ΤΟΜΕΑΣ ΣΗΜΑΤΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ,

ΡΟΜΠΟΤΙΚΗΣ ΚΑΙ ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ

4Η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ ΣΤΟ ΜΑΘΗΜΑ

“ΝΕΥΡΟ-ΑΣΑΦΗΣ ΕΛΕΓΧΟΣ ΚΑΙ ΕΦΑΡΜΟΓΕΣ”

ΜΠΕΚΡΗΣ ΔΗΜΗΤΡΙΟΣ

ΑΜ:03117116

Ασαφή Συστήματα και Έλεγχος

Άσκηση 1:

Για τη σχεδίαση του ασαφούς ελεγκτή, ακολουθήσαμε την μεθοδολογία που έχουμε διδαχθεί. Αρχικά, κανονικοποιούμε την είσοδο(control state). Επειδή, τα state variables αντικατοπτρίζουν ποσοστά, οι διαφορικές του δεδομένου συστήματος λύθηκαν μέχρις ότου να τα x_1, x_2 , να πάρουν τιμές ακραίες τιμές 0 και 1 αντίστοιχα.

Αυτό επιτυγχάνεται, καθώς τα α,β επιλέγονται πάντα θετικά και επομένως πάντα το ποσοστό των μη-μολυσμένων αυξάνεται. Επομένως, η είσοδος ανήκει στο σύνολο [0,1].

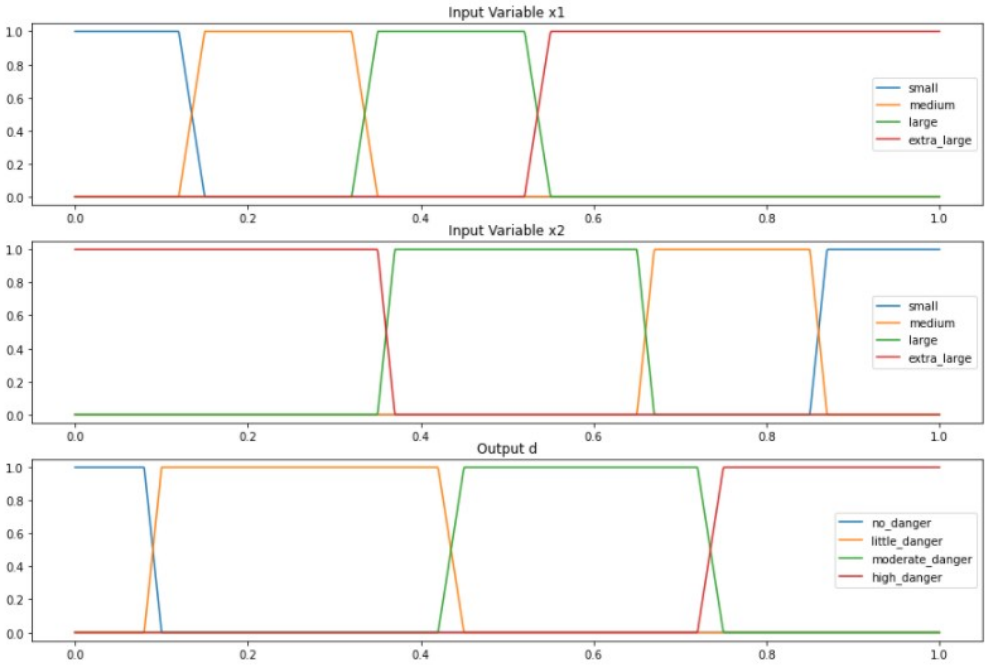
Στη συνέχεια, διαχωρίζουμε τα διαστήματα των linguistic variables για τις 2 μεταβλητές και την έξοδο.

```
#Create fuzzy set for x1
self.x1['small'] = fuzz.trapmf(self.x1.universe, [0,0,0.12,0.15])
self.x1['medium'] = fuzz.trapmf(self.x1.universe, [0.12,0.15,0.32,0.35])
self.x1['large'] = fuzz.trapmf(self.x1.universe, [0.32,0.35,0.52,0.55])
self.x1['extra_large'] = fuzz.trapmf(self.x1.universe, [0.52,0.55,1,1])

#Create fuzzy set for x2
self.x2['small'] = fuzz.trapmf(self.x2.universe, [0.85,0.87,1,1])
self.x2['medium'] = fuzz.trapmf(self.x2.universe, [0.65,0.67,0.85,0.87])
self.x2['large'] =fuzz.trapmf(self.x2.universe, [0.35,0.37,0.65,0.67])
self.x2['extra_large'] = fuzz.trapmf(self.x2.universe, [0,0,0.35,0.37])

#Create fuzzx2 set for control input d
self.d['no_danger'] = fuzz.trapmf(self.d.universe, [0,0,0.08,0.1])
self.d['little_danger'] = fuzz.trapmf(self.d.universe, [0.08,0.1,0.42,0.45])
self.d['moderate_danger'] =fuzz.trapmf(self.d.universe, [0.42,0.45,0.72,0.75])
self.d['high_danger'] = fuzz.trapmf(self.d.universe, [0.72,0.75,1,1])
```

Επιλέχθηκαν τραπεζοειδείς συναρτήσεις μετοχής. Επίσης, ορίστηκαν οι εξής κανόνες:

```
#Create fuzzx2 rules
#General Rules:
rule0 = ctrl.Rule(antecedent=(self.x1['small'] & self.x2['small']),
                              consequent=self.d['no_danger'], label='rule no danger 0')
rule1 = ctrl.Rule(antecedent=(self.x1['small'] & self.x2['medium']),
                              consequent=self.d['no_danger'], label='rule no danger 1')
rule2 = ctrl.Rule(antecedent=(self.x1['small'] & self.x2['large']),
                              consequent=self.d['little_danger'], label='rule little danger 2')
rule3 = ctrl.Rule(antecedent=(self.x1['small'] & self.x2['extra_large']),
                              consequent=self.d['moderate_danger'], label='rule moderate danger 3')
rule4 = ctrl.Rule(antecedent=(self.x1['medium'] & self.x2['small']),
                              consequent=self.d['no_danger'], label='rule no danger 4')
rule5 = ctrl.Rule(antecedent=(self.x1['medium'] & self.x2['medium']),
                              consequent=self.d['moderate_danger'], label='rule moderate danger 5')
rule6 = ctrl.Rule(antecedent=(self.x1['medium'] & self.x2['large']),
                              consequent=self.d['moderate_danger'], label='rule moderate danger 6')
rule7 = ctrl.Rule(antecedent=(self.x1['medium'] & self.x2['extra_large']),
                              consequent=self.d['high_danger'], label='rule high danger 7')
rule8 = ctrl.Rule(antecedent=(self.x1['large'] & self.x2['small']),
                              consequent=self.d['no_danger'], label='rule no danger 8')
rule9 = ctrl.Rule(antecedent=(self.x1['large'] & self.x2['medium']),
                              consequent=self.d['moderate_danger'], label='rule moderate danger 9')
rule10 = ctrl.Rule(antecedent=(self.x1['large'] & self.x2['large']),
                              consequent=self.d['high_danger'], label='rule high danger 10')
rule11 = ctrl.Rule(antecedent=(self.x1['large'] & self.x2['extra_large']),
                              consequent=self.d['high_danger'], label='rule high danger 11')
rule12 = ctrl.Rule(antecedent=(self.x1['extra_large'] & self.x2['small']),
                              consequent=self.d['little_danger'], label='rule little danger 12')
rule13 = ctrl.Rule(antecedent=(self.x1['extra_large'] & self.x2['medium']),
                              consequent=self.d['moderate_danger'], label='rule moderate danger 13')
rule14 = ctrl.Rule(antecedent=(self.x1['extra_large'] & self.x2['large']),
                              consequent=self.d['high_danger'], label='rule high danger 14')
rule15 = ctrl.Rule(antecedent=(self.x1['extra_large'] & self.x2['extra_large']),
                              consequent=self.d['high_danger'], label='rule high danger 15')
```

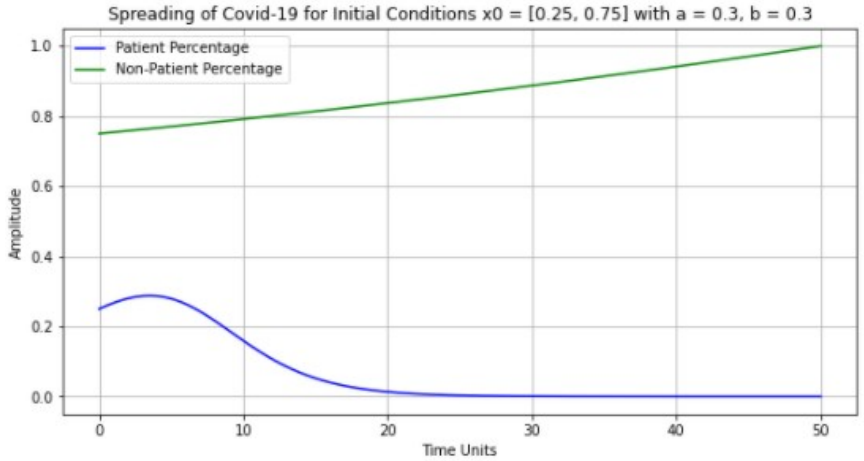
Παρακάτω, παρατίθενται οι συναρτήσεις μετοχής σε διάγραμμα για την καλύτερη κατανόηση του διαμερισμού του διαστήματος [0,1].

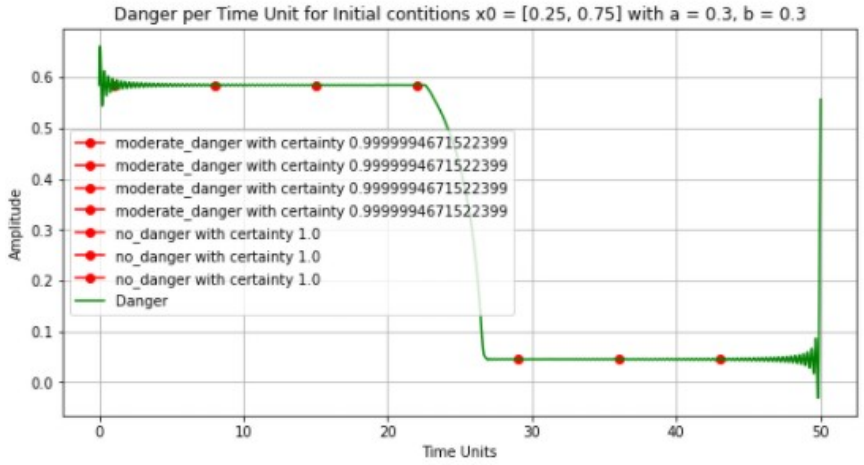


Σχόλιο:

1. Παρατηρούμε, επικάλυψη 50%, η οποία προτείνεται για καλύτερη απόδοση του ελεγκτή.
2. Οι κανόνες καλύπτουν όλο το εύρος τιμών.

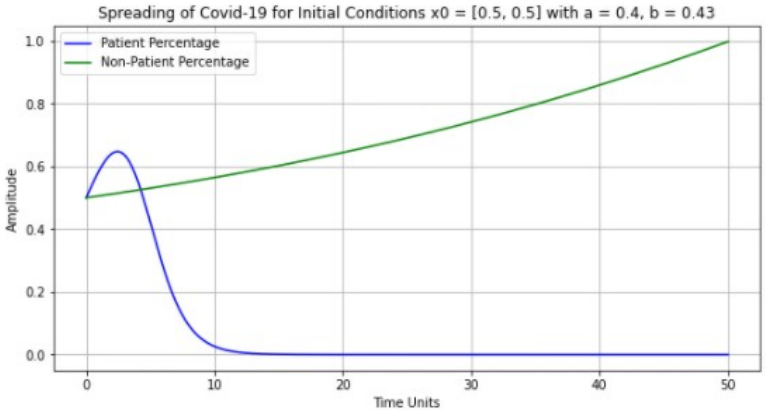
Τέλος, ο παραπάνω ελεγκτής δοκιμάστηκε για μια γκάμα αρχικών συνθηκών και τιμών α,β. Τα αποτελέσματα είναι τα εξής.

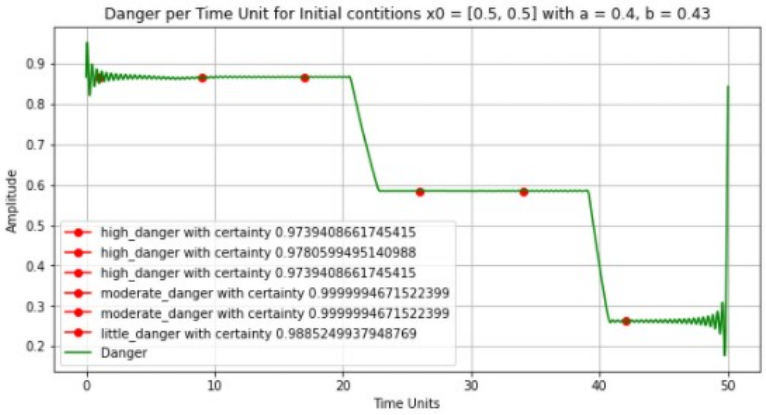


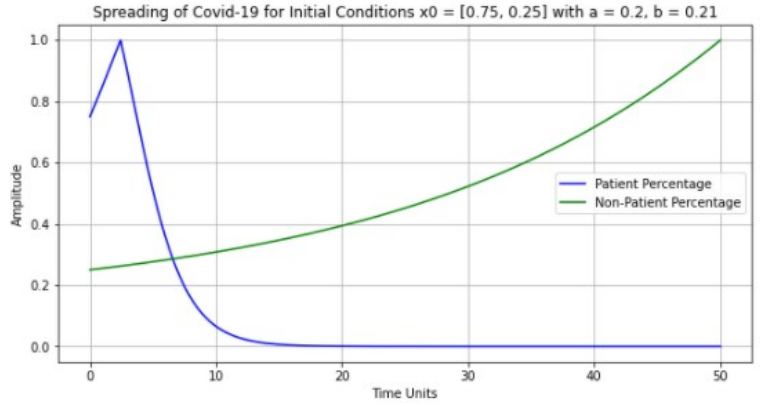


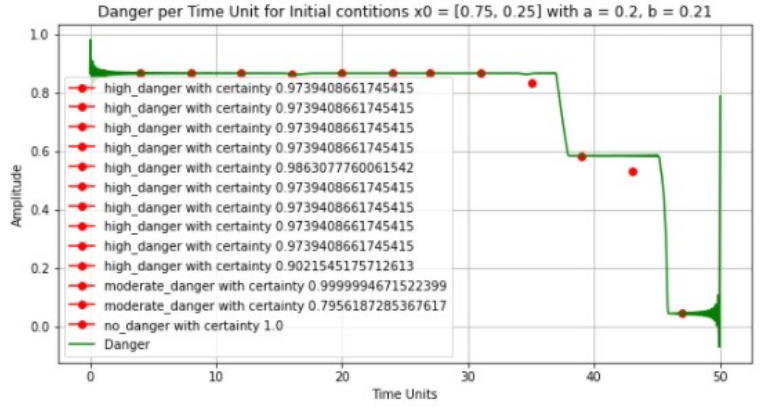
Σχόλια:

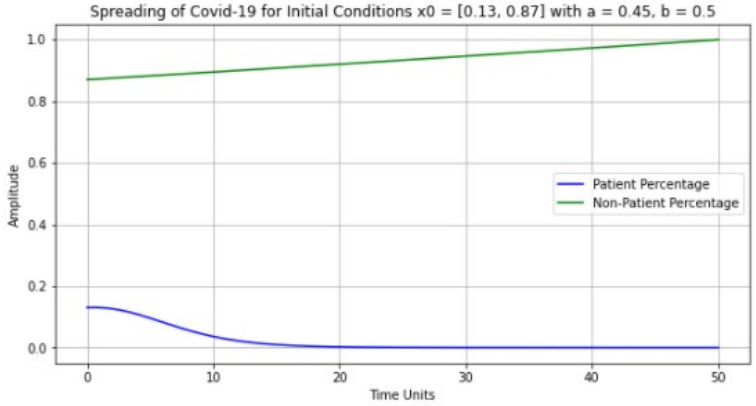
1. Μπορεί να δει κανείς, ότι οι linguistic περιγραφές συνάδουν με την απόκριση του αντίστοιχου συστήματος, καθώς για παράδειγμα στις πρώτες χρονικές μονάδες ο κίνδυνος είναι περίπου στη μέση, το οποίο είναι αναμενόμενο αφού, το ποσοστό μη μολυσμένων ατόμων είναι υψηλό ενώ των μολυσμένων χαμηλό.

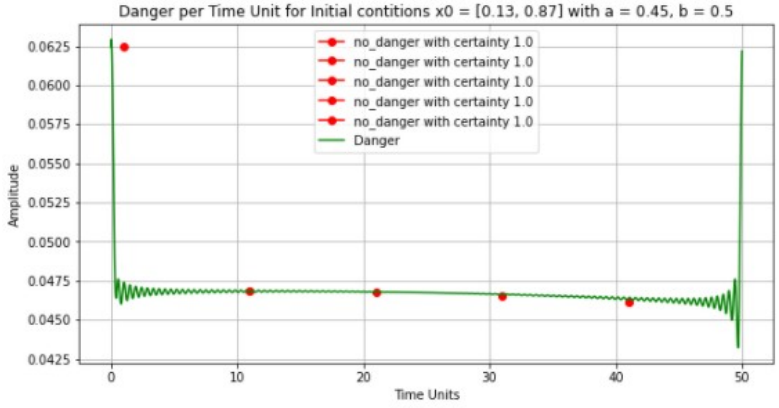

Spreading of Covid-19 for Initial Conditions x0 = [0.5, 0.5] with a = 0.4, b = 0.43
Patient Percentage
Non-Patient Percentage
Amplitude
Time Units

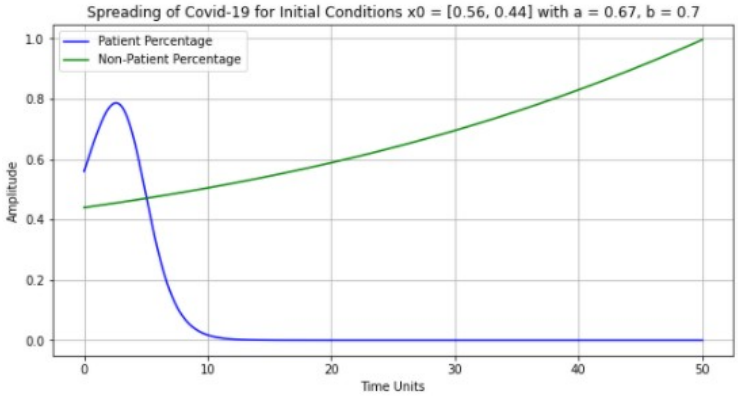

Danger per Time Unit for Initial contitions x0 = [0.5, 0.5] with a = 0.4, b = 0.43
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9780599495140988
high_danger with certainty 0.9739408661745415
moderate_danger with certainty 0.9999994671522399
moderate_danger with certainty 0.9999994671522399
little_danger with certainty 0.9885249937948769
Danger
Amplitude
Time Units

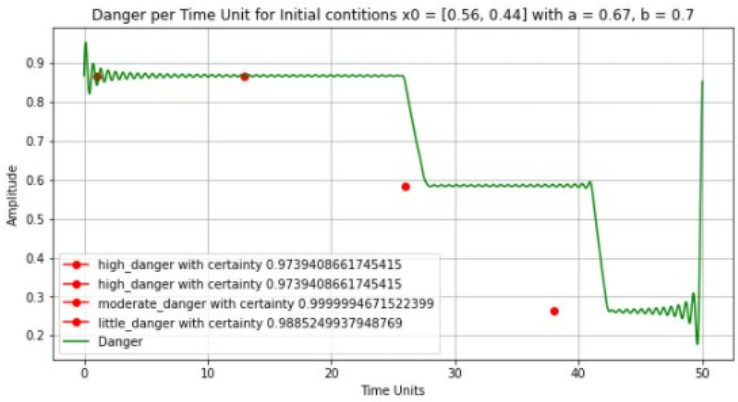

Spreading of Covid-19 for Initial Conditions x0 = [0.75, 0.25] with a = 0.2, b = 0.21
Patient Percentage
Non-Patient Percentage
Amplitude
Time Units


Danger per Time Unit for Initial contitions x0 = [0.75, 0.25] with a = 0.2, b = 0.21
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9863077760061542
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9021545175712613
moderate_danger with certainty 0.9999994671522399
moderate_danger with certainty 0.7956187285367617
no_danger with certainty 1.0
Danger
Amplitude
Time Units


Spreading of Covid-19 for Initial Conditions x0 = [0.13, 0.87] with a = 0.45, b = 0.5
Patient Percentage
Non-Patient Percentage
Amplitude
Time Units


Danger per Time Unit for Initial contitions x0 = [0.13, 0.87] with a = 0.45, b = 0.5
no_danger with certainty 1.0
no_danger with certainty 1.0
no_danger with certainty 1.0
no_danger with certainty 1.0
no_danger with certainty 1.0
Danger
Amplitude
Time Units


Spreading of Covid-19 for Initial Conditions x0 = [0.56, 0.44] with a = 0.67, b = 0.7
Patient Percentage
Non-Patient Percentage
Amplitude
Time Units


Danger per Time Unit for Initial contitions x0 = [0.56, 0.44] with a = 0.67, b = 0.7
high_danger with certainty 0.9739408661745415
high_danger with certainty 0.9739408661745415
moderate_danger with certainty 0.9999994671522399
little_danger with certainty 0.9885249937948769
Danger
Amplitude
Time Units

Σχόλια:

1. Για την εξαγωγή, της περιγραφής εργαστήκαμε ως εξής:
   - Δίνουμε σαν είσοδο x_1, x_2 και λαμβάνουμε την αντίστοιχη έξοδο, η οποία προκύπτει από τους κανόνες που ορίσαμε και το συνδυασμό των αντίστοιχων firings του καθένα.
   - Υπολογίζουμε την απόσταση της εξόδου από το κέντρο του κάθε κανόνα και κρατάμε την μικρότερη. Ο αντίστοιχος κανόνας μας δίνει την περιγραφή, που αναλογεί.
2. Για τον υπολογισμό της βεβαιότητας βρήκαμε το κοντινότερο κέντρο μάζας έστω d_1 από όλους τους κανόνες και το 2ο κοντινότερο έστω d_2 . Ύστερα, πήραμε το πηλίκο αυτών $\lambda = \frac{d_1}{d_2}, \lambda \leq 1, αφού d_1 \leq d_2$ και το αφαιρέσαμε από τη μονάδα $\alpha = 1 - \lambda, 0 \leq \alpha \leq 1$. Ακολούθησα αυτή τη λογική, καθώς σκέφτηκα ότι, μας αφορά η αβεβαιότητα, την οποία θα μπορούσε να πει κανείς ότι, είναι ο λόγος του φαινομενικά "σωστού" αποτελέσματος από του 2ου καλύτερου αποτελέσματος. Η αφαίρεση από τη μονάδα δίνει το δείκτη βεβαιότητας. Παρατίθεται το αντίστοιχο κομμάτι του κώδικα:

```
def find_perc(dist1, dist2, danger):
    res = abs(dist1/(dist2))
    return res
```

```
from scipy import signal
time = 50
Ts = 0.01
centers = fs.get_centers()
percentages = []
a, b = [.3, .4, .2, .45, .67], [.3, .43, .21, .5, .7]
x0 = [[.25, .75], [.5, .5], [.75, .25], [.13, .87], [.56, .44]]
t = np.linspace(0, time, int(time//Ts)+1)
offset  = 0
for i in range(len(x0)):

    danger = simulate(x0[i], a[i], b[i])
    samples = []
    percentages = []
    for j in range(0, len(danger), 50):
        samples.append(j)
        dist1 = np.inf
        dist2 = np.inf
        linguistic = 0
        for center in centers.keys():
            if dist1 > abs(danger[j] - centers[center]):
                dist2 = dist1
                dist1 = abs(danger[j] - centers[center])
                linguistic = center

        percentages.append([linguistic, find_perc(dist1, dist2, danger[j])])


    fig = plt.figure(figsize=(10,5))
    for j in range(len(percentages)):
        plt.plot(int(samples[j]//len(percentages))+1,danger[samples[j]],marker="o",color="red" ,label=percentages[j][0]+' with ce
    plt.plot(t, signal.resample(danger, int(time//Ts+1)), 'g', label='Danger')
    plt.legend(loc='best')
    plt.xlabel('Time Units')
    plt.ylabel('Amplitude')
    plt.grid()
    plt.title(f'Danger per Time Unit for Initial contitions x0 = [{str(x0[i][0])}, {str(x0[i][1])}] with a = {a[i]}, b = {b[i]}')
    plt.show()
```

Τέλος, παρατηρεί κανείς, ότι ο παραπάνω ελεγκτής λειτουργεί ικανοποιητικά και μας προειδοποιεί για τον κίνδυνο με σχετικά καλή ακρίβεια.

Άσκηση 3:

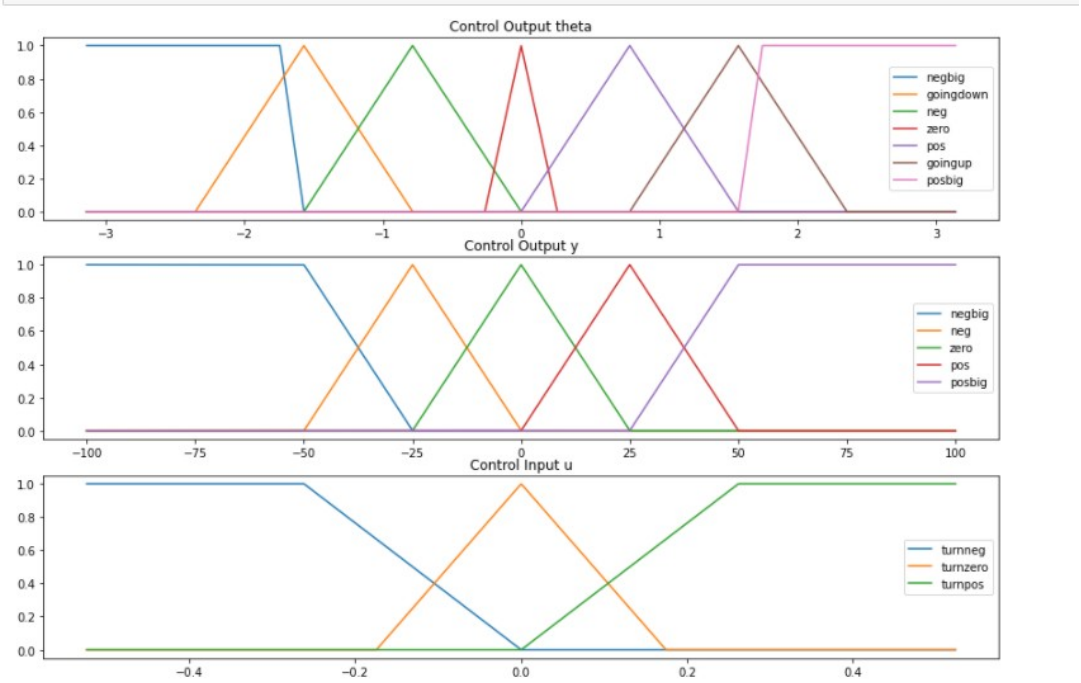
Εργαστήκαμε όμοια με την Άσκηση 1, με τη μόνη διαφορά ότι, δοκιμάσαμε να δώσουμε τις τιμές των μεταβλητών σε μη κανονικοποιημένο σύνολο αλλά στο [-π, π], [-100, 100] αντίστοιχα. Τα αποτελέσματα παρ' όλ' αυτά είναι ικανοποιητικά.

Στη συνέχεια, παρουσιάζουμε την διαμέριση των διαστημάτων για κάθε μία από τις μεταβλητές και το control input.

```
#Create fuzzy set for theta
self.theta['negbig'] = fuzz.trapmf(self.theta.universe, [-np.pi, -np.pi, -3*np.pi/4, -np.pi/2])
self.theta['goingdown'] = fuzz.trimf(self.theta.universe, [-3*np.pi/4, -np.pi/2, -np.pi/4])
self.theta['neg'] = fuzz.trimf(self.theta.universe, [-np.pi/2, -np.pi/4, 0])
self.theta['zero'] = fuzz.trimf(self.theta.universe, [-np.pi/12, 0, np.pi/12])
self.theta['pos'] = fuzz.trimf(self.theta.universe, [0, np.pi/4, np.pi/2])
self.theta['goingup'] = fuzz.trimf(self.theta.universe, [np.pi/4, np.pi/2, 3*np.pi/4])
self.theta['posbig'] = fuzz.trapmf(self.theta.universe, [np.pi/2, 3*np.pi/4, np.pi, np.pi])

#Create fuzzy set for y
self.y['negbig'] = fuzz.trapmf(self.y.universe, [-100, -100, -50, -25])
self.y['neg'] = fuzz.trimf(self.y.universe, [-50, -25, 0])
self.y['zero'] =fuzz.trimf(self.y.universe, [-25, 0, 25])
self.y['pos'] = fuzz.trimf(self.y.universe, [0 , 25 , 50])
self.y['posbig'] = fuzz.trapmf(self.y.universe, [25, 50, 100, 100])

#Create fuzzx2 set for control input u
self.u['turnneg'] = fuzz.trapmf(self.u.universe, [-np.pi/6, -np.pi/6, -np.pi/12, 0])
self.u['turnzero'] = fuzz.trimf(self.u.universe, [-np.pi/18, 0, np.pi/18])
self.u['turnpos'] =fuzz.trapmf(self.u.universe, [0, np.pi/12, np.pi/6, np.pi/6])
```

Η εντολές εξόδου, αναφέρονται κατά πόσο συνίσταται ο οδηγός να στρίψει το τιμόνι και με προς ποια κατεύθυνση για να οδηγηθεί στην ευθεία x'x.

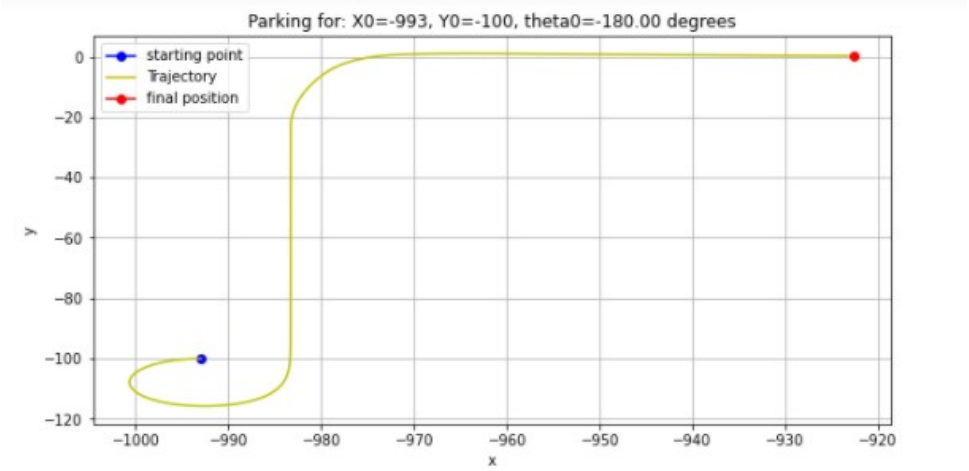
Οι συναρτήσεις μετοχής είναι οι εξής.

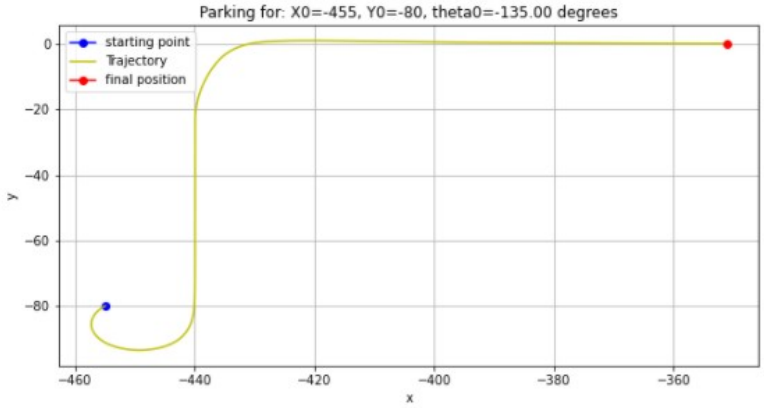


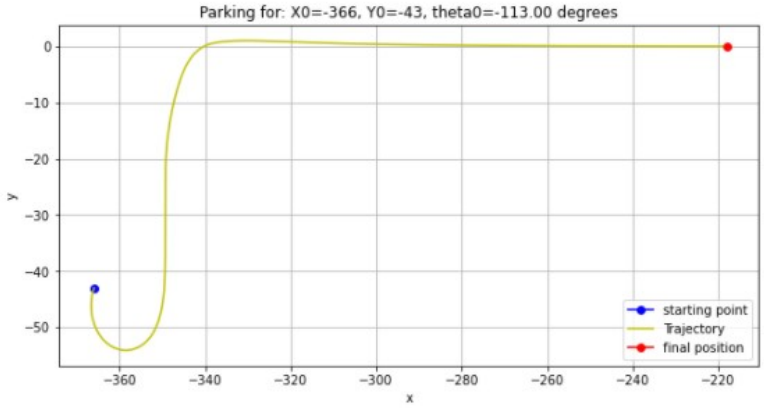
Στη συνέχεια, ακολουθεί επίλυση του γραμμικού συστήματος για χρόνο 400sec και διάφορες αρχικές συνθήκες.

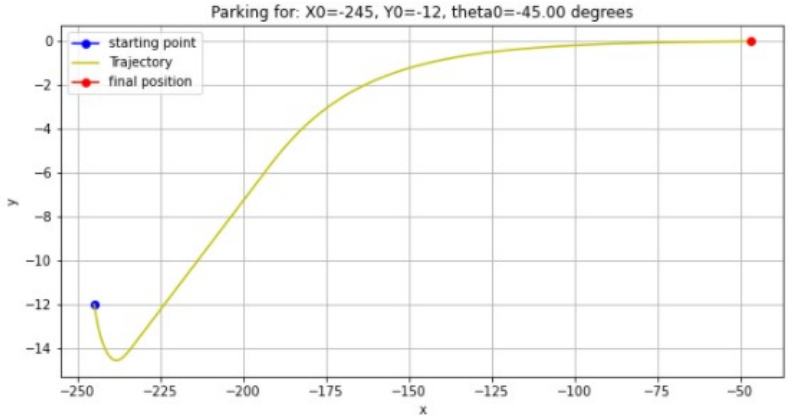
```
theta0 = [-np.pi, -3*np.pi/4, -113*np.pi/180, -np.pi/4, 0, 5*np.pi/18, 47*np.pi/90, 17*np.pi/30, 167*np.pi/180, np.pi]
y0 = [-100, -80, -43, -12, -3, 10, 20, 76, 80, 100]
x0 = [-993, -455, -366, -245, 0, 45, 34, 121, 789, 1000]
for i in range(len(theta0)):
    z0 = [theta0[i], y0[i], x0[i]]
    u = np.zeros(len(t))
    u[0] = fs.get_result(theta0[i], y0[i])
    theta = np.zeros(len(t))
    theta[0] = theta0[i]
    y = np.zeros(len(t))
    y[0] = y0[i]
    x = np.zeros(len(t))
    x[0] = x0[i]
    plt.figure(figsize=(10,5))
    plt.title(f'Parking for: X0={x0[i]}, Y0={y0[i]}, theta0={"%.2f" % (theta0[i]*180/np.pi)} degrees')
    plt.plot(x0[i], y0[i], marker="o",color="blue", label="starting point")
    for k in range(1, len(t)):

        dthetadt = T*V*np.tan(u[k-1])/L
        dxdt = T*V*np.cos(theta[k-1])
        dydt = T*V*np.sin(theta[k-1])

        theta[k] = theta[k-1] + dthetadt
        if(theta[k]<-np.pi):
            theta[k]=theta[k]+2*np.pi
        if(theta[k]>np.pi):
            theta[k]=theta[k]-2*np.pi
        y[k] = y[k-1] + dydt
        x[k] = x[k-1] + dxdt
        u[k] = fs.get_result(theta[k], y[k])


    plt.plot(x, y, 'y', label='Trajectory')
    plt.plot(x[len(x)-1],y[len(y)-1],marker="o",color="red" ,label="final position")
    plt.legend(loc='best')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid()
    plt.show()
```

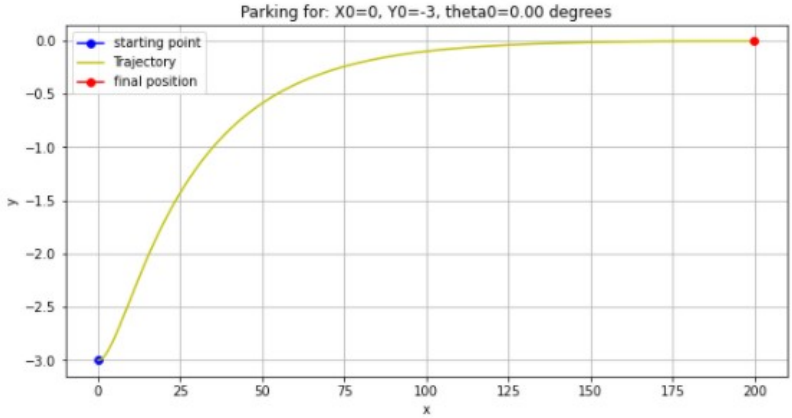
Τελικά, προκύπτουν τα ζητούμενα διαγράμματα, στα οποία το όχημα παρκάρει πάντα στην ευθεία x’x, της οποίας τα σημεία αποτελούν Σ.Ι. , καθώς όταν φτάσε εκεί δεν απομακρύνεται ξανά.

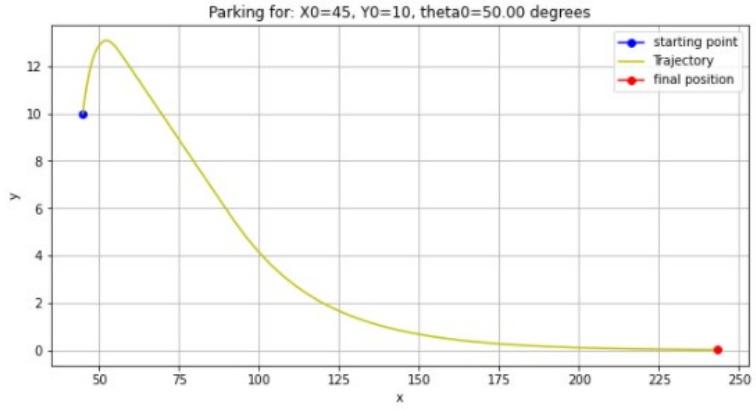


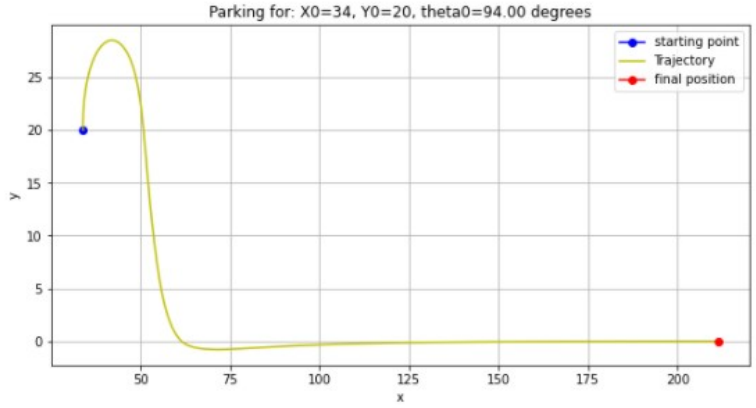

Parking for: X0=-455, Y0=-80, theta0=-135.00 degrees
starting point
Trajectory
final position
0
−20
−40
−60
−80
y
−460
−440
−420
−400
−380
−360
x

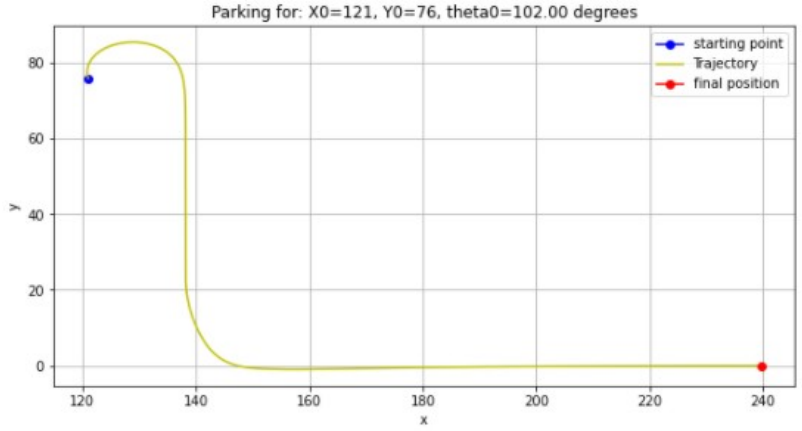

Parking for: X0=-366, Y0=-43, theta0=-113.00 degrees
0
−10
−20
−30
−40
−50
y
starting point
Trajectory
final position
−360
−340
−320
−300
−280
−260
−240
−220
x

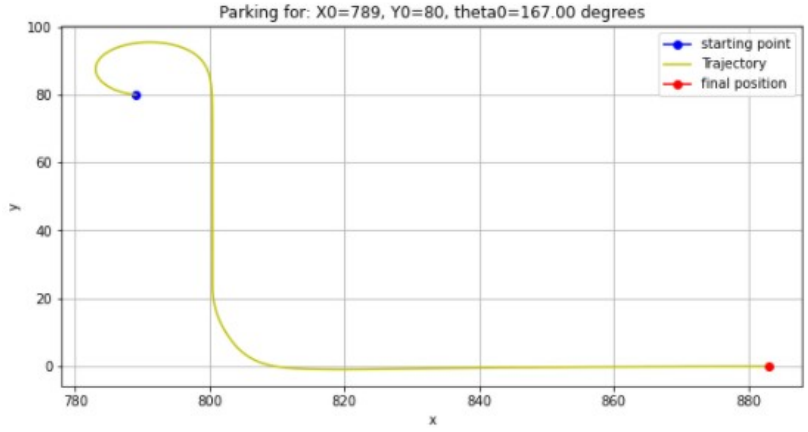

Parking for: X0=-245, Y0=-12, theta0=-45.00 degrees
starting point
Trajectory
final position
0
−2
−4
−6
−8
−10
−12
−14
y
−250
−225
−200
−175
−150
−125
−100
−75
−50
x

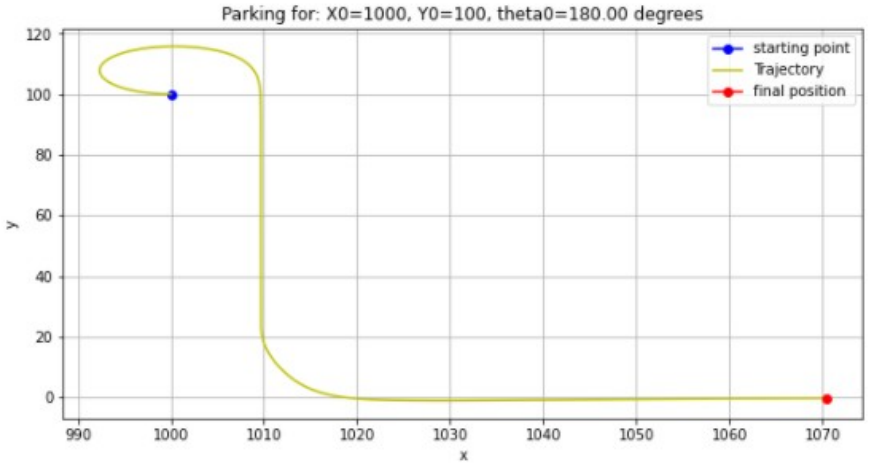

Parking for: X0=0, Y0=-3, theta0=0.00 degrees
starting point
Trajectory
final position
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
y
0
25
50
75
100
125
150
175
200
x


Parking for: X0=45, Y0=10, theta0=50.00 degrees
starting point
Trajectory
final position
x
y


Parking for: X0=34, Y0=20, theta0=94.00 degrees
starting point
Trajectory
final position
x
y


Parking for: X0=121, Y0=76, theta0=102.00 degrees
starting point
Trajectory
final position
x
y


Parking for: X0=789, Y0=80, theta0=167.00 degrees
starting point
Trajectory
final position
x
y



Σχόλια:

1. Το αποτέλεσμα είναι εκπληκτικό. Στα παραπάνω διαγράμματα φαίνεται ότι, οι ασαφείς ελεγκτές μπορούν να δώσουν τα ίδια αποτελέσματα με τους κλασικούς PID, σε προβλήματα όπως αυτό. Το όχημα οδηγείται σε ΣΙ, με απλές εντολές που δείχνουν την κατεύθυνση κι χωρίς αυστηρό φορμαλισμό στο control input.
2. Επίσης, παρατηρεί κανείς τον διαφορετικό χρόνο στο μεταβατικό φαινόμενο, καθώς εξαρτάται από την αρχική κατάσταση.