Ασαφή Συστήματα Υπολογιστική Νοημοσύνη

Εργασία 2-C_CarControl

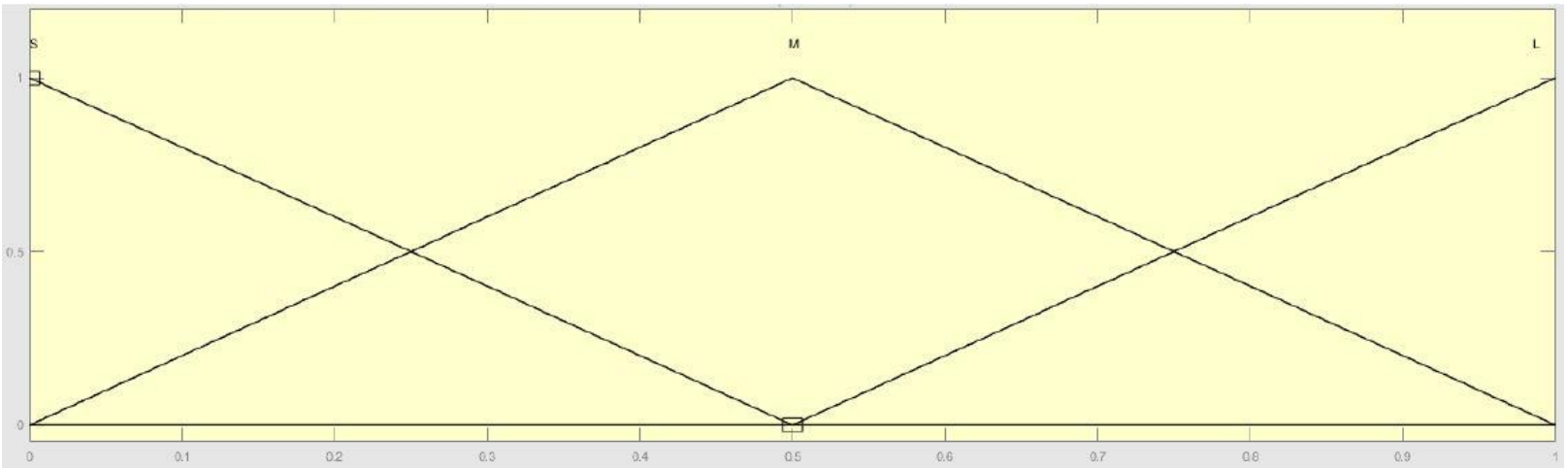
Παναγιώτης Σαββίδης
8094
11.7.2021

Σκοπός της εργασίας αυτής είναι ο σχεδιασμός ενός ασαφούς ελεγκτή (FLC) για τον έλεγχο της κίνησης ενός οχήματος με σκοπό την αποφυγή εμποδίων.

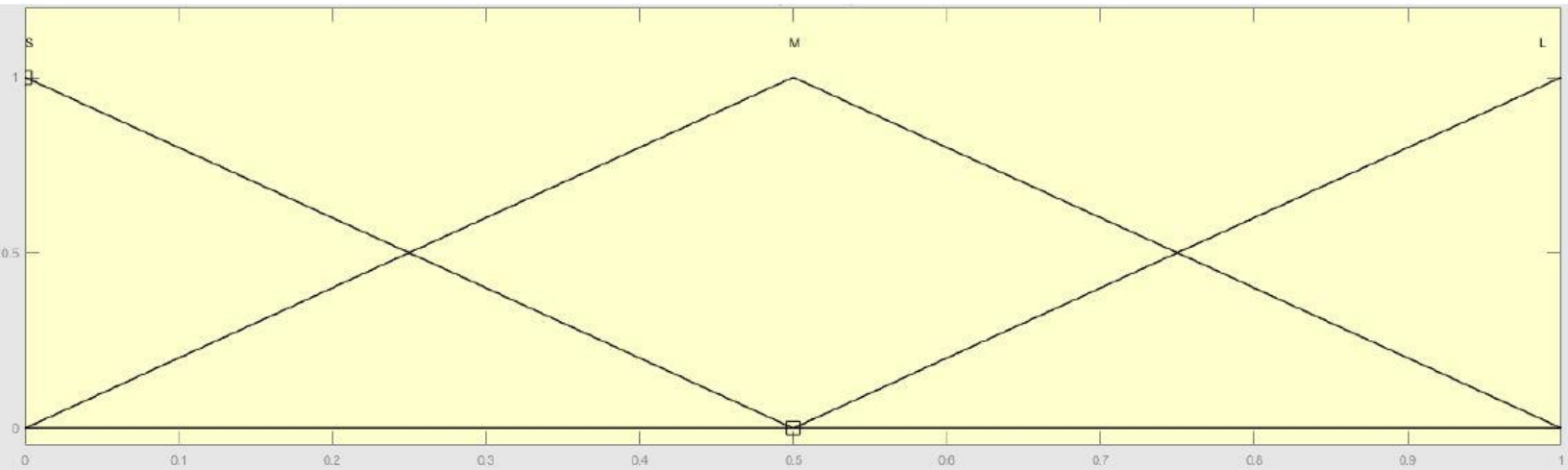
Για την υλοποίηση του παραπάνω προβλήματος θα χρησιμοποιήσουμε ένα αρχείο Fuzzy Logic Designer με όνομα Car_C_final.fis , στο οποίο υλοποίησα τις συναρτήσεις συμμετοχής των εισόδων και της εξόδου, μαζί με τους κανόνες του ασαφή ελεγκτή. Οι αρχικοί παράμετροι των συναρτήσεων καθώς και το αρχικό Δθ φαίνονται στο Car_C_1st.fis οι οποίες άλλαξαν για να επιτύχουμε το θεμιτό αποτέλεσμα.

Membership Functions

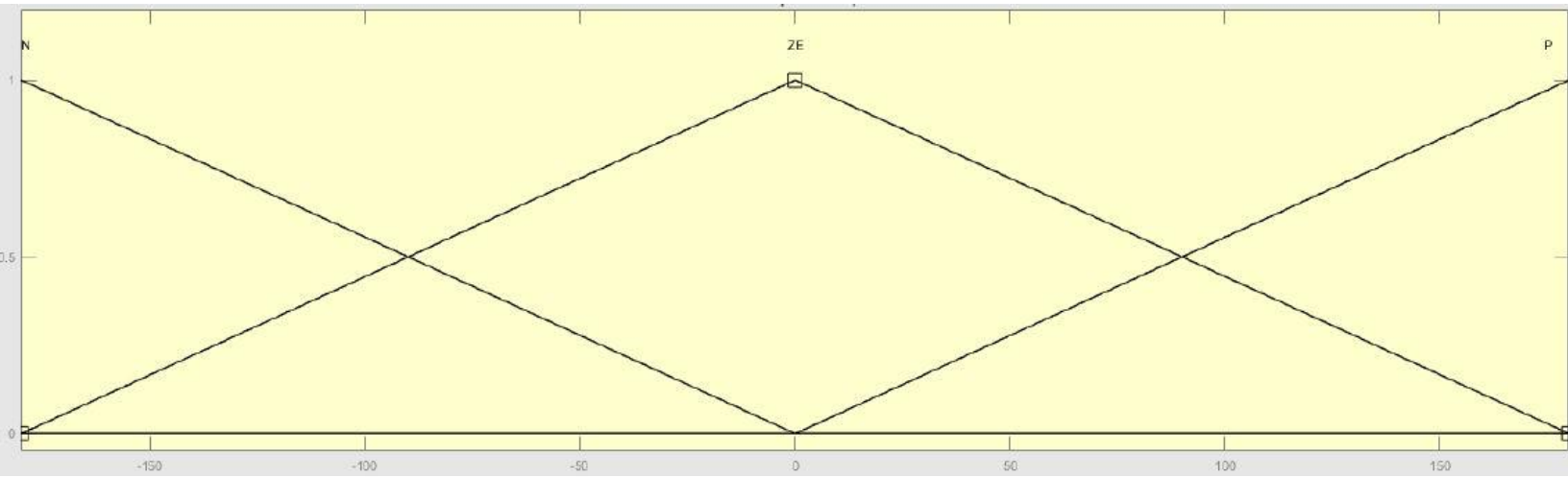
Input 1- d_H



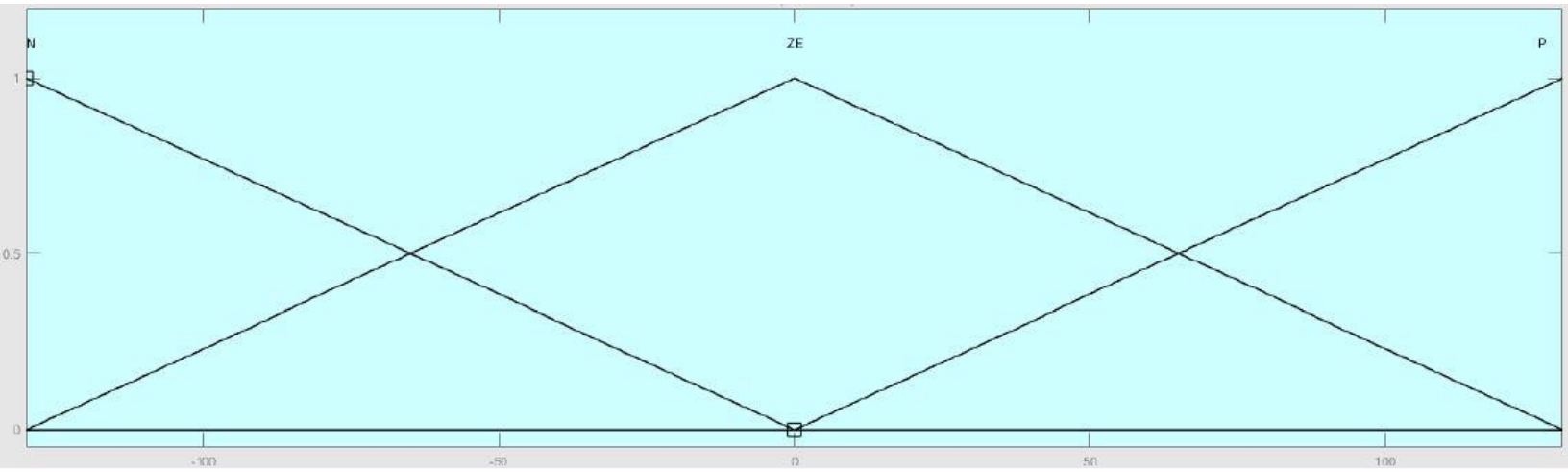
Input 2 - d_V



Input 3 – θ



Output 1 – Δθ



Rules

Οι κανόνες του ασαφούς ελεγκτή μας έχει την μορφή

$$IF\ \ d_V\ is\ S\ \ AND\ \ d_H\ \ is\ \ S\ \ AND\ \ \vartheta\ is\ \ N\ \ \ THEN\ \ \Delta\vartheta\ \ is\ \ P$$

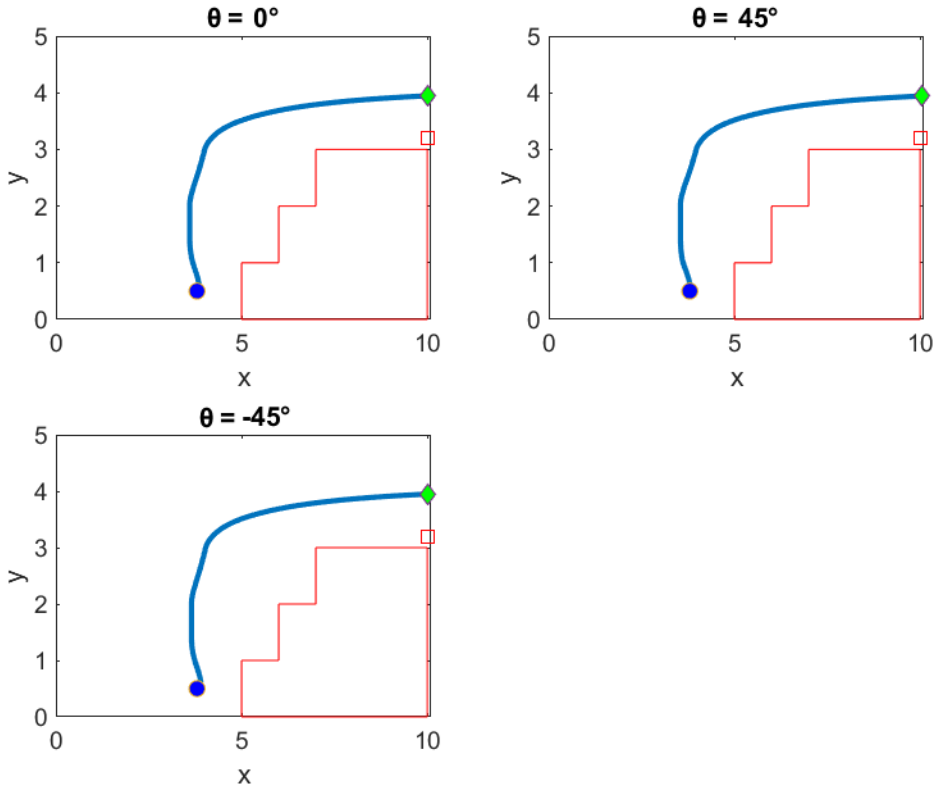
Γνωρίζουμε το αρχικό σημείο αλλά και τον προορισμό του οχήματος. Έτσι γίνεται εύκολα κατανοητή η πορεία που πρέπει να διανύσει το όχημα μας για να φτάσει στον προορισμό και με την βοήθεια της παραπάνω μορφής των κανόνων οι αρχικοί μας κανόνες είναι οι παρακάτω (όπως είναι γραμμένοι στο fis αρχείο):

```
[System]
Name='Car_C_1st'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='dh'
Range=[0 1]
NumMFs=3
MF1='S':'trimf',[-0.5 0 0.5]
MF2='M':'trimf',[0 0.5 1]
MF3='L':'trimf',[0.5 1 1.5]

[Input2]
Name='dv'
Range=[0 1]
NumMFs=3
MF1='S':'trimf',[-0.5 0 0.5]
MF2='M':'trimf',[0 0.5 1]
MF3='L':'trimf',[0.5 1 1.5]

[Input3]
Name='theta'
Range=[-180 180]
NumMFs=3
MF1='N':'trimf',[-360 -180 0]
MF2='P':'trimf',[0 180 360]
MF3='ZE':'trimf',[-180 0 180]

[Output1]
Name='Dtheta'
Range=[-130 130]
NumMFs=3
MF1='N':'trimf',[-260 -130 0]
MF2='ZE':'trimf',[-130 0 130]
MF3='P':'trimf',[0 130 240]
```

```
[Rules]
1 1 1, 3 (1) : 1
1 2 1, 3 (1) : 1
1 3 1, 3 (1) : 1
2 1 1, 3 (1) : 1
2 2 1, 3 (1) : 1
2 3 1, 3 (1) : 1
3 1 1, 3 (1) : 1
3 2 1, 3 (1) : 1
3 3 1, 3 (1) : 1
1 1 3, 3 (1) : 1
1 2 3, 3 (1) : 1
1 3 3, 3 (1) : 1
2 1 3, 2 (1) : 1
2 2 3, 2 (1) : 1
2 3 3, 2 (1) : 1
3 1 3, 2 (1) : 1
3 2 3, 2 (1) : 1
3 3 3, 2 (1) : 1
1 1 2, 2 (1) : 1
1 2 2, 1 (1) : 1
1 3 2, 1 (1) : 1
2 1 2, 1 (1) : 1
2 2 2, 1 (1) : 1
2 3 2, 1 (1) : 1
3 1 2, 1 (1) : 1
3 2 2, 1 (1) : 1
3 3 2, 1 (1) : 1
```

Trial and Error

Όμως με το αρχικό σύνολο κανόνων το όχημα δεν φτάνει στον προορισμό του, όπως φαίνεται στις παρακάτω εικόνες.



Τα λάθη που έγιναν στους αρχικούς κανόνες μπορούν εύκολα να διορθωθούν. Αρχικά βλέπουμε ότι η αρχική γωνία του οχήματος μπορεί να βελτιωθεί ελάχιστα για να βελτιωθεί η αρχική του πορεία και να μπορεί να καταλήγει στον προορισμό. Αλλάζοντας δηλαδή την αρχική τιμή της λεκτικής τιμής του μηδέν (ZE) της θ κάθε γωνία θα θεωρείται περισσότερο μη μηδενική και έτσι επηρεάζεται το Δθ. Επιπλέον μέχρι να φτάσει στο 2ο εμπόδιο (χ=6) η Δθ θα έπρεπε να αλλάζει με μεγαλύτερη συσχέτιση. Τέλος πρέπει για ψ>3 και χ>7 η γωνία του οχήματος να συνεχίζει να μειώνεται για να φτάσει στον προορισμό του.

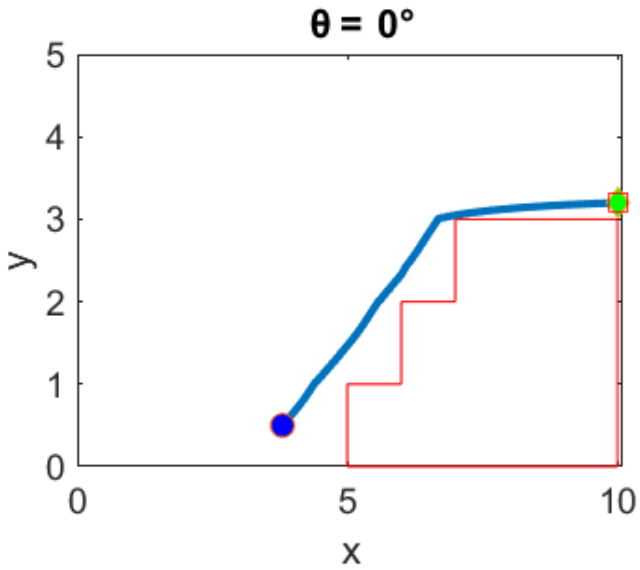
Με την κλασσική μέθοδο του Trial and Error κατέληξα στους παρακάτω τελικούς κανόνες (όπως βρίσκονται στο Car_C_final.fis) που όπως θα φανεί στην συνέχεια οδηγούν το αυτοκίνητο στον προορισμό του. Η μέθοδος που ακολούθησα ήταν να βάλω ένα βάρος (με ποσοστό επί τοις %) σε κάθε κανόνα.

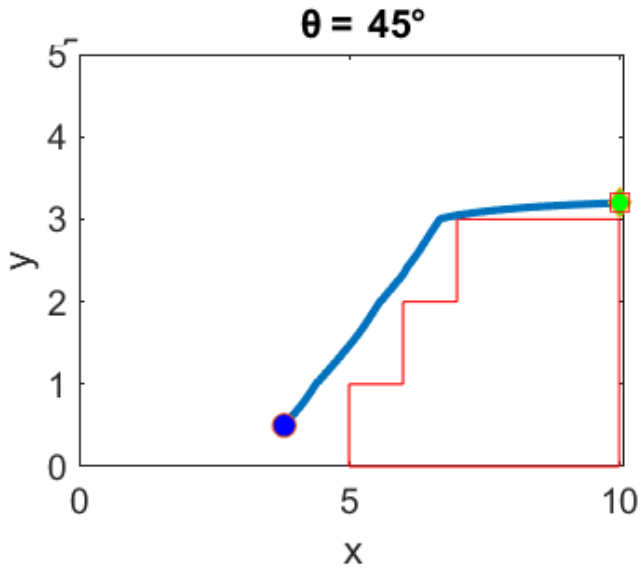
Success

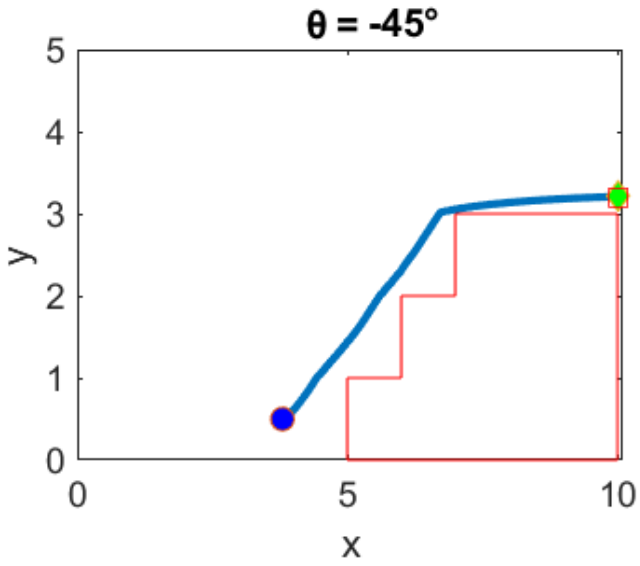
Μέσα σε κόκκινο πλαίσιο φαίνονται οι αλλαγές που έγιναν.

```
[System]
Name='Car_C_final'
Type='mamdani'
Version=2.0
NumInputs=3
NumOutputs=1
NumRules=27
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='dh'
Range=[0 1]
NumMFs=3
MF1='S':'trimf',[-0.5 0 0.5]
MF2='M':'trimf',[0 0.5 1]
MF3='L':'trimf',[0.5 1 1.5]

[Input2]
Name='dv'
Range=[0 1]
NumMFs=3
MF1='S':'trimf',[-0.5 0 0.5]
MF2='M':'trimf',[0 0.5 1]
MF3='L':'trimf',[0.5 1 1.5]

[Input3]
Name='theta'
Range=[-180 180]
NumMFs=3
MF1='N':'trimf',[-360 -180 0]
MF2='P':'trimf',[0 180 360]
MF3='ZE':'trimf',[-60 0 60]

[Output1]
Name='Dtheta'
Range=[-130 130]
NumMFs=3
MF1='N':'trimf',[-260 -130 0]
MF2='ZE':'trimf',[-130 0 130]
MF3='P':'trimf',[0 130 240]
```

```
[Rules]
1 1 1, 3 (1) : 1
1 2 1, 3 (1) : 1
1 3 1, 3 (1) : 1
2 1 1, 3 (1) : 1
2 2 1, 3 (0.65) : 1
2 3 1, 3 (0.35) : 1
3 1 1, 3 (1) : 1
3 2 1, 3 (1) : 1
3 3 1, 3 (0.65) : 1
1 1 3, 3 (0.35) : 1
1 2 3, 3 (0.35) : 1
1 3 3, 3 (0.35) : 1
2 1 3, 2 (0.35) : 1
2 2 3, 2 (0.35) : 1
2 3 3, 2 (0.35) : 1
3 1 3, 2 (0.35) : 1
3 2 3, 2 (0.35) : 1
3 3 3, 2 (0.35) : 1
1 1 2, 2 (0.35) : 1
1 2 2, 1 (0.35) : 1
1 3 2, 1 (0.35) : 1
2 1 2, 1 (0.35) : 1
2 2 2, 1 (0.65) : 1
2 3 2, 1 (0.65) : 1
3 1 2, 1 (1) : 1
3 2 2, 1 (1) : 1
3 3 2, 1 (1) : 1
```







Και όπως ανέφερα προηγουμένως, το όχημα βρίσκει τον δρόμο του και καταλήγει στον τελικό προορισμό μας οπότε δεν υπάρχει κάποια άλλη αλλαγή που χρειάζεται να κάνουμε στου κανόνες μας. Βλέπουμε ότι και για τις 3 αρχικές μας αρχικές διευθύνσεις η πορεία που ακολουθεί το όχημα είναι σχεδόν ίδια. Αυτό οφείλεται στις αλλαγές που κάναμε στα βάρη, οι οποίες επηρεάζουν κυρίως τις τιμές εισόδου 2 και 3.
Στο πρώτο σετ κανόνων το όχημα αποτύγχανε περισσότερο να βρει τον στόχο του κατά dv παρά dh ή με απλά λόγια, το όχημα δεν ήταν πολύ ψηλότερα από την θέση που θα έπρεπε να είναι, αλλά ήταν πολύ πιο αριστερά από εκεί π θα έπρεπε να είναι. Επίσης η γωνία που θα έπρεπε να έχει κάθε στιγμή ήταν πολύ πιο λάθος. Γι' αυτό οι περισσότεροι κανόνες που αλλάξαμε

περιλαμβάνουν περισσότερες αλλαγές στις εισόδους δύο (19 αλλαγές) και τρία (20 αλλαγές) παρά στην είσοδο ένα (14 αλλαγές). Αυτή ήταν και η λογική που οδήγησε στην λύση.