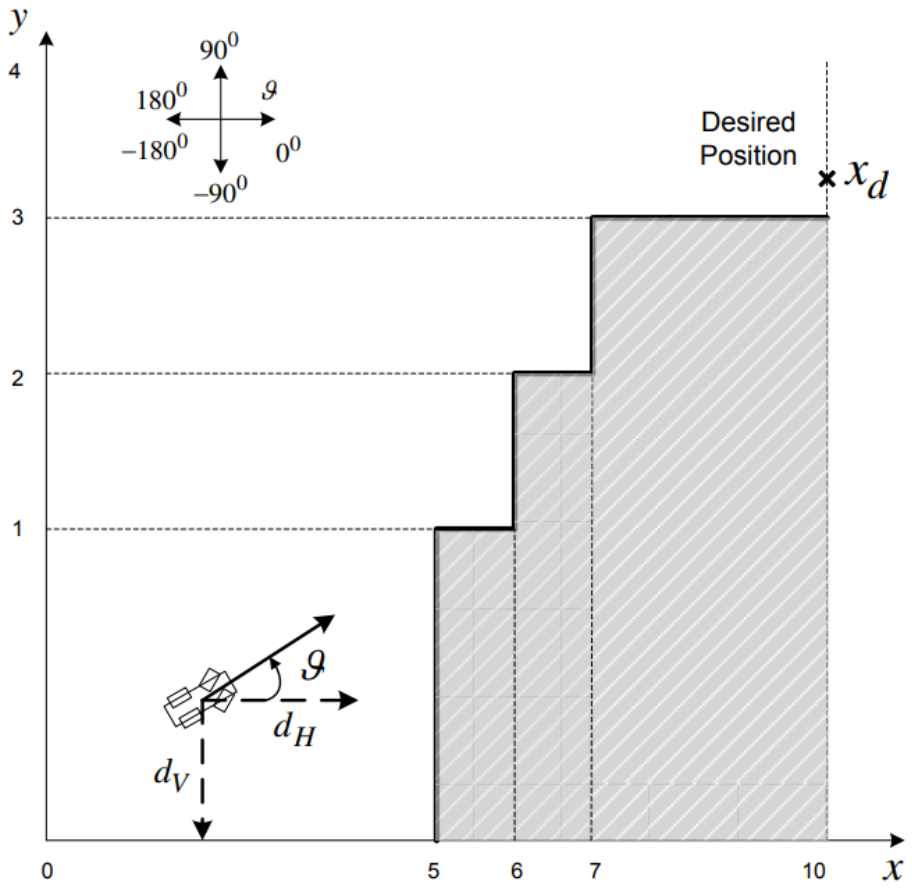


Figure 1: Οπτικοποίηση του προβλήματος.

1 Σχεδίαση FLC

Θα σχεδιάσουμε τον ασαφή ελεγκτή (FLC) που να οδηγεί ένα όχημα, με ταχύτητα $u = 0.05m/sec$ (σταθερή), με ασφάλεια (χωρίς δηλαδή να ακουμπήσει σε σταθερά εμπόδια) από μια αρχική θέση με συντεταγμένες $(xinit, yinit) = (4.1, 0.3)$ στην επιθυμητή θέση $(xd, yd) = (10, 3.2)$. Το όχημα αυτό διαθέτει τους κατάλληλους αισθητήρες ώστε κάθε χρονική στιγμή να υπολογίζει την καθετή (dv) και οριζόντια (dh) απόσταση του από τα εμπόδια αλλά και την διεύθυνση της ταχύτητας του (ϑ). Αξίζει επίσης να σημειώσουμε πως θεωρούμε σημαντικό να έχουμε μικρή απόκλιση στον κατακόρυφο άξονα, δηλαδή το y να είναι αρκετά κοντά στο yd. Στο Figure 1 φαίνεται και μια οπτικοποίηση του προβλήματος.

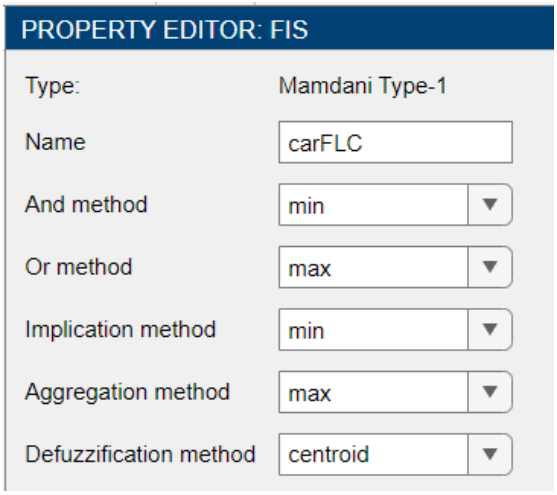


Figure 2: Παράμετροι σχεδίασης του ελεγκτή.

1.1 Υλοποίηση δοσμένων απαιτήσεων

Τα ζητούμενα χαρακτηριστικά του FLC προς σχεδίαση είναι τα εξής:

- Οι κανόνες υλοποιούνται με τον τελεστή συμπερασμού Mamdani, Rc.
- Το συνδετικό ALSO υλοποιείται με τον τελεστή max.
- Σαν τελεστή σύνθεσης χρησιμοποιούμε τον max-min.
- Ο από-ασαφοποιητής υλοποιείται με την τεχνική COA (Center of Area) (ονομάζεται αλλιώς και Center of Gravity (CoG)). Έτσι επιλέγουμε την ρύθμιση centroid για Defuzzification method.

Επιλέγουμε τώρα πάλι να τον φτιάξουμε μέσω του Fuzzy Logic Designer του MATLAB λόγω των ευκολιών και τον αυτοματισμών που παρέχει. Έτσι, οι ρυθμίσεις φαίνονται στο Figure 2.

Είσοδοι του FLC είναι $dv \in [0, 1]$ (m), $dh \in [0, 1]$ (m) και η $\theta \in$[-180° 180°] ενώ η έξοδος είναι η μεταβολή στην διεύθυνση της ταχύτητας $\Delta\theta \in$ [-130° 130°] του οχήματος. Οι λεκτικές μεταβλητές του dv και dh διαμερίζονται σε 5 ασαφή σύνολα (VS: Very Small, S: Small, M: Medium, L: Large, VL: Very Large), η λεκτική μεταβλητή ϑ διαμερίζεται σε 5 ασαφή σύνολα (NL: Negative Large, NS: Negative Small, ZE: Zero, PS: Positive Small, PL: Positive Large) ενώ τέλος η λεκτική μεταβλητή της εξόδου $\Delta\vartheta$ διαμερίζεται σε 5 ασαφή σύνολα (NL: Negative Large, NS: Negative Small, ZE: Zero, PS: Positive Small, PL: Positive Large), όπως φαίνεται στο Figure 3. Τα άκρα και το κέντρο κάθε τριγώνου φαίνεται στο Figure 4. Έτσι, παίρνουμε απευθείας μέσω του Fuzzy Logic Designer και το Figure 5 των συναρτήσεων συμμετοχής για τις τρεις εισόδους και για την έξοδο. Η γενική εικόνα της αρχιτεκτονικής φαίνεται στο Figure 6.

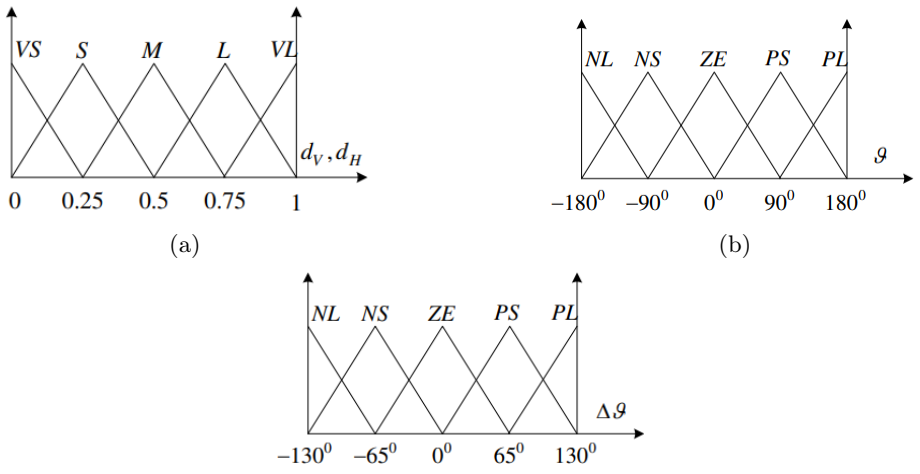


Figure 3: Λεκτικές τιμές των λεκτικών μεταβλητών εισόδου εξόδου.

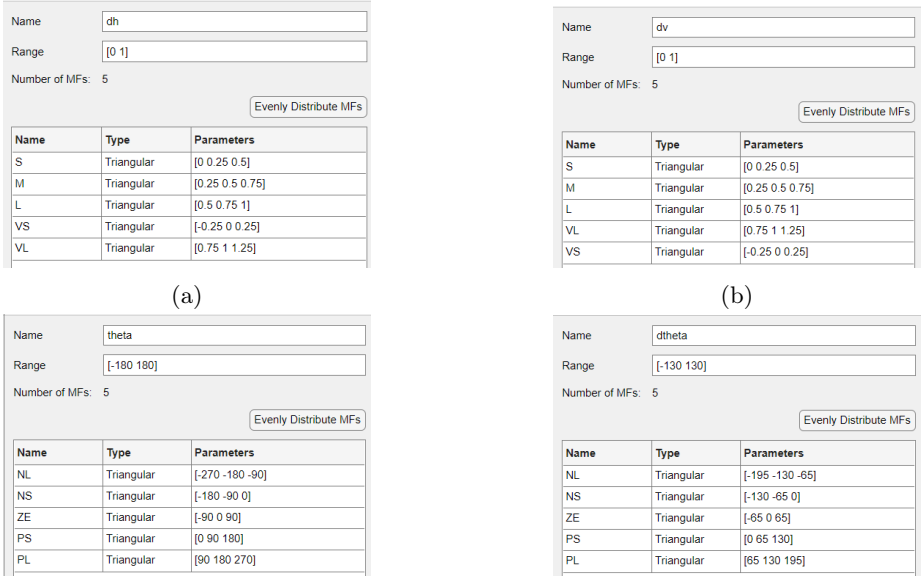


Figure 4: Άκρα και κέντρο τριγώνου κάθε λεκτικής μεταβλητής στο Fuzzy Logic Designer.

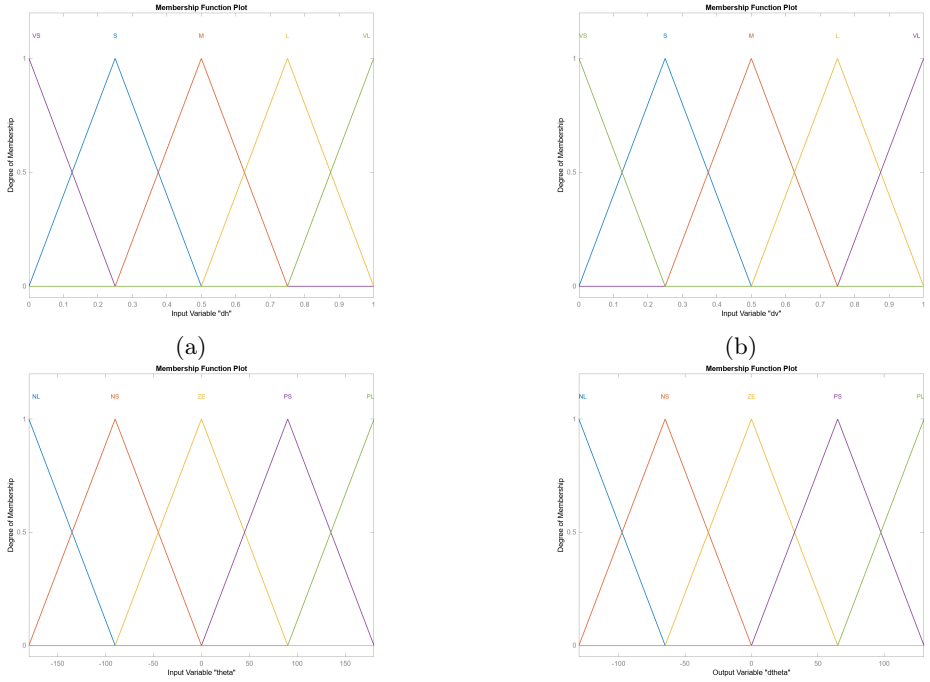


Figure 5: Διαγράμματα των συναρτήσεων συμμετοχής για τις εισόδους και την έξοδο.

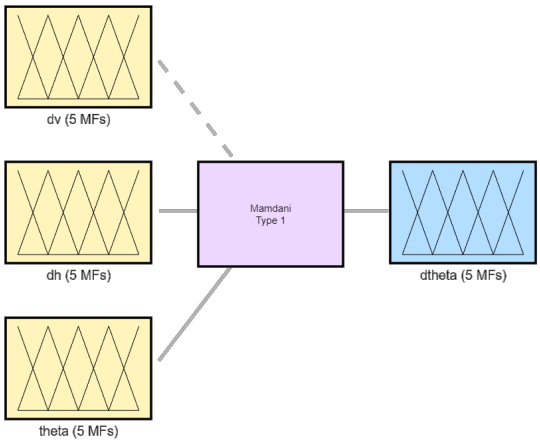


Figure 6: Είσοδοι-έξοδος ασαφούς ελεγκτή (FLC) στο Fuzzy Logic Designer.

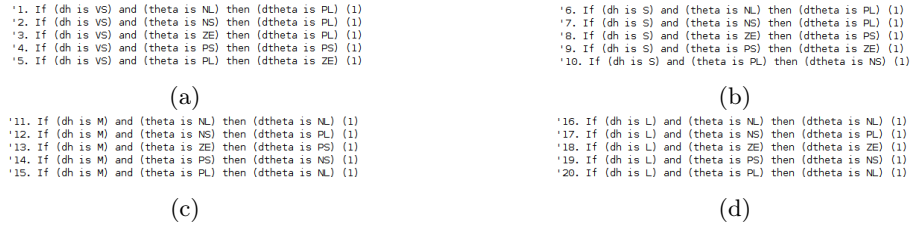


Figure 7: Κανόνες που προέκυψαν από τα παρατηρήσεις σχετικά με το πρόβλημα.

1.2 Υλοποίηση βάσης κανόνων

Το μόνο που απομένει είναι τώρα να καθορίσουμε την βάση κανόνων της μορφής "'If (dv is [.]) and (dh is [.]) and (ϑ is [.]) then ($\Delta\vartheta$ is [.])"', οπού [.] είναι μια λεκτική τιμή από τα ασαφή σύνολα της αντίστοιχής λεκτικής μεταβλητής, του FLC μέσω μίας διαδικασίας τύπου trial and error.

Κανονικά, για μια πλήρη βάση κανόνων θα πρέπει να έχουμε ένα κανόνα για κάθε πιθανό συνδυασμό των εισόδων, δηλαδή για τρεις εισόδους (dv, dh, ϑ) και για μια έξοδο με την κάθε λεκτική μεταβλητή να 3 περιγράφετε από πέντε λεκτικές μεταβλητές θέλουμε $5^3 = 125$ ασαφείς κανόνες. Όμως, εδώ θα ακολουθήσουμε μια διαφορετική λογική δημιουργίας κανόνων. καθώς κάποιοι από τους 125 κανόνες δεν θα χρησιμοποιηθούν ποτέ αφού έχουμε συγκεκριμένο σημείο εκκίνησης και τερματισμού και δεν μας ενδιαφέρει η διαδρομή που θα ακολουθήσουμε καθώς αρκεί να μην «χτυπήσουμε» στα εμπόδια.

Θα μελετήσουμε πρώτα ως προς μόνο την οριζόντια απόσταση (δηλαδή προς το παρόν δεν θα ελέγχουμε για την κατακόρυφη απόσταση dv) από τα εμπόδια καθώς φαίνεται να είναι και η πιο σημαντική αφού η καθετή απόσταση δεν φαίνεται να χρειαστεί διορθώσεις εκτός από ελάχιστα σημεία, όπως είναι αυτά πάνω από το εμπόδιο ή σε γωνίες των εμποδίων. Άρα καταλήγουμε στην σκέψη ότι

- Όταν η οριζόντια απόσταση από το εμπόδιο είναι πολύ μικρή (dh = VS) θέλουμε το όχημα (όπως είναι και λογικό) να στρίψει απότομα προς τα επάνω, για να αποφύγει την σύγκρουση, αλλά και δεξιά, για να δημιουργήσουμε μια φορά προς την επιθυμητή θέση που βρίσκεται πάνω δεξιά ως προς την αρχική θέση. Έτσι προκύπτουν, με την κοινή λογική, οι κανόνες στο Figure 7a, για κάθε συνδυασμό ϑ με σταθερό dh.

- Αντίστοιχα, θέλουμε όταν η οριζόντια απόσταση από το εμπόδιο είναι μικρή (dh = S) το όχημα να στρίψει πάλι προς τα πάνω (όχι όμως τόσο απότομα όσο με πριν) και πάλι δεξιά για να διατηρήσουμε μια τροχιά προς την επιθυμητή θέση που βρίσκεται πάνω δεξιά. Έτσι προκύπτουν ομοίως με πριν οι κανόνες στο Figure 7b.

- Ομοίως, όταν η οριζόντια απόσταση από το εμπόδιο είναι μέτρια (dh = M) θέλουμε το όχημα να δημιουργεί μια τροχιά «σιγά-σιγά» προς τα επάνω δεξιά όπου βρίσκεται και η επιθυμητή τελική θέση αλλά και λίγο προς το

εμπόδιο για να αφήσουμε τους προηγουμένους κανόνες να λειτουργήσουν. Έτσι προκύπτουν οι κανόνες στο Figure 7c.

- Όταν η οριζόντια απόσταση από το εμπόδιο είναι μεγάλη (dh = L) και άρα δεν έχουμε φόβο σύγκρουσης με εμπόδιο θέλουμε το όχημα απλά να διατηρήσει μια πορεία προς τα δεξιά για να πλησιάσει, ως προς τον οριζόντιο άξονα τουλάχιστον, τον επιθυμητό στόχο. Έτσι προκύπτουν οι κανόνες στο Figure 7d.

Στην περίπτωση που έχουμε όμως πολύ μεγάλη οριζόντια απόσταση από το εμπόδιο (dh = VL) οι κανόνες θα είναι λίγο πιο περίπλοκοι ενώ θα λάβουμε εδώ και υπόψη τώρα και την κατακόρυφη απόσταση dv για να συμπεριλάβουμε όλες τις περιπτώσεις. Έτσι

- Αρχικά, θέλουμε όταν το όχημα έχει και από κοντά στα δεξιά του και κοντά από κάτω του εμπόδιο, να μην πλησιάζει πολύ το εμπόδιο (όπου αναγκαστικά θα έχουμε dv = VS). Επομένως θέλουμε να αυξήσουμε ελάχιστα την γωνία της ταχύτητας του οχήματος για να αποφύγουμε το εμπόδιο από κάτω.

- Όταν, όμως, dv = S δεν έχουμε τέτοιο πρόβλημα και μπορούμε να οδηγήσουμε το όχημα απλά προς τα δεξιά και να αφήσουμε τους υπολοίπους παραπάνω κανόνες να «αναλάβουν» και άρα δεν χρειαζόμαστε μεταβολή γωνίας ταχύτητας ($\Delta\vartheta$ = ZE). Όμοια σκέφτομαι και για μεγαλύτερες τιμές της κατακόρυφης απόστασης dv (M, L, VL).

- Όταν φτάνουμε στο σημείο λίγο πριν την πάνω επιφάνεια του εμποδίου για την αποφυγή «κολλήματος» στο οριζόντιο εμπόδιο θέλουμε δίνουμε μια μικρή θετική μεταβολή στην κλίση της ταχύτητας (άρα $\Delta\vartheta$ = PS).

- Αν όμως μπει με οριζόντια κατεύθυνση το όχημα στο πάνω μέρος του εμποδίου τότε δεν χρειάζεται να μεταβάλουμε την γωνία της ταχύτητας (άρα $\Delta\vartheta$ = ZE) καθώς θέλουμε απλά να διατηρήσει την πορεία του.

Με τους επιπλέον κανόνες που προσθέσαμε καταλήγουμε στη λίστα κανόνων του Figure 8.

Ο ελεγκτής με αυτή την βάση κανόνων και τις συναρτήσεις συμμετοχής είναι στο MATLAB αρχείο ασαφούς ελεγκτή carFLC.fis ενώ η απόδειξη λειτουργίας τους θα φανεί παρακάτω.

1.3 Προσομοίωση προβλήματος / εύρεση πορείας

Πριν εξηγήσουμε την λογική του script προσομοίωσης του προβλήματος, δημιουργούμε την συνάρτηση που προσομοιώνει την λειτουργία των αισθητήρων του οχήματος [dh, dv] = getSensorDistances(x, y). Η συνάρτηση αυτή δέχεται ως είσοδο τις τωρινές συντεταγμένες του οχήματος (x, y) και επιστρέφει στην έξοδο την κατακόρυφη και οριζόντια απόσταση από το εμπόδια μπροστά του (dv, dh). Η λογική της συνάρτησης είναι πως ελέγχει σε πιο «επίπεδο» του εμποδίου βρίσκεται και μέσω απλών if ελέγχων υπολογίζει την διαφορά των συντεταγμένων από

```
'1. If (dh is VS) and (theta is NL) then (dtheta is PL) (1)
'2. If (dh is VS) and (theta is NS) then (dtheta is PL) (1)
'3. If (dh is VS) and (theta is ZE) then (dtheta is PL) (1)
'4. If (dh is VS) and (theta is PS) then (dtheta is PS) (1)
'5. If (dh is VS) and (theta is PL) then (dtheta is ZE) (1)
'6. If (dh is S) and (theta is NL) then (dtheta is PL) (1)
'7. If (dh is S) and (theta is NS) then (dtheta is PL) (1)
'8. If (dh is S) and (theta is ZE) then (dtheta is PS) (1)
'9. If (dh is S) and (theta is PS) then (dtheta is ZE) (1)
'10. If (dh is S) and (theta is PL) then (dtheta is NS) (1)
'11. If (dh is M) and (theta is NL) then (dtheta is NL) (1)
'12. If (dh is M) and (theta is NS) then (dtheta is PL) (1)
'13. If (dh is M) and (theta is ZE) then (dtheta is PS) (1)
'14. If (dh is M) and (theta is PS) then (dtheta is NS) (1)
'15. If (dh is M) and (theta is PL) then (dtheta is NL) (1)
'16. If (dh is L) and (theta is NL) then (dtheta is NL) (1)
'17. If (dh is L) and (theta is NS) then (dtheta is PL) (1)
'18. If (dh is L) and (theta is ZE) then (dtheta is ZE) (1)
'19. If (dh is L) and (theta is PS) then (dtheta is NS) (1)
'20. If (dh is L) and (theta is PL) then (dtheta is NL) (1)
'21. If (dh is VL) and (theta is NL) then (dtheta is NL) (1)
'22. If (dh is VL) and (theta is NS) then (dtheta is PS) (1)
'23. If (dh is VL) and (theta is PS) then (dtheta is NS) (1)
'24. If (dh is VL) and (theta is PL) then (dtheta is NL) (1)
'25. If (dv is VS) and (dh is VL) and (theta is ZE) then (dtheta is PS) (1)
'26. If (dv is S) and (dh is VL) and (theta is ZE) then (dtheta is ZE) (1)
'27. If (dv is M) and (dh is VL) and (theta is ZE) then (dtheta is ZE) (1)
'28. If (dv is VL) and (dh is VL) and (theta is ZE) then (dtheta is PS) (1)
```

Figure 8: Πλήρης λίστα ασαφών κανόνων.

```
while (pdist([x(end) y(end)], [x_final y_final]) > errorTolerance)
    …
end
```

Figure 9: While loop παράδειγμα κώδικα.

το εμπόδιο και επιστρέφει τις κατάλληλες αποστάσεις προβαλλόμενες στο εύρος του πεδίου ορισμού τους, δηλαδή το [0, 1] (m) με χρήση της min(). Η συνάρτηση αυτή βρίσκεται για χρήση στο MATLAB function αρχείο getSensorDistances.m .

Για την προσομοίωση της διαδικασίας ελέγχου της αποφυγής εμποδίων του οχήματος δημιουργούμε μια while-loop η οποία θα «τρέχει» μέχρι το όχημα να φτάσει στον στόχο του. Μέχρι να φτάσει, σε κάθε βήμα παίρνουμε τις μετρήσεις του αισθητήρα για την κατακόρυφη και οριζόντια απόσταση του οχήματος από τα εμπόδια με την χρήση της MATLAB συνάρτησης getSensorDistances(). Στην συνέχεια βρίσκουμε το νέο σημείο που θα βρεθεί το όχημα με βάση την κινηματική ανάλυση ξεχωριστά για κάθε άξονα, όπως γνωρίζουμε από την φυσική. Πιο συγκεκριμένα, αν το όχημα βρίσκεται σε μια θέση (x_i, y_i) με σταθερή ταχύτητα u (με γωνιά ταχύτητας ίση με θ_i), τότε η επόμενη θέση του θα είναι η (x_{i+1}, y_{i+1}), όπου

$$x_{i+1} = x_i + ucos(\theta_i) \tag{1}$$

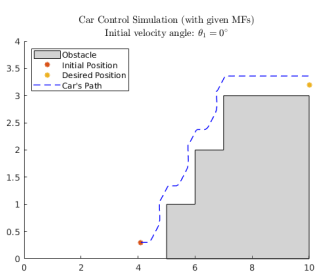
$$y_{i+1} = y_i + usin(\theta_i) \tag{2}$$

Αφού λοιπόν κάνουμε τον παραπάνω υπολογισμό του νέου σημείο και το αποθηκεύσουμε στα διανύσματα της πορείας του οχήματος x και y, υπολογίζουμε το ποσό θα πρέπει να μεταβληθεί η γωνιά ϑ κατά $\Delta\vartheta$ μέσω του FLC και την χρήση της MATLAB συνάρτησης evalfis() με είσοδο τις τωρινές αποστάσεις από τα εμπόδια και την τωρινή γωνιά θ_i και παίρνοντας στην έξοδο την απαιτούμενη μεταβολή της γωνίας $d\theta_i$. Έτσι, στο επόμενο βήμα θα έχουμε την νέα γωνία θ_{i+1} για την οποία θα ισχύει με βάση τον ορισμό της παραγώγου

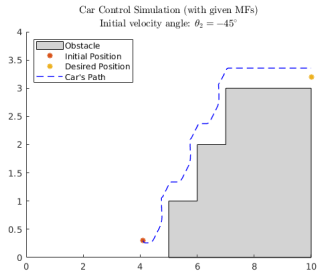
$$\theta_{i+1} = \theta_i + d\theta \tag{3}$$

Έτσι, ακολουθούμε την παραπάνω διαδικασία συνεχώς μέχρι να φτάσουμε στον επιθυμητό μας στόχο και στην συνέχεια δημιουργούμε τα plots προσομοίωσης με χρήση των διανυσμάτων πορείας x και y.

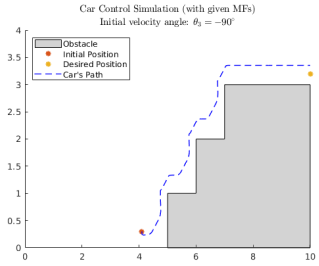
Παραπάνω αναφέρθηκε η χρήση μιας loop μέχρι να φτάσει το όχημα στον τελικό στόχο. Για την διευκόλυνση της εκτέλεσης του MATLAB προγράμματος, ορίσαμε μια ανοχή/tolerance για το τελικό σφάλμα θέσης του οχήματος κατά το οποίο θεωρούμε πως το όχημα έχει πέτυχει τον στόχο του. Για την αυστηρότητα της διαδικασίας, ορίσαμε αυτό το error tolerance ίσο με 0.03 [m] . Ο υπολογισμός της απόστασης από τον τελικό στόχο και άρα και του σφάλματος θέσης γίνεται μέσω της in-built MATLAB συνάρτησης pdist() που επιστρέφει την ευκλείδεια απόσταση δυο σημείων στον χώρο και έτσι κάνουμε πριν από κάθε βήμα τον έλεγχο μέσα στην while-loop όπως φαίνεται στο Figure 9, οπού [x_final y_final] οι συντεταγμένες του τελικό σημείου/στόχου και [x(end) y(end)] οι τωρινές συντεταγμένες του οχήματος.



(a) $\theta_1 = 0°$



(b) θ_2 = -45°



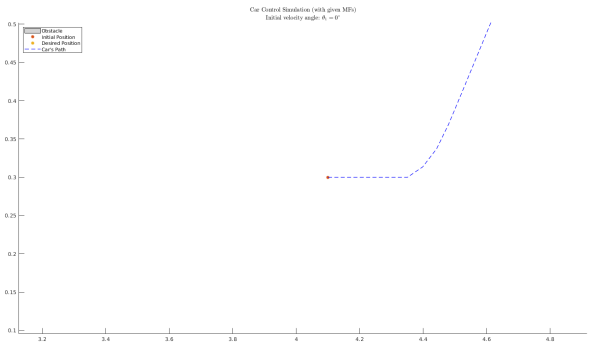
(c) θ_3 = -90°

Figure 10: Προσομοίωση ελέγχου οχήματος με FLC (με δοσμένες συναρτήσεις συμμετοχής) για αρχική γωνία ταχύτητας ϑ.

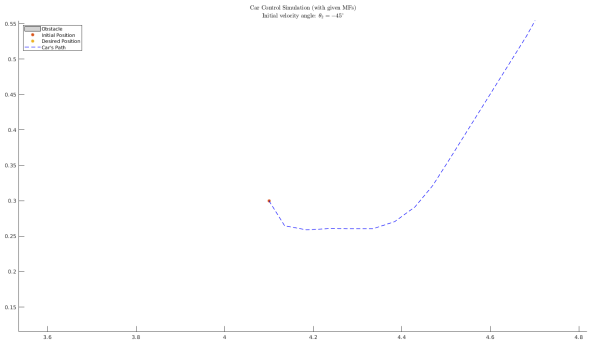
Αξίζει επίσης να σημειωθεί πως και σε κάθε βήμα κάνουμε τους κατάλληλους ελέγχους για τον τερματισμό του προγράμματος σε περίπτωση που το όχημα χτυπήσει πάνω στα εμπόδια ή έχει ξεφύγει κατά πολύ από τον τελικό στόχο (αποφεύγοντας έτσι ένα infinite loop) δίνοντας στην κονσόλα του MATLAB τα κατάλληλα μηνύματα, δηλαδή "OUT OF BOUNDS. Runtime stopped..." αν ξεφύγαμε κατά πολύ από τον στόχο και εκτός κάποιον ορίων και "Car hit at the obstacle" αν το όχημα χτύπησε πάνω στο εμπόδιο.

Έτσι τώρα παρακάτω θα δώσουμε τις πορείες που θα ακολουθήσει το όχημα για να φτάσει στην επιθυμητή θέση με την χρήση του FLC που περιγράφηκε παραπάνω και τις αρχικές παραμέτρους των συναρτήσεων συμμετοχής για κάθε μια από τις απαιτούμενες αρχικές διευθύνσεις a) $\theta_1 = 0°$, b) θ_2 = -45° και c) θ_3 = -90°, Figure 10. Ακόμη, παραθέτουμε πάλι τα ιδιά διαγράμματα, zoomed-in στην αρχή της κάθε πορείας για να δείξουμε την σωστή λειτουργία του κώδικα για είσοδο την κάθε ζητούμενη αρχική διεύθυνση, Figure 11.

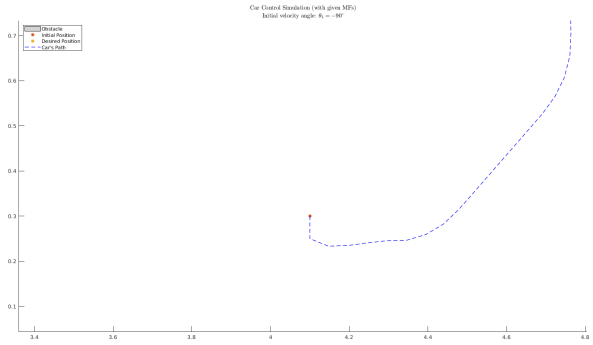
Και στις τρεις παραπάνω προσομοιώσεις η κονσόλα του MATLAB μας έδωσε το μήνυμα "OUT OF BOUNDS. Runtime stopped..." . Άρα προφανώς το όχημα δεν βρέθηκε σε απόσταση μικρότερη ή ίση από το error tolerance (0.03 [m]) από τον τελικό στόχο για να τερματίσει. Όπως φαίνεται και ευκολά από το Figure 10, ενώ το ο ασαφής ελεγκτής φαίνεται να χειρίζεται με σωστό τρόπο το όχημα μέχρι να φτάσει πάνω από το εμπόδιο, στο τέλος φαίνεται το όχημα να πηγαίνει προς το τελικό επιθυμητό σημείο του αλλά τελικά να το «προσπερνάει» από πάνω.



(a) $\theta_1 = 0°$

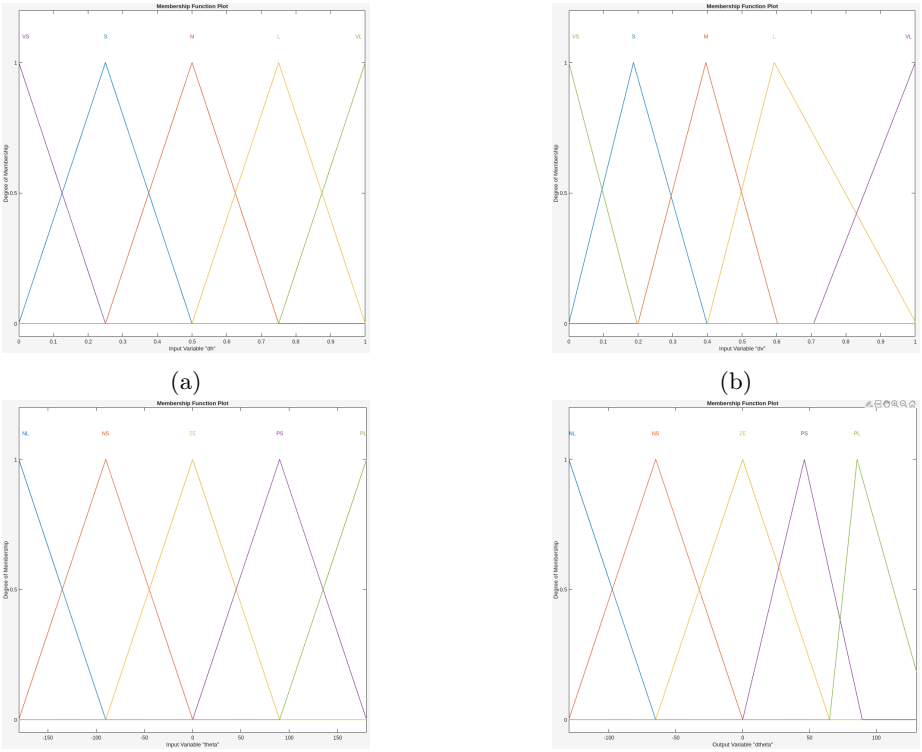


(b) θ_2 = -45°



(c) θ_3 = -90°

Figure 11: Προσομοίωση ελέγχου οχήματος με FLC (με δοσμένες συναρτήσεις συμμετοχής) για αρχική γωνία ταχύτητας ϑ.



(a) (b)

(c) (d)

Figure 12: Διαγράμματα των συναρτήσεων συμμετοχής για τις εισόδους και την έξοδο.

Συνεπώς μπορούμε να κάνουμε λόγο για overshooting αλλά και πως επιλέξαμε σωστά τους κανόνες της βάσης του FLC.

Πρέπει συνεπώς να μειώσουμε το μεγάλο αυτό σφάλμα που υπάρχει στον κατακόρυφο άξονα για να φέρουμε το όχημα στην επιθυμητή θέση.

Για την μείωση λοιπόν τώρα των σφαλμάτων, την σωστή λειτουργία του FLC και την επίτευξη του στόχου του οχήματος θα αλλάξουμε τις παραμέτρους των συναρτήσεων συμμετοχής ενώ θα ελέγξουμε και μήπως η αλλαγή του πεδίου ορισμού της μεταβλητής εξόδου $d\theta$ είναι απαραίτητη ή μειώνει το τελικο σφάλμα θέση.

1.4 Αλλαγή παραμέτρων συναρτήσεων συμμετοχής

Εφόσον είχαμε πρόβλημα απόκλισης ως προς τον κατακόρυφο άξονα, πρέπει να αλλάξουμε την συνάρτηση συμμετοχής του dv (και σίγουρα της εξόδου $\Delta\vartheta$) μικραίνοντας το εύρος τιμών των ασαφών τιμών για μεγάλες αλλαγές (L, VL, PL κ.λ.π.) και να φτάσουμε έτσι πιο προσεκτικά, με μικρές μεταβολές δηλαδή, στην επιθυμητή έξοδο. Επίσης, δεν κρίθηκε και απαραίτητη η αλλαγη του πεδίου

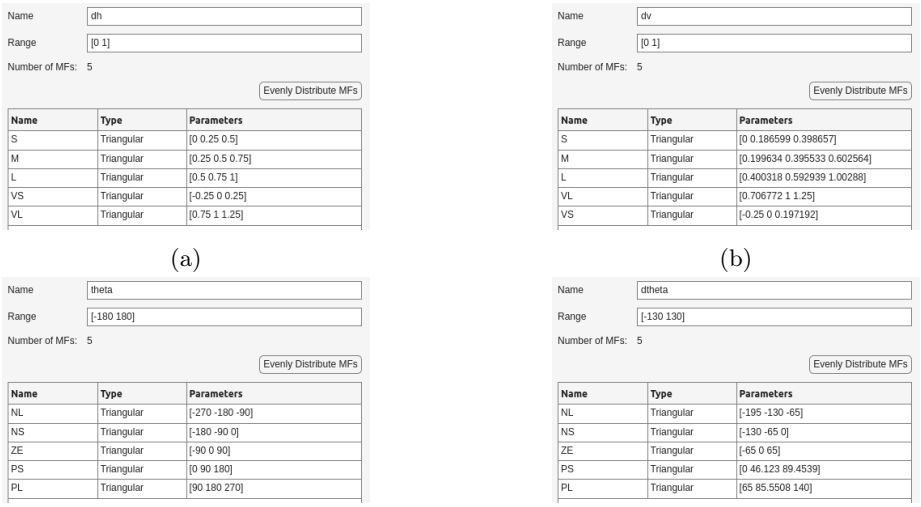


Figure 13: Άκρα και κέντρο τριγώνου κάθε λεκτικής μεταβλητής στο Fuzzy Logic Designer με βάση τις καινούργιες membership functions.

ορισμού της μεταβλητής εξόδου $d\theta$. Αξίζει να σημειωθεί πως οι νέες συναρτήσεις συμμετοχής Figure 12 προέκυψαν με την μέθοδο trial and error καθαρά πειραματικά. Οι αντίστοιχες χαρακτηριστηκές θέσεις των κορυφών και των άκρων φαίνονται στο Figure 13.

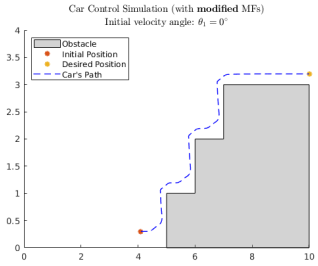
Ο ελεγκτής με την βάση κανόνων, αλλά με τις νέες συναρτήσεις συμμετοχής είναι στο MATLAB αρχείο ασαφούς ελεγκτή carFLC_modified.fis.

1.5 Προσομοίωση προβλήματος / εύρεση πορείας, νέος ελεγκτής

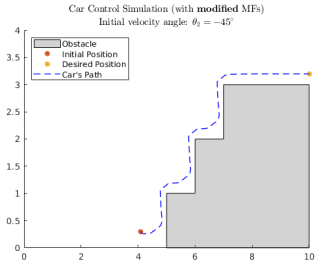
Τώρα θα δώσουμε τις νέες πορείες που θα ακολουθήσει το όχημα για να φτάσει στην επιθυμητή θέση με την χρήση του FLC με βάση τις νέες παραμέτρους των συναρτήσεων συμμετοχής που επιλέξαμε. Για κάθε μια από τις αρχικές διευθύνσεις a) $\theta_1 = 0°$, b) $\theta_2 =$ -45° και c) $\theta_3 =$ -90°. Τα αποτελέσματα φαίνονται στο Figure 14.

'Οπως φαίνεται και από τα διαγράμματα του σχήματος 11 αλλά και από την έξοδο της κονσόλας του script στο MATLAB, το όχημα φτάνει, σε κάθε περίπτωση αρχικής γωνίας ταχύτητας, επιτυχώς και με ασφάλεια στο επιθυμητό σημείο και το τελικο σφάλμα μειώθηκε δραστικά, Table 1.

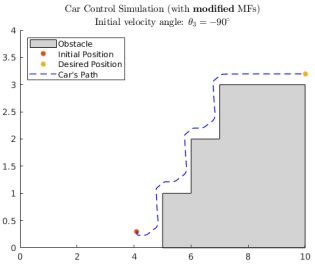
λες οι προσομοιώσεις, τόσο αυτές με τον αρχικό FLC (δηλαδή αυτόν με τις δοσμένες συναρτήσεις συμμετοχής) όσο και με τον τελικο FLC (δηλαδή αυτόν με τις αλλαγμένες/νέες συναρτήσεις συμμετοχής), τα διαγράμματα τους και οι έξοδοι γίνονται στο MATLAB script CarControl_Simulations.m.



(a) $\theta_1 = 0^\circ$



(b) θ_2 = -45°



(c) θ_3 = -90°

Figure 14: Προσομοίωση ελέγχου οχήματος με FLC (με δοσμένες συναρτήσεις συμμετοχής) για αρχική γωνία ταχύτητας ϑ.

ΑΡΧΙΚΗ ΓΩΝΙΑ ΤΑΧΥΤΗΤΑΣ	**ΤΕΛΙΚΟ ΣΦΑΛΜΑ ΘΕΣΗΣ**
0°	0.004116 [m]
-45°	0.004207 [m]
-90°	0.003944 [m]

Table 1: Τελικά σφάλματα απόκλισης από επιθυμητή θέση για κάθε αρχική γωνία ταχύτητας θ με χρήση του FLC με νέες συναρτήσεις συμμετοχής.

2 Σχολιασμός

Όπως φάνηκε, για την επίλυση του παραπάνω ασαφούς προβλήματος αποφυγής εμποδίου ήταν απαραίτητη η βαθιά κατανόηση του προβλήματος καθώς και η εφαρμογή μεθόδων τύπου trial and error. Χρειάστηκε να σκεφτούμε τι θα έπρεπε να κάνει και πως να σκεφτεί στην καθημερινή ζωή ένας οδηγός για τον χειρισμό ενός αντιστοίχου οχήματος για την αποφυγή ενός τέτοιου εμποδίου για να καλύψουμε όλες τις πιθανότητες αποφυγής σύγκρουσης και την δημιουργία μιας σωστής βάσης κανόνων για τον ασαφή ελεγκτή.

Αρά η λειτουργία του παραπάνω ελεγκτή δεν είναι καθολική για κάθε είδους εμπόδια και πολύ πιθανώς να δουλεύει μόνο για το συγκεκριμένο πρόβλημα, δηλαδή το συγκεκριμένο εμπόδιο που δόθηκε αλλά και το συγκεκριμένο αρχικό και τελικό/επιθυμητό σημείο του οχήματος.

Επίσης, είναι προφανές πως η παραπάνω λύση δεν είναι μοναδική και σιγουρά δεν είναι η βέλτιστη! Μπορούμε μάλιστα να δημιουργήσουμε ελεγκτές που να λαμβάνουν υπόψη και άλλους παράγοντες, όπως η ελαχιστοποίηση της απόστασης/πορείας μέχρι να φτάσει το όχημα στο επιθυμητό σημείο.