

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

**Σχεδιασμός και ανάπτυξη αλγορίθμων ελέγχου και λογισμικού για
εργαστηριακές ασκήσεις Αυτομάτου Ελέγχου**

υπό

ΑΡΓΥΡΗ ΚΩΝΣΤΑΝΤΙΝΟ

Υπεβλήθη για την εκπλήρωση μέρους των

απαιτήσεων για την απόκτηση του

Διπλώματος Μηχανολόγου Μηχανικού

2011



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΒΙΒΛΙΟΘΗΚΗ & ΚΕΝΤΡΟ ΠΛΗΡΟΦΟΡΗΣΗΣ
ΕΛΛΗΝΙΚΗ ΣΥΛΛΟΓΗ «ΓΚΡΙΖΑ ΒΙΒΛΙΟΓΡΑΦΙΑ»

Αριθ. Εισ. 9991/1
Ημερ. Εισ.: 26-10-2011
Δωρεά: Συγγραφέας
Ταξιθετικός Κωδικός: ΠΤ – ΜΜ
2011
ΑΡΓ

© 2011 Αργύρης Κωνσταντίνος

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανολόγων Μηχανικών της Πολυτεχνικής Σχολής του Πανεπιστημίου Θεσσαλίας δεν υποδηλώνει αποδοχή των απόψεων του συγγραφέα (Ν. 5343/32 αρ. 202 παρ. 2).

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής (Επιβλέπων)	Δρ. Κωνσταντίνος Βλάχος Συμβασιούχος Διδάσκων (Π.Δ. 407/80), Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Τάσος Σταματέλλος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Κωνσταντίνος Παπαδημητρίου Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Πρώτα απ' όλα, θέλω να ευχαριστήσω τον επιβλέποντα της διπλωματικής εργασίας μου, κ. Κωνσταντίνο Βλάχο, για την πολύτιμη βοήθεια και καθοδήγησή του κατά τη διάρκεια της δουλειάς μου.

Επίσης, είμαι ευγνώμων στα υπόλοιπα μέλη της εξεταστικής επιτροπής, Καθηγητές κ. Τάσο Σταματέλλο και κ. Κώστα Παπαδημητρίου για την ανάγνωση της εργασίας μου και για τις πολύτιμες υποδείξεις τους. Ακόμη, θα ήθελα να ευχαριστήσω τον Φίλιππο Παγωνόπουλο για τη βοήθεια του σχετικά με τη συνδεσμολογία της πειραματικής διάταξης.

Πάνω απ' όλα, είμαι ευγνώμων στους γονείς μου και στον αδερφό μου, στους οποίους και αφιερώνω αυτή την εργασία, για την αγάπη και την υποστήριξή τους όλα αυτά τα χρόνια.

Κωνσταντίνος Αργύρης

Σχεδιασμός και ανάπτυξη αλγορίθμων ελέγχου και λογισμικού για εργαστηριακές ασκήσεις Αυτομάτου Ελέγχου

Κωνσταντίνος Αργύρης

Πανεπιστήμιο Θεσσαλίας, Τμήμα Μηχανολόγων Μηχανικών, 2011

Επιβλέπων Καθηγητής: Δρ. Κωνσταντίνος Βλάχος, Συμβασιούχος Διδάσκων (Π.Δ. 407/80)

Περίληψη

Ο Αυτόματος Έλεγχος είναι μια από τις σημαντικότερες επιστήμες του σύγχρονου κόσμου καθώς τα πάντα τείνουν να αυτοματοποιούνται. Από οικιακές συσκευές όπως τα κλιματιστικά μέχρι βιομηχανίες μαζικής παραγωγής χρησιμοποιείται κάποιας μορφής έλεγχος. Σκοπός της παρούσας διπλωματικής εργασίας είναι η δημιουργία εργαστηριακών ασκήσεων για το μάθημα του Αυτομάτου Ελέγχου, έτσι ώστε οι φοιτητές να έχουν τη δυνατότητα να δούνε πως εφαρμόζονται στην πράξη οι αρχές του μαθήματος αυτού. Βέβαια τα πεδία εφαρμογής του Αυτομάτου Ελέγχου είναι πάρα πολλά και είναι αδύνατον να περιγραφούν σε μια μόνο εργασία. Στη συγκεκριμένη εργασία πραγματοποιείται έλεγχος θέσης και ταχύτητας σε έναν κινητήρα συνεχούς ρεύματος. Ο κινητήρας συνεχούς ρεύματος είναι ένα αρκετά καλό παράδειγμα εφαρμογής αυτομάτου ελέγχου γιατί είναι από τις πιο πολυχρησιμοποιημένες μηχανές. Αρχικά περιγράφεται η πειραματική διάταξη και το χρησιμοποιούμενο λογισμικό. Στη συνέχεια γίνεται η μοντελοποίηση του συστήματος και ο έλεγχός του, και δίνεται πλήρης περιγραφή του προγράμματος που γράφτηκε για την εκτέλεση των εργαστηριακών ασκήσεων. Τέλος, στο παράρτημα υπάρχει ο κώδικας του προγράμματος, ενώ διάφορα σχετικά αρχεία που χρησιμοποιήθηκαν (π.χ Simulink models) υπάρχουν στο συνοδευτικό cd.

Πίνακας Περιεχομένων

Κεφάλαιο 1 : Εισαγωγή.....	9
1.1 Κίνητρο και Υπόβαθρο	9
1.2 Βιβλιογραφική Ανασκόπηση	9
1.3 Οργάνωση Διπλωματικής Εργασίας.....	10
Κεφάλαιο 2 : Περιγραφή συστήματος.....	12
2.1 Περιγραφή πειραματικής διάταξης και συνδεσμολογίας	12
2.2 Περιγραφή του λογισμικού	17
Κεφάλαιο 3 : Μοντελοποίηση του συστήματος	19
3.1 Διατύπωση εξισώσεων κίνησης	19
3.2 Μοντελοποίηση στο πεδίο του χρόνου	23
3.3 Μοντελοποίηση στο πεδίο της συχνότητας.....	30
3.4 Σύγκριση μοντέλου με το πραγματικό σύστημα	35
Κεφάλαιο 4 : Έλεγχος ταχύτητας με αναλογικό ελεγκτή.....	39
4.1 Ανάλυση του συστήματος κλειστού βρόχου	39
4.2 Σφάλμα μόνιμης κατάστασης και επίδραση του συντελεστή ζ στην απόκριση του συστήματος	43
4.3 Κριτήριο ευστάθειας Routh-Hurwitz	47
4.4 Τρόποι αντιμετώπισης του σφάλματος μόνιμης κατάστασης	53
4.5 Ανάλυση του συστήματος με τη βοήθεια του τόπου των ριζών.....	56
Κεφάλαιο 5 : Έλεγχος της ταχύτητας με αναλογικό-ολοκληρωτικό ελεγκτή.....	63
5.1 Ανάλυση του συστήματος κλειστού βρόχου	63
5.2 Κριτήριο ευστάθειας Routh-Hurwitz	66
5.3 Σφάλμα μόνιμης κατάστασης και ανάλυση με τη βοήθεια του τόπου των ριζών	70
Κεφάλαιο 6 : Εκτέλεση εργαστηριακών ασκήσεων.....	94
6.1 Εισαγωγή.....	94
6.2 Χρησιμότητα του προγράμματος και περιγραφή των λειτουργιών του	95
6.3 Αναλυτική περιγραφή προγράμματος	96
6.4 Πληροφορίες για την εκτέλεση των πειραμάτων	99
Κεφάλαιο 7 : Σύνοψη και μελλοντικές εργασίες.....	101
7.1 Σύνοψη	101
7.2 Προτάσεις για μελλοντικές εργασίες.....	101
Βιβλιογραφία.....	102
Παράρτημα.....	103

Κατάλογος Σχημάτων

Σχήμα 2-1: Κινητήρας ΣΡ-MM με encoder HEDL-5540 της maxonmotors.....	12
Σχήμα 2-2: Servoamplifier 4-Q-DC 50V/5A της maxon.....	13
Σχήμα 2-3: Τροφοδοτικό Siemens input AC 230V/120V , output 24V DC/5A.....	13
Σχήμα 2-4: Κάρτα NI-PCI 6221 της National Instruments.	14
Σχήμα 2-5: Connector-block CB-68LP της NI για επικοινωνία με NI PCI-6221.	15
Σχήμα 2-6: Συνδεσμολογία συστήματος ελέγχου κινητήρα συνεχούς ρεύματος.	17
Σχήμα 2-7: Βιβλιοθήκη του Simulink με τα απαραίτητα blocks.....	18
Σχήμα 3-1: Ηλεκτρικό κύκλωμα κινητήρα συνεχούς ρεύματος.....	20
Σχήμα 3-2: Κύκλωμα και διάγραμμα ελευθέρου σώματος κινητήρα Σ.Ρ.	21
Σχήμα 3-3: Γωνιακή ταχύτητα σε συνάρτηση με το χρόνο.	28
Σχήμα 3-4: Ένταση ρεύματος σε συνάρτηση με το χρόνο.	28
Σχήμα 3-5: Συνάρτηση μεταφοράς γωνιακής ταχύτητας-τάσης.....	31
Σχήμα 3-6: Δομικό διάγραμμα θέσης-τάσης.	34
Σχήμα 3-7: Δομικό διάγραμμα ταχύτητας-τάσης.	34
Σχήμα 3-8: Απόκριση θέσης μοντέλου και πραγματικού συστήματος σε είσοδο τάσης 3 Volt.....	36
Σχήμα 3-9: Σήμα με θόρυβο και χωρίς θόρυβο.	38
Σχήμα 4-1: Δομικό διάγραμμα συστήματος κλειστού βρόχου.....	39
Σχήμα 4-2: Απλοποιημένο δομικό διάγραμμα συστήματος.....	40
Σχήμα 4-3: Απόκριση μοντέλου για $K_p=1$	42
Σχήμα 4-4: Απόκριση μοντέλου για $K_p=0.3293$	43
Σχήμα 4-5: Μεταβολή των πόλων με το ζ	45
Σχήμα 4-6: Απόκριση συστήματος ανάλογα με τη θέση των πόλων στο μιγαδικό επίπεδο.....	46
Σχήμα 4-7: Απόκριση συστήματος για πολύ μεγάλο K_p ($=50$).....	50
Σχήμα 4-8: Απόκριση συστήματος για πολύ μικρό K_p ($=0.01$).....	51
Σχήμα 4-9: Απόκριση συστήματος για αρνητικό K_p ($=-0.02$).....	51
Σχήμα 4-10: Απόκριση του συστήματος για K_p εκτός του πεδίου τιμών ($=-0.05$).....	52
Σχήμα 4-11: Τόπος των ριζών για θετικές τιμές του K	58
Σχήμα 4-12: Τόπος των ριζών για $-0.08 < K < 0.08$	59

Σχήμα 4-13: Πληροφορίες που μας δίνει ο T.P.	60
Σχήμα 4-14: Σχεδίαση πόλων με την εντολή <code>locfind</code>	61
Σχήμα 5-1: Δομικό διάγραμμα συστήματος με αναλογικό ελεγκτή.	63
Σχήμα 5-2: Δομικό διάγραμμα συστήματος με PI ελεγκτή.	64
Σχήμα 5-3: Πίνακας R-H με χρήση MATLAB	66
Σχήμα 5-4: Ασταθής απόκριση για αρνητικό K_i ($K_i=-1$)	67
Σχήμα 5-5: Ευσταθής απόκριση για $K_p=1$ και $K_i=240$	69
Σχήμα 5-6: Ασταθής απόκριση για $K_p=1$ και $K_i=260$	69
Σχήμα 5-7: Εντολές MATLAB για σχεδίαση του τόπου των ριζών.	74
Σχήμα 5-8: Τόπος των ριζών για $K_p=1$ και K_i θετικό.	74
Σχήμα 5-9: Πόλοι και χαρακτηριστικά τους ($K_p=1$)	76
Σχήμα 5-10: Εντολές MATLAB για σχεδίαση T.P με $K_p=2$ (K_i θετικό)	77
Σχήμα 5-11: Τόπος των ριζών για $K_p=2$, K_i θετικό	78
Σχήμα 5-12: Τόπος των ριζών για $K_p=0.5$, K_i θετικό	79
Σχήμα 5-13: Τόπος των ριζών για $K_p=0.3$, K_i θετικό	79
Σχήμα 5-14: Αρνητικοί πραγματικοί πόλοι στον τόπο των ριζών για $K_p=0.3$	80
Σχήμα 5-15: Απόκριση μοντέλου στο Simulink για $K_i=4.5$ και $K_p=0.3$	81
Σχήμα 5-16: Υπολογισμός συντελεστών R και p με το MATLAB	83
Σχήμα 5-17: Απόκριση για $K_i=4.5$ και $K_p=0.3$	84
Σχήμα 5-18: Πόλοι και μηδενιστής για $K_i=2.4$ ($K_p=0.3$)	88
Σχήμα 5-19: Απόκριση για $K_i=2.4$ και $K_p=0.3$	89
Σχήμα 5-20: Μηδενιστής ανάμεσα στους 2 πόλους	90
Σχήμα 5-21: Απόκριση για $K_i=1.5$ και $K_p=0.3$	91
Σχήμα 5-22: Χαρακτηριστικές θέσεις πόλων στον τόπο των ριζών	92
Σχήμα 6-1: Γραφικό περιβάλλον του προγράμματος 'Control Systems Lab GUI'	94

Κεφάλαιο 1 : Εισαγωγή

Σε αυτό το κεφάλαιο, παρουσιάζουμε πληροφορίες εισαγωγικού χαρακτήρα που δίνουν το κίνητρο και το υπόβαθρο αυτής της διπλωματικής εργασίας, παραθέτουμε μια ανασκόπηση της σχετικής με την εργασία βιβλιογραφίας και περιγράφουμε συνοπτικά τις βασικές ενότητες της διπλωματικής εργασίας.

1.1 Κίνητρο και Υπόβαθρο

Κίνητρο αυτής της διπλωματικής εργασίας ήταν η έλλειψη εργαστηριακών ασκήσεων στο μάθημα του Αυτομάτου Ελέγχου. Ο Αυτόματος Έλεγχος είναι ένα μάθημα του οποίου η θεωρία είναι αρκετά μεγάλη και μαθηματικοποιημένη με αποτέλεσμα να φαίνεται σαν ένα μάθημα καθαρά θεωρητικό. Η αλήθεια όμως είναι ότι είναι καθαρά πρακτικό. Και μάλιστα στην πράξη υπάρχουν πολύ περισσότερα προβλήματα προς λύση από ότι στην θεωρία. Για αυτόν ακριβώς τον λόγο είναι πολύ δύσκολο για κάποιον που έχει μελετήσει μόνο τη θεωρία να προσπαθήσει να την εφαρμόσει σε ένα πραγματικό σύστημα. Έτσι, μπορεί πολύ εύκολα να οδηγηθεί εσφαλμένα στο συμπέρασμα ότι αυτά που έμαθε στη θεωρία δεν εφαρμόζονται στην πράξη. Η συνεισφορά αυτής της διπλωματικής εργασίας είναι ότι γεφυρώνει αυτό το κενό χρησιμοποιώντας ως πραγματικό σύστημα έναν κινητήρα συνεχούς ρεύματος. Οι κινητήρες γενικά είναι από τις σημαντικότερες μηχανές στις οποίες εφαρμόζεται έλεγχος. Τέλος, δίνεται ιδιαίτερη έμφαση στη συστηματοποίηση της διαδικασίας των εργαστηριακών ασκήσεων έτσι ώστε να μπορούν να εκτελεστούν με ευκολία.

1.2 Βιβλιογραφική Ανασκόπηση

Η βιβλιογραφία που σχετίζεται με τη θεωρία του Αυτομάτου Ελέγχου είναι τεράστια. Στη συγκεκριμένη εργασία χρησιμοποιήθηκαν τα βιβλία [1] και [2] για τη θεωρητική ανάλυση των μοντέλων. Εκεί μπορεί να ανατρέξει ο αναγνώστης για περισσότερες πληροφορίες για θέματα όπως

ανάλυση ευστάθειας, τόπος των ριζών, ανάλυση μεταβατικής απόκρισης και ότι άλλο σχετίζεται με τη θεωρία του Αυτομάτου Ελέγχου.

Η βιβλιογραφία [3] έχει να κάνει με τις ηλεκτρικές μηχανές γενικά. Στην παρούσα εργασία χρησιμοποιήθηκαν τα κεφάλαια σχετικά με τους κινητήρες συνεχούς ρεύματος, για τη διαδικασία της μοντελοποίησης αλλά και για την κατανόηση της λειτουργίας τους.

Όσον αφορά στον προγραμματισμό με MATLAB και Simulink, χρησιμοποιήθηκε η βιβλιογραφία [4], [5] και [8]. Η [4] περιέχει γενικές πληροφορίες για το MATLAB καθώς και εξειδικευμένο κεφάλαιο για εφαρμογές στον Αυτόματο Έλεγχο. Οι βιβλιογραφίες [5] και [8] είναι ηλεκτρονικές. Η πρώτη είναι ο επίσημος ιστότοπος της εταιρίας ‘The MathWorks’ η οποία αναπτύσσει το MATLAB, και η δεύτερη είναι ένα ανεξάρτητο blog το οποίο ασχολείται με προγραμματισμό στο MATLAB και συγκεκριμένα με τη δημιουργία και προγραμματισμό γραφικού περιβάλλοντος επικοινωνίας χρήστη (GUI).

Για την επικοινωνία του κινητήρα με τον υπολογιστή χρησιμοποιήθηκαν πληροφορίες από το επίσημο site της ‘National Instruments’ (βιβλιογραφία [6]) της οποίας χρησιμοποιήθηκε μια κάρτα εισόδου-εξόδου.

Τέλος, απαραίτητες πληροφορίες και παράμετροι για τον κινητήρα και τον ενισχυτή λήφθηκαν από το επίσημο site της κατασκευάστριας εταιρίας ‘Maxonmotors’ (βιβλιογραφία [7]).

1.3 Οργάνωση Διπλωματικής Εργασίας

Στο κεφάλαιο 2 γίνεται αναλυτική περιγραφή του κάθε στοιχείου της πειραματικής διάταξης, της συνδεσμολογίας, καθώς και του λογισμικού που χρησιμοποιείται στην εργασία.

Στο κεφάλαιο 3 περιγράφεται η διαδικασία της μοντελοποίησης του συστήματος με δυο διαφορετικούς τρόπους και γίνεται σύγκριση των αποτελεσμάτων με αυτά που λαμβάνουμε από το πραγματικό σύστημα.

Στο κεφάλαιο 4 πραγματοποιείται έλεγχος της ταχύτητας του μοντέλου χρησιμοποιώντας αναλογικό ελεγκτή. Μελετάται η επίδραση του κέρδους του ελεγκτή στην απόκριση του συστήματος με τη βοήθεια του τόπου των ριζών και του κριτηρίου ευστάθειας Routh-Hurwitz. Επίσης μελετώνται θέματα όπως το σφάλμα μόνιμης κατάστασης και η μεταβατική απόκριση και προτείνονται τρόποι αντιμετώπισής τους.

Στο κεφάλαιο 5 εισάγεται και ολοκληρωτικός όρος στον ελεγκτή και έτσι έχουμε αναλογικό-ολοκληρωτικό ελεγκτή. Αυτή η αλλαγή μας οδηγεί σε ένα σύστημα τρίτης τάξης, πράγμα που κάνει την επεξεργασία του πιο δύσκολη. Γίνεται ανάλυση του καινούριου συστήματος με τον τόπο των ριζών και το κριτήριο ευστάθειας R.H. και μελετάται η επίδραση του ολοκληρωτικού όρου στην απόκριση. Τέλος, γίνεται σύγκριση με τον αναλογικό ελεγκτή.

Το κεφάλαιο 6 ασχολείται με το πρόγραμμα το οποίο γράφηκε στα πλαίσια της εργασίας για την εκτέλεση των εργαστηριακών ασκήσεων. Αναλύεται η λειτουργία του κάθε κουμπιού και δίνονται πληροφορίες για την σωστή εκτέλεση των πειραμάτων.

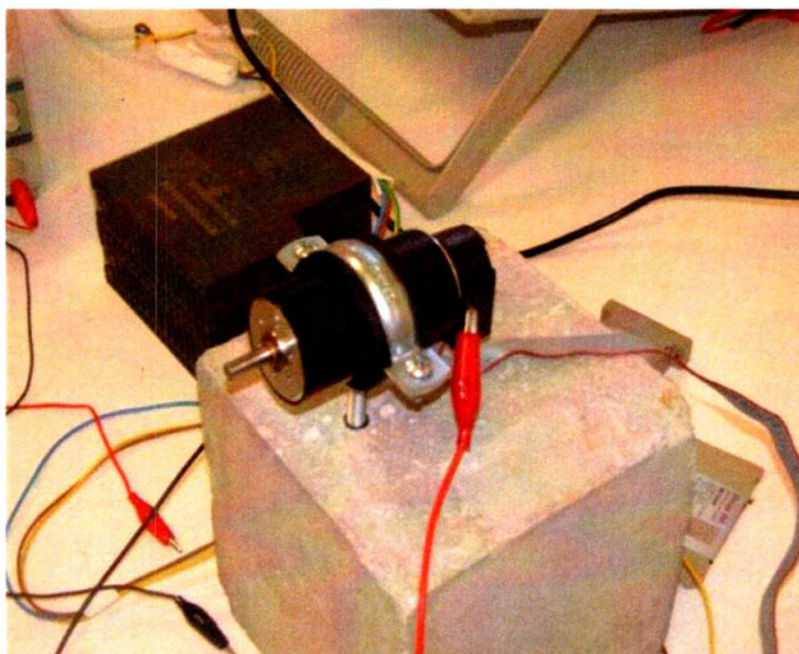
Τα τελικά συμπεράσματα της διπλωματικής εργασίας καθώς και κατευθύνσεις για περαιτέρω έρευνα παρουσιάζονται στο κεφάλαιο 7.

Κεφάλαιο 2 : Περιγραφή συστήματος

2.1 Περιγραφή πειραματικής διάταξης και συνδεσμολογίας

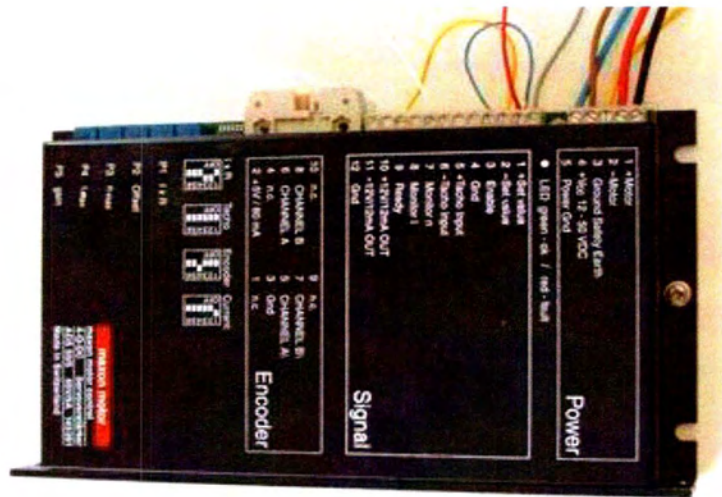
Το κυρίως σύστημα αποτελείται από τον κινητήρα και τον ενισχυτή ο οποίος χρησιμοποιείται για την οδήγηση του κινητήρα. Υπάρχουν όμως και άλλα επιμέρους στοιχεία όπως το τροφοδοτικό που τροφοδοτεί τον ενισχυτή, η κάρτα I/O (εισόδου - εξόδου) που συνδέεται στον υπολογιστή και το connector block. Τα στοιχεία αυτά φαίνονται στις παρακάτω φωτογραφίες μαζί με μια σύντομη περιγραφή.

1. Κινητήρας ΣΡ-MM 48V της maxonmotors. Στο πίσω μέρος του κινητήρα υπάρχει τοποθετημένος encoder (αισθητήρας θέσης) με 500 counts HEDL 5540.



Σχήμα 2-1: Κινητήρας ΣΡ-MM με encoder HEDL-5540 της maxonmotors.

2. Servoamplifier (ενισχυτής) 50V/5A της maxon με δυνατότητα 4 ελέγχων (modes).



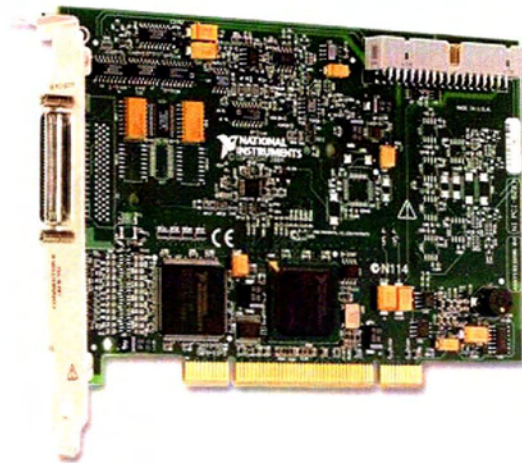
Σχήμα 2-2: Servoamplifier 4-Q-DC 50V/5A της maxon.

3. Τροφοδοτικό 24Volt DC/5A της Siemens. Χρησιμοποιείται για την μετατροπή του AC ρεύματος του δικτύου σε DC για την τροφοδοσία του ενισχυτή.



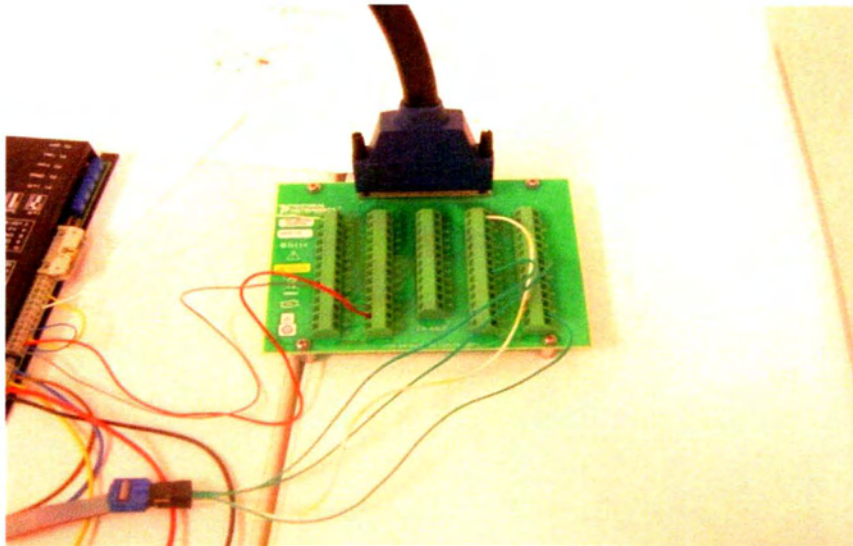
Σχήμα 2-3: Τροφοδοτικό Siemens input AC 230V/120V , output 24V DC/5A.

4. Κάρτα εισόδου/εξόδου της National Instruments, μοντέλο 6221-PCI, με αναλογική και ψηφιακή είσοδο/έξοδο και connector block για επικοινωνία με το σύστημα. Η αναλογική είσοδος και έξοδος της κάρτας είναι $\pm 10\text{VDC}$ (Volts, Direct Current). Περισσότερα χαρακτηριστικά της κάρτας βρίσκονται στο παράρτημα.



Σχήμα 2-4: Κάρτα NI-PCI 6221 της National Instruments.

5. Connector block, μοντέλο CB-68LP, της National Instruments για επικοινωνία της κάρτας με το περιβάλλον. Μέσω του connector block γίνεται η επικοινωνία του κινητήρα και του ενισχυτή με τον υπολογιστή.



Σχήμα 2-5: Connector-block CB-68LP της NI για επικοινωνία με NI PCI-6221.

Αφού παρουσιάσαμε τα στοιχεία που χρησιμοποιούμε ξεχωριστά ας δούμε τώρα πως λειτουργεί συνολικά το σύστημά μας. Στόχος μας είναι να πραγματοποιήσουμε έλεγχο θέσης ή ταχύτητας του κινητήρα σε πραγματικό χρόνο. Για να το πετύχουμε αυτό πρέπει να μπορούμε να ανταλλάσσουμε πληροφορίες με τον κινητήρα. Οι πληροφορίες αυτές είναι 2 , η είσοδος και η έξοδος. Η είσοδος είναι η τάση που τροφοδοτεί τον κινητήρα και η έξοδος είναι η γωνία στροφής του.

Την τάση την εφαρμόζει στους ακροδέκτες του κινητήρα ο ενισχυτής, ο οποίος συνδέεται με το connector block δηλαδή με τον υπολογιστή. Οπότε ανάλογα με τις ρυθμίσεις που έχουμε κάνει στο λογισμικό, η κάρτα I/O δίνει εντολή στον ενισχυτή να εφαρμόσει την αντίστοιχη τάση στους ακροδέκτες του κινητήρα, με αποτέλεσμα να αρχίσει να περιστρέφεται.

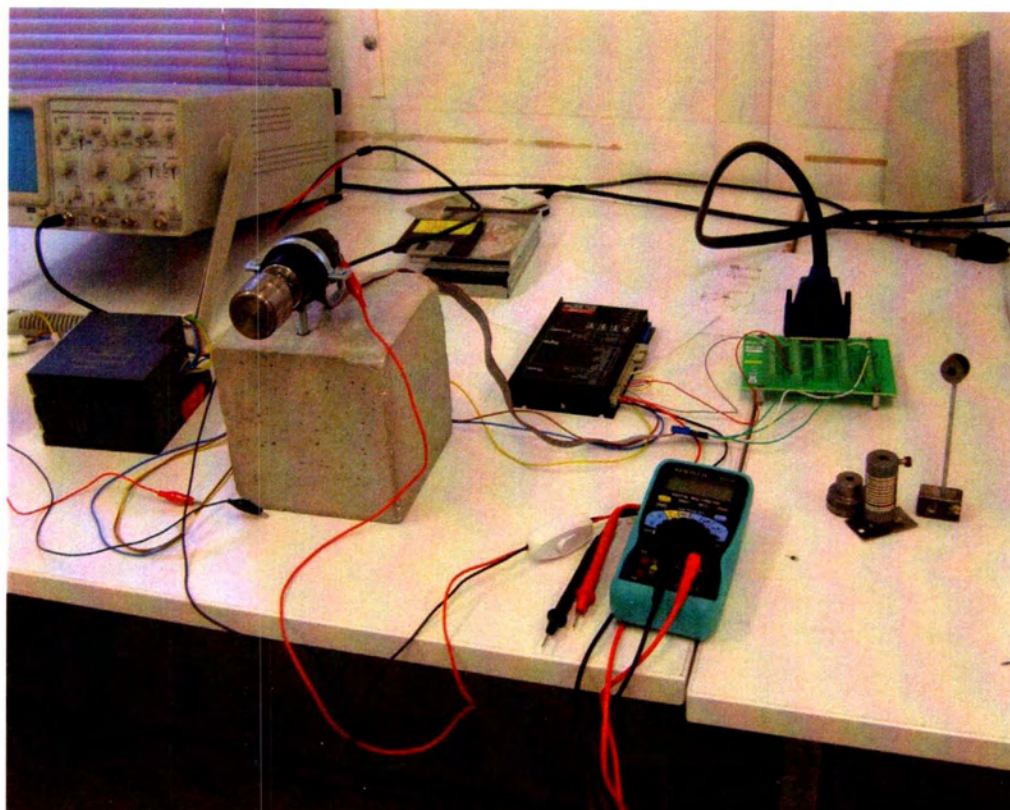
Όσο για τη γωνία στροφής του άξονα, στο πίσω μέρος του κινητήρα υπάρχει ενσωματωμένος αισθητήρας θέσης ο οποίος επικοινωνεί επίσης με το connector block και κατά συνέπεια με τον υπολογιστή. Για την ακρίβεια, ο αισθητήρας θέσης χρησιμοποιεί τέσσερα καλώδια για να επικοινωνεί με το connector block. Ένα για την τροφοδοσία του (5V) , ένα για τη γείωση, και δυο που μας δίνουν τη θέση. Τέλος, το connector block επικοινωνεί με την κάρτα εισόδου/εξόδου μέσω ενός καλωδίου

BNC-68. Έτσι, με χρήση λογισμικού που θα αναφέρουμε στη συνέχεια, διαβάζουμε το σήμα που μας στέλνει ο αισθητήρας θέσης και με κατάλληλο πολλαπλασιασμό το μετατρέπουμε σε γωνία.

Σχηματικά, ο τρόπος επικοινωνίας των διαφόρων στοιχείων είναι ο εξής :

- Αισθητήρας θέσης → connector block → κάρτα εισόδου/εξόδου → υπολογιστής , είναι ο τρόπος που διαβάζουμε τη γωνία που έχει στραφεί ο άξονας του κινητήρα (έξοδος συστήματος).
- Υπολογιστής → κάρτα εισόδου/εξόδου → connector block → ενισχυτής → κινητήρας , είναι ο τρόπος που τροφοδοτούμε με τάση τον κινητήρα (είσοδος συστήματος).
- Τροφοδοτικό → ενισχυτής, είναι ο τρόπος τροφοδότησης του ενισχυτή.

Έτσι με τον τρόπο που περιγράφηκε, επιτυγχάνεται η σε-πραγματικό-χρόνο επικοινωνία του κινητήρα και του ενισχυτή με τον υπολογιστή. Η παραπάνω συνδεσμολογία φαίνεται στο επόμενο σχήμα.



Σχήμα 2-6: Συνδεσμολογία συστήματος ελέγχου κινητήρα συνεχούς ρεύματος.

Αφού καλύψαμε το κομμάτι του hardware θα περάσουμε στο software.

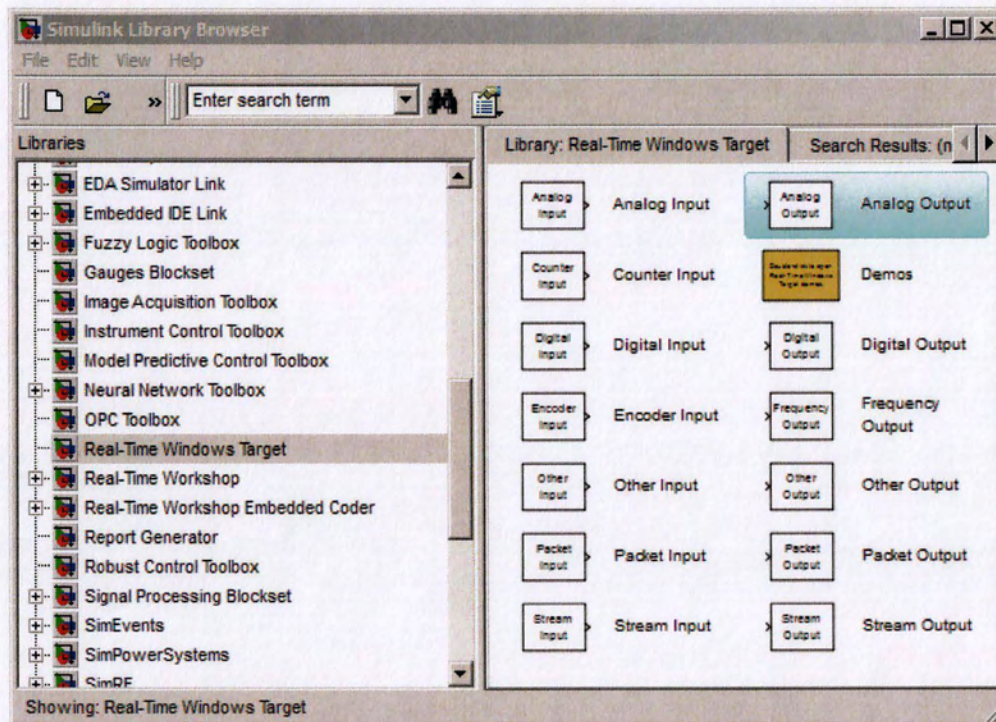
2.2 Περιγραφή του λογισμικού

Όπως είπαμε, όλη η επικοινωνία μας με το σύστημα ξεκινά και καταλήγει από και προς την κάρτα εισόδου/εξόδου της National Instruments. Από τη στιγμή που πραγματοποιήσουμε αυτή τη σύνδεση είναι θέμα λογισμικού το πως θα επιτευχθεί ο έλεγχος.

Το λογισμικό που χρησιμοποιήθηκε για την επικοινωνία με την κάρτα είναι το Simulink το οποίο είναι υπό-πρόγραμμα του MATLAB και πιο συγκεκριμένα το πακέτο του Simulink : “Real-Time Windows Target”.

Το πακέτο αυτό διαθέτει μεταξύ άλλων έτοιμα blocks για αναλογική είσοδο και έξοδο (analog input/output) από και προς την κάρτα, καθώς και blocks για είσοδο μετρητή (counter input). Αυτά τα

blocks χρησιμοποιούνται για να δώσουμε εντολή για τάση στην κάρτα (και συνεπώς στον κινητήρα), και για να διαβάσουμε το σήμα του αισθητήρα θέσης αντίστοιχα. Τα blocks αυτά φαίνονται στο επόμενο σχήμα, στη βιβλιοθήκη του Simulink στο πακέτο Real-Time Windows Target.



Σχήμα 2-7: Βιβλιοθήκη του Simulink με τα απαραίτητα blocks.

Τα blocks που μας ενδιαφέρουν είναι τα : Analog Output και Counter Input. Το Analog Output δίνει εντολή στην κάρτα να παράγει τόσα Volt όσα έχει για είσοδο. Το Counter Input έχει σαν έξοδο τα counts του αισθητήρα θέσης, τα οποία μετατρέπονται πολύ εύκολα σε μοίρες ή rad.

Πρέπει να προσέξουμε τη διαφορά μεταξύ εισόδου και εξόδου του κινητήρα και της κάρτας γιατί είναι δύο διαφορετικά πράγματα.

- Για τον κινητήρα, είσοδος είναι η τάση που εφαρμόζεται στα άκρα του και έξοδος η γωνία στροφής του άξονα.
- Για την κάρτα, είσοδος είναι το σήμα του αισθητήρα θέσης (η γωνία δηλαδή), και έξοδος είναι η τάση που θα τροφοδοτήσει τον κινητήρα.

Γι' αυτό τα blocks Analog Output και Counter Input έχουν και τις αντίστοιχες ονομασίες. Είναι σημαντικό να κατανοήσουμε ότι ο ανοιχτός και κλειστός βρόχος ορίζονται από το λογισμικό. Εμείς ορίζουμε τον τρόπο που θα γίνει ο έλεγχος, μέσα από το πρόγραμμα που θα φτιάξουμε. Το πρόγραμμα αυτό ανάλογα με τις ρυθμίσεις και τις παραμέτρους του στέλνει και τα αντίστοιχα σήματα από και προς τον κινητήρα. Τα προγράμματα που χρησιμοποιήθηκαν στην παρούσα διπλωματική εργασία υπάρχουν στο συνοδευτικό cd.

Κεφάλαιο 3 : Μοντελοποίηση του συστήματος

3.1 Διατύπωση εξισώσεων κίνησης

Με τον όρο μοντελοποίηση εννοούμε τη μαθηματική περιγραφή του συστήματός μας. Ένα καλό μοντέλο είναι πολύ χρήσιμο κυρίως για δύο λόγους.

- Μας βοηθάει να καταλάβουμε καλύτερα τη λειτουργία του συστήματος.
- Μπορούμε να κάνουμε αλλαγές και πειράματα πάνω στο μοντέλο αντί για το πραγματικό σύστημα.

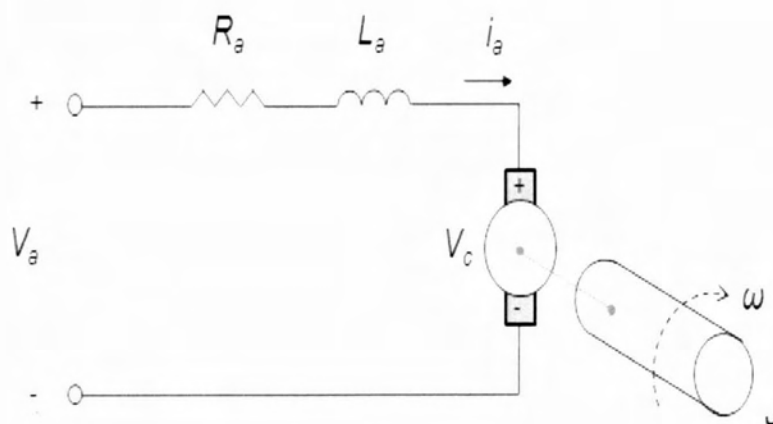
Ένα καλό μοντέλο πρέπει να προβλέπει με ικανοποιητική ακρίβεια τη συμπεριφορά του πραγματικού συστήματος αλλά να μην είναι και πολύ περίπλοκο. Οπότε πρέπει να γίνει ένας συμβιβασμός μεταξύ αυτών των δύο.

Έχοντας κατασκευάσει ένα αξιόπιστο μοντέλο μπορούμε να προβλέψουμε τη συμπεριφορά του πραγματικού μας συστήματος σε διάφορες εισόδους. Επίσης, αναλύοντας το μοντέλο με τη θεωρία αυτομάτου ελέγχου, μπορούμε να υπολογίσουμε τις τιμές διαφόρων παραμέτρων έτσι ώστε να πετύχουμε μια επιθυμητή συμπεριφορά. Εάν πετύχουμε αυτή τη συμπεριφορά στο μοντέλο, μπορούμε να την πετύχουμε (κατά προσέγγιση πάντα) και στο πραγματικό σύστημα. Αυτός είναι και ο σκοπός της μοντελοποίησης.

Βέβαια το μοντέλο δεν μπορεί ποτέ να προσομοιώσει ακριβώς το πραγματικό σύστημα γιατί αυτό είναι πολύ περίπλοκο. Υπάρχουν μη-γραμμικά φαινόμενα που είναι πολύ δύσκολο να μοντελοποιηθούν και για αυτό το λόγο κάνουμε αναγκαστικά κάποιες παραδοχές και απλοποιήσεις όπως θα δούμε παρακάτω.

Για να ξεκινήσουμε τη μοντελοποίηση πρέπει αρχικά να διατυπώσουμε τις εξισώσεις των φυσικών νόμων που διέπουν το σύστημά μας. Στην περίπτωση του κινητήρα συνεχούς ρεύματος οι νόμοι αυτοί είναι ο νόμος του Νεύτωνα για το μηχανικό κομμάτι και ο νόμος του Kirchhoff για το ηλεκτρικό κομμάτι.

Ας ξεκινήσουμε με τον νόμο του Kirchhoff. Το ηλεκτρικό κύκλωμα του κινητήρα φαίνεται στο παρακάτω σχήμα.



Σχήμα 3-1: Ηλεκτρικό κύκλωμα κινητήρα συνεχούς ρεύματος.

Στο παραπάνω σχήμα, τα σύμβολα είναι:

- V_a : Η τάση που εφαρμόζεται στα άκρα του κινητήρα. Δηλαδή η έξοδος του ενισχυτή που αποτελεί πηγή τάσης για τον κινητήρα.
- R_a : Η αντίσταση των τυλιγμάτων του κινητήρα.
- L_a : Η αυτεπαγωγή των τυλιγμάτων.
- I_a : Το ρεύμα που διαρρέει τα τυλίγματα του δρομέα.

- V_c : back EMF
- ω : Γωνιακή ταχύτητα του άξονα.
- J : Ροπή αδράνειας του άξονα.

Τότε ο νόμος του Kirchhoff για τις τάσεις παίρνει τη μορφή:

$$V_a - L \frac{di(t)}{dt} - i(t) * R - V_c = 0$$

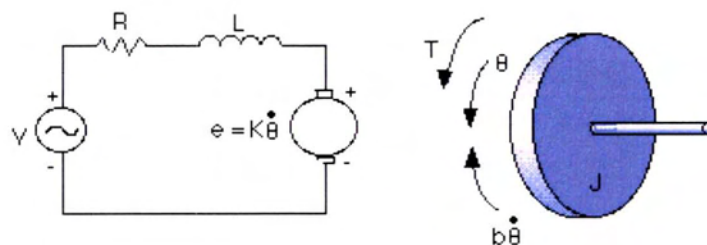
Όμως η V_c είναι ανάλογη της γωνιακής ταχύτητας άρα η σχέση γίνεται:

$$V_a - L \frac{di(t)}{dt} - i(t) * R - k_b * \omega(t) = 0 \quad (1)$$

Όπου k_b η σταθερά ταχύτητας του κινητήρα.

Ο νόμος του Νεύτωνα για την περιστροφική κίνηση είναι:

$$\sum T = J \frac{d\omega}{dt}$$



Σχήμα 3-2: Κύκλωμα και διάγραμμα ελευθέρου σώματος κινητήρα Σ.Ρ.

Η συνισταμένη ροπή που ασκείται στον άξονα είναι η ροπή που βγάζει ο κινητήρας μείον τη ροπή που προκαλούνε οι τριβές. Για κινητήρες συνεχούς ρεύματος η ροπή που παράγεται είναι ανάλογη του ρεύματος που διαρρέει τα τυλίγματα:

$$T = K_t * i(t) \quad (2)$$

Όπου K_t η σταθερά ροπής του κινητήρα.

Η ροπή των τριβών είναι ανάλογη της γωνιακής ταχύτητας:

$$T_{\text{τριβών}} = b * \omega(t)$$

Όπου b η σταθερά τριβής του κινητήρα.

Επομένως ο νόμος του Νεύτωνα γίνεται:

$$K_t * i(t) - b * \omega(t) = J * \frac{d\omega}{dt} \quad (3)$$

Συνοψίζοντας, οι δύο βασικές σχέσεις που καταλήξαμε από τους νόμους των Kirchhoff και Newton είναι:

$$V_a - L * \frac{di(t)}{dt} - i(t) * R - kb * \omega(t) = 0$$

και

$$K_t * i(t) - b * \omega(t) = J * \frac{d\omega}{dt}$$

Αυτές οι εξισώσεις περιγράφουν το σύστημά μας. Η είσοδος είναι η τάση V_a που εφαρμόζεται στα άκρα του κινητήρα, και η έξοδος είναι η γωνιακή ταχύτητα (ή θέση εάν θέσουμε όπου $\omega = d\theta/dt$).

Όπως είδαμε, κατά την κατάστρωση των εξισώσεων εμφανίστηκαν κάποιες σταθερές όπως η K_t και η K_b . Αυτές οι σταθερές είναι χαρακτηριστικά του συγκεκριμένου κινητήρα και κάθε κινητήρας συνεχούς ρεύματος έχει τις δικές του. Συνήθως οι περισσότερες από αυτές αναφέρονται στο εγχειρίδιο χρήσης του κινητήρα από τον κατασκευαστή, αλλά μπορούνε πάντα να υπολογιστούνε και πειραματικά. Στη συγκεκριμένη εργασία χρησιμοποιούνται οι τιμές που δίνει ο κατασκευαστής.

Αφού καταστρώσαμε τις εξισώσεις που περιγράφουν το σύστημά μας το επόμενο βήμα είναι να τις περάσουμε στον υπολογιστή (Simulink) ώστε να μπορούμε να χειριζόμαστε άνετα το μοντέλο.

Υπάρχουν διάφοροι τρόποι για να γίνει αυτό. Μια συνηθισμένη μέθοδος είναι να μετασχηματίσουμε τις εξισώσεις στο πεδίο της συχνότητας (με τον μετασχηματισμό Laplace) και να υπολογίσουμε τη συνάρτηση μεταφοράς του συστήματος. Μια άλλη μέθοδος είναι να κρατήσουμε τις εξισώσεις στο πεδίο του χρόνου και να τις αναπαραστήσουμε σε ένα δομικό διάγραμμα.

3.2 Μοντελοποίηση στο πεδίο του χρόνου

Σε αυτή τη μέθοδο θα υποθέσουμε για ευκολία ότι οι τριβές που παρουσιάζονται κατά την περιστροφή του κινητήρα είναι αμελητέες, δηλαδή $b=0$. Άρα η μόνη ροπή είναι αυτή που βγάζει ο κινητήρας.

Έτσι ο νόμος του Νεύτωνα γίνεται:

$$Kt * i(t) = J * \frac{d\omega}{dt}$$

Γράφουμε τις 2 εξισώσεις (Newton και Kirchhoff) ως εξής:

$$\frac{d\omega}{dt} = \frac{Kt}{J} * i(t)$$

και

$$\frac{di(t)}{dt} = \frac{1}{L} [Va - i(t) * R - kb * \omega(t)]$$

Οι παραπάνω σχέσεις αποτελούνε ένα ζεύγος διαφορικών εξισώσεων. Οι άγνωστοι είναι το $i(t)$ και το $\omega(t)$.

Για να περάσουμε τις σχέσεις στο Simulink είναι βολικό να μην έχουμε καθόλου παραγώγους γιατί η αριθμητική παραγωγή προκαλεί διάφορα προβλήματα που θα συναντήσουμε και στη συνέχεια. Αντιθέτως, η αριθμητική ολοκλήρωση δίνει καλά αποτελέσματα γι' αυτό και ολοκληρώνουμε πρώτα τις παραπάνω σχέσεις:

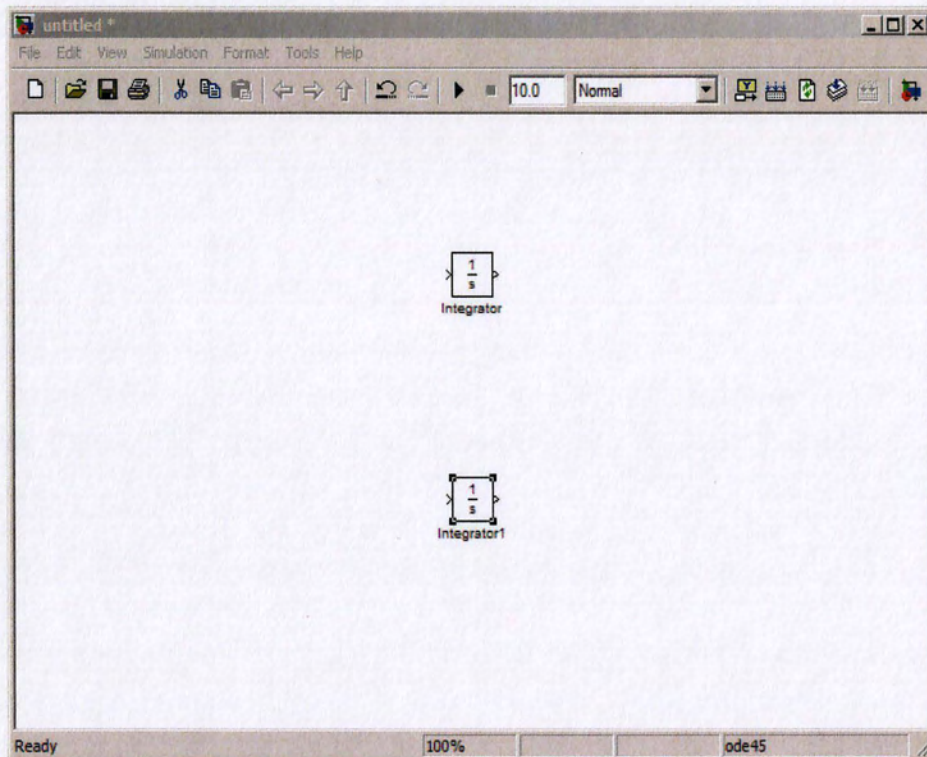
$$\omega(t) = \int \frac{Kt}{J} * i(t) \quad (4)$$

και

$$i(t) = \int \frac{1}{L} [Va - i(t) * R - kb * \omega(t)] \quad (5)$$

Κατά την ολοκλήρωση προέκυψαν φυσικά και οι αρχικές συνθήκες για το ρεύμα και την γωνιακή ταχύτητα αλλά υποθέτουμε ότι τη χρονική στιγμή μηδέν ο κινητήρας είναι ακίνητος και δεν διαρρέεται από ρεύμα οπότε έχουμε μηδενικές αρχικές συνθήκες.

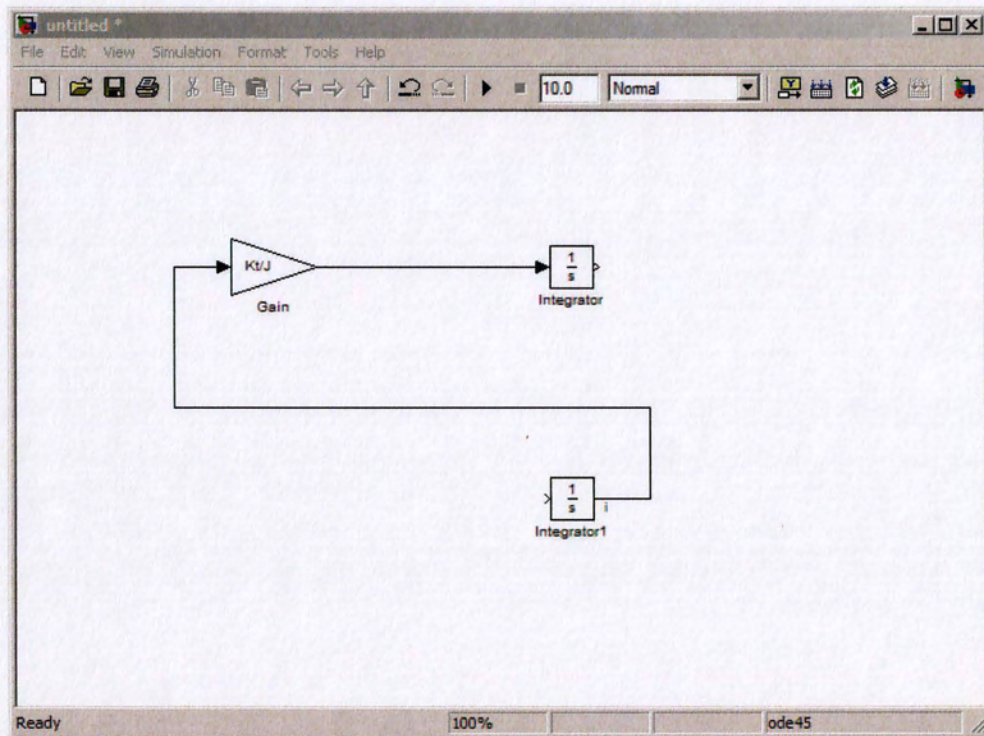
Τώρα είμαστε έτοιμοι να εισάγουμε το μοντέλο στο Simulink. Για να γίνει αυτό πρέπει να “χτίσουμε” τις εξισώσεις από τον κάθε ένα όρο ξεχωριστά. Αρχικά θα χρειαστούμε 2 ολοκληρωτές όπως φαίνεται στο σχήμα.



Η έξοδος του πρώτου είναι το $\omega(t)$ και η έξοδος του δεύτερου είναι το $i(t)$.

Από τη σχέση (4) βλέπουμε ότι η είσοδος στον πρώτο ολοκληρωτή πρέπει να είναι: $(Kt/J) * i(t)$

Άρα παίρνουμε το σήμα εξόδου του δεύτερου ολοκληρωτή που είναι το $i(t)$, το πολλαπλασιάζουμε με Kt/J , και το στέλνουμε για είσοδο στον πρώτο ολοκληρωτή όπως φαίνεται στο επόμενο σχήμα.



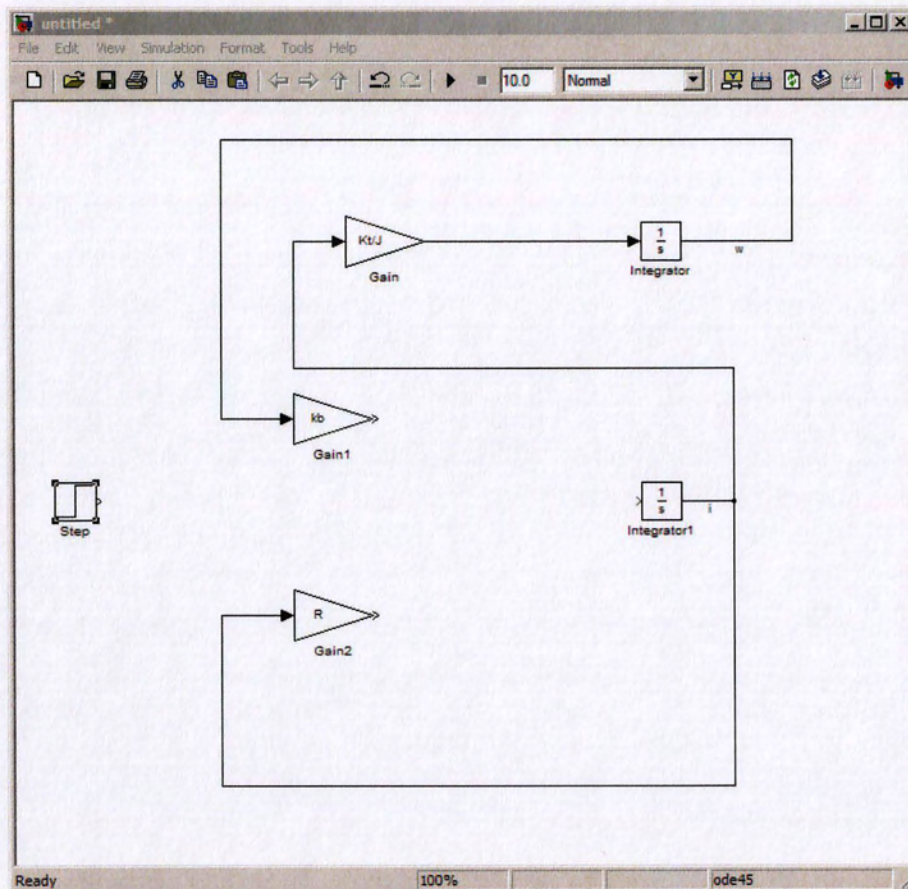
Τώρα για τον δεύτερο ολοκληρωτή, από τη σχέση (5) φαίνεται ότι η είσοδος του πρέπει να είναι:

$$\frac{1}{L} [V_a - i(t) * R - kb * \omega(t)]$$

Δηλαδή εδώ έχουμε 3 όρους που αθροίζονται.

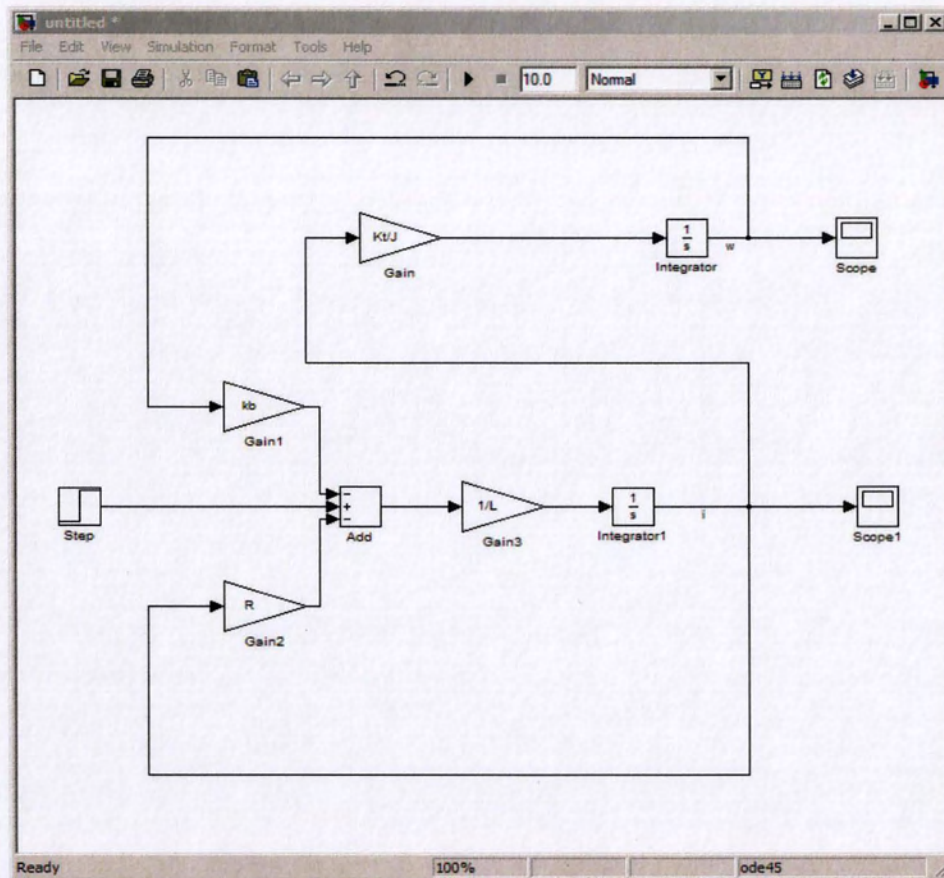
Το V_a είναι η σταθερή τάση που εφαρμόζουμε στα άκρα του κινητήρα, άρα προσθέτουμε μια συνάρτηση βαθμίδας για να μοντελοποιήσουμε αυτόν τον όρο.

Οι άλλοι 2 όροι είναι οι εξοδοι των 2 ολοκληρωτών πολλαπλασιασμένοι με κάποιες σταθερές, όπως φαίνεται στο σχήμα.



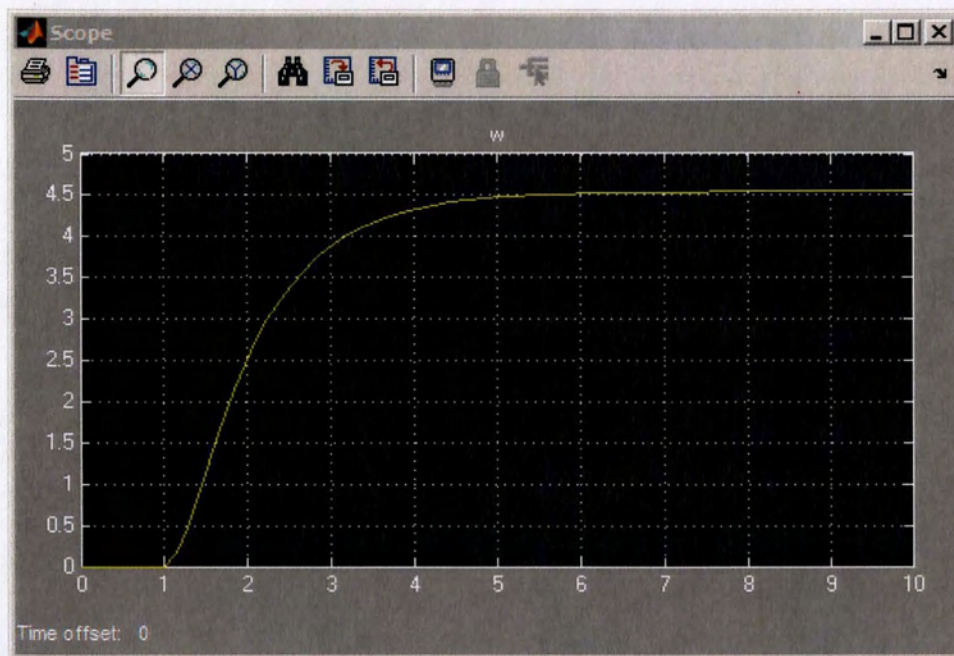
Τώρα το μόνο που απομένει είναι να αθροίσουμε αυτούς τους όρους κατάλληλα και να στείλουμε το σήμα για είσοδο στον 2^ο ολοκληρωτή.

Για να το κάνουμε αυτό χρησιμοποιούμε έναν αθροιστή με τα κατάλληλα πρόσημα. Αφού το κάνουμε αυτό έχουμε ολοκληρώσει το σύστημα των διαφορικών εξισώσεων και το μόνο που μένει είναι να δούμε την λύση. Για το λόγο αυτό προσθέτουμε από ένα Scope στον κάθε άγνωστο (ω και i) για να δούμε τη γραφική του παράσταση. Με αυτές τις προσθήκες το μοντέλο έχει ως εξής:

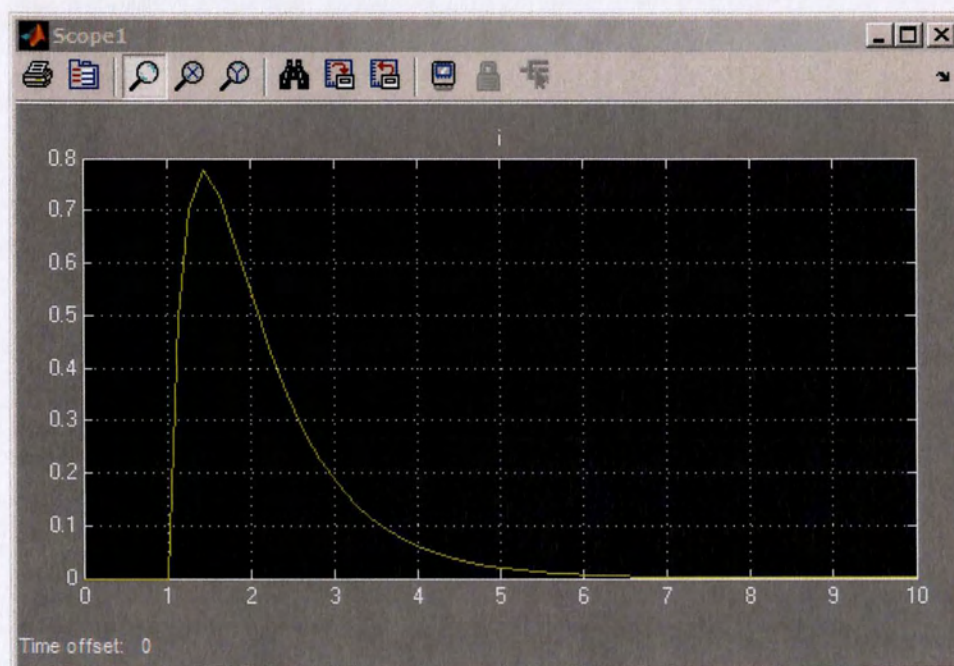


Αυτή είναι η τελική μορφή του μοντέλου για τον κινητήρα. Δίνοντας τιμές στις διάφορες παραμέτρους μπορούμε να το τρέξουμε και να δούμε τι αποκρίσεις παίρνουμε τόσο για το ρεύμα όσο και για τη γωνιακή ταχύτητα.

Όπως αναφέρθηκε και προηγουμένως, η μοντελοποίηση που χρησιμοποιείται σε όλη την έκταση της εργασίας είναι η μέθοδος του μετασχηματισμού Laplace. Όμως είναι σκόπιμο σε αυτό το σημείο να δούμε ένα παράδειγμα απόκρισης της παραπάνω μεθόδου (στο πεδίο του χρόνου), γιατί μας βοηθάει να καταλάβουμε μερικά σημαντικά χαρακτηριστικά των κινητήρων συνεχούς ρεύματος.



Σχήμα 3-3: Γωνιακή ταχύτητα σε συνάρτηση με το χρόνο.



Σχήμα 3-4: Ένταση ρεύματος σε συνάρτηση με το χρόνο.

Στα παραπάνω σχήματα φαίνονται οι χρονικές αποκρίσεις της γωνιακής ταχύτητας και της έντασης του ρεύματος σε είσοδο τάσης 1 Volt τη χρονική στιγμή $t=1$ sec.

Παρατηρούμε ότι η γωνιακή ταχύτητα ξεκινάει να αυξάνεται τη χρονική στιγμή που εφαρμόζεται η τάση και συνεχίζει να αυξάνεται μέχρι να φτάσει σε μια οριακή τιμή η οποία είναι 4.5 rad/s.

Η ένταση του ρεύματος ξεκινάει επίσης να αυξάνεται τη χρονική στιγμή $t=1$ sec, φτάνει σε μια οριακή τιμή περίπου 0.8A, και στη συνέχεια πέφτει μέχρι το 0. Ίσως φαίνεται παράξενο το ότι ο άξονας περιστρέφεται με σταθερή ταχύτητα χωρίς να διαρρέεται από ρεύμα, αλλά πρέπει να υπενθυμίσουμε σε αυτό το σημείο ότι στη συγκεκριμένη μοντελοποίηση δεν έχουμε λάβει υπόψη τη ροπή που προκαλούν οι τριβές του κινητήρα. Στην πραγματικότητα, όταν ο άξονας περιστρέφεται με σταθερή ταχύτητα υπάρχει πάντα κάποιο ρεύμα που δίνει τόση ροπή όση προκαλούν οι τριβές, έτσι ώστε η συνολική ροπή να είναι μηδέν.

Επίσης έχει ενδιαφέρον να εξετάσουμε γιατί ο κινητήρας (έστω και χωρίς τριβές) φτάνει μια οριακή ταχύτητα. Αυτό φαίνεται ότι είναι ένα γενικό χαρακτηριστικό των κινητήρων συνεχούς ρεύματος. Η σειρά των γεγονότων έχει ως εξής:

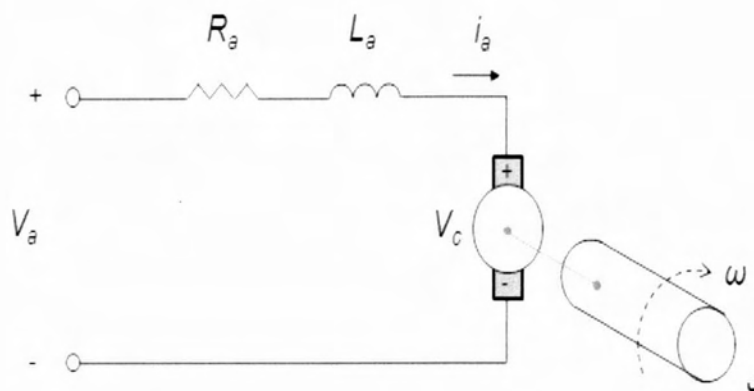
Τη χρονική στιγμή $t=1$ sec εφαρμόζεται τάση ίση με 1 Volt στον κινητήρα. Τα τυλίγματα διαρρέονται από ρεύμα και αυτό έχει σαν αποτέλεσμα την ανάπτυξη ροπής σύμφωνα με τη σχέση:

$$T = Kt * i(t)$$

Αυτή η ροπή με τη σειρά της οδηγεί σε ανάπτυξη γωνιακής επιτάχυνσης του άξονα σύμφωνα με τη σχέση:

$$Kt * i(t) = J * \frac{d\omega}{dt}$$

Άρα θα έπρεπε να έχουμε διαρκώς επιτάχυνση. Αυτό όμως δεν συμβαίνει γιατί καθώς η γωνιακή ταχύτητα αυξάνεται, αυξάνεται επίσης και η ηλεκτρεγερτική δύναμη η οποία είναι ανάλογη του ω με αποτέλεσμα η τάση που κινεί τον κινητήρα συνεχώς να μειώνεται. Αυτό φαίνεται από τη σχέση του νόμου του Kirchhoff για τις τάσεις και το κύκλωμα του κινητήρα.



$$V_a - L \frac{di(t)}{dt} - i(t) \cdot R - kb \cdot \omega(t) = 0$$

Καθώς το ω αυξάνεται, για να ισχύει η παραπάνω σχέση μειώνεται το ρεύμα. Έτσι φτάνουμε σε μια οριακή κατάσταση όπου το ω έχει γίνει τόσο μεγάλο ώστε το ρεύμα έχει πρακτικά μηδενιστεί. Έτσι μηδενίζεται και η ροπή άρα σταματάει να αυξάνεται και το ω . Η τιμή του ω για την οποία συμβαίνει αυτό είναι η οριακή ταχύτητα.

Βέβαια όπως είπαμε και προηγουμένως, η τιμή του ρεύματος δεν μηδενίζεται ποτέ στην πραγματικότητα. Πάντα απαιτείται κάποιο ρεύμα για να συνεχίσει ο άξονας να περιστρέφεται με σταθερή ταχύτητα. Αυτό συνέβη γιατί δεν υπολογίσαμε τις τριβές. Στην επόμενη μέθοδο υπολογίζουμε και τη ροπή που προκαλούν οι τριβές και τα αποτελέσματα είναι διαφορετικά.

3.3 Μοντελοποίηση στο πεδίο της συχνότητας

Στόχος μας εδώ είναι να βρούμε τη συνάρτηση μεταφοράς του συστήματός μας. Η συνάρτηση μεταφοράς ορίζεται ως ο λόγος του μ/χ Laplace της εξόδου του συστήματος ως προς τον μ/χ Laplace της εισόδου.

$$H(s) = \frac{Out(s)}{In(s)}$$

Είσοδος στο σύστημά μας είναι πάντα η τάση που εφαρμόζεται στα άκρα του κινητήρα.

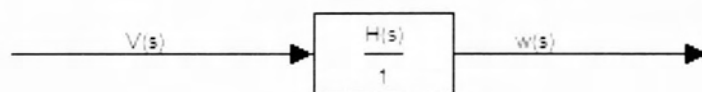
Έξοδος είναι η γωνία στροφής του άξονα ή η γωνιακή ταχύτητα. Αυτό το καθορίζουμε εμείς ανάλογα με το πως θα γράψουμε τις εξισώσεις, και ανάλογα με το τι θέλουμε να ελέγξουμε όπως θα δούμε στη συνέχεια. Για τώρα θεωρούμε σαν έξοδο τη γωνιακή ταχύτητα.

Είναι πολύ εύκολο να αποκτήσουμε τη σ.μ. ως προς την γωνία αν την έχουμε ως προς την ταχύτητα και το αντίστροφο γιατί στο πεδίο της συχνότητας η διαφορίση και η ολοκλήρωση είναι απλά πολλαπλασιασμός και διαίρεση με το s αντίστοιχα.

Άρα θέλουμε να υπολογίσουμε το λόγο:

$$H(s) = \frac{\Omega(s)}{V(s)}$$

Όπου $\Omega(s)$ και $V(s)$ οι μετασχηματισμοί Laplace της γωνιακής ταχύτητας του άξονα και της τάσης που εφαρμόζεται στον κινητήρα αντίστοιχα. Το σύστημά μας απεικονίζεται στο παρακάτω σχήμα.



Σχήμα 3-5: Συνάρτηση μεταφοράς γωνιακής ταχύτητας-τάσης.

Για να σχηματίσουμε τον επιθυμητό λόγο χρειαζόμαστε τις 2 εξισώσεις που χρησιμοποιήσαμε στο προηγούμενο κεφάλαιο. Αυτές είναι οι εξής:

$$Va - L * \frac{di(t)}{dt} - i(t) * R - kb * \omega(t) = 0$$

και

$$Kt * i(t) - b * \omega(t) = J * \frac{d\omega}{dt}$$

Μετασχηματίζοντας κατά Laplace την πρώτη εξίσωση έχουμε:

$$V(s) - L * s * I(s) - R * I(s) - kb * \Omega(s) = 0$$

και από την δεύτερη έχουμε:

$$kt * I(s) - b * \Omega(s) = J * s * \Omega(s)$$

Κανονικά κατά το μετασχηματισμό Laplace των παραγώγων πρέπει να συμπεριλάβουμε και τις αρχικές συνθήκες. Όμως η συνάρτηση μεταφοράς είναι εξ ορισμού ο λόγος εξόδου/εισόδου για μηδενικές αρχικές συνθήκες, οπότε δεν χρειάζεται καμία αλλαγή.

Για να σχηματίσουμε τον λόγο $\frac{\Omega(s)}{V(s)}$, πρώτα λύνουμε την δεύτερη σχέση ως προς $I(s)$ και αντικαθιστούμε στην πρώτη ώστε να το απαλείψουμε.

$$kt * I(s) - b * \Omega(s) = J * s * \Omega(s) \Rightarrow kt * I(s) = J * s * \Omega(s) + b * \Omega(s) \Rightarrow I(s) = \frac{J * s * \Omega(s) + b * \Omega(s)}{kt}$$

$$\Rightarrow I(s) = \frac{(J * s + b) * \Omega(s)}{kt}$$

Τώρα αντικαθιστούμε στην πρώτη και έχουμε:

$$V(s) - L * s * \frac{(J * s + b) * \Omega(s)}{kt} - R * \frac{(J * s + b) * \Omega(s)}{kt} - kb * \Omega(s) = 0$$

$$\Rightarrow V(s) - \Omega(s) * [L * s * \frac{(J * s + b)}{kt} + R * \frac{(J * s + b)}{kt} + kb] = 0$$

$$\Rightarrow V(s) = \Omega(s) * [L * s * \frac{(J * s + b)}{kt} + R * \frac{(J * s + b)}{kt} + kb]$$

$$\begin{aligned}
\Rightarrow \frac{\Omega(s)}{V(s)} * [L * s * \frac{(J * s + b)}{kt} + R * \frac{(J * s + b)}{kt} + kb] &= 1 \\
\Rightarrow \frac{\Omega(s)}{V(s)} &= \frac{1}{[L * s * \frac{(J * s + b)}{kt} + R * \frac{(J * s + b)}{kt} + kb]} \\
\Rightarrow \frac{\Omega(s)}{V(s)} &= \frac{1}{\frac{L * J * s^2 + L * b * s}{kt} + \frac{R * J * s + R * b}{kt} + kb} \\
\Rightarrow \frac{\Omega(s)}{V(s)} &= \frac{kt}{L * J * s^2 + L * b * s + R * J * s + R * b + kb * kt} \\
\Rightarrow \frac{\Omega(s)}{V(s)} &= \frac{kt}{L * J * s^2 + (L * b + R * J) * s + R * b + kb * kt}
\end{aligned}$$

Η τελευταία σχέση είναι η συνάρτηση μεταφοράς ταχύτητας-τάσης. Η συνάρτηση μεταφοράς θέσης-τάσης προκύπτει πολύ εύκολα εάν ολοκληρώσουμε. Η ολοκλήρωση στο πεδίο της συχνότητας είναι η διαίρεση με το s . Άρα η συνάρτηση μεταφοράς θέσης-τάσης είναι η:

$$\frac{\Theta(s)}{V(s)} = \frac{kt}{L * J * s^3 + (L * b + R * J) * s^2 + (R * b + kb * kt) * s}$$

Παρατηρούμε ότι ο βαθμός του πολυωνύμου του παρανομαστή έγινε 3. Κάτι τέτοιο δεν είναι επιθυμητό όπως θα διαπιστώσουμε και στη συνέχεια.

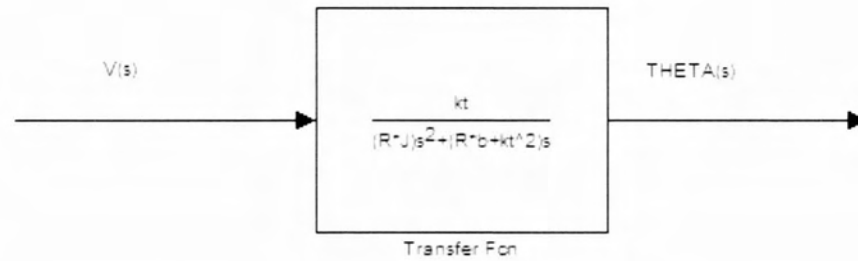
Επίσης, η ηλεκτρική σταθερά L είναι κατά πολύ μικρότερη από την μηχανική σταθερά k_t . Άρα μπορούμε να θεωρήσουμε $L=0$ χωρίς να έχουμε σημαντικό σφάλμα.

Αυτή είναι μια αποδεκτή απλοποίηση γιατί μας επιτρέπει να μειώσουμε την τάξη του συστήματος κατά ένα, πράγμα που κάνει το μοντέλο πολύ πιο εύκολο στην ανάλυση, έχοντας ένα αμελητέο σφάλμα όσον αφορά την απόκριση του μοντέλου και του πραγματικού συστήματος. Επίσης πρέπει να σημειώσουμε ότι η σταθερά k_b είναι αριθμητικά ίση με την k_t . Άρα τελικά οι συναρτήσεις μεταφοράς και τα αντίστοιχα δομικά διαγράμματα είναι:

Συνάρτηση μεταφοράς θέσης-τάσης.

$$\frac{\Theta(s)}{V(s)} = \frac{kt}{R * J * s^2 + (R * b + kt^2) * s}$$

Και το αντίστοιχο δομικό διάγραμμα:

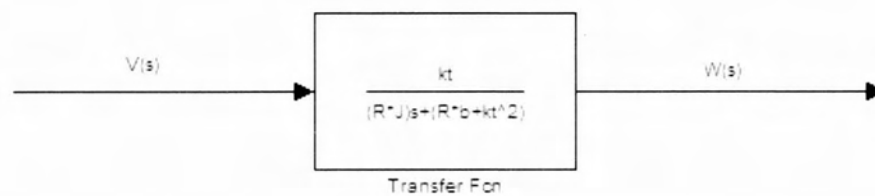


Σχήμα 3-6: Δομικό διάγραμμα θέσης-τάσης.

Συνάρτηση μεταφοράς ταχύτητας-τάσης.

$$\frac{\Omega(s)}{V(s)} = \frac{kt}{R * J * s + R * b + kt^2}$$

Και το αντίστοιχο δομικό διάγραμμα:



Σχήμα 3-7: Δομικό διάγραμμα ταχύτητας-τάσης.

Συνοψίζοντας, μέχρι τώρα διατυπώσαμε τις εξισώσεις που περιγράφουν τον κινητήρα και με βάση αυτές είδαμε πως γίνεται η μοντελοποίηση με δύο διαφορετικούς τρόπους. Δεν έχουμε μιλήσει ακόμα για ανοιχτό ή κλειστό βρόχο και έλεγχο.

3.4 Σύγκριση μοντέλου με το πραγματικό σύστημα

Τελικός στόχος είναι ο έλεγχος του πραγματικού συστήματος. Όμως για να επιτευχθεί αυτό πρέπει πρώτα να γίνει έλεγχος στο μοντέλο. Εάν το μοντέλο είναι αρκετά ακριβές, τότε ελέγχοντας το θα μπορούμε πολύ εύκολα να ελέγχουμε και το πραγματικό σύστημα. Άρα πρέπει να κλείσουμε το βρόχο και να εισάγουμε τον ελεγκτή. Όμως πριν γίνει αυτό πρέπει πρώτα να διαπιστωθεί η ορθότητα του μοντέλου. Εάν αποδειχτεί ότι αυτό προβλέπει με ικανοποιητική ακρίβεια τη συμπεριφορά του πραγματικού συστήματος τότε μπορούμε να σχεδιάσουμε τον ελεγκτή ανάλογα με την επιθυμητή απόκριση, και να μεταφέρουμε τα αποτελέσματα στο πραγματικό σύστημα. Βέβαια η απόκριση του πραγματικού συστήματος δεν θα ταυτίζεται ποτέ με αυτή του μοντέλου για λόγους που θα αναλυθούνε στη συνέχεια. Αλλά το σφάλμα όπως θα δούμε είναι αρκετά μικρό ώστε να γίνει αποδεκτό.

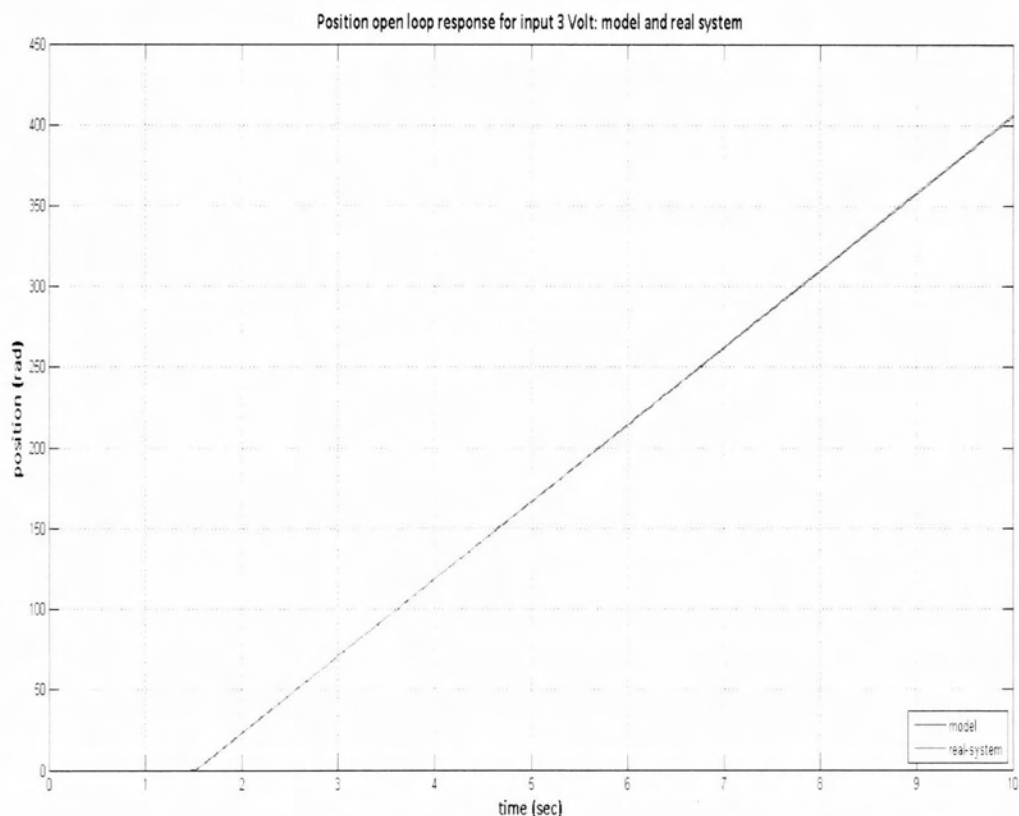
Προκειμένου να γίνει ο έλεγχος για την ορθότητα του μοντέλου πρέπει να γίνουν κάποιες δοκιμές. Θα δώσουμε μια γνωστή είσοδο στο μοντέλο και στο πραγματικό σύστημα και θα συγκρίνουμε τις αποκρίσεις. Εάν αυτές οι αποκρίσεις δεν διαφέρουν πολύ μεταξύ τους τότε σημαίνει πως το μοντέλο είναι αξιόπιστο και μπορεί να χρησιμοποιηθεί για τον έλεγχο του πραγματικού συστήματος.

Η συνάρτηση μεταφοράς που βρήκαμε, πάνω στην οποία θα βασιστεί η μοντελοποίηση είναι:

$$\frac{\Theta(s)}{V(s)} = \frac{kt}{R * J * s^2 + (R * b + kt^2) * s}$$

Υπενθυμίζουμε ότι σε αυτή τη συνάρτηση μεταφοράς είσοδος είναι η τάση και έξοδος είναι η γωνία στροφής του άξονα. Θα δώσουμε μία δοκιμαστική είσοδο 3 Volt τόσο στην παραπάνω συνάρτηση μεταφοράς όσο και στο πραγματικό σύστημα. (Τα προγράμματα που χρησιμοποιούνται

στο Simulink τόσο για το θεωρητικό όσο και για το πραγματικό μοντέλο υπάρχουν στο συνοδευτικό cd.) Η απόκριση που παίρνουμε είναι:



Σχήμα 3-8: Απόκριση θέσης μοντέλου και πραγματικού συστήματος σε είσοδο τάσης 3 Volt.

Δηλαδή τη χρονική στιγμή $t=1.5s$ εφαρμόζουμε μια σταθερή τάση και ο κινητήρας περιστρέφεται με μια σταθερή ταχύτητα περίπου 50 rad/s. Παρατηρούμε ότι υπάρχει ταύτιση της απόκρισης του μοντέλου και του πραγματικού συστήματος. Άρα η συνάρτηση μεταφοράς του κινητήρα μπορεί να χρησιμοποιηθεί για τον έλεγχο του πραγματικού συστήματος.

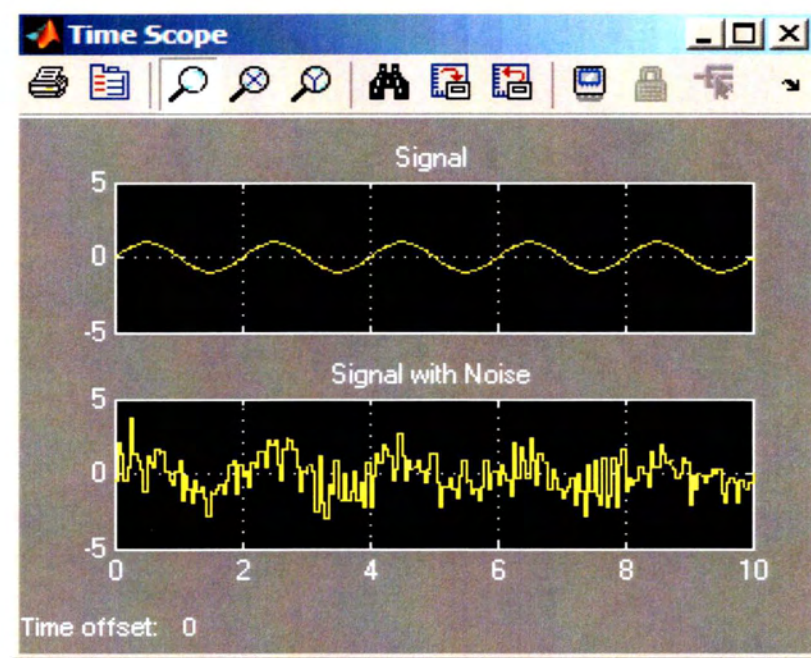
Όμως για να κάνουμε έλεγχο ταχύτητας δεν μας αρκεί η συνάρτηση μεταφοράς θέσης-τάσης. Μια πρώτη προσπάθεια θα ήταν να χρησιμοποιήσουμε τη συνάρτηση μεταφοράς ταχύτητας-τάσης:

$$\frac{\Omega(s)}{V(s)} = \frac{kt}{R * J * s + R * b + kt^2}$$

Όμως κάτι τέτοιο δεν θα ήταν σωστό γιατί θα υπονοούσε ότι η έξοδος του κινητήρα είναι η γωνιακή ταχύτητα. Σε αυτό το σημείο πρέπει να σκεφτούμε ποιό ακριβώς είναι το πραγματικό μας σύστημα και σε τι φυσικό μέγεθος αντιστοιχεί το κάθε κομμάτι του hardware.

Υπενθυμίζουμε ότι ο κινητήρας στο πίσω μέρος του έχει ενσωματωμένο αισθητήρα θέσης (encoder) που μετράει τη γωνία στροφής του άξονα. Αυτό το σήμα το στέλνουμε στον υπολογιστή μέσω του connector block και της κάρτας E/E και με κατάλληλο μετασχηματισμό το μετατρέπουμε σε μοίρες ή rad. Άρα το πραγματικό σήμα εξόδου του κινητήρα είναι η γωνία και όχι η ταχύτητα. Θα ήταν η ταχύτητα εάν αντί για αισθητήρα θέσης ο κινητήρας είχε ενσωματωμένο αισθητήρα ταχύτητας. Άρα θα κρατήσουμε τη συνάρτηση μεταφοράς $\Theta(s)/V(s)$ που αντιπροσωπεύει την πραγματική έξοδο του συστήματος.

Για να μπορέσουμε όμως να κάνουμε έλεγχο ταχύτητας πρέπει να μετατρέψουμε τη γωνία που λαμβάνουμε σε ταχύτητα. Αυτό γίνεται με παραγωγή του σήματος. Γενικά η παραγωγή ενός σήματος πραγματικού χρόνου είναι μια δύσκολη υπόθεση γιατί τα πραγματικά σήματα περιέχουν αυτό που αποκαλούμε "θόρυβο". Δηλαδή απότομες μεταβολές στις τιμές του μετρούμενου μεγέθους που οφείλονται μεταξύ άλλων στο πεπερασμένο χρονικό διάστημα ανάμεσα σε δύο διαδοχικές μετρήσεις καθώς και στην ακρίβεια του αισθητήρα που πραγματοποιεί τη μέτρηση. Το πρόβλημα του θορύβου γίνεται ακόμα μεγαλύτερο όταν το σήμα δεχτεί παραγωγή. Η παραγωγή προκαλεί ακόμα περισσότερες αυξομειώσεις γιατί παραγωγίζεται και ο θόρυβος ταυτόχρονα οπότε στα σημεία όπου έχουμε πολύ απότομες μεταβολές η τιμή της παραγωγού γίνεται πολύ μεγάλη. Στο επόμενο σχήμα φαίνεται ένα σήμα με θόρυβο και ένα χωρίς θόρυβο.



Σχήμα 3-9: Σήμα με θόρυβο και χωρίς θόρυβο.

Είναι προφανές ότι δεν μπορούμε να κλείσουμε τον βρόχο έχοντας στον κλάδο ανατροφοδότησης ένα σήμα με θόρυβο. Αυτό γιατί ο θόρυβος θα “μεταφερθεί” στο σφάλμα και από εκεί στον ελεγκτή και στην τάση εισόδου του κινητήρα.

Ο τρόπος να μειώσουμε τον θόρυβο σε αποδεκτό επίπεδο είναι να χρησιμοποιήσουμε για την διαφόριση όχι τη συνάρτηση μεταφοράς s αλλά $Ns/s+N$. Η προσθήκη του πόλου φιλτράρει το σήμα πριν τη διαφόριση και μειώνει την επίδραση του θορύβου. Η χρονική σταθερά N καθορίζει την ταχύτητα της παραγωγίσης και συνεπώς το φιλτράρισμα που δέχεται το σήμα. Όσο περισσότερο θόρυβο έχει ένα σήμα τόσο μικρότερη πρέπει να είναι αυτή η σταθερά. Στην συγκεκριμένη εργασία έγιναν δοκιμές για διάφορες τιμές του N μέχρι να διαπιστωθεί ικανοποιητική ταύτιση μεταξύ του θεωρητικού μοντέλου και του μοντέλου για το πραγματικό σύστημα. Η τιμή του N που καταλήξαμε για το μοντέλο του πραγματικού συστήματος είναι 2.7 ενώ για το θεωρητικό μοντέλο 8.

Τώρα το μόνο που απομένει για να κλείσουμε τον βρόχο είναι να εισάγουμε τον ενισχυτή και τον ελεγκτή. Ο ενισχυτής μοντελοποιείται ως ένα απλό κέρδος που αυξάνει την τάση εισόδου του κινητήρα. Αυτό το κέρδος έχει βρεθεί από πειράματα ότι είναι 1.25 όταν χρησιμοποιούμε τον

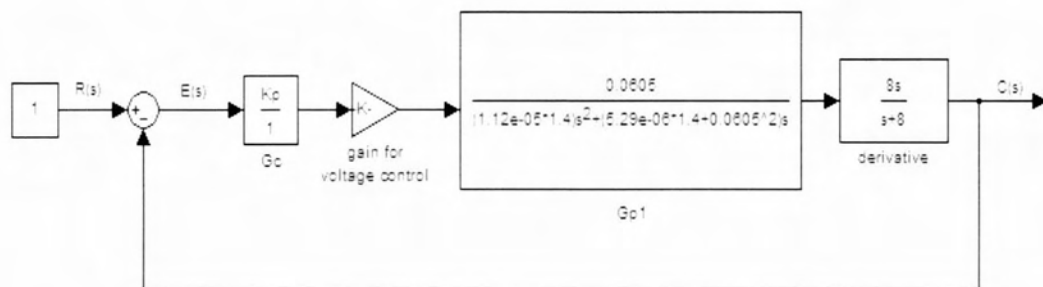
ενισχυτή για έλεγχο τάσης (όπως δηλαδή τον χρησιμοποιούμε στην παρούσα εργασία). Όσο για τον ελεγκτή, αρχικά θεωρούμε μόνο αναλογικό έλεγχο.

Έχοντας πλέον στη διάθεση μας τη συνάρτηση μεταφοράς του κινητήρα, της διαφορίσης, και του ενισχυτή, μπορούμε να κλείσουμε το βρόχο για να πετύχουμε τον έλεγχο της ταχύτητας. Αυτό είναι και το θέμα του επόμενου κεφαλαίου.

Κεφάλαιο 4 : Έλεγχος ταχύτητας με αναλογικό ελεγκτή

4.1 Ανάλυση του συστήματος κλειστού βρόχου

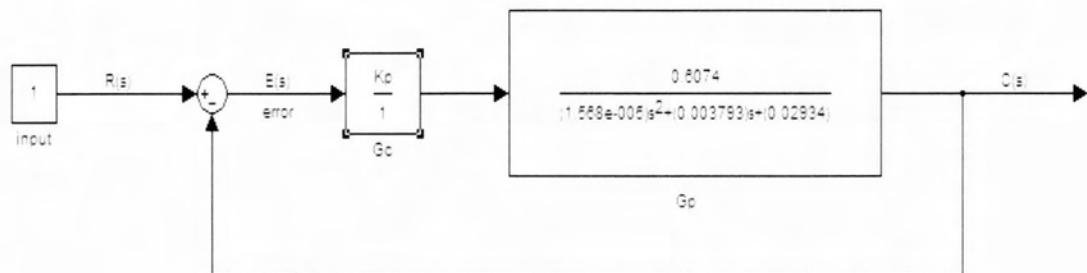
Στο παρακάτω σχήμα βλέπουμε πως γίνεται το δομικό διάγραμμα του συστήματος μας όταν κλείσουμε τον βρόχο.



Σχήμα 4-1: Δομικό διάγραμμα συστήματος κλειστού βρόχου

Τώρα πια η είσοδος στο σύστημα δεν είναι η τάση αλλά η επιθυμητή ταχύτητα. Αυτή συγκρίνεται με την πραγματική ταχύτητα εξόδου και η διαφορά τους αποτελεί το σφάλμα. Το σφάλμα με τη σειρά του αποτελεί το σήμα εισόδου του ελεγκτή ενώ η έξοδος του ελεγκτή είναι η τάση που θα τροφοδοτήσει τον κινητήρα. Βέβαια πριν η τάση τροφοδοτήσει τον κινητήρα περνάει πρώτα από τον ενισχυτή ο οποίος την αυξάνει κατά ένα παράγοντα 1.25. Στη συνέχεια η τάση αυτή αποτελεί είσοδο της συνάρτησης μεταφοράς θέσης-τάσης. Η θέση (δηλαδή γωνία) στη συνέχεια διαφορίζεται με την κατάλληλη συνάρτηση μεταφοράς ώστε να πάρουμε την ταχύτητα.

Για να βγάλουμε τη συνάρτηση μεταφοράς κλειστού βρόχου απλοποιούμε το παραπάνω διάγραμμα πολλαπλασιάζοντας τις 3 τελευταίες συναρτήσεις μεταφοράς (κέρδος ενισχυτή, σ.μ. θέσης/τάσης, σ.μ. παραγωγίσης) μεταξύ τους. Έτσι καταλήγουμε στο εξής απλοποιημένο δομικό διάγραμμα του σχήματος 2.



Σχήμα 4-2: Απλοποιημένο δομικό διάγραμμα συστήματος

Η συνάρτηση μεταφοράς εισόδου-εξόδου κλειστού βρόχου του συστήματος είναι:

$$\frac{C(s)}{R(s)} = \frac{G_c * G_p}{1 + G_c * G_p} = \frac{K_p * G_p}{1 + K_p * G_p}$$

Για λόγους ευκολίας από 'δω και πέρα θα αναφερόμαστε στην G_p ως $G_p = \frac{a}{bs^2 + cs + d}$

όπου:

$$a=0.6074, b=1,568*10^{-5}, c=0.003793, d=0.02934$$

Άρα τώρα η Σ.Μ. κλειστού βρόχου γίνεται:

$$\frac{C(s)}{R(s)} = \frac{\frac{K_p * a}{b * s^2 + c * s + d}}{1 + \frac{K_p * a}{b * s^2 + c * s + d}} = \frac{K_p * a}{b * s^2 + c * s + d + K_p * a} = \frac{\frac{K_p * a}{b}}{s^2 + \frac{c}{b} * s + \frac{d + K_p * a}{b}}$$

Συγκρίνουμε με την κλασική μορφή συναρτήσεων μεταφοράς συστημάτων 2^{ης} τάξης:

$\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$. Οπότε, από την χαρακτηριστική εξίσωση του συστήματος βρίσκουμε

τις παραμέτρους ζ και ω_n .

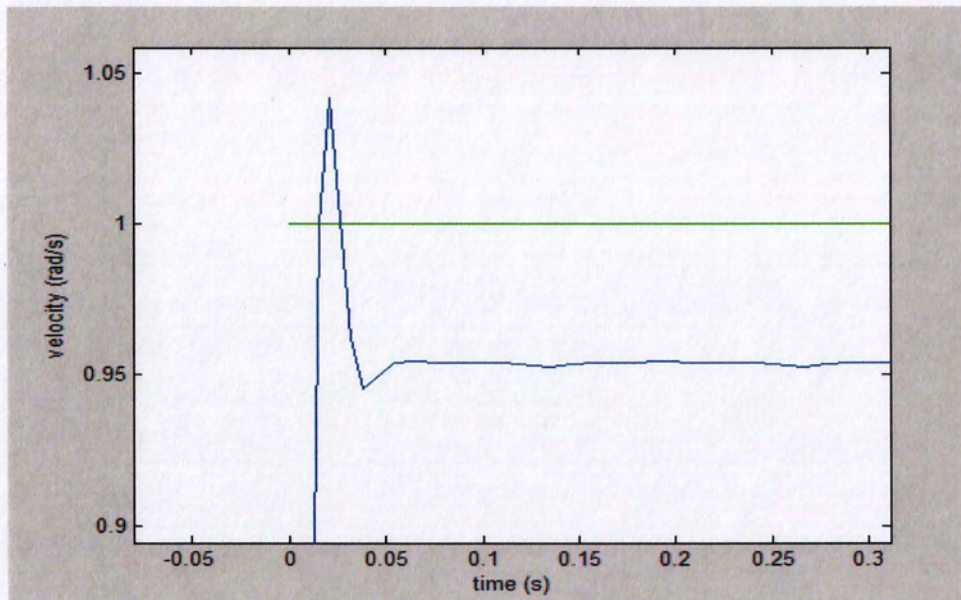
$$\text{Χαρακτηριστική εξίσωση: } s^2 + \frac{c}{b}s + \frac{d + Kp^*a}{b} = 0$$

$$\omega_n^2 = \frac{d + Kp^*a}{b} \Rightarrow \omega_n = \sqrt{\frac{d + Kp^*a}{b}} \quad (1)$$

$$2\zeta\omega_n = \frac{c}{b} \Rightarrow \zeta = \frac{c}{2b\omega_n} \Rightarrow \zeta = \frac{c}{2\sqrt{b(d + Kp^*a)}} \quad (2)$$

Αφού τα a, b, c, d είναι γνωστές σταθερές, από τις παραπάνω σχέσεις μπορούμε να βρίσκουμε τι ζ και ω_n αντιστοιχεί σε κάθε Kp ή και το αντίστροφο.

Το ζ έχει να κάνει με τη θέση των πόλων στο μιγαδικό επίπεδο και συνεπώς καθορίζει τη μεταβατική απόκριση και την ευστάθεια. Για παράδειγμα, για $Kp=1$, έχουμε από την παραπάνω σχέση $\zeta=0.6$. Δηλαδή οι ρίζες της Χ.Ε. (χαρακτηριστικής εξίσωσης) είναι συζυγείς μιγαδικοί με αρνητικά πραγματικά μέρη, πράγμα που σημαίνει ότι έχουμε ευσταθές σύστημα με ταλαντώσεις στη μεταβατική του κατάσταση. Αυτό επιβεβαιώνεται από την εκτέλεση του μοντέλου του σχήματος 2 στο Simulink, με $Kp=1$. Η απόκριση που παίρνουμε φαίνεται στο παρακάτω σχήμα.

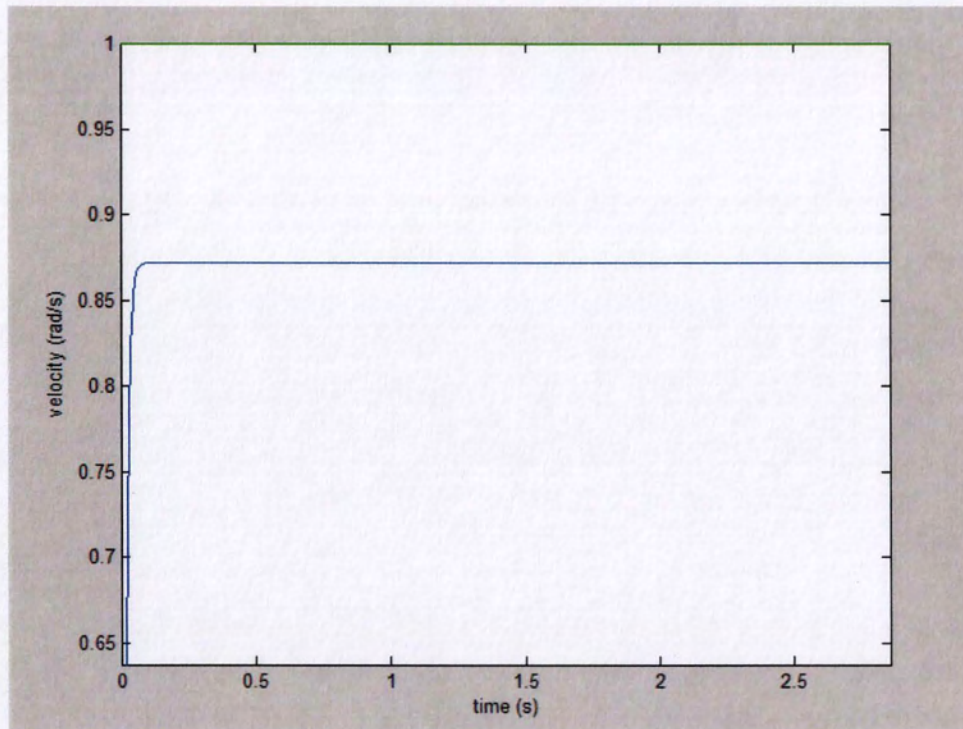


Σχήμα 4-3: Απόκριση μοντέλου για $K_r=1$

Η είσοδος είναι πάντα μοναδιαία συνάρτηση βαθμίδας και απεικονίζεται με την πράσινη γραμμή του σχήματος 3.

Παρατηρούμε ότι υπάρχει ένα σφάλμα μόνιμης κατάστασης της τάξης του 0.05 περίπου. Αυτό είναι αναμενόμενο γιατί το σύστημα μας είναι τύπου 0, δηλαδή η G_p δεν έχει ελεύθερο ολοκληρωτή. (Δεν βγαίνει κοινός παράγοντας το $1/s$). Οπότε σύμφωνα με τη θεωρία Αυτομάτου Ελέγχου, θα παρουσιάζει πεπερασμένο σφάλμα μόνιμης κατάστασης σε είσοδο μοναδιαίας συνάρτησης βαθμίδας. Αυτό το σφάλμα θα το υπολογίσουμε αναλυτικά στην επόμενη ενότητα. Κλείνοντας αυτή την ενότητα θα δούμε την περίπτωση που δεν θέλουμε καθόλου ταλαντώσεις στη μεταβατική μας απόκριση.

Σύμφωνα με τη θεωρία των συστημάτων $2^{\text{ης}}$ τάξης, ξέρουμε ότι για $\zeta=1$ οι ρίζες της χ.ε. είναι αρνητικοί πραγματικοί αριθμοί, δηλαδή δεν έχουν φανταστικό μέρος. Αυτό οδηγεί σε μια απόκριση χωρίς καθόλου ταλαντώσεις ή υπερακόντιση στη μεταβατική κατάσταση. Άρα για να πετύχουμε μια τέτοια απόκριση πρέπει να δώσουμε τέτοιο K_r στο σύστημά μας ώστε το ζ να γίνει 1. Από τη σχέση μεταξύ των K_r και ζ έχουμε για $\zeta=1$, $K_r=0.3293$. Η απόκριση που παίρνουμε για αυτό το K_r φαίνεται στο παρακάτω σχήμα.



Σχήμα 4-4: Απόκριση μοντέλου για $K_p=0.3293$

Όπως φαίνεται, δεν υπάρχουν καθόλου ταλαντώσεις στη μεταβατική κατάσταση. Το αντίστοιχο όμως είναι η εμφανής αύξηση του σφάλματος μόνιμης κατάστασης. Αυτό είναι και το θέμα της επόμενης παραγράφου.

4.2 Σφάλμα μόνιμης κατάστασης και επίδραση του συντελεστή ζ στην απόκριση του συστήματος

Για να υπολογίσουμε το σφάλμα μόνιμης κατάστασης (μ.κ.) πρέπει να μελετήσουμε τη συνάρτηση μεταφοράς σφάλματος. Από το δομικό διάγραμμα του σχήματος 2 έχουμε:

$$\frac{E(s)}{R(s)} = \frac{R(s) - C(s)}{R(s)} = 1 - \frac{C(s)}{R(s)} = 1 - \frac{GcGp}{1 + GcGp} = \frac{1 + GcGp}{1 + GcGp} - \frac{GcGp}{1 + GcGp} = \frac{1}{1 + GcGp}$$

$$\text{Οπότε } E(s) = \frac{1}{1 + GcGp} * R(s)$$

Το σφάλμα μ.κ. είναι στην ουσία η τιμή που παίρνει η χρονική συνάρτηση του σφάλματος $e(t)$ όταν το t τείνει στο άπειρο. Δηλαδή πρέπει να βρούμε το όριο $\lim_{t \rightarrow \infty} e(t)$. Όμως η χρήση ενός θεωρήματος από τα μαθηματικά μας επιτρέπει να υπολογίσουμε αυτό το όριο χωρίς να υπολογίσουμε πρώτα την $e(t)$. Το θεώρημα αυτό λέγεται Θεώρημα Τελικής Τιμής και είναι η εξής σχέση:

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s * E(s)$$

Άρα αντικαθιστώντας έχουμε:

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s * E(s) = \lim_{s \rightarrow 0} s * \frac{1}{1 + GcGp} * \frac{1}{s} = \lim_{s \rightarrow 0} \frac{1}{1 + GcGp} = \lim_{s \rightarrow 0} \frac{1}{1 + Kp * \frac{a}{bs^2 + cs + d}}$$

Η είσοδος μας είναι η μοναδιαία συνάρτηση βαθμίδας γι' αυτό βάλαμε $R(s)=1/s$.

Το τελευταίο όριο φαίνεται πολύ εύκολα ότι μας δίνει την εξής σχέση για το σφάλμα μ.κ.:

$$e_{ss} = \frac{1}{1 + \frac{Kp * a}{d}} \quad (3)$$

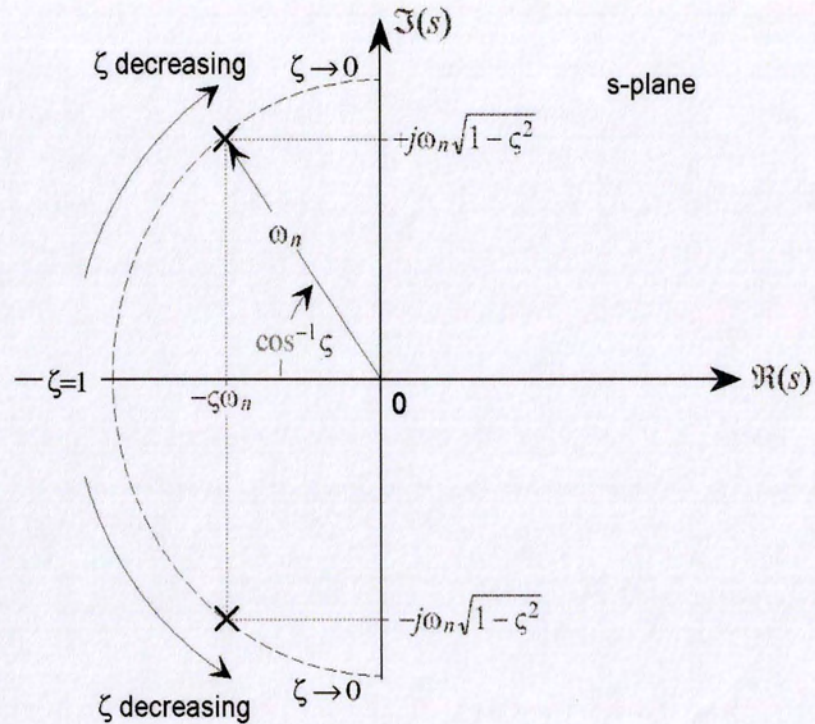
Από την παραπάνω σχέση μπορούμε να υπολογίζουμε το σφάλμα μ.κ. για κάθε τιμή του Kp ή και το αντίστροφο, δηλαδή ανάλογα με το σφάλμα μ.κ. που θέλουμε να έχουμε, να υπολογίζουμε την τιμή του Kp που το επιτυγχάνει.

Για τις 2 προηγούμενες περιπτώσεις ($Kp=1$ και $Kp=0.3293$) η σχέση (3) δίνει 0.0461 και 0.1279 αντίστοιχα. Αυτές οι τιμές επαληθεύονται εύκολα και οπτικά παρατηρώντας τις 2 γραφικές παραστάσεις των αποκρίσεων στα σχήματα 3 και 4.

Θεωρητικά, από τη σχέση (3) μπορούμε να συμπεράνουμε ότι γίνεται να μηδενίσουμε το σφάλμα μ.κ. με το Kp να τείνει στο άπειρο. Και όντως, για πολύ μεγάλες τιμές του Kp το σφάλμα μ.κ. τείνει στο 0. Όμως αυξάνοντας κατά πολύ το Kp μειώνεται κατά πολύ το ζ .

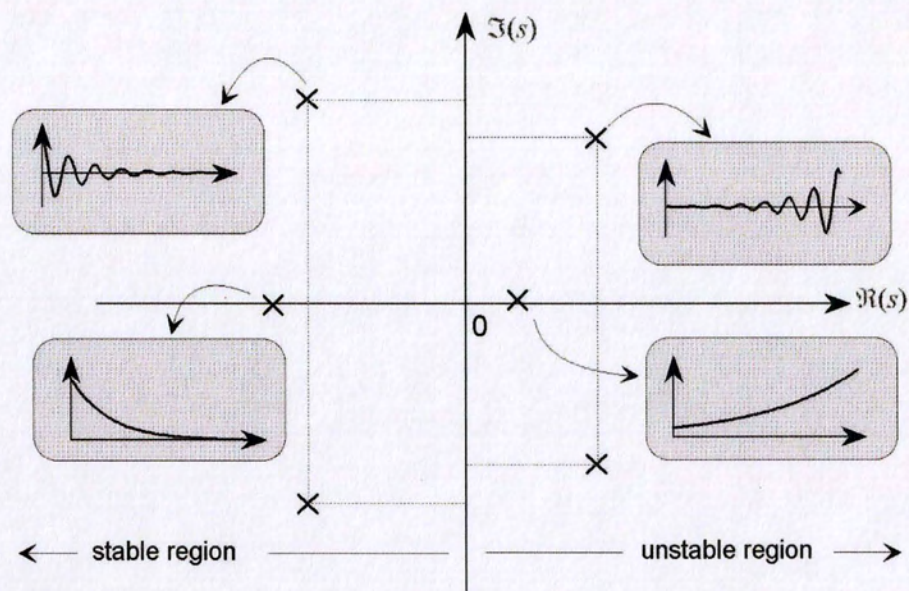
Όπως φαίνεται από τη σχέση (2), το ζ είναι αντιστρόφως ανάλογο του Kp . Όσο αυξάνουμε το Kp μειώνεται το ζ . Αυτό μπορεί να έχει πολύ ανεπιθύμητες επιπτώσεις στην απόκριση του

συστήματός μας. Όπως αναφέρθηκε νωρίτερα, το ζ έχει να κάνει με τη θέση των πόλων στο μιγαδικό επίπεδο. Αυτό φαίνεται στο επόμενο σχήμα.



Σχήμα 4-5: Μεταβολή των πόλων με το ζ .

Από το παραπάνω σχήμα συμπεραίνουμε ότι όσο το ζ τείνει στο 0, οι πόλοι μετακινούνται όλο και πιο κοντά στον φανταστικό άξονα (το πραγματικό τους μέρος τείνει στο 0). Οι επιπτώσεις που έχει αυτή η μετακίνηση, καθώς και κάθε θέση πόλου στο μιγαδικό επίπεδο, φαίνονται στο επόμενο σχήμα.



Σχήμα 4-6: Απόκριση συστήματος ανάλογα με τη θέση των πόλων στο μιγαδικό επίπεδο.

Είναι εμφανές από τα σχήματα 5 και 6 ότι όσο μικραίνει το ζ οι πόλοι μετακινούνται προς την ασταθή περιοχή. Δηλαδή οι ταλαντώσεις στη μεταβατική κατάσταση αυξάνονται όλο και περισσότερο μέχρι να γίνουν μόνιμες για $\zeta=0$ (το σύστημα δεν φτάνει ποτέ σε μια οριακή τιμή) και να αρχίσουν να αυξάνονται απεριόριστα για αρνητικές τιμές του ζ (πόλοι στο δεξί ημιεπίπεδο).

Γι' αυτό το λόγο συνήθως υπάρχουν κάποιοι περιορισμοί ως προς τις τιμές που μπορεί να πάρει το ζ , ώστε να αποφύγουμε μεγάλες ταλαντώσεις. Αυτό με τη σειρά του επιβάλλει κάποιον περιορισμό και στις τιμές που μπορεί να πάρει το K_p , σύμφωνα με το σχέση (2).

Οπότε το να προσπαθήσουμε να μειώσουμε το σφάλμα μ.κ. απλά αυξάνοντας το K_p δεν είναι επαρκής λύση, εκτός εάν οι απαιτήσεις μας για την απόκριση είναι πολύ ελαστικές και μπορούμε να δεχτούμε μεγάλες ταλαντώσεις. Σε διαφορετική περίπτωση πρέπει να προσθέσουμε και άλλο είδος ελέγχου εκτός από αναλογικό (ολοκληρωτικό, διαφορικό, ή και τα 2). Αυτή η περίπτωση θα εξεταστεί σε επόμενο κεφάλαιο.

4.3 Κριτήριο ευστάθειας Routh-Hurwitz

Μέχρι τώρα μιλήσαμε για την παράμετρο K_p και την επίδρασή της στη συμπεριφορά του συστήματος, δεν μιλήσαμε όμως για τις τιμές που μπορεί να πάρει. Αφού αυτή (η K_p) είναι η μόνη μεταβλητή παράμετρος του συστήματος, θα πρέπει να υπάρχει ένα εύρος τιμών για το οποίο το σύστημα θα είναι ευσταθές, και για τιμές εκτός αυτού του πεδίου θα είναι ασταθές. Αυτό το εύρος τιμών προσδιορίζεται με τη χρήση του κριτηρίου ευστάθειας Routh-Hurwitz.

Το κριτήριο αυτό είναι στην ουσία ένα μαθηματικό εργαλείο το οποίο μας πληροφορεί για την ύπαρξη και το πλήθος ριζών πολυωνυμικών εξισώσεων με θετικό πραγματικό μέρος. Ίσως να φαίνεται λίγη η πληροφορία που μας δίνει μιας και δεν μας λέει ποιές είναι οι ρίζες, αλλά όταν εφαρμόζεται σε ένα σύστημα αυτομάτου ελέγχου είναι ένα πολύ χρήσιμο εργαλείο γιατί έχει άμεση σχέση με την ευστάθεια του συστήματος.

Στα συστήματα αυτομάτου ελέγχου το κριτήριο R.H. εφαρμόζεται στην χαρακτηριστική εξίσωση του συστήματος και μας λέει αν και πόσες ρίζες της (πόλοι του συστήματος) έχουν θετικό πραγματικό μέρος. Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, πόλος στο δεξί ημιεπίπεδο (με θετικό πραγματικό μέρος) σημαίνει ασταθές σύστημα.

Για να εφαρμόσουμε το κριτήριο πρώτα γράφουμε την χαρακτηριστική εξίσωση στη μορφή:

$$a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0 = 0$$

Για να αποφασίσουμε αν το σύστημα είναι ευσταθές ή όχι ελέγχουμε τις επόμενες συνθήκες.

1. Δυο απαραίτητες αλλά όχι επαρκής συνθήκες για να έχουν όλες οι ρίζες αρνητικά πραγματικά μέρη είναι:

A) Όλοι οι συντελεστές του πολυωνύμου να έχουν το ίδιο πρόσημο.

B) Όλοι οι συντελεστές του πολυωνύμου να είναι μη-μηδενικοί.

2. Αν η συνθήκη 1 ικανοποιείται, τότε υπολογίζουμε τον πίνακα Routh-Hurwitz με τον εξής τρόπο:

$$\begin{array}{c|cccc}
 s^n & a_n & a_{n-2} & a_{n-4} & a_{n-6} & \dots \\
 s^{n-1} & a_{n-1} & a_{n-3} & a_{n-5} & a_{n-7} & \dots \\
 s^{n-2} & b_1 & b_2 & b_3 & & \dots \\
 s^{n-3} & c_1 & c_2 & c_3 & & \dots \\
 s^{n-4} & & & \vdots & & \\
 \vdots & & & \vdots & & \\
 s^1 & & & \vdots & & \\
 s^0 & & & \vdots & &
 \end{array}$$

Όπου τα a_i είναι οι συντελεστές του πολυωνύμου, και οι υπόλοιποι συντελεστές υπολογίζονται με τον εξής τρόπο:

$$\begin{aligned}
 b_1 &= \frac{-1}{a_{n-1}} \begin{vmatrix} a_n & a_{n-2} \\ a_{n-1} & a_{n-3} \end{vmatrix} = \frac{-1}{a_{n-1}} (a_n a_{n-3} - a_{n-2} a_{n-1}) & b_2 &= \frac{-1}{a_{n-1}} \begin{vmatrix} a_n & a_{n-4} \\ a_{n-1} & a_{n-5} \end{vmatrix} \\
 b_3 &= \frac{-1}{a_{n-1}} \begin{vmatrix} a_n & a_{n-6} \\ a_{n-1} & a_{n-7} \end{vmatrix} \dots & c_1 &= \frac{-1}{b_1} \begin{vmatrix} a_{n-1} & a_{n-3} \\ b_1 & b_2 \end{vmatrix} & c_2 &= \frac{-1}{b_1} \begin{vmatrix} a_{n-1} & a_{n-5} \\ b_1 & b_3 \end{vmatrix} \dots
 \end{aligned}$$

3. Η απαραίτητη συνθήκη ώστε να έχουν όλες οι ρίζες αρνητικά πραγματικά μέρη είναι η εξής:

Όλα τα στοιχεία της πρώτης στήλης του πίνακα πρέπει να έχουν το ίδιο πρόσημο. Ο αριθμός των αλλαγών του πρόσημου είναι ίσος με τον αριθμό των ριζών με θετικό πραγματικό μέρος.

Εάν υπάρχει κάποια μεταβλητή παράμετρος στην χαρακτηριστική εξίσωση, όπως ένα κέρδος, τότε θα εμφανιστεί και στον πίνακα Routh-Hurwitz. Τότε από την απαίτηση να έχουν όλα τα στοιχεία της πρώτης στήλης το ίδιο πρόσημο, θα προκύψουν κάποιες σχέσεις για την μεταβλητή παράμετρο.

Στην περίπτωσή μας η χαρακτηριστική εξίσωση του συστήματος είναι:

$$b * s^2 + c * s + d + Kp * a = 0$$

Ο πίνακας Routh-Hurwitz θα υπολογιστεί με τη βοήθεια του MATLAB. Η διαδικασία υπολογισμού φαίνεται παρακάτω.

```
Command Window

>> R=myRouth([b c d+Kp*a])

R =

[ 1156979788303063/73786976294838206464, (3037*Kp)/5000 + 1467/50000]
[ 8746062533947541/2305843009213693952, 0]
[ (3037*Kp)/5000 + 1467/50000, 0]

fx >>
```

Να υπενθυμίσουμε σε αυτό το σημείο ότι τα πολυώνυμα στο MATLAB ορίζονται γράφοντας τους συντελεστές μέσα σε αγκύλες ξεκινώντας από τη μεγαλύτερη δύναμη του s .

Έτσι, απ' ότι βλέπουμε στην παραπάνω εικόνα, ο πίνακας R.H. αποθηκεύεται στην μεταβλητή R. Μας ενδιαφέρει μόνο η πρώτη στήλη. Παρατηρούμε ότι τα 2 πρώτα στοιχεία της είναι θετικοί αριθμοί. Για να μην υπάρχουν πόλοι στο δεξιό ημιεπίπεδο, πρέπει να μην υπάρχει καμία αλλαγή πρόσημου στα στοιχεία της πρώτης στήλης. Άρα πρέπει και το 3^ο στοιχείο να είναι θετικό. Βλέπουμε όμως ότι είναι συνάρτηση του K_p , άρα θα προκύψει ένα εύρος τιμών για το K_p .

Την ανίσωση: $(3037*K_p)/5000 + 1467/50000 > 0$ τη λύνουμε στο MATLAB με την εντολή solve ως προς το K_p .

```
Command Window

>> solve('(3037*Kp)/5000 + 1467/50000 > 0', 'Kp')

ans =

Dom::Interval(-1467/30370, Inf)

>> -1467/30370

ans =

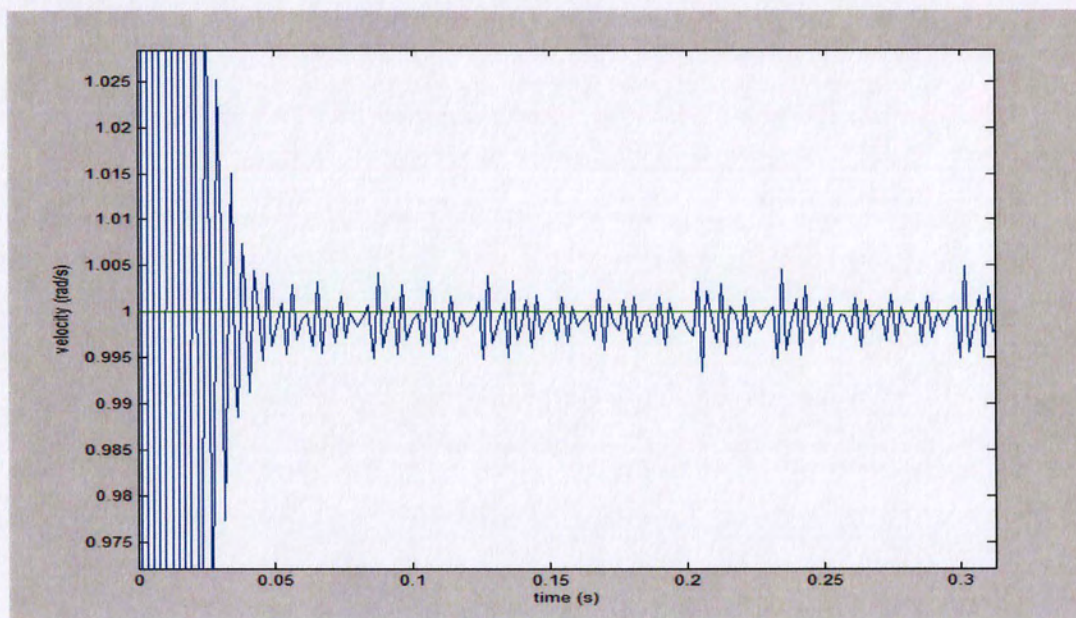
-0.0483

fx >>
```


Η λύση που παίρνουμε είναι το διάστημα $(-0.0483, +\infty)$. Αυτό σημαίνει ότι για τιμές του K_p που ανήκουν σε αυτό το πεδίο το σύστημα μας είναι ευσταθές, γιατί δεν υπάρχει πόλος με θετικό πραγματικό μέρος.

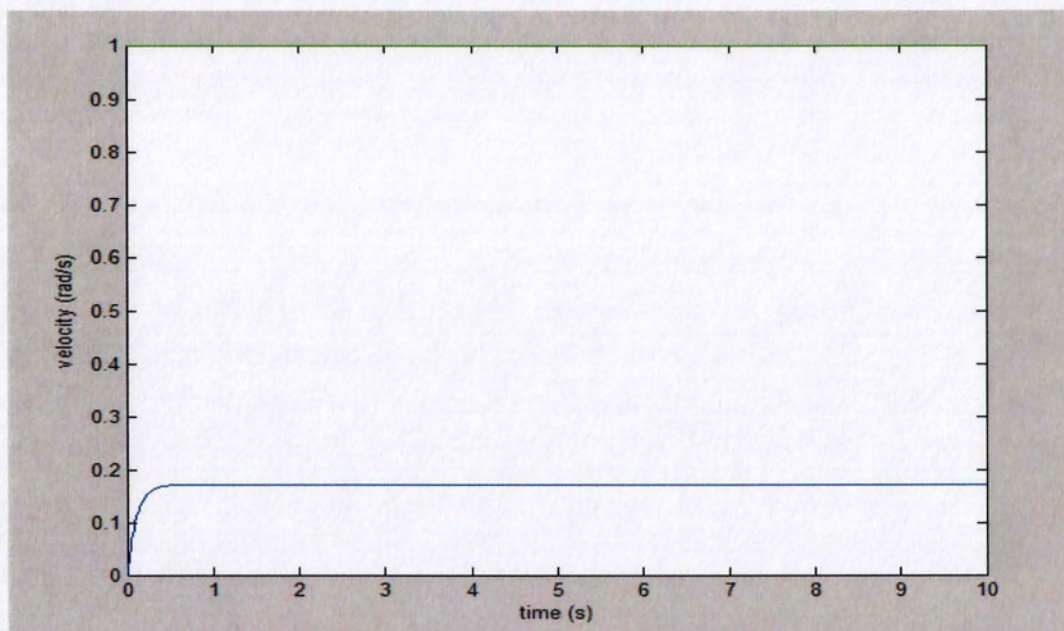
ΠΡΟΣΟΧΗ: Το κριτήριο μιλάει μόνο για την ευστάθεια του συστήματος και δεν μας λέει τίποτα για την μεταβατική του κατάσταση και το σφάλμα μόνιμης κατάστασης. Δηλαδή δεν αποκλείεται για κάποιες τιμές του K_p που ανήκουν στο παραπάνω πεδίο να έχουμε απαράδεκτη απόκριση χωρίς κανένα πρακτικό νόημα, αλλά ευσταθή.

Όπως για παράδειγμα για πολύ μεγάλες τιμές του K_p , η απόκριση δεν θα αυξάνεται απεριόριστα (θα είναι δηλαδή ευσταθές το σύστημα) αλλά θα παρουσιάζει τόσο έντονες ταλαντώσεις στη μεταβατική κατάσταση που θα είναι ανεπιθύμητο. Μία τέτοια απόκριση φαίνεται στο παρακάτω σχήμα.



Σχήμα 4-7: Απόκριση συστήματος για πολύ μεγάλο K_p ($=50$)

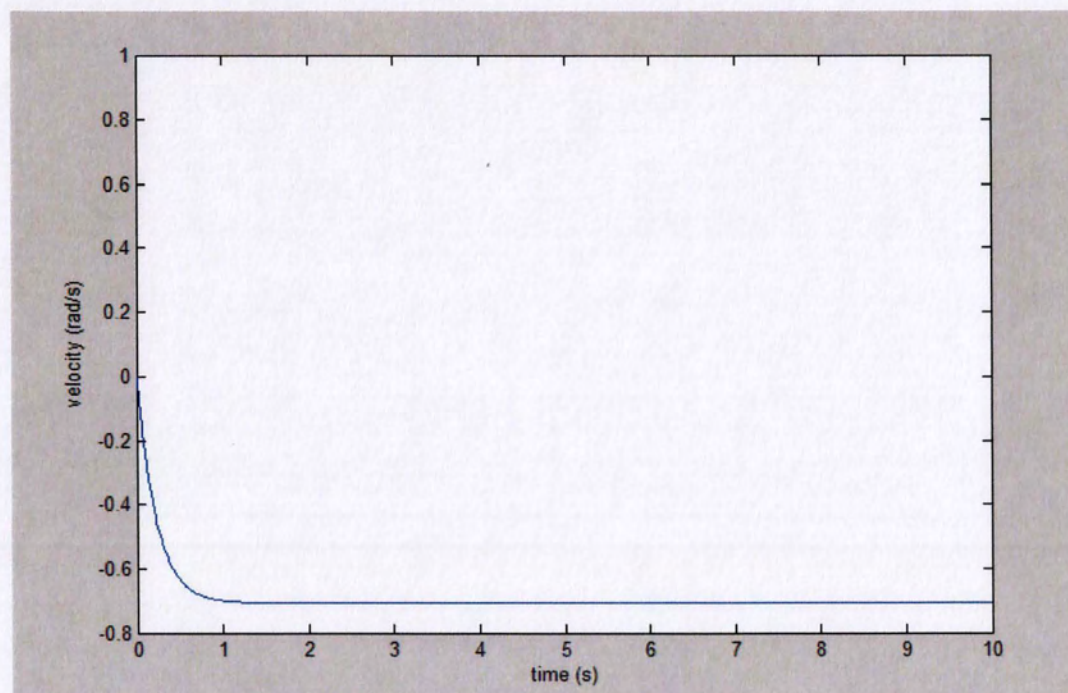
Στην περίπτωση του πολύ μικρού θετικού K_p η απόκριση είναι πάλι ευσταθής αλλά η μόνιμη τιμή που καταλήγει είναι τόσο μακριά από την επιθυμητή που την καθιστά ανούσια. Όπως η απόκριση του επόμενου σχήματος.



Σχήμα 4-8: Απόκριση συστήματος για πολύ μικρό K_p ($=0.01$)

Θα εξετάσουμε δυο ακόμα οριακές περιπτώσεις ευστάθειας που έχουν θεωρητικό ενδιαφέρον.

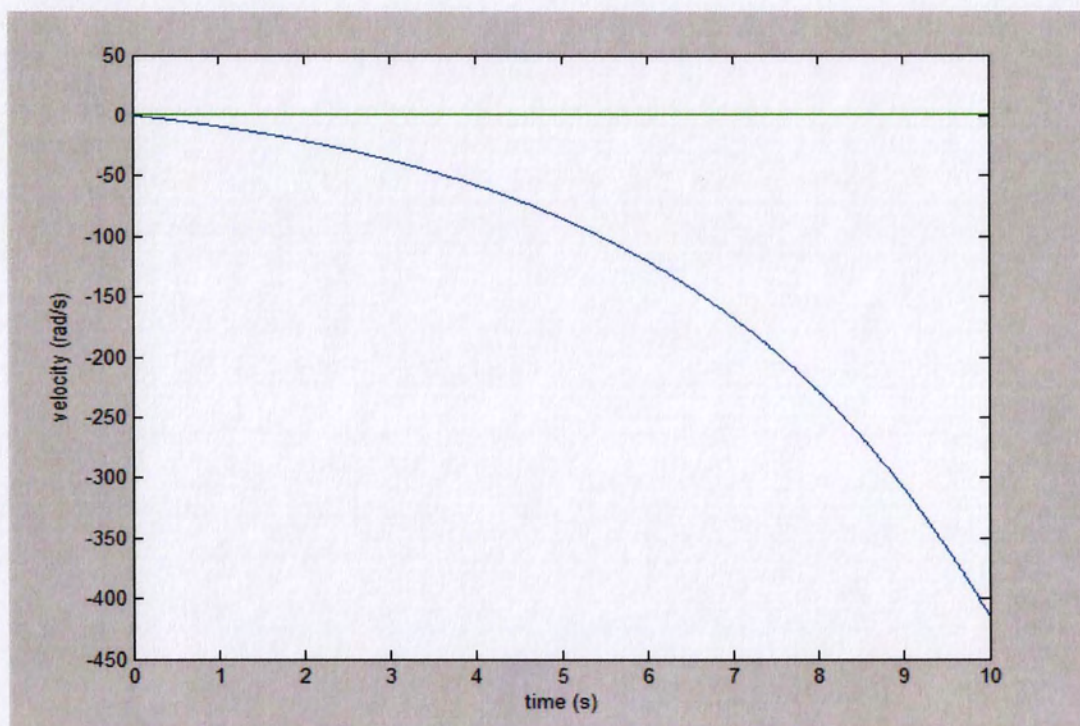
Σύμφωνα με το πεδίο τιμών που βρήκαμε πιο πάνω, το K_p μπορεί να πάρει και ορισμένες αρνητικές τιμές χωρίς να πέσει το σύστημα σε αστάθεια. Όπως για παράδειγμα την τιμή -0.02 .



Σχήμα 4-9: Απόκριση συστήματος για αρνητικό K_p ($=-0.02$)

Παρατηρούμε ότι το σύστημα είναι όντως ευσταθές. Βέβαια ένας τέτοιος ελεγκτής δεν έχει πρακτικό νόημα γιατί πετυχαίνει απόκριση αντίθετη από την επιθυμητή (ο κινητήρας στρέφεται προς την αντίθετη κατεύθυνση από αυτή που θέλουμε) αλλά δεν προκαλεί αστάθεια εφόσον καταλήγει σε μια ορισμένη τιμή. Άρα αρνητικό K_p δεν σημαίνει υποχρεωτικά και ασταθές σύστημα.

Για να κλείσουμε αυτή την ενότητα ας δούμε και μια περίπτωση αστάθειας. Για να το πετύχουμε αυτό αρκεί να διαλέξουμε μια οποιαδήποτε τιμή για το K_p εκτός του πεδίου που βρήκαμε με το κριτήριο Routh-Hurwitz. Ας πάρουμε την τιμή -0.05 για παράδειγμα. Η απόκριση του συστήματος για αυτή την τιμή φαίνεται στο επόμενο σχήμα.



Σχήμα 4-10: Απόκριση του συστήματος για K_p εκτός του πεδίου τιμών ($=-0.05$)

Είναι εμφανές ότι το σύστημά μας έχει πέσει σε αστάθεια γιατί η ταχύτητα αυξάνεται απεριόριστα χωρίς να προσεγγίζει κάποια συγκεκριμένη τιμή.

Αφού εξετάσαμε την ευστάθεια του συστήματός μας ας δούμε τώρα τι τρόπους έχουμε γενικά για να αντιμετωπίσουμε το σφάλμα μόνιμης κατάστασης.

4.4 Τρόποι αντιμετώπισης του σφάλματος μόνιμης κατάστασης

Όπως είπαμε και σε προηγούμενη ενότητα, το σύστημά μας είναι τύπου 0 γιατί η G_p δεν έχει ελεύθερο ολοκληρωτή.

$$G_p = \frac{a}{bs^2 + cs + d} \quad (\text{δεν βγαίνει κοινός παράγοντας το } 1/s)$$

Αυτό σημαίνει ότι σε είσοδο μοναδιαίας συνάρτησης βαθμίδας θα έχουμε πεπερασμένο σφάλμα μόνιμης κατάστασης το οποίο και υπολογίσαμε.

Γενικά, υπάρχουν 2 τρόποι να αντιμετωπίσουμε το σφάλμα μόνιμης κατάστασης. Ο ένας είναι να προσπαθήσουμε να το περιορίσουμε μέσα σε κάποια επιτρεπτά όρια, και ο άλλος να το μηδενίσουμε εντελώς.

Όπως είδαμε, μπορούμε να το μειώσουμε αυξάνοντας το K_p , αυτό όμως πολλές φορές δεν αρκεί λόγω μεγάλης μείωσης του ζ που οδηγεί σε μεγάλες ταλαντώσεις στη μεταβατική απόκριση. Αυτό συμβαίνει επειδή όταν κάνουμε αναλογικό έλεγχο μόνο, το σύστημά μας έχει μόνο μια παράμετρο, το K_p . Οπότε όλα τα χαρακτηριστικά του συστήματός μας εξαρτώνται από αυτή την παράμετρο. Αυτό έχει σαν αποτέλεσμα όταν καθορίζουμε ένα συγκεκριμένο μέγεθος (π.χ. σφάλμα μ.κ.), όλα τα άλλα μεγέθη να παίρνουν αυτόματα καθορισμένες τιμές.

Για να μπορέσουμε να ελέγξουμε περισσότερα από 1 μεγέθη ανεξάρτητα το ένα από το άλλο χρειαζόμαστε προφανώς περισσότερες από 1 μεταβλητές στο σύστημά μας. Αυτό επιτυγχάνεται με την είσοδο περισσότερων όρων στον ελεγκτή, όπως ολοκληρωτικό ή/και διαφορικό όρο. Έτσι, εκτός από το K_p , έχουμε και το K_i και το K_d στις μεταβλητές του συστήματος, πράγμα που μας επιτρέπει μεγαλύτερη ελευθερία στη ρύθμιση των παραμέτρων γιατί το κάθε μέγεθος εξαρτάται από 2 ή 3 μεταβλητές.

Ο ολοκληρωτικός και ο διαφορικός ελεγκτής έχουν διαφορετική επίδραση στη συμπεριφορά του συστήματος και θα εξεταστούν αναλυτικά σε επόμενη ενότητα. Ακολουθεί μια σύντομη περιγραφή.

ΠΕΡΙΓΡΑΦΗ ΕΠΙΔΡΑΣΗΣ ΒΑΣΙΚΩΝ ΕΛΕΓΚΤΩΝ

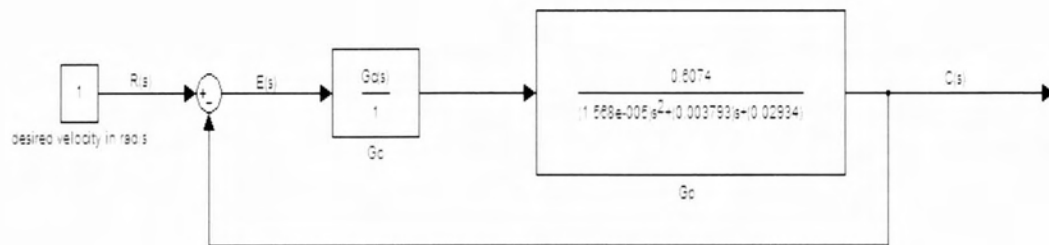
Σαν γενικός κανόνας, ο διαφορικός ελεγκτής έχει την τάση να βελτιώνει την μεταβατική απόκριση του συστήματος, με την έννοια ότι μειώνει τις ταλαντώσεις και την υπερακόντιση πριν πάρουν μεγάλες τιμές. Έχει επίδραση μόνο στη μεταβατική κατάσταση του συστήματος γι' αυτό και δεν χρησιμοποιείται ποτέ μόνος αλλά πάντα σε συνδυασμό με αναλογικό, ολοκληρωτικό, ή και τα 2. Λόγω της φύσης της διαφόρισης, ο διαφορικός ελεγκτής 'ανιχνεύει' τις απότομες μεταβολές στην απόκριση και δρα ανασταλτικά πριν γίνουν πολύ μεγάλες. Εισάγει δηλαδή μια επιπλέον απόσβεση στο σύστημα όπως θα δούμε αναλυτικά πιο μετά.

Από την παραπάνω παράγραφο φαίνεται ότι στην περίπτωση που θέλουμε μεγάλο K_p αλλά όχι και τις πολύ μεγάλες ταλαντώσεις που προκαλεί, μάλλον πρέπει να προσθέσουμε έναν διαφορικό όρο στον ελεγκτή μας.

Ο ολοκληρωτικός ελεγκτής έχει την τάση να μηδενίζει το σφάλμα μόνιμης κατάστασης γιατί αυξάνει τον τύπο του συστήματος κατά 1. Οπότε αν δεν θέλουμε απλά να περιορίσουμε το σφάλμα μ.κ. αλλά να το μηδενίσουμε εντελώς τότε πρέπει να εισάγουμε έναν ολοκληρωτικό όρο στον ελεγκτή μας. Σε αυτό το συμπέρασμα μπορούμε να καταλήξουμε πολύ απλά εξετάζοντας μόνο τον τύπο του σφάλματος μόνιμης κατάστασης.

$$e_{ss} = \frac{1}{1 + \frac{K_p * a}{d}} \quad (\text{σφάλμα μόνιμης κατάστασης})$$

Έστω G_c η άγνωστη συνάρτηση μεταφοράς του ελεγκτή που χρειαζόμαστε ώστε να μηδενίσουμε το σφάλμα μ.κ. Άρα το δομικό διάγραμμα του συστήματός μας είναι:



Κάνουμε την ίδια ανάλυση για το σφάλμα μ.κ. μόνο που αντί να θέσουμε όπου $G_c = K_p$, κρατάμε το G_c άγνωστο.

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s * E(s) = \lim_{s \rightarrow 0} s * \frac{1}{1 + G_c G_p} * \frac{1}{s} = \lim_{s \rightarrow 0} \frac{1}{1 + G_c G_p} = \lim_{s \rightarrow 0} \frac{1}{1 + G_c * \frac{a}{bs^2 + cs + d}} = \lim_{s \rightarrow 0} \frac{1}{1 + G_c * \frac{a}{d}}$$

Για να μηδενίζεται το παραπάνω όριο πρέπει ο παρανομαστής να απειρίζεται όταν το s τείνει στο μηδέν. Αφού το a/d είναι πεπερασμένη ποσότητα, είναι προφανές ότι πρέπει:

$$\lim_{s \rightarrow 0} G_c = \infty$$

Άρα η συνάρτηση μεταφοράς του ελεγκτή μας θα πρέπει να απειρίζεται όταν το s τείνει στο μηδέν. Αυτή η απαίτηση μας οδηγεί σε μια συνάρτηση μεταφοράς της μορφής:

$$G_c = \frac{K_i}{s}, G_c = K_p + \frac{K_i}{s}$$

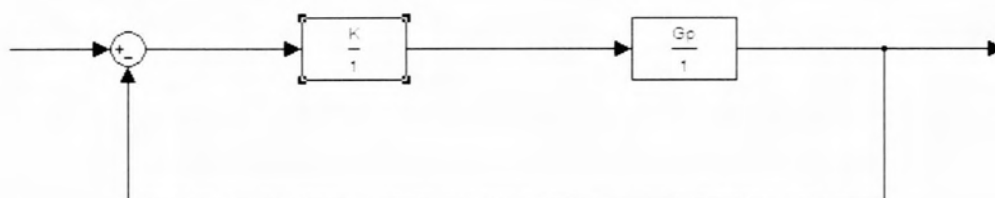
Δηλαδή ολοκληρωτικός ή αναλογικός/ολοκληρωτικός ελεγκτής. Το ελάττωμα του ολοκληρωτικού ελεγκτή είναι ότι συνήθως προκαλεί προβλήματα ευστάθειας καθώς προσθέτει έναν ακόμη πόλο κοντά στον φανταστικό άξονα. Γι' αυτό τις περισσότερες φορές χρησιμοποιείται μαζί με αναλογικό ή/και διαφορικό ελεγκτή ώστε να σταθεροποιείται το σύστημα.

4.5 Ανάλυση του συστήματος με τη βοήθεια του τόπου των ριζών

Ο τόπος των ριζών (T.P.) είναι η γραφική απεικόνιση των ριζών της χαρακτηριστικής εξίσωσης (πόλοι κλειστού βρόχου) καθώς μεταβάλλεται μια παράμετρος του συστήματος. Η παράμετρος αυτή μπορεί να είναι μια οποιαδήποτε παράμετρος του συστήματος αλλά συνήθως είναι το κέρδος κάποιου ελεγκτή. Το πεδίο τιμών της παραμέτρου για την οποία γίνεται ο τόπος των ριζών συνήθως είναι από μηδέν μέχρι άπειρο αλλά μπορούμε να τον σχεδιάσουμε για οποιοδήποτε πεδίο τιμών επιθυμούμε.

Ο τόπος των ριζών είναι ένα πολύ χρήσιμο εργαλείο στη σχεδίαση συστημάτων αυτομάτου ελέγχου γιατί μας βοηθάει να κατανοήσουμε τη συμπεριφορά του συστήματός μας. Θέματα όπως ευστάθεια και μεταβατική απόκριση γίνονται εύκολα αντιληπτά γιατί έχουν να κάνουν με τη θέση των πόλων στο μιγαδικό επίπεδο. Κάθε σημείο του T.P. είναι ένας πόλος του συστήματος στον οποίο αντιστοιχεί μια συγκεκριμένη τιμή της μεταβαλλόμενης παραμέτρου. Έτσι μπορούμε να διαλέξουμε έναν πόλο σε ένα σημείο που είναι βολικό, και μετά να μάθουμε τι τιμή της παραμέτρου του αντιστοιχεί. Ή και το αντίστροφο, έχοντας κάποια δεδομένη τιμή για την παράμετρο, να μάθουμε ποιός πόλος (ή καλύτερα ζευγάρι πόλων) αντιστοιχεί σε αυτή την τιμή.

Η σχεδίαση του μπορεί να γίνει είτε με το χέρι είτε με τη βοήθεια ηλεκτρονικού υπολογιστή. Στην παρούσα εργασία θα σχεδιαστεί με το MATLAB. Η εντολή που χρησιμοποιείται για τη σχεδίαση του T.P. είναι η `rlocus`. Για να τη χρησιμοποιήσουμε σωστά πρέπει να φέρουμε το σύστημα μας στη μορφή:



Όπου K είναι η παράμετρος ως προς την οποία θα γίνει ο τόπος των ριζών. Μετά γράφουμε την χαρακτηριστική εξίσωση στη μορφή: $1 + K*Gp(s)=0$

Το επόμενο βήμα είναι να ορίσουμε στο workspace του MATLAB τη συνάρτηση μεταφοράς ανοιχτού βρόχου. Αυτό γίνεται με την εντολή `tf(num,den)` η οποία ορίζει συναρτήσεις μεταφοράς με είσοδο τον αριθμητή και τον παρανομαστή. Άρα πρώτα ορίζουμε τα πολυώνυμα του αριθμητή και του παρανομαστή και μετά τη συνάρτηση μεταφοράς, όπως φαίνεται παρακάτω:

```
Command Window
>> num=[a]

num =

    0.607400000000000

>> den=[b c d]

den =

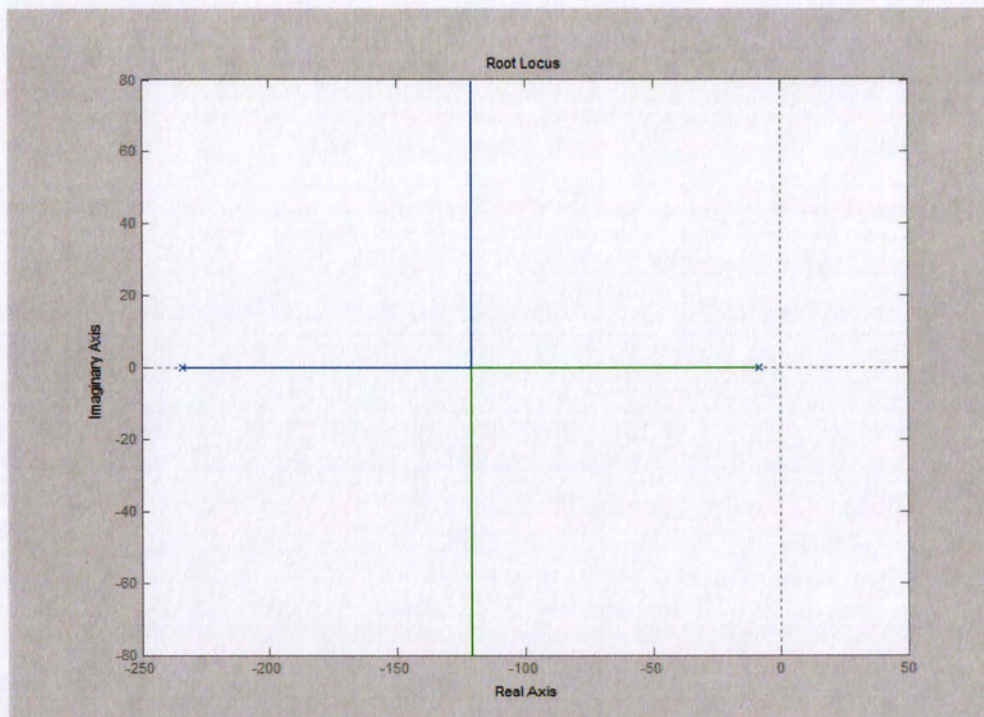
    0.000015680000000    0.003793000000000    0.029340000000000

>> Gp=tf(num,den)

Transfer function:
           0.6074
-----
1.568e-005 s^2 + 0.003793 s + 0.02934

fx >>
```

Τώρα που ορίσαμε τη συνάρτηση μεταφοράς ανοιχτού βρόχου εκτελούμε την εντολή `rlocus(Gp)`. Αν την εκτελέσουμε με μόνη είσοδο τη Σ.Μ. τότε το MATLAB θεωρεί θετικές τιμές του K . Αν θέλουμε να ορίσουμε εμείς τις τιμές του K τότε ορίζουμε ένα διάνυσμα K με τις επιθυμητές τιμές και εκτελούμε την εντολή έτσι: `rlocus(Gp,K)`. Ο τόπος των ριζών που παίρνουμε με την εντολή `rlocus(Gp)` φαίνεται στο επόμενο σχήμα.



Σχήμα 4-11: Τόπος των ριζών για θετικές τιμές του K .

Παρατηρούμε ότι ολόκληρος ο τόπος των ριζών βρίσκεται στο αριστερό ημιεπίπεδο, δηλαδή το σύστημά μας είναι πάντα ευσταθές. Αυτό είναι αναμενόμενο γιατί σε προηγούμενη ενότητα είδαμε ότι το σύστημά είναι ευσταθές στο διάστημα $(-0.0483, +\infty)$, και ο παραπάνω Τ.Ρ. σχεδιάστηκε για $K > 0$.

Εάν τον σχεδιάσουμε για ένα διάστημα $(-0.08, 0.08)$ θα πρέπει να δούμε και πόλους στο δεξί ημιεπίπεδο που θα αντιστοιχούν στις τιμές του K από -0.08 έως -0.0483 .

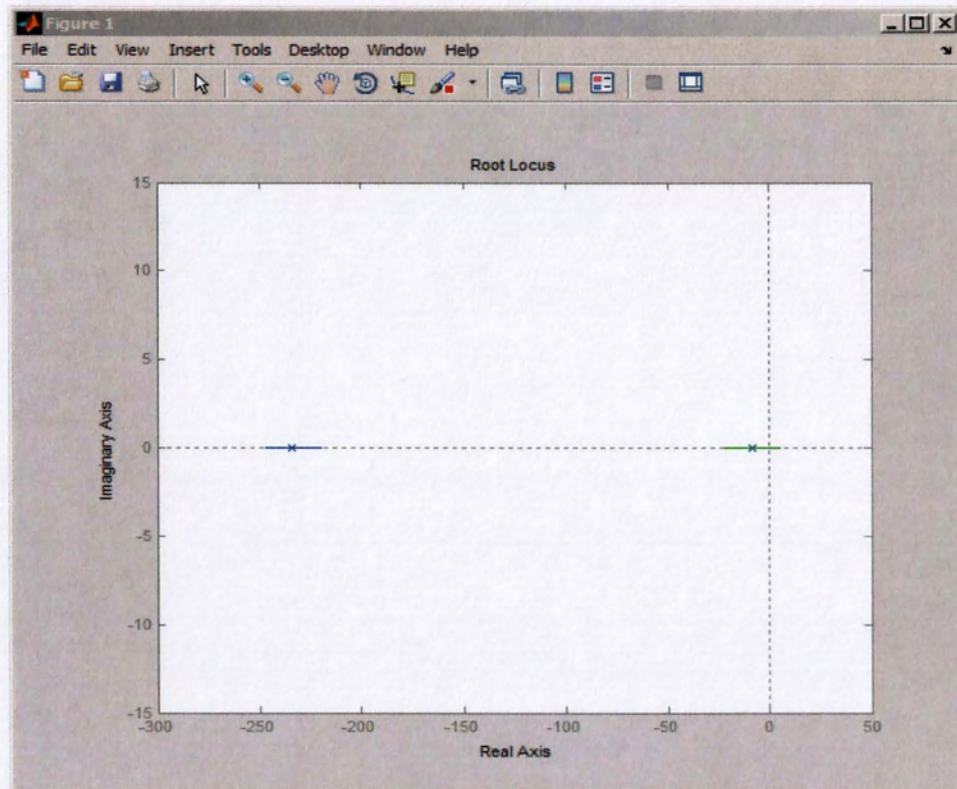
Για να το κάνουμε αυτό πρώτα ορίζουμε ένα διάνυσμα K με τις επιθυμητές τιμές και έπειτα εκτελούμε την εντολή `rlocus(Gp,K)` όπως φαίνεται παρακάτω.

```

Command Window
>> K=[-0.08:0.001:0.08];
>> rlocus(Gp,K)
fx >>

```


Ο τύπος των ριζών που παίρνουμε με την παραπάνω εντολή είναι αυτός που φαίνεται στο επόμενο σχήμα:



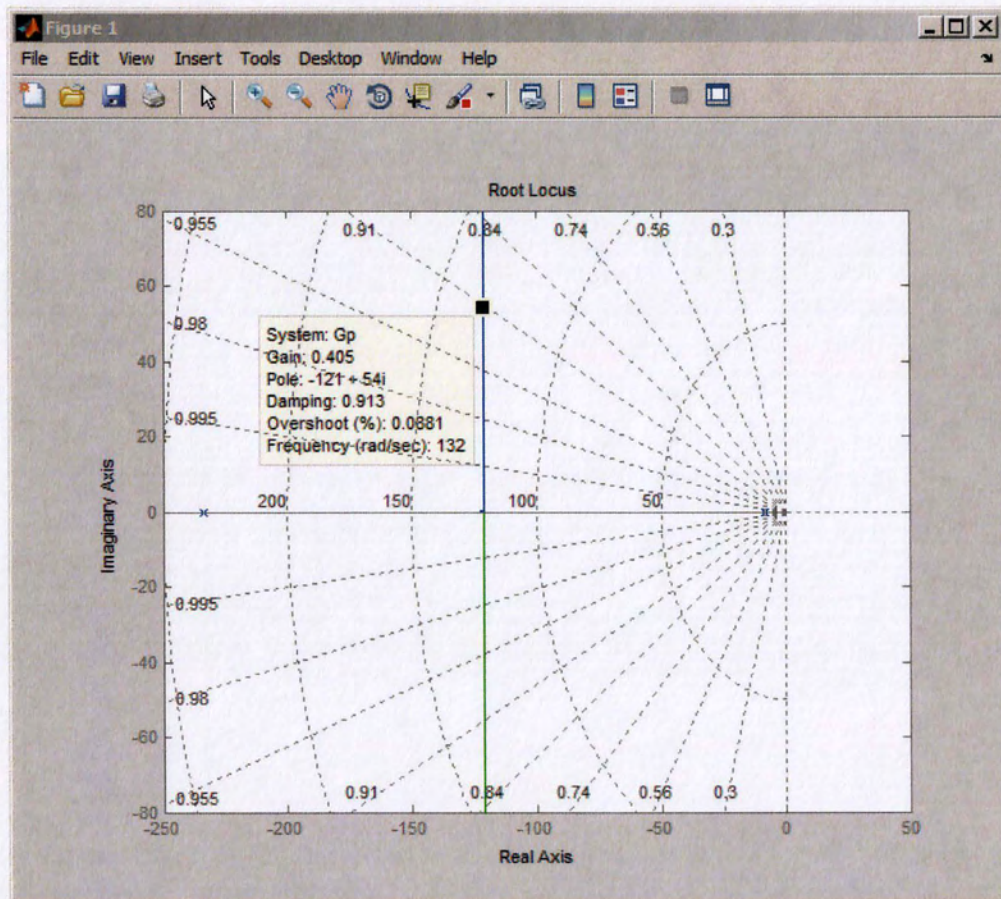
Σχήμα 4-12: Τύπος των ριζών για $-0.08 < K < 0.08$

Παρατηρούμε την ύπαρξη πόλων στο δεξί ημιεπίπεδο που υποδηλώνει αστάθεια. Αυτό συμβαίνει γιατί όπως είπαμε ο παραπάνω T.P. σχεδιάστηκε για ένα εύρος τιμών του K που περιλαμβάνει τιμές εκτός του πεδίου ευστάθειας.

Έτσι με τον παραπάνω τρόπο μπορούμε να σχεδιάζουμε τον T.P. για οποιεσδήποτε τιμές του K θέλουμε.

Κάνοντας αριστερό κλικ πάνω σε ένα οποιοδήποτε σημείο του T.P. παίρνουμε πληροφορίες για τον συγκεκριμένο πόλο όπως: ακριβής θέση στο μιγαδικό επίπεδο, το κέρδος που του αντιστοιχεί (τιμή της παραμέτρου K), συντελεστή απόσβεσης ζ κτλ. Κρατώντας πατημένο το πλήκτρο μπορούμε να μετακινούμε τον πόλο και να παρατηρούμε πως αλλάζουν οι τιμές μέχρι να βρούμε κάποιες ικανοποιητικές.

Κάνοντας δεξί κλικ και επιλέγοντας το Grid εμφανίζονται βοηθητικές γραμμές με τον συντελεστή ζ που μας βοηθάνε να επιλέξουμε κάποιον πόλο με βάση το ζ.



Σχήμα 4-13: Πληροφορίες που μας δίνει ο T.P.

Από το παραπάνω σχήμα φαίνεται η χρησιμότητα του T.P. ως σχεδιαστικό εργαλείο. Μας δίνει τη δυνατότητα να καταλάβουμε το σύστημά μας γραφικά και να αποφασίσουμε εύκολα για τις τιμές διαφόρων παραμέτρων.

Όμως το σύστημά μας είναι 2^{ης} τάξης, άρα έχει 2 πόλους για κάθε τιμή του K. Για να βρούμε όλους τους πόλους που αντιστοιχούν σε μια τιμή του K χρησιμοποιούμε την εντολή `rlocfind`. Η εντολή αυτή δέχεται για είσοδο ένα σημείο του T.P. που διαλέγει ο χρήστης με το ποντίκι και επιστρέφει σε 2 μεταβλητές το κέρδος που αντιστοιχεί σε εκείνο το σημείο καθώς και όλους τους πόλους που αντιστοιχούν σε εκείνο το κέρδος. Επίσης οι πόλοι σχεδιάζονται με κόκκινους σταυρούς. Στις επόμενες εικόνες φαίνεται η εκτέλεση της εντολής `rlocfind` και τα αποτελέσματα.


```
Command Window
>> [gain,poles]=rlocfind(Gp)
Select a point in the graphics window

selected_point =

-1.2109e+002 +5.7937e+001i

gain =

0.4160

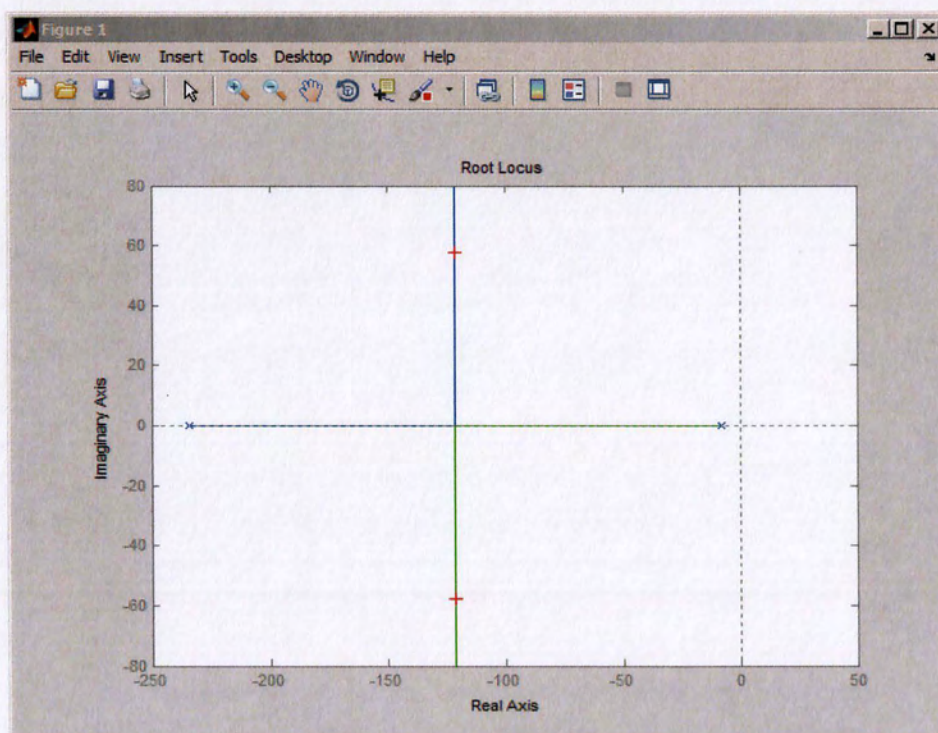
poles =

1.0e+002 *

-1.2095 + 0.5794i
-1.2095 - 0.5794i

fx >>
```

Όπως βλέπουμε στην αρχή το MATLAB μας ζητάει να διαλέξουμε ένα σημείο από τον T.P. Μόλις κάνουμε κλικ στον επιθυμητό πόλο, μας επιστρέφει στην μεταβλητή gain το κέρδος που του αντιστοιχεί, και στην μεταβλητή poles όλους τους υπόλοιπους πόλους που αντιστοιχούν στο συγκεκριμένο κέρδος. Στην περίπτωση μας επειδή έχουμε σύστημα 2^{ης} τάξης έχουμε δύο συζυγείς πόλους. Επίσης, οι πόλοι σχεδιάζονται με κόκκινους σταυρούς όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 4-14: Σχεδίαση πόλων με την εντολή rlocfind.

Εκτός από τη γραφική απεικόνιση που μόλις είδαμε, υπάρχει και άλλος τρόπος αναπαράστασης του τόπου των ριζών. Μπορούμε να τον αποθηκεύσουμε σε μια μεταβλητή ως πίνακα και να έχουμε μια 1 προς 1 αντιστοιχία μεταξύ τιμών του K και πόλων. Αυτός ο τρόπος δεν είναι τόσο εύκολα αντιληπτός γραφικά όπως ο προηγούμενος, αλλά βοηθάει εάν θέλουμε να ξέρουμε ακριβώς ποιός πόλος αντιστοιχεί σε κάθε τιμή του K για όλα τα K . Οι αντίστοιχες εντολές φαίνονται παρακάτω.

```

Command Window
>> R=rlocus(Gp,K)

R =

Columns 1 through 7
-246.8739 -245.3263 -243.7592 -242.1718 -240.5633 -238.9329 -237.2797
  4.9734   3.4258   1.8586   0.2712  -1.3372  -2.9676  -4.6208

Columns 8 through 14
-235.6026 -233.9006 -232.1726 -230.4174 -228.6335 -226.8195 -224.9739
 -6.2979  -7.9999  -9.7279 -11.4832 -13.2670 -15.0810 -16.9266

Columns 15 through 17
-223.0950 -221.1809 -219.2295
 -18.8055 -20.7196 -22.6710

>> K

K =

Columns 1 through 7
-0.0800 -0.0700 -0.0600 -0.0500 -0.0400 -0.0300 -0.0200

Columns 8 through 14
-0.0100      0  0.0100  0.0200  0.0300  0.0400  0.0500

Columns 15 through 17
 0.0600  0.0700  0.0800

fx >>

```

Η πρώτη εντολή αποθηκεύει στην μεταβλητή R τον τόπο των ριζών και η δεύτερη τυπώνει στην οθόνη το διάνυσμα με τις τιμές του K για τις οποίες σχεδιάστηκε. Η κάθε στήλη του R αντιστοιχεί σε μια στήλη του K . Για παράδειγμα, η 3^η στήλη του R έχει 2 πόλους από τους οποίους ο 2^{ος} είναι θετικός. Αυτή η στήλη αντιστοιχεί στην 3^η στήλη του διανύσματος K δηλαδή στο -0.06 οπότε

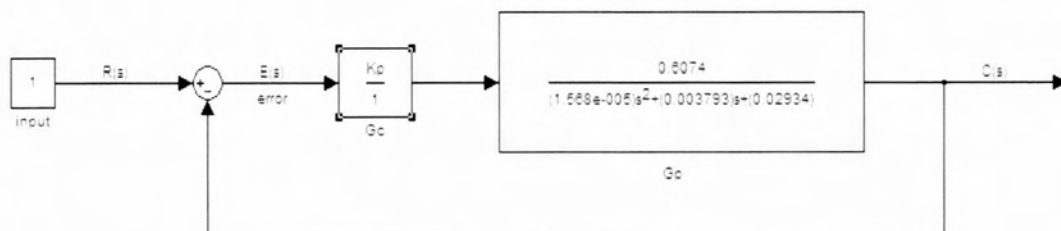
είναι αναμενόμενο να έχουμε έναν θετικό πόλο. Αν δούμε την 7^η στήλη, αυτή έχει αρνητικούς πόλους και αντιστοιχούν στην 7^η στήλη του K δηλαδή στο -0.02 που είναι μέσα στα όρια ευστάθειας.

Ο τύπος των ριζών μπορεί να γίνει για οποιαδήποτε παράμετρο του συστήματος. Συνήθως αυτές οι παράμετροι είναι τα κέρδη ενός PID ελεγκτή. Στο κεφάλαιο αυτό έγινε για την παράμετρο K_p γιατί εξετάζαμε την περίπτωση του αναλογικού ελεγκτή. Στο επόμενο κεφάλαιο θα εισάγουμε ολοκληρωτικό όρο στον ελεγκτή μας, και θα ξανασχεδιάσουμε τον καινούριο τύπο των ριζών για την παράμετρο K_i . Όπως θα δούμε, για να το πετύχουμε αυτό πρέπει να γράψουμε την χαρακτηριστική εξίσωση του συστήματος με μια συγκεκριμένη μορφή.

Κεφάλαιο 5 : Έλεγχος της ταχύτητας με αναλογικό-ολοκληρωτικό ελεγκτή

5.1 Ανάλυση του συστήματος κλειστού βρόχου

Στο προηγούμενο κεφάλαιο μελετήθηκε η επίδραση του αναλογικού ελεγκτή στο σύστημα κλειστού βρόχου. Δηλαδή η συνάρτηση μεταφοράς του ελεγκτή ήταν μόνο ένα απλό κέρδος K_p .



Σχήμα 5-1: Δομικό διάγραμμα συστήματος με αναλογικό ελεγκτή.

Όμως είδαμε ότι παρουσιάστηκαν κάποια προβλήματα στην απόκριση του συστήματος. Για την ακρίβεια, είχαμε καταλήξει στις εξής σχέσεις για το σφάλμα μόνιμης κατάστασης και τον συντελεστή ζ:

$$e_{ss} = \frac{1}{1 + \frac{Kp * a}{d}} \quad (\text{σφάλμα μόνιμης κατάστασης})$$

$$\zeta = \frac{c}{2\sqrt{b(d + Kp * a)}} \quad (\text{συντελεστής } \zeta)$$

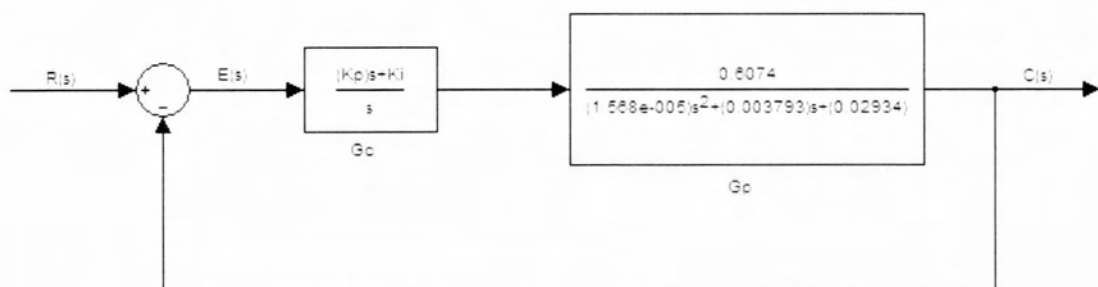
Το πρόβλημα είναι ότι εάν θέλουμε να έχουμε μεγάλο ζ (ώστε να έχουμε λίγες έως καθόλου ταλαντώσεις) πρέπει να διαλέξουμε μικρό Kp . Όμως με μικρό Kp έχουμε μεγάλο σφάλμα μόνιμης κατάστασης. Δεν γίνετε να τα ελέγξουμε και τα δύο.

Αυτό το πρόβλημα έρχεται να λύσει ο ολοκληρωτικός ελεγκτής, ο οποίος γενικά επιδρά στην μόνιμη κατάσταση της απόκρισης και τείνει να μηδενίσει το σφάλμα που σχετίζεται με αυτήν.

Στο τέλος δηλαδή αυτής της ενότητας θα έχουμε σχεδιάσει έναν αναλογικό-ολοκληρωτικό (PI) ελεγκτή με τον οποίο η απόκριση δεν θα εμφανίζει ταλαντώσεις αλλά ούτε σφάλμα μόνιμης κατάστασης. Εισάγοντας τον ολοκληρωτικό όρο η συνάρτηση μεταφοράς του ελεγκτή γίνεται:

$$Gc(s) = Kp + \frac{Ki}{s} \Rightarrow Gc(s) = \frac{sKp + Ki}{s}$$

Το δομικό διάγραμμα του συστήματος κλειστού βρόχου γίνεται:



Σχήμα 5-2: Δομικό διάγραμμα συστήματος με PI ελεγκτή.

Για λόγους ευκολίας από εδώ και πέρα θα αναφερόμαστε στην G_p ως $G_p = \frac{a}{bs^2 + cs + d}$

όπου:

$$a=0.6074, b=1,568 \cdot 10^{-5}, c=0.003793, d=0.02934$$

Η συνάρτηση μεταφοράς εισόδου-εξόδου κλειστού βρόχου είναι:

$$\begin{aligned} \frac{C(s)}{R(s)} &= \frac{G_c G_p}{1 + G_c G_p} = \frac{\frac{(K_p * s + K_i) * a}{s(bs^2 + cs + d)}}{1 + \frac{(K_p * s + K_i) * a}{s(bs^2 + cs + d)}} = \frac{\frac{(K_p * s + K_i) * a}{s(bs^2 + cs + d)}}{\frac{s(bs^2 + cs + d)}{s(bs^2 + cs + d)} + \frac{(K_p * s + K_i) * a}{s(bs^2 + cs + d)}} \\ &= \frac{\frac{(K_p * s + K_i) * a}{s(bs^2 + cs + d)}}{\frac{s(bs^2 + cs + d) + (K_p * s + K_i) * a}{s(bs^2 + cs + d)}} = \frac{(K_p * s + K_i) * a}{s(bs^2 + cs + d) + (K_p * s + K_i) * a} \\ &= \frac{K_p * a * s + K_i * a}{bs^3 + cs^2 + ds + K_p * a * s + K_i * a} = \frac{K_p * a * s + K_i * a}{bs^3 + cs^2 + (d + K_p * a)s + K_i * a} \end{aligned}$$

Συγκρίνοντας με την χαρακτηριστική μορφή συναρτήσεων μεταφοράς συστημάτων 2^{ης} τάξης:

$$\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Παρατηρούμε ότι με την είσοδο του ολοκληρωτικού ελεγκτή ο παρανομαστής έγινε πολυώνυμο τρίτου βαθμού αντί για δεύτερου και ο αριθμητής είναι πολυώνυμο πρώτου βαθμού αντί για μηδενικού.

Άρα δεν μπορούμε να το προσεγγίσουμε σαν σύστημα δεύτερης τάξης γιατί θα έχουμε σημαντικό σφάλμα, κυρίως λόγω του μηδενικού του αριθμητή. Συνεπώς, δεν μπορούμε να εξάγουμε κλειστές σχέσεις για το ζ και το ω του συστήματος με τον PI ελεγκτή, όπως κάναμε και όταν το σύστημα ήταν δεύτερης τάξης με τον P ελεγκτή. Επίσης οι παράμετροι αυτοί (ζ και ω) δεν επαρκούν

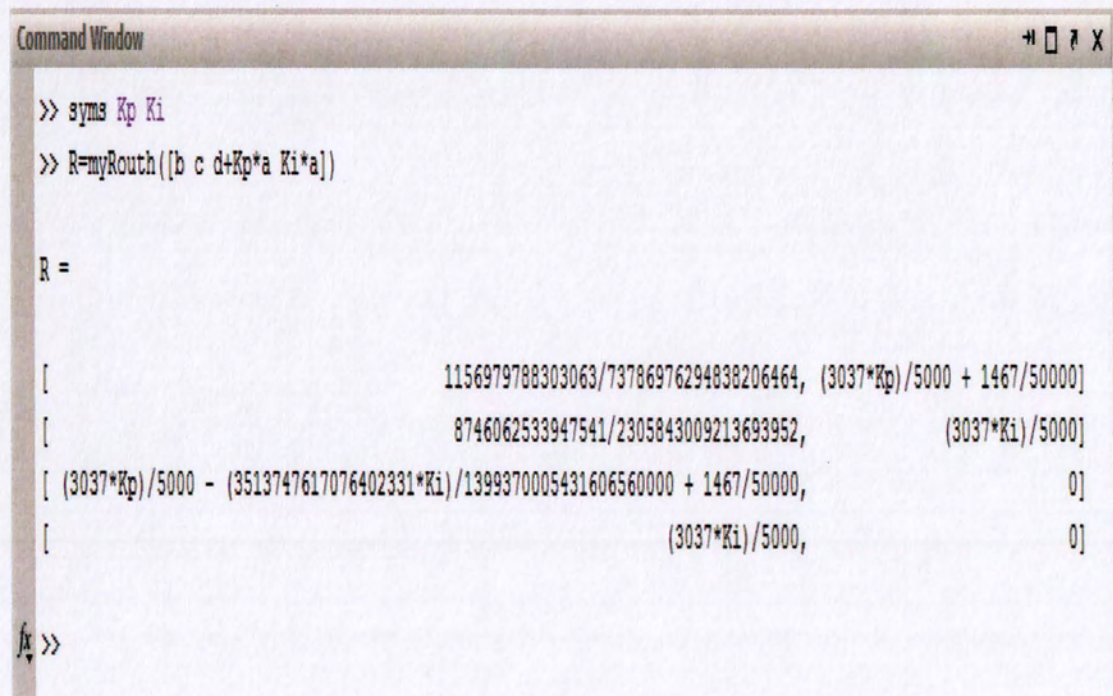
για να μας περιγράψουν πλήρως την απόκριση του συστήματος αλλά μας δίνουν μόνο μια καλή ένδειξη. Η επιπλέον παράμετρος που παίζει καθοριστικό ρόλο είναι η θέση του μηδενικού κλειστού βρόχου. Ο τρόπος για να μπορέσουμε να αναλύσουμε το σύστημα αυτό είναι να βασιστούμε σε μεθόδους που είναι ανεξάρτητες από την τάξη του συστήματος, όπως το κριτήριο Routh-Hurwitz και ο τόπος των ριζών.

5.2 Κριτήριο ευστάθειας Routh-Hurwitz

Η χαρακτηριστική εξίσωση του συστήματος μας είναι:

$$bs^3 + cs^2 + (d + Kp * a)s + Ki * a = 0$$

Εφαρμόζοντας το κριτήριο R-H στην χαρακτηριστική εξίσωση μπορούμε να βρούμε συνθήκες υπό τις οποίες το σύστημα θα είναι ευσταθές (δηλαδή δεν θα υπάρχουν πόλοι στο δεξί ημιεπίπεδο). Τώρα οι μεταβλητές μας είναι δύο, το Kp και το Ki , οπότε θα πρέπει να ορίζουμε μια συγκεκριμένη τιμή για το ένα και να λύνουμε ως προς το άλλο. Παρακάτω φαίνεται η εφαρμογή του κριτηρίου με χρήση MATLAB.



```

Command Window
>> syms Kp Ki
>> R=myRouth([b c d+Kp*a Ki*a])

R =

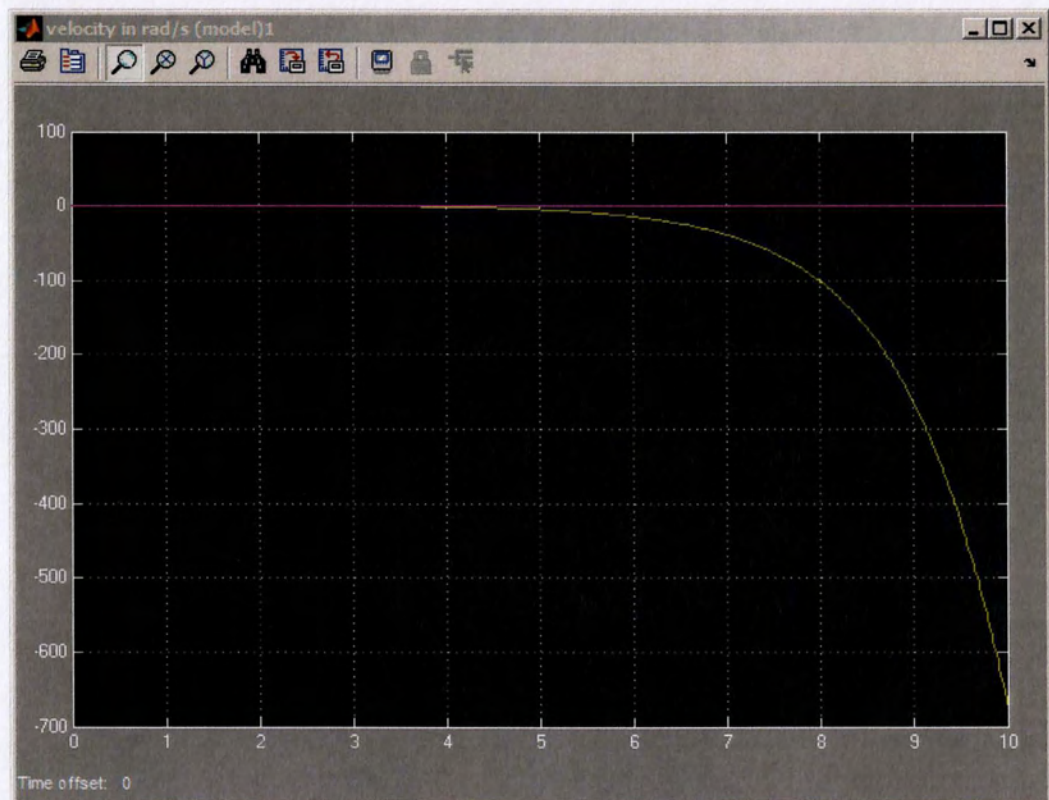
[ 1156979788303063/73786976294838206464, (3037*Kp)/5000 + 1467/50000]
[ 8746062533947541/2305843009213693952, (3037*Ki)/5000]
[ (3037*Kp)/5000 - (3513747617076402331*Ki)/1399370005431606560000 + 1467/50000, 0]
[ (3037*Ki)/5000, 0]

fx >>

```

Σχήμα 5-3: Πίνακας R-H με χρήση MATLAB

Μας ενδιαφέρει να μην υπάρχει αλλαγή πρόσημου στην πρώτη στήλη. Αφού τα 2 πρώτα στοιχεία της πρώτης στήλης είναι θετικοί αριθμοί, θα πρέπει και τα υπόλοιπα να είναι θετικά. Από το τελευταίο στοιχείο προκύπτει πολύ εύκολα ότι $K_i > 0$. Άρα ανεξάρτητα από το K_p , για να είναι ευσταθές το σύστημα πρέπει το K_i να είναι θετικό. Η απόκριση για αρνητικό K_i (-1) φαίνεται στο επόμενο σχήμα:



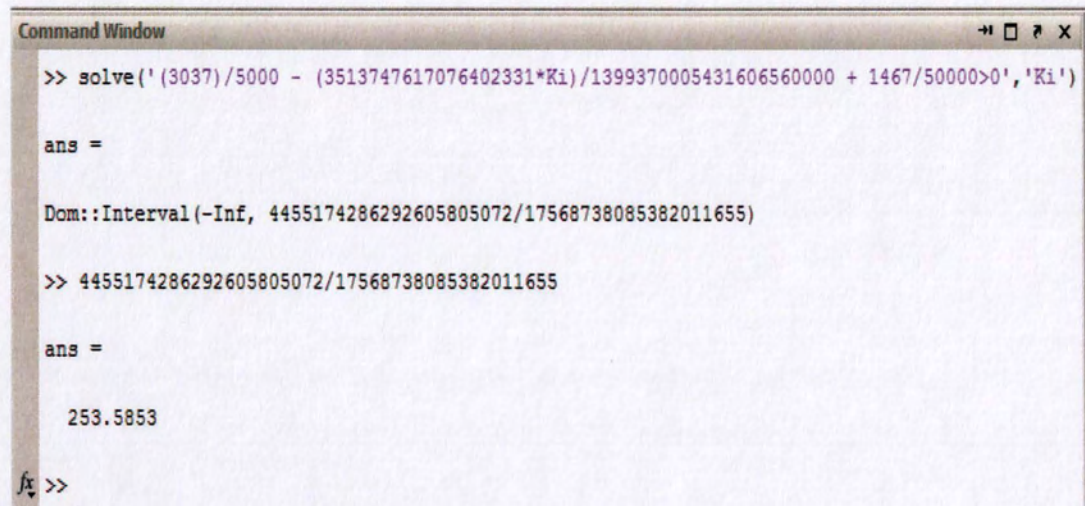
Σχήμα 5-4: Ασταθής απόκριση για αρνητικό K_i ($K_i = -1$)

Για να ολοκληρώσουμε την πληροφορία που μας δίνει ο πίνακας R-H εξετάζουμε και το τρίτο στοιχείο. Προκειμένου να μην υπάρχει καμία αλλαγή πρόσημου πρέπει και αυτό να είναι θετικό. Όμως εδώ έχουμε δύο μεταβλητές (K_p, K_i) και δεν γίνεται να λύσουμε ταυτόχρονα και για τις δύο. Οπότε ορίζουμε μια συγκεκριμένη τιμή για την μια και μετά λύνουμε την ανίσωση ως προς την άλλη. Έτσι βρίσκουμε ότι για την τιμή που ορίσαμε στην μια μεταβλητή, η άλλη πρέπει να ανήκει στο συγκεκριμένο διάστημα ώστε να έχουμε ευσταθές σύστημα.

Για παράδειγμα, έστω ότι $K_p=1$. Τότε το τρίτο στοιχείο γίνεται:

$$(3037)/5000 - (3513747617076402331 \cdot K_i)/1399370005431606560000 + 1467/50000$$

Για να έχουμε ευστάθεια πρέπει να είναι θετικό. Λύνοντας την ανίσωση ως προς K_i έχουμε:

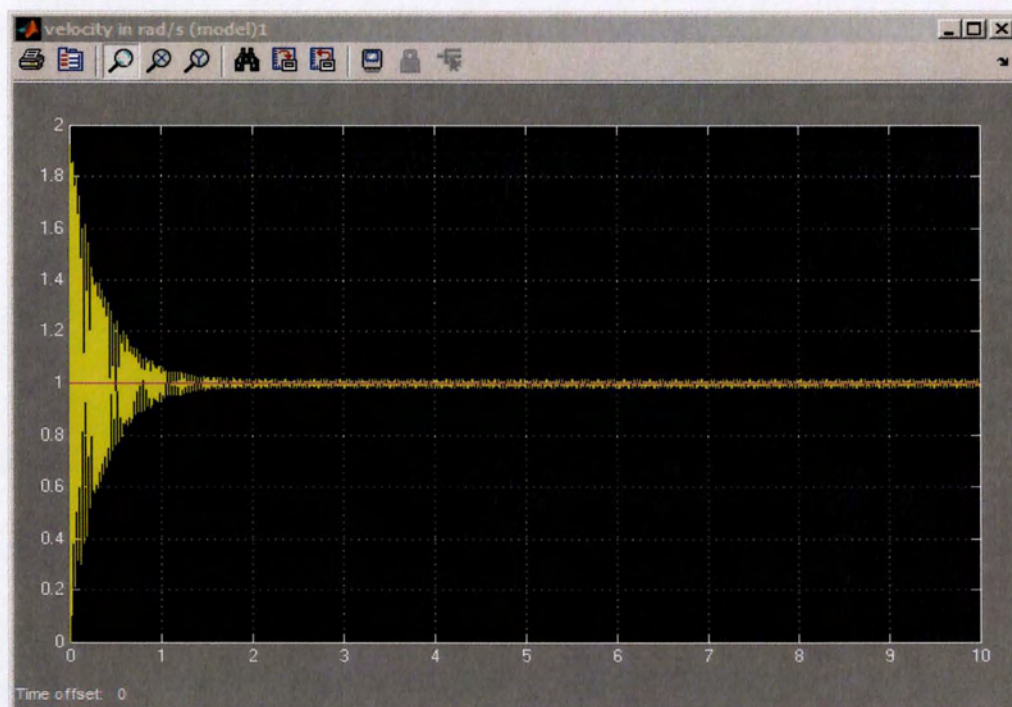


```
Command Window
>> solve('(3037)/5000 - (3513747617076402331*Ki)/1399370005431606560000 + 1467/50000>0','Ki')
ans =
Dom::Interval(-Inf, 4455174286292605805072/17568738085382011655)
>> 4455174286292605805072/17568738085382011655
ans =
253.5853
fx >>
```

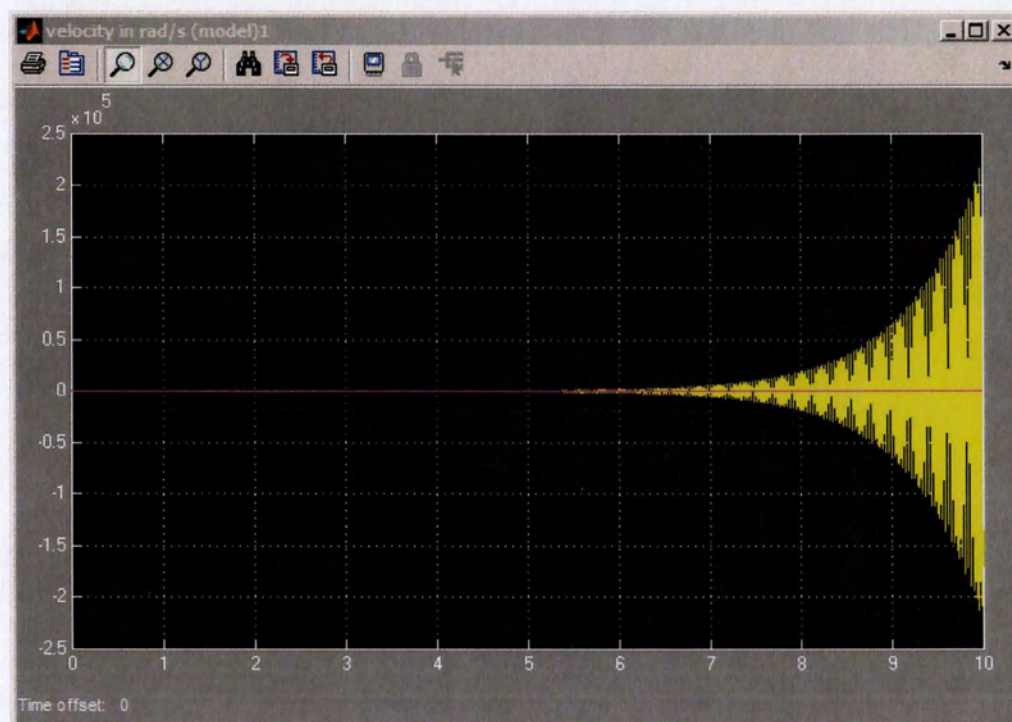
Η λύση της ανίσωσης είναι το διάστημα $(-\infty, 253.5853)$. Όμως το K_i πρέπει να είναι θετικό ανεξάρτητα του K_p (λόγω του τρίτου στοιχείου) οπότε το διάστημα είναι το $(0, 253.5853)$. Αυτό σημαίνει ότι για $K_p=1$ και $K_i \in (0, 253.5853)$ το σύστημά μας είναι ευσταθές.

Άρα συνοψίζοντας, για να μελετήσουμε την ευστάθεια του συστήματος ορίζουμε πρώτα το K_p ή το K_i , και λύνοντας την ανίσωση βρίσκουμε ένα πεδίο τιμών για την άλλη παράμετρο στο οποίο έχουμε ευστάθεια.

Για επιβεβαίωση των παραπάνω ακολουθούνε δύο αποκρίσεις, η μια ευσταθής και η άλλη ασταθής.



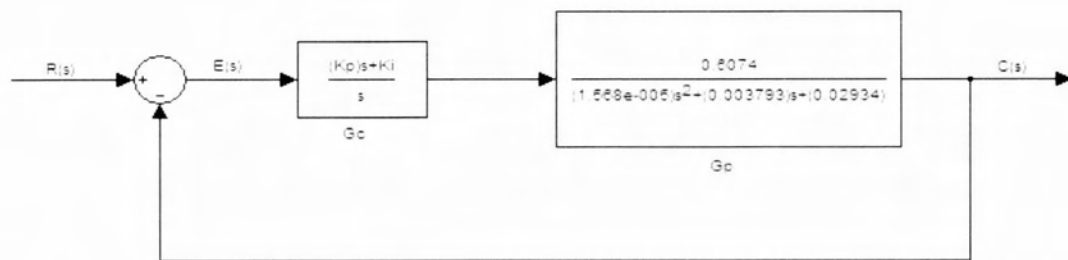
Σχήμα 5-5: Ευσταθής απόκριση για $K_p=1$ και $K_i=240$



Σχήμα 5-6: Ασταθής απόκριση για $K_p=1$ και $K_i=260$

5.3 Σφάλμα μόνιμης κατάστασης και ανάλυση με τη βοήθεια του τύπου των ριζών

Για να υπολογίσουμε το σφάλμα μόνιμης κατάστασης πρώτα υπολογίζουμε τη συνάρτηση μεταφοράς σφάλματος.



Από το δομικό διάγραμμα του συστήματος έχουμε:

$$\frac{E(s)}{R(s)} = \frac{R(s) - C(s)}{R(s)} = 1 - \frac{C(s)}{R(s)} = 1 - \frac{GcGp}{1 + GcGp} = \frac{1 + GcGp}{1 + GcGp} - \frac{GcGp}{1 + GcGp} = \frac{1}{1 + GcGp}$$

$$\Rightarrow E(s) = \frac{1}{1 + GcGp} * R(s)$$

Θεωρούμε για είσοδο τη μοναδιαία συνάρτηση βαθμίδας άρα:

$$E(s) = \frac{1}{1 + GcGp} * \frac{1}{s}$$

Σύμφωνα με το θεώρημα τελικής τιμής έχουμε:

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s * E(s) = \lim_{s \rightarrow 0} s * \frac{1}{1 + GcGp} * \frac{1}{s} = \lim_{s \rightarrow 0} \frac{1}{1 + GcGp}$$

$$\Rightarrow e_{ss} = \lim_{s \rightarrow 0} \frac{1}{1 + \frac{(Kp * s + Ki) * a}{s(bs^2 + cs + d)}} = \frac{1}{1 + \frac{Ki * a}{0}} = \frac{1}{1 + \infty} \Rightarrow e_{ss} = 0$$

Παρατηρούμε ότι με την είσοδο του ολοκληρωτικού όρου στον ελεγκτή μηδενίστηκε το σφάλμα μόνιμης κατάστασης σε είσοδο μοναδιαίας συνάρτησης βαθμίδας.

Υπενθυμίζουμε ότι με τον αναλογικό ελεγκτή είχαμε πεπερασμένο σφάλμα μόνιμης κατάστασης το οποίο μάλιστα δινόταν από τη σχέση:

$$e_{ss} = \frac{1}{1 + \frac{Kp * a}{d}}$$

ΤΟΠΟΣ ΤΩΝ ΡΙΖΩΝ

Όπως είπαμε και προηγουμένως, με την είσοδο του ολοκληρωτικού όρου το σύστημα μας έγινε τρίτης τάξης. Για την ακρίβεια, υπολογίσαμε την καινούρια συνάρτηση μεταφοράς:

$$\frac{C(s)}{R(s)} = \frac{Kp * a * s + Ki * a}{bs^3 + cs^2 + (d + Kp * a)s + Ki * a}$$

Αυτό δεν μας επιτρέπει να εξάγουμε κλειστές σχέσεις για τις παραμέτρους ζ και ω των πόλων του συστήματος. Όμως μπορούμε να βρούμε αυτές τις παραμέτρους για τον κάθε πόλο από τον τόπο των ριζών.

Οι αποστάσεις των πόλων από τον πραγματικό και φανταστικό άξονα συνεχίζουν να καθορίζουν τα βασικά χαρακτηριστικά της απόκρισης ανεξάρτητα από την τάξη του συστήματος. Απλά αντί να έχουμε 2 πόλους θα έχουμε 3.

Πρέπει να τονίσουμε επίσης και το εξής: Στη συνάρτηση μεταφοράς κλειστού βρόχου έχει εισαχθεί ένας μηδενιστής. Για το λόγο αυτό οι παράμετροι ζ και ω δεν επαρκούν πλέον για τον πλήρη καθορισμό της χρονικής απόκρισης, αλλά παρέχουν σε συνδυασμό με τη θέση του μηδενιστή κλειστού βρόχου μόνο μια ποιοτική ένδειξη της.

Η προσθήκη του μηδενιστή κλειστού βρόχου (closed-loop zero) επηρεάζει τη μεταβατική απόκριση χωρίς να μεταβάλει τη ποιοτική της δομή (η χαρακτηριστική εξίσωση παραμένει ανέπαφη). Γενικά, έχει την τάση να μειώνει την ευστάθεια του συστήματος, διότι μειώνει το χρόνο ανύψωσης

(αυξάνει την ταχύτητα απόκρισης) και αυξάνει τη μέγιστη υπερακόντιση (όπως ακριβώς θα συνέβαινε και με μείωση της τιμής του λόγου απόσβεσης ζ).

Από τα παραπάνω γίνεται φανερό ότι η ανάλυση με τον τόπο των ριζών γίνεται πιο περίπλοκη γιατί τώρα εκτός από τη θέση των πόλων (που καθορίζει το ζ και ω του συστήματος) πρέπει να συμπεριλάβουμε και τη θέση του μηδενιστή γιατί παίζει σημαντικό ρόλο στην μορφή της απόκρισης. Για παράδειγμα, στα συστήματα με υπέρ-απόσβεση, εάν ο μηδενιστής βρίσκεται μεταξύ των πραγματικών (αρνητικών) πόλων και της αρχής, η έξοδος υπερακοντίζει την τιμή της μόνιμης κατάστασης παρόλο που έχουμε $\zeta=1$ ή μεγαλύτερο.

Άρα με την είσοδο ενός μηδενιστή κλειστού βρόχου, είναι δυνατόν ένα σύστημα με υπέρ-απόσβεση (π.χ. $\zeta=1.25$), να συμπεριφερθεί όπως ένα σύστημα με υπό-απόσβεση (π.χ. $\zeta=0.7$).

Γενικά, η παρουσία μηδενιστών στη συνάρτηση μεταφοράς κλειστού βρόχου οδηγεί σε ποικίλες αποκρίσεις, των οποίων η μορφή εξαρτάται τόσο από την απόλυτη όσο και από τη σχετική θέση των πόλων και μηδενιστών. Όταν σε ένα σύστημα τρίτης τάξης ο πραγματικός μηδενιστής συμπίπτει με ένα πραγματικό πόλο, τότε αλληλοαπαλείφονται και το σύστημα γίνεται ουσιαστικά δεύτερης τάξης.

Συνεχίζουμε με τη σχεδίαση του τόπου των ριζών. Η χαρακτηριστική εξίσωση είναι:

$$bs^3 + cs^2 + (d + Kp * a)s + Ki * a = 0$$

Δεδομένου ότι είναι τρίτης τάξης θα έχει είτε τρεις πραγματικές ρίζες, είτε μια πραγματική και ένα ζεύγος μιγαδικών συζυγών ριζών. Για να σχεδιάσουμε τον τόπο των ριζών πρέπει να γράψουμε την χαρακτηριστική εξίσωση στη μορφή:

$$1 + K * G(s) = 0$$

Όπου K είναι η παράμετρος για την οποία γίνεται ο τόπος των ριζών. Μπορεί να είναι μια οποιαδήποτε μεταβλητή του συστήματος. Στην περίπτωση μας θέλουμε να σχεδιάσουμε τον Τ.Ρ. ως προς το K_i για να μελετήσουμε την επίδραση του κέρδους του ολοκληρωτικού ελεγκτή στην

απόκριση και ευστάθεια του συστήματος. Αυτό βέβαια προϋποθέτει ότι θα δώσουμε μια συγκεκριμένη τιμή στο K_p καθώς δεν γίνεται να σχεδιαστεί ο Τ.Ρ. για 2 μεταβλητές. Στη συνέχεια μπορούμε να δώσουμε μια άλλη τιμή στο K_p και να τον ξανασχεδιάσουμε.

Για να γίνει ο Τ.Ρ. ως προς το K_i πρέπει το K_i να εμφανίζεται σαν πολλαπλασιαστικός παράγοντας στην χαρακτηριστική εξίσωση. Για το σκοπό αυτό, διαιρούμε και τα 2 μέλη με το άθροισμα των όρων που δεν περιέχουν το K_i . Έτσι έχουμε:

$$bs^3 + cs^2 + (d + K_p * a)s + K_i * a = 0 \Rightarrow 1 + \frac{K_i * a}{bs^3 + cs^2 + (d + K_p * a)s} = 0$$

$$\Rightarrow 1 + K_i * \frac{a}{bs^3 + cs^2 + (d + K_p * a)s} = 0$$

Αφού φέραμε την χαρακτηριστική εξίσωση στην επιθυμητή μορφή σχεδιάζουμε τον τόπο των ριζών με το MATLAB. Πρέπει όμως πρώτα να δώσουμε μια τιμή στο K_p για να συνεχίσουμε. Από την ανάλυση του συστήματος με τον αναλογικό ελεγκτή είχαμε βρει ότι για $K_p=1$ έχουμε συζυγείς μιγαδικούς πόλους με αρνητικά πραγματικά μέρη. Θα χρησιμοποιήσουμε αυτή την τιμή σαν μια αρχική εκτίμηση και στη συνέχεια θα ξανασχεδιάσουμε τον τόπο των ριζών και για άλλες τιμές του K_p για να μελετήσουμε την επίδρασή του στους πόλους του συστήματος.

Πρώτα ορίζουμε τον αριθμητή και παρανομαστή της $G(s)$, μετά τη συνάρτηση μεταφοράς, και τέλος χρησιμοποιούμε την εντολή `flocus` για τη σχεδίαση του τόπου των ριζών.

Εάν δώσουμε για είσοδο μόνο την $G(s)$ τότε το MATLAB θα θεωρήσει θετικές τιμές του K_i . Κάτι τέτοιο είναι επιθυμητό γιατί είχαμε βρει με το κριτήριο ευστάθειας Routh-Hurwitz ότι το K_i πρέπει να είναι πάντα θετικό για να έχουμε ευστάθεια.

Οι απαραίτητες εντολές καθώς και ο τόπος των ριζών φαίνονται παρακάτω.


```
Command Window

>> num=[a]

num =

    0.6074

>> den=[b c d+a 0]

den =

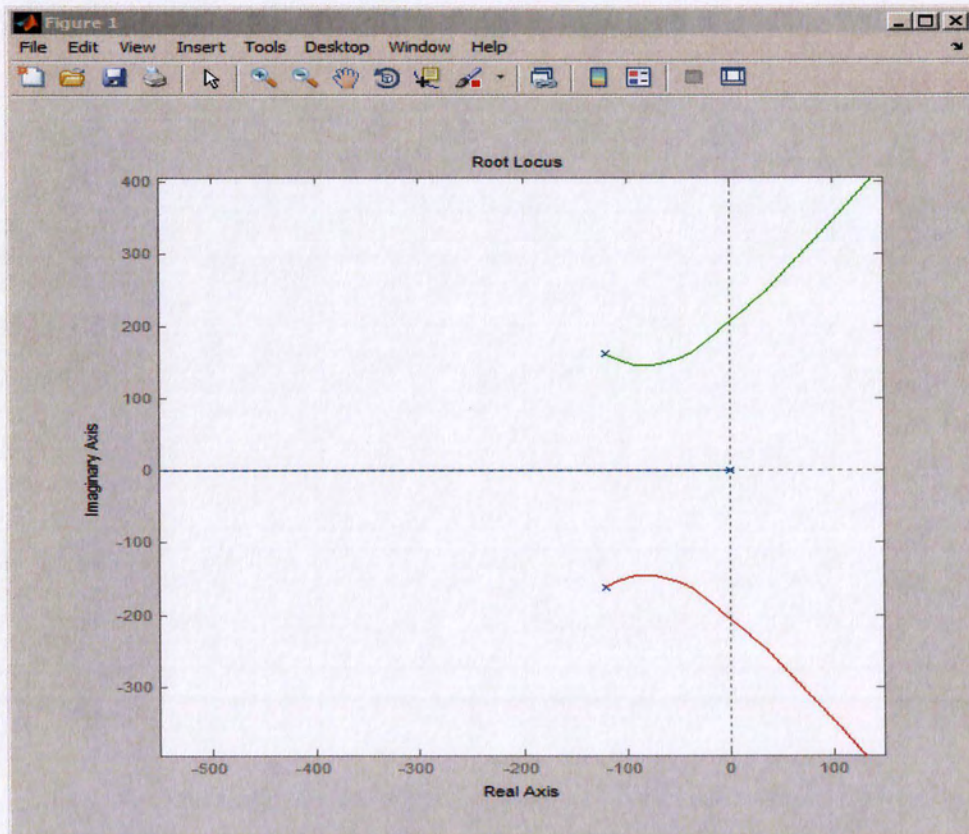
    0.0000    0.0038    0.6367    0

>> sys=tf(num,den)

Transfer function:
           0.6074
-----
1.568e-005 s^3 + 0.003793 s^2 + 0.6367 s

>> rlocus(sys)
fx >>
```

Σχήμα 5-7: Εντολές MATLAB για σχεδίαση του τόπου των ριζών



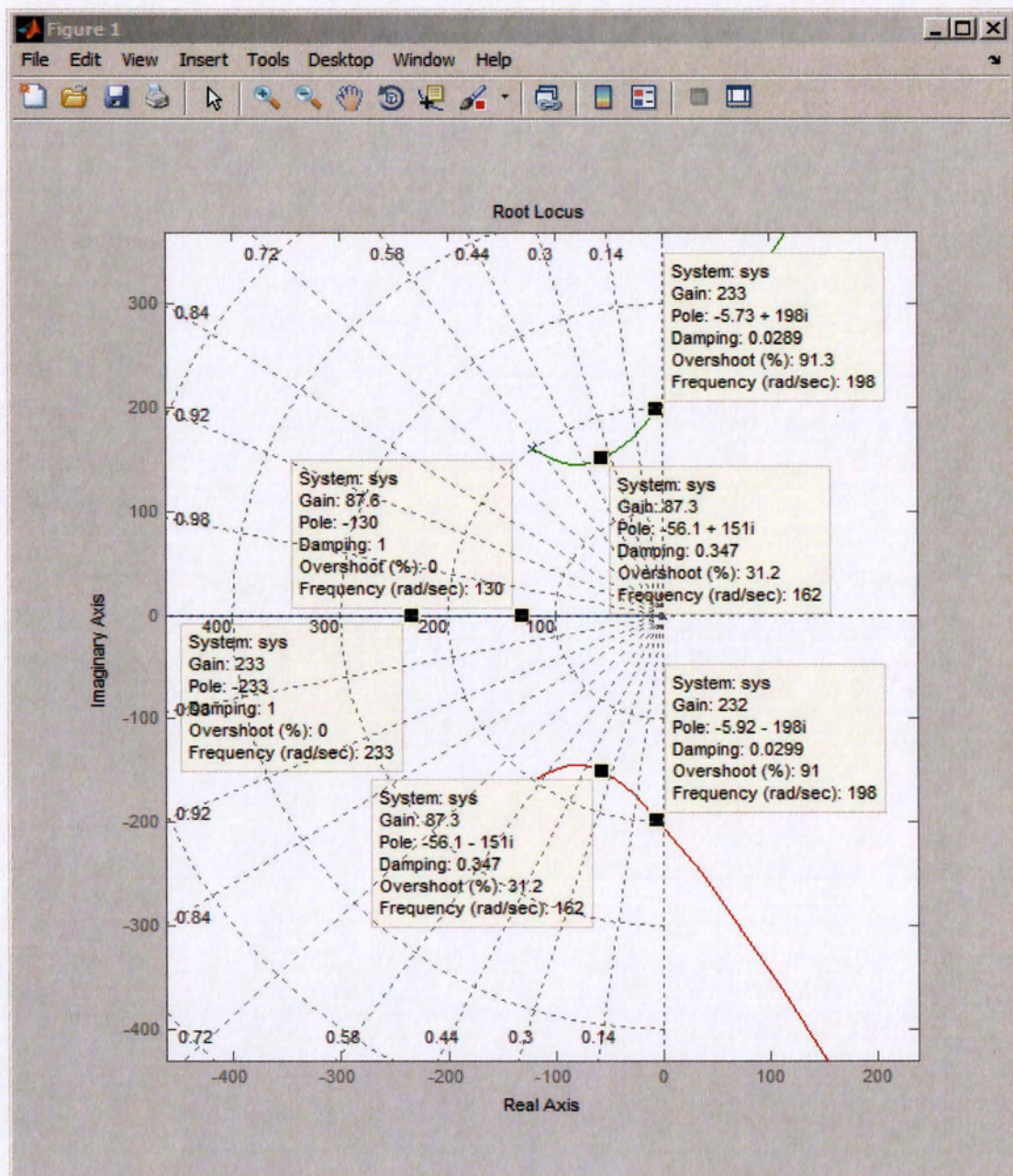
Σχήμα 5-8: Τόπος των ριζών για $K_p=1$ και K_i θετικό

Παρατηρούμε ότι ο τόπος των ριζών έχει τρεις κλάδους και ο κάθε κλάδος έχει το δικό του χρώμα. Αυτό γιατί η χαρακτηριστική εξίσωση είναι τρίτης τάξης και σε κάθε τιμή του K_i αντιστοιχούν τρεις πόλοι, ένας σε κάθε κλάδο.

Επομένως σε κάθε σημείο που βλέπουμε στον τόπο των ριζών αντιστοιχούν άλλα δύο που έχουν το ίδιο K_i . Αυτά βρίσκονται εύκολα γιατί από τους τρεις πόλους οι δύο είναι συζυγείς μιγαδικοί που φαίνονται στον πράσινο και κόκκινο κλάδο και ο ένας πραγματικός αριθμός που φαίνεται στον μπλε κλάδο.

Για να πάρουμε πληροφορίες για έναν πόλο κάνουμε αριστερό κλικ πάνω του. Έτσι βλέπουμε διάφορα χαρακτηριστικά του όπως το κέρδος (K_i) που του αντιστοιχεί και το συντελεστή απόσβεσης. Κρατώντας πατημένο το πλήκτρο του ποντικιού μπορούμε να μετακινούμε τον πόλο στο μιγαδικό επίπεδο και να βλέπουμε πως αλλάζουν οι τιμές των παραμέτρων. Έτσι μπορούμε να βρούμε τους 3 πόλους που αντιστοιχούν σε κάποιο K_i (έχουν το ίδιο κέρδος). Παράλληλα μπορούμε να δούμε και τον συντελεστή απόσβεσης των μιγαδικών πόλων που καθορίζει μαζί με τη θέση του μηδενιστή τις ταλαντώσεις στη μεταβατική κατάσταση.

Στο επόμενο σχήμα φαίνονται έξι πόλοι. Τρεις για $K_i=87$ και τρεις για $K_i=233$. Παρατηρούμε ότι οι γραμμές του T.P. εκτείνονται και στο δεξί ημιεπίπεδο, πράγμα που σημαίνει ότι έχουμε αστάθεια. Αυτό συμβαίνει για τιμές του K_i κοντά στο 250, γι' αυτό το λόγο οι πόλοι για $K_i=233$ είναι πολύ κοντά στον φανταστικό άξονα. Την αστάθεια αυτή την είχαμε υπολογίσει με το κριτήριο Routh-Hurwitz. Για την ακρίβεια είχαμε βρει ότι για $K_p=1$ και $K_i \in (0, 253.5853)$ το σύστημά μας είναι ευσταθές. Άρα οι γραμμές του T.P. που εκτείνονται στο δεξί ημιεπίπεδο αντιστοιχούν σε $K_i > 254$.



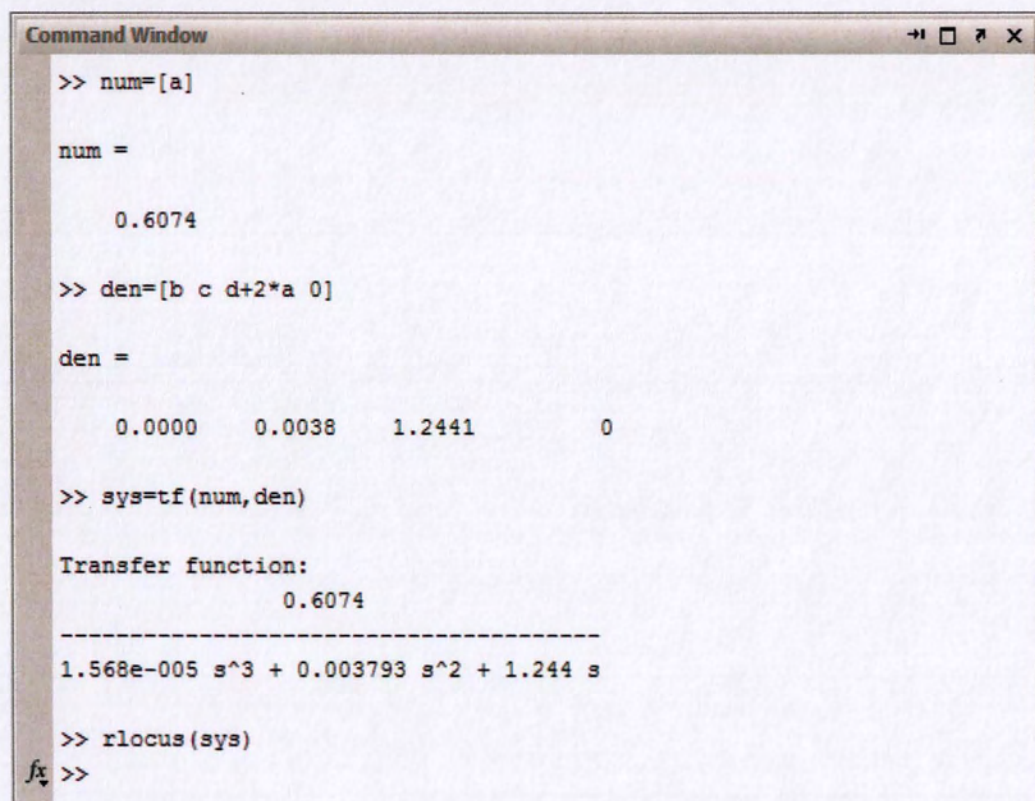
Σχήμα 5-9: Πόλοι και χαρακτηριστικά τους ($K_p=1$)

Σχετικά με το σχήμα 9 πρέπει να παρατηρήσουμε και το εξής. Όσο αυξάνεται το κέρδος (K_i) οι 2 μιγαδικοί πόλοι μετακινούνται προς τα δεξιά (προς τον φανταστικό άξονα) και ο πραγματικός πόλος προς τα αριστερά. Οι ταλαντώσεις στη μεταβατική κατάσταση εξαρτώνται ως ένα βαθμό από το συντελεστή απόσβεσης. Απ' ότι φαίνεται στο σχήμα, οι 2 μιγαδικοί κλάδοι ξεκινάνε από συντελεστή απόσβεσης περίπου 0.58, και όσο αυξάνεται το κέρδος και μετατοπίζονται προς τα δεξιά

μειώνεται όλο και περισσότερο. Άρα ο μέγιστος δυνατός συντελεστής απόσβεσης που μπορούμε να έχουμε είναι περίπου 0.58 καθώς οποιαδήποτε αύξηση του K_i θα τον μειώσει. Το αποτέλεσμα θα είναι βέβαια η αύξηση της υπερακόντισης και των ταλαντώσεων στη μεταβατική κατάσταση.

Αυτός ο περιορισμός που έχουμε στον συντελεστή απόσβεσης οφείλεται εν μέρει και στο K_p . Ας μην ξεχνάμε ότι ο Τ.Ρ. έχει γίνει για $K_p=1$. Εάν διαλέξουμε ένα διαφορετικό K_p θα πάρουμε έναν διαφορετικό τόπο ριζών. Όμως επειδή δεν έχουμε στη διάθεση μας αναλυτικές σχέσεις που να περιγράφουν τη σχέση του ζ με τα K_p και K_i , πρέπει να χρησιμοποιήσουμε τον τόπο των ριζών για να τη μάθουμε.

Επομένως θα δοκιμάσουμε να σχεδιάσουμε τον Τ.Ρ. για μεγαλύτερο K_p (έστω $K_p=2$) ώστε να τον συγκρίνουμε με τον προηγούμενο για $K_p=1$. Οι εντολές φαίνονται παρακάτω.



```
Command Window
>> num=[a]

num =

    0.6074

>> den=[b c d+2*a 0]

den =

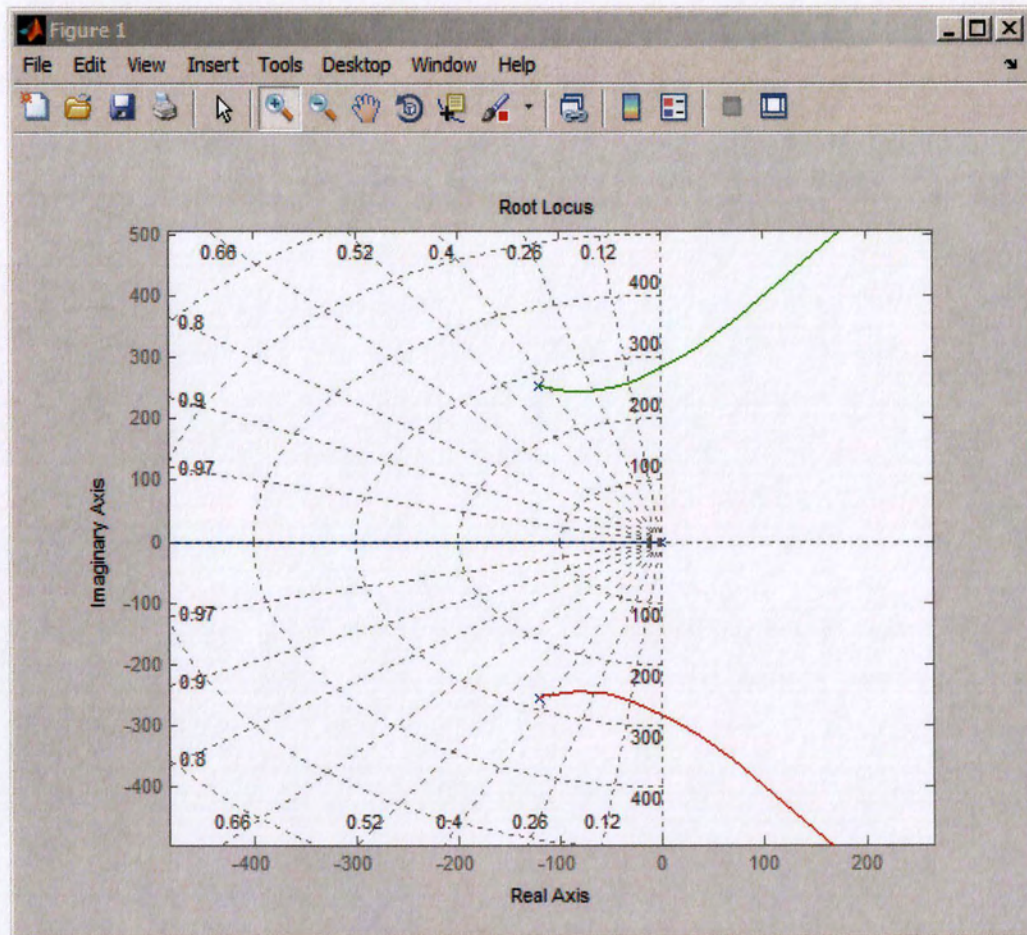
    0.0000    0.0038    1.2441         0

>> sys=tf(num,den)

Transfer function:
           0.6074
-----
1.568e-005 s^3 + 0.003793 s^2 + 1.244 s

>> rlocus(sys)
fx >>
```

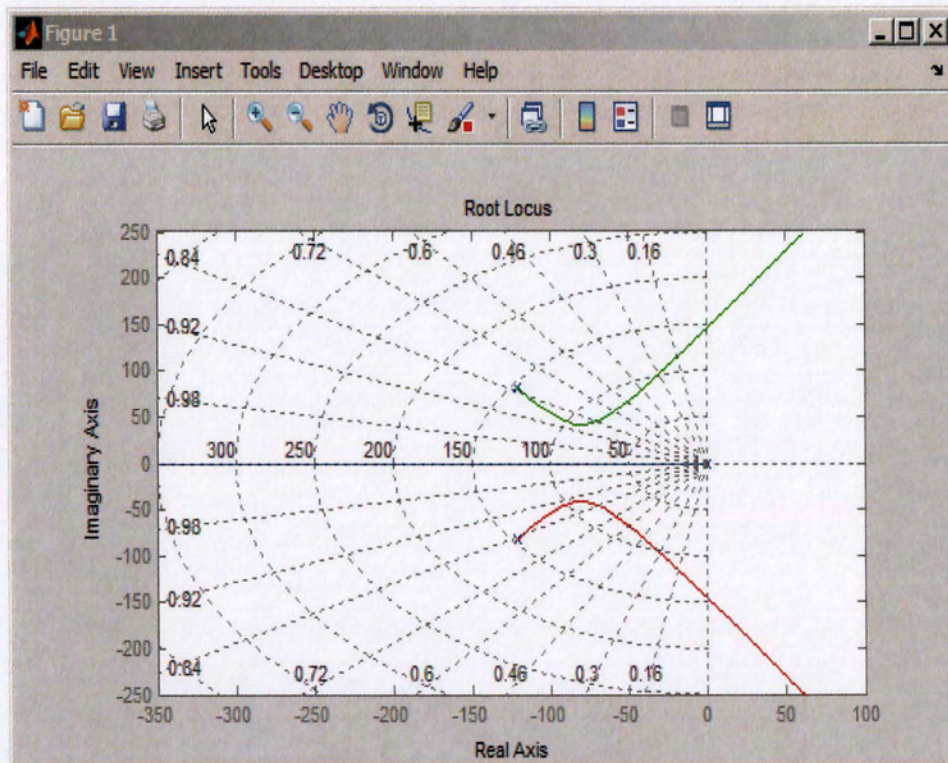
Σχήμα 5-10: Εντολές MATLAB για σχεδίαση Τ.Ρ με $K_p=2$ (K_i θετικό)



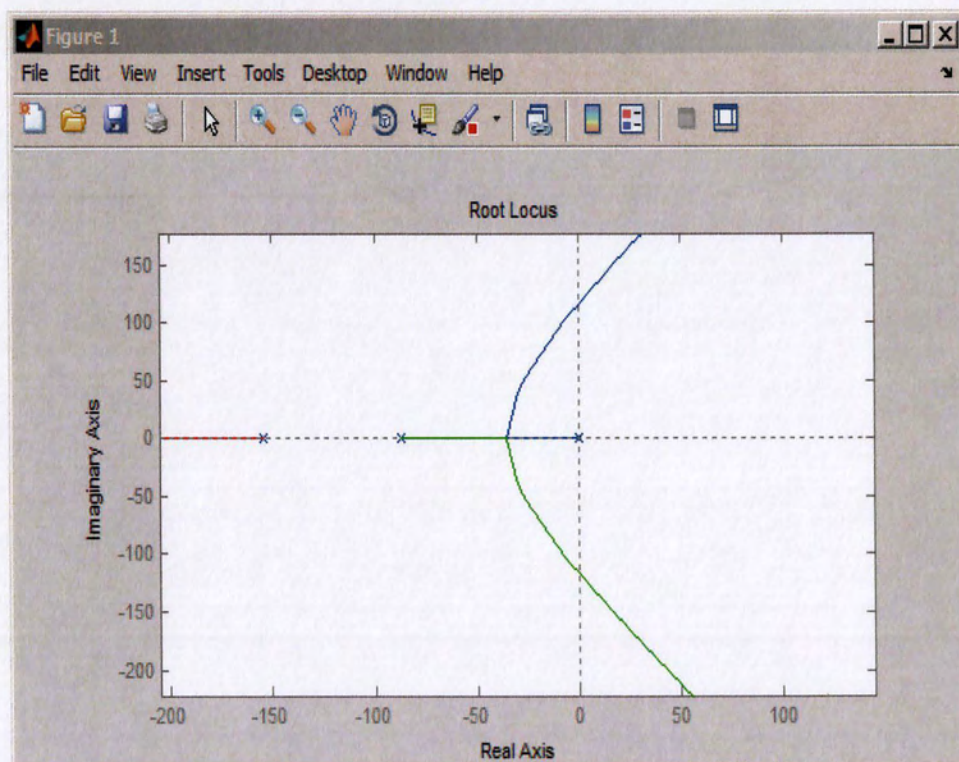
Σχήμα 5-11: Τόπος των ριζών για $K_p=2$, K_i θετικό

Παρατηρούμε ότι για $K_p=2$ οι κλάδοι του Τ.Ρ. ξεκινάνε από συντελεστή απόσβεσης περίπου 0.4 που είναι μικρότερο από το 0.58 που είχαμε πριν. Άρα μπορούμε να συμπεράνουμε ότι όσο αυξάνεται το K_p μειώνεται ο συντελεστής απόσβεσης.

Προφανώς και δεν μας ικανοποιεί ένας τόπος ριζών με μέγιστο συντελεστή απόσβεσης 0.4 ή 0.58 γιατί θα έχουμε αρκετά μεγάλη υπερακόντιση και ταλάντωση στη μεταβατική κατάσταση. Θα επιθυμούσαμε έναν πιο μεγάλο συντελεστή απόσβεσης που θα μπορούσε να φτάσει και τη μονάδα σε περίπτωση που δεν θέλουμε καθόλου ταλαντώσεις. Επομένως πρέπει να μειώσουμε το K_p . Παρακάτω ακολουθούν οι τόποι των ριζών για $K_p=0.5$ και $K_p=0.3$.



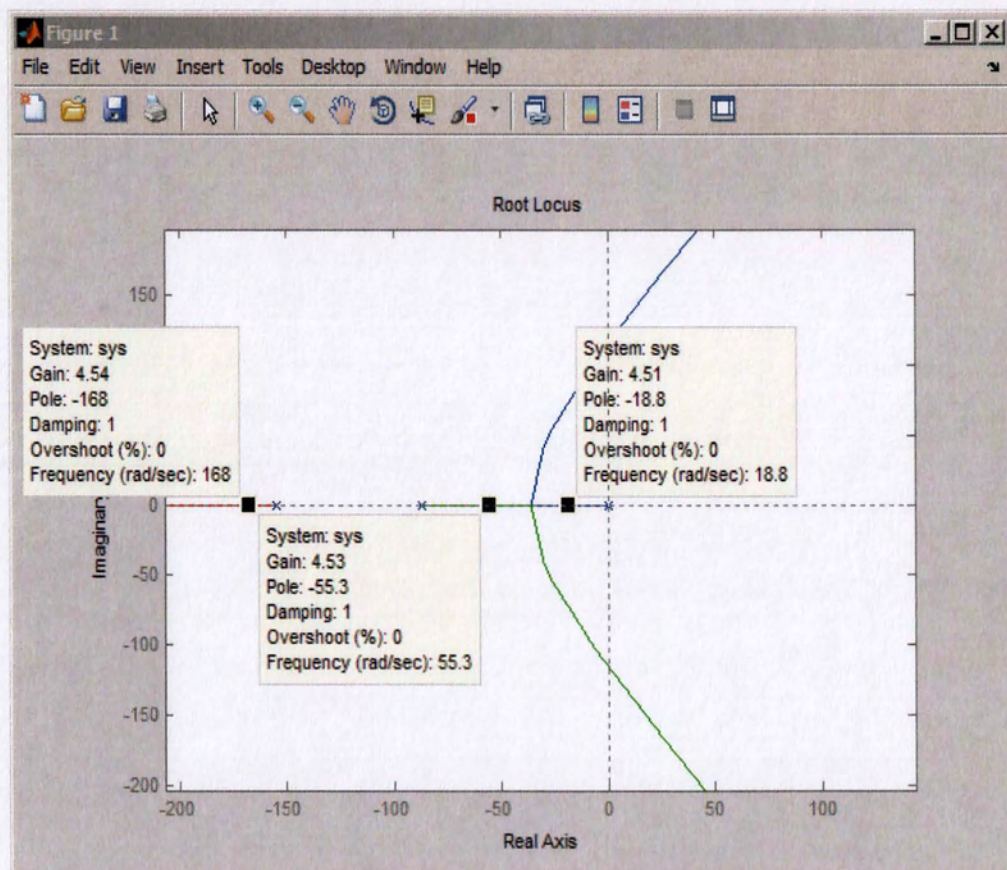
Σχήμα 5-12: Τόπος των ριζών για $K_p=0.5$, K_i θετικό



Σχήμα 5-13: Τόπος των ριζών για $K_p=0.3$, K_i θετικό

Παρατηρούμε ότι για $K_p=0.5$ έχουμε μεγαλύτερη απόσβεση που ξεκινάει από 0.84 περίπου ενώ για $K_p=0.3$ οι δύο μιγαδικοί κλάδοι έχουν και τμήματα πάνω στον αρνητικό πραγματικό άξονα, δηλαδή πόλους με συντελεστή απόσβεσης μεγαλύτερο ή ίσο με τη μονάδα.

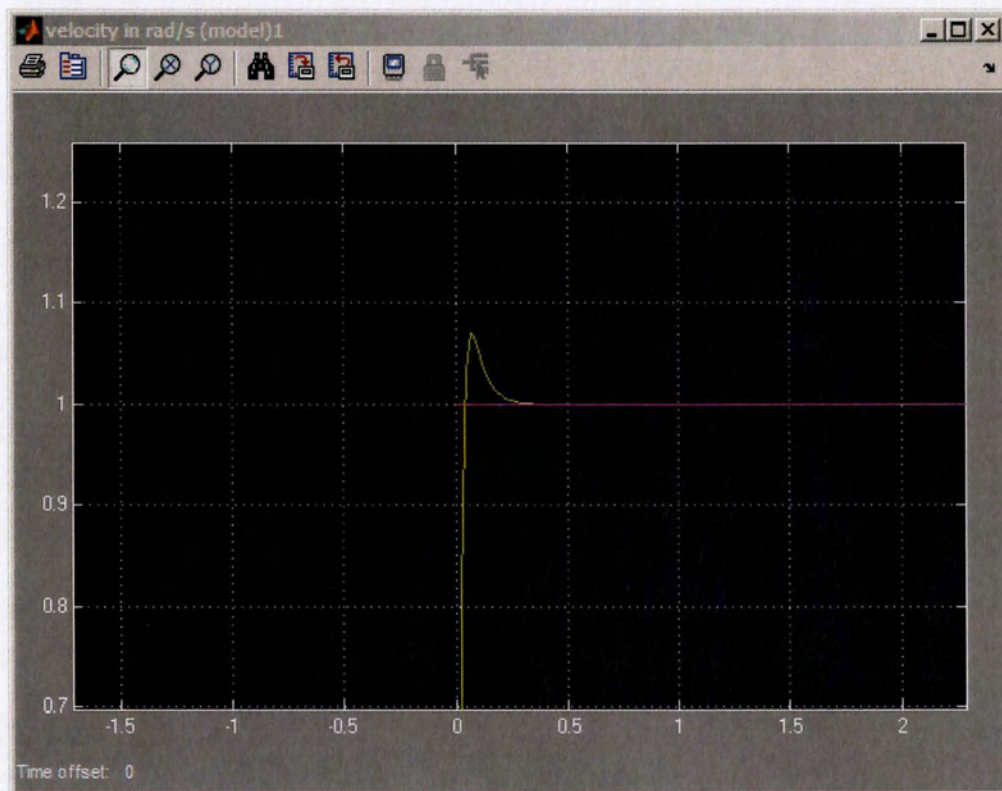
Ένα σημαντικό σημείο που πρέπει να προσέξουμε είναι το εξής. Στα συστήματα χωρίς μηδενιστή στη συνάρτηση μεταφοράς κλειστού βρόχου, όταν ο συντελεστής απόσβεσης είναι ίσος με τη μονάδα αυτό σημαίνει ότι στη μεταβατική κατάσταση δεν υπάρχουν ούτε ταλαντώσεις ούτε υπερακόντιση. Αυτό δεν ισχύει στα συστήματα με μηδενιστή. Δηλαδή είναι δυνατό να διαλέξουμε πόλους που έχουν συντελεστή απόσβεσης ίσο με τη μονάδα ή μεγαλύτερο (αρνητικούς και πραγματικούς πόλους) και να εμφανίζεται υπερακόντιση. Όπως για παράδειγμα στο επόμενο σχήμα.



Σχήμα 5-14: Αρνητικοί πραγματικοί πόλοι στον τόπο των ριζών για $K_p=0.3$

Στο παραπάνω σχήμα βλέπουμε τους πόλους που αντιστοιχούν σε $K_i=4.5$ (και $K_p=0.3$).

Παρατηρούμε ότι είναι όλοι πάνω στον αρνητικό πραγματικό άξονα, άρα έχουν συντελεστή απόσβεσης ίσο με τη μονάδα (ή περισσότερο). Όμως εάν εκτελέσουμε το μοντέλο στο Simulink (το οποίο χρησιμοποιεί αριθμητικές μεθόδους) για $K_i=4.5$ και $K_p=0.3$ η απόκριση που παίρνουμε είναι η εξής:



Σχήμα 5-15: Απόκριση μοντέλου στο Simulink για $K_i=4.5$ και $K_p=0.3$

Παρατηρούμε ότι ενώ το σύστημά μας δεν παρουσιάζει ταλαντωτική συμπεριφορά, υπάρχει μια μικρή υπερακόντιση στη μεταβατική του κατάσταση. Αυτό κανονικά δεν θα έπρεπε να συμβαίνει γιατί οι πόλοι για τις συγκεκριμένες τιμές είναι πραγματικοί (και αρνητικοί). Άρα δεν θα έπρεπε να υπάρχει καθόλου υπερακόντιση στη μεταβατική κατάσταση.

Όμως δεν πρόκειται για σφάλμα στον τύπο των ριζών ή στη μοντελοποίηση, αλλά έχει να κάνει με τον μηδενιστή της συνάρτησης μεταφοράς κλειστού βρόχου που αναφέραμε νωρίτερα:

$$\frac{C(s)}{R(s)} = \frac{K_p * a * s + K_i * a}{bs^3 + cs^2 + (d + K_p * a)s + K_i * a}$$

και μπορεί να αποδειχθεί παίρνοντας τον αντίστροφο μετασχηματισμό Laplace της απόκρισης σε είσοδο μοναδιαίας συνάρτησης βαθμίδας.

$$C(s) = \frac{C(s)}{R(s)} R(s) = \frac{Kp * a * s + Ki * a}{bs^3 + cs^2 + (d + Kp * a)s + Ki * a} * \frac{1}{s}$$

$$\Rightarrow C(s) = \frac{Kp * a * s + Ki * a}{bs^4 + cs^3 + (d + Kp * a)s^2 + Ki * as}$$

Αφού θέλουμε να μελετήσουμε την απόκριση για τους συγκεκριμένους πόλους θέτουμε $Ki=4.5$ και $Kp=0.3$ οπότε η απόκριση γίνεται:

$$C(s) = \frac{0.3 * a * s + 4.5 * a}{bs^4 + cs^3 + (d + 0.3 * a)s^2 + 4.5 * as}$$

Για να πάρουμε τον αντίστροφο μετασχηματισμό Laplace της απόκρισης πρώτα κάνουμε ανάλυση σε μερικά κλάσματα. Έτσι θα έχουμε την απόκριση στη μορφή:

$$C(s) = \frac{R_1}{s - p_1} + \frac{R_2}{s - p_2} + \frac{R_3}{s - p_3} + \frac{R_4}{s - p_4}$$

Όπου οι συντελεστές R και p υπολογίζονται με το MATLAB ως εξής:

```

Command Window
>> num=[0.3*a 4.5*a]

num =

    0.1822    2.7333

>> den=[b c d+0.3*a 4.5*a 0]

den =

    0.0000    0.0038    0.2116    2.7333    0

>> [r,p,k]=residue(num,den)

r =

    0.6344
   -2.0542
    0.4198
    1.0000

p =

  -167.6050
   -55.5843
   -18.7112
         0

k =

    []

fx >>

```

Σχήμα 5-16: Υπολογισμός συντελεστών R και p με το MATLAB

Άρα η απόκριση παίρνει τη μορφή:

$$C(s) = \frac{0.6344}{s - (-167.6050)} + \frac{-2.0542}{s - (-55.5843)} + \frac{0.4198}{s - (-18.7112)} + \frac{1}{s - 0}$$

$$\Rightarrow C(s) = \frac{0.6344}{s + 167.6050} + \frac{-2.0542}{s + 55.5843} + \frac{0.4198}{s + 18.7112} + \frac{1}{s}$$

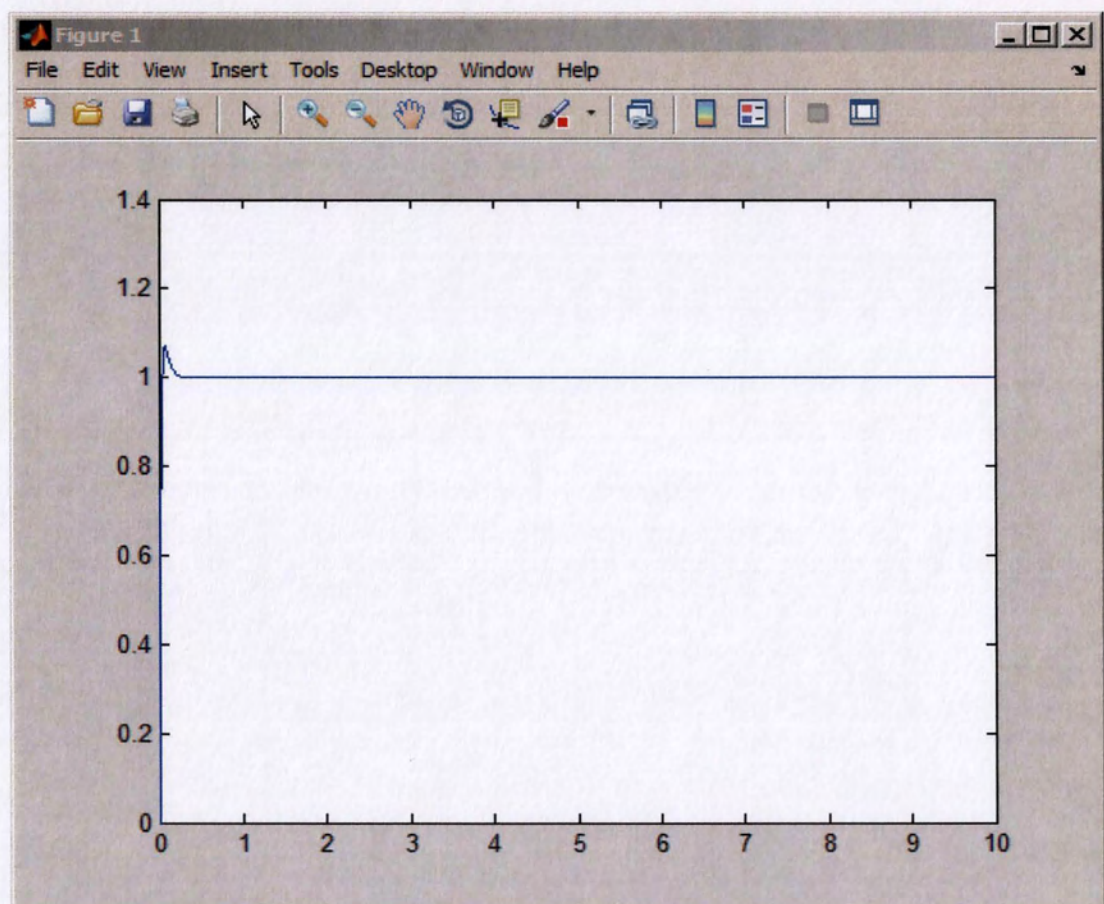
Τώρα είναι εύκολο να πάρουμε τον αντίστροφο μετασχηματισμό Laplace του κάθε όρου ξεχωριστά και να τους αθροίσουμε για να βρούμε την $c(t)$.

$$c(t) = 0.6344e^{-167.6050t} - 2.0542e^{-55.5843t} + 0.4198e^{-18.7112t} + 1$$

Τέλος, απομένει να κάνουμε τη γραφική παράσταση της παραπάνω συνάρτησης.

```
Command Window
>> t=[0:0.01:10];
>> h=0.6344*exp(-167.6050*t)-2.0542*exp(-55.5843*t)+0.4198*exp(-18.7112*t)+1;
>> plot(t,h,'DisplayName','h vs. t','XDataSource','t','YDataSource','h');figure(gcf)
fx >>
```

Με τον παραπάνω κώδικα MATLAB η γραφική παράσταση της απόκρισης είναι:



Σχήμα 5-17: Απόκριση για $K_i=4.5$ και $K_p=0.3$

Άρα βλέπουμε ότι η υπερακόντιση είναι όντως κομμάτι της απόκρισης ακόμα και για πραγματικούς αρνητικούς πόλους.

Η θέση του μηδενιστή που προκαλεί αυτή τη συμπεριφορά υπολογίζεται πολύ εύκολα εξισώνοντας με το μηδέν τον αριθμητή της συνάρτησης μεταφοράς κλειστού βρόχου. Έτσι έχουμε:

$$Kp * a * s + Ki * a = 0 \Rightarrow Kp * a * s = -Ki * a \Rightarrow s = -\frac{Ki}{Kp}$$

Έτσι μπορούμε να υπολογίζουμε τη θέση του μηδενιστή για κάθε K_i και K_p έτσι ώστε να τη συγκρίνουμε με τη θέση των αντίστοιχων πόλων για τα ίδια K_i και K_p . Για την περίπτωση που μόλις εξετάσαμε ($K_i=4.5$ και $K_p=0.3$) προκύπτει πολύ εύκολα ότι ο μηδενιστής βρίσκεται στη θέση $s = -15$.

Από το σχήμα 14 βλέπουμε ότι ο πόλος στα δεξιά βρίσκεται στο σημείο -18.8 . Άρα ο μηδενιστής βρίσκεται στα δεξιά του. Όπως είπαμε στην αρχή του κεφαλαίου, στα συστήματα με υπέρ-απόσβεση, όταν ο μηδενιστής βρίσκεται ανάμεσα στους πόλους και την αρχή, η απόκριση παρουσιάζει υπερακόντιση. Προσοχή, αυτό δεν σημαίνει ότι έχουμε υπό-απόσβεση. Απλά το σύστημα συμπεριφέρεται όπως ένα σύστημα με υπό-απόσβεση λόγω του μηδενιστή.

Εάν θέλουμε να εξαλείψουμε εντελώς την υπερακόντιση πρέπει να μετακινήσουμε τον μηδενιστή έτσι ώστε να συμπίπτει με τον δεξιό πόλο. Έτσι μηδενιστής και πόλος αλληλοαπαλείφονται και το σύστημα γίνεται ουσιαστικά δεύτερης τάξης. Αφού θα έχουμε σύστημα δεύτερης τάξης με 2 πραγματικούς αρνητικούς και άνισους πόλους τότε θα έχουμε (πραγματική) υπέρ-απόσβεση. Αυτή η υπέρ-απόσβεση θα είναι πλέον χωρίς μηδενιστή οπότε δεν θα έχει υπερακόντιση.

Άρα θέλουμε να βρούμε για ποιά τιμή του K_i (με $K_p=0.3$) ο μηδενιστής κλειστού βρόχου συμπίπτει με τον δεξιό πόλο κλειστού βρόχου.

Αυτό δεν είναι εύκολο γιατί τόσο ο μηδενιστής όσο και ο πόλος εξαρτώνται από το K_i με διαφορετικό τρόπο. Θα μπορούσαμε να ακολουθήσουμε διαδικασία δοκιμής και σφάλματος μέχρι να βρούμε μια τιμή για το K_i η οποία θα τοποθετούσε τον μηδενιστή στην ίδια θέση με τον δεξιό πόλο, αλλά κάτι τέτοιο θα απαιτούσε πολλές δοκιμές και δεν θα ήταν ακριβές.

Για να βρούμε ακριβώς την επιθυμητή τιμή χρειάζεται να αποθηκεύσουμε τον τόπο των ριζών σε μορφή πίνακα όπως κάναμε στο κεφάλαιο με τον αναλογικό ελεγκτή. Στη συνέχεια θα

αποθηκεύσουμε σε έναν άλλο πίνακα τις τιμές του μηδενιστή για τα ίδια K_i για τα οποία έγινε ο τόπος των ριζών. Τέλος με έναν απλό αλγόριθμο θα δούμε για ποιά τιμή του K_i έχουμε ταύτιση των τιμών των 2 πινάκων. Αυτή η τιμή θα είναι εκείνη για την οποία ο πόλος ταυτίζεται με τον μηδενιστή.

Ακολουθούνε οι εντολές στο MATLAB που εκτελούνε την παραπάνω διαδικασία (αφού γνωρίζουμε ότι για $K_i=4.5$ έχουμε υπερακόντιση θα κάνουμε τον τόπο των ριζών για $0 < K_i < 4.5$ για να δούμε αν η τιμή που αναζητούμε βρίσκεται σε εκείνο το διάστημα).

```
Command Window
>> K=[0:0.1:4.5];
>> R=rlocus(sys,K);
>> zero=-K/0.3;
>> R(1,:)

ans =

Columns 1 through 9
    0   -0.2886   -0.5802   -0.8750   -1.1730   -1.4743   -1.7790   -2.0872   -2.3990

Columns 10 through 18
   -2.7146   -3.0340   -3.3575   -3.6850   -4.0168   -4.3531   -4.6939   -5.0395   -5.3901

Columns 19 through 27
   -5.7457   -6.1067   -6.4733   -6.8456   -7.2240   -7.6087   -8.0000   -8.3983   -8.8037

Columns 28 through 36
   -9.2169   -9.6380  -10.0676  -10.5062  -10.9542  -11.4122  -11.8809  -12.3610  -12.8532

Columns 37 through 45
  -13.3585  -13.8777  -14.4121  -14.9628  -15.5314  -16.1194  -16.7289  -17.3621  -18.0218

Column 46
  -18.7112
```


Στο παραπάνω σχήμα βλέπουμε τον πίνακα R (μόνο την πρώτη του γραμμή που αντιστοιχεί στον δεξί πόλο). Κάθε στοιχείο της στήλης j αυτού του πίνακα είναι ο δεξιός πόλος που αντιστοιχεί στο $K(j)$.

```
>> zero
zero =

Columns 1 through 9
    0   -0.3333   -0.6667   -1.0000   -1.3333   -1.6667   -2.0000   -2.3333   -2.6667

Columns 10 through 18
   -3.0000   -3.3333   -3.6667   -4.0000   -4.3333   -4.6667   -5.0000   -5.3333   -5.6667

Columns 19 through 27
   -6.0000   -6.3333   -6.6667   -7.0000   -7.3333   -7.6667   -8.0000   -8.3333   -8.6667

Columns 28 through 36
   -9.0000   -9.3333   -9.6667  -10.0000  -10.3333  -10.6667  -11.0000  -11.3333  -11.6667

Columns 37 through 45
  -12.0000  -12.3333  -12.6667  -13.0000  -13.3333  -13.6667  -14.0000  -14.3333  -14.6667

Column 46
 -15.0000

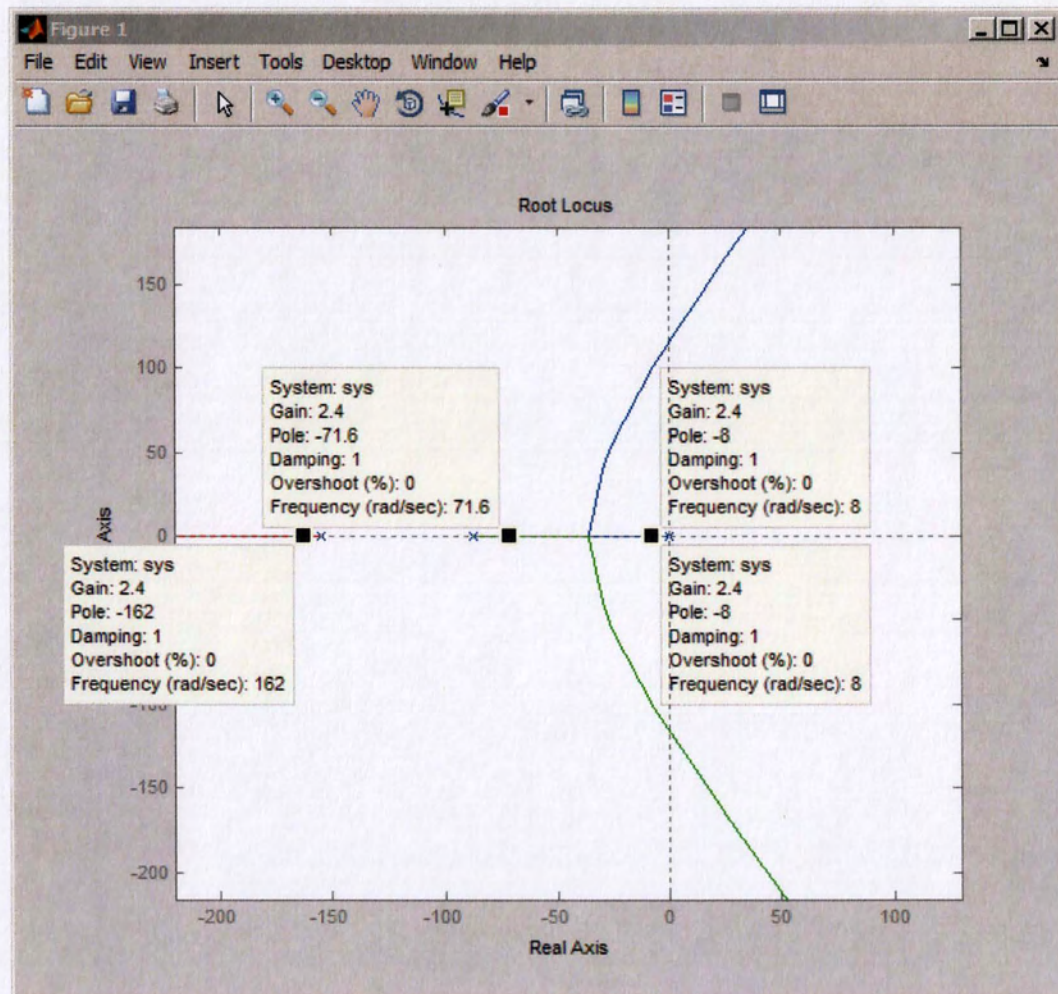
fx >>
```

Σε κάθε στήλη j της μεταβλητής zero υπάρχει η θέση του μηδενιστή για $K(j)$. Άρα πρέπει να συγκρίνουμε τα στοιχεία των πινάκων R και zero και να βρούμε σε ποιά στήλη έχουν την ίδια τιμή. Αυτό γίνεται πολύ εύκολα με τον απλό αλγόριθμο:

```
Editor - C:\Documents and Settings\Owner\My Documents\JSB\script1.m
File Edit Text Go Cell Tools Debug Desktop Window Help
+ - 1.0 + ÷ 1.1 × % + % -
1 - i=2;
2 - while i<47
3 -     if abs(abs(R(1,i))-abs(zero(i)))<0.0001
4 -         i;
5 -         K(i)
6 -         break
7 -     end
8 -     i=i+1;
9 - end
script Ln 9 Col 4 OVR
```


Ο παραπάνω αλγόριθμος μας δίνει για αποτέλεσμα την τιμή του K_i για την οποία έχουμε ταύτιση 2 τιμών, δηλαδή η θέση του μηδενιστή είναι ίδια με αυτή του δεξιού πόλου. Στη συγκεκριμένη περίπτωση αυτό συμβαίνει στη στήλη 25 του K_i η οποία αντιστοιχεί σε $K_i=2.4$

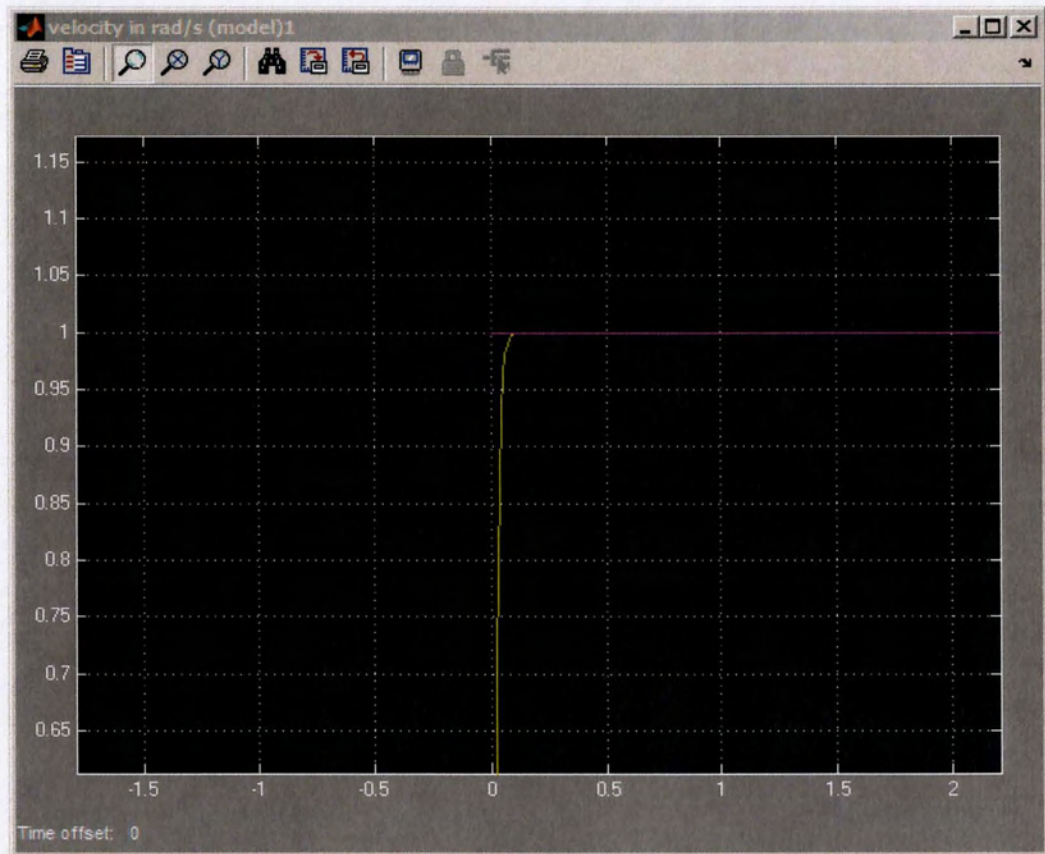
Για να το διαπιστώσουμε αυτό εντοπίζουμε τους πόλους στον τόπο των ριζών για $K_i=2.4$. Επίσης γνωρίζουμε ότι για $K_i=2.4$ ο μηδενιστής βρίσκεται στη θέση $s=-2.4/0.3=-8$



Σχήμα 5-18: Πόλοι και μηδενιστής για $K_i=2.4$ ($K_p=0.3$)

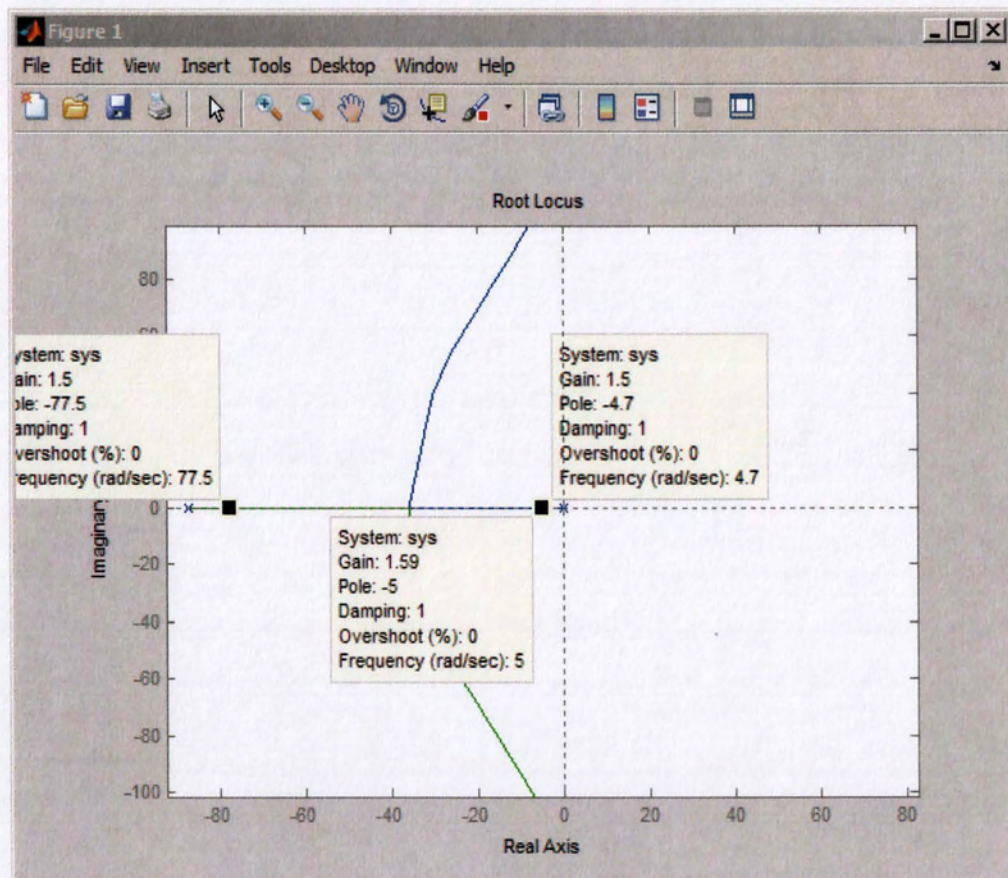
Στο σχήμα 18 παρατηρούμε ότι ο πόλος στα δεξιά έχει όντως συμπέσει με τον μηδενιστή που βρίσκεται στη θέση -8 . Άρα πόλος και μηδενιστής αλληλοαπαλείφονται και το σύστημά μας γίνεται δεύτερης τάξης με 2 πραγματικούς αρνητικούς και άνισους πόλους. Ο ένας στη θέση -162 και ο άλλος στη θέση -71.6 .

Η απόκριση που παίρνουμε για $K_i=2.4$ (και $K_p=0.3$) φαίνεται στο επόμενο σχήμα:



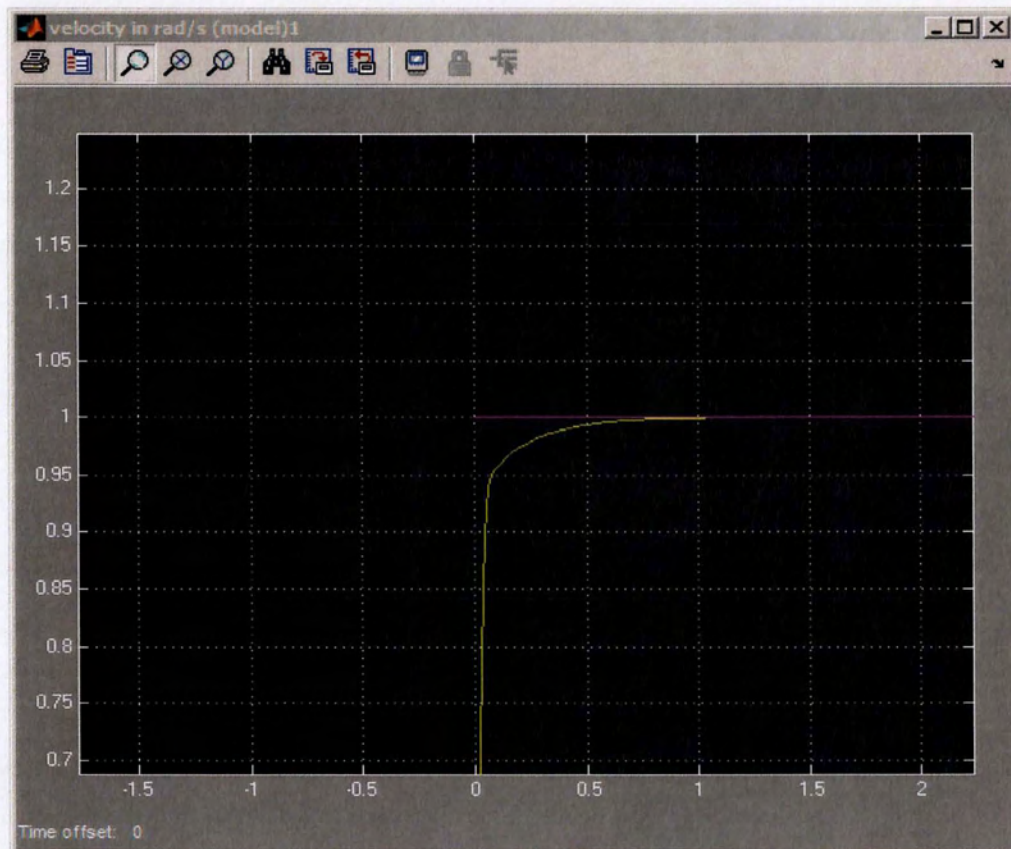
Σχήμα 5-19: Απόκριση για $K_i=2.4$ και $K_p=0.3$

Βλέπουμε ότι δεν υπάρχει καθόλου υπερακόντιση και ότι η έξοδος είναι η τυπική έξοδος που θα περιμέναμε από ένα σύστημα δεύτερης τάξης με 2 πραγματικούς αρνητικούς και άνισους πόλους. Για τιμές του K_i διάφορες από 2.4 ο μηδενιστής δεν ταυτίζεται με τον πόλο και έτσι το σύστημά μας είναι τρίτης τάξης. Τότε ανάλογα με τη θέση του μηδενιστή σε σχέση με τους πόλους και την αρχή των αξόνων έχουμε και διαφορετική απόκριση. Για τιμές του K_i μικρότερες από 2.4 ο μηδενιστής βρίσκεται ανάμεσα στους 2 πόλους. Όσο πιο κοντά στην αρχή (σε σχέση με τον πόλο) βρίσκεται ο μηδενιστής τόσο μεγαλύτερη επίδραση έχει στην ευστάθεια του συστήματος (τείνει να την μειώσει). Άρα όταν βρίσκεται ανάμεσα στους 2 πόλους ο δεξιός πόλος είναι πιο κοντά στην αρχή από τον μηδενιστή. Έτσι το σύστημα συμπεριφέρεται σαν να είχε μεγαλύτερη απόσβεση, όπως φαίνεται στα επόμενα σχήματα:



Σχήμα 5-20: Μηδενιστής ανάμεσα στους 2 πόλους

Στο σχήμα 20 βλέπουμε την περίπτωση για $K_i=1.5$ όπου ο μηδενιστής βρίσκεται στη θέση -5 και ο δεξιός πόλος στη θέση -4.7 (στο σχήμα φαίνεται ότι συμπίπτουν αλλά ο μηδενιστής βρίσκεται στα αριστερά του πόλου). Η αντίστοιχη απόκριση φαίνεται παρακάτω:

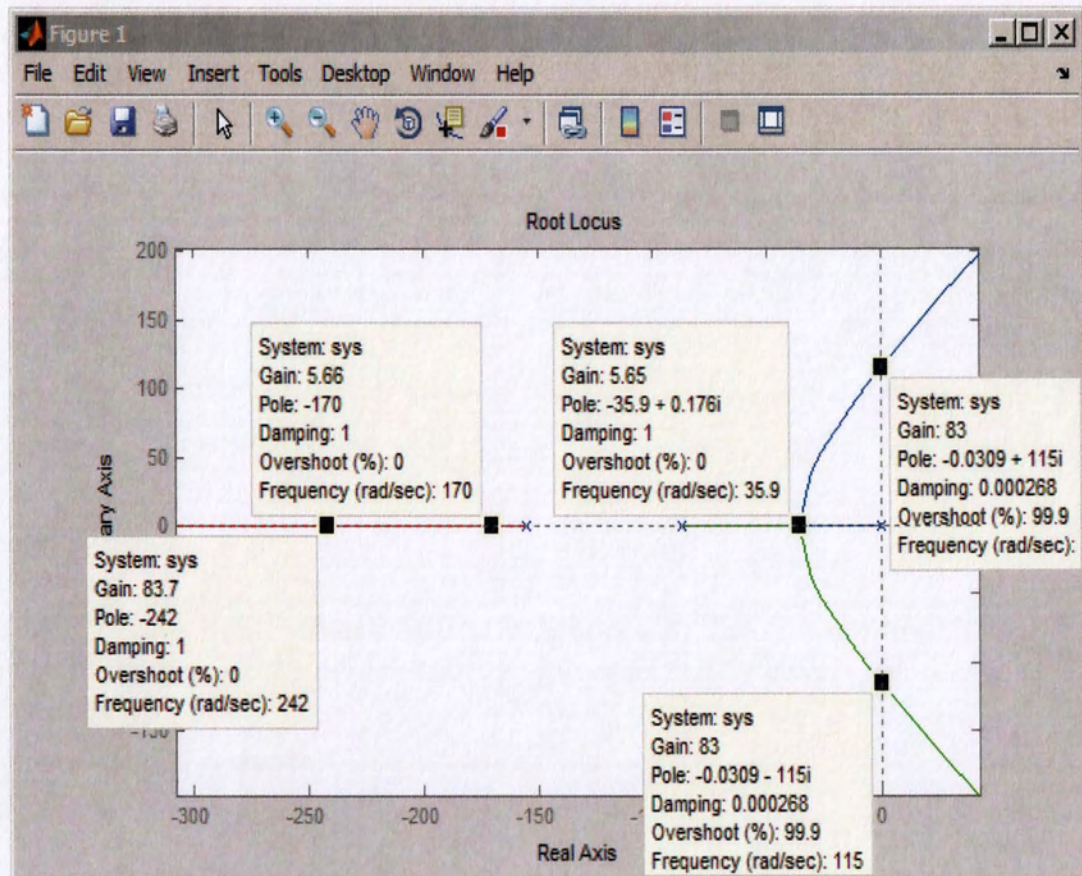


Σχήμα 5-21: Απόκριση για $K_i=1.5$ και $K_p=0.3$

Συγκρίνοντας με την απόκριση του σχήματος 19 γίνεται φανερή η επίδραση της θέσης του μηδενιστή ανάμεσα στους 2 πόλους (το σύστημα τώρα συμπεριφέρεται σαν να είχε μεγαλύτερη απόσβεση).

Για $K_i > 2.4$ ο μηδενιστής βρίσκεται στα δεξιά του πόλου άρα πιο κοντά στην αρχή των αξόνων απ' ότι ο πόλος. Έτσι έχει την τάση να μειώνει την ευστάθεια του συστήματος (προκαλεί υπερακόντιση) και να προκαλεί αποκρίσεις όπως αυτή του σχήματος 15 άσχετα με το γεγονός ότι οι πόλοι έχουν συντελεστή απόσβεσης μεγαλύτερο ή ίσο από τη μονάδα.

Για $K_i > 5.65$ οι πόλοι αρχίζουν να αποκτάνε και φανταστικό μέρος όπως φαίνεται στο επόμενο σχήμα.



Σχήμα 5-22: Χαρακτηριστικές θέσεις πόλων στον τόπο των ριζών

Επομένως για $K_i > 5.65$ η υπερακόντιση δεν οφείλεται πλέον μόνο στον μηδενιστή αλλά και στους πόλους οι οποίοι απέκτησαν και φανταστικό μέρος (ο συντελεστής απόσβεσης είναι πλέον μικρότερος από τη μονάδα).

Τέλος για $K_i > 83$ οι κλάδοι του τόπου των ριζών περνάνε στο δεξί ημιεπίπεδο και το σύστημα πέφτει σε αστάθεια.

Μία ακόμη παράμετρος που πρέπει να εξετάσουμε είναι η ταχύτητα του συστήματος. Μέχρι τώρα εξετάσαμε την επίδραση των δύο μιγαδικών κλάδων του τόπου των ριζών στην απόκριση του συστήματος. Συγκεκριμένα είδαμε ότι η θέση των πόλων πάνω σε αυτούς τους κλάδους σε συνδυασμό με τη θέση του μηδενιστή καθορίζει την ταλαντωτική συμπεριφορά του συστήματος. Όσο πιο ψηλά μετακινούνται οι πόλοι πάνω σε αυτούς τους κλάδους τόσο περισσότερες ταλαντώσεις ή υπερακόντιση έχουμε.

Όμως ο τόπος των ριζών έχει και έναν τρίτο κλάδο που φαίνεται με κόκκινο χρώμα και βρίσκεται ολόκληρος πάνω στον αρνητικό πραγματικό άξονα. Αυτός ο κλάδος σχετίζεται με την ταχύτητα του συστήματος. Η ταχύτητα (συχνότητα) ενός συστήματος είναι η δεύτερη σημαντική παράμετρος που χαρακτηρίζει τη μεταβατική κατάσταση εκτός από τις ταλαντώσεις και την υπερακόντιση.

Όσο πιο μακριά είναι ο πραγματικός πόλος από τον φανταστικό άξονα (δηλαδή όσο μεγαλύτερο είναι το K_i) τόσο πιο γρήγορο είναι το σύστημα. Γι' αυτό όταν μειώσαμε το K_i το σύστημα έγινε πιο αργό (ο πραγματικός πόλος μετακινήθηκε πιο κοντά στον φανταστικό άξονα). Αυτό φαίνεται συγκρίνοντας τα σχήματα 19 και 21.

Συνοψίζοντας, η παρουσία μηδενιστών στη συνάρτηση μεταφοράς κλειστού βρόχου οδηγεί σε ποικίλες αποκρίσεις, των οποίων η μορφή εξαρτάται τόσο από την απόλυτη όσο και από τη σχετική θέση των πόλων και μηδενιστών. Άρα οι παράμετροι ζ και ω δεν αρκούν για τον πλήρη καθορισμό της χρονικής απόκρισης αλλά σε συνδυασμό με τη θέση του μηδενιστή μας δίνουν μια ποιοτική ένδειξη της απόκρισης. Πιο συγκεκριμένα, ο τύπος της μεταβατικής απόκρισης προσδιορίζεται από τους πόλους κλειστού βρόχου, ενώ το σχήμα της μεταβατικής απόκρισης προσδιορίζεται κυρίως από τους μηδενιστές κλειστού βρόχου. Η επίδραση του μηδενιστή κλειστού βρόχου έχει πάντα σχέση και με τη θέση των πόλων αλλά σε γενικές γραμμές ισχύει ότι οδηγεί σε μείωση της ευστάθειας του συστήματος καθώς αυξάνει την συχνότητα και την μέγιστη ποσοστιαία υπερακόντιση. Όσο πλησιέστερα βρίσκεται ο μηδενιστής στην αρχή του επιπέδου s (σε σχέση με τους πόλους), τόσο ισχυρότερη είναι η επίδραση του στην απόκριση του συστήματος.

Άρα τελικά αυτό που πετύχαμε με τον PI ελεγκτή σε σχέση με τον P ελεγκτή είναι:

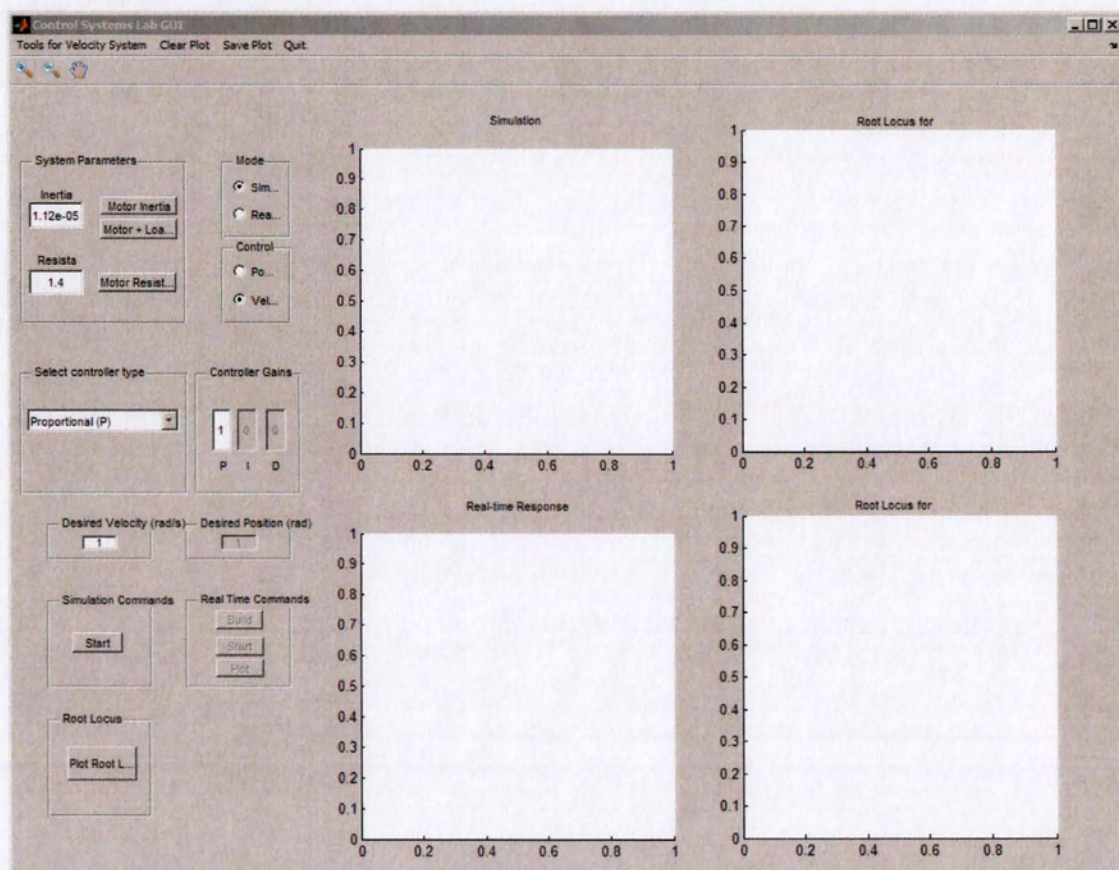
- Με τον P ελεγκτή υπάρχει πεπερασμένο σφάλμα μόνιμης κατάστασης το οποίο εξαρτάται από το K_p . Το ζ και το ω εξαρτώνται επίσης από το K_p . Άρα έχουμε να επιλέξουμε ανάμεσα σε κάποια από αυτές τις 3 παραμέτρους και οι υπόλοιπες 2 καθορίζονται αυτόματα. Δεν έχουμε τη δυνατότητα να μειώσουμε το σφάλμα μόνιμης κατάστασης και τις ταλαντώσεις ταυτόχρονα.

- Με τον ΠΙ ελεγκτή μηδενίζεται το σφάλμα μόνιμης κατάστασης. Έτσι μπορούμε να έχουμε μηδενικό σφάλμα και ταυτόχρονα λίγες ή και καθόλου ταλαντώσεις. Εδώ ο συμβιβασμός που πρέπει να κάνουμε είναι ανάμεσα στις ταλαντώσεις και στην ταχύτητα του συστήματος. Όσο πιο γρήγορο είναι το σύστημα τόσο περισσότερες ταλαντώσεις θα έχει.

Κεφάλαιο 6 : Εκτέλεση εργαστηριακών ασκήσεων

6.1 Εισαγωγή

Για την εκτέλεση των εργαστηριακών ασκήσεων χρησιμοποιείται το πρόγραμμα 'Control Systems Lab GUI'. (GUI, Graphical User Interface). Το γραφικό περιβάλλον του προγράμματος φαίνεται παρακάτω.



Σχήμα 6-1: Γραφικό περιβάλλον του προγράμματος 'Control Systems Lab GUI'

Το πρόγραμμα αυτό γράφτηκε στα πλαίσια της παρούσας εργασίας με σκοπό την εύκολη σχεδίαση, εκτέλεση, και ανάλυση εργαστηριακών ασκήσεων στο μάθημα του Αυτομάτου Ελέγχου.

Είναι γραμμένο αποκλειστικά στο MATLAB και συγκεκριμένα χρησιμοποιώντας το υπό-πρόγραμμα του MATLAB που λέγεται GUIDE. Το GUIDE είναι ενσωματωμένο στο MATLAB και χρησιμοποιείται για την κατασκευή γραφικού περιβάλλοντος χρήστη. Ο κώδικας του προγράμματος αποτελείται από MATLAB script και υπάρχει στο παράρτημα.

6.2 Χρησιμότητα του προγράμματος και περιγραφή των λειτουργιών του

Οι κύριες λειτουργίες του προγράμματος είναι οι εξής:

- Εκτέλεση του μοντέλου που προσομοιώνει το πραγματικό σύστημα και παρουσίαση των αποτελεσμάτων στους άξονες με τίτλο "Simulation".
- Εκτέλεση του μοντέλου που διαχειρίζεται το πραγματικό σύστημα και παρουσίαση των αποτελεσμάτων στους άξονες με τίτλο "Real-time Response".
- Σχεδίαση του τόπου των ριζών για θετικό P και I στους άξονες με τίτλους "Root Locus for positive P" και "Root Locus for positive I" αντίστοιχα.

Οι παραπάνω λειτουργίες αναφέρονται τόσο στον έλεγχο ταχύτητας όσο και θέσης καθώς ο χρήστης είναι ικανός να επιλέξει ανάμεσα στα δύο από το μενού "Control".

Εκτός από αυτές τις κύριες λειτουργίες το πρόγραμμα διαθέτει και μερικές επιμέρους λειτουργίες όπως τη δυνατότητα αποθήκευσης των αποτελεσμάτων για μετέπειτα επεξεργασία (Save Plot), καθώς και χρήσιμα εργαλεία για τον έλεγχο της ταχύτητας (Tools for Velocity System).

Τέλος, ο χρήστης έχει τη δυνατότητα να επιλέξει το είδος και τα κέρδη του ελεγκτή, τις παραμέτρους του μοντέλου προσομοίωσης, καθώς επίσης και το εάν επιθυμεί να γίνει προσομοίωση ή να "τρέξει" το πραγματικό σύστημα.

Η χρησιμότητα του προγράμματος έγκειται κυρίως στο ότι καθιστά μη-απαραίτητη τη γνώση περί MATLAB και Simulink. Αυτός είναι και ο βασικός λόγος δημιουργίας του. Κατά την μελέτη και

ανάλυση ενός πραγματικού συστήματος ελέγχου απαιτούνται γνώσεις πάνω σε αυτά τα προγράμματα οι οποίες πολλές φορές δεν υπάρχουν και έτσι είναι δύσκολο να γίνουν κατανοητές οι εργαστηριακές ασκήσεις ακόμα και εάν ο χρήστης έχει τις γνώσεις που απαιτούνται από την πλευρά του Αυτομάτου Ελέγχου. Έτσι το πρόγραμμα αυτό δημιουργήθηκε με σκοπό να περιορίσει στο ελάχιστο τις απαιτούμενες γνώσεις σε MATLAB και Simulink και να επιτρέψει στο χρήστη να χρησιμοποιήσει τις γνώσεις του περί Αυτομάτου Ελέγχου.

Ένας ακόμη λόγος που οδήγησε στη δημιουργία αυτού του προγράμματος ήταν η ανάγκη για τη συγκέντρωση όλων των δεδομένων και γραφικών παραστάσεων σε ένα μέρος, από όπου θα δίνονται οι εντολές και οι ρυθμίσεις και θα λαμβάνονται τα αποτελέσματα. Έτσι δεν είναι απαραίτητο να έχουμε ξεχωριστά προγράμματα για την κάθε λειτουργία, πράγμα που κάνει πολύ πιο εύκολη τη διεξαγωγή των εργαστηριακών ασκήσεων.

Ακολουθεί αναλυτική περιγραφή του κάθε στοιχείου του προγράμματος.

6.3 Αναλυτική περιγραφή προγράμματος

Πλαίσιο "Mode"

Η πιο βασική επιλογή που πρέπει να κάνουμε στην αρχή είναι το εάν θέλουμε να εκτελέσουμε μια προσομοίωση ή το πραγματικό σύστημα. Αυτό επιλέγεται από το πλαίσιο με τίτλο Mode. Η προσομοίωση μας δίνει τη δυνατότητα να προβλέψουμε τη συμπεριφορά του συστήματος και να χρησιμοποιήσουμε εργαλεία όπως ο τόπος των ριζών προκειμένου να σχεδιάσουμε έναν ελεγκτή που να ικανοποιεί τις απαιτήσεις μας. Όταν η απόκριση που λαμβάνουμε από την προσομοίωση γίνει αποδεκτή τότε μπορούμε να εφαρμόσουμε τον ίδιο ελεγκτή και στο πραγματικό σύστημα. Εδώ πρέπει να υπενθυμίσουμε ότι η πραγματική απόκριση δεν θα είναι ποτέ η ίδια με την προβλεπόμενη από το μοντέλο. Αυτό γιατί, (1) το ίδιο το μοντέλο δεν περιγράφει ακριβώς το πραγματικό σύστημα (διότι υπάρχουν μη-γραμμικά φαινόμενα όπως η στατική τριβή τα οποία δεν μοντελοποιούνται) και (2) οι μετρήσεις δεν είναι τέλειες και περιέχουν θόρυβο.

Πλαίσιο "Control"

Η δεύτερη πιο βασική επιλογή που πρέπει να κάνουμε είναι το φυσικό μέγεθος στο οποίο θα γίνει ο έλεγχος. Αυτή η επιλογή γίνεται από το πλαίσιο με τίτλο Control και μπορούμε να επιλέξουμε ταχύτητα ή θέση.

Πλαίσιο “System Parameters”

Αυτό το πλαίσιο ενεργοποιείται μόνο στην περίπτωση που εκτελούμε προσομοίωση και όχι το πραγματικό σύστημα. Μπορούμε να δώσουμε διάφορες τιμές στην αδράνεια του άξονα και στην ηλεκτρική αντίσταση του κινητήρα και να δούμε πως μεταβάλλονται οι αποκρίσεις. Οι πραγματικές τιμές αυτών των μεγεθών επανέρχονται με το πάτημα των αντίστοιχων κουμπιών.

Πλαίσια “Select controller type” και “Controller Gains”

Τα πλαίσια αυτά χρησιμοποιούνται για την επιλογή του είδους του ελεγκτή (P,PI,PID) και για τον καθορισμό των κερδών του αντίστοιχα.

Πλαίσια “Desired Velocity (rad/s)” και “Desired Position (rad)”

Χρησιμοποιούνται για τον προσδιορισμό της επιθυμητής ταχύτητας και θέσης αντίστοιχα (στις κατάλληλες μονάδες).

Πλαίσιο “Simulation Commands”

Δίνει την εντολή για την έναρξη της προσομοίωσης. Με το πάτημα του κουμπιού Start το πρόγραμμα συλλέγει όλα τα δεδομένα που του δόθηκαν παραπάνω και τα χρησιμοποιεί σαν βάση δεδομένων για την εκτέλεση του μοντέλου στο Simulink. Μετά την εκτέλεση του μοντέλου τυπώνονται τα αποτελέσματα στους άξονες με τίτλο Simulation Response. Το Simulink εκτελείται χωρίς να γίνεται ορατό από τον χρήστη και κλείνει αυτόματα μετά το τέλος της προσομοίωσης. Γι’ αυτό το λόγο δεν απαιτείται καμία αλληλεπίδραση του χρήστη με αυτό.

Πλαίσιο “Real Time Commands”

Σε αυτό το πλαίσιο βρίσκονται οι εντολές που αφορούν στην εκτέλεση του πραγματικού συστήματος και είναι τοποθετημένες στη σειρά με την οποία πρέπει να εκτελεστούνε. Πρώτη είναι η

εντολή Build. Αυτή η εντολή είναι απαραίτητη σε όλα τα μοντέλα τα οποία έχουν επικοινωνία σε πραγματικό χρόνο με hardware και εκτελείται πάντα πρώτη. Αυτό γιατί παράγει τον εκτελέσιμο C κώδικα που είναι απαραίτητος για την σε-ζωντανό-χρόνο επικοινωνία. Η διαδικασία αυτή διαρκεί για αρκετά δευτερόλεπτα και ο χρήστης ενημερώνεται όταν ολοκληρωθεί.

Επόμενη είναι η εντολή Start. Η εντολή αυτή εκτελεί τον κώδικα που παράγεται με την εντολή Build και έτσι έχουμε εκτέλεση του πραγματικού συστήματος σε πραγματικό χρόνο. Όταν ολοκληρωθεί ο χρόνος εκτέλεσης του πειράματος (10 δευτερόλεπτα), τα αποτελέσματα αποθηκεύονται και είναι έτοιμα για γραφική απεικόνιση με την εντολή Plot.

Πλαίσιο ‘‘Root Locus’’

Η εντολή Plot Root Locus σχεδιάζει τον τόπο των ριζών σε έναν από τους δεξιούς άξονες ανάλογα με το τι ελεγκτή έχουμε επιλέξει. Εάν έχουμε επιλέξει αναλογικό ελεγκτή τότε σχεδιάζεται στον πάνω άξονα για θετικό P. Εάν έχουμε επιλέξει αναλογικό-ολοκληρωτικό ελεγκτή τότε σχεδιάζεται στον κάτω άξονα για θετικό I και σταθερό P (όσο το έχουμε ορίσει στο πλαίσιο Controller Gains). Στην περίπτωση του αναλογικού-ολοκληρωτικού-διαφορικού ελεγκτή δεν είναι διαθέσιμος ο τόπος των ριζών. Ο τόπος των ριζών είναι διαθέσιμος τόσο για τον έλεγχο θέσης όσο και ταχύτητας. Για να σχεδιαστεί, χρησιμοποιούνται εντολές από το toolbox του MATLAB με τίτλο ‘Control Systems Toolbox’ όπως η sys=tf και η rlocus.

Ακολουθεί περιγραφή του μενού επιλογών.

Tools for Velocity System

Περιέχει 2 εργαλεία που σχετίζονται με τον έλεγχο της ταχύτητας. Το πρώτο υπολογίζει την τιμή που πρέπει να έχει το κέρδος P (όταν κάνουμε αναλογικό έλεγχο) έτσι ώστε να πετύχουμε δεδομένο σφάλμα μόνιμης κατάστασης. Το δεύτερο αναφέρεται στην περίπτωση που έχουμε αναλογικό-ολοκληρωτικό ελεγκτή και υπολογίζει τη θέση του μηδενιστή κλειστού βρόχου στο μιγαδικό επίπεδο. Η θέση του μηδενιστή κλειστού βρόχου παίζει πολύ σημαντικό ρόλο στη μορφή της απόκρισης και η επίδρασή του έχει εξηγηθεί στο κεφάλαιο 5.3 αναλυτικά.

Clear Plot

Διαγράφει οποιαδήποτε γραφική παράσταση έχει σχεδιαστεί πάνω σε κάποιον συγκεκριμένο άξονα ή και σε όλους μαζί.

Save Plot

Αποθηκεύει τη γραφική παράσταση του επιλεγμένου άξονα σε ένα εξωτερικό αρχείο με κατάληξη .fig. Το όνομα του αρχείου και τον φάκελο αποθήκευσης τα ορίζει ο χρήστης, και τα αρχεία με κατάληξη .fig ανοίγονται από το MATLAB.

Quit

Τερματίζει το πρόγραμμα.

6.4 Πληροφορίες για την εκτέλεση των πειραμάτων

Τα πειράματα έχουν ως σκοπό να δείξουν πως εφαρμόζονται στην πράξη οι αρχές του Αυτομάτου Ελέγχου μέσα από ένα πραγματικό σύστημα ελέγχου. Η συσκευή πάνω στην οποία εφαρμόζεται ο έλεγχος είναι ένας κινητήρας συνεχούς ρεύματος της εταιρίας Maxonmotors, του οποίου προσπαθούμε να ελέγξουμε τη θέση και την ταχύτητα. Για να πετύχουμε τον κλειστό βρόχο στην πράξη πρέπει να μπορούμε να ανταλλάσουμε σήματα με τον κινητήρα. Το σήμα που λαμβάνουμε από τον κινητήρα είναι η γωνία στροφής την οποία μας δίνει ο αισθητήρας θέσης (encoder) που είναι τοποθετημένος στο πίσω μέρος του. Το σήμα που στέλνουμε στον κινητήρα είναι η τάση τροφοδοσίας του μέσω κατάλληλου ενισχυτή. Και τα δυο σήματα επικοινωνούν με Η/Υ μέσω μιας κάρτας εισόδου-εξόδου της National Instruments. Από πλευράς software χρησιμοποιείται το MATLAB, το Simulink, και συγκεκριμένα το πακέτο του Simulink 'Real-Time Windows Target'. Πιο αναλυτική περιγραφή του συστήματος (hardware και software) υπάρχει στο Κεφάλαιο 2.

Χρησιμοποιώντας το πρόγραμμα 'Control Systems Lab GUI' που περιγράφηκε παραπάνω, πετυχαίνουμε τον έλεγχο ταχύτητας και θέσης του κινητήρα. Κάθε πείραμα έχει διαφορετικές απαιτήσεις οι οποίες πρέπει να ικανοποιηθούν. Για παράδειγμα, σε ένα πείραμα μπορεί να μην μας

ενδιαφέρει εάν υπάρχει σφάλμα μόνιμης κατάστασης όσο αυτό βρίσκεται ανάμεσα σε κάποια επιτρεπτά όρια, αρκεί η απόκριση να μην έχει μεγάλη υπερακόντιση. Σε ένα άλλο πείραμα μπορεί να μην μας ενδιαφέρει τόσο η υπερακόντιση όσο το σφάλμα μόνιμης κατάστασης. Γενικά τις απαιτήσεις τις ορίζει ο χρήστης και συνήθως είναι ένας συμβιβασμός ανάμεσα σε χαρακτηριστικά μεγέθη της απόκρισης όπως σφάλμα μόνιμης κατάστασης, ταλαντώσεις στη μεταβατική κατάσταση, υπερακόντιση, χρόνος ανύψωσης και χρόνος αποκατάστασης. Διαφορετικοί ελεγκτές με διαφορετικά κέρδη οδηγούν σε διαφορετικές αποκρίσεις, και πρέπει να βρούμε τον κατάλληλο ελεγκτή με τα κατάλληλα κέρδη έτσι ώστε να πετύχουμε την επιθυμητή απόκριση. Τα εργαλεία τα οποία μας βοηθάνε στην επιλογή του κατάλληλου ελεγκτή είναι ο τύπος των ριζών και τα δυο προγράμματα στο μενού 'Tools for Velocity System' (προφανώς μόνο στην περίπτωση που κάνουμε έλεγχο ταχύτητας). Αφού καταλήξουμε σε έναν συγκεκριμένο ελεγκτή πρώτα τον δοκιμάζουμε στο μοντέλο της προσομοίωσης. Εάν η απόκριση της προσομοίωσης δεν είναι η επιθυμητή τότε χρησιμοποιούμε πάλι τα παραπάνω εργαλεία για να επαναυπολογίσουμε τα κέρδη του ελεγκτή. Όταν η απόκριση γίνει αποδεκτή τότε εφαρμόζουμε τα ίδια κέρδη του ελεγκτή και στο πραγματικό σύστημα.

Λόγω της φύσης των πειραμάτων είναι δύσκολο να δοθεί παράδειγμα εφαρμογής σε αυτό το σημείο. Υπάρχει όμως αναλυτικό παράδειγμα στην παρουσίαση της εργασίας στο συνοδευτικό cd.

Κεφάλαιο 7 : Σύνοψη και μελλοντικές εργασίες

7.1 Σύνοψη

Σε αυτή την εργασία μελετήσαμε τον έλεγχο θέσης και ταχύτητας ενός κινητήρα συνεχούς ρεύματος. Είδαμε πως γίνεται η επικοινωνία του κινητήρα με τον υπολογιστή σε πραγματικό χρόνο, καθώς και τι λογισμικό χρησιμοποιήθηκε για να το πετύχουμε αυτό. Έπειτα μελετήθηκε η μοντελοποίηση του συστήματος και η βοήθεια που μας παρέχει το μοντέλο. Στη συνέχεια έγινε έλεγχος της ταχύτητας στο μοντέλο εφαρμόζοντας τεχνικές όπως ο τρόπος των ριζών, κριτήριο ευστάθειας Routh-Hurwitz, και γενικές γνώσεις από τη θεωρία Αυτομάτου Ελέγχου. Αναλύθηκε διεξοδικά η επίδραση των ελεγκτών στην απόκριση με σκοπό την καλύτερη κατανόηση του συστήματος κλειστού βρόχου. Κεντρική ιδέα της εργασίας υπήρξε η συστηματοποίηση της διαδικασίας έτσι ώστε να μπορεί να αποτελέσει εργαστηριακή άσκηση στο μάθημα του Αυτομάτου Ελέγχου. Για το σκοπό αυτό δημιουργήθηκε το πρόγραμμα 'Control Systems Lab GUI' το οποίο παρουσιάστηκε στο τελευταίο κεφάλαιο και μας επιτρέπει την εύκολη εκτέλεση των πειραμάτων και τη συλλογή των δεδομένων σε ένα μόνο μέρος.

7.2 Προτάσεις για μελλοντικές εργασίες

Μερικά σχετικά θέματα τα οποία θα μπορούσαν να αποτελέσουν (μεμονωμένα ή συνδυασμός τους) θέμα περαιτέρω εργασίας είναι:

- Μη-γραμμικός έλεγχος που συμπεριλαμβάνει τη μοντελοποίηση μη-γραμμικών φαινομένων όπως στατική τριβή.
- Βελτίωση των αποτελεσμάτων χρησιμοποιώντας αισθητήρα ταχύτητας αντί για αισθητήρα θέσης.
- Χρήση βελτιωμένου λογισμικού όπως π.χ. xPC Target.

Βιβλιογραφία

- [1] Κρικέλης Ν., «ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΑΥΤΟΜΑΤΟ ΕΛΕΓΧΟ», 3η Έκδοση, Εκδόσεις Συμμετρία, Αθήνα 2000.
- [2] Katsuhiko Ogata, «MODERN CONTROL ENGINEERING», Prentice Hall International, Series in Systems and Control Engineering, third edition, USA 1997.
- [3] Stephen J.Chapman, «Ηλεκτρικές Μηχανές AC-DC», Μετάφραση Θ.Π. Θεοδουλίδης, 3^η έκδοση, Εκδόσεις Τζιόλα, Θεσσαλονίκη 2003.
- [4] Adrian Biran & Moshe Breiner, «MATLAB 6 Για μηχανικούς», Μετάφραση Ιωάννης Πεταλάς, Εκδόσεις Τζιόλα , Θεσσαλονίκη 2003
- [5] www.mathworks.com
- [6] www.ni.com
- [7] www.maxonmotor.com
- [8] www.blinkdagger.com

Παράρτημα

Παρακάτω υπάρχει ο κώδικας MATLAB του προγράμματος 'Control Systems Lab GUI'. Τα μοντέλα που χρησιμοποιήθηκαν στο Simulink, καθώς και διάφορα σχετικά αρχεία (εγχειρίδιο κινητήρα και κάρτας εισόδου-εξόδου) υπάρχουν στο συνοδευτικό cd.

```
-----

% Copyright (C) Kostas Argiris 2011

function varargout = testGUI(varargin)

% Last Modified by GUIDE v2.5 08-Sep-2011 19:20:02

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',   @testGUI_OpeningFcn, ...
                  'gui_OutputFcn',    @testGUI_OutputFcn, ...
                  'gui_LayoutFcn',    [], ...
                  'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% --- Executes just before testGUI is made visible.
function testGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to testGUI (see VARARGIN)

% Choose default command line output for testGUI
handles.output = hObject;
set(handles.figure1, 'CloseRequestFcn', @closeGUI);
set(handles.popupmenu1, 'Value', 1);
set(handles.edit5, 'Enable', 'off');
set(handles.edit6, 'Enable', 'off');
set(handles.figure1, 'Position', [20.2000 6.6923 315.8000 65.7692])

% Update handles structure
guidata(hObject, handles);
```



```

% UIWAIT makes testGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = testGUI_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit1_Callback(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
% str2double(get(hObject,'String')) returns contents of edit1 as a
double

if isempty(str2num(get(hObject,'String')))
    msgbox('Input must be a number.','Warning !')
    set(hObject,'String','1.12e-05')
end

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
% str2double(get(hObject,'String')) returns contents of edit2 as a
double

if isempty(str2num(get(hObject,'String')))

```

```

        msgbox('Input must be a number.','Warning !')
        set(hObject,'String','1.4')
    end

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%       str2double(get(hObject,'String')) returns contents of edit4 as a
double

if isempty(str2num(get(hObject,'String')))
    msgbox('Input must be a number.','Warning !')
    set(hObject,'String','1')
end

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%       str2double(get(hObject,'String')) returns contents of edit5 as a
double

```



```

if isempty(str2num(get(hObject,'String')))
    msgbox('Input must be a number.','Warning !')
    set(hObject,'String','0')
end

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%         str2double(get(hObject,'String')) returns contents of edit6 as a
double

if isempty(str2num(get(hObject,'String')))
    msgbox('Input must be a number.','Warning !')
    set(hObject,'String','0')
end

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text

```

```

%         str2double(get(hObject,'String')) returns contents of edit7 as a
double

if isempty(str2num(get(hObject,'String')))
    msgbox('Input must be a number.','Warning !')
    set(hObject,'String','1')
end

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function closeGUI(src,evnt)
%this function is called when the user attempts to close the GUI window

%this command brings up the close window dialog
selection = questdlg('Do you want to close the application?',...
    'Close Confirmation',...
    'Yes','No','Yes');

%if user clicks yes, the GUI window is closed, otherwise
%nothing happens
switch selection,
    case 'Yes',
        delete(gcf)
    case 'No'
        return
end

% --- Executes on button press in build.
function build_Callback(hObject, eventdata, handles)
% hObject    handle to build (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.plot,'Enable','off')
P=str2num(get(handles.edit4,'String')); %#ok<*ST2NM>
I=str2num(get(handles.edit5,'String'));
D=str2num(get(handles.edit6,'String'));
handles.P=num2str(P);
handles.I=num2str(I);
assignin('base','P',P);
assignin('base','I',I);
assignin('base','D',D);
a=get(get(handles.uipanel7,'SelectedObject'),'Tag');
if isempty(a)

```



```

msgbox('You have not selected Position or Velocity control.')
a=1;
end
switch a
case 'control_vel'
    desiredvelocity=str2num(get(handles.edit7,'String'));
    assignin('base','desiredvelocity',desiredvelocity);
    % Change mouse cursor to hourglass
    set(handles.figure1,'Pointer','watch');
    % Waitbar appears
    wStr = sprintf('%s\n%s',...
        'Please wait while the model builds. ');
    hw = waitbar(0.5,wStr);
    rtwbuild('VelocityControl_real');
    % Waitbar disappears
    delete(hw);
    % Change mouse cursor to arrow
    set(handles.figure1,'Pointer','arrow');
    set(handles.edit4,'Enable','off')
    set(handles.edit5,'Enable','off')
    set(handles.edit6,'Enable','off')
    set(handles.edit7,'Enable','off')
    set(handles.edit8,'Enable','off')
    set(handles.radiobutton1,'Enable','off')
    set(handles.radiobutton4,'Enable','off')
    set(handles.control_pos,'Enable','off')
    set(handles.control_vel,'Enable','off')
    msgbox('Build is complete , you can press Start now to begin.')
    set(handles.start1,'Enable','on')
case 'control_pos'
    desiredposition=str2num(get(handles.edit8,'String'));
    assignin('base','desiredposition',desiredposition);
    % Change mouse cursor to hourglass
    set(handles.figure1,'Pointer','watch');
    % Waitbar appears
    wStr = sprintf('%s\n%s',...
        'Please wait while the model builds. ');
    hw = waitbar(0.5,wStr);
    rtwbuild('PositionControl_real');
    % Waitbar disappears
    delete(hw);
    % Change mouse cursor to arrow
    set(handles.figure1,'Pointer','arrow');
    set(handles.edit4,'Enable','off')
    set(handles.edit5,'Enable','off')
    set(handles.edit6,'Enable','off')
    set(handles.edit7,'Enable','off')
    set(handles.edit8,'Enable','off')
    set(handles.radiobutton1,'Enable','off')
    set(handles.radiobutton4,'Enable','off')
    set(handles.control_pos,'Enable','off')
    set(handles.control_vel,'Enable','off')
    msgbox('Build is complete , you can press Start now to begin.')
    set(handles.start1,'Enable','on')
end
guidata(hObject, handles);

```

```

% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject      handle to start (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
kt=0.0605;
b=5.29e-06;
axes(handles.axes1)
J=str2num(get(handles.edit1,'String'));
R=str2num(get(handles.edit2,'String'));
P=str2num(get(handles.edit4,'String'));
I=str2num(get(handles.edit5,'String'));
D=str2num(get(handles.edit6,'String'));
a=get(get(handles.uipanel7,'SelectedObject'),'Tag');
if isempty(a)
    msgbox('You have not selected Position or Velocity control.')
    a=1;
end
switch a
    case 'control_pos'
        % Change mouse cursor to hourglass
        set(handles.figure1,'Pointer','watch');
        % Disable enabled buttons
        H=findobj('Enable','on');
        set(H,'Enable','off')
        % Waitbar appears
        wStr = sprintf('%s\n%s',...
            'Loading...');
        hw = waitbar(0.5,wStr);
        desiredposition=str2num(get(handles.edit8,'String'));
        options=simset('SrcWorkspace','current');
        sim('PositionControl_model',[],options);
        axes(handles.axes1)
        plot(tout,yout)
        P1=num2str(P);
        I1=num2str(I);
        title(['for P=' P1 ' and I=' I1],'FontSize',8)
        xlabel('Time (s)','FontSize',8)
        ylabel('Position (rad)','FontSize',8)
        grid on
        % Add a legend
        handles.legend_axes1_pos=legend('Position in rads','Desired
position');
        % Waitbar disappears
        delete(hw);
        % Change mouse cursor to arrow
        set(handles.figure1,'Pointer','arrow');
        % Enable disabled buttons
        set(H,'Enable','on')
    case 'control_vel'
        % Change mouse cursor to hourglass
        set(handles.figure1,'Pointer','watch');
        % Disable enabled buttons
        H=findobj('Enable','on');
        set(H,'Enable','off')
        % Waitbar appears
        wStr = sprintf('%s\n%s',...
            'Loading...');
        hw = waitbar(0.5,wStr);

```



```

desiredvelocity=str2num(get(handles.edit7,'String'));
options=simset('SrcWorkspace','current');
sim('VelocityControl_model',[],options);
axes(handles.axes1)
plot(tout,yout)
P1=num2str(P);
I1=num2str(I);
title(['for P=' P1 ' and I=' I1'],'FontSize',8)
xlabel('Time (s)','FontSize',8)
ylabel('Velocity (rad/s)','FontSize',8)
grid on
% Add a legend
handles.legend_axes1_vel=legend('Velocity in rad/s','Desired
velocity');
% Waitbar disappears
delete(hw);
% Change mouse cursor to arrow
set(handles.figure1,'Pointer','arrow');
% Enable disabled buttons
set(H,'Enable','on')
end
guidata(hObject, handles);

% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2

% --- Executes when selected object is changed in uipanel4.
function uipanel4_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in uipanel4
% eventdata  structure with the following fields (see UIBUTTONGROUP)
%   EventName: string 'SelectionChanged' (read only)
%   OldValue: handle of the previously selected object or empty if none was
selected
%   NewValue: handle of the currently selected object
% handles    structure with handles and user data (see GUIDATA)
switch get(eventdata.NewValue,'String')
case 'Simulation'
    set(handles.build,'Enable','off')
    set(handles.start,'Enable','on')
    set(handles.edit1,'Enable','on')
    set(handles.edit2,'Enable','on')
    set(handles.start1,'Enable','off')
    set(handles.plot,'Enable','off')

case 'Real Time'
    set(handles.build,'Enable','on')
    set(handles.start,'Enable','off')
    set(handles.edit1,'Enable','off')
    set(handles.edit2,'Enable','off')
    set(handles.start1,'Enable','off')
    set(handles.plot,'Enable','off')
end

```

```

% --- Executes on button press in radiobutton3.
function radiobutton3_Callback(hObject, eventdata, handles)
% hObject      handle to radiobutton3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton3

% --- Executes on button press in start1.
function start1_Callback(hObject, eventdata, handles)
% hObject      handle to start1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
a=get(get(handles.uipanel7,'SelectedObject'),'Tag');
switch a
    case 'control_vel'
        desiredvelocity=str2num(get(handles.edit7,'String'));
        assignin('base','desiredvelocity',desiredvelocity);
        load_system('VelocityControl_real')
        set_param('VelocityControl_real','SimulationCommand','connect')
        set_param('VelocityControl_real','SimulationCommand','start')
        msgbox('Motor is rotating.As soon as you hear the beep sound you
can press Plot to view the results.')
        set(handles.plot,'Enable','on')
    case 'control_pos'
        desiredposition=str2num(get(handles.edit8,'String'));
        assignin('base','desiredposition',desiredposition);
        load_system('PositionControl_real')
        set_param('PositionControl_real','SimulationCommand','connect')
        set_param('PositionControl_real','SimulationCommand','start')
        msgbox('As soon as you hear the beep sound press Plot to view the
results.')
        set(handles.plot,'Enable','on')
end
guidata(hObject, handles);

% --- Executes on button press in plot.
function plot_Callback(hObject, eventdata, handles) %#ok<*DEFNU>
% hObject      handle to plot (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
axes(handles.axes2)
a=get(get(handles.uipanel7,'SelectedObject'),'Tag');
switch a
    case 'control_vel'
        velout=evalin('base','velout');
        axes(handles.axes2)
        plot(velout.time,velout.signals.values)
        xlabel('Time (s)','FontSize',8)
        ylabel('Velocity (rad/s)','FontSize',8)
        title(['for P=' handles.P ' and I=' handles.I],'FontSize',8)
        grid on
        % Add a legend

```



```

handles.legend_axes2_vel=legend('Velocity in rad/s','Desired
velocity');
set(handles.edit4,'Enable','on')
set(handles.edit5,'Enable','on')
set(handles.edit6,'Enable','on')
set(handles.edit7,'Enable','on')
set(handles.edit8,'Enable','off')
set(handles.radiobutton1,'Enable','on')
set(handles.radiobutton4,'Enable','on')
set(handles.control_pos,'Enable','on')
set(handles.control_vel,'Enable','on')
case 'control_pos'
posout=evalin('base','posout');
axes(handles.axes2)
plot(posout.time,posout.signals.values)
xlabel('Time (s)','FontSize',8)
ylabel('Position (rad)','FontSize',8)
title(['for P=' handles.P ' and I=' handles.I],'FontSize',8)
grid on
% Add a legend
handles.legend_axes2_pos=legend('Position in rads','Desired
position');
set(handles.edit4,'Enable','on')
set(handles.edit5,'Enable','on')
set(handles.edit6,'Enable','on')
set(handles.edit7,'Enable','off')
set(handles.edit8,'Enable','on')
set(handles.radiobutton1,'Enable','on')
set(handles.radiobutton4,'Enable','on')
set(handles.control_pos,'Enable','on')
set(handles.control_vel,'Enable','on')
end
guidata(hObject, handles);

% --- Executes on button press in motor_inertia.
function motor_inertia_Callback(hObject, eventdata, handles)
% hObject    handle to motor_inertia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit1,'String','1.12e-05')

% --- Executes on button press in motorandload_inertia.
function motorandload_inertia_Callback(hObject, eventdata, handles)
% hObject    handle to motorandload_inertia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit1,'String','7.033e-05')

% --- Executes on button press in motor_resistance.
function motor_resistance_Callback(hObject, eventdata, handles)
% hObject    handle to motor_resistance (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit2,'String','1.4')

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double

if isempty(str2num(get(hObject,'String')))
    msgbox('Input must be a number.','Warning !')
    set(hObject,'String','1')
end

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit8 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes when selected object is changed in uipanel7.
function uipanel7_SelectionChangeFcn(hObject, eventdata, handles)
% hObject      handle to the selected object in uipanel7
% eventdata    structure with the following fields (see UIBUTTONGROUP)
%      EventName: string 'SelectionChanged' (read only)
%      OldValue: handle of the previously selected object or empty if none was
selected
%      NewValue: handle of the currently selected object
% handles      structure with handles and user data (see GUIDATA)
switch get(eventdata.NewValue,'String')
    case 'Position'
        set(handles.edit7,'Enable','off')
        set(handles.edit8,'Enable','on')
    case 'Velocity'
        set(handles.edit7,'Enable','on')
        set(handles.edit8,'Enable','off')
end

% --- Executes during object creation, after setting all properties.
function uipanel7_CreateFcn(hObject, eventdata, handles)
% hObject      handle to uipanel7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% --- Executes during object deletion, before destroying properties.
function uipanel7_DeleteFcn(hObject, eventdata, handles)
% hObject      handle to uipanel7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

```



```

% -----
function uipanel7_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to uipanel7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
%         contents as cell array
%         contents{get(hObject,'Value')} returns selected item from
%         popupmenu1
switch get(hObject,'Value')
    case 1
        set(handles.edit4,'Enable','on')
        set(handles.edit5,'Enable','off')
        set(handles.edit5,'String','0')
        set(handles.edit6,'Enable','off')
        set(handles.edit6,'String','0')
    case 2
        set(handles.edit4,'Enable','on')
        set(handles.edit5,'Enable','on')
        set(handles.edit6,'Enable','off')
        set(handles.edit6,'String','0')
    case 3
        set(handles.edit4,'Enable','on')
        set(handles.edit5,'Enable','on')
        set(handles.edit6,'Enable','on')
    otherwise
end
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in rlocus_pushbutton.
function rlocus_pushbutton_Callback(hObject, eventdata, handles)
% hObject    handle to rlocus_pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

switch get(get(handles.uipanel7,'SelectedObject'),'Tag')
case 'control_vel'
    switch get(handles.popupmenu1,'Value')
    case 1
        % Waitbar appears
        wStr = sprintf('%s\n%s',...
            'Loading...');
        hw = waitbar(0.5,wStr);
        set(handles.figure1,'Pointer','watch')
        H=findobj('Style','PushButton','Enable','on');
        set(H,'Enable','off')
        axes(handles.axes3)
        num=0.6074;
        den=[1.5680e-005 0.0038 0.0293];
        sys=tf(num,den);
        rlocus(sys)
        title('Velocity System')
        set(handles.invisible_static_pano,'String','Velocity
System')
        msgbox('You have selected P controller for velocity
control.The root locus is plotted in the top right axes for positive P.')
        set(H,'Enable','on')
        set(handles.figure1,'Pointer','arrow')
        % Waitbar disappears
        delete(hw);
    case 2
        % Waitbar appears
        wStr = sprintf('%s\n%s',...
            'Loading...');
        hw = waitbar(0.5,wStr);
        set(handles.figure1,'Pointer','watch')
        H=findobj('Style','PushButton','Enable','on');
        set(H,'Enable','off')
        axes(handles.axes4)
        P=str2num(get(handles.edit4,'String'));
        num=0.6074;
        den=[1.5680e-005 0.0038 0.0293+P*0.6074 0];
        sys=tf(num,den);
        rlocus(sys)
        title('Velocity System')
        set(handles.invisible_static_kato,'String','Velocity
System')
        set(handles.PI_title_static_text,'String','Root Locus for
positive I and P =')
        set(handles.P_static_text,'String',num2str(P))
        msgbox('You have selected PI controller for velocity
control.The root locus is plotted in the bottom right axes for positive I
and constant P specified in Controller Gains.')
        set(H,'Enable','on')
        set(handles.figure1,'Pointer','arrow')
        % Waitbar disappears
        delete(hw);
    case 3
        msgbox('No root locus plot available for PID controller.')
    end
case 'control_pos'
    switch get(handles.popupmenu1,'Value')
    case 1
        % Waitbar appears

```



```

wStr = sprintf('%s\n%s',...
'Loading...');
hw = waitbar(0.5,wStr);
set(handles.figure1,'Pointer','watch')
H=findobj('Style','PushButton','Enable','on');
set(H,'Enable','off')
axes(handles.axes3)
num=0.0759275;
den=[1.5680e-005 5.29e-006*1.4+0.0605^2 0];
sys=tf(num,den);
rlocus(sys)
title('Position System')
set(handles.invisible_static_pano,'String','Position
System')

msgbox('You have selected P controller for position
control.The root locus is plotted in the top right axes for positive P.')
set(handles.figure1,'Pointer','arrow')
set(H,'Enable','on')
% Waitbar disappears
delete(hw);
case 2
% Waitbar appears
wStr = sprintf('%s\n%s',...
'Loading...');
hw = waitbar(0.5,wStr);
set(handles.figure1,'Pointer','watch')
H=findobj('Style','PushButton','Enable','on');
set(H,'Enable','off')
axes(handles.axes4)
P=str2num(get(handles.edit4,'String'));
num=0.0759275;
den=[1.5680e-005 5.29e-006*1.4+0.0605^2 0];
sys1=tf(num,den);
s=tf([1 0],[1]);
sys2=s*(1+P*sys1);
sys3=sys1/sys2;
rlocus(sys3)
title('Position System')
set(handles.invisible_static_kato,'String','Position
System')

set(handles.PI_title_static_text,'String','Root Locus for
positive I and P =')
set(handles.P_static_text,'String',num2str(P))
msgbox('You have selected PI controller for position
control.The root locus is plotted in the bottom right axes for positive I
and constant P specified in Controller Gains.')
set(H,'Enable','on')
set(handles.figure1,'Pointer','arrow')
% Waitbar disappears
delete(hw);
case 3
msgbox('No root locus plot available for PID controller.')
end
end

guidata(hObject, handles);

```

```

% -----
function toolsMenu_Callback(hObject, eventdata, handles)
% hObject    handle to toolsMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% -----
function ss_error_calculator_Callback(hObject, eventdata, handles)
% hObject    handle to ss_error_calculator (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A=0.6074/0.0293;
prompt={'Enter desired steady state error (for unit input)'};
name='Steady state error calculator';
numlines=1;
defaultanswer='';
options='on';
while 1==1
    answer=inputdlg(prompt,name,numlines,defaultanswer,options);
    e=str2num(answer{1});
    if isempty(e)
        %msgbox('Input must be a number.','Warning !')
    else
        P1=((1/e)-1)/A;
        P=num2str(P1);
        msgbox(['To achieve the desired steady state error P must be equal
to ' P], 'Desired steady state error')
        break
    end
end

% -----
function zero_calculator_Callback(hObject, eventdata, handles)
% hObject    handle to zero_calculator (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
prompt={'Enter I', 'Enter P'};
name='Closed-loop zero calculator';
numlines=1;
defaultanswer={'',''};
options='on';
while true
    answer=inputdlg(prompt,name,numlines,defaultanswer,options);
    if ( isempty(str2num(answer{1})) || isempty(str2num(answer{2})) )
    else
        a=-str2num(answer{1})/str2num(answer{2});
        zero=num2str(a);
        msgbox(['For the specific I and P the closed-loop zero position is
at s=' zero], 'Closed-loop zero position')
        break
    end
end

% -----
function clearplotMenu_Callback(hObject, eventdata, handles)

```



```

% hObject    handle to clearplotMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% -----
function top_left_Callback(hObject, eventdata, handles)
% hObject    handle to top_left (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla(handles.axes1, 'reset')

% -----
function top_right_Callback(hObject, eventdata, handles)
% hObject    handle to top_right (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla(handles.axes3, 'reset')

% -----
function bottom_left_Callback(hObject, eventdata, handles)
% hObject    handle to bottom_left (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla(handles.axes2, 'reset')

% -----
function bottom_right_Callback(hObject, eventdata, handles)
% hObject    handle to bottom_right (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla(handles.axes4, 'reset')
set(handles.PI_title_static_text, 'String', 'Root Locus for positive I')
set(handles.P_static_text, 'String', '')

% -----
function all_Callback(hObject, eventdata, handles)
% hObject    handle to all (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla(handles.axes1, 'reset')
cla(handles.axes2, 'reset')
cla(handles.axes3, 'reset')
cla(handles.axes4, 'reset')
set(handles.PI_title_static_text, 'String', 'Root Locus for positive I')
set(handles.P_static_text, 'String', '')

% -----
function save_plot_menu_Callback(hObject, eventdata, handles)
% hObject    handle to save_plot_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% -----
function save_top_left_Callback(hObject, eventdata, handles)

```

```

% hObject    handle to save_top_left (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Ask for path and filename where to save
[filename, pathname] = uiputfile({'*.fig','Figure (*.fig)'}, ...
'Save picture as','default');

% If user cancels save command, nothing happens
if isequal(filename,0) || isequal(pathname,0)
    return
end

% Isolate axes1 in new figure with handle fh
fh = isolate_axes(handles.axes1, true);

% Change properties before save
set(fh,'Name',filename)
set(fh,'toolbar','figure')
set(fh,'CloseRequestFcn','delete(gcf)')

% Change dimensions of fh
set(fh,'Position',[77.2000    22.8462   153.6000    39.0769])

% Get handle of axes
axes_handle=findobj(fh,'YAxisLocation','left','Tag','');

% Set axes position
set(axes_handle,'Position',[0.0711 0.0933 0.8813 0.8300])

% Save figure with axes in path specified above
saveas(fh,[pathname filename])

% Close the figure and notify
close(fh)
msgbox(['Plot has been saved in directory "' pathname '"'], 'File saved')

% -----
function save_top_right_Callback(hObject, eventdata, handles)
% hObject    handle to save_top_right (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Ask for path and filename where to save
[filename, pathname] = uiputfile({'*.fig','Figure (*.fig)'}, ...
'Save picture as','default');

% If user cancels save command, nothing happens
if isequal(filename,0) || isequal(pathname,0)
    return
end

% Create new figure and set its properties
newFig=figure;
set(newFig,'MenuBar','none')

```



```

set(newFig,'toolbar','figure')

% Copy axes3 to newFig
axesObject2=copyobj(handles.axes3,newFig);
set(axesObject2,'Position',[0.0711 0.0933 0.8813 0.8300])
set(newFig,'Name',[get(handles.invisible_static_pano,'String') ' - Root
Locus for positive P'])

% Save figure with axes in path specified above
saveas(newFig,[pathname filename])

% Close the figure and notify
close(newFig)
msgbox(['Plot has been saved in directory "' pathname '"'], 'File saved')

% -----
function save_bottom_left_Callback(hObject, eventdata, handles)
% hObject      handle to save_bottom_left (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Ask for path and filename where to save
[filename, pathname] = uiputfile({'*.fig','Figure (*.fig)'}, ...
'Save picture as','default');

% If user cancels save command, nothing happens
if isequal(filename,0) || isequal(pathname,0)
    return
end

% Isolate axes2 in new figure with handle fh
fh = isolate_axes(handles.axes2, true);

% Change properties before save
set(fh,'Name',filename)
set(fh,'toolbar','figure')
set(fh,'CloseRequestFcn','delete(gcf)')

% Change dimensions of fh
set(fh,'Position',[77.2000 22.8462 153.6000 39.0769])

% Get handle of axes
axes_handle=findobj(fh,'YAxisLocation','left','Tag','');

% Set axes position
set(axes_handle,'Position',[0.0711 0.0933 0.8813 0.8300])

% Save figure with axes in path specified above
saveas(fh,[pathname filename])

% Close the figure and notify
close(fh)
msgbox(['Plot has been saved in directory "' pathname '"'], 'File saved')

```

```

% -----
function save_bottom_right_Callback(hObject, eventdata, handles)
% hObject    handle to save_bottom_right (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Ask for path and filename where to save
[filename, pathname] = uiputfile({'*.fig','Figure (*.fig)'}, ...
'Save picture as','default');

% If user cancels save command, nothing happens
if isequal(filename,0) || isequal(pathname,0)
    return
end

% Create new figure and set its properties
newFig=figure;
set(newFig,'MenuBar','none')
set(newFig,'toolbar','figure')

% Copy axes4 to newFig
axesObject2=copyobj(handles.axes4,newFig);
set(axesObject2,'Position',[0.0711 0.0933 0.8813 0.8300])
set(newFig,'Name',[get(handles.invisible_static_kato,'String') ' - Root
Locus for positive I and P= ' get(handles.P_static_text,'String')])

% Save figure with axes in path specified above
saveas(newFig,[pathname filename])

% Close the figure and notify
close(newFig)
msgbox(['Plot has been saved in directory "' pathname '"'], 'File saved')

% ISOLATE_AXES Isolate the specified axes in a figure on their own
%
% Examples:
%   fh = isolate_axes(ah)
%   fh = isolate_axes(ah, vis)
%
% This function will create a new figure containing the axes specified, and
% also their associated legends and colorbars. The axes specified must all
% be in the same figure, but they will generally only be a subset of the
% axes in the figure.
%
% IN:
%   ah - An array of axes handles, which must come from the same figure.
%   vis - A boolean indicating whether the new figure should be visible.
%         Default: false.
%
% OUT:
%   fh - The handle of the created figure.

% Copyright (C) Oliver Woodford 2011

% Thank you to Rosella Blatt for reporting a bug to do with axes in GUIs

```



```

function fh = isolate_axes(ah, vis)
% Make sure we have an array of handles
if ~all(ishandle(ah))
    error('ah must be an array of handles');
end
% Check that the handles are all for axes, and are all in the same figure
fh = ancestor(ah(1), 'figure');
nAx = numel(ah);
for a = 1:nAx
    if ~strcmp(get(ah(a), 'Type'), 'axes')
        error('All handles must be axes handles.');
```

```

    end
    if ~isequal(ancestor(ah(a), 'figure'), fh)
        error('Axes must all come from the same figure.');
```

```

    end
end
% Tag the axes so we can find them in the copy
old_tag = get(ah, 'Tag');
if nAx == 1
    old_tag = {old_tag};
end
set(ah, 'Tag', 'ObjectToCopy');
```

```

% Create a new figure exactly the same as the old one
fh = copyfig(fh); %copyobj(fh, 0);
if nargin < 2 || ~vis
    set(fh, 'Visible', 'off');
```

```

end
% Reset the axes tags
for a = 1:nAx
    set(ah(a), 'Tag', old_tag{a});
end
% Find the objects to save
ah = findall(fh, 'Tag', 'ObjectToCopy');
```

```

if numel(ah) ~= nAx
    close(fh);
    error('Incorrect number of axes found.');
```

```

end
% Set the axes tags to what they should be
for a = 1:nAx
    set(ah(a), 'Tag', old_tag{a});
end
% Keep any legends and colorbars which overlap the subplots
lh = findall(fh, 'Type', 'axes', '-and', {'Tag', 'legend', '-or', 'Tag',
'Colorbar'});
nLeg = numel(lh);
if nLeg > 0
    ax_pos = get(ah, 'OuterPosition');
    if nAx > 1
        ax_pos = cell2mat(ax_pos(:));
    end
    ax_pos(:,3:4) = ax_pos(:,3:4) + ax_pos(:,1:2);
    leg_pos = get(lh, 'OuterPosition');
```

```

    if nLeg > 1;
        leg_pos = cell2mat(leg_pos);
    end
    leg_pos(:,3:4) = leg_pos(:,3:4) + leg_pos(:,1:2);
    for a = 1:nAx
        % Overlap test

```

```

        ah = [ah; lh(leg_pos(:,1) < ax_pos(a,3) & leg_pos(:,2) <
ax_pos(a,4) &...
        leg_pos(:,3) > ax_pos(a,1) & leg_pos(:,4) >
ax_pos(a,2))];
    end
end
% Get all the objects in the figure
axs = findall(fh);
% Delete everything except for the input axes and associated items
delete(axs(~ismember(axs, [ah; allchildren(ah); allancestors(ah)])));
return

function ah = allchildren(ah)
ah = allchild(ah);
if iscell(ah)
    ah = cell2mat(ah);
end
ah = ah(:);
return

function ph = allancestors(ah)
ph = [];
for a = 1:numel(ah)
    h = get(ah(a), 'parent');
    while h ~= 0
        ph = [ph; h];
        h = get(h, 'parent');
    end
end
return

function fh = copyfig(fh)
% Is there a legend?
if isempty(findobj(fh, 'Type', 'axes', 'Tag', 'legend'))
    % Safe to copy using copyobj
    fh = copyobj(fh, 0);
else
    % copyobj will change the figure, so save and then load it instead
    tmp_nam = [tempname '.fig'];
    hgsave(fh, tmp_nam);
    fh = hgload(tmp_nam);
    delete(tmp_nam);
end
return
% END OF ISOLATE_AXES

% -----
function quit_menu_Callback(hObject, eventdata, handles)
% hObject    handle to quit_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% -----
function exit_application_Callback(hObject, eventdata, handles)
% hObject    handle to exit_application (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
closeGUI

```


Αρχιερ. Κωνσταντ. Γ.	
ΣΥΓΓΡΑΦΕΑΣ	
"Σχεδιασμός & ανάπτυξη	
ΤΙΤΛΟΣ	
αλφριθμικών ελέγχων του..."	
ΛΗΞΗ	ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΔΑΝΕΙΖΟΜΕΝΟΥ
9/3/12	

